Université Paris-Diderot
École doctorale Paris Centre

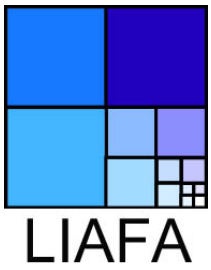
Thèse de Doctorat
Spécialité Informatique

Présentée par
Luc Dartois

# Méthodes algébriques pour la théorie des automates

Thèse soutenue devant un jury composé de :

M. Olivier Carton, Professeur, Paris-Diderot (directeur de thèse),
M. Jean-Éric Pin, Directeur de Recherche, CNRS (directeur de thèse),
M. Pascal Weil, Directeur de Recherche, CNRS (rapporteur),
M. Jean-Marc Talbot, Professeur, Université Aix-Marseille (rapporteur),
M. Emmanuel Filiot, Research Associate, Université Libre de Bruxelles-FNRS (examinateur),
M. Dominique Perrin, Professeur, Université Marne-la-Vallée (examinateur),
M. Manfred Kufleitner, Research Assistant, Universität Stuttgart (examinateur).


# Abstract

This thesis, written in french, focuses of the study of formal languages. More precisely, we are interested in the links between the different models of representation of rational languages that are automata, **MSO** logic and finite monoids, which originates from works by Julius Richard Büchi and Marcel-Paul Schützenberger. The contribution is divided in two axis, with an introductory chapter presenting these models.

The first contribution concerns logic on finite words. We are interested in the definability problem of a fragment of logic, which consists in deciding whether a given regular language can be defined by a formula of the said fragment. We study how the decidability of this question for a fragment of **MSO** can be transferred to the same fragment when we enrich its signature, in our case by predicates handling the modular information of the positions.
We first present a positive result for a specific fragment, the first-order logic with two variables, using automata and algebraic methods. These results can also be found in [DP13].

We then try to answer this question for a generic fragment. Through algebraic methods such as semidirect product and the theory of varieties of finite categories, we get partial but powerful transfer results, generalizing known results as well as getting new ones. Though unpublished yet, these results can be found in [DP14].

The second contribution concerns the deterministic two-way transducers, which is an extension of automata defining functions over words. Engelfriet and Hoogeboom proved in 2001 that these transducers have the same expressive power as **MSO** linear graphs transformations, defined by Courcelle. Our aim was to give a machine equivalent to the restriction of this logic to the first order.

In order to do this, we first propound the construction of the transitions monoid of two-way machines as a canonical algebraic object. This allows us to define restrictions on the two-way transducers, and we define the notion of aperiodic two-way transducers, as a natural candidate for equivalence. The contribution does not reach the equivalence with **FO** linear graphs translations, but we focus on proving that functions realized by aperiodic two-transducers are closed under composition, following a proof for generic two-way transducers by Chytil and Jákl. We finally prove that this class is stable by composition. This work is yet to be published.

# Remerciements

Avant d'entrer dans le vif du sujet et d'aborder le contenu scientifique de ma thèse, je tiens à remercier un certain nombre de personnes qui étaient présentes pendant ces quelques années, et sans qui ce manuscrit n'aurait probablement jamais abouti.

Je remercie tout d'abord mes parents et ma famille, qui ont toujours eu confiance dans mes choix, sans toujours comprendre ce que je faisais.

Merci au travail de soutien de fond réalisé par mes colocataires successifs : Damien "trop fat !" et Max "Boson" qui ont fait le gros du travail à la rue du Tage, puis Cédric "le dernier samaritain", Kader "scoudidou" et Karim "qui sait maintenant ce qu'est un automate" à La Défense. Merci à mes amis Clément, Pierrick, Charlotte, Célia, Pierre et Alex et tous les autres.

J'ai aussi beaucoup de personnes à remercier dans le monde de la recherche où j'ai trouvé des gens ouverts et accessibles. Tout d'abord merci à Jean-Éric et Olivier pour m'avoir guidé. Merci pour vos conseils et votre encadrement. Merci à FREC Junior : Laure toujours de bon conseil, Denis aux milles énigmes, Nathanaël le semi-polonais, et Sam et Lorijn et Arthur. Mention spéciale à Charles qui m'a le plus subi (et réciproquement).

Je remercie également le LIAFA dans son ensemble pour son accueil, ce fut une seconde maison pour moi pendant ces années, notamment grâce à l'équipe administrative Noëlle, Nathalie, Laifa et Houy. Grâce aussi à mes cobureaux Adeline, Antoine, Cezara, Élie, Irène, Jehanne, Jeremy et notre regretté mur des citations, et à tous les autres thésards, trop nombreux pour être nommés.

Finalement, je remercie aussi particulièrement Jean-Marc Talbot et Pascal Weil pour avoir accepté de relire ma thèse, ainsi qu'Emmanuel Filiot, Manfred Kufleitner et Dominique Perrin, je suis honoré que vous fassiez partie de mon jury de thèse.

Sur ce, bonne lecture et comme disait John McClane : "Yippee-ki-yay !"

# Table des matières

# Introduction

Chapitre 1

## Motivations

Entre les vérités absolues des mathématiques les plus fondamentales et les applications industrielles pratiques de l'informatique, il existe de nombreux mondes. À chacun de ces niveaux, des contributeurs récoltent les fruits du niveau plus théorique et les enrichissent pour les diffuser à différents niveaux plus pratiques.

Située quelque part dans cet arbre à la frontière entre les mathématiques et l'informatique, la théorie des langages formels s'appuie notamment sur des fondements algébriques pour proposer des réponses génériques à des questions de l'informatique telle que la vérification de programmes.

D'un côté l'algèbre, et plus particulièrement la théorie des ensembles, a pour but de définir une théorie consistante et cohérente à partir d'un ensemble minimal de vérités admises. Cela permet ainsi de développer un ensemble de propriétés qui sont alors conséquences irréfutables des axiomes admis. Après une interprétation méticuleusement vérifiée, elles sont alors applicables à tout objet que l'on a réussi à modéliser à l'intérieur de cette théorie.

De l'autre côté, la vérification est la branche de l'informatique visant à modéliser de façon automatique des systèmes informatiques afin de pouvoir obtenir une preuve mathématique de leur correction. En effet, lorsque l'on veut prouver la véracité d'un programme informatique, une batterie de tests, si extensive qu'elle soit, ne peut pas être totalement exhaustive. Cela se révèle particulièrement important dans des systèmes critiques tels que l'aérospatial, le médical ou même des systèmes financiers. La vérification permet alors, sous réserve d'une modélisation mathématiquement correcte, de réduire le système considéré à un objet mathématique connu et maitrisé tel que, par exemple, les monoïdes, issus de l'algèbre. Les considérations sont alors multiples. Cette modélisation doit être suffisamment expressive afin de pouvoir décrire les problèmes considérés. Elle doit également, dans certains cas, pouvoir se faire dans des temps raisonnables dépendant des questions posées. Il est cependant parfois utile de sacrifier du temps de modélisation automatique afin d'obtenir un modèle plus efficace, capable de répondre plus rapidement à une potentielle infinité de questions.

Il apparaît alors utile de pouvoir caractériser précisément les limites de chacun des modèles possibles afin d'être capable de choisir, pour chaque problème, le meilleur com-

promis entre expressivité du modèle et rapidité des réponses. La théorie des langages formels se trouve à l'interstice entre ces deux domaines. Elle remonte à la fin des années 50 et début des années 60 avec les travaux de Julius Richard Büchi du côté de la théorie des automates et Marcel-Paul Schützenberger du côté algébrique. Elle permet, grâce à l'algèbre, de fournir à la vérification des modèles maîtrisés. Plus précisément, les études présentes dans ce manuscrit reposent sur l'équivalence de trois modèles servant à représenter la partie dite rationnelle des langages formels, qui est un ensemble de langages permettant, par exemple, de modéliser par exemple le comportement de systèmes ou programmes informatiques dans le temps.

Le premier modèle est les automates. Un automate est une machine à mémoire bornée. Il s'agit du modèle le plus proche des ordinateurs et du modèle sous lequel est le plus souvent donné un langage rationnel modélisant un programme informatique. Les exécutions potentielles du programme sont données sous forme de mots que l'on donne à l'automate, qui répond alors à la question de l'appartenance du mot au langage sous forme d'un booléen.

Le second modèle est la logique, et plus particulièrement dans le cas des langages rationnels la logique monadique du second ordre. Elle sert à décrire les propriétés que le programme informatique doit satisfaire. Une formule logique et sa sémantique correspondent alors à un automate, dans le sens où elle reconnait un langage, qui est l'ensemble des mots vérifiant la propriété qu'elle décrit. Des sous-classes de cette logique appelés fragments de logique permettent alors d'obtenir des algorithmes plus rapides pour la décision en contrepartie d'une expressivité réduite.

Enfin, le troisième modèle est algébrique et correspond aux monoïdes finis. Cette structure algébrique simple capture le pouvoir d'expressivité des langages rationnels, des automates ainsi que de la logique. Les morphismes permettent alors d'associer un mot à un élément d'un monoïde, permettant la séparation de l'ensemble des mots en différents langages. L'existence de sous-classes décidables de monoïdes équivalentes aux sous-classes de logique permet alors de décider de l'expressivité d'un fragment de logique.

## Contributions

Ce manuscrit présente des résultats de la théorie des langages formels. Nous nous y appliquons à étendre la riche littérature explicitant les liens profonds entre logique, automates et théorie des monoïdes. Plus précisément, ce manuscrit se divise en deux chapitres principaux abordant deux thèmes très différents, ainsi qu'un chapitre introductif présentant les fondements de la théorie des langages rationnels. Ce chapitre introductif ne vise pas à être exhaustif, mais plutôt à donner les éléments principaux des théories considérées ainsi qu'une idée du fonctionnement des outils proposés.

### Prédicats modulaires

Le premier thème abordé est l'enrichissement de fragments logiques par les prédicats modulaires. En effet, la complexité des algorithmes de décision est fortement déterminée par la structure des formules autorisées, ce qui est décidé par la nature du fragment.

Ainsi, enrichir la signature d'un fragment permet d'augmenter son expressivité avec un coût en complexité limité. Il devient alors intéressant de pouvoir caractériser précisément le pouvoir d'expression de cette logique enrichie.

Nous étudions dans le Chapitre 3 l'effet de l'ajout des prédicats modulaires à des fragments connus. Ceci est tout d'abord fait en particulier sur le fragment connu qu'est la logique du premier ordre à deux variables, grâce à des méthodes proches des automates et des monoïdes.

La seconde partie de ce chapitre tente d'apporter une réponse générale à ce problème de la décidabilité de fragments connus lorsque l'on y ajoute les prédicats modulaires. L'étude nous amène à étendre le modèle des monoïdes et à considérer les classes de catégories au sens mathématique. Si la réponse obtenue n'est pas complète ni automatique, nous prouvons tout de même la préservation de la décidabilité pour la plupart des fragments connus.

Ce chapitre est le résultat d'un travail commun avec Charles Paperman, également étudiant en thèse sous la direction conjointe de Jean-Éric Pin et Olivier Carton.

**Transducteurs bidirectionnels**

Deuxièmement, le Chapitre 4 aborde l'extension des automates que sont les transducteurs. Les transducteurs sont des automates qui, au lieu de donner une réponse binaire pour un mot donné en entrée, répondent un autre mot sur un alphabet potentiellement différent. Si ainsi les automates permettent de modéliser des programmes informatiques et leur réponse à une entrée donnée, les transducteurs modélisent les traitements de données tels que la conversion de données.

Nous étudierons plus particulièrement les transducteurs bidirectionnels, qui sont des transducteurs dont la tête de lecture peut se déplacer dans deux directions. Nous préférerons ce terme au terme classique mais peu connu de boustrophédon, qui suppose un parcours de l'intégralité de l'entrée de gauche à droite puis inversement. Ce modèle de calcul a beaucoup été étudié ces dernières années, avec l'obtention notamment d'un équivalent logique, par les transductions de graphe, et par des modèles plus pratiques tels que les Streaming string transducers.

Nous donnerons dans ce chapitre les définitions nécessaires à l'établissement de bases pour une théorie algébrique des automates bidirectionnels, et définissons les transducteurs bidirectionnels apériodiques, comme un appel du pied à des transductions de graphes du premier ordre. En particulier, nous prouvons que la classe que nous définissons est stable par composition, en reprenant les idées principales d'une preuve par Chytil et Jákl sur la stabilité par composition, prouvant par là-même la robustesse des transducteurs bidirectionnels apériodiques.

# Préliminaires

Chapitre 2

La théorie des automates et des langages formels repose sur l'étude parallèle de quatre modèles et représentations équivalentes. Dans ce chapitre nous donnons les outils standards et résultats classiques de la littérature nécessaires à l'appréhension de chacun de ces modèles, ainsi que les liens profonds les réunissant.

Les quatre modèles que présentons sont les langages rationnels, les automates, la logique et les monoïdes, issus de l'algèbre. Notre étude se placera entièrement sur les structures linéaires et finies que sont les mots finis, bien que ces modèles aient été étendus avec succès à d'autres structures telles que les mots infinis ou les arbres.

Cette introduction formelle ne pourra évidement qu'être partielle, et nous renvoyons le lecteur aux livres suivants pour une introduction plus complète pour chaque domaine considéré : Langages rationnels et automates [HU79, Sak03, Car08], Logique [Str94], Monoïdes et variétés [Pin86, Pin97].

## 2.1 Mots finis et langages rationnels

Dans cette section, nous définissons les mots finis et les langages rationnels. Les mots finis ont été utilisés depuis les débuts de l'informatique et les travaux d'Alan Turing afin de représenter une information séquentielle linéaire. Ils sont entre autres utilisés pour décrire des nombres ou le comportement d'un programme dans le temps.

Nous définissons un *alphabet* comme un ensemble fini dont les éléments sont appelés des *lettres*. La concaténation de lettres forme les mots finis et nous en donnons tout de suite une définition formelle.

**Définition 2.1: Mot fini**
Étant donné un alphabet $A$, un *mot fini* $u$ sur $A$ est formellement défini par un entier appelé la longueur de $u$, que l'on notera $|u|$ et une fonction $u : \{0, \dots, |u| - 1\} \to A$ associant à chaque position de $u$ une lettre de l'alphabet. Il existe un unique mot de longueur nulle appelé le *mot vide*, et que l'on notera $\epsilon$.

Pour des raisons de simplicité, on identifiera dans la suite le mot avec la fonction le définissant, et pour tout entier $i < |u|$, on notera $u_i = u(i)$ et l'on dira alors que la position $i$ de $u$ est *étiquetée* par $u_i$.

De plus, on définit *l'alphabet* de $u$ comme l'image de la fonction $\alpha : A^* \to \mathcal{P}(A)$ sur le mot $u$.

Enfin, remarquons que les positions d'un mot sont étiquetées à partir de l'entier 0, ceci étant un choix de convention usuel.

Alternativement, les mots se définissent de façon inductive sur leur longueur de la manière suivante :

- Il existe un unique mot de longueur nulle noté $\epsilon$.
- L'ensemble des mots $v$ de longueur $n+1$ s'obtient en concaténant une lettre $a$ de $A$ à la fin d'un mot $u$ de longueur $n$. On écrit alors $v = u \cdot a$ ou $v = ua$ par abus de notation et la lettre $a$ est alors l'étiquette de la position $n$.

Les deux formalismes sont évidement équivalents et l'on passe aisément de l'un à l'autre.

Par exemple, si l'on considère $A$ un alphabet à deux lettres, usuellement notées $a$ et $b$, on a ainsi $A = \{a, b\}$ et la fonction $u : \{0, 1, 2, 3\} \to A$ définie par $u_i = a$ si $i = 0$ ou $i = 3$ définit le mot $u = abba$. Dans la suite, nous utiliserons de préférence cette seconde notation à la notation par fonction.

Nous noterons usuellement un alphabet par $A$ et l'ensemble des mots sur $A$ par $A^*$. Cette notation correspond à l'étoile de Kleene que nous définissons ci-après, et correspond aux listes arbitrairement longues d'éléments de $A$. D'un point de vue algébrique l'ensemble $A^*$, équipé de la concaténation, forme le monoïde libre engendré par $A$. Ceci sera abordé plus longuement à la Section 2.4.

Les ensembles de mots de $A^*$ sont appelés *langages* sur $A$. Il est à noter que si l'on considère des mots dont la longueur est finie, les langages que l'on étudiera ne seront pas a priori finis. Nous nous intéresserons cependant dans ce manuscrit aux langages finiment descriptibles, dans un sens que l'on précise plus loin. Nous allons donc définir l'ensemble de ces langages, appelés langages rationnels. À cette fin nous définissons d'abord l'étoile de Kleene, qui est une opération sur les langages de $A^*$.

**Définition 2.2: Étoile de Kleene**
Étant donné un ensemble $L$ de mots sur un alphabet $A$, *l'étoile de Kleene* de $L$, notée $L^*$, est l'ensemble des mots s'écrivant comme la concaténation ordonnée arbitrairement longue de mots de $L$.

Ainsi, si $L^0 = \{\epsilon\}$ et pour tout entier strictement positif $n$,

$$L^n = \{u_0 \dots u_{n-1} | \text{ pour tout } i < n \ u_i \in L\}$$

est la concaténation de $n$ éléments de $L$, alors $L^* = \bigcup_{n \geqslant 0} L^n$.

**Remarque 2.1.** Cette notation en étoile est la même que pour l'ensemble des mots $A^*$. En effet, $L^*$ pourrait être vu comme l'ensemble des mots sur l'alphabet $L$ potentiellement infini. Ce n'est cependant pas le cas, car les mots de $L^*$ sont quand même vus comme

des mots sur l'alphabet $A$, et ainsi la position $i$ d'un mot de $L^*$ n'est pas étiquetée par le *ième* mot de $L$ mais bien par la *ième* lettre de $A$ écrite.

Nous sommes maintenant prêts à définir les langages rationnels, sujets de l'ensemble de notre étude.

**Définition 2.3: Langages rationnels**
Soit $A$ un alphabet fini. Pour toute paire de langages $L$ et $L'$ dans $A^*$, nous définissons les opérations rationnelles suivantes :

Union : $L + L' = \{u \in A^* \mid u \in L \text{ ou } u \in L'\}$,
Produit : $LL' = \{w \in A^* \mid \text{ il existe } u \in L \text{ et } v \in L' \text{ tels que } w = u \cdot v\}$,
Étoile : $L^* = \bigcup_{n \geqslant 0} L^n$.

Les *langages rationnels* (ou langages *réguliers*) sur $A$ sont donc le plus petit ensemble des sous-ensembles de $A$ contenant le langage vide $\emptyset$ et les singletons $\{\epsilon\}$ et $\{a\}$ pour tout $a$ dans $A$, et clos par les opérations rationnelles d'union, de produit et d'étoile.
Il est également important de noter que si $L$ est un langage rationnel, alors son complémentaire $L^c = \{u \in A^* \mid u \notin L\}$, également noté $A^* - L$, est un langage rationnel.

Tout langage rationnel peut ainsi être écrit comme combinaison des singletons $\{\epsilon\}$ et $\{a\}$ pour tout $a$ dans $A$. Ces combinaisons sont appelées expressions rationnelles et permettent d'exprimer la complexité de la description d'un langage. Formellement, les *expressions rationnelles* se définissent récursivement de la façon suivante :

- Le langage vide $\emptyset$ et le langage $\{\epsilon\}$ réduit au mot vide sont des expressions rationnelles,
- Pour toute lettre $a$ de $A$, $a$ est une expression rationnelle,
- Si $e$ et $e'$ sont des expressions rationnelles, alors $(ee')$, $(e + e')$ et $(e^*)$ sont des expressions rationnelles.

Nous oublierons les parenthèses lorsqu'il n'y a pas de confusion possible. Dans la suite de ce manuscrit, nous identifierons une expression rationnelle avec le langage qu'elle décrit.

**Exemple 2.1.** Voici quelques exemples d'expressions rationnelles et les langages associés, pour un alphabet $A = \{a, b\}$ :

- $(a + b)aa(b + \epsilon)$ est le langage fini $\{aaa, aaab, baa, baab\}$.
- $(a + b)^* = A^*$ est l'ensemble des mots finis.
- $A^*aaA^*$ est l'ensemble des mots où $aa$ apparaît comme facteur.
- $((a + b)(a + b))^* = (A^2)^*$ est l'ensemble des mots de longueur paire.

En appliquant certaines restrictions concernant la construction des expressions rationnelles, on peut ainsi obtenir des sous-classes de langages. Par exemple, si l'on s'interdit l'utilisation de l'étoile de Kleene, l'ensemble des langages que l'on pourra obtenir sera exactement l'ensemble des langages finis.

On définit maintenant la sous-classe des langages rationnels que l'on peut considérer comme la plus fondamentale. Il s'agit des langages sans-étoile, définis de la façon suivante :

**Définition 2.4: Langages sans-étoile**
On définit les *langages sans-étoile* de la façon inductive suivante :
- Le langage vide $\emptyset$ et le singleton $\{\epsilon\}$ sont sans-étoile,
- pour toute lettre $a$ de $A$, le singleton $\{a\}$ est sans-étoile,
- si $L$ et $L'$ sont des langages sans-étoile, alors $L \cup L'$, $LL'$ et $L^c$ sont sans-étoile.

**Exemple 2.2.** Voici quelques exemples de langages sans-étoile.
- Les langages finis, et donc leurs complémentaires les langages co-finis, sont sans-étoile.
- Le langage $A^* = \emptyset^c$ est sans-étoile,
- Le langage $(ab)^* = aA^* \cap A^*b \cap (A^*aaA^* \cup A^*bbA^*)^c$ est de façon inattendue sans-étoile.

Nous donnons également à titre d'exemple une autre sous-classe de langages, dont la définition ne repose plus uniquement sur des expressions rationnelles

**Exemple 2.3** (Polynômes non-ambigus)**.** Étant donné un alphabet $A$, un *monôme de degré $n$* est un langage de la forme $A_0^*a_1A_1^* \dots a_nA_n^*$, où pour tout $i \geqslant 0$, $A_i$ est un sous-ensemble de $A$ et pour $i > 0$, $a_i$ est une lettre de $A$.
Un monôme $L = A_0^*a_1A_1^* \dots a_nA_n^*$ sera dit *non-ambigu* si pour tout mot $u$ de $L$, il existe une unique factorisation $u = u_0a_1u_1 \dots a_nu_n$ telle que $\alpha(u_i) \subseteq A_i$.
Un *polynôme non-ambigu* est finalement une union finie et disjointe de monômes non-ambigus.

Plus généralement, les sous-classes de langages que nous considérons sont les *variétés de langages* que nous définissons maintenant.

**Définition 2.5: Variété de langages**
Une variété de langage $\mathcal{V}$ associe à chaque alphabet un ensemble de langages rationnels sur $A$ tels que :
- pour chaque alphabet $A$, $\mathcal{V}(A^*)$ est une algèbre booléenne, c'est-à-dire un ensemble de langages clos par union, intersection et complémentation,
- pour tout morphisme de monoïdes (voir Définition 2.15) $\varphi : A^* \to B^*$ et pour tout langage $L$ de $\mathcal{V}(B^*)$, $\varphi^{-1}(L)$ appartient à $\mathcal{V}(A^*)$,
- pour tout langage $L$ de $\mathcal{V}(A^*)$ et tout mot $u$ de $A^*$, les langages $u^{-1}L$ et $Lu^{-1}$ appartiennent à $\mathcal{V}(A^*)$, où $u^{-1}L = \{v \in A^* \mid uv \in L\}$ et $Lu^{-1}$ est défini symétriquement.

Nous venons de présenter le type de structure sur lequel les études de ce manuscrit reposent. Les sections suivantes présentent les modèles de calcul que nous utiliserons afin de les étudier, et les questions qu'ils soulèvent.

## 2.2 Automates

Un automate est une machine séquentielle permettant de traiter l'information linéaire d'un mot. Le résultat de ce calcul est booléen, l'automate acceptant ou rejetant un mot. L'ensemble des mots acceptés est alors un langage, que l'on peut comparer avec les langages rationnels définis à la section précédente. Les automates ont été développés en premier lieu par Stephen Cole Kleene dans l'article [Kle56]. Son but initial était de nature biologique et les automates servaient à modéliser les réseaux de neurones. Ils ont été par la suite généralisés sous l'impulsion d'un mouvement collectif aux États-Unis afin de modéliser un programme ou un ordinateur ayant une mémoire fixe, dans le sens où, contrairement à une machine de Turing, la quantité de mémoire disponible pour effectuer le calcul est indépendante de la taille de l'entrée.

Nous commencerons par définir formellement un automate, puis nous donnerons des explications relatives au fonctionnement de ces machines ainsi que les résultats fondamentaux les concernant.

**Définition 2.6: Automates**
Un *automate* $\mathcal{A}$ est donné par le quintuplet $(Q, A, \Delta, I, F)$ où :
- $Q$ est un ensemble fini appelé l'ensemble des états de $\mathcal{A}$,
- $A$ est un alphabet,
- $\Delta$ est un sous-ensemble de $Q \times A \times Q$ appelé la relation de transition de $\mathcal{A}$. Une transition $(p, a, q)$ de $\Delta$ sera notée $p \xrightarrow{a} q$,
- $I$ est un sous-ensemble de $Q$ et correspond à l'ensemble des états initiaux de $\mathcal{A}$,
- $F$ est un sous-ensemble de $Q$ et correspond à l'ensemble des états finaux de $\mathcal{A}$.

Cette définition technique ne permet cependant pas d'expliciter le fonctionnement d'un automate. On définit maintenant les parcours d'un automate sur un mot.

**Définition 2.7: Parcours d'un automate**
Soient un automate $\mathcal{A} = (Q, A, \Delta, I, F)$ et un mot $u = u_0 \dots u_n$ sur $A^*$.

Une suite d'états $(q_0, \dots, q_{n+1})$ est un *parcours* de $\mathcal{A}$ sur $u$ si pour tout $i \leqslant n$, le triplet $q_i \xrightarrow{u_i} q_{i+1}$ est une transition de $\Delta$.

Un parcours $(q_0, \dots, q_{n+1})$ sera un *calcul* de $\mathcal{A}$ sur $u$ si $q_0$ appartient à $I$, et sera un *parcours acceptant* si $q_0$ appartient à $I$ et $q_{n+1}$ appartient à $F$.

L'ensemble des mots ayant un parcours acceptant sur un automate $\mathcal{A}$ est appelé le *langage reconnu par* $\mathcal{A}$ et est noté $L(\mathcal{A})$.

**Exemple 2.4.** Considérons le langage $L = A^*baaA^*$, c'est-à-dire l'ensemble des mots contenant le facteur $baa$. L'automate $\mathcal{A}$ représenté par la Figure 2.1 reconnait le langage $L$. Cette représentation des automates par un graphe orienté et étiqueté est standard dans la littérature. Les sommets du graphes représentent chacun un état, et une transition $p \xrightarrow{a} q$ est représentée par une flèche du sommet $p$ vers le sommet $q$ étiquetée par $a$. Les

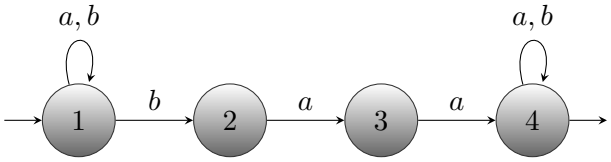



Figure 2.1: L'automate $\mathcal{A} = (Q, A, \Delta, I, F)$ reconnait $L$.

états initiaux sont marqués par une flèche entrante, et les états finaux par une flèche sortante.

Le calcul d'un mot $u = ababaab$ sur $\mathcal{A}$ se fait en parcourant le graphe représentant $\mathcal{A}$ et en suivant les transitions choisies au fur et à mesure selon la lettre courante. Dans ce cas, un calcul sera acceptant si, après avoir lu un préfixe dans l'état initial 1, on devine que l'on va lire le sous-mot $baa$ et l'on prend la transition $1 \xrightarrow{b} 2$. On termine alors le calcul dans l'état acceptant 4.

Cette nécessité de devoir deviner n'est pas commode à manipuler, et peut augmenter la complexité du calcul d'un mot. On définit alors les automates *déterministes* comme les automates dont la relation de transition est une fonction, c'est-à-dire les automates pour lesquels, étant donné un état et une lettre, il existe au plus un état tel que le triplet appartienne à la relation de transition. De plus, l'ensemble des états initiaux d'un automate déterministe est réduit à un singleton. Le résultat est qu'un automate déterministe ne possède, au plus, qu'un unique calcul pour chaque entrée. La fonction de transition d'un automate déterministe est alors notée $\cdot$ afin d'expliciter le fait que l'on considère un automate déterministe, ce qui est primordial dans certains problèmes, et l'on note $q \cdot a = q'$ les triplets.

Il peut également arriver que certains états de l'automate ne puissent apparaitre dans un parcours acceptant, soit car ils ne sont pas atteignables (état non accessibles), soit car ils ne peuvent mener à un état final (état non co-accessible). Un automate dont tous les états sont accessibles et co-accessible est appelée automate émondé .

Un des résultats les plus classiques de la théorie des automates affirme que pour tout automate non-déterministe, il existe un automate déterministe et émondé reconnaissant le même langage. En effet, il est possible de déterminiser un automate $\mathcal{A}$ en construisant un nouvel automate dont les états sont des ensembles d'états de $\mathcal{A}$ atteignables lors d'un calcul acceptant. Ce nouvel automate peut alors simuler en parallèle tous les parcours possibles de $\mathcal{A}$ de façon déterministe, et décider s'il existe un parcours acceptant.

**Exemple 2.5.** Le langage $L = A^*baaA^*$ considéré à l'exemple précédent est également reconnu par l'automate déterministe $\mathcal{B}$ donné en Figure 2.2.

Nous venons de voir qu'un langage donné peut être reconnu par différents automates. Afin de pouvoir associer un langage à un des automates le reconnaissant, nous définissons un ordre partiel sur les automates déterministes de la façon suivante. Soient

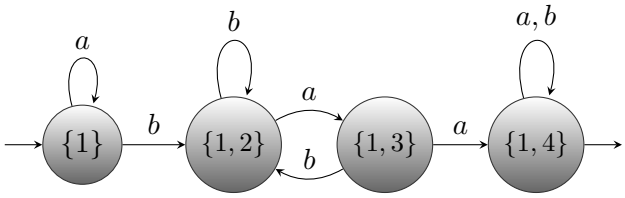



Figure 2.2: L'automate $\mathcal{B} = (\mathcal{P}(Q), A, \delta, \{1\}, F)$ reconnait $L$ de façon déterministe.

$\mathcal{A} = (Q, A, \cdot, i, F)$ et $\mathcal{B} = (P, A, ., j, G)$ deux automates déterministes sur le même alphabet. Alors $\mathcal{A} \leqslant \mathcal{B}$ s'il existe une fonction surjective $f : P \to Q$ telle que $f(j) = i$, $f^{-1}(F) = G$, et pour toute lettre $a$ de $A$ et tout état $p$ de $P$, $f(p.a) = f(p) \cdot a$. Il existe alors, pour tout langage $L$ reconnu par un automate, un *automate minimal* au sens de cet ordre partiel qui reconnait $L$. Il est en outre minimal au sens du nombre d'états parmi les automates reconnaissant $L$, à isomorphisme de graphe près. De plus, étant donné un automate $\mathcal{A}$, il existe des algorithmes effectifs permettant de construire l'automate minimal reconnaissant le même langage. Le premier de ces algorithmes est du à Hopcroft [Hop71] et repose sur le calcul des classes d'équivalence de la relation de Myhill-Nérode [Ner58].

**Remarque 2.2.** Étant donné un alphabet $A$, tout langage $L \subseteq A^*$ n'est pas reconnaissable par un automate. Par exemple, le langage $L = \{a^n b^n \mid n \geqslant 0\}$ n'est pas régulier. Pour se convaincre, il suffit de considérer un automate $\mathcal{A}$ reconnaissant $L$, et $n$ son nombre d'états. Alors $\mathcal{A}$ ne peut différencier $a^{n!}b^{n!}$ de $a^{(n+1)!}b^{n!}$ et acceptera ou refusera les deux. Or seul le premier mot appartient à $L$.

Avant de conclure notre présentation des automates, nous discutons des *langages reconnaissables* par automates et donnons le théorème fondateur de la théorie des automates dû à Kleene [Kle56], qui explicite leur lien avec les langages rationnels.

**Théorème 2.1** (Kleene)**.** *Un langage est reconnaissable par un automate si, et seulement si, il est rationnel.*

Dans la suite, nous pourrons confondre les langages reconnaissables et les langages rationnels ou réguliers, les notions décrivant les mêmes langages. La différence vient de leur définition originelle.

## 2.3 Logique sur les mots finis

Le second modèle de calcul que nous définissons et utiliserons dans ce manuscrit est la logique sur les mots finis. De façon similaire aux automates, les formules logiques seront utilisées afin de définir des langages.

Les formules logiques sont construites à partir d'atomes que sont les constantes ou des prédicats, des opérateurs booléens et des quantifications. Les formules comportent des variables. Les variables quantifieront dans le cas de la logique sur les mots finis des positions sur les mots, ou des ensembles de positions dans le cas de variables du second ordre. Nous donnons maintenant une définition inductive formelle des formules de la logique **MSO**[<]. **MSO** signifie logique monadique du second ordre. La sémantique de ces constructeurs est donnée plus loin.

**Définition 2.8: Logique monadique du second ordre**
Les formules de **MSO**[<] sur un alphabet $A$ se définissent de la façon suivante :
- Les constantes Vrai $\top$ et Faux $\bot$ sont des formules.
- Pour toute lettre $a$ de $A$ et toute variable du premier ordre $x$, le prédicat de lettre $\mathbf{a}(x)$ est une formule.
- Pour toute paire de variables du premier ordre $x$ et $y$, le prédicat binaire $x < y$ est une formule.
- Pour toute variable du premier ordre $x$ et toute variable du second ordre $X$, le prédicat d'appartenance $x \in X$ est une formule.
- Pour toute paire de formule $\varphi$ et $\psi$, la conjonction $\varphi \wedge \psi$, la disjonction $\varphi \vee \psi$ et la négation $\neg\varphi$ sont des formules.
- Pour toute formule $\varphi$ et toute variable du premier ordre $x$, les quantifications existentielle $\exists x\varphi$ et universelle $\forall x\varphi$ sont des formules.
- Pour toute formule $\varphi$ et toute variable du second ordre $X$, les quantifications existentielle $\exists X\varphi$ et universelle $\forall X\varphi$ du second ordre sont des formules.

Nous pouvons maintenant construire les formules de **MSO**[<]. Ainsi $\exists x \forall y (x < y) \vee (\mathbf{a}(x) \wedge \mathbf{b}(y))$ est une formule. Cependant, comme pour la définition de parcours d'un automate, nous devons maintenant définir le comportement d'une formule sur un mot d'entrée. Il s'agit de la sémantique de notre logique, ou comment interpréter chaque constructeur.

**Définition 2.9: Sémantique de la logique MSO**
Une formule s'interprète sur un mot non vide, c'est-à-dire un ensemble de positions étiquetées par des lettres, équipé de ce que nous appellerons un *environnement*, qui est composé d'un ensemble de paire $(x, i)$ où $x$ est une variable du premier ordre et $i$ une position du mot d'entrée et de paires $(X, I)$ où $X$ est une variable du second ordre et $I$ un ensemble de positions. On dira alors que $x$ (resp. $X$) quantifie $i$ (resp. $I$). Le résultat de l'évaluation d'une formule sur un mot et un environnement est alors un booléen, de façon similaire à la réponse d'un automate sur un mot.

Dans la suite, nous considérerons que nous évaluons une formule $\varphi$ sur un mot non vide $u = u_0 \dots u_n$ et un environnement $E$.
- les formules $\top$ et $\bot$ sont respectivement vraie et fausse.
- $\mathbf{a}(x)$ est vraie si, et seulement si, $x$ quantifie un entier $i$ tel que $u_i = a$.
- $x < y$ est vraie ssi $x$ et $y$ quantifient des positions $i$ et $j$ telles que $i < j$.

- $x \in X$ est vraie ssi la position quantifiée par $x$ appartient à l'ensemble quantifié par $X$.
- $\varphi \wedge \psi$ est vraie ssi $\varphi$ et $\psi$ sont toutes deux vraies sur $(u, E)$.
- $\varphi \vee \psi$ est vraie ssi au moins l'une des formules $\varphi$ ou $\psi$ est vraie sur $(u, E)$.
- $\neg\varphi$ est vraie ssi $\varphi$ est fausse sur $(u, E)$.
- $\exists x\varphi$ est vraie s'il existe une position $i$ telle que $\varphi$ est vraie sur $(u, E \cup \{(x, i)\})$. Il est à noter que si $x$ quantifiait déjà une position $j$ dans $E$, alors $(x, j)$ est supprimé de $E$ avant d'ajouter $(x, i)$.
- $\forall x\varphi$ est vraie ssi pour toute position $i$ du mot d'entrée, $\varphi$ est vraie sur $(u, E \cup \{(x, i)\})$. Il est à noter que si $x$ quantifiait déjà une position $j$ dans $E$, alors $(x, j)$ est supprimé de $E$ avant d'ajouter $(x, i)$.
- $\exists X\varphi$ est vraie s'il existe un ensemble de positions $I \subseteq [0, n-1]$ telle que $\varphi$ est vraie sur $(u, E \cup \{(X, I)\})$. Il est à noter que si $X$ quantifiait déjà un ensemble $J$ dans $E$, alors $(X, J)$ est supprimé de $E$ avant d'ajouter $(X, I)$.
- $\forall X\varphi$ est vraie ssi pour tout ensemble de positions $I$, $\varphi$ est vraie sur $(u, E \cup \{(X, I)\})$. Il est à noter que si $X$ quantifiait déjà un ensemble $J$ dans $E$, alors $(X, J)$ est supprimé de $E$ avant d'ajouter $(X, I)$.

Une formule $\varphi(X_1, \ldots, X_k, x_1, \ldots, x_l)$ est alors satisfaite par un couple $(u, E)$, noté $(u, E) \models \varphi$, si $E$ contient des valeurs pour chaque variable libre $X_i$, $x_j$ et si le résultat de l'évaluation de ce couple sur $\varphi$ aboutit au booléen vrai.
Un mot $u$ satisfait une formule close $\varphi$, c'est-à-dire sans quantificateurs, si on a $(u, \emptyset) \models \varphi$, ce qui sera alors noté $u \models \varphi$. On note également $L(\varphi)$ le langage formé de l'ensemble des mots satisfaisant la formule $\varphi$.
De plus, étant donné que les formules logiques ne s'interprètent pas sur le mot vide, un langage $L$ est *défini* par une formule $\varphi$ si on a $L - \{\epsilon\} = L(\varphi)$.

**Exemple 2.6.** Soit la formule $\varphi = \exists x(\mathbf{a}(x) \wedge \forall y\big((x < y) \vee \mathbf{b}(y)\big)$. En utilisant la sémantique définie précédemment, nous pouvons décrire les mots satisfaisant $\varphi$. Un mot $u$ satisfera donc $\varphi$ s'il existe une position $i$ de $u$ qui soit étiquetée par un $a$, et telle que pour toute position $j$ de $u$, l'on ait soit $i < j$, soit $u_j = b$.
Cela se traduit par l'existence sur $u$, d'une lettre $a$ précédée uniquement de lettres $b$. Les mots satisfaisant $\varphi$ sont donc ceux qui sont décrit par l'expression rationnelle $b^*aA^*$. On peut également noter que dans le cas d'un alphabet à deux lettres $a$ et $b$, l'expression ci-dessus est équivalente à $A^*aA^*$.

**Remarque 2.3.** Ayant maintenant la sémantique associée à nos formules, on peut remarquer que notre construction inductive n'est pas minimale, dans le sens ou certains opérateurs s'expriment en fonction d'autres. Ainsi les formules $\varphi \vee \psi$ et $\neg\varphi \wedge \neg\varphi$ seront satisfaites par les mêmes modèles, tout comme les formules $\exists x\varphi$ et $\neg\forall x\neg\varphi$.

Avant d'approfondir notre description de la logique avec les fragments, nous donnons un premier résultat de Büchi [Büc60] caractérisant les langages définissables par la logique $\mathbf{MSO}[<]$.

**Théorème 2.2** (Büchi)**.** *Un langage est définissable par une formule de* **MSO**[<] *si, et seulement si, il est reconnaissable par un automate.*

Les formules logiques telles que nous les avons définies ont donc la même puissance de calcul que les automates, et définissent les langages rationnels.

De la même façon que nous avons défini la sous-classe des langages sans-étoile (voir Définition 2.4), nous désirons maintenant pouvoir appliquer des restrictions sur la construction de nos formules afin de définir des sous-classes de formules, et par là-même des sous-classes de langages définissables.

Une sous-classe de formules sera appelée un *fragment logique* si elle forme un ensemble de formules clos par conjonction, disjonction et substitution atomique. La *substitution atomique* est une opération sur les formules consistant à remplacer dans une formule une sous-formule sans quantificateurs par une autre sous-formule également sans quantificateurs.

Lorsque l'on considérera un fragment de logique, on considérera alors également l'ensemble des prédicats que l'on s'autorise. Cet ensemble, comprenant les prédicats de lettres ainsi que le prédicat d'appartenance si le fragment comporte des variables du second ordre, s'appelle la *signature* du fragment, et est notée entre crochets à la suite de la dénomination du fragment.

Un langage sera *définissable* dans une classe de logique s'il existe une formule de cette classe le définissant. Étant donné un fragment $\mathcal{F}[\sigma]$ et un alphabet $A$, l'ensemble des langages sur $A^*$ définissables par une formule de $\mathcal{F}[\sigma]$ sera noté $\mathcal{F}[\sigma](A^*)$. Le problème de la *définissabilité* pour un fragment est de pouvoir décider, étant donné un langage défini par un automate, s'il existe une formule du fragment le définissant. Cependant, s'il existe, pour un langage donné, un automate minimal, il n'existe pas de formule canonique pour un langage. Ce problème ne peut alors pas être décidé par une étude logique. Il est souvent résolu par des méthodes algébriques qui seront données dans la prochaine section.

Nous définissons maintenant le fragment de la logique du premier ordre **FO**[<].

**Définition 2.10: Logique du premier ordre**
La restriction de la logique **MSO**[<] aux formules ne comportant pas de variable du second ordre s'appelle la *logique du premier ordre* et est notée **FO**[<].

L'ensemble des langages définissables par les formules de **FO**[<] correspond exactement aux langages rationnels sans-étoile définis à la Définition 2.4. Ce résultat est dû à McNaughton et Papert [MP71].

Nous mentionnons également d'autres fragments qui seront étudiés dans la suite de ce manuscrit.

**Exemple 2.7** (Logique du premier ordre à deux variables)**.** Si l'on restreint la logique du premier ordre aux formules n'utilisant que deux symboles de variables différents,

on obtient la logique $\mathbf{FO}^2[<]$, étudiée notamment dans [TW99]. Ce fragment semble à première vue extrêmement restrictif car deux variables ne semblent pouvoir définir qu'un nombre fini de formules. Ces variables sont cependant réutilisables, chaque utilisation de variable portant sur la dernière quantification existante du symbole associé.

Par exemple, le langage $L = A^*aA^*bA^*aA^*$ est défini par la formule

$$\varphi = \exists x\ \mathbf{a}(x) \wedge \Big(\exists y\ \mathbf{b}(y) \wedge (x < y) \wedge \big(\exists x\ \mathbf{a}(x) \wedge (y < x)\big)\Big)$$

L'intérêt de ce fragment est dû à l'égalité $\mathbf{FO}^3[<] = \mathbf{FO}[<]$. En effet, avec seulement trois variables nous obtenons l'expressivité du premier ordre entier. Nous renvoyons également le lecteur à l'étude [DGK08] de Diekert, Gastin et Kufleitner pour une description complète de ce fragment. Ce fragment sera enfin plus longuement étudié dans la première partie du Chapitre 3.

**Exemple 2.8** (Le fragment $\Sigma_1$)**.** Le fragment $\Sigma_1$ est l'ensemble des formules composées d'un bloc de quantificateurs existentiels, suivi d'une formule ne comportant pas de quantificateur.

Par exemple, le langage $L = A^*aA^*bA^*aA^*$ est défini par la formule

$$\varphi = \exists x \exists y \exists z \mathbf{a}(x) \wedge \mathbf{b}(y) \wedge \mathbf{a}(z) \wedge (x < y) \wedge (y < z)$$

Dans la définition initiale de $\mathbf{MSO}[<]$, nous avons restreint les prédicats aux prédicats de lettres $\mathbf{a}(x)$, au prédicat d'ordre $x < y$ et au prédicat d'appartenance $x \in X$.
On peut songer à de nouveaux prédicats exprimant d'autres propriétés. On pourrait par exemple vouloir exprimer le successeur $S(x, y)$ vrai si la position quantifiée par $x$ précède la position quantifiée par $y$. Cependant, puisque les langages définissables par $\mathbf{MSO}[<]$ sont exactement les langages rationnels, il n'existe pas de prédicat régulier à ajouter à la signature $[<]$ permettant d'en augmenter l'expressivité. Par exemple, l'égalité de deux variables s'exprime par la formule à deux variables libres $\varphi(x, y) = \neg\big(x < y \vee y < x\big)$, et le prédicat successeur que nous venons d'évoquer se définit par la formule à deux variables libres $\varphi(x, y) = (x < y) \wedge \forall z\big((x \geqslant z) \vee (y \leqslant z)\big)$, où $x \leqslant y$ est un raccourci pour $x = y \vee x < y$.

L'intérêt de ces nouveaux prédicats est que même s'ils sont définissables dans $\mathbf{MSO}[<]$, voire même dans $\mathbf{FO}[<]$ dans le cas du successeur, ils ne sont pas forcement définissable dans tout fragment de logique. Ainsi le fragment $\mathbf{FO}^2[<, S]$ a un plus grand pouvoir d'expression que $\mathbf{FO}^2[<]$.

On définit alors deux ensembles de prédicats pouvant être ajouté à la plupart des fragments étudiés dans la littérature.

**Définition 2.11: Prédicats locaux**
Les *prédicats locaux* est un ensemble des prédicats permettant de définir les propriétés de positions situées à une distance fixe. Il s'agit des prédicats suivants :
- une constante *min* désignant la première position de l'entrée,
- une constante *max* désignant la dernière position de l'entrée,
- le prédicat binaire successeur $S(x, y)$, vérifié si les variables $x$ et $y$ désignent

des positions successives du mot d'entrée.
On notera LOC l'ensemble de ces prédicats locaux.

**Définition 2.12: Prédicats Modulaires**
Les *prédicats modulaires* permettent de discuter de l'information modulaire des positions quantifiées. Ainsi, pour tout entier $d > 0$ et tout entier $i < d$, on définit le prédicat unaire $MOD_i^d(x)$ qui sera vrai si $x$ quantifie une position congrue à $i$ modulo $d$. On définit également, ce qui sera utile lorsque l'on considérera des fragments où le prédicat local $max$ n'est pas définissable, les prédicats 0-aires $D_i^d$ vrais si le mot d'entrée est d'une longueur congrue à $i$ modulo $d$.

On note MOD l'ensemble des prédicats modulaires. Ils seront largement étudiés dans le Chapitre 3 de ce manuscrit .

Un prédicat est dit numérique si son évaluation est indépendante de l'étiquetage du mot d'entrée, mais peut dépendre de sa longueur. Ainsi, l'ensemble des tuples satisfaisant un prédicat numérique peut être vu comme un sous-ensemble de $\mathbb{N}^{k+1}$, où $k$ est son arité. Il sera dit régulier si le sous-ensemble ainsi définit est régulier. Les trois ensembles de prédicats que sont $<$, LOC et MOD forment l'ensemble des prédicats numériques réguliers. Nous renvoyons à [Pél92] pour une étude poussée des prédicats réguliers.

Nous terminons cette partie par la définition de profondeur de quantification, qui sert dans de nombreux cas de témoin à la complexité d'une formule.

**Définition 2.13: Profondeur de quantification**
Étant donné une formule $\varphi$ de **MSO**, la *profondeur (ou rang) de quantification* de $\varphi$, notée $qr(\varphi)$, est le nombre maximal de symboles de quantificateurs imbriqués. Formellement, elle est définit de façon inductive sur les formules de la façon suivante :

- $qr(\top) = qr(\bot) = qr(\mathbf{a}(x)) = qr(x < y) = qr(x \in X) = 0$,
- $qr(\neg\varphi) = qr(\varphi)$,
- $qr(\varphi \wedge \psi) = qr(\varphi \vee \psi) = max(qr(\varphi), qr(\psi))$,
- $qr(\exists x\varphi) = qr(\forall x\varphi) = qr(\exists X\varphi) = qr(\forall X\varphi) = 1 + qr(\varphi)$.

## 2.4 Monoïdes et variétés

Dans cette dernière section, nous abordons l'algèbre, et plus particulièrement la théorie des semigroupes. Nous utiliserons les semigroupes finis comme autre modèle de définition des langages rationnels, et ils serviront, grâce aux morphismes, d'équivalent aux automates et aux formules logique.

### 2.4.1 Monoïdes et morphismes

Nous commençons par définir formellement les semigroupes et les monoïdes.

**Définition 2.14:**
Un *semigroupe* $(S, \cdot)$ est un ensemble $S$ équipé d'une loi de composition interne associative. Cette loi peut être vue comme une fonction $\cdot : S \times S \to S$ telle que pour tout $x, y, z$ dans $S$, on a $(x \cdot y) \cdot z = x \cdot (y \cdot z)$.

Un *monoïde* est un semigroupe $(S, \cdot)$ comportant un élément neutre $1_S$ tel que pour tout élément $X$ de $S$, on a $1_S \cdot x = x \cdot 1_S = x$.

Si $S$ est un semigroupe, on note $S^1$ le monoïde obtenu en ajoutant un nouvel élément défini comme neutre si $S$ n'en possède pas, et égal à $S$ sinon.

Lorsqu'il n'y a pas de confusion possible sur la définition de la loi interne, nous l'omettrons par abus de notation et noterons $xy$ le produit $x \cdot y$. On remarque également que l'élément neutre d'un monoïde est unique. En effet, si $e$ et $f$ sont des éléments neutres, alors par définition de l'élément neutre, $e = ef = f$. Nous donnons maintenant des exemples de monoïdes.

**Exemple 2.9.**
- L'ensemble des entiers naturels positifs $\mathbb{N}^+$, équipé de l'addition, forme un semigroupe.
- L'ensemble booléen $(\{\top, \bot\}, \wedge)$ forme un monoïde dont l'élément neutre est $\top$ appelé le monoïde booléen, noté $U_1$. Inversement $(\{\top, \bot\}, \vee)$ est un monoïde dont l'élément neutre est $\bot$.
- Étant donné un alphabet $A$, l'ensemble des mots $A^*$, équipé de la concaténation, est un monoïde, appelé le *monoïde libre* engendré par $A$ dont l'élément neutre est le mot vide $\epsilon$.
- Étant donné un entier $d$, le *groupe cyclique d'ordre d*, noté $C_d$, est l'ensemble $\{0, \ldots, d-1\}$ muni de la loi additive $i \oplus j = i + j \bmod d$.

La différence entre semigroupe et monoïde est simplement l'existence de cet élément neutre. Ceci peut sembler anodin, pourtant cela se révélera crucial dans l'étude des langages rationnels. En effet, si $A^*$ est un monoïde, $A^+ = A^* - \{\epsilon\}$ est l'ensemble des mots non vides, et forme un semigroupe. Certaines logiques, telles que certains fragments comportant les prédicats locaux, sont caractérisées par des classes de semigroupes plutôt que des classes de monoïdes. Leur étude nécessite donc d'utiliser la théorie des semigroupes. Les monoïdes étant un cas particulier des semigroupes, nos définitions seront, quand cela est possible, sur les semigroupes, avec une adaptation naturelle aux monoïdes.

Dans ce modèle de calcul, les monoïdes finis joueront le rôle des automates ou des formules logiques. Nous devons donc, comme dans les sections précédentes, définir un moyen de relier les mots et les langages à notre modèle. Ce rôle de sémantique, ou d'interprétation des mots dans un monoïde, est rempli par les morphismes, dont nous donnons une définition maintenant.

**Définition 2.15: Morphisme**
Soient deux semigroupes $S$ et $T$. Un *morphisme* de $S$ dans $T$ est une application $\eta$ vérifiant, pour tout couple $x, y$ d'éléments de $S$, $\eta(xy) = \eta(x)\eta(y)$.

Si en particulier $S$ et $T$ sont des monoïdes, on impose que $\eta(1_S) = 1_T$.

Étant donné un alphabet $A$, l'ensemble $A^*$ des mots sur $A$ forme un monoïde. On peut alors définir des morphismes de $A^*$ vers d'autres monoïdes. On appelle *timbre* un morphisme surjectif d'un monoïde libre finiment engendré, c'est-à-dire l'ensemble des mots sur un alphabet fini, vers un monoïde fini. De par la définition de morphisme, un timbre est entièrement défini par l'image des lettres. Nos timbres seront simplement décrits par l'image des lettres de l'alphabet de départ.

Nous définissons maintenant de façon formelle la notion de reconnaissance d'un langage par un morphisme.

**Définition 2.16: Reconnaissance par monoïde fini**
Soient $A$ un alphabet, $M$ un monoïde fini, $P$ une partie de ce monoïde et $\eta$ un timbre de $A^*$ vers $M$.

Un langage $L$ est *reconnu* par le triplet $(M, \eta, P)$ si, et seulement si, pour tout mot $u$ de $A^*$,

$$u \in L \Leftrightarrow \eta(u) \in P.$$

On peut le noter de façon équivalente par $L = \eta^{-1}(P)$, où $\eta^{-1}(P) = \{u \in A^* \mid \eta(u) \in P\}$.

Par abus de langage, nous dirons qu'un langage $L$ est reconnu par un monoïde $M$ s'il existe un timbre $\eta$ et une partie $P$ de $M$ tels que $L$ est reconnu par $(M, \eta, P)$.

**Exemple 2.10.** Soit $A = \{a, b\}$ un alphabet et $C_2 = \{0, 1\}$ le groupe cyclique d'ordre 2 équipé de l'addition. Soit maintenant le timbre $\eta$ définit par $\eta(a) = 1$ et $\eta(b) = 0$. Alors le langage $L$ des mots possédant un nombre pair de $a$ est exactement l'image inverse $\eta^{-1}(\{0\})$. Le langage $L$ est donc reconnu par $C_2$.

Il semble naturel de dresser un parallèle entre le triplet $(M, \eta, P)$ reconnaissant un langage par un monoïde et le quintuplet $(Q, A, \delta, i, F)$ définissant un automate. Ce parallèle est tout à fait pertinent, et il est possible de construire un automate à partir d'un monoïde et inversement.

Si $\mathcal{A} = (Q, A, \delta, i, F)$ est un automate déterministe, on peut voir un mot $u$ de $A^*$ comme une fonction $f_u : Q \to Q$ qui associe à un état $q$ l'état $q'$ tel que l'on ait $q \xrightarrow{u} q'$. On définit alors un monoïde $M$ comme l'ensemble des fonctions de $Q$ dans $Q$ réalisées par au moins un mot de $A^*$, et l'on équipe $M$ de la loi interne de composition de fonction inversée : $f_u f_v = f_v \circ f_u$. En posant naturellement $\eta(a) = f_a$ pour toute lettre $a$ et $P = \{f_u \in M \mid f_u(i) \in F\}$, alors le triplet $(M, \eta, P)$ reconnait exactement le même langage que $\mathcal{A}$. On appelle $M$ le *monoïde de transitions* de $\mathcal{A}$.
Inversement, si $(M, \eta, P)$, avec $M$ un monoïde fini, reconnait un langage $L$, on définit

alors $\mathcal{A} = (M, A, \delta, 1_M, P)$ tel que $\delta(m, a) = m\eta(a)$ pour tout élément $m$ de $M$ et toute lettre $a$. On a alors $L(\mathcal{A}) = L$.
Il est tout de même nécessaire de remarquer que ces constructions ne sont pas inverses l'un de l'autre. Néanmoins, ces constructions prouvent le résultat suivant qui, bien qu'attendu, est fondamental.

**Théorème 2.3.** *Les langages reconnaissables par monoïde fini sont exactement les langages réguliers.*

En revenant à ce parallèle entre automates et monoïdes, on peut rappeler qu'il existe, pour tout langage régulier, un automate minimal reconnaissant ce langage. Une notion analogue existe également dans la théorie des semigroupes. C'est la notion de monoïde syntaxique d'un langage régulier.

**Définition 2.17: Monoïde syntaxique**
Soit $L$ un langage rationnel sur un alphabet $A$. On définit la *congruence syntaxique* de $L$ sur $A^*$ de la façon suivante : pour toute paire de mots $u$ et $v$, on a $u \equiv_L v$ si, et seulement si :

$$\forall w, w' \in A^* \ \left(wuw' \in L \Leftrightarrow wvw' \in L\right)$$

**Remarque 2.4.** Cette congruence est d'indice fini si, et seulement si, $L$ est un langage rationnel. Le quotient $A^*/\equiv_L$, équipé de la concaténation, forme alors un monoïde fini appelé le *monoïde syntaxique* de $L$.

Il est de plus important de noter le lien entre monoïde syntaxique et automate minimal. En effet, pour tout langage régulier $L$, le monoïde syntaxique $M_L$ est isomorphe au monoïde de transitions de son automate minimal. Il s'agit d'ailleurs de l'algorithme principalement utilisé pour le construire.

Le monoïde syntaxique d'un langage est le plus petit monoïde reconnaissant ce langage. Il s'agit ici de plus petit au sens de la division de monoïde, dont nous donnons la définition ci-après.

**Définition 2.18: Division de monoïdes**
Soient deux semigroupes $S$ et $T$.
Un *sous-semigroupe* de $S$ est un ensemble clos par la loi interne de $S$. On dit que $S$ est un *quotient* de $T$ s'il existe un morphisme surjectif de $T$ sur $S$, et que $T$ *divise* $S$ si $T$ est le quotient d'un sous-semigroupe de $S$. La terminologie de quotient provient du fait que si $\mu : S \to T$ est surjectif, alors $T$ est isomorphe au quotient de $S$ par la relation $\sim_\mu$ (noté $S/\sim_\mu$) définie par $x \sim_\mu y$ si, et seulement si, $\mu(x) = \mu(y)$.

**Exemple 2.11.** Considérons le langage $L = (a + ba)^*$. Nous donnons d'abord son automate minimal dans la Figure 2.3, puis les fonctions sur les états réalisées par les mots

et nous décrivons le monoïde associé par une table de composition dans la Figure 2.4. Le monoïde syntaxique $M_L$ comporte 6 éléments $1, a, b, ab, ba, bb$.

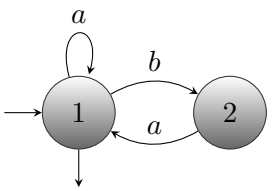



Figure 2.3: Automate minimal du langage $L = (a + ba)^*$

| $\epsilon$ | 1 | 2 |
|---|---|---|
| $a$ | 1 | 1 |
| $b$ | 2 | - |
| $ab$ | 2 | 2 |
| $ba$ | 1 | - |
| $bb$ | - | - |

| | $a$ | $b$ | $ab$ | $ba$ | $bb$ |
|---|---|---|---|---|---|
| $a$ | $a$ | $ab$ | $ab$ | $a$ | $bb$ |
| $b$ | $ba$ | $bb$ | $b$ | $bb$ | $bb$ |
| $ab$ | $a$ | $b$ | $ab$ | $bb$ | $bb$ |
| $ba$ | $ba$ | $b$ | $b$ | $ba$ | $bb$ |
| $bb$ | $bb$ | $bb$ | $bb$ | $bb$ | $bb$ |

Figure 2.4: Le tableau de gauche donne les fonctions de $Q$ dans $Q$, et celui de droite la table de composition de $M_L$, où l'on effectue le produit des lignes par les colonnes. L'élément neutre 1 y est omis.

Avant d'aborder les classes de monoïdes que sont les variétés, et leurs liens avec les fragments de logique, nous donnons quelques dernières définitions utiles sur les monoïdes.

**Définition 2.19: Éléments idempotents**
Soit un semigroupe $S$. Un élément $e$ de $S$ est dit *idempotent* si $ee$ est égal à $e$.
On note $E(S)$ l'ensemble des éléments idempotents de $S$.

Tout élément $x$ d'un semigroupe fini $S$ possède une *puissance idempotente*, c'est-à-dire un entier $n$ tel que $x^n$ soit idempotent. En effet, puisque nous sommes dans un semigroupe fini, il existe en particulier deux entiers $i$ et $k > 0$ tels que $x^i = x^{i+k} = x^{i+nk}$ pour tout $n$, en particulier $x^i = x^{i+ik}$. On a alors $x^i \cdot x^{i(k-1)} = x^{i+ik} \cdot x^{i(k-1)}$ et donc $x^{ik} = x^{2ik}$ est idempotent.

**Définition 2.20: Relations de Green [Gre51]**
En 1951, Green a introduit 5 relations de semigroupes décrivant la structure interne d'un semigroupe. Nous présentons ici les 4 relations qui se révéleront utiles dans notre étude. Ces relations peuvent être considérées comme une généralisation de la

notion de division sur les entiers.
Soit un semigroupe $S$ et deux éléments $x$ et $y$ de $S$. On définit les relations de Green $\leqslant_{\mathcal{R}}$, $\leqslant_{\mathcal{L}}$, $\leqslant_{\mathcal{H}}$ et $\leqslant_{\mathcal{J}}$ de la façon suivante :
- $x \leqslant_{\mathcal{R}} y$ si, et seulement si, il existe un élément $z$ de $S^1$ tel que $x = zy$,
- $x \leqslant_{\mathcal{L}} y$ si, et seulement si, il existe un élément $z$ de $S^1$ tel que $x = yz$,
- $x \leqslant_{\mathcal{H}} y$ si, et seulement si, on a $x \leqslant_{\mathcal{R}} y$ et $x \leqslant_{\mathcal{L}} y$ ,
- $x \leqslant_{\mathcal{J}} y$ si, et seulement si, il existe deux éléments $w$ et $z$ de $S^1$ tel que $x = wyz$,

Ces relations sont des pré-ordres pour tout semigroupe et s'étendent naturellement à des relations d'équivalence :
Pour $\mathcal{X}$ un des symboles $\mathcal{R}$, $\mathcal{L}$, $\mathcal{H}$ ou $\mathcal{J}$, on notera $x\mathcal{X}y$ si, et seulement si, $x \leqslant_{\mathcal{X}} y$ et $y \leqslant_{\mathcal{X}} x$.

Nous ne faisons qu'effleurer l'étude de ces relations qui sont cependant utilisées afin de prouver de nombreuses propriétés sur les semigroupes. Nous renvoyons le lecteur à [Pin97] pour de plus amples détails.

Nous donnons maintenant la définition de l'indice de stabilité d'un morphisme. Cette notion, introduite par Straubing (voir [Str94]), n'est pas aussi standard que la plupart des outils présentés dans ce chapitre. Elle se révélera néanmoins primordiale au travers des études présentes dans ce manuscrit.

**Définition 2.21: Indice de stabilité d'un morphisme**
Soit un timbre $\eta : A^* \to M$. On considère alors l'ensemble $\eta(A)$ des images des lettres de l'alphabet. Cet ensemble est un élément du monoïde fini $(\mathcal{P}(M), \cdot)$ des parties de $M$, où $\mathcal{P}(M)$ est équipé de la loi

$$XY = \{m \in M \mid \text{ il existe } x \in X \text{ et } y \in Y \ xy = m\}.$$

On appelle alors *indice de stabilité* du morphisme $\eta$ la puissance d'idempotence de $\eta(A)$ dans $\mathcal{P}(M)$. On note usuellement cet entier $s$, et l'on a donc $\eta(A^s) = \eta(A^{2s})$. L'ensemble $\eta(A^s)$ est un sous-semigroupe de $M$ appelé le *semigroupe stable* de $\eta$.

Du point de vue des mots, cela signifie que pour tout mot de longueur $s$, il existe un mot de longueur $2s$ équivalent selon $\eta$, et réciproquement. Cela se généralise d'ailleurs à tout mot de longueur multiple de $s$. L'indice de stabilité, comme cela se vérifiera dans ce manuscrit, joue un rôle prépondérant dans l'étude des prédicats modulaires introduits à la section précédente.

### 2.4.2 Caractérisation algébrique de fragments logiques

Après avoir défini l'objet qu'est un monoïde, et ses liens avec les automates ainsi que les langages rationnels, nous abordons maintenant les classes de monoïdes que sont les variétés, qui seront utilisées pour caractériser de façon effective les classes de langages définies par des fragments logiques.

Nous définissons formellement les variétés de monoïdes finis, définition due à Eilenberg [Eil76], puis nous aborderons la caractérisation d'un fragment par une variété, comment cette caractérisation mène à la décidabilité des langages définissables par un fragment, et enfin nous verrons certaines extensions connues de ces caractérisations.

Nous définissons tout d'abord le produit cartésien de deux monoïdes $(S, \cdot_S)$ et $(T, \cdot_T)$ comme le monoïde défini sur l'ensemble $S \times T$ des couples d'éléments de $S$ et $T$, équipé de la loi interne $(s,t) \cdot (s',t') = (s \cdot_S s', t \cdot_T t')$.

**Définition 2.22: Variété de monoïdes finis**
Une classe de monoïdes finis $\mathbf{V}$ est une *variété* si elle vérifie les conditions suivantes, pour toute paire de monoïdes $S$ et $T$ :
- Si $S \in \mathbf{V}$ et si $T$ divise $S$, alors $T \in \mathbf{V}$,
- le monoïde trivial $\{1\}$ appartient à $\mathbf{V}$,
- Si $S$ et $T$ appartiennent à $\mathbf{V}$, alors le produit $S \times T$ appartient également à $\mathbf{V}$.

Il est important de noter que nous ne définissons et n'utiliserons dans ce manuscrit que des variétés de monoïdes finis, appelées pseudo-variétés dans la littérature algébrique. Puisqu'aucune confusion n'apparaitra, nous nous permettons ici cet abus de langage dans un but de clarté.

**Exemple 2.12.** Nous donnons ici quelques exemples de variétés. On rappelle également que l'on ne considère dans ce manuscrit que les monoïdes finis. Les variétés définies ci-après sont donc implicitement restreintes aux monoïdes finis.
- L'ensemble réduit, à isomorphisme près, au monoïde trivial,
- la classe **Com** des monoïdes commutatifs,
- la classe **A** des monoïdes apériodiques, c'est-à-dire des monoïdes divisés par aucun groupe à part le groupe trivial. Alternativement, il s'agit des monoïdes tels qu'il existe un entier $n$ et pour tout élément $x$, $x^n = x^{n+1}$,
- la classe **G** des groupes.

Cette dénomination de variété est commune avec les variétés de langages définie en 2.5. Ceci n'est pas anodin et est justifiée par le Théorème des variétés d'Eilenberg et Schützenberger [ES76].

**Théorème 2.4** (Théorème des variétés)**.** *Il existe une bijection naturelle entre les variétés de langages et les variétés de monoïdes.*

Ainsi, chaque variété de monoïdes définit de façon unique une variété de langages, et réciproquement. La vision logique de ce théorème est que si un fragment de logique est robuste, dans le sens où les langages qu'il définit forment une variété de langages, il sera alors caractérisé de façon unique par une variété de monoïdes. Nous donnons maintenant un premier exemple de cette caractérisation pour notre exemple fil rouge des langages

sans-étoile. Cette caractérisation est due conjointement à McNaughton&Papert [MP71] pour le côté logique, et Schützenberger [Sch65] pour le côté algébrique.

**Théorème 2.5.** *Soit un langage régulier $L$. Les propositions suivantes sont équivalentes :*
- *$L$ est un langage sans-étoile,*
- *$L$ est définissable par une formule de $\mathbf{FO}[<]$,*
- *le monoïde syntaxique de $L$ est apériodique ($M_L$ appartient à $\mathbf{A}$).*

Nous donnons d'autres caractérisations à titre d'exemples.

**Exemple 2.13.**
- Les langages définissables par des formules de $\mathbf{FO}^1[]$, c'est-à-dire le premier ordre n'ayant qu'un unique symbole de variable, est caractérisé par $\mathbf{J}_1$, l'ensemble des monoïdes idempotents et commutatifs (voir par exemple [DGK08]).
- Les langages définissables par une formule de $\mathbf{MSO}[=]$ sont exactement les langages reconnaissables par des monoïdes commutatifs.

Avant d'aborder la décidabilité des variétés, nous donnons une dernière notion, qui est le produit en couronne de monoïdes. Cette définition donne lieu à une opération de variétés qui se révélera primordiale dans le Chapitre 3.

**Définition 2.23: Produit en couronne**
Soient deux monoïdes $(M, \cdot)$ et $(N, +)$. Le produit en couronne de $M$ et $N$, noté $M \circ N$, est défini sur l'ensemble $M^N \times N$ où $M^N$ désigne l'ensemble des fonctions de $N$ dans $M$. Il est équipé de la loi de composition interne $\times$ définie de la façon suivante :

$$(f, n) \times (g, n') = (h, n + n') \text{ où } h(x) = f(x) \cdot g(n + x).$$

**Remarque 2.5.** Cette loi de composition peut être vue comme une action du monoïde $N$ sur le monoïde $M$. En effet, la première composante est une fonction de $N$ dans $M$, et correspond donc à une paramétrisation de $M$ par $N$. La composition sert alors à effectuer en parallèle un calcul sur $N$, et la fonction composée est calculée grâce aux fonction des deux éléments en prenant en compte l'image selon $N$ du premier élément.

Cette notion se généralise aux variétés de monoïdes, et l'on appelle le *produit semi-direct* de $\mathbf{V}$ par $\mathbf{W}$, noté $\mathbf{V} * \mathbf{W}$, l'ensemble des monoïdes divisant le produit en couronne d'un monoïde de $\mathbf{V}$ par un monoïde de $\mathbf{W}$. Pour toutes variétés de monoïdes $\mathbf{V}$ et $\mathbf{W}$, le produit semi-direct $\mathbf{V} * \mathbf{W}$ est également une variété.

À titre d'exemple, nous signalons une conséquence directe du Théorème de décomposition de Krohn-Rhodes [KR65], stipulant que les monoïdes apériodiques est la variété engendrée par itération du produit en couronne du monoïde booléen $U_1$, défini en exemple plus tôt dans ce manuscrit.

**Décidabilité des variétés par les équations**

Une question importante dans l'étude d'un fragment de logique est la définissabilité : étant donné un langage régulier, peut-on décider s'il existe une formule du fragment définissant ce langage. D'un point de vue algébrique, cette question devient : existe-t-il un monoïde de la variété reconnaissant le langage. Étant donné qu'une variété est close par division de monoïde, et que le monoïde syntaxique d'un langage divise tout monoïde le reconnaissant, cette question se réduit à pouvoir décider si le monoïde syntaxique du langage appartient à la variété.

Nous sommes intéressés par la possibilité de décider si un monoïde appartient à une variété. Cette décidabilité peut s'obtenir grâce à des propriétés particulières de la variété. Ainsi, on peut décider si un monoïde est apériodique en calculant chacun de ses sous-monoïdes et tester si ce sont des groupes. Les monoïdes considérés étant finis, ils ont un nombre fini de sous-semigroupes et la variété **A** est décidable.

Il existe cependant une approche plus générale pour obtenir la décidabilité de variétés de monoïde, qui est la description par un ensemble fini d'équations profinies calculables. Afin de pouvoir définir ces équations, nous allons devoir faire maintenant une excursion rapide dans la topologie, et plus particulièrement la topologie profinie sur les mots.

Nous définissons une distance sur les mots. Pour cela, nous commençons par définir la fonction suivante, pour tout couple de mots $u$ et $v$ sur un alphabet $A$ :

$$r(u,v) = min\{|M| \mid \text{ il existe un morphisme } \eta : A^* \to M \text{ tel que } \eta(u) \neq \eta(v)\}$$

puis la fonction $d(u,v) = 2^{-r(u,v)}$. Il est alors prouvé que la fonction $d$ est une distance ultra-métrique sur $A^*$. Cette distance permet alors la définition d'une topologie, et la complétion de $A^*$ pour cette métrique donne alors l'espace des mots profinis sur $A^*$, noté $\widehat{A^*}$.

Il est connu (voir [Pin97]) que l'espace des mots profinis $\widehat{A^*}$ est compact si $A$ est fini. De plus, tout timbre $\eta : A^* \to M$ s'étend de façon unique en un morphisme uniformément continu $\widehat{\eta} : \widehat{A^*} \to M$. L'espace $\widehat{A^*}$ est strictement plus grand que $A^*$. Cependant, même s'il existe une infinité de mots profinis, il n'est pas si aisé d'en donner des exemples concrets. L'exemple le plus fondamental de mots profinis , et un des seuls connus, est la puissance $\omega$ d'un mot. Étant donné un mot $u$ sur un alphabet $A$, on considère la suite $(u^{n!})_{n\in\mathbb{N}}$. On peut alors prouver que cette suite est une suite de Cauchy, et converge dans le complété $\widehat{A^*}$ de $A^*$. Sa limite est alors notée $u^\omega$, la puissance oméga de $u$. Pour tout timbre $\eta : A^* \to M$ et tout mot $u$, il est aisé de constater que l'unique extension $\widehat{\eta} : \widehat{A^*} \to M$ de $\eta$ envoie le mot profini $u^\omega$ sur la puissance idempotente de $\eta(u)$. En effet, si $k$ est la puissance idempotente de $u$ sur $\eta$, alors pour tout $n \geqslant k$, $u^{n!}$ est une puissance de $u^k$ et par conséquent s'envoie sur un idempotent de $M$. Ainsi, le mot profini $u^\omega$ correspondra toujours à la puissance idempotente du mot $u$.

Forts de cet objet, nous sommes désormais à même de définir les équations profinies, qui nous serviront à caractériser les variétés de monoïdes.

**Définition 2.24: Équations profinies**
Soit un alphabet fini $A$. Une *équation profinie* est l'égalité d'une paire de mots profinis $u$ et $v$ de $\widehat{A^*}$, notée naturellement $u = v$.
Un monoïde $M$ *satisfait* l'équation profinie $u = v$ si, et seulement si, pour tout timbre $\eta : A^* \to M$, on a $\widehat{\eta}(u) = \widehat{\eta}(v)$. On note $[\![u = v]\!]$ la classe des monoïdes finis satisfaisant l'équation $u = v$.

Le lien entre variétés de monoïdes et équations profinies est explicité par le Théorème de Reiterman, que nous redonnons ici.

**Théorème 2.6** (Théorème de Reiterman [Rei82])**.** *Une classe de monoïdes finis est une variété si, et seulement si, elle est définissable comme l'ensemble des monoïdes finis satisfaisant un ensemble donné d'équations profinies.*

**Exemple 2.14.**
- La variété **Com** des monoïdes commutatifs est naturellement caractérisée par l'équation $[\![xy = yx]\!]$. L'utilisation de variables $x$ et $y$ signifie ici que l'équation est vérifiée pour tout couple de mots profinis.
- La variété **A**, équivalente aux langages sans-étoile, est égale à $[\![x^\omega = x^{\omega+1}]\!]$, où $x^{\omega+1} = x \cdot x^\omega$. Ainsi pour chaque monoïde apériodique $M$, il existe un entier $n$ tel que pour tout élément $x$ de $M$, $x^n = x^{n+1}$. L'entier $n$ est alors *l'indice d'apériodicité* de $M$.
- La variété **DA**, équivalente au fragment $\mathbf{FO}^2[<]$, est caractérisée par l'équation $[\![(xy)^\omega(yx)^\omega(xy)^\omega = (xy)^\omega, x^\omega = x^{\omega+1}]\!]$ ou $[\![(xyz)^\omega y(xyz)^\omega = (xyz)^\omega]\!]$.
- La variété des monoïdes $\mathcal{J}$-triviaux, notée **J**, et équivalente aux combinaisons booléennes de formules de $\Sigma_1[<]$, est caractérisée par $[\![y(xy)^\omega = (xy)^\omega = (xy)^\omega x]\!]$.
- La variété des semigroupes localement triviaux **LI** est caractérisée par l'équation $[\![x^\omega y x^\omega = x^\omega]\!]$.

Ainsi, si l'on obtient une caractérisation simple d'une variété, on peut alors en déduire la décidabilité du problème de l'appartenance. En effet, décider si un monoïde donné appartient à une variété revient donc à vérifier les équations de la variété sur ce monoïde. S'il n'existe qu'un nombre fini d'équations calculables à tester, le problème devient alors automatiquement décidable. C'est ainsi le cas lorsqu'une variété est décrite par un ensemble fini d'équations du type $f(x_1, \ldots, x_n) = g(y_1, \ldots, y_m)$.

Il existe cependant plusieurs extensions de cette théorie, qui peut se révéler assez restrictive. En effet, telle qu'énoncée pour l'instant, cette théorie ne permet d'étudier que des fragments logiques correspondant à des variétés de langages. Cependant, un certain nombre de fragments naturels ne définissent pas une variété de langages, mais toutefois quelque chose approchant.

Ainsi, il est possible de restreindre la condition de clôture par morphisme inverse. En effet, une des conditions de la définition de variété est la suivante :

$$\text{Pour tout morphisme } \eta : A^* \to B^*, L \in \mathcal{V}(B^*) \implies \eta^{-1}(L) \in \mathcal{V}(A^*).$$

Cependant, certains fragments de logique, notamment certains fragments contenant les prédicats modulaires, définissent des classes de langages qui ne sont pas closes par inverse de tout morphisme. Nous affaiblissons alors cette hypothèse de la façon suivante :

$$\text{Pour tout morphisme } \eta : A^* \to B^* \text{ dans une classe } \mathcal{C}, L \in \mathcal{V}(B^*) \implies \eta^{-1}(L) \in \mathcal{V}(A^*).$$

où $\mathcal{C}$ est une classe de morphismes comportant l'identité et stable par composition.

Afin de définir les variétés de timbres, on a alors besoin des notions de produit ainsi que de $\mathcal{C}$-division de timbres que nous donnons maintenant :

- Soient $\varphi : A^* \to M$ et $\psi : A^* \to N$ deux timbres. On considère alors le monoïde produit $M \times N$ équipé de la loi interne $(x, s)(y, t) = (xy, st)$ pour tout $x, y \in M$ et $s, t \in N$. Le *timbre produit* de $\varphi$ et $\psi$ est alors défini sur le monoïde produit pour toute lettre $a$ de $A$ par $\eta(a) = (\varphi(a), \psi(a))$. Il est à noter que l'image de $\eta$ n'est pas le monoïde produit en entier mais le sous-monoïde généré par l'image des lettres, c'est-à-dire l'ensemble $\{(\varphi(u), \psi(u)) \mid u \in A^*\}$.
- Un timbre $\varphi : A^* \to M$ *$\mathcal{C}$-divise* un autre timbre $\psi : B^* \to N$ si, et seulement si, il existe une paire de morphismes $(\alpha, \beta)$ telle que $\alpha$ est un $\mathcal{C}$-morphisme de $A^*$ dans $B^*$, $\beta : N \to M$ est un morphisme partiel surjectif et nous avons l'égalité $\varphi = \beta \circ \psi \circ \alpha$, correspondant à la commutativité du diagramme de la Figure 2.5. La paire $(\alpha, \beta)$ est appelée la *$\mathcal{C}$-division*.

Une *variété de timbres* est alors un ensemble de timbres clos par division et produit fini. On ne considère alors plus des variétés de monoïdes, mais des variétés de timbres, et la théorie équationnelle s'étend en remplaçant, dans toutes les définitions, les morphismes par $\mathcal{C}$-morphismes.

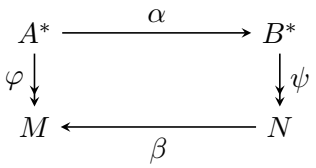



Figure 2.5: $\mathcal{C}$-division de timbres

Voici maintenant les principales classes de morphismes utilisées en pratique :

*all* Si l'on s'autorise tous les morphismes, on retrouve alors les variétés de monoïdes classiques. Les variétés de timbres forment donc une extension des variétés de monoïdes.

*ne* Les morphismes non effaçant (ou *ne*-morphismes) sont les morphismes tels que l'image d'une lettre n'est jamais vide. Les variétés non effaçantes sont équivalentes aux variétés de semigroupes, et ne considèrent ainsi pas le mot vide.

*lm* Les morphismes multipliant la longueur (*lm*-morphismes) est l'ensemble des morphismes où les images de toutes les lettres ont une même longueur non nulle. En particulier, les variétés caractérisant des fragments avec prédicats modulaires seront des *lm*-variétés de timbres.

*lp* Les morphismes préservant la longueur (*lp*-morphismes) est l'ensemble des morphismes qui envoient chaque lettre sur une lettre. Ils préservent ainsi la longueur entre un mot et son image. Il s'agit de la classe la plus restrictive.

**Exemple 2.15.** Si **V** est une variété de monoïdes, alors l'ensemble des timbres dont le monoïde stable appartient à **V** forme est appelé la **Q**-*variété* **QV**. Elle forme une *lm*-variété de timbres.

Finalement, nous recensons dans le tableau 2.6 différentes classes de langages largement étudiées, ainsi que leurs caractérisations en terme de logique et d'algèbre. Nous définissons, pour une classe de langage $\mathcal{L}$, la *clôture booléenne* $\mathcal{BL}$ de $\mathcal{L}$ comme le plus petit ensemble de langages contenant $\mathcal{L}$ et clos par complément ainsi que par union et intersection finies.

| Langages | Logique | Variété | Équations |
|---|---|---|---|
| Langages rationnels | $\mathbf{MSO}[<]$ | Monoïdes finis | - |
| Langages commutatifs | $\mathbf{MSO}[=]$ | **Com** | $xy = yx$ |
| Sans-étoile | $\mathbf{FO}[<]$ | **A** | $x^\omega = x^{\omega+1}$ |
| Polynômes non-ambigus | $\mathbf{FO}^2[<]$ | **DA** | $(xy)^\omega x(xy)^\omega = (xy)^\omega$ |
| $\mathcal{B}((\{u \mid \alpha(u) = B\})_{B\subseteq A})$ | $\mathbf{FO}^1[<] = \mathbf{FO}^1[]$ | $\mathbf{J}_1$ | $x^2 = x, xy = yx$ |
| $\mathcal{B}(A^*a_1A^* \dots a_nA^*)$ | $\mathcal{B}\Sigma_1[<]$ | **J** | $y(xy)^\omega = (xy)^\omega = (xy)^\omega x$ |

Figure 2.6: Quelques classes de langages connues

# Chapitre 3

# De l'ajout des prédicats modulaires

Dans ce chapitre, nous présentons la contribution principale de cette thèse. Il s'agit d'un travail en commun avec Charles Paperman. Les résultats présentés peuvent être retrouvés dans [DP13] et [DP14]. Ce chapitre se concentre sur l'étude du comportement de fragments de la logique sur les mots finis, lorsque l'on y ajoute les prédicats modulaires. Plus précisément, nous étudions comment le problème de décision se comporte lorsque l'on ajoute les prédicats modulaires à un fragment dont la décidabilité est connue. Cette question n'est pas nouvelle et a déjà été abordée pour certains fragments spécifiques. Ainsi Barrington et al. [BCST92] ont prouvé que la logique du premier ordre enrichie avec les prédicats modulaires $\mathbf{FO}[<, \mathrm{MOD}]$ était algébriquement caractérisée de façon effective par la $lm$-variété de timbres $\mathbf{QA}$. Plus tard, Chaubard, Pin et Straubing [CPS06] ont prouvé la décidabilité du problème de décision pour $\Sigma_1[<, \mathrm{MOD}]$ et $\mathcal{B}\Sigma_1[<, \mathrm{MOD}]$, en utilisant également une approche algébrique.
Ce chapitre suit cette idée. La première partie s'intéresse au fragment $\mathbf{FO}^2[<, \mathrm{MOD}]$, et nous obtenons une caractérisation algébrique effective du fragment $\mathbf{FO}^2[<, \mathrm{MOD}]$, en adaptant la preuve de Thérien et Wilke [TW99] pour $\mathbf{FO}^2[<]$. La seconde partie prend une approche plus générale. Grâce à des outils de plus haut niveau tels que les variétés de catégories, nous obtenons des critères suffisants sur les fragments de logique qui préservent la décidabilité du problème de décision lorsque l'on leur ajoute les prédicats modulaires.
Bien que la seconde partie apporte une nouvelle preuve plus générique du résultat prouvé avant, les idées abordées et l'approche plus précise de la première partie m'ont incité à la placer ici.

## 3.1 Prédicats modulaires et alphabet enrichi

Nous abordons ici une notion clé dans l'étude des prédicats modulaires. Il s'agit d'enrichir notre alphabet par des entiers symbolisant la congruence d'une lettre modulo un entier donné. Soit $d$ un entier positif. On note $\mathrm{MOD}_d$ la restriction des prédicats MOD aux congruences modulo $d$. Le lemme suivant, dont la preuve est relativement simple, prouve que dans les cas que nous considérerons, cela n'engendre aucune perte de

généralité.

**Lemme 3.1.** *Soit $\mathcal{F}$ un fragment de logique et $L$ un langage définissable dans $\mathcal{F}[<, \mathrm{MOD}]$. Alors il existe un entier $d$ tel que $L$ est définissable dans $\mathcal{F}[<, \mathrm{MOD}_d]$.*

*Démonstration.* Puisque $L$ appartient à $\mathcal{F}[<, \mathrm{MOD}]$, il existe une formule $\varphi$ de $\mathcal{F}[<, \mathrm{MOD}]$ telle que $L = L(\varphi)$. Comme $\varphi$ est finie, elle utilise un nombre fini de congruences. Or si $e$ est un entier et $d = ke$ est un multiple de $e$, alors pour tout $i < e$, le prédicat $\mathrm{MOD}_i^e(x)$ est équivalent à $\bigwedge_{\ell=0}^{k-1} \mathrm{MOD}_{i+\ell e}^d(x)$. Or puisque $\mathcal{F}$ est un fragment, on peut remplacer un atome par une disjonction d'atomes sans sortir de $\mathcal{F}$. Si alors on définit $d$ comme le plus petit commun multiple des congruences apparaissant dans $\varphi$, on peut définir $\varphi'$ comme étant la formule $\varphi$ où l'on a remplacé chaque prédicat modulaire par son équivalent modulo $d$. On a bien $\varphi' \in \mathcal{F}[<, \mathrm{MOD}_d]$ et $L = L(\varphi')$. □

Nous définissons maintenant l'alphabet enrichi selon un entier fixé $d$. Il faut remarquer que les fonctions et ensembles définis ci-après devraient, pour plus de précision, être indexés par $d$. Cependant, puisque nous fixons l'entier $d$, et qu'il n'apparait pas de risque de confusion, nous omettrons souvent l'entier $d$ pour plus de clarté. On rappelle que $C_d$ est le groupe cyclique d'ordre $d$.

**Définition 3.1: Alphabet enrichi**
Soit $A$ un alphabet. On appelle l'ensemble $A_d = A \times C_d$ *l'alphabet enrichi* de $A$ selon $d$, et l'on note $\pi_d : A_d^* \to A^*$ la projection sur la première coordonnée : $\pi_d(a, i) = a$ pour tout $(a, i) \in A_d$.

Par exemple, le mot $(a,2)(b,1)(b,2)(a,0)$ est un des mots enrichis du mot *abba* pour $d = 3$. On dit que *abba* est le mot *sous-jacent* à $(a,2)(b,1)(b,2)(a,0)$. On définit maintenant les mots bien formés. De façon informelle, il s'agit des mots dont la deuxième composante de chaque lettre correspond à sa position modulo $d$.

**Définition 3.2: Mots bien formés**
Un mot $(a_0, i_0)(a_1, i_1) \cdots (a_n, i_n)$ de $A_d$ est *bien formé* si pour tout $0 \leqslant j \leqslant n$, $i_j \equiv j \bmod d$. On note $K_d$ l'ensemble des mots bien formés de $A_d^*$.

Par définition, pour tout mot de $A^*$, il existe un unique mot bien formé $v$ tel que $u$ soit le mot sous-jacent à $v$. On définit une opération qui prend un mot donné et retourne l'unique mot bien formé associé.

**Définition 3.3:**
Pour tout mot $u = a_0 a_1 \cdots a_n \in A^*$, on définit le mot $\overline{u} = (a_0, 0)(a_1, 1) \cdots (a_i, i \bmod d) \cdots (a_n, n \bmod d)$. Le mot $\overline{u}$ est alors le *mot bien formé associé* à u.

**Remarque 3.1.** L'application $u \to \overline{u}$ n'est pas un morphisme. En effet, si la longueur d'un mot $u$ n'est pas égale à 0 modulo $d$, alors $\overline{uv}$ sera différent de $\overline{u}\,\overline{v}$. Pour remédier à

ce problème, on définit les opérations de *k-décalage*, notée $\overline{u}^k$, qui envoient un mot vers son mot bien formé décalé d'un entier $k$. Ainsi le mot $u = u_0 \cdots u_n$ est envoyé vers le mot enrichi $(u_0, k \bmod d)(u_1, k+1 \bmod d) \cdots (u_n, n+k \bmod d)$. Ainsi, si le mot $u$ est d'une longueur congrue à $k$ modulo $d$, alors $\overline{uv} = \overline{u}\,\overline{v}^k$.

Sur l'ensemble des mots bien formés, la projection $\pi_d$ est une bijection. On a $\pi_d(\overline{u}) = u$ mais $\overline{\pi_d(v)}$ est différent de $v$ si $v$ n'est pas bien formé. Le lemme suivant est facile et découle de cette observation.

**Lemme 3.2.** *Soient $L$ et $L'$ deux langages de $A_d^*$, alors les égalités suivantes sont vérifiées :*

1. $\pi_d((L \cup L') \cap K_d) = \pi_d(L \cap K_d) \cup \pi_d(L' \cap K_d)$,
2. $\pi_d((L \cap L') \cap K_d) = \pi_d(L \cap K_d) \cap \pi_d(L' \cap K_d)$
3. $\big(\pi_d(L \cap K_d)\big)^c = \pi_d(L^c \cap K_d)$.

## 3.2 Un exemple concret : $\mathbf{FO}^2[<, \mathrm{MOD}]$

Cette section se concentre sur l'étude du fragment de logique $\mathbf{FO}^2[<, \mathrm{MOD}]$. Le résultat principal est l'obtention d'une caractérisation algébrique effective de ce fragment (voir Théorème 3.4), prouvant ainsi la décidabilité du problème de décision. De par notre approche, similaire à celle de Thérien et Wilke pour le fragment $\mathbf{FO}^2[<]$, nous obtenons également une définition équivalente en termes de logique temporelle (voir Proposition 3.21) ainsi qu'une description des langages associés comme expressions régulières (voir Proposition 3.22).

### 3.2.1 Les automates itérés

Il s'agit ici de définir l'équivalent pour les automates de la notion de monoïde stable. Cette notion sera utilisée dans la partie 3.2.3. On définit les automates itérés comme des automates qui ne lisent pas des lettres mais des mots d'une longueur donnée.
Formellement, on prend un automate déterministe $\mathcal{A} = (Q, A, \cdot)$ sans état initial ni états finals et $k$ un entier positif. Nous définissons le *k-automate* de $\mathcal{A}$ comme l'automate déterministe $\mathcal{A}_k = (Q, A^k, \cdot^k)$ où la fonction de transition est définie comme suit : $q \cdot^k (a_1 a_2 \cdots a_k) = (\cdots (q \cdot a_1) \cdot a_2) \cdots) \cdot a_k)$. On remarque que si $M$ est le monoïde de transitions de $\mathcal{A}$, et $M_k$ celui de $\mathcal{A}_k$, alors $M_k$ est le sous-monoïde de $M$ engendré par l'image des mots de longueur $k$.
On définit maintenant un équivalent pour les automates de la propriété de stabilité d'un monoïde.

**Définition 3.4:**
Soit un automate déterministe $\mathcal{A} = (Q, B, \cdot)$. On dit que $\mathcal{A}$ est *stable* si pour tout

mot de deux lettres, il existe une lettre ayant la même action sur les états de $\mathcal{A}$, et inversement pour toute lettre de $B$, il existe un mot de deux lettres ayant la même action.

**Exemple 3.1.** Nous donnons l'exemple de l'automate de la Figure 3.1. Dans cet automate, les fonctions réalisées par les mots $ab$ et $ba$ sont égales à la fonction réalisée par la lettre $a$. Les mots $aa$ et $bb$ réalisent la même fonction que la lettre $b$. Inversement les fonctions réalisées par les lettres sont équivalentes aux fonctions réalisées par les mots de longueur 2.

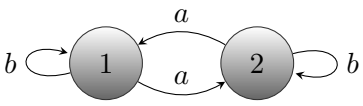



Figure 3.1: Un automate stable

Cette définition revient à dire que le morphisme associé à un automate stable à un indice de stabilité de 1. Ainsi, la proposition suivante montre qu'il s'agit d'une définition compatible avec celle de semigroupe stable.

**Proposition 3.3.** *Soit un automate déterministe $\mathcal{A}$. Alors il existe un entier $k$ tel que le $k$-automate $\mathcal{A}_k$ soit stable.*

Cette proposition se prouve aisément en constatant que les itérations d'un automate à isomorphisme près forment un semigroupe fini, et que l'indice de stabilité correspond alors à la puissance d'idempotence de l'automate initial.

Le plus petit entier $k$ satisfaisant cette proposition est appelé l'*indice de stabilité* de l'automate. Il est égal à l'indice de stabilité du timbre associé. Dans la suite, l'indice de stabilité d'un langage fera référence indifféremment à celui de son automate minimal ou de son timbre syntaxique.

### 3.2.2 Caractérisation algébrique

Nous donnons ici la caractérisation algébrique de fragment $\mathbf{FO}^2[<, \mathrm{MOD}]$. On rappelle que si $\mathbf{V}$ est une variété de monoïdes, alors la $\mathbf{Q}$-variété $\mathbf{QV}$ est la classe des timbres tels que leur monoïde stable est dans $\mathbf{V}$ (voir Exemple 2.15). On rappelle également que les langages définis par le fragment $\mathbf{FO}^2[<]$ sont exactement ceux dont le monoïde syntaxique est dans $\mathbf{DA}$ [TW99].

**Théorème 3.4.** *Soit $L$ un langage régulier. Alors $L$ est définissable dans $\mathbf{FO}^2[<, \mathrm{MOD}]$ si, et seulement si, son timbre syntaxique appartient à $\mathbf{QDA}$.*

De plus, étant donné un langage régulier, sous forme d'une expression régulière ou d'un automate, on peut effectivement construire le monoïde stable de son timbre syntaxique. Alors puisque l'on peut décider si un monoïde appartient à la variété **DA**, on obtient le corollaire suivant.

**Corollaire 3.5.** *Étant donné un langage régulier $L$, on peut effectivement décider si $L$ est définissable dans $\mathbf{FO}^2[<, \mathrm{MOD}]$.*

Nous prouverons la première inclusion $\mathbf{FO}^2[<, \mathrm{MOD}] \subseteq \mathbf{QDA}$ dans la prochaine section, grâce à des arguments sur les automates et la logique. La seconde inclusion est prouvée dans la Section 3.2.4, grâce à des jeux d'Ehrenfeucht-Fraïssé et des outils algébriques définis dans le Chapitre 2.

### 3.2.3 L'inclusion $\mathbf{FO}^2[<, \mathrm{MOD}] \subseteq \mathbf{QDA}$

Nous prouvons ici la première inclusion du Théorème 3.4. La preuve utilise l'alphabet enrichi, les mots bien formés et les automates. Cette première proposition permet de relier, grâce aux mots bien formés, les formules et langages définis sur un alphabet donné par le fragment $\mathbf{FO}^2[<, \mathrm{MOD}]$ avec les langages définis par $\mathbf{FO}^2[<]$ sur l'alphabet enrichi.

**Proposition 3.6.** *Soit $d$ un entier positif. Alors pour tout alphabet $A$, les langages définissables dans $\mathbf{FO}^2[<, \mathrm{MOD}_d]$ sont exactement les langages pouvant s'écrire $\pi_d(L' \cap K_d)$, où $L'$ appartient à $\mathbf{FO}^2[<](A_d^*)$.*

*Démonstration.* La preuve repose sur une transformation syntaxique des formules. On transfère l'information modulaire sur les prédicats de lettre grâce à la seconde composante de l'alphabet enrichi. On a alors besoin de se restreindre aux mots bien formés car il s'agit des mots où cette transformation est correcte. On identifie dans la preuve un ensemble de formules avec l'ensemble des langages qu'elles définissent. Soit $\psi(\vec{x})$ une formule $\mathbf{FO}^2[<, \mathrm{MOD}_d](A^*)$, où $\vec{x}$ est l'ensemble de ses variables libres. On définit de façon inductive une nouvelle formule $\overline{\psi}$ de $\mathbf{FO}^2[<]$ sur l'alphabet enrichi en utilisant les règles suivantes :

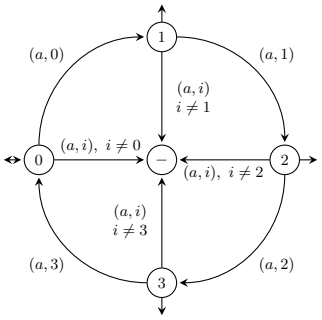
- $\overline{\exists y \psi(\vec{x})} = \exists y \overline{\psi(\vec{x})}$
- $\overline{(\neg \psi(\vec{x}))} = \neg(\overline{\psi(\vec{x})})$
- $\overline{\psi_1(\vec{x}) \wedge \psi_2(\vec{y})} = \overline{\psi_1(\vec{x})} \wedge \overline{\psi_2(\vec{y})}$
- $\overline{x < y} = x < y$
- $\overline{MOD_i^d(x)} = \bigvee_{a \in A} (\mathbf{a}, \mathbf{i})(x)$
- $\overline{\mathbf{a}(x)} = \bigvee_{0 \leqslant i < d} (\mathbf{a}, \mathbf{i})(x)$
- $\overline{D_i^d} = \exists x (\forall y, y \leqslant x \wedge \overline{MOD_i^d(x)})$

Cette traduction transforme effectivement une formule de $\mathbf{FO}^2[<, \mathrm{MOD}_d](A^*)$ en une formule équivalente de $\mathbf{FO}^2[<](A_d^*)$. Cette formule remplace les prédicats modulaires par des prédicats de lettre de façon à transférer l'information modulaire sur la

seconde composante des lettres. Les prédicats 0-aires de longueur sont remplacés par de courtes formules closes de $\mathbf{FO}^2[<]$ indiquant la congruence de la dernière lettre du mot. Maintenant, comparons les langages définis par $\psi$ et $\overline{\psi}$. Si un mot $u$ vérifie la formule $\psi$ alors $\overline{u}$ vérifie $\overline{\psi}$. De façon inverse, si un mot bien formé $v$ vérifie $\overline{\psi}$, alors $\pi_d(v)$ vérifie $\psi$. Ainsi, on a prouvé que $\mathbf{FO}^2[<, \mathrm{MOD}_d](A^*)$ est inclus dans l'ensemble des langages pouvant s'écrire $\pi_d(L' \cap K_d)$, où $L'$ appartient à $\mathbf{FO}^2[<](A_d^*)$.

Pour l'inclusion inverse, on définit une transformation inverse similaire, affectant seulement les prédicats de lettre, de sorte que l'on remplace les prédicats $(\mathbf{a}, \mathbf{i})(x)$ par $\mathbf{a}(x) \wedge \mathrm{MOD}_i^d(x)$. Cette transformation change une formule de $\mathbf{FO}^2[<]$ sur l'alphabet enrichi $A_d^*$ en une formule de $\mathbf{FO}^2[<, \mathrm{MOD}_d]$ sur l'alphabet $A^*$. De plus, il est facile de voir que si un mot bien formé vérifie la première formule, alors son projeté vérifiera la formule transformée. Réciproquement, si un mot de $A^*$ vérifie la seconde formule, alors son mot bien formé attaché vérifiera la première formule.
On a donc l'inclusion inverse, ce qui conclut la preuve. □

Étant donné son importance ici, nous étudions maintenant le langage des mots bien formés. Il s'agit de déterminer sa complexité algébrique.




| | 0 | 1 | 2 | 3 | – |
|---|---|---|---|---|---|
| $(a,0)$ | 1 | – | – | – | – |
| $(a,1)$ | – | 2 | – | – | – |
| $(a,2)$ | – | – | 3 | – | – |
| $(a,3)$ | – | – | – | 0 | – |

Figure 3.2: Automate minimal et table de transition de $K_d$ (pour $d = 4$).

On considère le semigroupe $B_d = (C_d \times C_d) \cup \{0\}$ où 0 est un zéro de $B_d$ et pour tout $(i, j)$ et $(k, \ell)$ dans $C_d \times C_d$, on a

$$(i,j)(k,\ell) = \begin{cases} (i,\ell) & \text{si } j = k \\ 0 & \text{sinon.} \end{cases}$$

Ce semigroupe est un semigroupe de Brandt (voir [How95]).

On rappelle maintenant que $\mathbf{J_1}$ est la variété des monoïdes idempotents et commutatifs. La proposition suivante donne une idée de la complexité de $K_d$.

**Proposition 3.7.** *L'ensemble des mots bien formés $K_d$ est reconnu par un timbre de* $\mathbf{QJ_1}$.

*Démonstration.* En calculant le timbre syntaxique de $K_d$ depuis son automate minimal (cf Figure 3.2), on obtient le timbre $\eta : A_d^* \to B_d^1$ défini pour toute lettre $(a, i)$ par $\eta(a, i) = (i, i + 1)$. L'idée ici est que si, dans un mot, il y a deux lettres successives dont les secondes composantes ne sont pas successives modulo $d$, le mot s'enverra sur 0. Dans le cas contraire où le mot est bien formé, à un décalage initial près, il s'envoie sur l'élément $(i, j)$ tel que $i$ est la congruence de la première lettre du mot et $j$ la seconde composante de la dernière lettre à laquelle on ajoute 1.
On cherche maintenant à calculer son semigroupe stable. On veut donc le plus petit entier $k$ tel que $\eta(A_d^k) = \eta(A_d^{2k})$. L'image d'un mot bien formé de longueur $k$ est de la forme $(i, i+k \bmod d)$. L'image de la composition de deux mots de longueur $k$ étant de la forme $(j, j+2k)$, l'indice de stabilité est donc le plus petit entier $k$ non nul tel que $k \equiv 2k \bmod d$. On obtient donc $k = d$ et le semigroupe stable de $K_d$ est $\eta(A_d^d) = (\{0\} \times \mathbb{Z}/d\mathbb{Z}) \cup \{0\}$. La loi interne de ce semigroupe est donnée par $xx = x$ et $xy = 0$ si $x \neq y$. Il s'agit bien d'un semigroupe idempotent et commutatif et donc $\eta$ appartient à $\mathbf{QJ_1}$. □

Cette complexité nous donne un exemple de pouvoir d'expression nécessaire pour définir les langages bien formés. Puisque $\mathbf{J_1}$ est inclus dans $\mathbf{DA}$ et qu'une variété est close par produit, on obtient le corollaire suivant, qui est un bon exemple de l'utilité de cette complexité.

**Corollaire 3.8.** *Soit $L$ un langage de $\mathcal{DA}(A_d^*)$. Alors le langage $L \cap K_d$ appartient à $\mathcal{QDA}(A_d^*)$.*

Nous sommes maintenant prêt à prouver la première inclusion $\mathbf{FO}^2[<, \mathrm{MOD}] \subseteq \mathbf{QDA}$. La preuve donnée repose plus sur les automates que sur les propriétés de $\mathbf{DA}$. Elle peut être adaptée pour convenir notamment à tout fragments plus expressif que $\mathbf{FO}^2[<]$, où la Propriété 3.6 et le Corollaire 3.8 restent vrais.

**Théorème 3.9.** *Le timbre syntaxique d'un langage $\mathbf{FO}^2[<, \mathrm{MOD}]$-définissable appartient à* $\mathbf{QDA}$.

*Démonstration.* Soit $A$ un alphabet et $L$ un langage régulier définissable dans $\mathbf{FO}^2[<, \mathrm{MOD}](A^*)$. Alors le Lemme 3.1 stipule que l'on peut restreindre les prédicats modulaires à une seule congruence $d$ telle que $L$ est définissable dans $\mathbf{FO}^2[<, \mathrm{MOD}_d](A^*)$. Quitte à augmenter $d$, on peut également considérer qu'il s'agit d'un multiple de l'indice de stabilité de $L$. En utilisant la Proposition 3.6, on sait qu'il existe une formule $\varphi$ de $\mathbf{FO}^2[<](A_d^*)$ telle que $L = \pi_d(L')$ où $L' = L(\varphi) \cap K_d$. De plus, puisque $d$ est un multiple de l'indice de stabilité de $L$ et que $L = \pi_d(L')$, il est également multiple de l'indice de stabilité de $L'$. Puisque $\mathbf{FO}^2[<] = \mathbf{DA}$ (voir [TW99]), on sait, grâce au Lemme 3.8, que le langage $L'$ est dans la variété de langages $\mathcal{QDA}(A_d^*)$. Soit maintenant $\mathcal{A}' = (Q, A_d, \cdot, i, F)$ l'automate déterministe minimal et émondé de $L'$. Notons que puisque

$\pi_d$ est une bijection sur les mots bien formés, l'automate $\pi_d(\mathcal{A}')$ obtenu en supprimant la seconde composante sur les transitions de $\mathcal{A}'$, reconnait $L$. Étant donné que $\mathcal{A}'$ est émondé et ne reconnait que des mots bien formés, toutes les transitions sortant d'un état donné possèdent la même seconde composante. En effet, si cela n'était pas le cas, le fait que l'automate soit émondé nous dit que l'automate reconnaitrait alors deux mots ayant une lettre à la même position mais ayant une seconde composante différente, ce qui est impossible puisque $L'$ est un langage de mots bien formés. Alors pour $0 \leqslant i < d$, on définit

$$Q_i = \{q \in Q \mid \text{ il existe } a \in A \text{ tel que } q \cdot (a, i) \text{ soit défini}\}$$

et soit $Q_E$ l'ensemble des états n'ayant aucune transition de sortie. Les ensembles $Q_i$ $(0 \leqslant i < d)$ forment avec $Q_E$ une partition de l'ensemble $Q$.

Observons maintenant qu'une transition lie forcément un état de $Q_i$ avec un état de $Q_{i+1 \bmod d} \cup Q_E$. En étendant ceci aux mots de longueur $d$, on remarque que la fonction de transition du $d$-automate $\mathcal{A}'_d$ est comprise dans $\bigcup_{0 \leqslant i < d} \big(Q_i \times A_d^d \times (Q_i \cup Q_E)\big)$. Ceci signifie que chaque ensemble $Q_i$ est stable dans le $d$-automate et induit un sous-automate indépendant des autres. La fonction de transition de ce sous-automate induit alors un monoïde $M_i$ qui est un sous-monoïde du monoïde syntaxique de $L'$. Maintenant, si l'on reprend le $d$-automate projeté $\pi_d(\mathcal{A}')_d$, on peut constater que l'action d'un mot $u$ de $A^d$ sur l'ensemble $Q_i$ est l'action du mot $(u_0, i) \cdots (u_d, i-1)$ sur $Q_i$ dans l'automate $\mathcal{A}'_d$, et est par conséquent décrit dans $M_i$. La Figure 3.3 montre les différentes constructions et leurs liens et la Figure 3.4 illustre ces constructions sur un exemple.

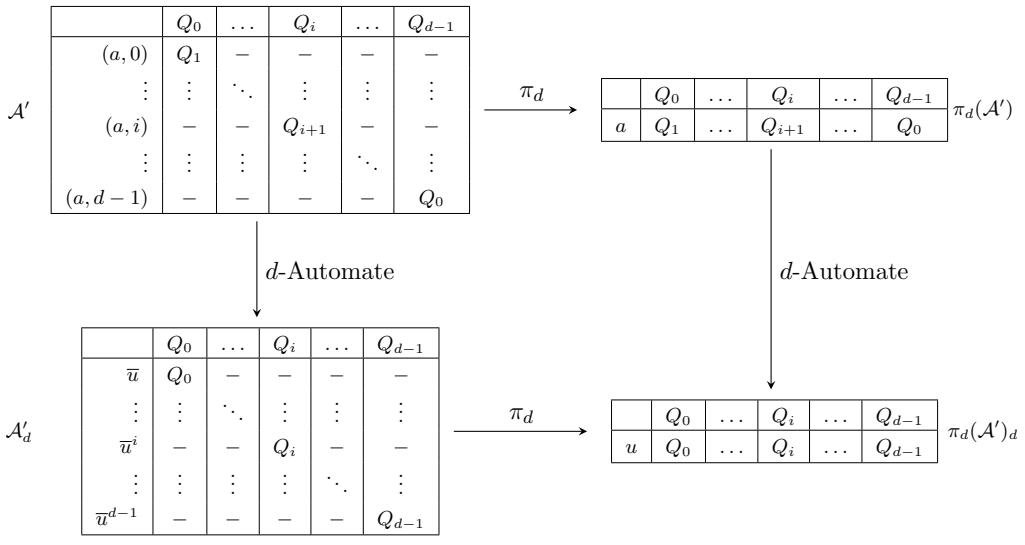



Figure 3.3: Monoïdes de transitions

Alors l'action complète de $u$ sur $Q$ est décrite complètement grâce à chacun des monoïdes $M_i$. Ainsi le monoïde de transition de $\pi_d(\mathcal{A}')_d$ est un sous-monoïde du monoïde

produit $\prod_{i=0}^{d} M_i$. Enfin, puisque $d$ est un indice de stabilité pour $L'$, le monoïde de transition du $d$-automate $\mathcal{A}'_d$ est dans **DA**, et puisque **DA** est une variété, elle est close par sous-monoïde et par produit, donc chaque monoïde $M_i$ est dans **DA** ainsi que $\prod_{i=0}^{d} M_i$ et c'est également le cas pour le monoïde de transition de $\pi_d(\mathcal{A}')_d$.
On peut maintenant conclure puisque $L$ est reconnu par $\pi_d(\mathcal{A}')$, un automate dont la $d$ itération a son monoïde de transition dans **DA**. □

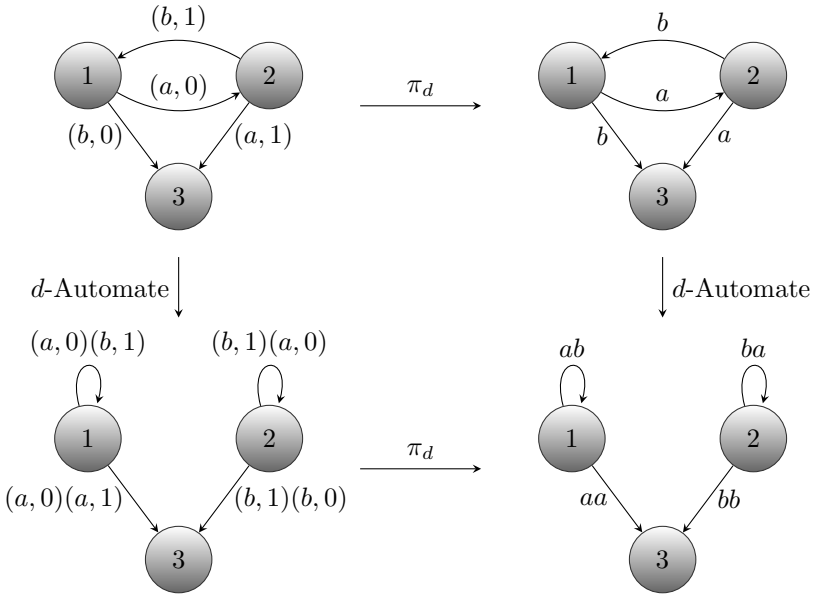



Figure 3.4: Ici, $d = 2$, $Q_0 = \{1\}$, $Q_1 = \{2\}$ et $Q_E = \{3\}$.

### 3.2.4 L'inclusion réciproque QDA $\subseteq \mathbf{FO}^2[<, \mathrm{MOD}]$

**Les jeux d'Ehrenfeucht-Fraïssé**

Nous commençons cette section par des rappels sur les jeux d'Ehrenfeucht-Fraïssé et plus particulièrement les jeux d'Ehrenfeucht-Fraïssé correspondant au fragment $\mathbf{FO}^2[<$ ]. Nous renvoyons à [Imm82] pour des définitions complètes. Les jeux d'Ehrenfeucht-Fraïssé, introduits par Roland Fraïssé et formalisés en tant que jeux par Andrzej Ehrenfeucht, servent à tester l'équivalence de deux structures. Dans notre cas, les jeux d'Ehrenfeucht-Fraïssé sont joués par deux joueurs, Spoiler et Duplicator, sur deux mots. Le but de Spoiler est d'expliciter une différence entre les deux mots alors que Duplicator essaye de préserver l'équivalence.
Formellement, on considérera le jeu suivant, adapté à notre fragment de logique.

**Définition 3.5: Jeux d'Ehrenfeucht-Fraïssé à 2 jetons**
Soient deux mots et un nombre de tours fixé. Chaque joueur possède deux jetons marqués qui sont initialement placés en dehors des mots. À chaque tour, Spoiler choisit un des jetons sur un des mots et le déplace. Duplicator répond en déplaçant le jeton correspondant sur l'autre mot.
Spoiler gagne le jeu si, à la fin du tour, les deux paires de jetons ne décrivent pas la même structure, c'est-à-dire que les ensembles de positions marquées par les jetons des deux mots n'induisent pas le même sous-mot. Duplicator gagne si Spoiler n'arrive pas à gagner avant la fin du dernier tour.

**Exemple 3.2.** À titre d'exemple, nous donnons en Figure 3.5 le schéma suivant illustrant une stratégie gagnante de Spoiler sur le jeu à 2 tours sur les mots *ababa* et *babab*. Les jetons sont marqués □ et ◯. Spoiler choisit de jouer sur le mot *ababa* et Duplicator doit ainsi répondre sur *babab*. La stratégie de Spoiler se traduit vers la formule $\exists x\ \mathbf{a}(x) \wedge \big(\exists y\ x < y \wedge \mathbf{a}(y)\big) \wedge \big(\exists y\ y < x \wedge \mathbf{a}(y)\big)$, satisfaite par *ababa* mais pas par *babab*.

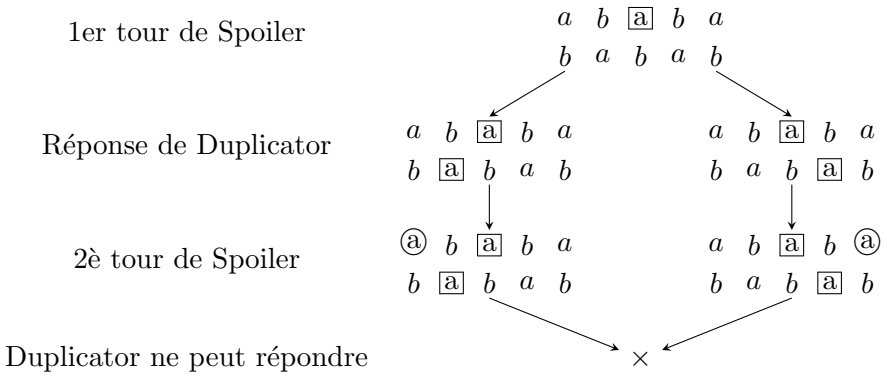



Figure 3.5

Le résultat suivant, originalement prouvé pour tout nombre de variables par Immerman [Imm82], explicite le lien entre $\mathbf{FO}^2[<]$ et ces jeux :

**Proposition 3.10** ([Imm82])**.** *Deux mots sur un même alphabet sont séparés par une formule de* $\mathbf{FO}^2[<]$ *de profondeur de quantificateur n si, et seulement si, Spoiler a une stratégie gagnante pour le jeu d'Ehrenfeucht-Fraïssé à deux jetons et n tours.*

L'idée de la preuve, omise ici, est que les deux jetons symbolisent les deux variables autorisées. Spoiler est autorisé à bouger un des jetons et Duplicator doit répondre en conservant les valeurs de vérité des deux variables sur les prédicats, soit l'étiquetage des lettres ainsi que l'ordre entre les deux variables. Ainsi chaque tour décrit une quantification nouvelle et une instance du jeu décrit ainsi une formule que l'on peut construire,

et la stratégie gagnante de Spoiler nous donne une formule séparant les deux structures considérées.
On veut maintenant avoir une version de ces jeux pour notre fragment $\mathbf{FO}^2[<, \mathrm{MOD}]$. La Proposition 3.6 nous donne ce que nous voulons :

**Proposition 3.11.** *Soient $u$ et $v$ deux mots de $A^*$ et $n \geqslant 2$. Alors il existe une formule de $\mathbf{FO}^2[<, \mathrm{MOD}_d]$ séparant $u$ et $v$ et de profondeur de quantificateurs $n$ si, et seulement si, Spoiler gagne le jeu d'Ehrenfeucht-Fraïssé à $n$ tours et 2 jetons sur la paire bien formée $(\overline{u}, \overline{v})$.*

*Démonstration.* La preuve de cette proposition vient du fait que s'il existe telle formule de $\mathbf{FO}^2[<, \mathrm{MOD}]$ séparant nos deux mots, alors par le Lemme 3.1 il existe un entier $d$ tel qu'il existe une formule de $\mathbf{FO}^2[<, \mathrm{MOD}_d]$ les séparant également. De plus la nouvelle formule n'augmente pas la profondeur de quantificateur (voir la preuve du lemme). Ainsi, par la Proposition 3.6, il existe une formule de $\mathbf{FO}^2[<]$ sur l'alphabet enrichi de même profondeur de quantificateur, qui sépare les mots bien formés $\overline{u}$ et $\overline{v}$. On doit tout de même remarquer que la traduction donnée dans la preuve de la Proposition 3.6 augmente la profondeur de quantificateur de 2 lorsque l'on rencontre un prédicat de longueur. Cependant, ces prédicats peuvent être placés à profondeur zéro, et on obtient ainsi le résultat pour $n \geqslant 2$. □

### Congruences et opérations syntaxiques sur $\mathbf{FO}^2[<, \mathrm{MOD}]$

Nous abordons maintenant la preuve de la seconde inclusion. Nous reprenons tout d'abord certaines définitions issues de [TW99], en particulier les congruences. Nous montrerons après comment ces définitions peuvent être étendues pour prendre en compte l'information modulaire.

**Définition 3.6:**
Soit un mot $u$ sur l'alphabet $A^*$, et soit $a \in A$ une lettre apparaissant dans $u$. On appelle *$a$-décomposition gauche* de $u$ l'unique triplet $(u_0, a, u_1)$ tel que $u = u_0 a u_1$ et $u_0$ ne contient aucun $a$. On définit également la *$a$-décomposition droite* d'une façon symétrique.

On donne maintenant la définition de la congruence $\equiv_n$ sur $A^*$ provenant de [TW99]. Cette congruence va servir d'intermédiaire entre **QDA** et $\mathbf{FO}^2[<, \mathrm{MOD}]$. L'idée est de prouver que pour tout timbre de **QDA**, il existe un entier $n$ tel que notre congruence est plus fine que celle induite par le timbre donné. De l'autre coté, on va prouver que les classes d'équivalence de notre congruence peuvent être séparées par des formules de $\mathbf{FO}^2[<, \mathrm{MOD}]$. Ainsi, on pourra séparer les classes d'équivalence des timbres de **QDA** par des formules de $\mathbf{FO}^2[<, \mathrm{MOD}]$ et ainsi conclure. On rappelle que la fonction $\alpha$ donne l'alphabet d'un mot, i.e. l'ensemble des lettres apparaissant dans ce mot.

**Définition 3.7:**
[TW99] Soient $u, v \in A^*$ deux mots. Alors on a toujours $u \equiv_0 v$.
Pour $n > 0$, on a $u \equiv_n v$ si, et seulement si, les conditions suivantes sont satisfaites :

1. $\alpha(u) = \alpha(v)$, ce qui signifie que les deux mots ont le même alphabet,
2. Pour chaque $a$ apparaissant dans $u$, si $(u_0, a, u_1)$ est la $a$-décomposition gauche de $u$ et $(v_0, a, v_1)$ celle de $v$, alors on a $u_0 \equiv_n v_0$ et $u_1 \equiv_{n-1} v_1$,
3. Pour chaque $a$ apparaissant dans $u$, si $(u_0, a, u_1)$ est la $a$-décomposition droite de $u$ et $(v_0, a, v_1)$ celle de $v$, alors $u_0 \equiv_{n-1} v_0$ et $u_1 \equiv_n v_1$.

Les définitions données sont inductives et leur terminaison doit être prouvée. Supposons que $u \equiv_n v$ pour deux mots $u$ et $v$, et $n$ un entier strictement positif. Si on considère maintenant le paramètre $|\alpha(u)| + n$, la première condition indique qu'il est égal à $|\alpha(v)| + n$. Alors pour chaque décomposition gauche ou droite de chaque itération, le paramètre décroit, soit en réduisant $n$, soit en réduisant la taille de l'alphabet par définition des $a$-décompositions.

**Proposition 3.12.** *[TW99] La relation $\equiv_n$ est une congruence.*

Ces définitions doivent maintenant être étendues pour prendre en compte l'information modulaire. Ceci est fait via l'alphabet enrichi et les mots bien formés.

On définit la relation $\equiv_n^d$ en posant $u \equiv_n^d v$ si, et seulement si, $\overline{u} \equiv_n \overline{v}$. Nous donnons maintenant quelques propriétés de notre relation qui se révéleront utiles dans la preuve.

**Lemme 3.13.** *Soient deux entiers strictement positifs $n$ et $d$, et $u$ et $v$ deux mots tels que $u \equiv_n^d v$. Alors les propriétés suivantes sont satisfaites :*

1. *si $u$ est le mot vide, alors $v$ l'est également,*
2. *$|u| \equiv |v| \bmod d$,*
3. *si $u = u_0 a u_1$, $v = v_0 b v_1$ avec $|u_0| < d$, $|v_0| < d$ et $|au_1| \equiv |bv_1| \bmod d$, alors $a = b$, $u_0 = v_0$ et $u_1 \equiv_{n-1}^d v_1$,*
4. *si $u = u_0 a u_1$, $v = v_0 b v_1$ avec $|au_1| < d$, $|bv_1| < d$ et $|u_0| \equiv |v_0| \bmod d$, alors $a = b$, $u_1 = v_1$ et $u_0 \equiv_{n-1}^d v_0$,*
5. *la relation $\equiv_n^d$ est une congruence.*

*Démonstration.* Nous allons prouver et expliquer point par point ce lemme.

1. Le mot vide est seul dans sa classe. En effet, il s'agit du seul mot avec un alphabet vide.
2. Deux mots équivalents ont la même longueur modulo $d$. Pour constater cela, il suffit de considérer la décomposition droite selon la dernière lettre de $\overline{u}$. Cette lettre doit apparaître dans $v$, et le suffixe suivant cette lettre doit être équivalent selon $n$ et $d$ au mot vide. On conclut grâce au premier point.

3. Ce point indique que deux mots équivalents ont le même préfixe de longueur $d$. On prouve cette affirmation en considérant la décomposition gauche sur $\overline{u}$. Alors le préfixe de $u$ doit être en relation selon $n$ et $d$ au préfixe de $v$. Or le préfixe de $u$ ne contient pas de lettre dont la congruence est supérieure à $|u_0|$, et une unique lettre pour les autres congruences. Ces lettres doivent donc apparaître dans $v_0$ et leur congruence impose l'égalité des deux préfixes.
4. Cette affirmation est le symétrique du point précédent et est prouvée de la même façon.
5. Ce point est prouvé directement en utilisant la proposition précédente.

□

On désire maintenant relier notre congruence à notre logique. De la même façon que dans [TW99], ceci est fait grâce aux jeux d'Ehrenfeucht-Fraïssé (cf. Proposition 3.11).

**Théorème 3.14.** *Soient $u$ et $v$ deux mots de $A^*$. Si $u \not\equiv_n^d v$ alors il existe une formule de $\mathbf{FO}^2[<, \mathrm{MOD}_d]$ de profondeur de quantificateur au plus $n + |\alpha(\overline{u})|$ qui sépare $u$ et $v$.*

*Démonstration.* Nous prouvons le résultat en utilisant les jeux d'Ehrenfeucht-Fraïssé, et utilisons la Proposition 3.11 afin de conclure. Pour se faire, nous utilisons une récurrence sur $k = n + |\alpha(\overline{u})|$, en supposant que Spoiler a une stratégie gagnante.
Tout d'abord, si $n + |\alpha(\overline{u})| = 0$, alors puisque $n = 0$ toutes les paires de mots sont équivalentes.
Considérons le théorème vrai jusqu'à l'entier $k$, et soient $u$ et $v$ tels que $u \not\equiv_n^d v$ avec $n + |\alpha(\overline{u})| = k + 1$.
Par définition, si $u \not\equiv_n^d v$, alors au moins une des trois conditions suivantes est vérifiée.

1. $\alpha(\overline{u}) \neq \alpha(\overline{v})$. Il s'agit ici du cas simple. Cette inégalité signifie qu'il existe une lettre de $\overline{u}$ ou $\overline{v}$ qui n'apparait pas dans le second mot. En jouant sur cette position, Spoiler gagne alors le jeu.
2. Il existe une lettre $(a, i)$ de $\overline{u}$ telle que si $(u_0, (a, i), u_1)$ est la décomposition gauche de $\overline{u}$ selon $(a, i)$ et $(v_0, (a, i), v_1)$ celle de $\overline{v}$, avec deux sous cas possibles.
   – $u_0 \not\equiv_n^d v_0$. Dans ce cas, puisque la lettre $(a, i)$ n'apparaît pas dans $u_0$, et que par hypothèse de récurrence Spoiler gagne le jeu d'Ehrenfeucht-Fraïssé sur la paire $(u_0, v_0)$ en au plus $n + |\alpha(\overline{u})| - 1$ tours, la stratégie de Spoiler est de jouer sur $(u_0, v_0)$ en suivant sa stratégie gagnante. En au plus $n + |\alpha(\overline{u})| - 1$ tours, Duplicator est alors forcé soit de perdre, soit de jouer en dehors de cette paire. Notons qu'il ne peut pas jouer sur la lettre $(a, i)$ sans perdre immédiatement, puisque ni $u_0$ ni $v_0$ ne contiennent cette lettre. Sans perdre de généralité, supposons que Duplicator joue sur $u_1$. Alors Spoiler gagne en jouant l'autre jeton de $u$ sur la lettre marquée $(a, i)$. Duplicator ne peut alors répondre, puisqu'il doit répondre sur $v_0$ qui ne contient pas cette lettre.

– La seconde possibilité est que $u_1 \not\equiv^d_{n-1} v_1$. Spoiler a alors une stratégie gagnante sur ces mots en au plus $n - 1 + |\alpha(\overline{u})|$ tours par hypothèse de récurrence. La stratégie gagnante de Spoiler sur $(u, v)$ est alors de suivre celle sur la paire $(u_1, v_1)$. À un moment, Duplicator perd le jeu ou est forcé de jouer sur la paire $(u_0a, v_0a)$. Si Duplicator pose un jeton sur $u_0a$, alors Spoiler gagne en plaçant le second jeton de $v$ sur la lettre marquée $(a, i)$, Duplicator ne pouvant alors répondre tout en préservant l'ordre sur les deux paires de jetons.

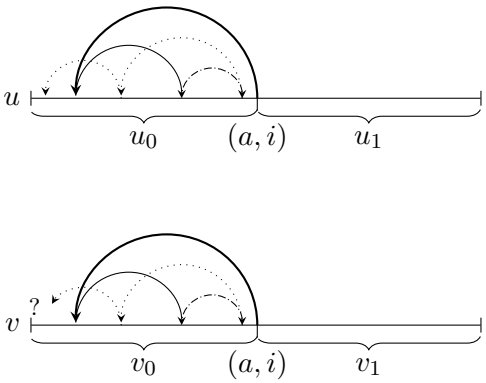



Figure 3.6: En jouant sur $(u_0, v_0)$, Spoiler a une stratégie gagnante.

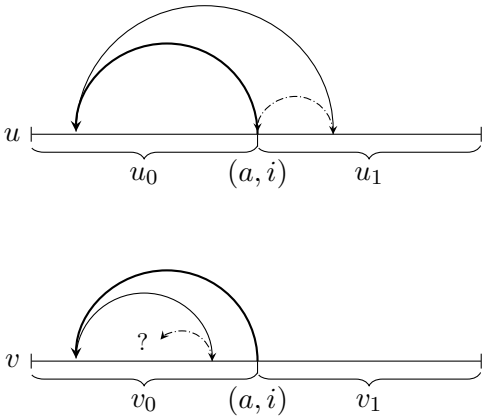



Figure 3.7: Si Duplicator répond en dehors de $u_0$, alors Spoiler gagne le jeu.

3. La dernière possibilité est que la non-équivalence provienne d'une des décompositions droites. Des arguments symétriques au dernier point permettent alors d'arriver aux mêmes conclusions.

□

On a ainsi prouvé que les classes d'équivalence de notre congruence peuvent être séparées par des formules de $\mathbf{FO}^2[<, \mathrm{MOD}]$. On va maintenant s'intéresser au lien entre notre congruence et les timbres de **QDA**. Ceci est l'objet de la prochaine partie.

### Congruence et opérations algébriques sur QDA

Une des caractérisations connues de la variété **DA** repose sur les relations de Green : les $\mathcal{J}$-classes régulières forment des semigroupes apériodiques. Cependant on considère ici la variété de timbres **QDA**. On définit ici une version des pré-ordres de Green adaptée au semigroupe stable. On ne va donc garder des relations de Green que leurs traces sur le semigroupe stable. Formellement, on a :

Soit un timbre $h : A^* \to M$ et $S$ son semigroupe stable. Pour tous éléments $x$ et $y$ dans $M$ on définit :

– $x \leqslant_{\mathcal{R}_{st}} y$ si, et seulement si, $xM \cap S \subseteq yM \cap S$
– $x \leqslant_{\mathcal{L}_{st}} y$ si, et seulement si, $Mx \cap S \subseteq My \cap S$
– $x \leqslant_{\mathcal{H}_{st}} y$ si, et seulement si, $x \leqslant_{\mathcal{R}_{st}} y$ et $x \leqslant_{\mathcal{L}_{st}} y$.

Ces définitions s'étendent naturellement aux relations de Green. On obtient une version des relations de Green adaptée au semigroupe stable.

- $x \mathcal{R}_{st} y$ si, et seulement si, $x \leqslant_{\mathcal{R}_{st}} y$ et $y \leqslant_{\mathcal{R}_{st}} x$
- $x \mathcal{L}_{st} y$ si, et seulement si, $x \leqslant_{\mathcal{L}_{st}} y$ et $y \leqslant_{\mathcal{L}_{st}} x$
- $x \mathcal{H}_{st} y$ si, et seulement si, $x \leqslant_{\mathcal{H}_{st}} y$ et $y \leqslant_{\mathcal{H}_{st}} x$

On désire maintenant que les éléments du semigroupe stable ne soient atteints que par des mots d'une longueur multiple de l'indice de stabilité. Ceci permet d'éviter des effets de bords au sein de nos relations de Green modifiées. On dira qu'un timbre est *fidèle à la longueur* si $h^{-1}(S^1) = (A^d)^*$. Cette notion se révèle nécessaire dans le prochain lemme. Elle n'induit cependant aucune perte de généralité, comme le prouvera le Corollaire 3.20.

**Lemme 3.15.** *Soit un timbre $h : A^* \to M$ et soit $S$ son semigroupe stable. Si $h$ est fidèle à la longueur, alors la restrictions de $\leqslant_{\mathcal{R}_{st}}$ (resp. $\leqslant_{\mathcal{L}_{st}}$) à $S$ correspond à la relation de Green classique $\leqslant_{\mathcal{R}}$ (resp. $\leqslant_{\mathcal{L}}$) sur le semigroupe $S$.*

*Démonstration.* Soit un élément $x$ de $S$, et $y$ un élément de $M$ tel que $xy$ appartient à $S$. Alors, puisque $h$ est fidèle à la longueur, $h^{-1}(xy)$ est contenu dans $(A^d)^*$. De plus, comme $x$ appartient à $S$, nous avons également $h^{-1}(x) \subseteq (A^d)^*$. Ainsi pour tout mot $u$ tel que $h(u) = x$, et tout mot $v$ tel que $h(v) = y$, on a $|u| \equiv |uv| \equiv 0 \bmod d$, donc $|v| \equiv 0 \bmod d$. Par conséquent, $y$ appartient à $S$.
On a ainsi prouvé que pour tout $x$ appartenant à $S$, on a $xM \cap S = xS$, et par conséquent pour tout éléments $x$ et $y$ dans $S$, $x \leqslant_{\mathcal{R}_{st}} y$ si, et seulement si, $x \leqslant_{\mathcal{R}} y$ où $\mathcal{R}$ désigne la relation de Green classique sur $S$.
La preuve pour la relation $\leqslant_{\mathcal{L}_{st}}$ s'obtient de façon similaire avec des arguments symétriques. □

Ce dernier lemme nous permet de transférer certaines propriétés de la variété **DA** vers le semigroupe stable des timbre de la $lm$-variété **QDA**.

**Corollaire 3.16.** *Soit un timbre fidèle à la longueur $h : A^* \to M$ appartenant à* **QDA**. *Alors, la restrictions des $\mathcal{H}_{st}$-classes à $S$ est triviale.*

*Démonstration.* Puisque le timbre $h$ est fidèle à la longueur, le lemme précédent nous dit que la restriction des $\mathcal{R}_{st}$ et $\mathcal{L}_{st}$-classes à $S$ correspond aux $\mathcal{R}$ et $\mathcal{L}$-classes de $S$. Par conséquent, cette restriction s'étend par définition aux $\mathcal{H}_{st}$-classes de $S$. Étant donné que $S$ appartient à **DA**, il s'agit d'un semigroupe apériodique donc ses $\mathcal{H}$-classes sont triviales. Alors les $\mathcal{H}_{st}$-classes de $S$ le sont également. □

Pour établir le lien avec la congruence $\equiv_n^d$ définie à la section précédente, on définit la *$\mathcal{R}_{st}$-décomposition*. Il s'agit d'une définition originellement trouvée dans [TW99], adaptée ici aussi au semigroupe stable.

**Définition 3.8:**
Soit un mot $u$ et un timbre $h : A^* \to M$. On définit la $\mathcal{R}_{st}$-décomposition de $u$

comme le multiplet $(u_0, a_1, u_1, \ldots, a_s, u_s)$ tel que $u = u_0a_1u_1 \cdots a_su_s$ et tel que les conditions suivantes soient vérifiées :

1. $|u_0a_1u_1 \cdots a_iu_i| \equiv 0 \bmod d$ pour tout $0 \leqslant i < s$,
2. $h(u_0a_1u_1 \cdots u_{i-1}a_i) >_{\mathcal{R}_{st}} h(u_0 \cdots u_ia_{i+1})$,
3. Pour tout préfixe $v$ de $u_i$ d'une longueur multiple de $d$, $h(u_0 \cdots u_{i-1}a_i)\,\mathcal{R}_{st}\,h(u_0 \cdots a_iv)$,
4. Pour tout préfixe $v$ et $v'$ de $u_0$ de longueur multiple de $d$, on a $h(v)\,\mathcal{R}_{st}\,h(v')$.

L'idée derrière cette décomposition est de marquer les positions "utiles" aux $\mathcal{R}_{st}$-classes. Ces positions sont les premières positions multiples de $d$ après avoir changé de $\mathcal{R}_{st}$-classes. Si l'on considère le fait qu'un préfixe est toujours $\mathcal{R}_{st}$ supérieur au mot dont il est préfixe, alors chaque position marquée désigne un moment où l'on descend dans l'ordre $\mathcal{R}_{st}$. On peut ainsi borner le nombre de positions marquées dans une $\mathcal{R}_{st}$-décomposition, et ceci indépendamment du mot considéré.
Les deux prochains lemmes servent à relier la $\mathcal{R}_{st}$-décomposition des timbres de **QDA** à notre congruence $\equiv_n^d$.

**Lemme 3.17.** *Soit un timbre fidèle à la longueur $h : A^* \to M$ dans* **QDA**, *et $S$ son semigroupe stable. Soit maintenant $u$ dans $S$ et $a$ et $x$ deux éléments de $M$. Si $ax$ appartient à $S$, alors $uax\,\mathcal{R}_{st}\,u$ implique $uaxa\,\mathcal{R}_{st}\,u$.*

L'idée ici est que si l'apparition d'une lettre à une congruence donnée ne nous a pas fait chuter dans les $\mathcal{R}_{st}$-classes, alors ce ne sera pas le cas si elle apparaît une seconde fois.

*Démonstration.* Si les éléments $u$ et $uax$ sont $\mathcal{R}_{st}$-équivalent et si $h$ est fidèle à la longueur, alors grâce au Lemme 3.15, il existe un élément $t$ de $S$ tel que $u = uaxt$. En itérant ceci, on obtient $u = u(axt)^\omega$. Mais on sait que $S$ appartient à **DA**, et $S$ satisfait donc l'équation $(xy)^\omega x(xy)^\omega = (xy)^\omega$. On obtient donc $(axt)^\omega ax(axt)^\omega = (axt)^\omega$ et par conséquent $u = u(axt)^\omega = u(axt)^\omega ax(axt)^\omega = uax(axt)^\omega$. En réécrivant cette dernière équation, on obtient $u = uaxa(xt(axt)^{\omega-1})$. Finalement, on a $u \in uaxaM \cap S$, et donc $u\,\mathcal{R}_{st}\,uaxa$. □

**Corollaire 3.18.** *Soit un timbre fidèle à la longueur $h : A^* \to M$ qui appartient à* **QDA** *et $u$ un mot. Alors si $(u_0, a_1, u_1, \ldots, a_s, u_s)$ est la $\mathcal{R}_{st}$-décomposition de $u$ alors $(a_{i+1}, 0) \notin \alpha(\overline{a_iu_i})$ pour tout $i < s$.*

*Démonstration.* Soit $(u_0, a_1, u_1, \ldots, a_s, u_s)$ la $\mathcal{R}_{st}$-décomposition de $u$. Supposons qu'il existe un entier $i$ tel que $(a_{i+1}, 0) \in \alpha(\overline{a_iu_i})$ avec $i < s$. Alors, grâce au lemme précédent, $h(a_iu_ia_{i+1})\,\mathcal{R}_{st}\,h(a_iu_i)$ ce qui est en contradiction avec la définition de la $\mathcal{R}_{st}$-décomposition de $u$. □

Ces outils étaient les derniers outils nécessaires pour prouver le théorème suivant, qui relie la congruence $\equiv_n^d$ aux timbres de **QDA**.

**Théorème 3.19.** *Soit un timbre fidèle à la longueur $h : A^* \to M$ appartenant à* **QDA** *et soit $d$ son indice de stabilité. Il existe alors un entier $n$ tel que pour tous mots $u$ et $v$, $u \equiv_n^d v$ implique $h(u) = h(v)$.*

*Démonstration.* Grâce au Lemme 3.13, si deux mots sont équivalents selon la congruence $\equiv_{n+1}^d$, alors leurs suffixes de taille inférieure à $d$ sont égaux et les préfixes correspondants sont équivalents selon la congruence $\equiv_n^d$. On peut ainsi se limiter à prouver le résultat pour les mots de longueur multiple de $d$ et un entier $n$, et le résultat général sera vrai pour $n+1$.
Soient $u$ et $v$ deux mots de longueur multiple de $d$, et $n$ un entier supérieur à $|\alpha(\overline{u})| \cdot |S|$ tels que $u \equiv_n^d v$. Nous allons prouver par récurrence sur $|\alpha(\overline{u})|$ que nous avons l'égalité $h(u) = h(v)$.
Si $|\alpha(\overline{u})| = 0$, alors $h(u) = h(v) = 1$.

Considérons maintenant le résultat vrai jusqu'au rang $k-1$ et soit un mot $u$ tel que $|\alpha(\overline{u})| = k$. Soit maintenant $(u_0, a_1, u_1, \ldots, a_\ell, u_\ell)$ la $\mathcal{R}_{st}$-décomposition de $u$. On remarque tout d'abord que $\ell$ est inférieur à la taille de $|S|$, puisque chaque $a_i$ induit un nouveau préfixe dont la $\mathcal{R}_{st}$-classe est inférieure à la précédente et que le nombre de telles classes est borné par la taille de $S$. En utilisant le corollaire précédent, on note que $(u_i, a_{i+1}, u_{i+1} \cdots u_\ell)$ est la décomposition gauche de $x_i = u_i \cdots u_\ell$ pour tous $i < \ell$. Puisque $u \equiv_n^d v$, il existe alors une décomposition similaire $(v_0, a_1, \ldots, a_\ell, v_\ell)$ pour $v$ telle que $a_i u_i \equiv_{n-i}^d a_i v_i$ où $(a_{i+1}, 0) \notin \alpha(\overline{a_i u_i})$ selon le Corollaire 3.18 et donc $|\alpha(\overline{a_i u_i})| \leqslant |\alpha(\overline{u})| - 1$. Puisque $i$ est inférieur à $\ell$, on a $n - i \geqslant (k-1)|S| \geqslant |\alpha(\overline{a_i u_i})||S|$. Par hypothèse d'induction, pour $i < \ell$, on a $h(a_i u_i) = h(a_i v_i)$. Et ainsi on obtient $h(u)\, \mathcal{R}_{st}\, h(u_1 \cdots a_\ell) = h(v_1 \cdots a_\ell) \geqslant_{\mathcal{R}_{st}} h(v)$. Par des arguments symétriques, on obtient également que $h(v) \geqslant_{\mathcal{R}_{st}} h(u)$ et donc $h(u)\, \mathcal{R}_{st}\, h(v)$.
De part la symétrie gauche/droite, on a de façon similaire que $h(v)\, \mathcal{L}_{st}\, h(u)$ et ainsi par définition $h(v)\, \mathcal{H}_{st}\, h(u)$. En utilisant le Corollaire 3.16, on sait que les $\mathcal{H}_{st}$-classes d'un timbre dans **QDA** sont triviales pour des mots de longueur multiple de $d$, ce qui prouve finalement que $h(u) = h(v)$. □

En compilant finalement toutes les données obtenues, on obtient le corollaire suivant, qui finit de prouver le Théorème 3.4.

**Corollaire 3.20.** $\mathbf{QDA} \subseteq \mathbf{FO}^2[<, \mathrm{MOD}]$

*Démonstration.* Soit un langage régulier $L$ dont le timbre syntaxique $\eta : A^* \to M$ appartient à **QDA** et soient l'entier $s$ et le semigroupe $S$ respectivement indice de stabilité et semigroupe stable. On remarque alors que le timbre $h : A^* \to M \times C_s$ défini, pour tout mot $u$, par $h(u) = (\eta(u), |u| \bmod s)$ est fidèle à la longueur. En effet, le semigroupe stable de $h$ est égal à $S \times \{0\}$ et $h^{-1}(S \times \{0\})$ est l'ensemble des mots de longueur multiple de $s$.
Selon le Théorème 3.19, il existe un entier $n$ tel que la congruence $\equiv_n^s$ soit plus fine que la congruence induite par le timbre $h$, qui est elle-même plus fine que celle du timbre $\eta$

caractérisant la congruence syntaxique du langage $L$. Par conséquent, $L$ est une union finie de classe de $\equiv_n^s$, chacune d'elle étant, selon le Théorème 3.14, définissable par une formule de $\mathbf{FO}^2[<, \mathrm{MOD}_s]$ de profondeur de quantificateurs inférieure à $n + |A|^s$. □

### 3.2.5 Autres caractérisations

Ayant obtenu la décidabilité du problème de décision pour notre fragment, nous nous tournons désormais vers d'autres formalismes afin d'obtenir d'autres caractérisations de $\mathbf{FO}^2[<, \mathrm{MOD}]$, de façon similaire à ce qui est connu, et recensé notamment dans [DGK08].
Ainsi par exemple, considérons le fragment $\mathbf{TL}[X_a, Y_a]$ de la *logique temporelle linéaire* dont les formules sont définies de manière inductive de la façon suivante :

$$\varphi \equiv \top \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid X_a\varphi \mid Y_a\varphi.$$

Les opérateurs unaires $X_a$, existant pour toute lettre $a$ de $A$, signifient ne**X**t $a$, c'est-à-dire que l'on saute jusqu'au prochain $a$, et les opérateurs $Y_a$ signifient symétriquement **Y**esterday $a$ et nous donnons maintenant la sémantique de ce fragment de la logique temporelle.

Formellement, une formule de ce fragment s'évalue sur un couple formé d'un mot ainsi qu'un entier. Étant donnés un mot $u$ et $i$ un entier tel que $i \leqslant |u|$, on a :

- $(u, i) \models \top$.
- $(u, i) \models X_a\varphi$ si, et seulement si, il existe une position $i < j < |u|$ telle que $u_j = a$, pour tout entier $k$ tel que $i < k < j$, $u_k \neq a$ et $(u, j) \models \varphi$.
- Symétriquement, $(u, i) \models Y_a\varphi$ si, et seulement si, il existe une position $0 \leqslant j < i$ telle que $u_j = a$, pour tout entier $k$ tel que $j < k < i$, $u_k \neq a$ et $(u, j) \models \varphi$.
- Afin de pouvoir évaluer un mot sur une formule, on considère que l'on lit la formule à partir d'une position hors du mot. Ainsi, on a $u \models X_a\varphi$ si $(u, i) \models \varphi$, où $i$ désigne la première position étiquetée par un $a$. Symétriquement, on a $u \models Y_a\varphi$ si $(u, i) \models \varphi$, où $i$ désigne la position de la dernière occurrence de $a$.
  En particulier, si $a$ n'apparait pas dans $u$, alors pour toute formule $\varphi$, le mot $u$ ne vérifie ni $X_a\varphi$, ni $Y_a\varphi$.
- Étant donné une formule $\varphi$ de $\mathbf{TL}[X_a, Y_a]$ , le langage défini par $\varphi$ est l'ensemble des mots satisfaisant $\varphi$. Formellement, on a $L(\varphi) = \{u \in A^* \mid u \models \varphi\}$.

Les opérateurs logiques $\wedge$, $\vee$ et $\neg$ s'interprètent de façon classique tels que définis dans le Chapitre 2.

Ce fragment de logique temporelle $\mathbf{TL}[X_a, Y_a]$ est connu [DGK08] comme ayant le même pouvoir d'expressivité que la variété **DA**. Il est donc naturel de regarder le fragment $\mathbf{TL}[X_a^{r \bmod d}, Y_a^{r \bmod d}]$, où on définit les prédicats $X_a^{r \bmod d}$ de la façon suivante. Pour un mot $u$ et une position $x$ de ce mot, on note $(u, x) \models X_a^{r \bmod d}\varphi$ si $\varphi$ est vrai à la prochaine position étiquetée par un $a$ et qui est congrue à $r$ modulo $d$. De la même façon que dans la Proposition 3.6, on peut transférer l'information modulaire depuis les prédicats vers les mots bien formés d'un alphabet enrichi.

**Proposition 3.21.** *Soit un entier strictement positif $d$ et un alphabet $A$. Alors, les langages définis par $\mathbf{TL}[X_a^{r \bmod d}, Y_a^{r \bmod d}](A^*)$ sont exactement les langages pouvant s'écrire $\pi_d(L' \cap K_d)$, où $L'$ appartient à $\mathbf{TL}[X_{(a,r \bmod d)}, Y_{(a,r \bmod d)}](A_d^*)$.*

En la reliant à la Proposition 3.6 et en utilisant l'égalité $\mathbf{FO}^2[<] = \mathbf{TL}[X_a, Y_a]$, on obtient ainsi l'égalité $\mathbf{TL}[X_a^{r \bmod d}, Y_a^{r \bmod d}] = \mathbf{FO}^2[<, \mathrm{MOD}]$, caractérisation en logique temporelle de notre fragment.

On veut maintenant obtenir une caractérisation de $\mathbf{FO}^2[<, \mathrm{MOD}]$ en terme de langage rationnels. Dans [Sch77], il a été prouvé que les langages dont le timbre syntaxique était dans la variété **DA** étaient exactement les polynômes non-ambigus (voir Exemple 2.3). On désire donc naturellement étendre ces définitions afin de prendre en compte l'information modulaire et obtenir une caractérisation de $\mathbf{FO}^2[<, \mathrm{MOD}]$ en termes de langages rationnels. On définit ainsi les *monômes modulaires* comme les langages s'écrivant

$$(A_{0,0} \cdots A_{d-1,0})^* a_1 (A_{0,1} \cdots A_{d-1,1})^* \cdots a_n (A_{0,n} \cdots A_{d-1,n})^*$$

où $d$ est un entier strictement positif, et les ensembles $A_{i,k}$ sont des sous-ensembles de $A$ et les $a_i$ sont des lettres. De manière similaire aux monômes définis précédemment, un monôme sera *non-ambigu* si tout mot se décompose de façon unique dans le monôme. Un *polynôme modulaire* est une union finie de monômes. Il est non-ambigu s'il s'agit d'une union disjointe de monômes non-ambigus. La proposition suivante justifie l'introduction de notre définition.

**Proposition 3.22.** *Un langage régulier $L$ est dans la $lm$-variété de langages $\mathcal{QDA}(A^*)$ si, et seulement si, $L$ est un polynôme modulaire non-ambigu.*

*Démonstration.* Le Théorème 3.4 et la Proposition 3.6 nous indiquent qu'un langage régulier $L$ est dans la $lm$-variété de langages $\mathcal{QDA}(A^*)$ si, et seulement si, il existe un entier $d$ tel que $L$ est la projection d'un langage de mots bien formés $L'$ appartenant à $\mathcal{DA}(A_d^*)$. Alors, selon [Sch77], $L'$ est union disjointe de monômes non-ambigus. Puisque la projection $\pi_d$ préserve, sur les mots bien-formés, l'union disjointe, il nous suffit de prouver que chaque monôme non-ambigu se projette sur une union disjointe de monômes modulaires non-ambigus.
Ainsi on considère un monôme non-ambigu $B_0^* b_1 B_1^* \cdots b_n B_n^*$ sur l'alphabet enrichi avec $b_i = (a_i, r_i)$. Alors la projection de ses mots bien formés donne l'expression rationnelle

$$(A_{0,0} \cdots A_{d-1,0})^* A_{0,0} \cdots A_{r_1,0} a_1 (A_{i+1,1} \cdots A_{i,1})^* A_{i+1,1} \cdots A_{r_2,1} a_2 \cdots$$

où $A_{j,i} = \{a \mid (a,j) \in B_i\}$. Cette union peut ensuite être réécrite comme union disjointe de monômes modulaires non-ambigus. Ceci s'effectue en généralisant la réécriture

suivante qui préserve la non-ambiguïté :

$$A_{0,0} \cdots A_{r_1,0} a_1 := \bigcup_{c_1 \dots c_r \in A_{0,0} \dots A_{r,0}} c_1 \emptyset^* \dots c_r \emptyset^* a_1.$$

Réciproquement, considérons un polynôme modulaire non ambigu sur un alphabet $A^*$. Les langages de $\mathcal{QDA}(A^*)$ étant clos par union, il suffit de prouver que les monômes non-ambigu appartiennent à $\mathcal{QDA}(A^*)$. Soit $L$ un monôme non ambigu se décomposant de la façon suivante :

$$L = (A_{0,0} \cdots A_{d-1,0})^* a_1 (A_{0,1} \cdots A_{d-1,1})^* \cdots a_n (A_{0,n} \cdots A_{d-1,n})^*.$$

On enrichit alors notre alphabet de la même façon, et posons, pour tout $i$ inférieur à $n$, $B_i = \{(a, j) \mid a \in B_{j,i}\}$. Alors, si l'on note $C_i = \cup_{a \in A}(a, i)$, le langage $L' = \big(B_0^*(a_1, 0)B_1 \dots B_{n-1}(a_n, (n-1) \bmod d)B_n^*\big) \cap K_d \cap A_d^* C_{(n-1 \bmod i)}$ est tel que $\pi_d(L') = L$. On remarque également que pour tout alphabet $A$ et toute lettre $a$ de $A$, le langage $A^*a$ appartient à $\mathcal{DA}(A^*)$. Pour se convaincre, il suffit de constater qu'un tel langage est facilement descriptible par une formule de $\mathbf{FO}^2[<](A^*)$. Alors, le langage $L'$ appartient à $\mathcal{DA}(A_d^*)$ et l'on conclut la preuve grâce au Théorème 3.4 et la Proposition 3.6. □

Nous pouvons maintenant résumer les différentes caractérisations de $\mathbf{FO}^2[<, \mathrm{MOD}]$ obtenues dans un seul théorème, qui est une conséquence directe des Propositions 3.6, 3.21, 3.22 et du Théorème 3.4.

**Théorème 3.23.** *Soit un langage régulier $L$. Les propositions suivantes sont alors équivalentes :*

- *Le timbre syntaxique de $L$ appartient à* **QDA**,
- *$L$ est définissable dans $\mathbf{FO}^2[<, \mathrm{MOD}]$,*
- *$L$ est définissable dans $\mathbf{TL}[X_a^{r \bmod d}, Y_a^{r \bmod d}]$,*
- *$L$ s'écrit comme union disjointe et finie de monômes modulaires non-ambigus.*

Nous tenons à préciser une dernière caractérisation logique, obtenue dans [KW13], stipulant que $\mathbf{FO}^2[<, \mathrm{MOD}] = \Delta_2[<, \mathrm{MOD}]$ où $\Delta_2[\sigma] = \Sigma_2[\sigma] \cap \Pi_2[\sigma]$, étendant ainsi le résultat classique $\mathbf{FO}^2[<] = \Delta_2[<]$.

Nous obtenons ici la décidabilité du fragment $\mathbf{FO}^2[<, \mathrm{MOD}]$ grâce à la décidabilité du fragment $\mathbf{FO}^2[<]$. La preuve utilise des outils classiques de la théorie des automates, tels que les jeux d'Ehrenfeucht-Fraïssé et les congruences, afin d'exploiter les spécificités de notre fragment et en extraire la décidabilité. Le résultat est qu'en plus d'obtenir cette décidabilité, nous sommes à même d'obtenir aisément, grâce à notre étude, l'extension des différents formalismes connus comme équivalents à notre fragment, afin d'y inclure les prédicats modulaires.

La question suivante qui nous a alors semblé naturelle est la suivante : n'existe-t-il pas d'approche générale permettant de conclure sur le comportement d'un fragment de logique lorsque l'on l'enrichit par les prédicats modulaires ?
Cet enrichissement correspond-il toujours à l'opération de **Q**-variété ?

La section suivante prend ce parti et va chercher à obtenir des arguments génériques permettant de conclure sur la transférabilité de la décidabilité lors de l'ajout des prédicats modulaires.

## 3.3 Une approche générale

Après nous être intéressés à la décidabilité du problème de décision pour le fragment $\mathbf{FO}^2[<, \mathrm{MOD}]$, nous désirons maintenant répondre à la question générale : l'ajout des prédicats modulaires à un fragment préserve-t-il la décidabilité du problème de décision ?

Ceci soulève tout d'abord la question de la définition d'un fragment de logique. En effet, à quel moment peut-on dire qu'un ensemble de formules logiques possède une structure de fragment. Cette question a été étudiée en détails par Kufleitner et Lauser dans [KL12], et nous explicitons dans la première section les propriétés qui pour nous seront nécessaires pour un fragment de logique.
La section suivante s'attache à obtenir une caractérisation algébrique de l'ajout des prédicats modulaires à un fragment. Malheureusement, il s'avère que cette caractérisation ne préserve pas la décidabilité. Les sections suivantes ont alors pour but d'étudier cette caractérisation afin de donner des conditions suffisantes sur les fragments pour le transfert de la décidabilité du problème de décision. Ainsi, même si l'on échoue à obtenir un théorème de transfert général, on obtient tout de même des preuves de décidabilité pour un grand nombre de fragments. Les preuves reposent sur une réduction de notre question à un problème sur les catégories, et plus particulièrement sur l'étude de la décidabilité, ainsi que de la caractérisation en termes d'équations, de variétés de catégories.

Nous donnerons lorsque cela s'avérera nécessaire ces définitions non classiques dans la théorie des semigroupes.

### 3.3.1 Fragments logiques

Nous justifions ici notre définition personnelle de la notion de fragment de logique. Le but n'est pas de nous démarquer mais de donner des conditions minimales pour l'application de nos théorèmes. En ce sens nos définitions sont plus faibles que celles données par Kufleitner et Lauser dans [KL12]. En effet, dans cet article, le but des auteurs est donner des conditions suffisantes mais a priori non nécessaires afin qu'un ensemble de formules soit caractérisé par une $\mathcal{C}$-variété de timbres.
En revanche, de par les méthodes utilisées, nous partons de fragments dont la décidabilité et une caractérisation algébrique sont connues. Nous désirons donner les conditions syntaxiques minimales sur les fragments qui nous permettent d'appliquer nos théorèmes.
On rappelle que dans le cadre de ce manuscrit, un ensemble de formules $\mathcal{F}$ de $\mathbf{MSO}[Reg]$ est appelé un *fragment de logique* si, et seulement si, l'ensemble $\mathcal{F}$ est syntaxiquement clos par conjonction, disjonction et substitution atomique.
Cette condition a une origine technique. Elle nous permettra de transformer les formules avec prédicats modulaires et de changer d'alphabet, permettant ainsi la caractérisation algébrique de l'ajout des prédicats modulaires.
On rappelle également que pour tout alphabet $A$, on notera $\mathcal{F}(A^*)$ l'ensemble des langages de $A^*$ définissables par une formule de $\mathcal{F}$.

Le but de notre étude est un transfert de décidabilité, et nos méthodes sont algébriques. Ainsi, on ne va considérer, dans nos théorèmes, que des fragments dont la

caractérisation algébriques est connue.

Soit $\mathbf{V}$ une $\mathcal{C}$-variété de timbres et $\mathcal{V}$ la $\mathcal{C}$-variété de langages correspondantes On dira qu'un fragment $\mathcal{F}$ est *équivalent* à une $\mathcal{C}$-variété $\mathbf{V}$ (ou à $\mathcal{V}$) si pour tout alphabet $A$, on a $\mathcal{F}(A^*) = \mathcal{V}(A^*)$.

### 3.3.2 Caractérisation algébrique de l'ajout de prédicats modulaires

Nous rappelons ici la définition du produit en couronne, déjà donnée dans les préliminaires de ce manuscrit, mais clarifiée ici lorsque l'on effectue le produit par un timbre de la *lm*-variété **MOD**. Nous rappelons aussi que la *lm*-variété **MOD** est l'ensemble des timbres $\pi : A^* \to C_d$ tels que $A$ est un alphabet, $d$ un entier positif et pour toutes lettres $a$ et $b$ de $A$, $\pi(a) = \pi(b)$.
Alors on note le produit en couronne d'un monoïde $M$ et du groupe cyclique $C_d$ d'ordre $d$ par $M \circ C_d$. Ce produit est défini sur le monoïde $M^{C_d} \times C_d$ par la loi interne suivante :

$$(f, i)(g, j) = (f \cdot g_j, i + j)$$

où $\cdot$ est le produit terme à terme sur $M^{C_d}$ et $g_j : C_d \to M$ est l'application $g$ décalée de $j$ définie par $g_j(t) = g(t + j)$.

Maintenant que l'on a défini le produit en couronne d'un monoïde par un timbre de **MOD**, nous pouvons généraliser au produit en couronne d'une variété $\mathbf{V}$ par **MOD**. Puisque **MOD** est une *lm*-variété, le résultat du produit en couronne sera lui aussi une *lm*-variété. Un timbre $\eta : A^* \to M$ appartient à $\mathbf{V} * \mathbf{MOD}$ si, et seulement si, il existe un entier positif $d$ et un timbre $\mu : (B \times C_d)^* \to N$, avec $\mu \in \mathbf{V}$ tels que $\eta$ *lm*-divise $\mu' : B^* \to N \circ C_d$, défini par $\mu'(b) = \big(f, 1\big)$ avec $f(i) = \mu(b, i)$.

Cette définition en tant que clôture par *lm*-division de produits en couronne n'est pas simple à manipuler, bien que nécessaire pour préserver une structure de $\mathcal{C}$-variété. Straubing, dans [Str79, Str89], a défini le principe du produit en couronne, théorème donnant une description des langages du produit en couronne de deux monoïdes, en fonction des langages de chacun des monoïdes. Bien qu'ayant de nombreuses applications, comme la caractérisation des langages sans étoile par les apériodiques, nous lui préférons ici une version adaptée à la *lm*-variété **MOD**, qui est une spécialisation du principe de produit en couronne pour les variétés de timbres, et présenté initialement dans [CPS06]. Dans cet article, les auteurs décrivent les langages de $\mathbf{V} * \mathbf{MOD}$ en terme de fonctions séquentielles inverses. Nous lui préférons ici une version utilisant la projection $\pi_d$ définie en Section 3.1.

**Théorème 3.24** (Principe du Produit en Couronne pour **MOD** [CPS06])**.** *Soit une variété $\mathbf{V}$, $\mathcal{V}$ la variété de langages correspondante, $L$ un langage régulier de $A^*$ et $d$ un entier strictement positif. Alors les propositions suivantes sont équivalentes :*

1. *Le langage $L$ est reconnu par un timbre appartenant à $\mathbf{V} * \mathbf{MOD}_d$,*
2. *Le langage $L$ appartient au treillis de langage engendré par les langages de la forme $(A^d)^* A^i$ où $i$ est un entier positif inférieur à $d$ et les langages de la forme $\pi_d(L' \cap K_d)$ où $L' \in \mathcal{V}(A_d^*)$.*

On décrit ainsi les langages reconnus par $\mathbf{V} * \mathbf{MOD}_d$. Naturellement, la *lm*-variété $\mathbf{V} * \mathbf{MOD}$ est l'union sur l'ensemble des entiers strictement positifs $d$ des *lm*-variétés $\mathbf{V} * \mathbf{MOD}_d$.

Cette description va nous permettre de relier le produit en couronne avec un fragment auquel on a ajouté les prédicats modulaires. Le prochain théorème, qui constitue le résultat principal de la section, nous donne la caractérisation algébrique non effective de l'ajout des prédicats modulaires. Il est à noter que ce résultat a été obtenu parallèlement dans [KW13].

**Théorème 3.25.** *Soit un fragment $\mathcal{F}[\sigma]$ équivalent à une variété de monoïdes $\mathbf{V}$, un langage régulier $L$ et un entier strictement positif $d$. Alors les propositions suivantes sont équivalentes :*

1. *$L$ est défini par une formule de $\mathcal{F}[\sigma, \mathrm{MOD}_d]$,*
2. *$\eta_L$ appartient à $\mathbf{V} * \mathbf{MOD}_d$,*
3. *il existe des langages $L_0, \ldots, L_{d-1}$ appartenant à $\mathcal{V}(A_d^*)$ tels que :*

$$L = \bigcup_{i=0}^{d-1} \left( (A^d)^* A^i \cap \pi_d(L_i \cap K_d) \right) \tag{3.1}$$

Le corollaire suivant est conséquence directe de ce théorème et de la définition de $\mathbf{V} * \mathbf{MOD}$.

**Corollaire 3.26.** *Soit $\mathcal{F}[\sigma]$ un fragment. Un langage $L$ est définissable dans $\mathcal{F}[\sigma, \mathrm{MOD}]$ si, et seulement, si $L$ est reconnu par un timbre dans $\mathbf{V} * \mathbf{MOD}$.*

*Démonstration.* On traite ici le cas d'une variété de monoïdes. En réalité, le cas d'une variété de semigroupes, ou *ne*-variété, peut être traité de façon similaire.

(3) implique (2) est une conséquence directe du Théorème 3.24.

(2) implique (3). Supposons que le timbre syntaxique $\eta_L$ appartienne à $\mathbf{V} * \mathbf{MOD}$. Selon le Théorème 3.24, il suffit de prouver que tout langage du treillis décrit peut s'écrire sous la forme de l'équation 3.1. On remarque que nous travaillons dans un treillis distributif et que les langages $(A^d)^* A^i$ avec $i$ un entier inférieur à $d$ forment une partition de $A^*$. Alors, il existe des langages $H_0, \ldots, H_{d-1}$ du treillis engendré par les langages $\pi_d(L' \cap K_d)$ où $L' \in \mathcal{V}(A_d^*)$ tels que $L = \bigcup_{i=0}^{d-1} \left( (A^d)^* A^i \cap H_i \right)$. Grâce au Lemme 3.2, on sait donc qu'il existe, pour tout $i < d$, des langages $L_i$ dans la variété de langages $\mathcal{V}(A_d^*)$ tels que $H_i = \pi_d(L_i \cap K_d)$.

Afin de prouver l'équivalence entre (1) et (3) nous avons besoin d'un résultat préliminaire de décomposition des formules de $\mathcal{F}[<, \mathrm{MOD}]$, entrainant ainsi un résultat similaire sur les langages associés. Cette décomposition est donnée par le lemme suivant, qui repose sur les propriétés syntaxiques des fragments.

**Lemme 3.27.** *Soit un fragment de logique $\mathcal{F}[\sigma, \mathrm{MOD}]$ et une formule $\varphi$ de $\mathcal{F}[\sigma, \mathrm{MOD}_d]$. Alors il existe des formules $(\psi_i)_{0 \leqslant i < d}$ de $\mathcal{F}[\sigma, \mathrm{MOD}_d]$ ne contenant aucun prédicat de longueur modulaire $D_j^d$ et telles que $\varphi \equiv \vee_{i=0}^{d-1}(\psi_i \wedge D_i^d)$. De plus, on a :*

$$L(\varphi) = \bigcup_{i=0}^{d-1} \big((A^d)^* A^i \cap L(\psi_i)\big).$$

*Démonstration.* On définit les formules $\psi_i$ de façon syntaxique. Pour $i < d$, la formule $\psi_i$ est définie comme la formule $\varphi$ où chaque prédicat de longueur $D_i^d$ est remplacé par *vrai* et chaque autre prédicat $D_j^d$ est remplacé par *faux*. Cette transformation est une substitution syntaxique, et par conséquent chaque formule $\psi_i$ est une formule de $\mathcal{F}[\sigma, \mathrm{MOD}_d]$. Ainsi, puisque la formule simple $D_i^d$ reconnait le langage $(A^d)^* A^i$, les prédicats $D_i^d$ induisent une partition naturelle des mots de $A^*$. Alors tout mot $u$ satisfaisant $\varphi$, il existe un entier $i$ tel que $u$ satisfait $\phi_i \wedge D_i^d$, et inversement, tout mot satisfaisant l'un des $\phi_i \wedge D_i^d$ satisfera $\varphi$. On obtient la première affirmation du lemme. La seconde affirmation découle naturellement de la première équivalence. □

On peut maintenant conclure la preuve du Théorème 3.25. Soit un langage régulier $L$ définissable dans $\mathcal{F}[\sigma, \mathrm{MOD}]$. Il existe alors un entier strictement positif $d$ et une formule $\varphi$ appartenant à $\mathcal{F}[\sigma, \mathrm{MOD}_d]$ telle que $L = L(\varphi)$. En utilisant le Lemme 3.27, on sait que l'on peut réduire $\varphi$ à des formules ne contenant pas de prédicat de longueur modulaire. Soit ainsi $\varphi$ une formule de $\mathcal{F}[\sigma, \mathrm{MOD}_d]$ ne contenant pas de tel prédicat. On transforme syntaxiquement $\varphi$ en une formule $\psi$ en remplaçant chaque prédicat modulaire $\mathrm{MOD}_i^d(x)$ par $\bigvee_{a \in A} (\mathbf{a}, \mathbf{i})(x)$ où les prédicats $(\mathbf{a}, \mathbf{i})$ sont des prédicats de lettre sur l'alphabet enrichi. On remplace également chaque prédicat de lettre $\mathbf{a}(x)$ par $\bigvee_{0 \leqslant i < d} (\mathbf{a}, \mathbf{i})(x)$. Notre transformation effectuant des substitutions syntaxiques, la formule $\psi$ est donc une formule de $\mathcal{F}[\sigma](A_d^*)$. En remarquant que si un mot de $A^*$ satisfait $\varphi$, alors son mot bien formé associé vérifiera $\psi$, et qu'inversement si un mot bien formé vérifie $\psi$ alors sa projection vérifie $\varphi$, on obtient l'égalité $L(\varphi) = \pi_d(L(\psi) \cap K_d)$.

Réciproquement, une formule $\psi$ de $\mathcal{F}[\sigma](A_d^*)$ peut être syntaxiquement transformée en une formule $\varphi$ de $\mathcal{F}[\sigma, \mathrm{MOD}_d]$ en remplaçant chaque prédicat de lettre $(\mathbf{a}, \mathbf{i})(x)$ dans $\psi$ par $\mathbf{a}(x) \wedge \mathrm{MOD}_i^d(x)$. On obtient également $L(\varphi) = \pi_d(L(\psi) \cap K_d)$, concluant ainsi la preuve. □

On a ainsi obtenu une caractérisation algébrique de l'ajout des prédicats modulaires à un fragment. Cependant, le produit semi-direct ne préserve pas la décidabilité. Ainsi, dans [Aui10], Auinger a donné explicitement deux variétés de monoïdes décidables dont

le produit semi-direct est indécidable. Notre caractérisation n'est donc pas utilisable telle quelle. Les prochaines sections s'attacheront alors à donner des conditions suffisantes où la décidabilité est préservée, en utilisant notamment cette caractérisation, ainsi que les variétés de catégories que nous introduisons dans la prochaine partie.

### 3.3.3 De la puissance des catégories

#### Définitions

Dans [Til87], Tilson explicite un lien entre monoïdes et catégories. Plus précisément, le Théorème de la catégorie dérivée relie le produit semi-direct de deux variétés de monoïdes au *global* d'une variété, celui-ci étant une variété de catégories. Nous donnerons tout d'abord les définitions nécessaires puis nous formulerons le Théorème de la catégorie dérivée. Nous montrerons dans un second temps en quoi une propriété sur certaines variétés de catégories spécifiques nous donne la décidabilité du produit semi-direct par **MOD**. Toutes les catégories considérées ici seront finies.

On rappelle la définition de *catégorie*. Formellement, une catégorie $C$ est définie par :

- Un ensemble fini $Obj(C)$ de sommets ou objets de $C$.
- Pour chaque paire $(u, v)$ d'éléments dans $Obj(C)$, un ensemble fini $C(u, v)$ de flèches de $u$ à $v$.
- Une loi de composition associative qui associe à deux flèches consécutives $x \in C(u, v)$ et $y \in C(v, w)$ une flèche $xy$ appartenant à $C(u, w)$.
- Pour chaque élément $u$ de $Obj(C)$, une flèche $1_u$ de $C(u, u)$ agissant comme une identité locale, i.e $x1 = x$ et $1y = y$ pour tout $x \in C(v, u)$ et $y \in C(u, w)$.

Comme précisé par la définition, nous ne considérerons que des catégories finies. Nous voyons les catégories comme une généralisation des monoïdes finis. En effet, si l'on considère une catégorie à un objet, l'ensemble de ses flèches, équipé de la loi de composition des flèches consécutives, forme un monoïde dont l'élément neutre est l'identité de son unique objet. Et inversement tout monoïde peut être vu comme une catégorie à un objet. Plus généralement, l'ensemble des boucles autour d'un objet d'une catégorie forme un monoïde, appelé le *monoïde local* autour de cet objet.

On donne maintenant la définition de la catégorie dérivée pour **MOD** donnée dans [CPS06], dont la définition générale pour toute variété vient de Tilson dans [Til87]. Toutes les catégories que nous considérerons dans la suite seront des catégories dérivées.

**Définition 3.9:**
Soit un timbre $\varphi : A^* \to M$ et un entier strictement positif $d$. La *catégorie $d$-dérivée* de $\varphi$, que l'on note $C_d(\varphi)$, est la catégorie dont l'ensemble des objets est $\{0, \ldots, d-1\}$. Pour chaque paire d'entiers $(i, j)$ inférieurs à $d$, l'ensemble des flèches de $i$ à $j$ est l'ensemble des éléments de $M$ tels qu'il existe un mot $u$ de $A^*$ tel que $\varphi(u) = m$ and $i + |u| \equiv j \bmod d$. Étant donné un langage régulier $L$, la catégorie $d$-dérivée de $L$, que l'on note $C_d(L)$, est la catégorie $d$-dérivée de son timbre syntaxique $C_d(\eta_L)$.

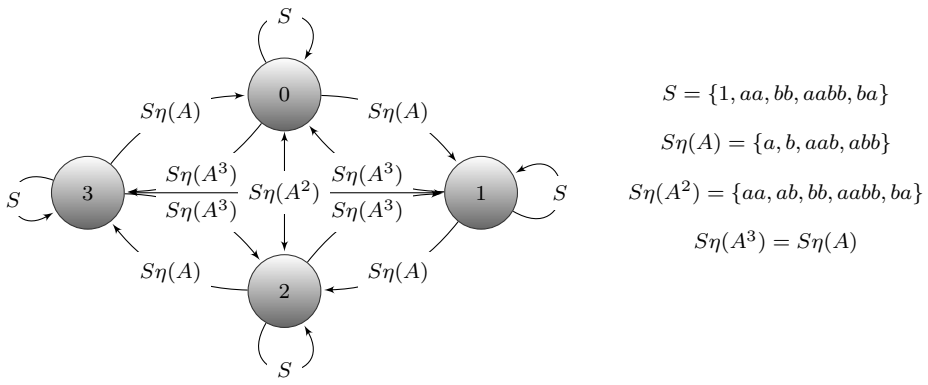



Figure 3.8: Catégorie 4-dérivée de $(aa)^*ab(bb)^*$

Dans le cas d'une catégorie dérivée selon **MOD**, les monoïdes locaux jouent un rôle particulier, explicité dans le lemme suivant, qui est une conséquence directe de la définition.

**Lemme 3.28.** *Soit un entier strictement positif $d$ et un langage régulier $L$ d'indice de stabilité $s$. Alors les monoïdes locaux de $C_d(L)$ sont égaux entre eux et isomorphes au sous monoïde $\eta_L((A^d)^*)$. En particulier, les monoïdes locaux de $C_s(L)$ sont isomorphes au monoïde stable de $L$.*

**Exemple 3.3.** Nous donnons dans la Figure 3.8 un exemple de catégorie dérivée, mettant en avant les nombreuses symétries que comporte cette construction. Est ainsi donnée la catégorie 4-dérivée du langage régulier $(aa)^*ab(bb)^*$. On note $\eta$ son timbre syntaxique et $S$ son monoïde stable. On remarque également que son indice de stabilité vaut 4.
Le monoïde $M_L$ est l'ensemble $\{1, a, b, aa, ab, ba, bb, aab, aabb\}$.

### Variétés de catégories

On désire définir des variétés de catégories spécifiques. Par définition des variétés, on a alors d'abord besoin d'une définition de division de catégories. Cette définition est en accord avec l'idée que les catégories étendent les monoïdes, car la division de catégorie étend celle de monoïdes.
Formellement, soient deux catégories $C$ et $D$. On dit que $C$ *divise* $D$ si, et seulement si, il existe une application $\tau : Obj(C) \to Obj(D)$, et pour chaque paire $(u, v)$ d'objets de $C$, une relation $\tau : C(u, v) \to D(\tau(u), \tau(v))$ vérifiant les propriétés suivantes :

1. $\tau(x)\tau(y) \subseteq \tau(xy)$ pour toute paire de flèches consécutives $x, y$,
2. $\tau(x) \neq \emptyset$ pour toute flèche $x$,
3. $\tau(x) \cap \tau(y) \neq \emptyset$ implique $x = y$ si $x$ et $y$ sont coterminales (i.e. ont même origine et destination),

4. $1_{\tau(u)} \in \tau(1_u)$ pour tout objet $u$ de $C$.

Afin d'illustrer notre définition de division de catégorie, nous donnons maintenant une proposition simple sur les catégories dérivées d'un langage, qui peut être interprétée comme une version pour les catégories du fait que l'on puisse facilement exprimer les prédicats $\mathrm{MOD}_d$ par une disjonction de prédicats de $\mathrm{MOD}_{dn}$, pour tout $n > 1$, et qu'ainsi les langages définissables par un fragment $\mathcal{F}[\sigma, \mathrm{MOD}_d]$ sont définissables par $\mathcal{F}[\sigma, \mathrm{MOD}_{dn}]$.

**Proposition 3.29.** *Soit $L$ un langage régulier. Alors pour tout entiers $0 < d \leqslant d'$, si $d$ divise $d'$, alors la catégorie $C_{d'}(L)$ divise la catégorie $C_d(L)$.*

*Démonstration.* Définissons la relation $\tau : C_{d'}(L) \to C_d(L)$ comme suit. L'application objet $\mathrm{Ob}(\tau) : \{0, \ldots, d'-1\} \to \{0, \ldots, d-1\}$ est naturellement définie par $\mathrm{Ob}(\tau)(x) = x \bmod d$ pour tout $x \in \mathbb{Z}_{d'}$. Soit maintenant une flèche $(x, m, y)$ de $C_{d'}(L)$. Par définition de la catégorie dérivée,, il existe un mot $u \in A^*$ tel que $\eta_L(u) = m$ et $|u| \equiv y - x \bmod d'$. Soient maintenant $a = x \bmod d$ et $b = y \bmod d$. Alors, étant donné que $d$ divises $d'$, on a $|u| \equiv b - a \bmod d$. Ainsi, la flèche $(a, m, b)$ apparaît dans $C_d(L)$. On définit $\tau(x, m, y) = (a, m, b)$. L'application résultante $\tau$ est donc un morphisme et toute paire de flèches $(x, m, y) \neq (x, m', y)$, nous avons $\tau(x, m, y) \neq \tau(x, m', y)$. Ainsi, $\tau$ définit une division de $C_{d'}$ dans $C_d$. □

On peut maintenant définir une *variété de catégories*, de façon similaire aux autres types de variétés connues, comme étant un ensemble de catégories finies clos par produit cartésien fini et par division. Ainsi, pour une variété de monoïdes $\mathbf{V}$, on considère le *global de* $\mathbf{V}$, que l'on note $\mathbf{gV}$, comme l'ensemble des catégories qui divisent un monoïde de $\mathbf{V}$ interprété comme une catégorie à un élément. Cet ensemble est clos par division par définition, et clos par produit puisque le produit cartésien de catégories sera divisible par le produit des monoïdes divisant chacune des catégories.
Le Théorème de la catégorie dérivée provient originellement de Tilson [Til87], et concerne les variétés de monoïdes et de semigroupes. Dans sa thèse, Laura Chaubard [Cha] a étendu ce théorème aux $\mathcal{C}$-variétés. Nous donnons ici la version de Chaubard, adaptée à la *lm*-variété **MOD**.

**Théorème 3.30** (Théorème de la catégorie dérivée pour **MOD** [Cha])**.** *Soit une ne-variété $\mathbf{V}$, un langage régulier $L$ et un entier strictement positif $d$. Le timbre syntaxique de $L$ appartient à $\mathbf{V} * \mathbf{MOD}_d$ si, et seulement si, la catégorie $d$-dérivée $C_d(L)$ appartient à $\mathbf{gV}$.*

On obtient ainsi une nouvelle approche pour la décidabilité de $\mathbf{V} * \mathbf{MOD}$. Cependant la décidabilité du global d'une variété reste une question difficile à traiter. On s'intéresse dorénavant à la décidabilité des variétés de catégories. Si l'on regarde le cas classique des variétés de monoïdes, la décidabilité de ces classes est souvent obtenue par une

caractérisation à l'aide d'un système fini d'équations profinies calculables que l'on teste de façon exhaustive afin de déterminer si un monoïde donné appartient à la variété cible. Le parallèle avec les catégories s'étend ici aussi puisqu'il existe une variante du théorème de Reiterman pour les variétés de catégories.

Le rôle des équations est joué, dans le cas des catégories, par des équations de chemin sur des catégories libres profinies sur un graphe étiqueté, et nous donnons maintenant les définitions formelles.

**Définition 3.10: Catégorie libre**
Soit un graphe $X$ connexe et $E$ l'ensemble de ses flèches. On définit alors $X^*$ comme le graphe ayant le même support que $X$ et dont les flèches sont l'ensemble des mots $u = u_1 \dots u_n$ de $E^*$ tels que pour tout $i < n$, $u_i$ et $u_{i+1}$ sont des flèches consécutives. On nomme ainsi $X^*$ la *catégorie libre* sur $X$.

Une catégorie libre joue alors naturellement le rôle de monoïde librement engendré, et l'on équipe une catégorie libre d'une notion de distance, menant ainsi à une notion de topologie sur notre catégorie libre.

**Définition 3.11: Catégorie libre profinie**
Soit $X$ un graphe connexe et $u$ et $v$ deux chemins coterminaux de $X^*$. On définit alors

$$r(u,v) = \min\left\{n \mid \text{ il existe } \varphi : X^* \to C \text{ avec } n = |C| \text{ et } \varphi(u) \neq \varphi(v)\right\}$$

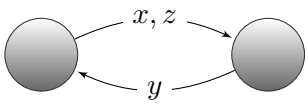
où $\varphi$ est un morphisme de catégorie et $C$ une catégorie finie. On définit alors la distance $d(u,v) = 2^{-r(u,v)}$ qui forme une distance ultramétrique sur $X^*$. La complétion de $X$ pour cette métrique s'appelle la *catégorie libre profinie* sur $X$ et l'on la note $\widehat{X}^*$.

Nous étendons maintenant une nouvelle propriété des mots profinis aux chemins profinis, correspondant à l'extension continue des morphismes de $X^*$ jusqu'à $\widehat{X^*}$.

**Proposition 3.31.** *Soit un graphe $X$, une catégorie finie $C$ et $\varphi : X^* \to C$ un morphisme de catégories. Il existe alors une unique fonction continue $\widehat{\varphi} : \widehat{X^*} \to C$ telle que pour chemin fini $u$ de $X^*$, $\widehat{\varphi}(u) = \varphi(u)$.*

*De plus, pour tout chemin profini $u$ de $\widehat{X^*}$, il existe un chemin fini $v$ de $X^*$ tel que $\widehat{\varphi}(u) = \widehat{\varphi}(v)$.*

Nous sommes maintenant prêts à définir les équations de chemin profinis. Là où les équations sur les variétés de monoïdes ont pour support un alphabet, les équations de chemin utilisent un graphe. Voici la définition formelle.




$xyz = zyx$

Figure 3.9: Équation de chemin caractérisant les catégories commutatives

**Définition 3.12: Équation de chemin**
Soit un graphe $X$ et $u$ et $v$ deux chemins profinis de $X^*$ coterminaux. Une catégorie finie $C$ satisfait l'équation $(X, u = v)$ si pour tout morphisme $\varphi$ de $X$ dans $C$, nous avons l'égalité $\widehat{\varphi}(u) = \widehat{\varphi}(v)$.

Les équations de chemins ont le même pouvoir pour les variétés de catégories que les équations ont pour les variétés de monoïdes. Cette extension du théorème de Reiterman a été prouvée par Almeida et Weil et nous redonnons l'énoncé ici.

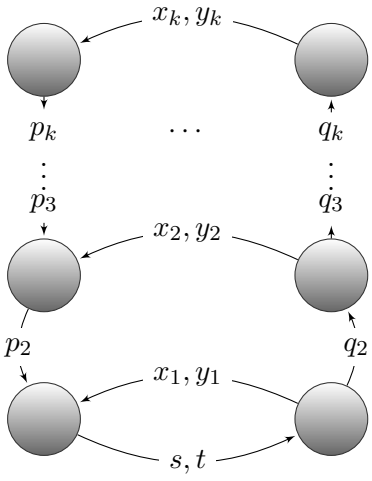
**Théorème 3.32.** *[AW98] Toute variété de catégories finie non triviale est caractérisée par un ensemble d'équations de chemins profinis.*

*Réciproquement, l'ensemble des catégories finies satisfaisant un ensemble d'équations de chemins profinis sur un graphe donné forme une variété de catégories.*

Une équation de chemins profinis est alors donnée par un graphe support et l'égalité de deux chemins profinis sur ce graphe. Nous discutons maintenant du graphe support, et plus particulièrement de sa taille. Nous dirons ainsi qu'une variété de catégories est de *rang fini* si elle est définie par des équations de chemins sur un graphe fini.

Plus particulièrement, nous définissons le *rang* d'une variété de catégories $\mathbf{V}$ comme la taille minimale des graphes supports permettant de définir $\mathbf{V}$. Par abus de langage, et puisque nous ne considérerons ici, en tant que variété de catégories, uniquement des variétés globales, nous dirons ici qu'une variété de monoïdes $\mathbf{V}$ est de rang fini (resp. de rang $k$) si sa variété globale $\mathbf{gV}$ est de rang fini (resp. de rang $k$). Nous donnons maintenant quelques exemples connus de variétés de rang fini, ainsi que leurs équations de chemins profinis.

**Exemple 3.4.**
- Les variétés de rang 1 sont appelées variétés locales. C'est le cas de beaucoup de variétés largement étudiées telles que la variété des monoïdes formant un semi-treillis $\mathbf{J}_1 = [\![xy = yx, x^2 = x]\!]$, la variété étudiée en Section 3.2 $\mathbf{DA} = [\![(xy)^\omega x (xy)^\omega = (xy)^\omega]\!]$ ou la variété des monoïdes apériodiques $\mathbf{A} = [\![x^\omega = x^{\omega+1}]\!]$. Le cas des variétés locales mérite sa propre étude, qui est faite dans la partie suivante.
- La variété des monoïdes commutatifs $\mathbf{Com} = [\![xy = yx]\!]$. Son global $g\mathbf{Com}$ est défini par l'équation de chemin donnée dans la Figure 3.9 (voir [Til87]).
- On note $\mathcal{B}\mathbf{\Sigma}_k^2[<]$ les formules de $\mathbf{FO}^2[<]$ ayant au plus $k$ alternances de quantificateurs existentiels et universels. Une récente description algébrique des langages




Nous posons $U_1 = (sx_1)^\omega s(y_1 t)^\omega$ et $V_1 = (sx_1)^\omega t(y_1 t)^\omega$

$$U_m = (p_m U_{m-1} q_m x_m)^\omega p_m U_{m-1} q_m (y_m p_m U_{m-1} q_m)^\omega$$

$$V_m = (p_m U_{m-1} q_m x_m)^\omega p_m V_{m-1} q_m (y_m p_m U_{m-1} q_m)^\omega$$

$\mathbf{gV}_k$ satisfait l'équation $U_k = V_k$

définissables par les formules de $\mathcal{B}\mathbf{\Sigma}^2_{k+1}[<]$ a été établie dans [KS12, KW12]. On note $\mathbf{V}_k$ la variété de monoïdes définie par ces fragments de logique. Ce résultat fut également étendu aux fragments $\mathcal{B}\mathbf{\Sigma}^2_{k+1}[<, +1]$ dans [KL12]. De ce dernier résultat nous pouvons dériver l'équation de chemins décrivant $\mathbf{gV}_k$ et nous la donnons dans le schéma ci-dessus.

Nous allons maintenant nous intéresser au cas des variétés locales, où la décidabilité de $\mathbf{gV}$ se réduit à la décidabilité de $\mathbf{V}$. Cela induit une caractérisation simple de l'ajout des prédicats modulaires.

### Exemple des variétés locales

On définit maintenant le *local* d'une variété de monoïdes. Étant donnée une variété de monoïdes $\mathbf{V}$, on note $\ell\mathbf{V}$ le local de $\mathbf{V}$, qui est par définition l'ensemble des catégories dont tous les monoïdes locaux sont des monoïdes de $\mathbf{V}$. L'ensemble $\ell\mathbf{V}$ forme une variété de catégories.
On rappelle également la définition de la $\mathbf{Q}$-variété $\mathbf{QV}$ : il s'agit de la $lm$-variété de timbres dont le monoïde stable appartient à $\mathbf{V}$. Le prochain théorème explicite le lien entre ces deux notions.

**Théorème 3.33.** *Soit une ne-variété $\mathbf{V}$ et un langage régulier $L$ d'indice de stabilité $s$. Alors les propositions suivantes sont équivalentes :*

1. *Le timbre syntaxique de $L$ appartient à $\mathbf{QV}$,*
2. *Il existe un entier strictement positif $d$ tel que $C_d(L)$ appartient à $\ell\mathbf{V}$,*
3. *$C_s(L)$ appartient à $\ell\mathbf{V}$.*

*Démonstration.*
(1) → (3). Si le timbre syntaxique de $L$ est dans $\mathbf{QV}$, alors par définition son monoïde stable appartient à $\mathbf{V}$. Mais, selon le Lemme 3.28, chacun des monoïdes locaux de $C_s(L)$ appartiennent à $\mathbf{V}$, et par conséquent $C_s(L)$ appartient à $\ell\mathbf{V}$ par définition de $\ell\mathbf{V}$.
(3) → (2). Est évident.
(2) → (1). Supposons que la catégorie dérivée $C_d(L)$ appartienne à $\ell\mathbf{V}$. Alors chaque monoïde local de $C_d(L)$ appartient à $\mathbf{V}$, et le Lemme 3.28 nous dit qu'ils sont tous égaux et isomorphes à $\eta_L((A^d)^*)$. Alors le sous-monoïde $\eta_L((A^{ds})^*)$ de $\eta_L((A^d)^*)$ appartient également à $\mathbf{V}$. Finalement, en utilisant la définition du monoïde stable, on obtient que $\eta_L((A^s)^*) = \eta_L((A^{ds})^*)$ appartient donc à $\mathbf{V}$ et par conséquent $\eta_L$ appartient à $\mathbf{QV}$. □

On remarque que tout monoïde d'une variété $\mathbf{V}$, lorsque l'on le voit comme une catégorie à un élément, appartient à $\ell\mathbf{V}$. Par conséquent la définition de $\mathbf{gV}$ nous dit que toute catégorie de $\mathbf{gV}$ divise, au sens des catégories, un élément de $\ell\mathbf{V}$. Puisque $\ell\mathbf{V}$ est une variété, elle est close par division et on a donc $\mathbf{gV} \subseteq \ell\mathbf{V}$.
Lorsque l'égalité est vérifiée, on a $\mathbf{gV} = \ell\mathbf{V}$ et on dit que $\mathbf{V}$ est une *variété locale*. Cela correspond en effet à la définition de l'exemple 3.4 des variétés locales, car si une variété de catégories est définissable par un système d'équations sur un sommet, alors ces équations peuvent être vues comme des équations profinies, et l'appartenance d'une catégorie à cette variété se réduit à l'appartenance de ses monoïdes locaux à la variété caractérisée par ces équations. On obtient ainsi la caractérisation effective suivante.

**Théorème 3.34.** *Soit une ne-variété $\mathbf{V}$. Alors on a l'inégalité $\mathbf{V} * \mathbf{MOD} \subseteq \mathbf{QV}$. Si de plus $\mathbf{V}$ est une variété locale, alors on a $\mathbf{V} * \mathbf{MOD} = \mathbf{QV}$.*

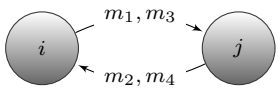
Étant donné que l'indice de stabilité et le monoïde stable d'un timbre sont des objets calculables, on obtient le corollaire suivant.

**Corollaire 3.35.** *Soit un fragment de logique $\mathcal{F}[\sigma]$ équivalent à une variété locale $\mathbf{V}$. Alors $\mathcal{F}[\sigma]$ est décidable si, et seulement si, $\mathcal{F}[\sigma, \mathrm{MOD}]$ est décidable.*

On obtient ainsi la décidabilité du problème de décision dans le cas de fragment équivalent à une variété locale. C'est le cas pour $\mathbf{FO}[<]$ et $\mathbf{FO}^2[<]$ par exemple. On retrouve ainsi la caractérisation donnée à la partie précédente, ainsi que le résultat de [BCST92]. Cependant le résultat de localité d'une variété est généralement un problème plus complexe que celui considéré. Cette caractérisation est utile dans le cas de variétés largement connues. Cela donne une preuve plus rapide et générique de certains résultats. Cela donne aussi des liens entre le produit semi-direct et la théorie des catégories.

**Remarque 3.2.** L'égalité $\mathbf{V} * \mathbf{MOD} = \mathbf{QV}$ n'est cependant pas toujours exacte. Par exemple, la variété $\mathbf{J}$ est connue pour être non-locale. Chaubard, Pin et Straubing ont néanmoins prouvé la décidabilité de $\mathbf{J} * \mathbf{MOD}$ dans [CPS06], en utilisant la caractérisation de $\mathbf{gJ}$ par Knast [Kna83] utilisant des équations de chemins profinis donnée en Figure 3.10. Cette caractérisation nous permet de séparer $\mathbf{J} * \mathbf{MOD}$ de $\mathbf{QJ}$. En effet, le

langage régulier $(aa)^*ab(bb)^*$, donné dans l'exemple 3.3, a son monoïde stable dans **J**. Sa catégorie $s$-dérivée ne vérifie cependant pas les équations de Knast pour $m_1 = m_2 = a$ et $m_3 = m_4 = b$. En effet, on a $(m_1m_2)^\omega(m_3m_4)^\omega = (aa)^\omega(bb)^\omega = aabb$ qui n'est pas dans notre langage alors que $(m_1m_2)^\omega m_1m_4(m_3m_4)^\omega = (aa)^\omega ab(bb)^\omega = ab$ et appartient donc à notre langage. On a donc $\mathbf{J} * \mathbf{MOD} \subsetneq \mathbf{QJ}$.




$$(m_1m_2)^\omega(m_3m_4)^\omega = (m_1m_2)^\omega m_1m_4(m_3m_4)^\omega$$

Figure 3.10: Équation de chemin de **gJ**.

### 3.3.4 Un théorème de délai partiel pour MOD

Le Théorème 3.25 nous donne une description des langages définissables dans $\mathcal{F}[\sigma, \mathrm{MOD}]$ en fonction des langages de $\mathcal{F}[\sigma]$, ainsi qu'une caractérisation algébrique malheureusement non effective. Cette caractérisation permet cependant, grâce au Théorème de la catégorie dérivée (Théorème 3.30), une réduction du problème de transfert de décidabilité à un problème de décidabilité sur les catégories. Une conséquence notable est le problème du transfert de la décidabilité se divise en deux sous-problèmes pouvant être traités séparément. Le premier problème est de pouvoir effectivement calculer un témoin de l'appartenance à un fragment enrichi, c'est-à-dire, étant donné un langage $L$ et une variété **V**, un entier $d_L$ tel que $L$ appartienne à $\mathbf{V} * \mathbf{MOD}$ si, et seulement si, la catégorie $d_L$-dérivée de $L$ appartienne au global **gV**. Le second problème se réduit alors à la décidabilité du global **gV**, problème déjà étudié dans la littérature. Ces deux sous-problèmes se traduisent également à tous les niveaux de notre étude. Ainsi, calculer cet entier est donc naturellement équivalent à pouvoir se restreindre aux prédicats modulaires selon un entier pour décider de l'appartenance d'un langage à un fragment $\mathcal{F}[\sigma, \mathrm{MOD}]$, ou décider avec quel groupe cyclique enrichir notre alphabet de départ. Le second problème revient à effectivement définir la formule de $\mathcal{F}[\sigma, \mathrm{MOD}]$, où calculer les langages $L_0, \ldots, L_{d-1}$ de la caractérisation (3.1) du Théorème 3.25.

Dans cette section, nous nous intéresserons au premier problème, qui est de prouver l'existence d'un témoin calculable, dépendant du langage d'entrée, mais également de la variété considérée, regroupant en un sens les propriétés du langage vis à vis de la variété. Nous appelons cette question un problème de délai, en référence au Théorème de Délai paru dans [Str85, Til87] et qui résout une question similaire concernant le produit semi-direct non plus avec **MOD**, mais avec la variété **LI**. L'entier que l'on cherche ainsi à déterminer sera appelé l'indice de Délai.

Nous énonçons tout de suite notre Théorème de Délai partiel, puis nous nous appliquerons à le prouver, avant de discuter de ses répercussions et de ses applications.

**Théorème 3.36** (Un Théorème de Délai partiel pour **MOD**)**.** *Soit une variété* $\mathbf{V}$ *de rang fini* $k$*, et* $L$ *un langage régulier d'indice de stabilité* $s$*.*

*Alors le timbre syntaxique de* $L$ *appartient à* $\mathbf{V} * \mathbf{MOD}$ *si, et seulement si, il appartient à* $\mathbf{V} * \mathbf{MOD}_{ks}$*.*

Le rang étant une propriété issue de l'étude des catégories, notre preuve reposera évidement sur les catégories, et les équations de chemins définies à la Section 3.3.3. Nous abordons la preuve dans la sous-section suivante.

### Preuve du Théorème de Délai partiel

Bien que ce soit évident, il est tout d'abord nécessaire de remarquer que si $L$ appartient à $\mathbf{V} * \mathbf{MOD}_{ks}$, alors il appartient à $\mathbf{V} * \mathbf{MOD}$. Nous ne traiterons donc que la réciproque.

Nous remarquons également que si $\mathbf{V}$ est une variété de rang 1, nous sommes alors dans le cas d'une variété locale traitée dans la section précédente et la caractérisation peut alors s'affranchir des catégories.

Soit maintenant une variété $\mathbf{V}$ de rang $k > 1$, un langage régulier $L$ d'indice de stabilité $s$, et supposons que le timbre syntaxique $\eta$ de $L$ appartienne à $\mathbf{V} * \mathbf{MOD}$. Par le Théorème 3.30, il existe alors un entier $d$ tel que la catégorie $d$-dérivée $C_d(L)$ appartienne au global $\mathbf{gV}$ de $\mathbf{V}$.

Nous prouvons le résultat pour un enier $d$ plus grand que $k$. Ceci à prouver le théorème, puisque si $C_d(L)$ appartient à $\mathbf{gV}$, alors pour tout multiple $d'$ de $d$, la Proposition 3.29 stipule que $C_{d'}(L)$ divise $C_d(L)$ et appartient donc également à $\mathbf{gV}$. Dans la suite de la preuve, on supposera donc que $ds$ est plus grand que $ks$. Ainsi, grâce à la Proposition 3.29, puisque $C_d(L)$ appartient à $\mathbf{gV}$, alors $C_{ds}(L)$, qui la divise, appartient également au global $\mathbf{gV}$. Alors $C_{ds}(L)$ satisfait toutes les équations de chemin $(X, u = v)$ définissant la variété globale $\mathbf{gV}$.

Le but de cette preuve est maintenant de prouver que s'il existe une équation de chemin définissant $\mathbf{gV}$ qui n'est pas satisfaite par $C_{ks}(L)$, alors $C_{ds}(L)$ ne peut pas la satisfaire non plus, contredisant notre hypothèse.

Soit une équation de chemin $(X, u = v)$ de rang $k$ et définissant $\mathbf{gV}$, qui ne soit pas satisfaite par $C_{ks}(L)$. Il existe donc un morphisme de catégorie $\varphi : X^* \to C_{ks}(L)$ tel que $\widehat{\varphi}(u) \neq \widehat{\varphi}(v)$. On note alors $V = \varphi(\mathrm{Ob}(X))$ l'ensemble des objets de $C_{ks}(L)$ dans le domaine image de $\varphi$, et de façon similaire on notera

$$E = \{(i, m, j) \in C_{ks}(L) \mid \text{ il existe } e \in X \ \varphi(e) = (i, m, j)\}$$

l'ensemble des flèches dans l'image de $\varphi$. On remarque naturellement que $E$ est inclus dans $V \times M_L \times V$.

Nous allons donc construire un morphisme de catégorie $\psi : X^* \to C_{ds}(L)$ tel que $\widehat{\psi}(u) \neq \widehat{\psi}(v)$. Afin de réaliser ceci, nous allons définir une fonction $\theta : V \to C_{ds}(L)$

qui envoie les objets de $V$ sur des objets de $C_{ds}(L)$ de telle façon que pour toute flèche $(i, m, j)$ de $E$, $(\theta(i), m, \theta(j))$ soit une flèche existante dans $C_{ds}(L)$.

Avant de définir $\theta$, nous donnons maintenant un premier lemme simple illustrant l'idée principale qui permet de définir l'application $\theta$.

**Lemme 3.37.** *Il existe un plus petit entier $i_V < ks$ tel que $\{i_V + 1, \ldots, i_V + s - 1 \bmod ks\} \cap V = \emptyset$.*

*Démonstration.* Puisque la taille de $X$ est $k$, la taille de $V = \varphi(\mathrm{Ob}(X))$ est au plus $k$. Alors la distance maximale entre deux objets consécutifs de $V$, où la distance entre deux objets $i$ et $j$ est $|j - i|$, est d'au moins $ks/k = s$. □

Nous faisons ici quelques abus de notations afin de garder un énoncé concis et clair. L'idée derrière cet entier $i_V$ est qu'il existe sur $C_{ks}(L)$ une plage de taille $s - 1$ ne contenant aucun élément de $V$. Il est important de noter que cette plage peut débuter par les objets ayant l'étiquette la plus élevée et terminer par les objets numérotés $0, 1, 2, \ldots$.

On définit alors l'application $\theta : V \to \mathrm{Ob}(C_{ds}(L))$ de la façon suivante :

$$\theta : \begin{cases} i \mapsto i \text{ si } i \leqslant i_V \\ i \mapsto ds + i - ks \text{ sinon.} \end{cases}$$

L'idée derrière cette application est que l'on aura $i = \theta(i)$ si $i$ apparaît avant la zone franche et $ks - i = ds - \theta(i)$ si $i$ apparaît après. Ainsi, nous allons pouvoir utiliser la propriété de l'indice de stabilité afin d'étendre les flèches passant par dessous cette zone et prouver leur existence dans $C_{ds}(L)$. Les autres flèches existeront naturellement car nous préservons les propriétés des bords.

**Lemme 3.38.** *Pour toute flèche $(i, m, j)$ de $E$, $(\theta(i), m, \theta(j))$ est une flèche de $C_{ds}(L)$.*

*Démonstration.* Soit $(i, m, j)$ une flèche de $E$. Il existe donc par définition un mot $u$ tel que $\eta_L(u) = m$ et $i + |u| = j \bmod ks$. On rappelle que si les égalités considérées sont écrites modulo un entier, les entiers $i$ et $j$ sont des valeurs exactes et seuls les mots peuvent être de longueur supérieure au modulo considéré. Nous divisons maintenant le problème en plusieurs sous-cas suivant la longueur de $u$.

- Si on a $|u| \geqslant s$, nous savons alors, par définition de l'indice de stabilité, que pour tout entier positif $\ell$, il existe un mot $u_\ell$ tel que $\eta_L(u_\ell) = \eta_L(u)$, de même longueur modulo $s$ et tel que $\ell s \leqslant |u_\ell| < (\ell + 1)s$. Il existe donc un entier $n$ tel que l'on a $|u| = j - i + nks$. Alors, puisque $\theta$ préserve la congruence modulo $s$, on peut conclure pour $\ell = ds$ que $|u_{ds}| = j - i \bmod ds$ et le triplet $(\theta(i), \eta_L(u_{ds}), \theta(j)) = (\theta(i), m, \theta(j))$ est une flèche de $C_{ds}(L)$.
- Si par contre $|u| < s$, nous séparons à nouveau le problème en sous-cas, suivant les possibles valeurs du doublet $(\theta(i), \theta(j))$ :

- Si $\theta(i) = i$ et $\theta(j) = j$, alors puisque $|u| < s$, on a $|u| = j - i = \theta(j) - \theta(i)$. Par conséquent $(\theta(i), m, \theta(j))$ est naturellement une flèche de $C_{ds}(L)$.
- Si on a simultanément $\theta(i) = ds+i-ks$ et $\theta(j) = ds+j-ks$, alors puisque la taille de $u$ est inférieure à $s$, on a $0 \leqslant j - i < s$ et par conséquent on a obligatoirement $i < j$ et donc $\theta(j) - \theta(i) = j - i$. Ainsi, on a $\theta(i) + |u| = \theta(j) \bmod ds$ et $(\theta(i), m, \theta(j))$ est donc une flèche de $C_{ds}(L)$.
- Si $\theta(i) = ds + i - ks$ et $\theta(j) = j$, on sait que l'on a $i + |u| = j + ks$. Donc $\theta(i) + |u| = ds + i - ks + |u| = j + ds$. Le même mot $u$ va étiqueter une flèche depuis $\theta(i)$ vers $j$ et ainsi $(\theta(i), m, \theta(j))$ est une flèche de $C_{ds}(L)$.
- Enfin, le cas $\theta(i) = i$ et $\theta(j) = ds + j - ks$ ne peut arriver car il implique que l'on ait simultanément $i \leqslant i_V$ et $j > i_V + s$ donc que $|u| = j - i > s \bmod ks$ ce qui contredit l'hypothèse stipulant que $|u|$ soit de longueur inférieure à $s$.

□

Nous sommes désormais prêts à définir un nouveau morphisme $\psi : X^* \to C_{ds}$. Ce morphisme a pour but de transférer l'hypothèse de non-satisfaction de l'équation $(X, u = v)$ supposée sur $C_{ks}(L)$ vers $C_{ds}(L)$. Soit $\psi$ défini de la façon suivante :

- Soit tout d'abord $\mathrm{Ob}(\psi)$ défini par $\theta \circ \mathrm{Ob}(\varphi)$.
- Nous devons également définir $\psi$ sur les flèches de $X$.

  Pour toute flèche $e$ de $X$, soit $(i, m, j) = \varphi(e)$ et on définit alors $\psi(e) = \big(\theta(i), m, \theta(j)\big)$. Cette application est bien définie, grâce au Lemme 3.38.

Nous devons maintenant prouver que cette application est un morphisme, dans le sens où elle s'étend de façon confluente sur les chemins finis de $X^*$. C'est alors le rôle du lemme suivant.

**Lemme 3.39.** *Soit $u$ un chemin de $X^*$. Si l'on pose $\varphi(u) = (i, m, j)$, on a alors $\psi(u) = \big(\theta(i), m, \theta(j)\big)$.*

*Démonstration.* Soit un chemin $u = u_1 \cdots u_n \in X^*$ et l'on pose $\varphi(u_\ell) = (i_\ell, m_\ell, j_\ell)$ pour tout $\ell \leqslant n$ et $\varphi(u) = (i, m, j)$. Par conséquent, pour tout $\ell \leqslant n$, $\psi(u_\ell) = \big(\theta(i_\ell), m_\ell, \theta(j_\ell)\big)$.

Cependant, $\varphi$ étant un morphisme de catégorie, on sait que pour tout $\ell < n$, on a $j_\ell = i_{\ell+1}$. Les flèches se composent alors naturellement et l'on a $\psi(u) = \big(\theta(i_1), m_1 \cdots m_n, \theta(j_n)\big) = \big(\theta(i), m, \theta(j)\big)$. □

Nous allons maintenant conclure la preuve de notre Théorème de Délai partiel et nous rappelons maintenant à cette fin que par hypothèse $C_{ks}(L)$ ne satisfait pas l'équation $(X, u = v)$ satisfaite par $\mathbf{gV}$, où $u$ et $v$ sont des chemins profinis. Plus précisément, l'équation est supposée mise en défaut par le morphisme de catégorie $\varphi$ et l'on a donc $\widehat{\varphi}(u) \neq \widehat{\varphi}(v)$. Les catégories $C_{ks}(L)$ et $C_{ds}(L)$ étant finies, on peut trouver deux chemins co-terminaux et finis $u', v'$ dans $X^*$ tels que $\varphi(u') = \widehat{\varphi}(u)$ et $\varphi(v') = \widehat{\varphi}(v)$, mais également $\psi(u') = \widehat{\psi}(u)$ et $\psi(v') = \widehat{\psi}(v)$.

Nous possédons désormais tous les outils pour aboutir à une contradiction. En effet, d'un coté les chemins $u'$ et $v'$ ont la même image $\varphi$ sur $C_{ks}(L)$ que $u$ et $v$ respectivement et on a donc $\varphi(u') = (i, m, j) \neq \varphi(v') = (i, m', j)$.

De l'autre, les chemins $u'$ et $v'$ sont identifiés sur $C_{ds}(L)$ aux chemins $u$ et $v$ par $\psi$ et l'on a donc $\psi(u) = \psi(v)$. Or, le Lemme 3.39 nous dit que $\psi(u) = \theta \circ \varphi(u') = \big(\theta(i), m, \theta(j)\big)$ et que $\psi(v) = \theta \circ \varphi(v') = \big(\theta(i), m', \theta(j)\big)$. Puisque $m$ est différent de $m'$, on a $\psi(u') \neq \psi(v')$ et la catégorie $C_{ds}(L)$ ne satisfait pas l'équation de chemins $(X, u = v)$ et donc ne peut pas non plus appartenir au global $\mathbf{gV}$, ce qui conclut notre preuve. □

#### Conséquences du Théorème

Le Théorème de Délai partiel est un outil important dans l'étude de la question du transfert de la décidabilité lors de l'ajout des prédicats modulaires, mais il ne permet pas, à lui seul, de conclure quant à la décidabilité des fragments $\mathcal{F}[\sigma, \mathrm{MOD}]$. Nous procédons maintenant à une discussion, et énonçons plusieurs résultats découlants de ce théorème et menant à la résolution partielle du problème considéré. En effet, la question de la décidabilité du global d'une variété n'est pas un problème nouveau et a déjà été étudié dans le passé. Les résultats existants nous permettent donc, maintenant que l'on sait répondre au premier sous-problème, de conclure si ce n'est dans tous les cas, au moins dans la plupart des cas régulièrement étudiés.

Tout d'abord, il est connu [Aui10, Til87] que le global $\mathbf{gV}$ d'une variété est décidable si, et seulement si, le produit semi-direct $\mathbf{V} * \mathbf{LI}$ l'est également. Ceci est dû à la construction du semigroupe consolidé d'une catégorie évoqué plus loin, construction qui conduit à un produit semi-direct avec un semigroupe de $\mathbf{LI}$. Réciproquement, le problème de la décidabilité de $\mathbf{V} * \mathbf{LI}$ se réduit par une catégorie dérivée adaptée à la décidabilité de $\mathbf{gV}$, de façon similaire au Théorème 3.30. Nous obtenons alors naturellement le corollaire suivant des Théorèmes 3.30 et 3.36.

**Corollaire 3.40.** *Soit* $\mathbf{V}$ *une variété de rang fini. Si la variété* $\mathbf{V} * \mathbf{LI}$ *est décidable, alors la variété* $\mathbf{V} * \mathbf{MOD}$ *l'est également.*

Le corollaire suivant est également prouvé de façon directe lorsque l'on considère le fait que le global de toute variété définissant le langage $(ab)^*$ est décidable [Til87].

**Corollaire 3.41.** *Soit* $\mathbf{V}$ *une variété de rang fini pouvant définir le langage* $(ab)^*$*. Alors* $\mathbf{V}$ *est décidable si, et seulement si,* $\mathbf{V} * \mathbf{MOD}$ *l'est également.*

## 3.4 Conclusion

Nous avons conclu notre étude de l'ajout des prédicats modulaires. Finalement, même si nous ne disposons pas de procédé automatique et universel prouvant la préservation de la décidabilité des fragments de logique lors de l'ajout des prédicats modulaires, nous avons développé toute une batterie de résultats nous permettant de conclure dans la

vaste majorité des cas des fragments de monoïdes ou de semigroupes largement étudiés dans la littérature.

Le tableau donné ci-après recense une partie des conséquences de nos théorèmes.

| | $\mathcal{B}\mathbf{\Sigma_1} = \mathbf{FO}^2_0$ | $\mathbf{FO}^2_k$ | $\mathbf{FO}^2$ | $\mathbf{FO}$ |
|---|---|---|---|---|
| $[<]$ | $\mathbf{J}$ [PP86, Tho82] | $\mathbf{V}_k$ [KW12, KS12] | $\mathbf{DA}$ [TW99] | $\mathbf{A}$ [MP71, Sch65] |
| $[<, \mathrm{LOC}]$ | $\mathbf{J} * \mathbf{LI}$ [Kna83] | $\mathbf{V}_k * \mathbf{LI}$ [KL13] | $\mathbf{LDA}$ [TW99] | $\mathbf{A}$ [MP71, Sch65] |
| $[<, \mathrm{MOD}]$ | $\mathbf{J} * \mathbf{MOD}$ [CPS06] | $\mathbf{V}_k * \mathbf{MOD}$ Cor 3.40 | $\mathbf{QDA}$ Thm 3.4 ou Thm 3.34 | $\mathbf{QA}$ Thm 3.34 ou [Str94, BCST92] |
| $[<, \mathrm{LOC}, \mathrm{MOD}]$ | $\mathbf{J} * \mathbf{LI} * \mathbf{MOD}$ Cor 3.41 ou [MPT00] | $\mathbf{V}_k * \mathbf{LI} * \mathbf{MOD}$ Cor 3.41 | $\mathbf{LDA} * \mathbf{MOD}$ Cor 3.41 | $\mathbf{QA}$ Thm 3.34 ou [Str94, BCST92] |

Nous tenons à préciser que les études menées dans les Section 3.2 et 3.3 représentent deux approches valables du problème. Le premier cas est une approche technique et spécifique d'un fragment donné. Elle utilise les propriétés internes du fragment ainsi que de la variété la caractérisant, afin d'extraire une caractérisation effective pour le fragment enrichi des prédicats modulaires. Les objets manipulés restent proches des outils traditionnels du domaine, tels que les jeux d'Ehrenfeucht-Fraïssé, les automates ou les monoïdes. On obtient une preuve technique et non généralisable ç d'autre fragments. Elle permet cependant d'obtenir des équivalents modulaires à la plupart des représentations existantes de notre fragment, en l'occurrence $\mathbf{FO}^2$.

D'un autre coté, la seconde approche, généraliste, permet d'obtenir en quelques théorèmes la décidabilité de l'enrichissement par les prédicats modulaires pour une infinité de variétés, s'appliquant par là-même à tout fragment équivalent à l'une de ces variétés. On utilise cependant des outils plus haut niveau tels que les catégories, et l'on doit alors s'appuyer sur de plus puissants résultats mathématiques. Le prix à payer est ainsi une possible perte algorithmique en ne profitant pas des caractéristiques spécifiques d'un fragment qui permettraient éventuellement d'affiner nos résultats.
Ces deux aspects sont complémentaires et si l'approche générale permet de conclure quant à la décidabilité d'une infinité de fragments enrichis par les prédicats modulaires, on lui préférera peut-être en pratique la première approche, qui se révélera souvent moins gourmande algorithmiquement.

Les grands absents de notre étude sont les variétés ordonnées, caractérisant les fragments non clos par complémentation tels que les demi-niveaux de la hiérarchie Dot-Depth $\Sigma_i[<]$. Les variétés ordonnées sont caractérisées par des inégalités profinies. Il faudrait alors compléter notre étude par la création d'inégalités de chemins profinis. Une autre direction complémentaire sera de relier le produit semi-direct de variétés par la variété $\mathbf{LI}$ à l'enrichissement de fragments par des prédicats locaux, comme il a été constaté dans la littérature existante, sans toutefois de preuve générique. Une approche similaire à celle faite dans ce chapitre est envisageable, mais la définition standard des prédicats

locaux, le successeur étant un prédicat binaire, ne permet pas d'adapter la preuve du Théorème 3.25. Une redéfinition des prédicats locaux de façon équivalente est alors à envisager.

La conclusion d'une telle étude serait alors que la décidabilité d'un fragment comportant l'ordre comme unique prédicat numérique suffirait à obtenir la décidabilité du même fragment, quels que soient les prédicats numériques réguliers de la signature.

# Transducteurs bidirectionnels apériodiques

Chapitre 4

Dans ce dernier chapitre, nous étudions les transducteurs bidirectionnels comme extension des automates. Là où traditionnellement les automates donnent une réponse binaire pour chaque mot, un transducteur va réaliser une relation depuis les mots sur un alphabet d'entrée vers les mots sur un alphabet de sortie. Ainsi, si les automates servent à modéliser des problèmes de vérification tels que la satisfaction d'une propriété par une machine, les transducteurs permettent de modéliser des programmes de traitement de données en ligne ou Streaming.

Techniquement, cela s'effectue en ajoutant à chaque transition une sortie, qui est un mot potentiellement vide. Un transducteur possède alors une tête de lecture qui parcourt le mot en entrée, et une tête d'écriture qui compose la sortie au fur et à mesure de l'exécution. Le modèle que nous considérerons dans ce chapitre est celui des transducteurs bidirectionnels, introduit notamment par Hopcroft et Ullman dans [HU67], où la tête de lecture peut se déplacer de gauche à droite mais également de droite à gauche. La tête d'écriture reste cependant unidirectionnelle et les sorties des transitions prises sont alors mises bout à bout pour construire la sortie.

Alors que dans le cas des automates, il a été prouvé que les modèles unidirectionnels et bidirectionnels ont la même expressivité [She59], les transducteurs bidirectionnels déterministes sont strictement plus expressifs que leur pendant unidirectionnels (il est en effet impossible de construire une transduction unidirectionnelle effectuant la fonction miroir). En particulier, les fonctions réalisées par les transducteurs bidirectionnels déterministes sont exactement les transformations régulières. Ce modèle forme donc un compromis efficace entre expressivité et décidabilité des propriétés. Il a été étudié notamment par Engelfriet et Hoogeboom dans [EH01]. Dans cet article, les auteurs établissent une caractérisation logique des fonctions réalisées par les transducteurs bidirectionnels grâce aux transductions de graphes définies par Courcelle (voir notamment [Cou94]). Plus récemment, il a été prouvé que les transducteurs bidirectionnels sont également équivalents aux Streaming Strings Transducers définis par Alur et Černỳ dans [AČ10].

Dans ce chapitre, nous proposons une extension de la notion de monoïde de transitions des automates unidirectionnels aux transducteurs bidirectionnels. Nous utilisons cette définition afin d'étudier les transducteurs bidirectionnels apériodiques en prouvant leur stabilité par composition. La preuve repose sur une suite de constructions largement inspirées de [AHU69, AU70, CJ77]. Nous prouvons que chacune de ces constructions

préserve l'apériodicité. Ce résultat est primordial dans l'étude de ce modèle, car il justifie de la robustesse de notre classe et donc de sa pertinence. Il s'agit d'une pierre solide sur laquelle il est nécessaire de s'appuyer afin de prouver l'équivalence de ce modèle avec les éventuelles restrictions d'autres modèles équivalents aux transducteurs bidirectionnels.

Nous tenons également à préciser que la notion de transducteur apériodique bidirectionnel avait déjà été définie par McKenzie, Schwentick, Thérien et Vollmer dans [MSTV00]. Cependant, même si nous nous accordons sur la définition de monoïde syntaxique, notre modèle diffère de leurs définitions qui se restreignaient aux transductions lettre à lettre, i.e. aux relations dont la longueur de(s) sortie(s) est égale à celle de l'entrée.

## 4.1 Transducteurs bidirectionnels

Nous posons dans cette première section les définitions basiques de notre modèle, ainsi que les outils algébriques permettant de définir l'apériodicité d'un transducteur.

### 4.1.1 Définitions

Nous donnons ici notre définition de transducteur bidirectionnel. Il est à noter que notre définition n'est pas la plus générique possible puisque nous nous limitons aux transducteurs déterministes. Il est cependant aisé d'en déduire une généralisation non déterministe, de même qu'il est aisé de retrouver les définitions classiques d'automate bidirectionnel ou de transducteur unidirectionnel en appliquant certaines restrictions aux objets considérés. Ces variantes seront précisées lorsque nous les utiliserons.

De plus, nous prenons le parti peut-être moins utilisé mais toutefois équivalent de séparer la fonction de transition de la fonction de production à des fins de clarté et de concision.

**Définition 4.1: Transducteur bidirectionnel**
Un *transducteur bidirectionnel* $\mathcal{A}$ est un tuple $\mathcal{A} = (Q, A, B, \delta, \gamma, q_0, F))$ où :

1. $Q$ est un ensemble fini désignant l'ensemble des états du transducteur.
2. $A$ est l'alphabet d'entrée du transducteur.
3. $B$ est l'alphabet de sortie du transducteur.
4. $\delta : Q \times (A \uplus \{\vdash, \dashv\}) \to Q \times \{-1, 0, +1\}$ est la fonction de transition et définit le comportement de l'état interne de la machine ainsi que les mouvements de la tête de lecture. L'alphabet est enrichi par deux symboles, appelés marqueurs de début et de fin de mot, servant à contrôler les déplacements de la tête de lecture aux bords du mot d'entrée.
5. $\gamma : Q \times (A \uplus \{\vdash, \dashv\}) \to B^*$ est la fonction de production. On ne lit ainsi qu'une lettre à la fois, mais on produit un mot fini qui peut être vide.
6. $q_0$ appartient à $Q$ et constitue l'état initial.

7. $F$ est un sous-ensemble de $Q$ qui constitue l'ensemble des états finaux.

Afin de pouvoir contrôler les déplacements de la tête de lecture, un mot donné en entrée d'un automate sera encadré en ajoutant $\vdash$ avant le mot et $\dashv$ après le mot. Ainsi pour tout état $q$, on a $\delta(q, \vdash) \in Q \times \{0, +1\}$ et $\delta(q, \dashv) \in Q \times \{-1, 0\}$ afin de garder la tête de lecture à l'intérieur du mot d'entrée.

La définition donnée ici se restreint aux transducteurs déterministes qui sont le sujet de notre étude. La définition d'un transducteur bidirectionnel dans le cas non déterministe s'obtient en remplaçant les fonctions de transition et de production par une unique relation sur $Q \times (A \uplus \{\vdash, \dashv\}) \times B^* \times Q \times \{-1, 0, +1\}$. Le regroupement des relations de transition et de production est dû au fait que la production dépend de la transition, donc de l'état cible de la transition. La séparation dans le cas déterministe est possible car l'état cible d'une transition est uniquement déterminé pour un état et une lettre donnés.

Un transducteur non déterministe sera dit *non-ambigu* si pour tout mot sur son alphabet d'entrée, il existe au plus une exécution valide du transducteur sur ce mot.

La simple définition de transducteur donnée ici ne permet pas de définir le comportement d'un transducteur sur un mot d'entrée. En particulier, la condition d'acceptation et de terminaison n'est pas évidente lorsque l'on considère une machine dont la tête de lecture est bidirectionnelle. On définit maintenant l'exécution d'un transducteur sur un mot d'entrée.

**Définition 4.2: Exécution d'un transducteur sur un mot**
Soient un transducteur bidirectionnel déterministe $\mathcal{A} = (Q, A, B, \delta, \gamma, q_0, F)$ et $u = u_1 \dots u_n$ un mot sur l'alphabet $A$. On pose également $u_0 = \vdash$ et $u_{n+1} = \dashv$.
L'exécution de $\mathcal{A}$ sur $u$, si elle existe, est l'unique suite $\big((q_0, k_0), (q_1, k_1), \dots, (q_m, k_m)\big)$, où les $q_i$ sont des états de $Q$ et les $k_i$ sont des entiers compris entre 0 et $n + 1$. Il s'agit de la suite des couples état/position de la tête de lecture du transducteur $\mathcal{A}$ sur l'entrée $\vdash u \dashv$.
On note $a_i$ la lettre $u_{k_i}$ de $\vdash u \dashv$. Ainsi on a $a_0 = \vdash$, et pour tout $i \leqslant m$, si $\delta(q_i, a_i) = (p, d)$ où $d \in \{-1, 0, +1\}$ est la direction de la tête de lecture, alors $q_{i+1} = p$ et $k_{i+1} = k_i + d$.

L'exécution s'arrête lorsque l'on atteint un état final sur le marqueur $\dashv$, i.e. si $q_m$ appartient à $F$ et que $a_m = \dashv$. Le résultat, que l'on notera $\mathcal{A}(u)$, est alors le mot $\gamma(q_0, a_0) \dots \gamma(q_m, a_m)$ qui est la concaténation des productions de chacune des étapes de l'exécution.

Afin d'illustrer ces définitions techniques, nous donnons maintenant un exemple de transducteur bidirectionnel.

**Exemple 4.1.** On considère le transducteur $\mathcal{A}$ représenté dans la Figure 4.1. Il possède trois états, l'état initial étant l'état 1 et seul l'état 3 étant final. Il fonctionne sur l'alphabet d'entrée $A = \{a, b\}$ qui est également son alphabet de sortie.
Une transition $a|b, -1$ signifie que l'on lit la lettre $a$, on sort le mot $b$ et la tête de lecture se déplace d'une position vers l'arrière. De façon habituelle, $\epsilon$ désigne le mot vide.

Ainsi ce transducteur commence par lire et recopier les $a$ qu'il voit, puis change d'état lorsqu'il rencontre un $b$ ou la fin du mot. Il parcourt alors les $a$ lus en sens inverse en écrivant des $b$. Lorsqu'il lit un $b$, ou qu'il voit le début du mot, il change d'état et lit une dernière fois le bloc de $a$. Il revient finalement à son état initial lorsqu'il revoit la lettre $b$. S'il voit le marqueur de fin de mot, l'exécution termine.

La transduction ainsi réalisée copie donc les blocs de $a$, puis écrit autant de $b$ qu'elle a lu de $a$. Étant donné un mot $u$ de la forme $a^{i_1}ba^{i_2}\dots ba^{i_n}$ où les $i_k$ sont des entiers éventuellement nuls, on a donc $\mathcal{A}(u) = a^{i_1}b^{i_1}\dots a^{i_n}b^{i_n}$.
Tout mot pouvant s'écrire comme succession de blocs éventuellement vides de $a$ entrecoupés de $b$, on a ainsi caractérisé la fonction réalisée par $\mathcal{A}$.

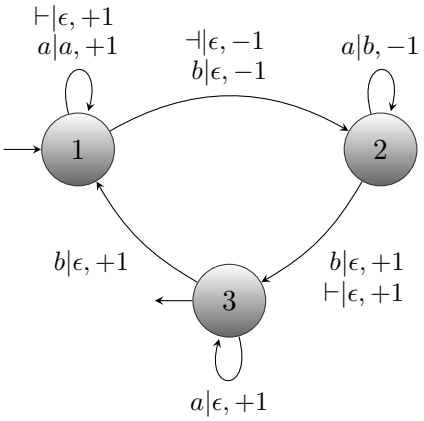



Figure 4.1: Le transducteur $\mathcal{A}$

### 4.1.2 Congruence bidirectionnelle

Nous définissons maintenant la notion de congruence syntaxique d'un automate bidirectionnel, ce qui conduit naturellement aux notions de timbre et monoïde de transitions. Il est essentiel de remarquer que l'on parlera dans cette partie de congruence d'un automate bidirectionnel et non d'un transducteur. En effet, la congruence que l'on définit ne dépend que de la structure de la machine, et non des productions. La congruence d'un transducteur sera la congruence induite par l'automate sous-jacent, c'est-à-dire l'automate obtenu en oubliant la fonction de production. Il est également à noter que contrairement au cas des automates traditionnels, il n'existe pas, pour un langage donné, d'algorithme de minimisation des transducteurs. Ainsi, si dans la littérature il est souvent question de timbre ou monoïde syntaxique d'un langage plutôt que d'un automate, nous nous contenterons dans le cadre de ce chapitre de parler de timbre ou monoïde d'un transducteur, et non de la fonction qu'il réalise.
La conséquence directe de ceci est qu'étant donné une fonction sur les mots, décider s'il existe un transducteur apériodique la réalisant est un problème ouvert.

Dans la suite, un *parcours partiel gauche à gauche* d'un automate sur un mot $u$ est un couple d'états $(q, q')$ tel que si l'on positionne la tête de lecture sur la première lettre de $u$ dans l'état $q$, alors l'exécution de $\mathcal{A}$ sur $u$ entraîne à terme la tête de lecture à sortir de $u$ par la gauche dans l'état $q'$ (voir Figure 4.2). On définit également de la même façon les parcours partiels gauche à droite, qui représentent le mode traditionnel de fonctionnement, et les parcours partiels droite à droite et droite à gauche.

**Définition 4.3:**
Soit $\mathcal{A} = (Q, A, \delta, q_0, F)$ un automate bidirectionnel. On définit alors quatre relations sur $A^*$ de la façon suivante pour tout couple de mots $u$ et $v$ :
- $u \sim_{ll} v$ si $u$ et $v$ ont les mêmes parcours partiels gauche à gauche.
- $u \sim_{lr} v$ si $u$ et $v$ ont les mêmes parcours partiels gauche à droite.
- $u \sim_{rl} v$ si $u$ et $v$ ont les mêmes parcours partiels droite à gauche.
- $u \sim_{rr} v$ si $u$ et $v$ ont les mêmes parcours partiels droite à droite.

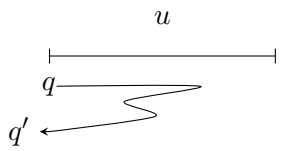



Figure 4.2: Un parcours partiel gauche à gauche $(q, q')$ sur un mot $u$.

Nous tenons à préciser que ces relations ont déjà été décrites par le passé, notamment dans [Péc85, Bir89]. En particulier, la notion de relation induite par les parcours gauche à gauche avait également déjà été définie par Shepherdson dans [She59] dans sa preuve de l'équivalence des automates classiques et bidirectionnels. Ces quatre relations nous permettent de définir une cinquième relation les rassemblant, également évoquée dans [Bir89].

**Définition 4.4: Relation syntaxique bidirectionnelle**
Soit $\mathcal{A}$ un automate bidirectionnel. Le mot $u$ est dit $\mathcal{A}$-équivalent à $v$, noté $u \sim_{\mathcal{A}} v$, si l'on a simultanément $u \sim_{ll} v$, $u \sim_{lr} v$, $u \sim_{rl} v$ et $u \sim_{rr} v$.

**Lemme 4.1.** *Soit $\mathcal{A}$ un automate bidirectionnel. Alors la relation $\sim_{\mathcal{A}}$ est une congruence d'indice fini.*

*Démonstration.* Nous prouvons tout d'abord que la relation $\sim_{\mathcal{A}}$ est une congruence. Soit un automate bidirectionnel $\mathcal{A}$ et soient quatre mots $u$, $v$, $u'$ et $v'$ tels que l'on ait $u \sim_{\mathcal{A}} u'$ et $v \sim_{\mathcal{A}} v'$. Prouvons alors que l'on a $uv \sim_{\mathcal{A}} u'v'$.

$uv \sim_{lr} u'v'$ Un parcours gauche à droite sur $uv$ peut s'écrire comme la composition d'un parcours gauche à droite sur $u$, une suite éventuellement vide de couples formés d'un parcours gauche à gauche sur $v$ suivi par un parcours droite à droite sur $u$, et finalement un parcours gauche à droite sur $v$. Puisque l'on a $u \sim_{\mathcal{A}} u'$ et $v \sim_{\mathcal{A}} v'$, alors $uv$ et $u'v'$ ont les mêmes parcours gauche à droite.

$uv \sim_{ll} u'v'$ Un parcours gauche à gauche sur $uv$ est de la forme suivante : soit il s'agit d'un parcours gauche à gauche sur $u$ ne visitant pas $v$, soit il peut s'écrire comme la composition d'un parcours gauche à droite sur $u$, une suite éventuellement vide de doublets formés d'un parcours gauche à gauche sur $v$ suivi par un parcours droite à droite sur $u$, un parcours gauche à gauche sur $v$ et finalement un parcours droite à gauche sur $u$. Les parcours gauche à gauche de $uv$ sont donc les mêmes que les parcours gauche à gauche sur $u'v'$.

Des arguments symétriques permettent de prouver que l'on a $uv \sim_{rr} u'v'$ et $uv \sim_{rl} u'v'$, et que l'on a donc $uv \sim_{\mathcal{A}} u'v'$.

Nous prouvons que la congruence $\sim_{\mathcal{A}}$ est d'indice fini en remarquant qu'une classe d'une relation $\sim_{ll}$, $\sim_{lr}$, $\sim_{rl}$ ou $\sim_{rr}$ est définie par un ensemble de paires d'états. L'ensemble des états étant fini, il existe au plus $2^{|Q|^2}$ classes pour chaque relation, et la congruence $\sim_{\mathcal{A}}$ est donc d'indice fini. □

**Remarque 4.1.** Le monoïde de transitions d'un automate bidirectionnel $\mathcal{A}$ est défini comme le monoïde quotient $A^*/\sim_{\mathcal{A}}$. Notre définition de monoïde de transitions étend la définition de monoïde de transitions dans le cas des automates unidirectionnels. En effet, dans le cas d'un automate unidirectionnel, il n'existe aucun parcours partiel gauche à gauche, droite à gauche ou droite à droite et donc pour tout couple de mots $u$ et $v$, on a $u \sim_{ll} v$, $u \sim_{rl} v$ et $u \sim_{rr} v$ et donc $\sim_{\mathcal{A}}=\sim_{lr}$.

Nous pouvons maintenant définir les *automates bidirectionnels apériodiques*.

**Définition 4.5: Automates bidirectionnels apériodiques**
Un automate bidirectionnel $\mathcal{A}$ est *apériodique* si son monoïde syntaxique $A^*/\sim_{\mathcal{A}}$ est apériodique, i.e. s'il vérifie l'équation $x^\omega = x^{\omega+1}$.

Une relation sur $A^*$ sera dite *apériodique* s'il existe un entier $n$ tel que pour tout mot $u$ de $A^*$, $u^n$ est en relation avec $u^{n+1}$. Le plus petit entier $n$ vérifiant cette propriété est appelé *indice d'apériodicité* de la relation. Il est important de noter que le monoïde des transitions d'un automate bidirectionnel $\mathcal{A}$ sera apériodique si, et seulement si, chacune des quatre relations $\sim_{ll}$, $\sim_{lr}$, $\sim_{rl}$ et $\sim_{rr}$ est apériodique. Nous donnons maintenant une première propriété de ces relations, qui sera utilisée dans la suite pour prouver l'apériodicité de nos constructions.

**Lemme 4.2.** *Si les relations $\sim_{lr}$ et $\sim_{rl}$ sont apériodiques, alors les relations $\sim_{ll}$, $\sim_{rr}$ et donc $\sim_{\mathcal{A}}$ sont également apériodiques.*

*Démonstration.* Nous allons prouver que si $\sim_{lr}$ et $\sim_{rl}$ sont apériodiques avec un indice d'apériodicité $n$, alors les relations $\sim_{ll}$ et $\sim_{rr}$ sont apériodiques avec un indice d'apériodicité au plus $n+2$. Premièrement remarquons que tout parcours gauche à gauche sur $u^{n+2}$ est également un parcours gauche à gauche sur $u^{n+3}$. En effet, ceci est dû à la remarque plus générale : si $(q, q')$ est un parcours gauche à gauche sur un mot $v$, alors pour tout mot $w$, $(q, q')$ est un parcours gauche à gauche sur $vw$.

Considérons maintenant un parcours gauche à gauche sur $u^{n+3}$ et prouvons qu'il apparaît sur $u^{n+2}$. Pour ce faire, nous isolons symboliquement une itération de $u$ de chaque coté de $u^{n+3}$ (voir Figure 4.3). On peut alors décomposer le parcours en succession de parcours sur $u^{n+1}$ et de parcours sur les itérations de $u$ aux frontières. Alors, puisque les relations $\sim_{lr}$ et $\sim_{rl}$ ont un indice d'apériodicité $n$, les parcours gauche à droite et droite à gauche sur $u^{n+1}$ apparaissent sur $u^n$ et peuvent donc être raccourcis. Ensuite, considérons les parcours gauche à gauche et droite à droite apparaissant sur $u^{n+1}$ dans la décomposition, mais n'étant éventuellement pas présents sur $u^n$. Lors du passage de $u^{n+3}$ à $u^{n+2}$, ces parcours gauche à gauche sont alors déplacés sur les $n+1$ dernières itérations, et les parcours droite à droite sont déplacés sur les $n+1$ premières itérations (voir Figure 4.3).

Nous pouvons ainsi recomposer le parcours initial sur $u^{n+2}$, prouvant ainsi l'apériodicité de $\sim_{ll}$. La preuve se conclut en remarquant que les mêmes arguments seront valides pour la relation $\sim_{rr}$, de par la nature symétrique des définitions. □

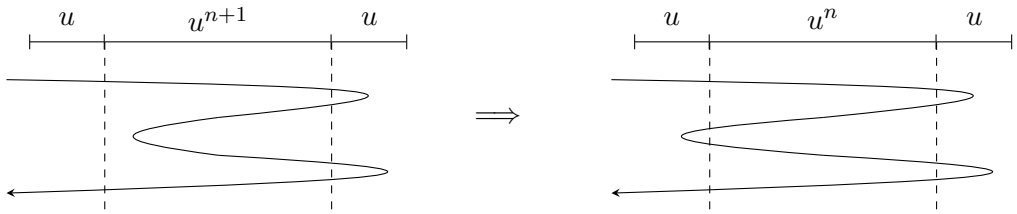



Figure 4.3: Les parcours transversaux sont raccourcis et les boucles sont décalées sur l'itération isolée.

## 4.2 Composition des transducteurs bidirectionnels apériodiques

Ayant défini la congruence syntaxique d'un automate bidirectionnel, nous tenons maintenant une définition des transducteurs bidirectionnels apériodiques. Cependant afin de justifier l'existence de notre classe, et avant de chercher à en obtenir une caractérisation logique, nous devons prouver sa robustesse. Dans cette partie, nous prouvons la clôture par composition des relations réalisées par des transducteurs bidirectionnels apériodiques, dans le sens où si deux transducteurs apériodiques $\mathcal{A} : A^* \to B^*$ et

$\mathcal{A}' : B^* \to C^*$ réalisent des relations composables, on désire être capable de construire un transducteur apériodique $\mathcal{C} : A^* \to C^*$ tel que pour tout mot $u$ de $A^*$, $\mathcal{C}(u) = \mathcal{A}' \circ \mathcal{A}(u)$.

Le schéma de preuve que nous utilisons provient de la preuve de la clôture par composition des automates à ballon, parue dans [HU67], et également présente dans [CJ77] Il s'agit de décomposer les deux transducteurs en éléments plus simples. La difficulté supplémentaire est ici de prouver que chacune des constructions intermédiaires préserve l'apériodicité. Nous décrivons maintenant le schéma de preuve, qui sera appliqué dans la suite de cette partie.

Soient deux transducteurs bidirectionnels apériodiques $\mathcal{A} : A^* \to B^*$ et $\mathcal{A}' : B^* \to C^*$. Nous définissons les tranches (ou slices dans la littérature anglophone) comme une lettre suivie d'une suite finie d'états. Étant donné un transducteur et un mot, une tranche décrit la suite des états atteints par le transducteur sur une position donnée. On construit alors un transducteur unidirectionnel non-ambigu $\mathcal{S}(\mathcal{A})$ (Section 4.2.1) qui, prenant en entrée un mot de $A^*$, construit la suite des tranches décrivant le comportement de $\mathcal{A}$ sur $u$. On construit parallèlement un transducteur bidirectionnel déterministe $\widehat{\mathcal{A}'}$ (Section 4.2.2) qui, prenant en entrée un mot de tranches correspondant à une sortie de $\mathcal{S}(\mathcal{A})(u)$, produit le mot $\mathcal{A}' \circ \mathcal{A}(u)$. Nous décomposons ensuite le transducteur des tranches comme la composition de deux transducteurs déterministes $\mathcal{L}$ et $\mathcal{R}$, l'un ayant une tête de lecture n'allant que vers la droite, et le second ayant une tête de lecture n'allant que vers la gauche (Section 4.2.3). Enfin, étant donnés un transducteur bidirectionnel $\mathcal{A}'$ (resp. $\widehat{\mathcal{A}'_{\mathcal{L}}}$) et un transducteur unidirectionnel $\mathcal{L}$ (resp. $\mathcal{R}$) , nous construisons un transducteur bidirectionnel $\widehat{\mathcal{A}'_{\mathcal{L}}}$ (resp. $\mathcal{C}$) réalisant la composition de ces deux machines (Section 4.2.4). Le schéma de preuve est résumé dans la Figure 4.4 où l'on décrit les étapes successives permettant d'aboutir à un transducteur bidirectionnel réalisant la composition.

$$
\begin{array}{lcccl}
 & & \mathcal{A}' & \circ & \mathcal{A} \\
\text{Sections 4.2.1 et 4.2.2 :} & = & \widehat{\mathcal{A}'} & \circ & \mathcal{S}(\mathcal{A}) \\
\text{Section 4.2.3 :} & = & \widehat{\mathcal{A}'} & \circ & \mathcal{L} \circ \mathcal{R} \\
\text{Section 4.2.4 :} & = & \widehat{\mathcal{A}'_{\mathcal{L}}} & \circ & \mathcal{R} \\
\text{Section 4.2.4 :} & = & & \mathcal{C} &
\end{array}
$$

Figure 4.4: La suite de constructions de notre preuve de composabilité.

### Transducteur normalisé

Avant de débuter notre suite de constructions, nous donnons notre définition de transducteur normalisé.

Un transducteur normalisé nous permettra de correctement contrôler notre position dans l'exécution du transducteur décrit par les mots de tranches donnés en entrée, et surtout de détecter efficacement lorsque l'arrivée du parcours aux extrémités.

La Proposition 4.3 prouvera que les transducteurs normalisés n'entraînent pas de perte de généralité. Dans la suite, nous considérerons donc que les transducteurs que nous considérons sont normalisés.

**Définition 4.6: Transducteur normalisé**
Un transducteur bidirectionnel déterministe est dit *normalisé* s'il n'existe aucune transition vers l'état initial et aucune transition sortant d'un état final.
Formellement, si $\mathcal{A} = (Q, A, B, \delta, \gamma, i, F)$, alors $\mathcal{A}$ est normalisé si
- $\delta(Q, A) \cap \big(\{i\} \times \{-1, 0, 1\}\big) = \emptyset$,
- $\delta(F, A) = \emptyset$.

On donne maintenant la proposition prouvant que l'on pourra toujours ne considérer que des transducteurs normalisés sans perte de généralité.

**Proposition 4.3.** *Pour tout transducteur bidirectionnel déterministe on peut effectivement construire un transducteur normalisé réalisant la même fonction sur les mots.*
*De plus cette construction préserve l'apériodicité.*

*Démonstration.* Soit $\mathcal{A} = (Q, A, B, \delta, \gamma, i, F)$ un transducteur bidirectionnel déterministe.
On définit $\mathcal{A}_n = (Q \uplus \{i', f\}, A, B, \delta', \gamma', i', \{f\})$ de la façon suivante :
- $i'$ est un état neuf, l'unique état initial.
- $f$ est un état neuf qui devient le seul état final.
- $\delta'$ est égale à $\delta$ sur $Q$ et est étendue sur les nouveaux états par :
  - $\delta'(i', \vdash) = (i, 0)$,
  - $\forall q \in F\ \delta'(q, \dashv) = (f, 0)$.

  Il s'agit simplement d'ajouter des mouvements stationnaires afin de désambiguïser le début et la fin de l'exécution. Cela préserve le déterminisme puisque si le transduction $\mathcal{A}$ est sur le marqueur de fin dans un état final, l'exécution s'arrête. Le déterminisme de $\mathcal{A}$ assure donc que la fonction de transitions n'y est pas définie.
- $\gamma'$ est égal à $\gamma$ sur les états déjà existant, et étendu de façon triviale sur les nouveaux états en ne produisant que le mot vide.

Le transducteur $\mathcal{A}_n$ ainsi construit est normalisé et déterministe. Si de plus $\mathcal{A}$ est apériodique, alors puisque la construction n'ajoute aucune exécution périodique, $\mathcal{A}_n$ sera également apériodique. □

### 4.2.1 Le transducteur des tranches

Dans cette partie nous définissons les tranches d'un transducteur, et construisons un transducteur capable de calculer la suite de tranches apparaissant lors de l'exécution d'un transducteur bidirectionnel sur un mot donné.

Le transducteur construit est unidirectionnel mais non-ambigu, contrairement aux transducteurs déterministes que l'on a en entrée. Une machine non-ambigüe est une machine semi-déterministe, dans le sens où certains de ces états peuvent être non-déterministe, mais pour tout mot donné en entrée, il n'existe au plus qu'un seul calcul acceptant. Notre construction doit deviner quelles sont les tranches composant l'exécution du transducteur donné sur le mot d'entrée, mais étant donné que l'on considère initialement des transducteurs déterministes, il n'existe au plus qu'un seul mot de tranches possible comme résultat. La Section 4.2.3 nous permet de nous ramener au cas déterministe et ainsi de conclure.

### Définition des tranches

Comme énoncé plus haut, une tranche est une suite d'états décrivant l'ensemble des transitions prises à une position donnée lors de l'exécution d'un automate bidirectionnel sur un mot, ainsi que l'ordre dans lequel ces transitions apparaissent dans le parcours. Il s'agit ainsi d'une information locale et complète dépendant de l'automate, mais aussi du mot d'entrée, et l'on peut entièrement décrire l'exécution à partir du mot des tranches pour chacune des positions.

Il s'agit d'une notion classique des machines bidirectionnelles utilisée notamment dans [AU70, Car12]. Nous donnons ci-dessous la définition formelle d'une tranche.

**Définition 4.7: Tranche d'un automate bidirectionnel**
Soit $\mathcal{A} = (Q, A, \delta, i, F)$ un automate bidirectionnel. On appelle *tranche* de $\mathcal{A}$ une suite $(a, q_1, \dots, q_k)$ où $a$ appartient à $A \cup \{\vdash, \dashv\}$, et pour tout $i \leqslant k$, $q_i \in Q$.
Nous restreignons les tranches aux suites satisfaisant les propriétés suivantes :
- pour tout $i$, $\delta$ est défini pour $(q_i, a)$ avec $a \in A$, ou $i = k$, $a = \dashv$ et $q_i$ est final.
- les états $q_i$ sont deux à deux distincts. Ainsi on a $k \leqslant |Q|$ et un nombre fini de tranches,
- si $\delta(q_i, a) = (q, 0)$ alors $q_{i+1} = q$.

Nous noterons $S_{\mathcal{A}}$ l'ensemble des tranches d'un automate $\mathcal{A}$.

On notera que l'on a défini les tranches pour les automates plutôt que pour les transducteurs. Cependant, dans le cas considéré ici des transducteurs déterministes, nous sommes capable de retrouver la production à partir de l'état et de la lettre lue et donc l'information que nous considérons est suffisante.

Étant donnés un mot et son parcours sur un automate déterministe, deux tranches consécutives, i.e. deux tranches décrivant le parcours à deux positions consécutives, sont intrinsèquement liées par les transitions allant d'une position à l'autre. Ainsi la tranche choisie à une position détermine en partie la tranche de la position suivante. Cela permet de limiter le non-déterminisme du transducteur des tranches, mais également d'écarter de nouveaux comportements non désirables.

Ainsi, si l'on considère un automate bidirectionnel $\mathcal{A}(Q, A, \delta, i, F)$ et deux tranches $s = (a, q_1, \dots, q_k)$ et $t = (b, p_1, \dots, p_l)$ de $\mathcal{A}$, on dira que $t$ est un *successeur* de $s$, où

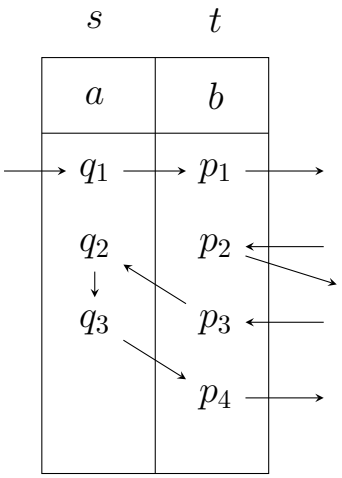



Figure 4.5: Cohérence des tranches.

que $s$ est *cohérent* avec $t$, si le mot $st$ forme des parcours partiels sur $ab$. De façon plus formelle, $t$ est un successeur de $s$ si la paire de tranches satisfait la propriété suivante :

– Si $q_{i_1}, \ldots, q_{i_m}$ est la sous-suite de $s$ (resp. $t$) des états allant à droite (resp. gauche) par $\delta$ lorsque l'on lit la lettre $a$ (resp. $b$), alors la suite $p_{i_1}, \ldots, p_{i_m}$ définie par $\delta(q_{i_j}, a) = (p_{i_j}, +1)$ (resp. $\delta(q_{i_j}, b) = (p_{i_j}, -1)$) correspond exactement à la sous-suite de $t$ (resp. $s$) des états tels que la direction de $\delta(p_{i_j-1}, b)$ (resp. $\delta(p_{i_j-1}, a)$) est $-1$ (resp. $+1$). Pour $i_j = 1$, la direction est définie par défaut à $-1$.

L'idée derrière la cohérence, illustrée par la Figure 4.5, est que tout mouvement entre les deux tranches initié par l'une des tranches apparaît sur l'autre tranche, et dans le même ordre. En effet, si l'on quitte une tranche par la gauche, alors l'exécution ne peut revenir sur cette position que par la gauche, de par la nature linéaire de la structure d'entrée. Le respect de l'ordre exprimé par la suite de la tranche implique également que le retour sera fait dans le bon état. Dans l'exemple générique donné par la Figure 4.5, la tranche $s$ stipule que l'on arrive en $a$ dans l'état $q_1$. Par déterminisme de nos transducteurs, il n'existe donc au plus qu'une transition possible. Alors la tranche $t$ ne peut être cohérente avec la tranche $s$ que si son premier état $p_1$ est tel que $\delta(q_1, a) = (p_1, +1)$. De même, lorsque le transducteur revient à la position $a$, ce doit être dans le second état de $s$, qui est tel que $\delta(p_3, b) = (q_2, -1)$.

Soient un automate bidirectionnel déterministe $\mathcal{A}$ et un mot de tranches $u = s_0 \cdots s_n$ sur $S_\mathcal{A}$ avec les marqueurs de fin de mots $\vdash$ et $\dashv$. Si pour tout $i$ on a $s_i = (a_i, q_{i_1}, \ldots, q_{i_{|s_i|}})$, on appelle le mot $a_1 \cdots a_{n-1}$ le *mot sous-jacent* à $u$, c'est-à-dire le mot dont $u$ prétend décrire l'exécution. Réciproquement, si $v$ est un mot de $A^*$, alors il existe un unique mot de tranches $u$ dont les tranches sont cohérentes deux à deux et ayant $v$ comme mot sous-jacent. On appelle ce mot $u$ *mot de tranches valide* de $v$, et il décrit entièrement l'exécution de $\mathcal{A}$ sur $v$.

**Exemple 4.2.** À titre d'exemple, nous donnons le mot des tranches valide pour le mot $u = aababb$ et l'automate $\mathcal{A}$ donné à la Figure 4.1.

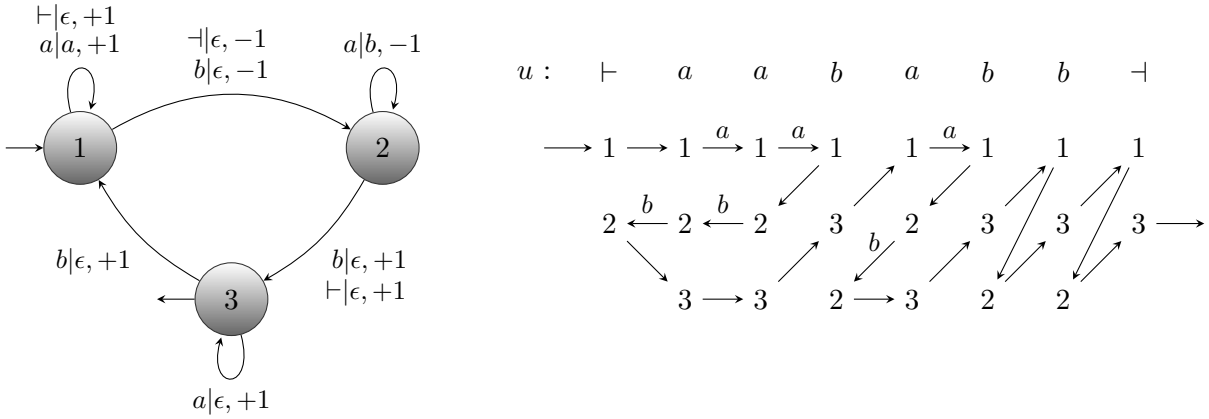



Figure 4.6: Ici, $\mathcal{A}(u) = aabbab$.

### Transducteur des tranches

Étant donné un transducteur bidirectionnel apériodique et déterministe, nous allons maintenant montrer comment construire un transducteur unidirectionnel apériodique et non-ambigu produisant les mots de tranches valides du transducteur d'entrée. Nous commençons par donner la construction et prouverons les propriétés annoncées ensuite.

**Définition 4.8: Transducteur des tranches**
Soit $\mathcal{A} = (Q, A, B, \delta, \gamma, i, F)$ un transducteur bidirectionnel déterministe.
On définit alors le transducteur unidirectionnel non-déterministe $\mathcal{S}(\mathcal{A}) = (S_\mathcal{A} \uplus \{final\}, A, S_\mathcal{A}, \Delta, \gamma', I, \{final\})$ de la façon suivante :

- $S_\mathcal{A} \uplus \{final\}$ est l'ensemble des états de $\mathcal{S}(\mathcal{A})$ et correspond à l'ensemble des tranches de $\mathcal{A}$ auquel on ajoute un nouvel état final.
- L'alphabet d'entrée est $A$, le même que celui de $\mathcal{A}$.
- L'alphabet de sortie est l'ensemble des tranches de $\mathcal{A}$, et correspond à l'ensemble des états, privé de l'état final.
- L'ensemble des états initiaux $I$ est l'ensemble des tranches étiquetées par le marqueur de début $\vdash$ et telles que le premier élément de la suite soit $i$.
- L'ensemble des états finaux est réduit à l'unique nouvel état appelé final.
- La relation de transition $\Delta \subseteq S_\mathcal{A} \times A \times S_\mathcal{A}$ est l'ensemble des triplets $(s, a, s')$ tels que $s$ est étiqueté par $a$ et $s'$ est un successeur de $s$.
  On ajoute à cette relation les triplets $(s, \dashv, final)$ où $s$ est une tranche étiquetée par le marqueur de fin $\dashv$ telle que seul le dernier état de la suite appartienne à $F$.

– La fonction de production $\gamma'$ écrit simplement l'état courant de $S(\mathcal{A})$ sur toutes les positions, y compris les marqueurs de début et de fin de mot.

Après avoir défini notre construction, nous devons prouver sa correction. Nous commençons par prouver sa validité. La conséquence principale est que le transducteur produit sera non-ambigu si le transducteur bidirectionnel donné en entrée est déterministe.

**Lemme 4.4.** *Soient $\mathcal{A}$ un transducteur bidirectionnel déterministe sur $A^*$ et $v$ un mot de $A^*$, alors $\mathcal{S}(\mathcal{A})(v)$ est exactement le mot des tranches valide de $v$.*

*Démonstration.* Soient $\mathcal{A}$ un transducteur bidirectionnel déterministe sur $A^*$ et $v$ un mot de $A^*$ tel que $\mathcal{A}$ est défini sur $v$.
Tout d'abord, remarquons que le mot de tranches valide de $v$ représente une exécution possible de $\mathcal{S}(\mathcal{A})$ ainsi que la production associée. Le mot $v$ possède donc une image par $\mathcal{S}(\mathcal{A})$.
Soit maintenant $u$ la production d'une exécution de $\mathcal{S}(\mathcal{A})$ sur $v$. Par construction de $\mathcal{S}(\mathcal{A})$, alors $v$ est le mot sous-jacent de $u$, et les tranches de $u$ sont cohérentes deux à deux. Le mot de tranches $u$ décrit alors une exécution de $\mathcal{A}$, commençant par l'état initial $i$ et terminant dans un état final. Le transducteur $\mathcal{A}$ étant déterministe, $u$ décrit l'unique exécution de $\mathcal{A}$ sur $v$ et est donc son mot de tranche valide. Sur les mots où $\mathcal{A}$ est défini, le transducteur $\mathcal{S}(\mathcal{A})$ possède donc une unique exécution, produisant le mot de tranches valide.
Si maintenant $\mathcal{A}$ ne possède pas d'exécution sur un mot $v$, alors $\mathcal{S}(\mathcal{A})$ ne pourra trouver de mot de tranches valide ayant $v$ comme mot sous-jacent, et alors $\mathcal{S}(\mathcal{A})(v)$ n'est pas défini. □

La production du transducteur $\mathcal{S}(\mathcal{A})$ étant exactement la suite des états visités, le fait que la relation réalisée par $\mathcal{S}(\mathcal{A})$ soit fonctionnelle implique le corollaire suivant.

**Corollaire 4.5.** *Soit $\mathcal{A}$ un transducteur bidirectionnel déterministe.*
*Alors le transducteur $\mathcal{S}(\mathcal{A})$ est non-ambigu.*

Ayant prouvé la correction de notre construction, nous nous attelons maintenant à prouver qu'elle préserve l'apériodicité. Cette preuve d'apériodicité sera la plus simple à exécuter du fait des liens étroits entre les exécutions de $\mathcal{A}$ et celles de $\mathcal{S}(\mathcal{A})$.

**Proposition 4.6.** *Soit $\mathcal{A}$ un transducteur bidirectionnel apériodique.*
*Alors le transducteur $\mathcal{S}(\mathcal{A})$ est également apériodique.*

*Démonstration.* Soit $\mathcal{A}$ un transducteur apériodique, $n$ son indice d'apériodicité et $u$ un mot d'entrée de $\mathcal{A}$. Prouvons alors que $u^n \sim_{\mathcal{S}(\mathcal{A})} u^{n+1}$.
À cette fin, considérons un parcours partiel de $\mathcal{S}(\mathcal{A})$ sur $u^n$. Soient alors les tranches, donc les états, produites aux deux extrémités de $u^n$. Ces deux tranches décrivent alors une suite de parcours partiels sur $u^n$ correspondant aux passages de la tête de lecture

sur $u^n$ dans l'exécution décrite.
Puisque l'on a $u^n \sim_{\mathcal{A}} u^{n+1}$, chacun des parcours partiels décrit par ces tranches de début et de fin existe également sur $u^{n+1}$. Il existe alors un parcours partiel du transducteur des tranches sur $u^{n+1}$ sortant du mot dans un état correspondant à une tranche décrivant les mêmes parcours partiels, et possédant donc la même suite d'états.
Nous pouvons ensuite conclure notre preuve grâce à la symétrie de la relation $\sim$. □

### 4.2.2 Le transducteur $\widehat{\mathcal{A}'}$

Après avoir défini les tranches d'un transducteur bidirectionnel, et expliqué leur fonctionnement ainsi qu'une méthode pour construire le transducteur des tranches, nous allons maintenant expliquer comment, à partir de l'information donnée par les tranches, nous pouvons retrouver l'exécution du transducteur original et l'utiliser pour produire le résultat de la composition. L'idée principale est que les tranches fournissent l'information nécessaire pour construire l'exécution, mais également pour reculer dans l'exécution, ce qui s'avère nécessaire lorsque l'on considère la composition de machines bidirectionnelles.

#### Construction de $\widehat{\mathcal{A}'}$

Nous donnons maintenant la construction du transducteur $\widehat{\mathcal{A}'}$. Étant donnés deux transducteurs composables, ce transducteur produit, à partir d'un mot de tranches valide, le résultat de la composition des deux transducteurs sur le mot sous-jacent au mot donné en entrée.

La nécessité d'utiliser les tranches provient de la nature bidirectionnelle des machines. En effet, la composition s'effectue traditionnellement à l'aide d'une mémoire tampon finie qui stocke l'exécution au fil de l'eau. Ici, étant donné que le second transducteur peut a priori revenir dans l'exécution aussi loin que possible, il devient nécessaire de pouvoir revenir dans l'exécution du premier transducteur afin de conserver une mémoire tampon finie. Les transducteurs considérés n'étant pas co-déterministes, nous utilisons les tranches, qui permettent de retrouver le parcours effectué sur le mot d'entrée de façon finie.

Le théorème suivant donne la construction formelle ainsi que les explications nécessaires à son fonctionnement. La seconde partie de cette section s'attache à prouver que la construction donnée préserve l'apériodicité.

**Théorème 4.7.** *Soient $\mathcal{A}$ et $\mathcal{A}'$ deux transducteurs bidirectionnels déterministes et composables, avec $\mathcal{A}$ un transducteur normalisé, et soit $\mathcal{S}(\mathcal{A})$ le transducteur des tranches tel que défini dans la section précédente.*
*Alors on peut effectivement construire un transducteur déterministe $\widehat{\mathcal{A}'}$ tel que pour tout mot $u$, $\widehat{\mathcal{A}'} \circ \mathcal{S}(\mathcal{A})(u) = \mathcal{A}' \circ \mathcal{A}(u)$.*

*Démonstration.* L'idée derrière cette construction est d'utiliser les mots de tranches valides de $\mathcal{A}$ afin de simuler son exécution. Le transducteur enregistre des parties du par-

cours dans une mémoire tampon, et utilise cette mémoire pour simuler $\mathcal{A}'$ via une tête de lecture symbolique. Cette mémoire tampon est utilisée pour enregistrer les productions de $\mathcal{A}$ pour une transition donnée. Sa taille est ainsi bornée à la taille maximale des productions de $\mathcal{A}$ Lorsque la simulation de $\mathcal{A}'$ sort de la partie enregistrée dans la mémoire, il utilise l'information contenue dans les tranches pour simuler $\mathcal{A}$, dans un sens ou dans l'autre. La notion principale est que si $\mathcal{A}'$ sort de la mémoire par la droite, cela correspond à une progression dans l'exécution de $\mathcal{A}$ et il suffit de suivre la fonction de transition de $\mathcal{A}$. Réciproquement si $\mathcal{A}'$ sort de la mémoire par la gauche, on doit alors revenir dans l'exécution de $\mathcal{A}$ et calculer la production précédente. Le fonctionnement du transducteur $\widehat{\mathcal{A}'}$ est schématisé dans la Figure 4.7.

Les transitions du transducteur $\widehat{\mathcal{A}'}$ peuvent être divisées en quatre types que nous décrivons maintenant. Les transitions de type 1 seront utilisées lorsque la tête de lecture de la mémoire tampon n'est pas sortie des limites de la mémoire. On est alors capable de continuer la simulation de $\mathcal{A}'$, et le transducteur $\widehat{\mathcal{A}'}$ suivra des transitions de ce type jusqu'à épuisement de la mémoire tampon. Dans ce mode, la partie de $\widehat{\mathcal{A}'}$ servant à simuler $\mathcal{A}$ ne change pas.
Les transitions de type 2 sont utilisées lorsque la tête de lecture de la mémoire tampon sort de ses limites par la droite. Cela correspond à une progression dans l'exécution de $\mathcal{A}$. Le transducteur $\widehat{\mathcal{A}'}$ poursuit alors la fonction de transition de $\mathcal{A}$ là où elle s'était arrêtée jusqu'à remplir la mémoire avec un mot non vide.
Les transitions de type 3 apparaissent lorsque la tête de lecture de la mémoire tampon s'échappe par la gauche. Le transducteur doit alors déterminer quelle était la transition précédente dans l'exécution de $\mathcal{A}$. Le transducteur $\widehat{\mathcal{A}'}$ lit alors la tranche accessible et détermine sa position dans le parcours en comparant la tranche avec son état interne de $\mathcal{A}$. Il peut alors déterminer la direction précédemment prise en regardant la direction prise la dernière fois que le calcul de $\mathcal{A}$ est passé par cette position. Si le parcours était stationnaire, il peut déterminer la transition prise précédemment, ou suivre cette direction et utiliser le dernier type de transition afin de remplir sa mémoire. Cela crée un délai d'une étape entre l'exécution simulée de $\mathcal{A}$ et l'état interne de $\mathcal{A}$ enregistré par $\widehat{\mathcal{A}'}$.
Enfin, les transitions de type 4 sont utilisées après les transitions précédentes. En lisant la tranche actuelle, il peut déterminer l'état, et donc la transition qui a menée à l'état interne stocké dans $\mathcal{A}$. On peut alors remplir la mémoire interne si la production à cette transition était non vide, ou recommencer si nécessaire. Le transducteur $\widehat{\mathcal{A}'}$ ne bouge pas, mais met à jour son état interne de $\mathcal{A}$, effaçant le délai créé à l'étape précédente.

Nous définissons maintenant formellement ce qui vient d'être évoqué. Soient $\mathcal{A} = (Q, A, B, \delta, \gamma, i, F)$ et $\mathcal{A}' = (P, B, C, \alpha, \beta, j, G)$ deux transducteurs bidirectionnels déterministes composables, et soit $\mathcal{S}(\mathcal{A}) = (S_{\mathcal{A}}, A, S_{\mathcal{A}}, \Delta, \gamma', I, H)$ le transducteur des tranches de $\mathcal{A}$.
Puisque l'entrée d'un transducteur est donnée entourée de marqueurs de fin de mot, nous devons les produire dans la mémoire tampon de façon adéquate. C'est alors que la notion de transducteur normalisé va se révéler utile et nous considérerons donc, sans perte de généralité, que $\mathcal{A}$ est normalisé. De plus, afin d'alléger la construction, nous

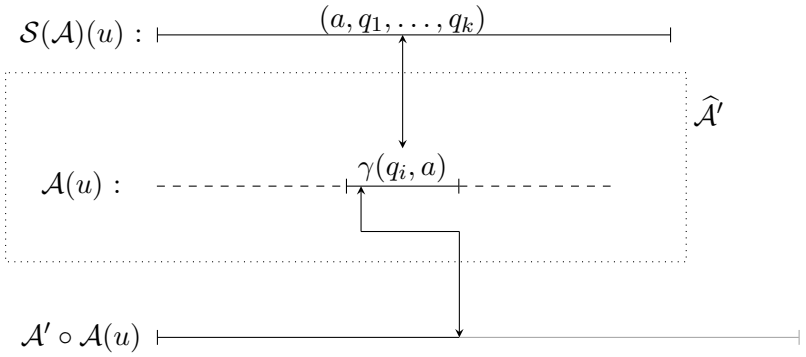



Figure 4.7: La mémoire tampon (trait plein de $\mathcal{A}(u)$) contient une partie de l'image de $u$ par $\mathcal{A}$ ($\delta(q_i, a)$). Suivant les déplacements de la tête sur cette mémoire, il faut alors revenir ou avancer dans le calcul de $\mathcal{A}(u)$ pour fournir l'information adéquat à la simulation de $\mathcal{A}'$.

considérerons que la fonction de production $\gamma$ de $\mathcal{A}$ est modifiée de façon à produire les marqueurs de fin de mot de la façon suivante : la production initiale $\gamma(i, \vdash) = w$ est remplacée par $\gamma(i, \vdash) = \vdash w$, et pour tout couple $(q, a)$ tel que $\delta(q, a) = (f, d)$ où $f$ appartient à $F$, nous ajoutons le marqueur de fin de mot $\dashv$ à la fin de $\gamma(q, a)$. De cette façon, les marqueurs de fin de mot seront produits dans la mémoire tampon et pourront être lus lors de la simulation de $\mathcal{A}'$. La normalisation de $\mathcal{A}$ nous assure que les marqueurs n'apparaîtront qu'au bon endroit, c'est-à-dire lorsque l'on atteindra le début ou la fin de l'exécution de $\mathcal{A}$.

On définit alors le transducteur $\widehat{\mathcal{A}'} = (R, S_{\mathcal{A}}, C, \mu, \eta, r_0, R_r)$ de la façon suivante :

- $R = P \times Q \times (Z \cup \{-1, +1\})$ où $Z = \{(v, i) \mid \exists q, a$ tels que $\gamma(q, a) = v$ et $0 \leqslant i \leqslant |v| + 1\} \cup \{(\epsilon, 1)\}$,
- $r_0 = (j, i, (\epsilon, 1))$,
- $R_r = \{p, q, (v, |v|) \mid p \in G, q \in F\}$.

Les définitions des fonctions de transition $\mu$ et de production $\eta$ sont plus complexes et nous les séparons en différents types comme évoqué plus haut. Nous considérerons dans la suite que le transducteur est dans un état $r$ et lit une tranche $s = (a, q_1, \cdots, q_k)$.

1. Si $r = (p, q, (v, i))$ avec $1 \leqslant i \leqslant |v|$, alors soit $(p', d) = \alpha(p, v_i)$ et $w = \beta(p, v_i)$. On définit alors $\mu(r, s) = ((p', q, (v, i + d)), 0)$ et $\eta(r, s) = w$.
   Si la simulation de $\mathcal{A}'$ est au milieu de la mémoire tampon, on continue simplement la simulation en déplaçant la tête de lecture de la mémoire et produisant le résultat attendu, sans toucher à la partie $\mathcal{A}$ de l'état de $\widehat{\mathcal{A}'}$. Les transitions de ce type représentent le seul cas où la fonction de production $\eta$ sera éventuellement non vide. Ainsi dans la suite la fonction $\eta$ sera omise.
2. Si $r = (p, q, (v, |v| + 1))$, alors soit $(q', d) = \delta(q, a)$ et $v = \gamma(q, a)$. on pose alors $\mu(r, s) = ((p, q', (v, 1)), d)$, et ce également pour $v = \epsilon$.
   Si la simulation de $\mathcal{A}'$ sort de la mémoire tampon par la droite, on utilise $\mathcal{A}$ pour

déterminer les prochaines transitions et productions du parcours de $\mathcal{A}$, et avançons dans la simulation de $\mathcal{A}$.
Si la production $v$ est le mot vide, alors à la prochaine étape nous serons à nouveau dans cette configuration et continuerons à progresser.

3. Si $r = (p, q, (v, 0))$, alors soit $k$ l'entier tel que $q_k = q$. Notons que puisque nous fournissons à $\widehat{\mathcal{A}'}$ un mot de tranches valide, il existe nécessairement un tel $k$. On définit maintenant $d = -1$ si $k = 1$ et $d$ comme la direction de $\delta(q_{k-1}, a)$ autrement.
Si $d = 0$, alors soit $v = \gamma(q_{k-1}, a)$ et on définit $\mu(r, s) = ((p, q_{k-1}, (v, |v|)), 0)$. Autrement, on définit $\mu(r, s) = ((p, q, -d), d)$.
Si la simulation de $\mathcal{A}'$ sort des limites par la gauche, on doit alors remonter dans l'exécution de $\mathcal{A}$. Cette transition sert à déterminer la direction par laquelle le parcours est arrivé à la position courante. Si la dernière transition prise était statique, on peut alors conclure et reprendre la simulation de $\mathcal{A}'$. Dans le cas contraire, on déplace la tête de lecture de $\widehat{\mathcal{A}'}$ afin de pouvoir déterminer la transition précédente. Pour se faire notons que l'on enregistre alors l'inverse du mouvement effectué. Cette information correspond à la direction prise lors de la transition précédente du calcul simulé de $\mathcal{A}$ et sera utilisée par la transition suivante (voir transition 4).
Remarquons que dans ce dernier cas, ainsi que dans le prochain, que si la production $v$ est le mot vide, alors $|v|$ sera égal à 0 et la prochaine transition prise par $\widehat{\mathcal{A}'}$ sera encore de ce type. On continuera à remonter dans l'exécution de $\mathcal{A}$ afin de remplir la mémoire tampon.
4. Si $r = (p, q, d)$, où $d$ est une direction, alors soit $k$ tel que $\delta(q_k, a) = (q, d)$ et soit $v = \gamma(q_k, a)$. On définit alors $\mu(r, s) = ((p, q_k, (v, |v|)), 0)$.
Si nous arrivons dans une transition où la troisième composante n'est pas une mémoire tampon mais $-1$ ou $+1$, c'est qu'alors la transition précédente était de type 3. La direction alors stockée nous permet de déterminer quelle a été la dernière transition prise dans le parcours de $\mathcal{A}$. Cette information nous permet de remplir la mémoire tampon et de continuer la simulation de $\mathcal{A}'$ en plaçant sa tête de lecture à la fin de la mémoire.

□

Maintenant que l'on a construit le transducteur $\widehat{\mathcal{A}'}$, nous devons prouver que notre construction préserve l'apériodicité. Nous commençons par énoncer les deux lemmes suivants qui éclairent le fonctionnement du transducteur $\widehat{\mathcal{A}'}$.

**Lemme 4.8.** *Entre deux mouvements de sa tête de lecture, le transducteur $\widehat{\mathcal{A}'}$ simule un parcours de $\mathcal{A}'$ sur une production éventuellement vide de $\mathcal{A}$, tout en mettant à jour son état interne de $\mathcal{A}'$.*

*Démonstration.* Lorsque $\widehat{\mathcal{A}'}$ remplit sa mémoire tampon, il y enregistre une des productions de $\mathcal{A}$, et place la tête de lecture de la mémoire tampon à une des extrémités. Le transducteur $\widehat{\mathcal{A}'}$ se met alors à simuler $\mathcal{A}'$ sur cette production, et ne se déplace alors

pas. Les mouvements de la tête de lecture sont donc liés à la simulation de $\mathcal{A}$ sur le mot d'entrée, et elle n'est donc déplacée que lorsque $\widehat{\mathcal{A}'}$ a fini de simuler $\mathcal{A}'$ sur la production enregistrée en mémoire tampon.

□

De plus, remarquons qu'étant donné que $\widehat{\mathcal{A}'} \circ \mathcal{S}(\mathcal{A})$ réalise la même fonction que $\mathcal{A}' \circ \mathcal{A}$, nous n'utilisons le transducteur $\mathcal{A}'$ que sur des productions de $\mathcal{A}$. Il est utile de remarquer que par définition des transducteurs, tout mot produit par le transducteur $\mathcal{A}$ peut se décomposer comme la concaténation de productions de $\mathcal{A}$. Ces productions forment donc une base des mots de $\mathcal{A}(A^*)$ et il est alors suffisant de connaître le comportement de $\mathcal{A}'$ sur ces briques élémentaires.

**Lemme 4.9.** *Soit $u$ un mot de tranches valide de $\mathcal{A}$ et $w$ son mot sous-jacent. Alors s'il existe un parcours partiel gauche à droite de $\widehat{\mathcal{A}'}$ sur $u$ depuis un état $(p, q, (v, i))$ vers un état $(p', q', (v', j))$, alors exactement une de ces deux alternatives est vérifiée :*

- *Il existe un parcours partiel gauche à droite de $\mathcal{A}$ sur $w$ depuis $q$ vers $q'$ produisant un mot que l'on notera $x$, ainsi qu'un parcours partiel gauche à droite de $\mathcal{A}'$ depuis $p$ vers $p'$ sur $wxw'$, où $w$ (resp. $w'$) est un suffixe (resp. préfixe) de $v$ (resp. $v'$).*
- *Il existe un parcours partiel droite à gauche de $\mathcal{A}$ sur $w$ depuis $q$ vers $q'$ produisant un mot que l'on notera $x$, ainsi qu'un parcours partiel droite à gauche de $\mathcal{A}'$ depuis $p$ vers $p'$ sur $wxw'$, où $w$ (resp. $w'$) est un suffixe (resp. préfixe) de $v$ (resp. $v'$).*

Le lemme se prouve aisément de façon inductive grâce à la construction de $\widehat{\mathcal{A}'}$ ainsi que du lemme précédent. Un propriété symétrique existe sur les parcours partiels droite à gauche, que l'on résume dans le tableau ci-dessous.

| | | parcours de $\mathcal{A}$ | |
|---|---|---|---|
| | | gauche à droite | droite à gauche |
| parcours de $\mathcal{A}'$ | gauche à droite | gauche à droite | droite à gauche |
| | droite à gauche | droite à gauche | gauche à droite |

Nous sommes désormais prêts à prouver que notre construction préserve l'apériodicité, ce qui termine la partie de notre étude concernant les tranches.

**Théorème 4.10.** *Si $\mathcal{A}$ et $\mathcal{A}'$ sont des transducteurs bidirectionnels déterministes et apériodiques composables, alors le transducteur $\widehat{\mathcal{A}'}$ construit est également apériodique.*

*Démonstration.* Grâce au Lemme 4.2, il nous suffit de prouver l'apériodicité des relations $\sim_{lr}$ et $\sim_{rl}$. Nous explicitons la preuve dans le cas de la relation $\sim_{lr}$. En fait, grâce aux mouvements bidirectionnels des deux transducteurs, une preuve symétrique sera aussi valable pour la relation $\sim_{rl}$. Soient $n_{\mathcal{A}}$ et $n_{\mathcal{A}'}$ les indices d'apériodicité de $\mathcal{A}$ et $\mathcal{A}'$ respectivement. Nous allons prouver que la relation $\sim_{lr}$ est apériodique d'indice d'apériodicité au plus $n = n_{\mathcal{A}} + n_{\mathcal{A}'}$.
Soit $u$ un mot de tranche valide tel que $u^2$ soit également un mot valide de tranche. Alors $u^n$ et $u^{n+1}$ seront également des mots valides.
Nous devons prouver que $u^n$ et $u^{n+1}$ possèdent les mêmes parcours partiels gauche-droite. Soit un parcours partiel gauche-droite de $\widehat{\mathcal{A}'}$ sur $u^n$ commençant dans un état $(p, q, z)$ et finissant dans un état $(p', q', z')$. Prouvons que ce parcours existe sur $u^{n+1}$. Grâce au Lemme 4.9, nous pouvons séparer ce problème en deux sous-cas distincts que nous traitons maintenant séparément.

- Le premier cas est décrit dans la Figure 4.8. Elle décrit le parcours sous-jacent de $\mathcal{A}$ décrit par le mot de tranches $u^n$. Après $n_{\mathcal{A}}$ itérations de $u$, étant donné que le mot sous-jacent à $u$ est fixé, le parcours partiel sur le mot sous-jacent à $u$ devient constant sur chaque itération de $u$. Ainsi la production de $\mathcal{A}$ sur chaque partie est alors la même, que nous notons $w$, et puisque l'on considère un parcours gauche à gauche, après $n_{\mathcal{A}'}$ itérations supplémentaire de $u$, on a simulé $\mathcal{A}'$ sur $w^{n_{\mathcal{A}'}}$. La partie $\mathcal{A}'$ du parcours de $\widehat{\mathcal{A}'}$ est alors constante sur chaque nouvelle itération de $u$, et ainsi le parcours peut-être étendu à un parcours sur $u^{n+1}$.
  De façon plus formelle, ce premier cas apparaît lorsqu'il existe des parcours partiels gauche à droite de $\mathcal{A}$ et $\mathcal{A}'$. Considérons les sous-parcours depuis le début de $u^n$ jusqu'à la première fois que le parcours entre dans une itération donnée de $u$. Alors par le Lemme 4.9, ces sous-parcours décrivent des parcours sous-jacent de $\mathcal{A}$, et puisque $\mathcal{A}$ est apériodique d'indice d'apériodicité $n_{\mathcal{A}}$, chaque itération de $u$ après la $n_{\mathcal{A}}$-*ième* sera d'abord atteinte par des états dont la composante selon $\mathcal{A}$ est la même. Nous la noterons $q'$ dans la suite. Nous considérons maintenant le sous-parcours entre la $n_{\mathcal{A}}$-*ième* itération de $u$ et la dernière. Sur chaque itération, le transducteur $\widehat{\mathcal{A}'}$ simule un parcours gauche à droite de $q'$ à $q'$ sur $u$. Par déterminisme de $\mathcal{A}$, la production sur chaque itération est donc la même et la troisième composante des états de $\widehat{\mathcal{A}'}$ est, en entrant la première fois une itération donnée de $u$, sera la même. Rappelons que nous notons $w$ cette production. Si $w$ est vide, alors la composante $\mathcal{A}'$ des états de $\widehat{\mathcal{A}'}$ ne change plus durant cette partie du parcours, étant donné que $\widehat{\mathcal{A}'}$ va simplement parcourir les itérations successives de $u$ en cherchant à remplir sa mémoire tampon. On peut alors étendre le parcours à un parcours sur $u^{n+1}$. Dans le cas contraire, $w$ est non vide et puisque $n = n_{\mathcal{A}} + n_{\mathcal{A}'}$, le Lemme 4.8 stipule que le parcours décrit par $\mathcal{A}'$ est un parcours sur $w^{n_{\mathcal{A}'}}$. Par apériodicité de $\mathcal{A}'$, il existe un parcours similaire sur $w^{n_{\mathcal{A}'}+1}$ et par conséquent nous pouvons également étendre ce parcours à un parcours sur $u^{n+1}$.
- Le second cas apparaît lorsque le parcours sur $\widehat{\mathcal{A}'}$ décrit des parcours droite à gauche de $\mathcal{A}$, mais également de $\mathcal{A}'$ qui par conséquent remonte le parcours décrit par $\mathcal{A}$. Alors de façon similaire nous savons que, jusqu'à la $(n - n_{\mathcal{A}})$-*ième* itération

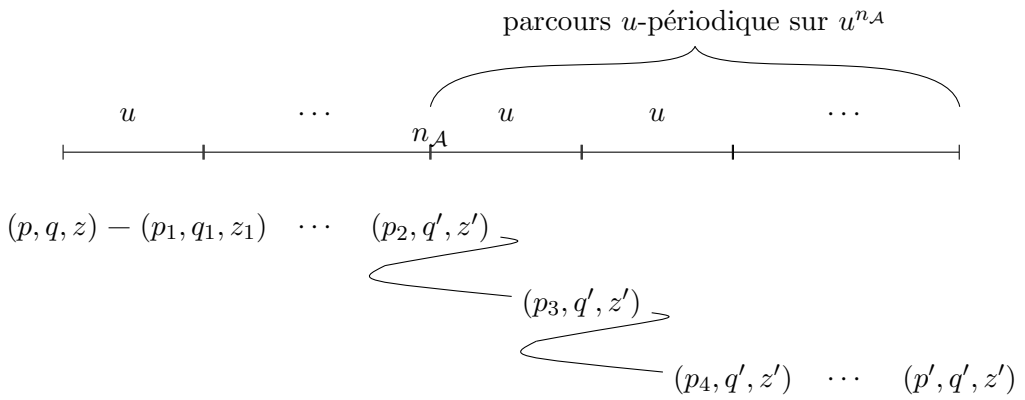



Figure 4.8: Cas d'un parcours gauche à gauche de $\mathcal{A}$ et $\mathcal{A}'$

de $u$, la composante $\mathcal{A}$ des états de $\widehat{\mathcal{A}'}$ est la même chaque fois que l'on entre dans une itération de $u$ pour la première fois puisque tout parcours sur $\mathcal{A}$ est stabilisé après $n_{\mathcal{A}}$ itérations, signifiant par là que les parcours décrit sur chaque $u$ sont les mêmes. Ainsi on sait que la partie $\mathcal{A}$ du parcours peut-être étendue à $u^{n+1}$. De plus, entre la première et la $n_{\mathcal{A}'}$-*ième* itération de $u$, le parcours simulé a produit dans la mémoire tampon un mot de la forme $w^{n_{\mathcal{A}'}}$, où $w$ est une production de $\mathcal{A}$. Pour les mêmes raisons que précédemment, on peut alors étendre le parcours à un parcours sur $u^{n+1}$.

On remarque finalement que les mêmes arguments nous permettant de raccourcir un parcours de $u^{n+1}$ à un parcours de $u^n$, nous obtenons l'apériodicité de la relation $\sim_{lr}$. Enfin, comme précisé plus haut, des arguments symétriques nous donnent l'apériodicité de la relation $\sim_{rl}$, et le Lemme 4.2 nous permet de conclure sur l'apériodicité de $\widehat{\mathcal{A}'}$. □

### 4.2.3 Décomposition des transducteurs non-ambigus

Dans les deux dernières sections, nous avons prouvé que l'on peut réduire la compositions de deux transducteurs bidirectionnels apériodiques à la composition d'un transducteur unidirectionnel apériodique non-ambigu avec un transducteur bidirectionnel apériodique. Les deux prochaines sections prouvent que cette composition peut être effectuée par un transducteur bidirectionnel apériodique. Tout d'abord, nous prouvons dans cette section que l'on peut décomposer un transducteur unidirectionnel non-ambigu en la composition d'un transducteur lisant une entrée de droite à gauche avec un transducteur lisant l'entrée de gauche à droite. Dans la prochaine section nous prouverons finalement que l'on peut inclure un transducteur unidirectionnel dans un transducteur bidirectionnel.

Nous rappelons qu'un transducteur non-ambigu est un transducteur non déterministe tel qu'il n'existe au plus qu'une seule exécution valide pour tout mot d'entrée, et nous donnons la définition des transducteurs que nous appellerons séquentiel gauche et séquentiel droite.

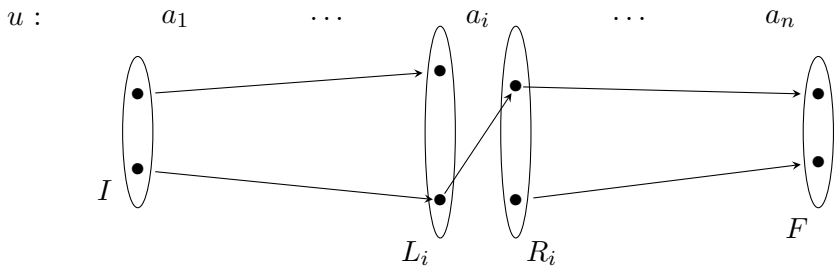



Figure 4.9: S'il l'on connaît les ensembles $L_i$ et $R_i$, on connait alors la transition prise.

Un transducteur est dit *séquentiel gauche* (resp. *séquentiel droit*) s'il est unidirectionnel et déterministe avec une tête de lecture se déplaçant de la gauche vers la droite (resp. de la droite vers la gauche).
Ainsi un transducteur unidirectionnel déterministe classique est séquentiel gauche.

De façon similaire aux parties précédentes, nous commençons par énoncer le théorème ainsi que donner les constructions, afin de prouver par la suite qu'elles préservent l'apériodicité.

**Théorème 4.11.** *[EM65] Soit $\mathcal{A}$ un transducteur unidirectionnel non-ambigu. On peut alors effectivement construire deux transducteurs $\mathcal{L}$ et $\mathcal{R}$, respectivement séquentiels gauche et droite, tels que $\mathcal{A} = \mathcal{L} \circ \mathcal{R}$.*

*Démonstration.* Soit un transducteur unidirectionnel non-ambigu $\mathcal{A} = (Q, A, B, \Delta, I, F)$. Par définition de la non-ambiguïté, étant donné un mot d'entrée $u$, il existe au plus une exécution acceptante de $\mathcal{A}$ sur $u$. Décrivons maintenant comment construire nos transducteurs séquentiels $\mathcal{L}$ et $\mathcal{R}$ afin de déterminiser les parcours de $\mathcal{A}$. Le transducteur séquentiel droit, lisant l'entrée de droite à gauche, sera utilisé en premier et sert à calculer et produire l'ensemble des états $R_i$ tels qu'il existe un parcours acceptant depuis la position courante jusqu'à la fin du mot d'entrée.
Le transducteur séquentiel gauche passant après calcule l'ensemble symétrique $L_i$ des états ayant un parcours depuis un état initial jusqu'à la position courante, et est capable de déterminer, grâce à ces deux ensembles, l'unique transition prise par $\mathcal{A}$ lors de l'exécution acceptante de $\mathcal{A}$ sur $u$ si elle existe.

L'argument clé de cette construction est que si le transducteur $\mathcal{L}$ se retrouve dans un état $L_i$ en lisant une lettre $a_i$ accompagnée de l'ensemble $R_i$ calculé par $\mathcal{R}$, la non-ambiguïté de $\mathcal{A}$ fait qu'il existe au plus une unique transition $(q, a_i, q')$ dans $\Delta$ telle $q$ appartienne à $L_i$ et $q'$ appartienne à $R_i$ (voir Figure 4.9). En effet, s'il existe deux transitions différentes satisfaisant cette propriété, alors nous pouvons déduire deux exécutions différentes de $\mathcal{A}$ sur $u$. S'il n'en existe aucune, alors la fonction réalisée par $\mathcal{A}$ n'est pas définie sur $u$.

Formellement, soit un transducteur unidirectionnel non-ambigu $\mathcal{A} = (Q, A, B, \Delta, \gamma, I, F)$. Nous définissons les transducteurs séquentiels $\mathcal{L}$ et $\mathcal{R}$ de la façon suivante :

- $\mathcal{R} = (\mathcal{P}(Q), A, A \times \mathcal{P}(Q), \alpha, \beta, F, J)$ où
  - $\mathcal{P}(Q)$ est l'ensemble des états de $\mathcal{R}$. Il s'agit des parties de $Q$,
  - $F$ est l'état initial de $\mathcal{R}$ et correspond à l'ensemble des états finaux de $\mathcal{A}$,
  - $J = \{P \subseteq Q \mid I \cap P \neq \emptyset\}$ est l'ensemble des état finaux de $\mathcal{R}$ et correspond à l'ensemble des parties de $Q$ contenant au moins un état initial de $\mathcal{A}$,
  - Pour tout ensemble d'états $S$ de $Q$ et toute lettre $a$ de $A$, on définit la fonction de transition $\alpha(a, S) = \{q \in Q \mid \exists q' \in S \ (q, a, q') \in \Delta\}$ de façon déterministe comme l'ensemble des états ayant une transition vers $S$,
  - Pour tout ensemble d'états $S$ et toute lettre $a$, la fonction de production est simplement définie par $\beta(a, S) = (a, S)$.
- $\mathcal{L} = (\mathcal{P}(Q), A \times \mathcal{P}(Q), B, \mu, \nu, I, G)$ où
  - $I$ est l'état initial de $\mathcal{L}$ et correspond à l'ensemble des états initiaux de $\mathcal{A}$,
  - l'ensemble des états finaux est $G = \{P \subseteq Q \mid P \cap F \neq \emptyset\}$ et correspond à l'ensemble des parties de $Q$ contenant au moins un état final,
  - Pour toute paire d'ensembles $S$ et $S'$, et toute lettre $a$, la fonction de transition est définie par $\mu(S, (a, S')) = \{q' \in Q \mid \exists q \in S \ (q, a, q') \in \Delta\}$, i.e. par l'ensemble des états accessibles en lisant la lettre $a$ depuis un état de $S$,
  - Pour toute paire d'ensembles $S$ et $S'$, et toute lettre $a$ telle qu'il existe exactement une transition de $(q, a, q') \in \Delta$ avec $q \in S$ et $q' \in S'$, on définit la fonction de production $\nu(S, (a, S')) = \gamma(q, a, q')$.

Nous définissons également les ensembles $R_i$ et $L_i$ évoqués plus haut. On remarque qu'ils dépendent du mot d'entrée, et nous prétendons qu'il s'agit des états visités par les transducteurs $\mathcal{R}$ et $\mathcal{L}$ respectivement. Pour tout mot $u$ sur $A^*$ et tout entier $i < |u|$, on définit

$$R_i = \{q \in Q \mid \ q \cdot (a_{i+1} \dots a_n) \cap F \neq \emptyset\}$$

$$\text{et } L_i = \{q \in Q \mid \exists q' \in I \ q \in q' \cdot (a_1 \dots a_{i-1})\}$$

avec $R_{|u|} = F$ et $L_1 = I$.

Nous prétendons que sur une entrée $u = a_1 \dots a_n$, le transducteur séquentiel droit $\mathcal{R}$ visite successivement les états $F = R_n, \dots, R_1$ et produit ainsi le mot $w = (a_1, R_1) \dots (a_n, R_n)$. Symétriquement, nous prétendons également que le transducteur séquentiel gauche $\mathcal{L}$, lorsqu'il lit la production $w$ de $\mathcal{R}$, visite successivement les états $I = L_1, \dots, L_n$ et produit $\mathcal{A}(u) = v_1 \dots v_n$ (voir Figure 4.10). Ceci se prouve en effet aisément par induction, en constatant que initialement $R_{|u|} = F$ et $L_1 = I$ et que de plus à chaque étape des parcours de $\mathcal{L}$ et $\mathcal{R}$, les ensembles $R_i$ et $L_i$ sont mis à jour selon la relation de transition de $\mathcal{A}$.

□

Nous nous attaquons maintenant à la préservation de l'apériodicité de notre construction. Comme dans la Section 4.2.1, l'apériodicité des transducteurs construits découle naturellement de l'apériodicité du transducteur de départ puisque la construction ne fait

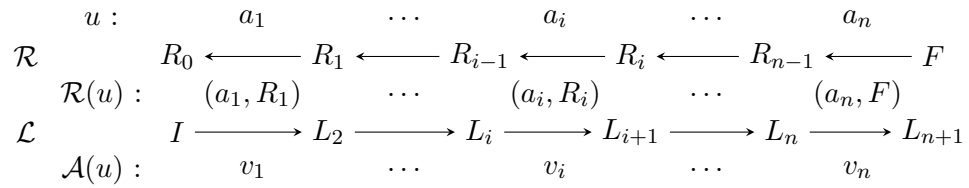



Figure 4.10: Vue d'ensemble de $\mathcal{L} \circ \mathcal{R}$.

que décomposer le transducteur donné en entrée, et réalise la même fonction que $\mathcal{A}$, de la même façon.

**Théorème 4.12.** *Soit $\mathcal{A}$ un transducteur non-ambigu apériodique. Alors les transducteurs $\mathcal{L}$ et $\mathcal{R}$ construit par le Théorème 4.11 sont également apériodiques.*

*Démonstration.* Notons tout d'abord que puisque l'apériodicité de $\mathcal{A}$ implique qu'il existe un entier $n$ tel que pour tout mot d'entrée $u$, les mêmes parcours partiels existent pour $u^n$ and $u^{n+1}$, nous pouvons déduire que les mêmes ensembles de parcours sont également possibles. Les parcours des transducteurs $\mathcal{L}$ et $\mathcal{R}$ correspondant à des ensembles de parcours possibles de $\mathcal{A}$ sur une entrée donnée, leur apériodicité dépend donc de celle de $\mathcal{A}$.
De plus, l'indice d'apériodicité des transducteurs $\mathcal{L}$ et $\mathcal{R}$ est le même que celui de $\mathcal{A}$. □

### 4.2.4 Composition d'un transducteur bidirectionnel avec un transducteur séquentiel

Dans cette section, nous donnons la dernière construction, qui prouve l'on peut réaliser la composition d'un transducteur séquentiel avec un transducteur bidirectionnel. Lorsque l'on aura prouvé la validité ainsi que l'apériodicité de la construction ci-après, nous aurons alors tout les éléments pour conclure sur la robustesse de la classe des transducteurs bidirectionnels apériodiques.

Comme dans les sections précédentes, nous commencerons par donner notre construction et par prouver sa validité, et nous prouverons qu'elle préserve l'apériodicité dans un second temps. La construction donnée ci-après s'apparente naturellement avec celle donnée dans la Section 4.2.2. En effet, contrairement aux autres constructions, celles-ci se proposent de réduire la composition de deux transducteurs à une seule machine. Ainsi nous utiliserons ici aussi une mémoire tampon sur laquelle naviguera une tête de lecture fictive. Cependant, ne possédant pas l'information des tranches, les méthodes utilisées pour remonter dans l'exécution du transducteur unidirectionnel seront moins directes. On remarquera également que par la nature du problème considéré, nous perdons les symétries qui nous permettaient notamment de traiter les relations $\sim_{lr}$ et $\sim_{rl}$ de la même façon.

Nous donnons maintenant la construction, et expliquons son fonctionnement en détail. Bien que ce résultat soit déjà connu, nous donnons ici une preuve avec notre formalisme, qui sera réutilisé afin d'obtenir l'apériodicité du transducteur construit.

**Théorème 4.13.** *[HU67] Soient $\mathcal{A}$ un transducteur unidirectionnel et $\mathcal{B}$ un transducteur bidirectionnel tout deux déterministes tels que l'on puisse composer $\mathcal{A}$ avec $\mathcal{B}$. Il existe alors un transducteur bidirectionnel déterministe $\mathcal{C}$ effectivement constructible tel que pour tout mot d'entrée $u$ de $\mathcal{A}$, on ait $\mathcal{C}(u) = \mathcal{B} \circ \mathcal{A}(u)$.*

*Démonstration.* L'idée de la construction est similaire à celle du transducteur $\widehat{\mathcal{A}'}$ de la Section 4.2.2. La différence fondamentale est qu'étant donné qu'ici nous ne possédons pas l'information essentielle présente dans les tranches, nous utiliserons le fait que le premier transducteur soit unidirectionnel pour pouvoir remonter dans l'exécution de $\mathcal{A}$. Cependant, le calcul, ainsi que les mouvements de la tête de lecture, deviendront plus complexe qu'une simple simulation de $\mathcal{A}$ ou de $\mathcal{B}$. L'ensemble des états de $\mathcal{C}$ est composé d'une union disjointe d'ensembles correspondant à différents modes de travail du transducteur.

Le premier mode sera le mode principal et sera très similaire au mode simulation de $\mathcal{A}'$ du transducteur $\widehat{\mathcal{A}'}$. Dans ce mode, les états de $\mathcal{C}$ possèdent trois composantes. Les deux premières sont des états internes de $\mathcal{A}$ et $\mathcal{B}$ respectivement. La troisième correspond à la mémoire tampon et sera composée d'une production de $\mathcal{A}$ sur une transition ainsi que d'un entier correspond à la position de la tête de lecture fictive sur la mémoire tampon. Le transducteur y simule un parcours partiel de $\mathcal{B}$ sur la mémoire tampon en mettant à jour son état interne $\mathcal{B}$ et restera dans ce mode tant que la tête de lecture reste dans les limites de la mémoire tampon. Tant que le transducteur reste dans ce mode, la tête de lecture sur le mot d'entrée ne bouge pas, de même que l'état interne de $\mathcal{A}$ ainsi que la mémoire tampon.

Lorsque la tête de lecture fictive quitte la mémoire tampon, par la gauche ou par la droite, on remonte ou progresse dans la simulation du calcul de $\mathcal{A}$ en mettant à jour l'état interne selon $\mathcal{A}$ de la façon suivante. Si la mémoire tampon est quittée par la droite, la mise à jour se fera naturellement et facilement en suivant la fonction de transition de $\mathcal{A}$ sur l'état interne selon $\mathcal{A}$ et la lettre courante. La tête de lecture bouge alors d'une position vers la droite. On réinitialise également la mémoire tampon avec la production de la transition prise, et l'on place la tête de lecture au début de la mémoire.

Si par contre l'on quitte la mémoire tampon vers la gauche, le transducteur $\mathcal{C}$ doit remonter dans sa simulation de $\mathcal{A}$, et le calcul devient alors plus complexe puisque $\mathcal{A}$ n'est a priori pas co-déterministe. Ce retour en arrière dans l'exécution est effectué grâce à un aller-retour sur l'entrée permettant de lever l'ambiguïté. Ce mouvement est fait en deux temps, respectivement par le mode aller puis le mode retour.

Le mode aller est décrit dans la Figure 4.11. Supposons que le transducteur $\mathcal{C}$ est à une position $j$ avec $q$ comme composante interne selon $\mathcal{A}$. On désire maintenant remonter d'une étape la simulation de $\mathcal{A}$ et l'on doit donc déterminer quelle a été la transition prise à la position $j-1$ pour arriver dans cet état. Le transducteur $\mathcal{C}$ se déplace alors

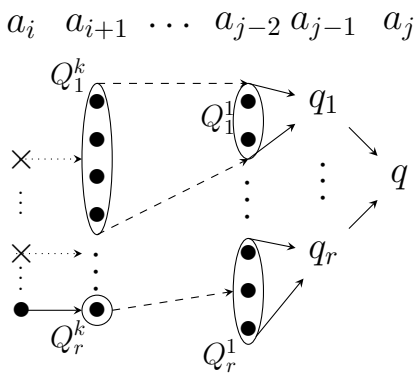



Figure 4.11: Description d'un parcours dans le mode aller

d'une position vers la gauche et calcule les $r$ états $q_1, \ldots, q_r$ envoyés sur $q$ en lisant la lettre $a_{j-1}$. Si $r = 1$, cela signifie qu'il n'y a qu'un seul candidat possible et l'on sait alors quelle a été la précédente transition prise. On peut alors remplir la mémoire tampon et mettre à jour la composante $\mathcal{A}$ de $\mathcal{C}$. On place la tête de lecture à droite de la mémoire tampon, et l'on peut repasser dans le mode principal.
Si maintenant $r$ est strictement plus grand que 1, le transducteur $\mathcal{C}$ a besoin de reculer plus en arrière sur le mot d'entrée, jusqu'à pouvoir réduire le nombre de parcours potentiels à une seule possibilité. Il devra ensuite revenir à la position $j-1$ afin de ne pas se perdre dans son mot d'entrée. Le transducteur va alors se déplacer vers la gauche, en stockant les ensembles $Q_\ell^k$ d'états ayant un chemin menant à $q_\ell$ sur le sous-mot $a_{j-(k+1)} \ldots a_{j-2}$. Ceci peut être fait de façon inductive et avec une mémoire finie, en remarquant que $Q_\ell^k$ ne dépend que de $Q_\ell^{k-1}$. Notons que puisque $\mathcal{A}$ est déterministe, un état ne peut, à $k$ fixé, appartenir à deux ensembles $Q_\ell^k$ différents. Enfin, étant donné que l'on a supposé que la simulation de $\mathcal{A}$ est arrivée jusqu'à la position $j$, à chaque étape au moins un $Q_\ell$ sera non vide.
Si à un moment nous arrivons à la situation décrite à la position $i$ de la Figure 4.11, où un seul des ensembles $Q_\ell$ est non vide, on peut alors conclure quant à la transition prise à la position $j-1$. Le transducteur $\mathcal{C}$ sélectionne alors le bon candidat $Q_\ell$ et doit maintenant retourner à la position $j-1$. Afin d'effectuer ceci, il utilise le fait qu'au moins deux ensembles $Q_\ell$ étaient non vides à la position $i+1$ et stocke ainsi deux états menant à deux candidats différents. Le premier état sera un état de $Q_\ell$, le bon candidat, et le second servira de témoin afin de pouvoir retrouver la position $j-1$. Le parcours retour est fait dans le mode retour décrit ci-après.
Si par contre la situation décrite dans la Figure 4.11 n'apparaît pas, alors le transducteur va reculer jusqu'au début de l'entrée. À ce moment, on sait que le candidat correct est celui dont l'ensemble contient l'état initial. Le transducteur agit alors comme dans la situation précédente après avoir déterminé le candidat correct.

Le troisième et dernier mode, le mode retour, est décrit dans la Figure 4.12. Rappelons que lorsque le transducteur $\mathcal{C}$ est passé dans le mode retour, la tête de lecture était à une

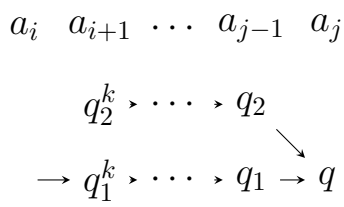



Figure 4.12: Un parcours dans le mode retour.

position $i+1$, succédant celle où il a pu déterminer le bon candidat. Il a stocké deux état $q_1^k$ et $q_2^k$ dont les parcours partiels sur $a_{i+1} \dots a_{j-1}$ mènent à deux états différents $q_1$ et $q_2$, et doit s'arrêter à la position $j-1$ dans l'état $q_1$ selon $\mathcal{A}$. La stratégie du transducteur $\mathcal{C}$ est d'alors suivre les deux chemins en parallèle. À chaque étape, il compare les états atteints par les deux parcours à l'étape suivante. Nous savons que puisque les deux parcours mènent à des candidats différents, ils se rejoindront exactement à la position $j$ sur l'état $q$. Puisque la comparaison s'effectue avec une étape d'avance, on sait que l'on doit s'arrêter lorsque les deux états coïncident. À ce moment, grâce à la lettre $a_{j-1}$ et l'état $q_1$, il est possible de remplir la mémoire tampon et l'on place sa tête de lecture à l'extrémité droite. On peut finalement repasser dans le mode principal et simuler $\mathcal{B}$.

De façon formelle, soient $\mathcal{A} = (Q, A, B, \delta, \gamma, q_0, F)$ un transducteur unidirectionnel et $\mathcal{B} = (P, B, C, \alpha, \beta, p_0, G)$ un transducteur bidirectionnel tout deux déterministes. Nous considérerons que les deux transducteurs sont normalisés. Soit maintenant le transducteur bidirectionnel $\mathcal{C} = (R, A, C, \mu, \eta, r_0, R_f)$ défini de la façon suivante :

- $R = R_p \uplus R_a \uplus R_r$ est l'ensemble des états de $\mathcal{C}$ où
  - $R_p = P \times Q \times Z$ est le mode principal où $Z = \{(v, i) \mid \exists q, a$ tels que $\gamma(q, a) = v$ et $0 \leqslant i \leqslant |v| + 1\} \cup \{(\epsilon, 1)\}$ est la mémoire tampon avec la tête de lecture fictive,
  - $R_a = P \times Q \uplus P \times 2^{Q \times Q}$ est le mode aller. Cet ensemble d'états est en deux parties. La première est utilisée lors de la première étape lorsque l'on remonte dans la simulation de $\mathcal{A}$. Elle sert à calculer l'ensemble des candidats potentiels $q_1, \dots, q_r$. La seconde partie sert à calculer l'ensemble des parcours partiels pour chaque candidat. L'information est stockée sous forme d'une relation où $q$ est en relation avec $q'$ si $q$ a un parcours partiel vers le candidat $q'$.
  - $R_r = P \times Q \times Q$ correspond au mode retour. Les états de ce mode contiennent un état de $\mathcal{B}$ comme composante interne de la simulation de $\mathcal{B}$, ainsi que deux états de $\mathcal{A}$ concurrents servant à retrouver notre position, le premier menant au bon candidat.
- $r_0 = (p_0, q_0, (\epsilon, 1))$ est l'état initial où les deux transducteurs sont initialisés et la mémoire tampon est vide.
- $R_f = \{(p, q, (v, |v|) \mid p \in F, q \in G\}$ est l'ensemble des états finaux. Il s'agit de l'ensemble des états correspondant à la fin des simulations de $\mathcal{A}$ et $\mathcal{B}$.

Nous décrivons maintenant les fonctions de transitions $\mu$ et de production $\eta$ à travers une succession de cas, suivant l'état $r$ du transducteur $\mathcal{C}$. On suppose dans chaque cas

que le transducteur lit une lettre $a$ en un état $r$.

1. Si on a $r = (p, q, (v, i))$ un état de $R_p$ le mode principal, on traite alors trois sous-cas suivant la valeur de $i$ :
   - Si on a $1 \leqslant i \leqslant |v|$, c'est-à-dire si la tête de lecture de la mémoire se trouve à l'intérieur de la mémoire, soit alors $(p', d) = \alpha(p, v_i)$ et on définit $\mu(r, a) = ((p', q, (v, i+d)), 0)$ et $\eta(r, a) = \beta(p, a)$. Il s'agit ici du seul sous-cas où la fonction de production $\eta$ est non vide. Elle sera donc omise dans la suite.
   - Si on a $i = |v| + 1$, soit alors $q' = \delta(q, a)$ et $v' = \gamma(q, a)$ et l'on définit $\mu(r, a) = ((p, q', (v', 1)), +1)$,
   - Si enfin $i = 0$, on définit $\mu(r, a) = ((p, q), -1)$ un état de la première partie du mode aller.
2. Si $r = (p, q)$ est un état de la première partie de $R_a$, soit alors $Q' = \{q' \in Q \mid \delta(q', a) = q\}$ l'ensemble des candidats.
   - Si $Q' = \{q'\}$ est un singleton, soit alors $v = \gamma(q', a)$ et on définit $\mu(r, a) = ((p, q', (v, |v|)), 0)$,
   - sinon, soit $S = \{(q', q') \mid q' \in Q'\}$ la relation définie plus haut et $\mu(r, a) = ((p, S), -1)$.

   Nous calculons ici les candidats potentiels. S'il n'en existe qu'un seul, nous pouvons alors décider la transition précédente et repasser dans le mode principal. Sinon, nous stockons les candidats dans une relation symétrique partielle et continuons à reculer sur l'entrée.
3. Si $r = (p, S) \in R_a$, soit alors $T = \{(q, q_1) \mid (\delta(q, a), q_1) \in S\}$ la nouvelle relation avec une étape à gauche supplémentaire. Nous divisons alors en deux sous-cas supplémentaires : si l'on peut déterminer le bon candidat ou s'il faut continuer.
   - S'il n'existe qu'un seul état $q$ tel que $Q \times \{q\} \cap T \neq \emptyset$, soit alors $q' \neq q$ tel qu'il existe deux états $q_1$ et $q_2$ avec $(q_1, q)$ et $(q_2, q')$ appartenant à $S$. On définit alors $\mu(r, a) = ((p, q_1, q_2), +1)$ où $(p, q_1, q_2)$ appartient au mode retour $R_r$.

     S'il n'existe qu'un seul candidat restant à cette position, nous stockons alors deux états menant à deux candidats différents à partir de la position suivante, et nous passons dans le mode retour.

     On remarque ici qu'il peut y avoir introduction de non-déterminisme dans le choix des états que l'on prend. Cette indéterminisme peut cependant être levée aisément en considérant par example un ordre dans le choix des états que l'on choisit.
   - Autrement, si $a = \vdash$ est le marqueur de début de mot, nous procédons de la même façon que précédemment en prenant comme bon candidat celui qui est en relation avec l'état initial. Il existe nécessairement un tel état étant donné que nous simulons un parcours de $\mathcal{A}$. L'état $q_0$ est donc un état co-accessible depuis notre position initiale.
   - Si nous ne sommes pas dans le cas précédent, on définit alors $\mu(r, a) = ((p, T), -1)$. S'il ne nous est pas possible de décider le candidat correct à cette étape, nous continuons à reculer sur l'entrée.

4. Si $r = (p, q_1, q_2)$ est un état du mode retour $R_r$, alors deux comportements sont à distinguer :
   - si $\delta(q_1, a) \neq \delta(q_2, a)$, on définit $\mu(r, a) = (p, \delta(q_1, a), \delta(q_2, a))$.
     Si les deux chemins ne coïncident pas à la prochaine étape, alors nous ne sommes pas encore arrivés et continuons à simuler les deux parcours en parallèle.
   - si par contre $\delta(q_1, a) = \delta(q_2, a)$ soit alors $v = \gamma(q_1, a)$ et on définit $\mu(r, a) = ((p, q_1, (v, |v|)), 0)$.
     Si les deux chemins se rencontreront à la prochaine étape, alors nous sommes arrivés et nous pouvons remplir la mémoire tampon, en plaçant la tête de lecture à l'extrémité droite de la mémoire.

□

Après avoir défini et expliqué comment notre construction fonctionne, nous devons montrer qu'elle conserve l'apériodicité. Ceci paraît moins évident que pour les autres constructions, car nous ajoutons ici des mouvements aller-retours qui ne sont présents dans aucun des deux transducteurs de départ. Nous commençons par donner un lemme simple servant à éclairer sur la machinerie derrière le transducteur $\mathcal{C}$, et plus particulièrement le rapport entre les états des différents modes et les positions relatives atteintes par chacun des modes. Ceci se révélera utile lorsque l'on voudra caractériser les différents parcours partiels possibles du transducteur $\mathcal{C}$ pour chaque relation.

**Lemme 4.14.** *Lors d'une exécution de transducteur $\mathcal{C}$, si l'état courant appartient au mode principal $R_p$, alors toute position située à droite de la position courante sera atteinte en premier par un état du mode principal.*
*De plus, tout mouvement de la tête de lecture vers la gauche est effectué vers un état du mode aller $R_a$.*

*Démonstration.* Par construction de $\mathcal{C}$, les états pouvant déplacer la tête de lecture vers la droite sont exactement les états des modes principal et retour. De même, les transitions déplaçant la tête de lecture vers la gauche sont depuis $R_p$ vers $R_a$ ou de $R_a$ vers $R_a$.
De plus, les modes aller et retour sont utilisés pour revenir exactement une position avant la position à laquelle un état du mode principal était atteint. Ainsi, si le transducteur $\mathcal{C}$ est dans un état du mode principal, un passage dans les modes aller-retour ne ramènera la tête de lecture qu'à la position précédant la position actuelle et toute position à droite ne peut être atteinte que dans un état du mode principal. □

Nous concluons notre série de constructions par la preuve de l'apériodicité du transducteur construit.

**Théorème 4.15.** *Si $\mathcal{A}$ et $\mathcal{B}$ sont deux transducteurs apériodiques tels qu'évoqués dans le Théorème 4.13, alors le transducteur $\mathcal{C}$ résultant est également apériodique.*

*Démonstration.* Prouvons que l'apériodicité est préservée par notre construction. Grâce au Lemme 4.2, il est suffisant de prouver l'apériodicité pour les relations $\sim_{lr}$ et $\sim_{rl}$. Rappelons qu'étant donné que, contrairement aux autres constructions, le problème considéré ici n'est pas symétrique gauche-droite, il est nécessaire de prouver l'apériodicité des deux relations séparément. Notons $n_{\mathcal{A}}$ et $n_{\mathcal{B}}$ les indices d'apériodicité de $\mathcal{A}$ et $\mathcal{B}$ respectivement, et prouvons que les relations $\sim_{rl}$ et $\sim_{lr}$ sont apériodiques d'indice d'apériodicité au plus $n = n_{\mathcal{A}} + n_{\mathcal{B}}$.
Nous traitons tout d'abord la relation $\sim_{lr}$, puis la relation $\sim_{rl}$.

$u^n \sim_{lr} u^{n+1}$ Considérons tout d'abord un parcours de $u^n$ commençant à gauche dans un état $r$ et sortant par la droite dans un état $r'$. Nous traitons plusieurs sous-cas suivant les modes auxquels $r$ et $r'$ appartiennent.

- Si $r$ appartient au mode principal $R_p$, alors le Lemme 4.14 stipule que l'état $r'$ appartient également à $R_p$. De plus, chaque position de $u^n$ sera tout d'abord atteinte par un état du mode principal. Puisque $\mathcal{C}$ ne progresse vers la droite dans le mode principal qu'en simulant $\mathcal{A}$ et sa production dans la mémoire tampon, le parcours sur $u^n$ décrit naturellement un parcours sous-jacent sur $\mathcal{A}$ en considérant l'état interne selon $\mathcal{A}$ des états de $\mathcal{C}$ sur chaque position lorsqu'elle est atteinte la première fois par le parcours. Ainsi après $n_{\mathcal{A}}$ itérations de $u$, le parcours décrit est le même sur chaque itération de $u$, un parcours $q \xrightarrow{u} q$ et produit donc le même mot $v$. Si $v$ est le mot vide, alors le parcours de $\mathcal{C}$ correspond simplement, à partir de la $n_{\mathcal{A}}$-*ième* itération de $u$, à un parcours de l'entrée cherchant à remplir la mémoire tampon afin de simuler $\mathcal{B}$. Le même parcours existera alors avec une itération supplémentaire de $u$.
  Dans le cas contraire, le transducteur $\mathcal{C}$ ne se déplace vers l'avant que s'il a simulé $\mathcal{B}$ sur la mémoire tampon, et est sorti vers la droite. Ainsi si le parcours avance jusqu'à $n_{\mathcal{A}}+n_{\mathcal{B}}$, $\mathcal{C}$ aura à la fin simulé $\mathcal{B}$ sur un mot $wv^{n_{\mathcal{B}}}$ où $w$ est une partie de la production initiale du parcours, avant la $n_{\mathcal{A}}$-*ième* itération. Par apériodicité de $\mathcal{B}$ le parcours simulé sur $v^{n_{\mathcal{B}}}$ existe sur $v^{n_{\mathcal{B}}+1}$ et puisque la relation $\sim_{\mathcal{B}}$ est une congruence, le parcours simulé sur $wv^{n_{\mathcal{B}}}$ peut être étendu à un parcours sur $wv^{n_{\mathcal{B}}}$. Le parcours peut ainsi être étendu à $u^{n+1}$, pour tout $n \geqslant n_{\mathcal{A}} + n_{\mathcal{B}}$.
- Deuxièmement, si les deux états $r$ et $r'$ appartiennent au mode retour $R_r$, alors le Lemme 4.14 stipule que le parcours n'atteint jamais un état du mode principal. En effet, si c'était le cas, $r'$ ne pourrait appartenir à $R_r$. On peut alors en déduire, étant donné que le transducteur $\mathcal{C}$ ne quitte le mode retour que pour le mode principal, que le parcours entier se place dans le mode retour. Or, le mode retour n'est que la simulation de deux parcours de $\mathcal{A}$ en parallèle, sans même considérer la production. Alors le parcours est ici simplement un double parcours de $\mathcal{A}$ sur $u^n$. Alors pour tout $n$ supérieur à $n_{\mathcal{A}}$, tout parcours de cette sorte sur $u^n$ peut être étendu à $u^{n+1}$.
- Si $r$ appartient au mode retour $R_r$ mais que $r'$ appartient au mode principal $R_p$, considérons alors l'état $r''$ atteint par $\mathcal{C}$ lorsque le parcours entre la première fois dans la $n_{\mathcal{A}}$-*ième* itération de $u$. Si $r''$ appartient à $R_r$, ce début de parcours s'étend alors, d'après le point précédent, à toute nouvelle itération de $u$, ce qui

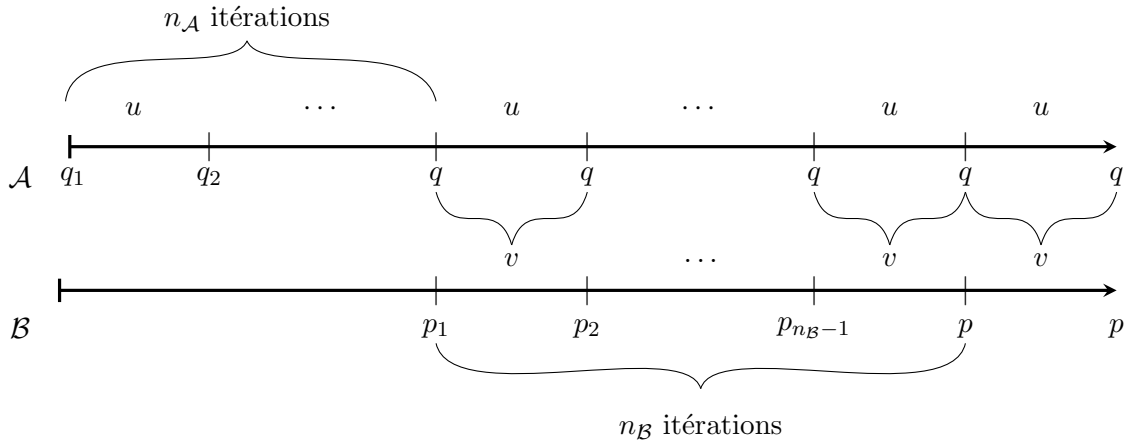



Figure 4.13: Relation $\sim_{lr}$ : Description d'un parcours dans le mode principal.

est impossible puisque $r'$ appartient à $R_p$. Alors $r'' = (p, q, z)$ appartient à $R_p$, et nous sommes ramenés au premier cas évoqué avec $q \xrightarrow{u} q$ sur $\mathcal{A}$. Après $n_{\mathcal{B}}$ itérations supplémentaires de $u$, nous sommes donc sûrs de pouvoir étendre le parcours à un parcours sur $u^{n+1}$.

Les arguments avancés étant toujours valables si l'on voulait initialement réduire un parcours sur $u^{n+1}$, on obtient alors l'apériodicité de la relation $\sim_{lr}$.

$u^n \sim_{rl} u^{n+1}$ Soit un parcours sur $u^n$ commençant par la droite dans l'état $r$ et sortant par la gauche dans l'état $r'$. Prouvons que ce parcours existe sur $u^{n+1}$. Grâce au Lemme 4.14, on sait que les états $r$ et $r'$ appartiennent au mode aller $R_a$. Ainsi le parcours correspond à un retour en arrière d'un parcours sur $\mathcal{A}$, avec éventuellement des simulations de parcours de $\mathcal{B}$ au milieu.

- Supposons que le parcours reste en mode aller suffisamment longtemps, c'est-à-dire pendant $n_{\mathcal{A}}$ itérations de $u$. Alors l'ensemble $S$ calculé dans les états de $\mathcal{A}$, décrivant les parcours de $\mathcal{A}$ sur le suffixe commençant à la position actuelle jusqu'à la fin du mot, et peut-être au delà. Ainsi, l'apériodicité de $\mathcal{A}$ nous stipule que puisque $u^n \sim_{\mathcal{A}} u^{n+1}$, l'état de $\mathcal{C}$ après avoir remonté $n$ itérations de $u$ sera le même qu'après en avoir remonté $n + 1$. On peut donc étendre le parcours à $u^{n+1}$.
- Dans l'autre cas, cela signifie que le transducteur $\mathcal{C}$ est capable de trouver le bon candidat, revenir à la bonne position, mais que la simulation de $\mathcal{B}$ l'amène à recommencer le processus. On sait ainsi que la simulation de $\mathcal{B}$ remonte la production de $\mathcal{A}$ sur les $(n - n_{\mathcal{A}}) = n_{\mathcal{B}}$-*ième* premières itérations sur la droite. Puisque l'on a simulé $\mathcal{A}$ sur $u^n$, ces $n_{\mathcal{B}}$ dernières itérations (en comptant depuis la gauche), produisent le même mot $v$ et donc le transducteur $\mathcal{C}$ a simulé $\mathcal{B}$ sur $v^{n_{\mathcal{B}}}$. Ainsi, par apériodicité de la relation $\sim_{rl}$ de $\mathcal{B}$, on sait que l'on peut étendre le parcours et la simulation de $\mathcal{B}$ remontera $v^{n_{\mathcal{B}}+1}$. Le parcours existe donc sur $u^{n+1}$. Il est à noter que, contrairement à la relation $\sim_{lr}$, le parcours est

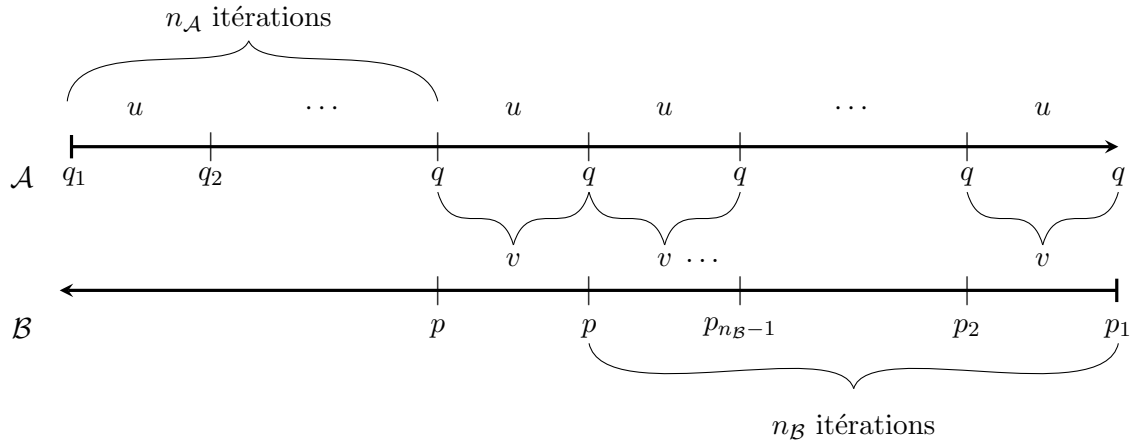



Figure 4.14: Relation $\sim_{rl}$ : Alors que $\mathcal{B}$ va de droite à gauche, $\mathcal{C}$ remonte le parcours de $\mathcal{A}$. Après suffisamment d'itérations, on peut étendre ou réduire les parcours.

"stabilisé" non pas à la fin de l'entrée, mais au milieu, après $u^{n_{\mathcal{A}}}$ et avant $u^{n_{\mathcal{B}}}$ (voir Figure 4.14). On obtient alors une apériodicité avec un indice $n_{\mathcal{A}} + n_{\mathcal{B}}$.
En remarquant que les arguments apportés permettent également de réduire un parcours sur $u^{n+1}$ à un parcours sur $u^n$, on peut alors conclure sur l'apériodicité du transducteur $\mathcal{C}$.

□

## 4.3 Conclusion

Nous sommes désormais prêts à énoncer et prouver le théorème annoncé au début de ce chapitre, résultat des différentes constructions et de leur apériodicité.

**Théorème 4.16** (Composition des transducteurs bidirectionnels apériodiques)**.** *Soient deux transducteurs bidirectionnels déterministes, apériodiques et composables, de telle façon que l'alphabet de sortie de l'un soit l'alphabet d'entrée de l'autre.*
*On peut alors effectivement construire un transducteur bidirectionnel apériodique et déterministe réalisant exactement la composition de ces deux transducteurs.*

*Démonstration.* Soient deux transducteurs bidirectionnels $\mathcal{A}$ et $\mathcal{A}'$ déterministes et apériodiques, tels que $\mathcal{A}$ soit composable avec $\mathcal{A}'$.

Par la Définition 4.8 et la Proposition 4.6, on peut alors construire $\mathcal{S}(\mathcal{A})$ un transducteur unidirectionnel non-ambigu et apériodique calculant les mots de tranche valides de

$\mathcal{A}$. Le transducteur $\mathcal{S}(\mathcal{A})$ étant non-ambigu, il est, par les Théorèmes 4.11 et 4.12, décomposable en la composition d'un transducteur séquentiel droite $\mathcal{R}$ puis d'un transducteur séquentiel gauche $\mathcal{L}$, tout deux apériodiques. On a alors $\mathcal{S}(\mathcal{A}) = \mathcal{L} \circ \mathcal{R}$.

De l'autre coté, nous savons également, grâce aux Théorèmes 4.7 et 4.10, que nous pouvons construire un transducteur déterministe et apériodique $\widehat{\mathcal{A}'}$ tel que $\widehat{\mathcal{A}'} \circ \mathcal{S}(\mathcal{A}) = \mathcal{A}' \circ \mathcal{A}$. Ainsi nous obtenons l'égalité $\widehat{\mathcal{A}'} \circ \mathcal{L} \circ \mathcal{R} = \mathcal{A}' \circ \mathcal{A}$. Les Théorèmes 4.13 et 4.15 stipule qu'il existe alors un transducteur apériodique et déterministe $\mathcal{C}$ tel que $\widehat{\mathcal{A}'} \circ \mathcal{L} = \mathcal{C}$ et qu'ainsi $\mathcal{A}' \circ \mathcal{A} = \mathcal{C} \circ \mathcal{R}$.

On obtient finalement un transducteur $\mathcal{C}'$ réalisant $\mathcal{C} \circ \mathcal{R}$ par une construction symétrique de celle de la Section 4.2.4. Le transducteur $\mathcal{C}'$ est alors un transducteur bidirectionnel apériodique et déterministe réalisant la composition $\mathcal{A}' \circ \mathcal{A}$, concluant notre preuve. □

Bien que fournissant un résultat théorique appréciable, notre preuve est algorithmiquement très lourde. En effet, chaque construction induit une explosion exponentielle du nombre d'états. On obtient finalement une tour en quadruple exponentielle par rapport à la taille des transducteurs en entrée.

Ce résultat reste intéressant car il justifie de la robustesse de la classe que nous avons considérée. Il se révélera également primordial lors de l'obtention de nouveaux résultats théoriques tels que l'équivalence avec d'autres sous-modèles ou d'autres logiques. Ainsi, il semble naturel de vouloir faire correspondre notre classe de transducteurs avec une sous-classe de transductions de graphes telles que définies par Courcelle dans [Cou94] et prouvées équivalentes aux transducteurs bidirectionnels par Engelfriet et Hoogeboom dans [EH01]. La sous-classe naturelle qui semble s'imposer est la sous-classe des transductions de graphes $FO$ définissables, ce qui étendrait l'équivalence entre logique du premier ordre et monoïdes apériodiques existante sur les langages réguliers [Sch65, MP71].

Il serait également intéressant de comparer notre classe à des restrictions de modèles connus comme équivalents aux transducteurs bidirectionnels, tels que les récents Streaming String Transducers apériodiques définis par Filiot, Krishna et Trivedi dans [FKT14], dont l'équivalence avec les transductions de graphes du premier ordre a été prouvée.

# Conclusion

Chapitre 5

## Résumé

Dans ce manuscrit, nous avons étudié différents moyens d'étendre l'expressivité de fragments de logique tout en préservant leur structure interne. Le but est de permettre d'étendre la simplicité des algorithmes les concernant à des classes de langages plus larges.

Les preuves présentées reposent essentiellement sur des arguments algébriques de la théorie des semigroupes. Nous abordons et étendons de nombreuses notions essentielles de cette théorie telles que les relations de Green ou les variétés de timbres et de catégories.

### Prédicats modulaires

Sur le thème de l'ajout des prédicats modulaires à un fragment décidable, notre approche algébrique s'est révélée fructueuse. En effet, nous avons ainsi obtenu tout d'abord un résultat technique et précis sur le fragment de la logique du premier ordre à deux variables. Ce résultat utilise des techniques de la théorie des automates ainsi que de la théorie des semigroupes, en exploitant les caractéristiques syntaxiques de la variété de monoïdes décrivant ce fragment. Nous avons ainsi obtenu une preuve spécifique mais claire et applicable du transfert de la décidabilité du problème de définissabilité.

Avec une approche plus générale, nous sommes également parvenus à obtenir une caractérisation algébrique générique de l'ajout des prédicats modulaires. Bien que cette caractérisation ne soit a priori pas décidable, l'utilisation des catégories comme généralisation des monoïdes, étendant ainsi les notions de variétés et d'équations profinies, nous a permis d'obtenir la décidabilité de fragments enrichis si ce n'est dans tous les cas, au moins dans la majorité des fragments de logique connus et étudiés.

Cependant si dans certains cas tels que la logique du premier ordre, ou sa restriction à deux variables, la complexité supplémentaire engendrée est négligeable devant la complexité initiale, il est tout de même à noter que dans certains cas la complexité supplémentaire, dépendant d'une variété de catégories dérivée du fragment, doit être prise en compte.

### Transducteurs bidirectionnels

Dans notre étude des transducteurs bidirectionnels, nous avons défini avec succès la notion de congruence syntaxique d'un automate bidirectionnel, et par là même son monoïde syntaxique. Cette définition naturelle et déjà évoquée dans la littérature étend celle des machines unidirectionnelles.

Ces notions nous ont ainsi permis de définir les transducteurs bidirectionnels apériodiques, comme extension des langages sans-étoile et reconnus par des monoïdes apériodiques. En reprenant les constructions mentionnées dans la preuve originale de la stabilité par composition des machines bidirectionnelles, nous sommes parvenus à expliciter des constructions préservant l'apériodicité. Nous avons ainsi prouvé la stabilité par composition de notre sous-classe de transducteurs, et justifions ainsi sa robustesse et sa pertinence.

Nous sommes cependant conscients que les complexités mises en jeux lors de ses constructions successives rendent peu attrayante l'implémentation pratique et donc l'utilisation de cet algorithme.

## Perspectives

La suite à donner à ce travail peut se diviser en deux axes principaux.

En premier lieu, l'étude faite de l'ajout des prédicats modulaires est prometteuse et peut être étendue de différentes façons. Tout d'abord, il est envisageable d'améliorer la borne proposée dans le Chapitre 3 via le global des variétés de monoïdes. En effet, il n'existe pas à ce jour de contre-exemple connu où l'indice de congruence à considérer est strictement supérieur à l'indice de stabilité, même dans le cas de variétés non locales telles que les monoïdes commutatifs.

De façon plus générale, il est également possible de considérer d'autres ensembles de prédicats, tels que les prédicats locaux. Il a systématiquement été constaté, sans qu'il existe de preuve générique, que l'ajout des prédicats locaux correspondait algébriquement à effectuer un produit semi-direct par la variété des semigroupes localement idempotents. Nous supposons fortement que l'ajout des prédicats locaux peut alors se réduire à un produit semi-direct s'ils sont définis comme des prédicats unaires descriptifs. L'étude serait alors similaire, et le transfert de la décidabilité se réduirait alors également à la décidabilité du global de la variété.

À long terme, il est également envisageable de trouver des équivalents algébriques à d'autres opérations logiques. Ainsi, dans [Str94], Straubing a prouvé qu'une simple quantification existentielle du premier ordre correspond à un produit en bloc par le monoïde booléen simple. Il serait alors envisageable de construire un monoïde et le morphisme associé depuis une formule logique, et éventuellement obtenir la variété caractérisant un fragment de logique d'après les opérations de clôture la définissant. On serait finalement à même d'expliciter des fragments de logique non-naturels mais caractérisés par une variété.

L'autre axe principal de travaux futurs est l'étude de sous-classes des transducteurs

bidirectionnels. Il s'agit d'un modèle riche et équivalent à plusieurs autres modèles largement étudiés tels que les transductions de graphes et les Streaming string transducers définis par Alur et Černỳ.

Il existe donc un modèle logique de ces transducteurs. Cependant, ce modèle est une restriction aux graphes linéaires d'une logique sur les graphes. Il n'est donc pas complètement satisfaisant et il serait utile de définir une logique plus naturelle rendant compte de l'expressivité des transducteurs bidirectionnels. Nous avons également défini une notion de congruence syntaxique. Bien que suffisante pour définir les transducteurs bidirectionnels apériodiques, elle reste incomplète puisque définie pour les automates et non les transducteurs. Il apparaît nécessaire d'étendre la notion de morphisme afin de prendre en compte la fonction réalisée par le transducteur. La conséquence possible de ceci serait la possibilité de caractériser les fonctions réalisées par les transducteurs bidirectionnels. Dans cette optique, il est à noter le récent travail de Bojanczyk [Boj14], qui définit les transducteurs avec origine et les transductions du premier-ordre avec succès.

Dans la lignée de ce qui a été écrit dans ce manuscrit, il serait également intéressant de pouvoir définir des sous-classes de ces transducteurs. Nous avons commencé avec la classe naturelle des transducteurs apériodiques. La suite évidente est alors de prouver leur équivalence avec les transductions de graphes du premier ordre. Cette étude aurait également beaucoup à profiter de l'existence des objets envisagés plus haut. Un autre point important serait alors d'obtenir la décidabilité de la classe des transducteurs apériodiques non pas pour le transducteur donné en entrée, mais pour la fonction qu'il réalise. C'est-à-dire que l'on voudrait obtenir la décidabilité de la question suivante :
Étant donné un transducteur bidirectionnel, existe-t-il un transducteur bidirectionnel apériodique réalisant la même fonction ?
Cette question, à laquelle on répond dans le cas des automates classiques par la déterminisation, devient ici difficile, et l'on peut espérer qu'elle devienne conséquence de définitions efficaces de logique et de monoïdes adaptés aux transducteurs.

# Index

# Bibliographie


[AČ10] Rajeev Alur and Pavol Černý. Expressiveness of streaming string transducers. In *30th International Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 8 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 1–12. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2010.

[AHU69] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. A general theory of translation. *Math. Systems Theory*, 3 :193–221, 1969.

[AU70] A. V. Aho and J. D. Ullman. A characterization of two-way deterministic classes of languages. *J. Comput. System Sci.*, 4 :523–538, 1970.

[Aui10] K. Auinger. On the decidability of membership in the global of a monoid pseudovariety. *Internat. J. Algebra Comput.*, 20(2) :181–188, 2010.

[AW98] J. Almeida and P. Weil. Profinite categories and semidirect products. *J. Pure Appl. Algebra*, 123(1-3) :1–50, 1998.

[BCST92] David A. Mix Barrington, Kevin Compton, Howard Straubing, and Denis Thérien. Regular languages in $NC^1$. *J. Comput. System Sci.*, 44(3) :478–499, 1992.

[Bir89] Jean-Camille Birget. Concatenation of inputs in a two-way automaton. *Theoret. Comput. Sci.*, 63(2) :141–156, 1989.

[Boj14] Mikolaj Bojanczyk. Transducers with origin information. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP (2)*, volume 8573 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 2014.

[Büc60] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math.*, 6 :66–92, 1960.

[Car08] Olivier Carton. *Langages formels, calculabilité et complexité*. Vuibert, 2008. 240 pages.

[Car12] Olivier Carton. Two-way transducers with a two-way output tape. In *Developments in language theory*, volume 7410 of *Lecture Notes in Comput. Sci.*, pages 263–272. Springer, Heidelberg, 2012.

[Cha] Laura Chaubard. *Méthodes algébriques pour les langages formels*. PhD thesis, Université Paris Diderot, 2007.

[CJ77] Michal P. Chytil and Vojtěch Jákl. Serial composition of 2-way finite-state transducers and simple programs on strings. In *Automata, languages and programming (Fourth Colloq., Univ. Turku, Turku, 1977)*, pages 135–137. Lecture Notes in Comput. Sci., Vol. 52. Springer, Berlin, 1977.

[Cou94] Bruno Courcelle. Monadic second-order definable graph transductions : a survey [see MR1251992 (94f :68009)]. *Theoret. Comput. Sci.*, 126(1) :53–75, 1994. Seventeenth Colloquium on Trees in Algebra and Programming (CAAP '92) and European Symposium on Programming (ESOP) (Rennes, 1992).

[CPS06] Laura Chaubard, Jean-Éric Pin, and Howard Straubing. First order formulas with modular predicates. In *21st Annual IEEE Symposium on Logic in Computer Science (LICS 2006)*, pages 211–220. IEEE, 2006.

[DGK08] Volker Diekert, Paul Gastin, and Manfred Kufleitner. A survey on small fragments of first-order logic over finite words. *Internat. J. Found. Comput. Sci.*, 19(3) :513–548, 2008.

[DP13] Luc Dartois and Charles Paperman. Two-variable first order logic with modular predicates over words. In Portier and Wilke [PW13], pages 329–340.

[DP14] Luc Dartois and Charles Paperman. Adding modular predicates. ANR 2010 BLAN 0202 02 FREC ; Fondation CFM pour la recherche, April 2014.

[EH01] Joost Engelfriet and Hendrik Jan Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Log.*, 2(2) :216–254, 2001.

[Eil76] Samuel Eilenberg. *Automata, languages, and machines. Vol. B.* Academic Press [Harcourt Brace Jovanovich Publishers], New York, 1976. With two chapters by Bret Tilson, Pure and Applied Mathematics, Vol. 59.

[EM65] C. C. Elgot and J. E. Mezei. On relations defined by generalized finite automata. *IBM J. Res. Develop*, 9 :47–68, 1965.

[ES76] Samuel Eilenberg and M. P. Schützenberger. On pseudovarieties. *Advances in Math.*, 19(3) :413–418, 1976.

[FKT14] Emmanuel Filiot, Shankara Narayanan Krishna, and Ashutosh Trivedi. First-order definable string transformations. In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, volume 29 of *LIPIcs*, pages 147–159. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.

[Gre51] J. A. Green. On the structure of semigroups. *Ann. of Math. (2)*, pages 163–172, 1951.

[Hop71] John Hopcroft. An $n$ log $n$ algorithm for minimizing states in a finite automaton. In *Theory of machines and computations (Proc. Internat. Sympos., Technion, Haifa, 1971)*, pages 189–196. Academic Press, New York, 1971.

[How95] John M. Howie. *Fundamentals of semigroup theory*, volume 12 of *London Mathematical Society Monographs. New Series*. The Clarendon Press Oxford University Press, New York, 1995. Oxford Science Publications.

[HU67] J. E. Hopcroft and J. D. Ullman. An approach to a unified theory of automata. *Bell System Tech. J.*, 46 :1793–1829, 1967.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley Publishing Co., Reading, Mass., 1979. Addison-Wesley Series in Computer Science.

[Imm82] Neil Immerman. Upper and lower bounds for first order expressibility. *J. Comput. System Sci.*, 25(1) :76–98, 1982.

[KL12] Manfred Kufleitner and Alexander Lauser. Lattices of logical fragments over words. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *ICALP (2)*, volume 7392 of *Lecture Notes in Computer Science*, pages 275–286. Springer, 2012.

[KL13] Manfred Kufleitner and Alexander Lauser. Quantifier alternation in two-variable first-order logic with successor is decidable. In Portier and Wilke [PW13], pages 305–316.

[Kle56] S. C. Kleene. Representation of events in nerve nets and finite automata. In *Automata studies*, Annals of mathematics studies, no. 34, pages 3–41. Princeton University Press, Princeton, N. J., 1956.

[Kna83] Robert Knast. A semigroup characterization of dot-depth one languages. *RAIRO Inform. Théor.*, 17(4) :321–330, 1983.

[KR65] Kenneth Krohn and John Rhodes. Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines. *Trans. Amer. Math. Soc.*, 116 :450–464, 1965.

[KS12] Andreas Krebs and Howard Straubing. An effective characterization of the alternation hierarchy in two-variable logic. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *FSTTCS*, volume 18 of *LIPIcs*, pages 86–98. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[KW12] Manfred Kufleitner and Pascal Weil. The $FO^2$ alternation hierarchy is decidable. In *Computer science logic 2012*, volume 16 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 426–439. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2012.

[KW13] Manfred Kufleitner and Tobias Walter. One quantifier alternation in first-order logic with modular predicates. *CoRR*, abs/1310.5043, 2013.

[MP71] Robert McNaughton and Seymour Papert. *Counter-free automata*. The M.I.T. Press, Cambridge, Mass.-London, 1971.

[MPT00] Alexis Maciel, Pierre Péladeau, and Denis Thérien. Programs over semigroups of dot-depth one. *Theoret. Comput. Sci.*, 245(1) :135–148, 2000. Semigroups and algebraic engineering (Fukushima, 1997).

[MSTV00] Pierre McKenzie, Thomas Schwentick, Denis Thérien, and Heribert Vollmer. The many faces of a translation. In *Automata, languages and programming (Geneva, 2000)*, volume 1853 of *Lecture Notes in Comput. Sci.*, pages 890–901. Springer, Berlin, 2000.

[Ner58] A. Nerode. Linear automaton transformations. *Proc. Amer. Math. Soc.*, 9 :541–544, 1958.

[Péc85] J.-P. Pécuchet. Automates boustrophédon, semi-groupe de Birget et monoïde inversif libre. *RAIRO Inform. Théor.*, 19(1) :71–100, 1985.

[Pél92] Pierre Péladeau. Logically defined subsets of $\mathbb{N}^k$. *Theoret. Comput. Sci.*, 93(2) :169–183, 1992.

[Pin86] Jean-Éric Pin. *Varieties of formal languages*. North Oxford, LondonPlenum, New-York, 1986. (Traduction de Variétés de langages formels).

[Pin97] Jean-Éric Pin. Syntactic semigroups. In *Handbook of formal languages, Vol. 1*, pages 679–746. Springer, Berlin, 1997.

[PP86] Dominique Perrin and Jean-Éric Pin. First-order logic and star-free sets. *J. Comput. System Sci.*, 32(3) :393–406, 1986.

[PW13] Natacha Portier and Thomas Wilke, editors. *30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, February 27 - March 2, 2013, Kiel, Germany*, volume 20 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[Rei82] Jan Reiterman. The Birkhoff theorem for finite algebras. *Algebra Universalis*, 14(1) :1–10, 1982.

[Sak03] Jacques Sakarovitch. *Éléments de théorie des automates*. Vuibert, 2003.

[Sch65] M. P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8 :190–194, 1965.

[Sch77] M. P. Schützenberger. Sur le produit de concaténation non ambigu. *Semigroup Forum*, 13(1) :47–75, 1976/77.

[She59] J. C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM J. Res. Develop.*, 3 :198–200, 1959.

[Str79] Howard Straubing. Families of recognizable sets corresponding to certain varieties of finite monoids. *J. Pure Appl. Algebra*, 15(3) :305–318, 1979.

[Str85] Howard Straubing. Finite semigroup varieties of the form $V * D$. *J. Pure Appl. Algebra*, 36(1) :53–94, 1985.

[Str89] Howard Straubing. The wreath product and its applications. In *Formal properties of finite automata and applications (Ramatuelle, 1988)*, volume 386 of *Lecture Notes in Comput. Sci.*, pages 15–24, Berlin, 1989. Springer.

[Str94] Howard Straubing. *Finite automata, formal logic, and circuit complexity.* Birkhäuser Boston Inc., Boston, MA, 1994.

[Tho82] Wolfgang Thomas. Classifying regular events in symbolic logic. *J. Comput. System Sci.*, 25(3) :360–376, 1982.

[Til87] Bret Tilson. Categories as algebra : an essential ingredient in the theory of monoids. *J. Pure Appl. Algebra*, 48(1-2) :83–198, 1987.

[TW99] Denis Thérien and Thomas Wilke. Over words, two variables are as powerful as one quantifier alternation. In *STOC '98 (Dallas, TX)*, pages 234–240. ACM, New York, 1999.