AIX MARSEILLE UNIVERSITÉ
E.D. 184
U.F.R. SCIENCES
Laboratoire d'informatique fondamentale de Marseille, CNRS UMR 7279

Thèse présentée pour obtenir le grade universitaire de docteur

Discipline : informatique

Mathieu CARALP

Problèmes de bornes pour les automates et les transducteurs à pile visible

Soutenue le 18/12/2015 devant le jury :

Véronique BRUYÈRE	Département d'informatique, Université de Mons	Rapporteur
Olivier CARTON	LIAFA, Université de Paris Diderot	Rapporteur
Benedikt BOLLIG	LSV, ENS Cachan, CNRS & Inria	Examinateur
Emmanuel FILIOT	Université Libre de Bruxelles	Examinateur
Denis LUGIEZ	LIF, Aix-Marseille Université	Examinateur
Pierre-Alain REYNIER	LIF, Aix-Marseille Université	Directeur de thèse
Jean-Marc TALBOT	LIF, Aix-Marseille Université	Directeur de thèse

Résumé

L'étude des automates est un sujet fondamental de l'informatique. Ce modèle apporte des solutions pratiques à divers problèmes en compilation et en vérification notamment. Dans ce travail nous proposons l'extension aux automates à pile visible de résultats existants pour les automates.

Nous proposons une définition d'automate à pile visible émondé et donnons un algorithme s'exécutant en temps polynomial émondant un automate en préservant son langage. Nous donnons aussi un algorithme de complexité exponentielle qui, pour un automate à pile visible donné, construit un automate équivalent à la fois émondé et déterministe. Cette complexité exponentielle se révèle optimale.

Étant donné un automate à pile visible, nous pouvons associer à ses transitions des coûts pris dans un semi-anneau $\mathcal{S}$. L'automate associe ainsi un mot d'entrée à un élément de $\mathcal{S}$. Le coût d'un automate est le supremum des coûts associés aux mots d'entrée. Pour les semi-anneaux des entiers naturels et Max-plus, nous donnons des caractérisations et des algorithmes polynomiaux pour décider si le coût d'un automate est fini. Puis, nous étudions pour les entiers naturels la complexité du problème de la majoration du coût par un entier k.

Les transducteurs à pile visibles produisent des sorties sur chaque mot accepté. Un problème classique est de décider s'il existe une borne sur le nombre de sorties de chaque mot accepté. Pour une sous-classe des transducteurs à pile visible, nous proposons des propriétés caractérisant les instances positives de ce problème. Nous montrons leur nécessité et discutons d'approches possibles afin de montrer leur suffisance.

Abstract

The study of automata is a central subject of computer science. This model provides practical solutions to several problems including compilation and verification. In this work we extend existing results of automata to visibly pushdown automata.

We give a definition of trimmed visibly pushdown automata and a polynomial time algorithm to trim an automata while preserving its language. We also provide an exponential time algorithm which, given a visibly pushdown automaton, produces an equivalent automaton, both deterministic and trimmed. We prove the optimality of the complexity.

Given a visibly pushdown automaton, we can equip its transitions with a cost taken from a semiring $\mathcal{S}$, and thus associate each input word to an element of $\mathcal{S}$. The cost of the automaton is the supremum of the input words cost. For the semiring of natural integers and Max-plus, we give characterisations and polynomial time algorithms to decide if the cost of a visibly pushdown automaton is finite. Then in the case of natural integers we study the complexity of deciding if the cost is bounded by a given integer k.

Visibly pushdown transducers produce output on each accepted word. A classical problem is to decide if there exists a bound on the number of outputs of each accepted word. In the case of a subclass of visibly pushdown transducers, we give properties characterizing positive instances of this problem. We show their necessity and discuss of possible approaches to prove their sufficiency.

Remerciements

On m'a dit qu'écrire les remerciements est la chose la plus difficile à faire lors de la rédaction d'une thèse. J'ai découvert que c'était surtout difficile de savoir où s'arrêter ! Alors si vous n'y êtes pas ne vous en offusquez pas, et profitez en pour lire le reste du manuscrit !

Je tiens tout d'abord à remercier ma famille, toujours présente à mes cotés quand il faut : mes parents, mon frère et ma soeur, je suis comme je suis grâce à vous. Vous m'avez toujours soutenu et je suis fier d'être un Caralp. Un jour je parviendrai à vous expliquer ce que je fais !

Je souhaite aussi remercier ma chérie Aline, qui est un soutien permanent dans les bons comme dans les mauvais moments. Tu es toujours là pour me motiver quand j'en ai besoin, et pour ça je ne t'en remercierai jamais assez !

Mes directeurs de thèse, Jean-Marc et Pierre-Alain, merci pour tout ! Ça n'a pas toujours été facile, et je sais que ça n'est pas toujours évident de traiter avec moi. Mais j'ai énormément appris grâce à vous, et je vous remercie de votre patience et de votre soutien. Jean-Marc, tu es un puit sans fond de connaissances. Tes vannes me manqueront, et j'attends toujours mon barbecue ! Pierre-Alain, tu es excessivement gentil et talentueux. Tu vannes moins mais quand tu t'y mets ça fait mal ! Vous faites bien la paire.

Je remercie aussi mon jury, Véronique Bruyère et Olivier Carton d'avoir accepté de relire mon manuscrit et aussi pour vos retours détaillés. Je remercie aussi Benedikt Bollig, Emmanuel Filliot et Denis Lugiez d'avoir accepté de faire parti du jury.

Je remercie le laboratoire pour son accueil. Je remercie tout particulièrement Nadine, Sylvie et Martine pour leur présence et leur gentilesse. Je remercie aussi tous les doctorants et post-doctorants, Florent, Makki, Élie, Antoine, Worachet, Christina, Didier, Laure et Luc Darviaud et Youssouf. Vous déchirez, le bureau me manque déjà ! Je remercie aussi Arnaud, avec qui j'ai pu animer le cours d'automates et circuits et théorie des ensembles et algèbre et dénombrement et tout ce que vous voulez.

Enfin, je tiens à remercier Peter, Guillaume et Yasmine, me permettant de faire des choses différentes mais tout autant passionnantes en parralèle de la thèse. Led's Chat vaincra !

Merci à tous !

“Sans menace de mort imminente, est-ce encore de la science ?”
— GLaDOS

Table des matières

Table des matières **6**

Introduction **9**

1 Définitions **13**
1.1 Quelques notions mathématiques . . . 13
1.2 Mots et arbres . . . 14
1.2.1 Mots . . . 14
1.2.2 Arbres . . . 16
1.3 Automates finis et automates d'arbres . . . 19
1.3.1 Automates finis . . . 19
1.3.2 Automates d'arbres . . . 21
1.4 Automates à pile visible . . . 23
1.4.1 Définition . . . 23
1.4.2 Propriétés des automates à pile visible . . . 26

2 Coût et transduction **33**
2.1 Automates à coûts . . . 33
2.2 Transducteurs . . . 35
2.3 Équivalence des automates d'arbres et des automates à pile visible . . . 39
2.3.1 Construire un automate d'arbre depuis un automate à pile visible . . . 40
2.3.2 Construire un automate à pile visible depuis un automate d'arbre . . . 44
2.3.3 Extension aux automates à coûts . . . 49
2.3.4 Extension aux transducteurs . . . 50

3 Émonder un automate à pile visible **55**
3.1 Préliminaires . . . 56
3.2 État de l'art . . . 57
3.3 Émonder un automate à pile visible bien imbriqué . . . 61
3.3.1 Construction de freduire(A) et reduire(A) . . . 61
3.3.2 Construction de coreduire(A)et fcoreduire(A) . . . 68
3.3.3 La procédure complète . . . 71
3.4 Émonder un automate à pile visible . . . 71
3.4.1 Des automates à pile visible arbitraires vers les bien imbriqués... . . . 72
3.4.2 ... Et vice-versa. . . . 75
3.4.3 La construction confpile . . . 80
3.4.4 La procédure complète . . . 86
3.5 Automate à pile visible déterministe et émondé . . . 86
3.5.1 La déterminisation préserve la réduction et la rétractabilité . . . 86

3.5.2 Construction d'un APV déterministe et émondé. 87
3.5.3 Une borne inférieure pour les entrées déterministes 87
3.6 Application aux automates à pile visible avec coûts 89
3.7 Résumé 91

4 Finitude et K-borne des automate à pile visible à coûts **93**
4.1 État de l'art pour le semi-anneau des entiers naturels 94
4.1.1 Décider si le coût d'un automate fini est infini 94
4.1.2 Décider si le coût d'un automate d'arbres est infini 94
4.2 Décidabilité du coût infini des automates à pile visible sur $\mathcal{N}$ 95
4.2.1 Caractérisation 95
4.2.2 La décision du coût infini est dans PTIME 104
4.3 Bornes finies du coût des automates à pile visible sur $\mathcal{N}$ 108
4.3.1 Décidabilité du coût K-borné 108
4.3.2 Décidabilité du coût K-borné (pour K fixé) 111
4.4 Retour aux arbres 113
4.5 Décidabilité du coût infini des automates à pile visible sur Max-plus 115
4.5.1 Automates d'arbres à coûts sur Max-plus 115
4.5.2 Caractérisation du coût infini d'un automate à pile visible sur Max-plus . 116
4.5.3 La décision du coût infini est dans PTIME 118

5 Décidabilité de l'infinité de la norme d'un transducteur à pile visible **119**
5.1 Caractérisation de la norme infinie d'un transducteur fini 120
5.1.1 Une caractérisation par les critères **(W1)** et **(W2)** 120
5.1.2 Un transducteur satisfaisant **(W1)** ou **(W2)** possède une norme infinie . . 121
5.1.3 Un transducteur avec une norme infinie satisfait **(W1)** ou **(W2)** 122
5.2 Caractérisation de la norme infinie d'un transducteur d'arbres 127
5.3 Nouvelle caractérisation de la norme infinie d'un transducteur fini 128
5.3.1 Définition de délai 128
5.3.2 Une nouvelle caractérisation par les critères **(C1)** et **(C2)** 129
5.3.3 Un transducteur satisfaisant **(C1)** ou **(C2)** possède une norme infinie . . . 130
5.3.4 Un transducteur avec une norme infinie satisfait **(C1)** ou **(C2)** 130
5.4 Transducteur de mot vers contexte 133
5.4.1 Définition 133
5.4.2 Équivalence avec les automates à pile visible linéaires 134
5.4.3 Extension aux transducteurs 139
5.5 Décidabilité de la norme finie transducteur mot-vers-contexte 140
5.5.1 Propriétés 140
5.5.2 Combinatoire des mots 142
5.5.3 Preuve de la condition suffisante 146
5.5.4 Discussion sur la condition nécessaire 158

Conclusion **163**

Bibliographie **164**

Introduction

Automates

Les automates finis constituent un modèle théorique depuis longtemps traité. Ses propriétés sont bien connues, et son étude constitue un classique de l'informatique théorique. Il permet de modéliser de nombreux problèmes, comme par exemple en model-checking où l'on souhaite vérifier qu'un programme correspond à une spécification ([CGP01]).

Les automates finis sont un modèle d'automate définissant la famille des langages réguliers et possédant, comme son nom l'indique, une mémoire finie. Ce modèle est étudié depuis longtemps et la recherche sur le sujet a été initiée dans les années 1950 par Stephen Cole Kleene dans [Kle56] et par Michael Oser Rabin et Dana Scott dans [RS59]. Il est de plus intéressant car il possède de bonnes propriétés de clôtures et de décidabilité. Nous pouvons par exemple citer l'union, l'intersection , le complémentaire ou la concaténation pour les propriétés de clôture et le test du vide, l'inclusion ou l'appartenance d'un mot pour les propriétés décidables ([HMU03]).

Les automates finis ont été étendus plus tard aux automates à pile par Noam Chomsky dans [Cho62] et aux automates d'arbres par John Doner dans [Don65] et par James W. Thatcher dans [Tha65] . Le premier modèle se différencie des automates finis par le le fait qu'il ne lit pas des mots, mais des arbres. Les arbres sont très utilisés pour modéliser des données représentées de manière arborescente, comme des fichiers XML ou JSON ([Sch07]). Par exemple, le document XML <*xml*><*a*></*a*><*b*></*b*></*xml*> est un arbre enraciné dans <*xml*></*xml*> et avec des feuilles <*a*></*a*> et <*b*></*b*>. Les automates à pile, quant à eux, étendent les automates finis à l'aide d'une pile, celle-ci permettant d'augmenter la puissance d'expressivité du modèle. Un automate à pile est donc capable d'exprimer des langages non reconnus par un automate fini. Nous pouvons par exemple citer le langage $a^n b^n$ où n est un entier naturel. Cependant ce modèle est plus complexe, et même s'il préserve certaines propriétés de clôture et de décidabilité des automates finis comme l'union ou la concaténation, de nombreuses autres ne fonctionnent plus, comme par exemple l'intersection ou le complémentaire ([HMU03]).

Dans ce manuscrit nous allons étudier un modèle plus récent, celui des automates à pile visibles. Il a été présenté par Rajeev Alur et Parthasarathy Madhusudan dans [AM04], dans le but de fournir à la fois une alternative aux automates d'arbres et une sous-classe des automates à pile possédant de nombreuses propriétés décidables.

Les automates à pile visible reconnaissent des mots imbriqués, c'est à dire des mots sur une structure arborescente. Par exemple, un fichier XML est un mot imbriqué où les balises ouvrantes et fermantes représentent la structure imbriquée. Le rapport avec les automates d'arbre devient alors évident : un mot imbriqué peut-être vu comme un arbre avec une arité nonbornée et réciproquement. A l'aide de cette traduction, nous pouvons montrer que les langages reconnus par les automates à pile visible et les automates d'arbres sont équivalents. Cependant l'étude de ce modèle reste intéressante car il propose une vision différente des automates d'arbres. En effet, un automate à pile visible ne lit pas simultanément l'ouvrant et le fermant

correspondant comme le fait un automate d'arbre. Il lit les lettres uniquement quand il en a besoin, ceci permettant une lecture en *streaming* d'un mot, contrairement aux automates d'arbres qui ont besoin de plus d'informations avant de pouvoir entamer la lecture. D'un point de vue pratique, nous pourrions comparer les automates à pile visible à l'API SAX([MB02]), et les automates d'arbres à l'API DOM ([HWW]). Elles permettent de lire et de traiter un fichier XML toutes les deux, mais d'un point de vue différent : la première propose une approche événementielle compatible avec une lecture en streaming, alors que la deuxième stocke tout l'arbre en mémoire avant de l'analyser.

Un automate à pile possède une mémoire supplémentaire et peut empiler ou dépiler des symboles de manière arbitraire, mais ce n'est pas le cas des automates à pile visible. Ce modèle définit un ensemble de lettres ouvrantes ou fermantes, à la manière d'un fichier XML avec ses balises ouvrantes ou fermantes. Les actions sur la pile sont restreintes par le type de lettre lu, et il n'est donc pas possible d'empiler et de dépiler sur la même lettre. Cette simple restriction nous assure que le modèle possède de bonnes propriétés de clôture, comme pour les automates finis. Par exemple, il est possible de construire l'intersection et le complément de deux automates à pile visible, contrairement aux automates à pile classiques.

Automates à coûts et transducteurs

Tous ces modèles permettent de définir des langages. Il est possible de les étendre en ajoutant une notion de sortie d'automate, par exemple un entier ou un mot. On parle alors d'automate à coût dans le premier cas et de transducteur dans le second.

Un automate à coût est un automate muni d'un semi-anneau, dans lequel un élément de ce semi-anneau équipe chaque transition. Il est alors possible de calculer le coût d'un mot en utilisant les opérateurs du semi-anneau. La théorie des automates finis à coûts a été introduite par Marcel-Paul Schützenberger dans [Sch61], et permet de modéliser des problèmes quantitatifs. Elle a été étendue aux automates d'arbres par Jean Berstel et Christophe Reutenauer dans [BR82].

Comme pour les automates finis, les automates à pile visible s'étendent tout aussi naturellement aux automates à pile visible à coûts. Ce modèle est équivalent à celui des automates d'arbres à coûts, comme dans le cas des automates sans sorties. Ils apportent ainsi une approche *streaming* absente des automates d'arbres à coûts.

Les transducteurs produisent un mot ou un arbre sur chaque transition, et peuvent donc se voir comme une machine reconnaissant une relation. A chaque entrée lue, une ou plusieurs sorties peuvent être produites. La théorie des transducteurs finis à été introduite par Seymour Ginsburg dans [Gin66], et a été étendue aux automates d'arbres par William C. Rounds dans [Rou68]. L'intérêt d'un tel modèle est évident en pratique, car il est naturel de vouloir réaliser des modifications sur une entrée. Nous pouvons par exemple citer XSLT ([Ama14]), une technologie réalisant la transformation de fichiers XML. De nombreux problèmes ont été étudiés par le passé sur ces deux modèles. Citons par exemple les problèmes de la fonctionnalité et de l'équivalence, le deuxième n'étant pas décidable dans le cas général. Pour les deux modèles, une sous-classe décidable est celle des k-normés ([Web93]), c'est à dire les transducteurs pour lesquels chaque mot d'entrée produit au plus k sorties, cette propriété étant aussi décidable ([Web90]).

Les automates à pile visible sont au croisement des automates à pile et des automates d'arbres. Ce n'est pas le cas des transducteurs. En effet, ce modèle introduit par Frédéric Servais dans [Ser11] possède une expressivité particulière incomparable avec celle des transducteurs d'arbres. Par exemple, un transducteur d'arbre peut copier ou échanger des sous-arbres de son entrée, ce que la lecture en *streaming* des automates à pile visible empêche. Inversement un

transducteur à pile visible peut aplatir son entrée, c'est à dire supprimer les ouvrants et les fermants, et peut même produire autre chose qu'un arbre, ce qu'un transducteur d'arbre ne peut pas faire. Pour ces raisons, les transformations produites sont différentes, et nous ne ne pouvons donc pas appliquer directement les résultats des transducteurs d'arbres aux transducteurs à pile visible, même s'ils représentent une bonne piste de réflexion.

Contribution

Dans ce manuscrit nous allons traiter plusieurs questions.

Émonder un automate à pile visible Tout d'abord, nous allons simplement considérer les automates à pile visible et le problème de l'émondage. Un automate fini est *émondé* si tout état peut-être traversé par un calcul acceptant, c'est à dire commençant par un état initial et finissant par un état final. Les intérêts d'un automate émondé sont multiples. Tout d'abord, tout calcul peut toujours être continué, il n'existe pas de cul-de-sac. En pratique cela assure un temps de parcours de l'automate plus faible. De plus, celui-ci est plus petit en mémoire. Nous allons présenter une définition d'automate à pile visible émondé, et montrer qu'il est possible d'en construire un en temps polynomial à partir d'un automate à pile visible arbitraire, ceci en préservant le langage. Nous traiterons ensuite du cas des automates à pile visible déterministes, et nous montrerons qu'il n'existe pas de méthode pour émonder un automate à pile visible déterministe en temps polynomial tout en préservant ce déterminisme. En revanche nous présenterons un algorithme pour déterminiser et émonder en même temps, en temps exponentiel. Enfin, nous étendrons nos résultats aux automates à pile visible à coût et aux transducteurs.

Finitude et K-borne des automates à pile visible à coûts Nous traiterons ensuite deux problèmes des automates à pile visible à coûts. Nous n'allons considérer que des semi-anneaux basés sur l'ensemble des entiers naturels.

Étant donné un automate à pile visible à coûts S, existe-t-il un entier k tel que les coûts de tous les mots acceptés par S sont inférieurs à k ?

Étant donnés un entier k et un automate à pile visible à coûts S, existe-t-il un mot accepté avec un coût supérieur à k ?

Nous allons voir que ces problèmes ont déjà étudiés dans les cas des automates finis, et nous allons proposer deux algorithmes pour décider ces deux problèmes dans le cas du semi-anneau des entiers naturels muni du produit et de la somme, le premier en temps polynomial et le deuxième en temps exponentiel. Nous prouverons aussi que le deuxième problème est ExpTime-difficile, et que si k et fixé non pas donné en entrée alors il peut-être être résolu en temps polynomial.

Nous parlerons aussi du premier problème dans le cas du semi-anneau Max-plus, c'est à dire du semi-anneau des entiers naturels muni des opérateurs max et de l'addition, ceci en utilisant les résultats existants pour les automates d'arbres à coûts.

Finitude des transducteurs à pile visible Enfin, nous traiterons du problème suivant des transducteurs à pile visible :

Étant donné un transducteur à pile visible T, existe-t-il un entier k tel que chaque entrée de T à au plus k sorties ?

Nous ne savons pas entièrement résoudre ce problème, mais nous présenterons l'état de l'art et des idées pour étendre ces résultats à une sous-classe des transducteurs à pile visible, les transducteurs à pile visible linéaires. Afin de caractériser les transducteurs satisfaisant l'énoncé ci-dessus, nous allons introduire deux propriétés. Nous démontrerons qu'elles sont nécessaires, et expliquerons pourquoi nous pensons qu'elles sont aussi suffisantes.

Organisation du manuscrit

Ce manuscrit se divise en cinq chapitres.

Le premier Chapitre présentera de manière non-exhaustive les modèles d'automates utilisés, c'est à dire les automates finis, les automates d'arbres et les automates à pile visible. Ensuite, nous définirons plusieurs propriétés des automates à pile visible que nous utiliserons dans le manuscrit.

Dans le Chapitre 2, nous présenterons les modèles d'automates à coûts et de transducteurs que nous utiliserons par la suite. Puis nous traiterons des équivalences entre les automates à pile visibles et les automates d'arbres, que nous étendrons aux transducteurs et aux automates à coûts.

Au Chapitre 3 nous présenterons une définition d'émondage pour les automates à pile visible. Nous commencerons par faire un état de l'art sur les différents modèles que nous avons définis, puis nous présenterons nos algorithmes pour émonder un automate à pile visible. Ensuite nous traiterons le cas des automates à pile visible déterministes, des automates à pile visible à coûts et des transducteurs.

Les automates à pile visible à coûts seront abordés dans le Chapitre 4. Comme pour le chapitre précédent nous allons commencer par un état de l'art. Nous présenterons ensuite dans un premier temps nos algorithmes pour décider des deux problèmes présentés dans le cas du semi-anneau des entiers naturels muni du produit et de la somme, puis dans un second temps nous traiterons du premier problème présenté dans le cas du semi-anneau Max-plus.

Enfin, au chapitre 5, nous parlerons de transducteurs à pile visible. Encore une fois, nous commencerons par un état de l'art, puis nous présenterons la sous-classe des transducteurs à pile visible linéaires. Pour cette sous-classe nous donnerons un ensemble de propriétés que nous pensons caractériser le problème présenté. Nous montrerons alors dans un premier temps que ces propriétés sont nécessaires, puis dans un second temps nous introduirons nos idées basées sur des résultats existants quant à leur suffisance.

Chapitre 1

Définitions

Dans ce premier chapitre nous allons présenter diverses notions utilisées par la suite. Nous allons discuter de plusieurs modèles classiques, les automates finis et les automates d'arbres, puis nous parlerons ensuite d'un modèle plus récent, celui des automates à pile visible. Les deux premiers modèles nous serviront principalement pour étudier les résultats déjà existants et les comparer à nos solutions, et le dernier modèle est celui que nous traiterons le plus en détail. Dans ce but nous présenterons à la fin de ce chapitre quelques résultats simples et que nous utiliserons tout le long du manuscrit.

Plan du chapitre Premièrement dans la Section 1.1, nous allons introduire quelques notions préliminaires. Nous donnons les définitions de mots et d'arbres dans la Section 1.2. Dans la Section 1.3 nous présenterons les modèles d'automates finis et d'arbres utilisés ainsi que les différentes propriétés liées à ces modèles. Enfin, nous prouverons les automates à pile visible et plusieurs résultats sur ses propriétés basiques à la Section 1.4.2

1.1 Quelques notions mathématiques

Avant de présenter les modèles d'automates utilisés nous allons discuter des notations basiques que nous utiliserons par la suite.

Ensemble Nous appelons $\mathbb{N}$ l'ensemble des entiers naturels et pour $n \in \mathbb{N}$ nous appelons $\mathbb{N}_{>n}$ l'ensemble $\mathbb{N}\backslash\{0, \ldots, n\}$. Étant donnés deux entiers $n, n' \in \mathbb{N}$, nous notons $[n, n']$ l'ensemble $\{i \mid n \leqslant i \leqslant n'\}$.

Soit E un ensemble fini, nous notons par $\#E$ la cardinalité de E. Soient E_1, E_2 $\ldots E_n$ avec $n \in \mathbb{N}$ des ensembles, nous notons par $E_1 \times E_2 \times \ldots \times E_n$ l'ensemble des n-uplet sur E_1, E_2, $\ldots$, E_n. Nous appellerons un 2-uplet un *couple*. De plus si pour tout $i, j \in [1, n]$ nous avons $E_i = E_j$ alors nous écrivons $E_1 \times E_2 \times \ldots \times E_n$ sous la forme E_1^n. Soit $(e_1, e_2, \ldots, e_n) \in E_1 \times E_2 \times \ldots \times E_n$ un *n-uplet*, nous définissons la projection $\pi_i(e_1, e_2, \ldots, e_n)$ avec $i \in [1, n]$ comme l'élément e_i.

Séquence Soit E un ensemble, nous notons E^* l'ensemble des séquences finies sur E. Soit $n \in \mathbb{N}$ un entier. La séquence $s = (e_1, e_2, \ldots, e_n) \in E^*$ peut être aussi écrite $e_1 e_2 \ldots e_n$ ou $e_1 \cdot e_2 \cdot \ldots \cdot e_n$ si le contexte le permet. Nous appelons ϵ la séquence vide. Étant donnée une deuxième séquence $s' = (e'_1, e'_2, \ldots, e'_{n'}) \in E^*$ avec $n' \in \mathbb{N}$ nous définissons ss' ou $s \cdot s'$ comme étant la concaténation de s et s', c'est à dire $ss' = (e_1, e_2, \ldots, e_n, e'_1, e'_2, \ldots, e'_{n'})$. Soit $\ll$ un ordre total sur les éléments de E. Nous définissons l'extension lexicographique sur E^* de $\ll$ de la manière suivante. Nous avons $s \preceq s'$ si et seulement si :

- il existe $i \in [1, \min(\{n, n'\})]$ tel que $e_1 = e'_1, \ldots, e_{i-1} = e'_{i-1}$, $e_i \neq e'_i$ et $e_i \ll e'_i$, ou
- $n < n'$ et pour tout $i \in [1, n]$ nous avons $e_i = e'_i$.

Matrice Étant donnés trois ensembles E_1, E_2 et V tels que E_1 et E_2 sont finis, nous appelons V^{E_1,E_2} l'ensemble des matrices où les lignes sont indexées par les éléments de E_1, les colonnes sont indexées par les éléments de E_2 et chaque valeur de la matrice est un élément de V. Pour tout $e_1 \in E_1$, $e_2 \in E_2$ et $M \in V^{E_1,E_2}$, nous appelons $M(e_1, e_2)$ la valeur $v \in V$ à la ligne e_1 et à la colonne e_2 de M. Si $\#E_2 = 1$ alors V^{E_1,E_2} est un ensemble de vecteurs et se réécrit V^{E_1} et pour tout $e_1 \in E_1$, $e_2 \in E_2$ et $M \in V^{E_1,E_2}$, $M(e_1, e_2)$ se réécrit $M(e_1)$.

Monoïde Un *monoïde* $\mathcal{M}$ est une structure algébrique $(E, \otimes, \mathbf{1})$ avec E un ensemble, $\otimes : E^2 \to E$ une opération et $\mathbf{1} \in E$ tels que :

- Pour tout x, y, z de E, $x \otimes (y \otimes z) = (x \otimes y) \otimes z$ (associativité), et
- Pour tout x de E, $x \otimes \mathbf{1} = \mathbf{1} \otimes x = x$ ($\mathbf{1}$ est neutre).

De plus, Nous dirons que $\mathcal{M}$ est un monoïde *commutatif* si pour tout x, y de E, $x \otimes y = y \otimes x$.

Semi-anneau Un semi-anneau $\mathcal{S}$ est une structure algébrique $(E, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ avec E un ensemble, $\oplus : E^2 \to E$ et $\otimes : E^2 \to E$ des opérations et $\mathbf{0}, \mathbf{1} \in E$ tels que :

- $(E, \oplus, \mathbf{0})$ est un monoïde commutatif,
- $(E, \otimes, \mathbf{1})$ est un monoïde,
- Pour tout $x, y, z \in E$, $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$ (distributivité à gauche),
- Pour tout $x, y, z \in E$, $(x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z)$ (distributivité à droite), et
- Pour tout $x \in E$, $x \otimes \mathbf{0} = \mathbf{0} \otimes x = \mathbf{0}$ ($\mathbf{0}$ est absorbant).

Nous appelons $\mathcal{S}$ un semi-anneau *commutatif* si $(E, \otimes, \mathbf{1})$ est un monoïde commutatif.

1.2 Mots et arbres

Les automates sont des objets mathématiques reconnaissant des langages. Dans cette section nous allons présenter les notions de langages de mots et d'arbres, reconnus respectivement par les automates finis et les automates d'arbres. Nous allons aussi présenter plusieurs sous-classes de langages de mots utilisés par les automates à pile visible, ainsi que des notions de contextes de mots ou d'arbres.

1.2.1 Mots

Alphabet Soit Σ un ensemble fini, que nous appelons un *alphabet*. Les éléments de Σ sont appelés *lettres*. Les séquences finies de Σ^* sont appelées *mots*. Un sous-ensemble L de Σ^* est appelé *langage*. Toutes les opérations sur les séquences se reportent naturellement sur les mots.

Taille d'un mot Soient $a \in \Sigma$ une lettre et $w \in \Sigma^*$ un mot. Nous notons $|w|_a$ la longueur de w en a, formellement définie par :

$$|w|_a = \begin{cases} 0 & \text{si } w = \epsilon, \\ 1 + |w'|_a & \text{si } w = aw' \text{ avec } w' \in \Sigma^*, \\ 0 + |w'|_a & \text{si } w = bw' \text{ avec } w' \in \Sigma^*, b \in \Sigma \text{ et } a \neq b. \end{cases}$$

Nous notons $|w|$ la longueur de w, c'est à dire :

$$|w| = \sum_{a \in \Sigma} |w|_a$$

Soient $n \in \mathbb{N}$ un entier et $w = a_1 \dots a_n \in \Sigma^*$ un mot. Soient $i, j \in [1, n]$ deux entiers tels que $i \leqslant j$. Nous notons $w(i, j)$ le sous-mot $a_i \dots a_{j-1}$. Si $i + 1 = j$ alors $w(i, j)$ peut-être réécrit $w(i)$, et si $j = |u| + 1$ alors $w(i, j)$ peut-être réécrit $w(i \dots)$. Notez que si $i = j$ alors $w(i, j) = \epsilon$.

Alphabet structuré Soit Δ un alphabet. Δ est appelé *alphabet structuré* s'il existe trois alphabets Δ_c, Δ_r et Δ_ι tels que Δ est l'union disjointe de Δ_c, Δ_r et Δ_ι. Une lettre de Δ_c est appelée un ouvrant, de Δ_r un fermant et de Δ_ι un interne. L'ensemble des mots bien imbriqués Δ^*_{bi} est le plus petit sous-ensemble de Δ^* tel que :

- $\epsilon \in \Delta^*_{\mathsf{bi}}$,
- Pour tout $a \in \Delta_\iota$ et tout $w \in \Delta^*_{\mathsf{bi}}$, nous avons $aw \in \Delta^*_{\mathsf{bi}}$,
- Pour tout $c \in \Delta_c$, tout $r \in \Delta_r$ et $w, w' \in \Delta^*_{\mathsf{bi}}$ nous avons $cwrw' \in \Delta^*_{\mathsf{bi}}$

Un mot bien imbriqué est un mot sur lequel l'alphabet induit une notion de correspondance. Typiquement, un tel mot devrait faire correspondre une lettre de Δ_c à une lettre de Δ_r, et nous appelons ceci une *correspondance*. Soit $w = w_1 c w w_2 r w_3 \in \Delta^*_{\mathsf{bi}}$ avec $c \in \Delta_c$, $r \in \Delta_r$ et $w_1, w_2, w_3 \in \Delta^*_{\mathsf{bi}}$. Nous disons que c *correspond* avec r. Si c ne correspond avec aucune lettre, alors c est *sans correspondance*, il en va de même pour r. De plus, nous appelons r le *fermant* de c et c l'*ouvrant* de r.

Exemple 1.2.1

Soient Δ un alphabet structuré et $w = c_1 a r_1 c_2 c_3 b r_3 c_4 r_4 r_3 r_2 \in \Delta^*_{\mathsf{bi}}$ avec $c_1, c_2, c_3, c_4 \in \Delta_c$, $r_1, r_2, r_3, r_4 \in \Delta_r$ et $a, b \in \Delta_\iota$. La lettre c_i correspond avec r_i pour $i \in [1, 4]$. Notez que comme $w \in \Delta^*_{\mathsf{bi}}$ chaque lettre de Σ_c possède un fermant, et chaque lettre de Σ_r possède un ouvrant. De plus w n'est pas linéaire, car il contient des lettres de Δ_c après la lettre r_1.

L'ensemble des mots linéaires Δ^*_{lin} est le plus petit sous-ensemble de Δ^* tel que :

- $\Delta^*_\iota \subseteq \Delta^*_{\mathsf{lin}}$,
- Pour tout $c \in \Delta_c$, $r \in \Delta_r$, $u, v \in \Delta^*_\iota$ et tout $w \in \Delta^*_{\mathsf{lin}}$, nous avons $ucwrv \in \Delta^*_{\mathsf{lin}}$.

Notez que nous avons $\Delta^*_{\mathsf{lin}} \subset \Delta^*_{\mathsf{bi}}$. Un mot linéaire ne peut plus contenir de lettre de Δ_c une fois une lettre de Δ_r rencontrée dans un parcours de gauche à droite.

Soit $u \in \Delta^*_{\mathsf{bi}}$ un mot bien imbriqué. La *hauteur* du mot u en position $i \in [1, |u|]$ est l'entier $h_u(i)$ défini par :

$$h_u(i) = \#\{j \in [1, i] \mid u(j) \text{ est un ouvrant sans correspondance dans } u(1, i)\}$$

Contexte de mot Soit Σ un alphabet . Nous définissons un *contexte de mot* c comme une paire de mots de $\Sigma^* \times \Sigma^*$. Étant donnés deux contextes de mot $c = (u, v), c' = (u', v') \in \Sigma^* \times \Sigma^*$, nous appelons cc' le contexte $(uu', v'v)$ produit par la *concaténation* de c et c'. Étant donné un mot $w \in \Sigma^*$, nous appelons le mot uwv le produit de la *concaténation* de c et w, et la dénotons par $c[w]$.

Exemple 1.2.2

Soient c_1 et c_2 les contextes $c_1 = (a, \epsilon)$ et $c_2 = (\epsilon, a)$. Pour tout $n, m \in \mathbb{N}$ nous avons $c_1^n c_2^m = c_2^m c_1^n = (a^n, a^m)$. A noter que $c_1^n \neq c_2^n$, mais $c_1^n[\epsilon] = c_2^n[\epsilon]$.

1.2.2 Arbres

Alphabet gradué Nous appelons *alphabet gradué* un couple (Σ, ar) où Σ est un alphabet et ar une fonction d'arité $\mathsf{ar} : \Sigma \to \mathbb{N}$. Pour toute lettre a de Σ, nous notons $\mathsf{ar}(a)$ l'*arité* de a. L'ensemble des lettres de Σ d'arité r est noté Σ_r. Nous appelons $\mathcal{T}_\Sigma$ le plus petit ensemble tel que :

- $\Sigma_0 \subseteq \mathcal{T}_\Sigma$,
- Si $f \in \Sigma_m$ et $t_1, \ldots, t_m \in \mathcal{T}_\Sigma$ alors $f(t_1, \ldots, t_m) \in \mathcal{T}_\Sigma$.

Nous appellerons *arbre* un élément de $\mathcal{T}_\Sigma$. Un sous-ensemble L de $\mathcal{T}_\Sigma$ est appelé *langage d'arbres*. Il existe aussi des alphabets gradués ne possédant pas d'arité, un symbole pouvant donc apparaître plusieurs fois dans un arbre avec des arités différentes. Un tel arbre possède une arité *non-bornée*.

Exemple 1.2.3

Nous considérons l'alphabet gradué (Σ, ar) avec $f \in \Sigma_2$ et $a, b \in \Sigma_0$. $f(a, f(a, b))$, a, $f(b, b)$ sont des arbres de $\mathcal{T}_\Sigma$. L'arbre $f(a, f(a, b))$ est représenté à la Figure 1.1.

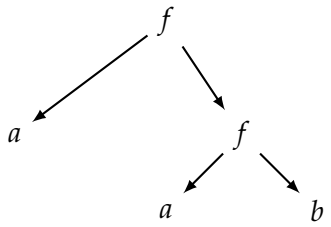


FIGURE 1.1 – Un arbre $f(a, f(a, b))$, avec $f \in \Sigma_2$ et $a, b \in \Sigma_0$.

Positions d'un arbre Soit $t = f(t_1, t_2, \ldots, t_m) \in \mathcal{T}_\Sigma$ un arbre avec $m \in \mathbb{N}$, $f \in \Sigma_m$ et $t_i \in \mathcal{T}_\Sigma$ pour tout $i \in [1, m]$. L'ensemble des *positions* de t, noté $\mathsf{pos}(t)$, est le sous-ensemble de $\mathbb{N}^*$ (séquences d'entiers) défini par :

$$\mathsf{pos}(t) = \{\epsilon\} \cup \{i \cdot r \mid i \in [1, m], r \in \mathsf{pos}(t_i)\}$$

De plus pour tous $r \in \mathsf{pos}(t)$ et $t' \in \mathcal{T}_\Sigma$ nous définissons les notations suivantes :

$$t\langle r\rangle = \begin{cases} f & \text{si } r = \epsilon \\ t_i\langle r'\rangle & \text{si } r = i \cdot r' \text{ et } t = f(t_1, \ldots, t_m) \text{ avec } m \in \mathbb{N} \end{cases}$$

$$t|_r = \begin{cases} t & \text{si } r = \epsilon \\ t_i|_{r'} & \text{si } r = i \cdot r' \text{ et } t = f(t_1, \ldots, t_m) \text{ avec } m \in \mathbb{N} \end{cases}$$

$$t[r \leftarrow t'] = \begin{cases} t' & \text{si } r = \epsilon \\ f(t_1, \ldots, t_i[r' \leftarrow t'], \ldots, t_m) & \text{si } r = i \cdot r' \text{ et } t = f(t_1, \ldots, t_m) \text{ avec } m \in \mathbb{N} \end{cases}$$

La notation $t\langle r\rangle$ renvoie le symbole associé à la position r dans t et $t|_r$ renvoie le sous-arbre de t enraciné en position r. Étant donné un alphabet gradué (Σ, ar) et un arbre $t \in \mathcal{T}_\Sigma$, nous appelons *racine* de t la position ϵ et par *feuille* les éléments $r \in \mathsf{pos}(t)$ tels que $t\langle r\rangle \in \Sigma_0$.

Taille d'un arbre Soient $f \in \Sigma_m$ une lettre avec $m \in \mathbb{N}$ et $t \in \mathcal{T}_\Sigma$ un arbre. Nous notons $|t|_f$ le nombre d'occurrences de f dans t, formellement défini par :

$$|t|_f = \begin{cases} 1 + \sum_{i \in [m]} |(t|_i)|_f & \text{si } t = f(t|_1, \ldots t|_m) \text{ avec } m \in \mathbb{N}, \\ 0 + \sum_{i \in [m']} |(t|_i)|_f & \text{si } t = f'(t|_1, \ldots t|_{m'}), \text{ avec } f' \in \Sigma_{m'} \text{ et } f \neq f'. \end{cases}$$

Enfin nous notons $|t|$ la taille de t, c'est à dire :

$$|t| = \sum_{f \in \Sigma} |t|_f$$

Exemple 1.2.4

Soient (Σ, ar) un alphabet structuré et $t = g(a, f(a, f(a, a)), g(a, b, b)) \in \mathcal{T}_\Sigma$ un arbre avec $g \in \Sigma_3$, $f \in \Sigma_2$ et $a, b \in \Sigma_0$. Nous avons :

$$\mathsf{pos}(t) = \{\epsilon, 1, 2, 2.1, 2.2, 2.2.1, 2.2.2, 3, 3.1, 3.2, 3.3\}$$

Si $n = 2.2$ alors $t\langle r\rangle = f$ et $t|_n = f(a, a)$. De plus nous avons $|t|_g = 2$, $|t|_f = 2$, $|t|_a = 5$, $|t|_b = 2$ et $|t| = |t|_g + |t|_f + |t|_a + |t|_b = 11$.

Contexte d'arbre Soit (Σ, ar) un alphabet gradué et $\mathcal{X}$ un ensemble fini de variables distinctes de Σ. Nous définissons un contexte d'arbre t comme un arbre sur l'alphabet $\Sigma \cup \mathcal{X}$, et nous fixons à 0 l'arité des variables de $\mathcal{X}$. Les contextes d'arbre peuvent être vus comme des termes avec des variables. Afin de mettre en exergue les variables, nous notons $\mathcal{T}_{\Sigma \cup \mathcal{X}}$ par $\mathcal{T}_\Sigma(\mathcal{X})$. Nous appelons t *sans copies* si pour tout $x \in \mathcal{X}$ nous avons $|t|_x \leqslant 1$, *sans pertes* si pour tout $x \in \mathcal{X}$ nous avons $|t|_x > 0$ et *exact* si pour tout $x \in \mathcal{X}$ nous avons $|t|_x = 1$. Soit $\ll$ une relation d'ordre total sur les variables de $\mathcal{X}$ et $\preceq$ l'extension lexicographique de cet ordre. Nous appelons t *sans échanges* si pour toutes positions $r, r' \in \mathsf{pos}(t)$ telles que $r \preceq r'$ et $t\langle r\rangle, t\langle r'\rangle \in \mathcal{X}$ alors $t\langle r\rangle \ll t\langle r'\rangle$.

Étant donnés un contexte d'arbre $t \in \mathcal{T}_\Sigma(\mathcal{X})$, des variables distinctes deux à deux $x_1, \ldots, x_n \in \mathcal{X}$ et des contextes d'arbre $t_1, \ldots, t_n \in \mathcal{T}_\Sigma(\mathcal{X})$ avec $n \in \mathbb{N}$, la *substitution d'arbres* de $t_1, \ldots, t_n$ pour $x_1, \ldots, x_n$ dans t, que nous dénotons $\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}(t)$ est définie récursivement de la manière suivante. Soit $\theta = \{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$:

$$\theta(t) = \begin{cases} t_i & \text{si } t = x_i \text{ avec } i \in [1, n], \\ f(\theta(t|_1), \ldots, \theta(t|_m)) & \text{si } t = f(t|_1, \ldots, t|_m) \text{ avec } m \in \mathbb{N} \text{ et } f \in \Sigma_m. \end{cases}$$

Enfin étant donné $\ll$ une relation d'ordre total sur les variables de $\mathcal{X}$ nous notons par $t[t_1, \ldots, t_n]$ le contexte d'arbre produit par $\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}(t)$ où $x_1 \ll x_2 \ldots \ll x_n$.

Exemple 1.2.5

Nous considérons l'ensemble de variables $\mathcal{X} = \{x_1\}$ et l'alphabet Σ avec $f \in \Sigma_2$ et $a, b \in \Sigma_0$. Nous notons $t_1 = f(x_1, b)$ et $t_2 = f(f(x_1, b), a)$ deux contextes d'arbres de $\mathcal{T}_\Sigma(\mathcal{X})$. Le contexte $f(f(f(x_1, b), a), b)$ est produit par $t_1[t_2[x_1]]$, c'est à dire par la substitution d'arbres de t_2 pour x_1 dans t_1. Ce contexte est représentée à la Figure 1.2.

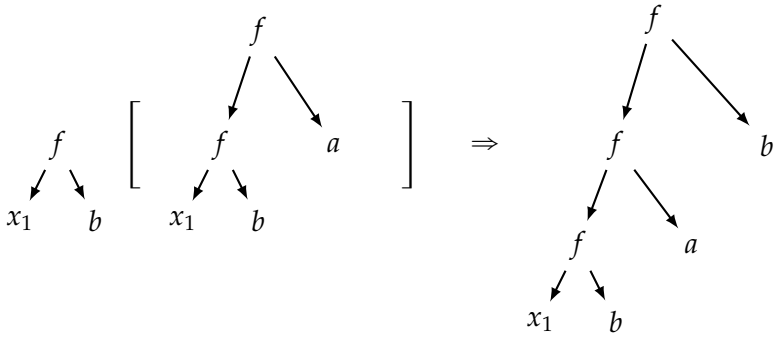


FIGURE 1.2 – Le contexte $t_1[t_2[x_1]]$.

Production d'arbres depuis un mot Soit Δ un alphabet structuré. Nous allons définir une fonction produisant un arbre à partir d'un mot de Δ^*_{bi}. Nous considérons $\top$ comme un symbole spécial distinct de Δ. Nous construisons l'alphabet gradué $(\tilde{\Delta}, \mathsf{ar})$ avec :

$$\tilde{\Delta} = \{(c, r) \mid c \in \Delta_c \text{ et } r \in \Delta_r\} \cup \Delta_\iota \cup \{\top\}$$

et $\mathsf{ar}((c, r)) = 2$ pour tout $c \in \Delta_c$ et $r \in \Delta_r$, $\mathsf{ar}(a) = 1$ pour tout $a \in \Delta_\iota$ et $\mathsf{ar}(\top) = 0$. Nous définissons la fonction $\mathsf{fcns} : \Delta_{\mathsf{bi}}* \to \mathcal{T}_{\tilde{\Delta}}$ de la manière suivante :

$$\mathsf{fcns}(w) = \begin{cases} (c, r)(\mathsf{fcns}(u), \mathsf{fcns}(v)) & \text{si } w = curv, c \in \Delta_c, r \in \Delta_r \text{ et } u, v \in \Delta^*_{\mathsf{bi}}, \\ a(\mathsf{fcns}(v)) & \text{si } w = av, a \in \Delta_\iota \text{ et } v \in \Delta^*_{\mathsf{bi}}, \\ \top & \text{si } w = \epsilon. \end{cases}$$

Intuitivement, fcns produit le codage *first-child-next-sibling* du mot d'entrée. Ce codage est utilisé pour transformer un arbre dont les nœuds sont d'arité non bornée en un arbre où les nœuds sont d'arité 2 ou moins. Avec ce codage, le *frère* d'un nœud de l'arbre est transformé en son second fils, son premier fils restant le même. Un mot sur un alphabet structuré est un un arbre d'arité non-bornée, et la construction fcns transforme son entrée en arbre d'arité bornée par 2.

Exemple 1.2.6

Soit Δ un alphabet structuré tel que $\Delta_c = \{c\}$, $\Delta_r = \{r\}$ et $\Delta_\iota = \{a, b\}$. Nous avons $\tilde{\Delta} = \{(c, r), a, b, \top\}$, $\mathsf{ar}((c, r)) = 2$, $\mathsf{ar}(a) = 1$, $\mathsf{ar}(b) = 1$ et $\mathsf{ar}(\top) = 0$. Nous considérons les mots *acbr* et *carcbr*. La construction fcns produit les arbres $a((c, r)(b(\top), \top))$ et $(c, r)(a(\top), (c, r)(b(\top), \top))$.

Production de mots imbriqués Soit (Σ, ar) un alphabet gradué. Nous allons maintenant définir une fonction produisant un mot à partir d'un arbre de $\mathcal{T}_\Sigma$. Nous définissons l'alphabet

structuré $\hat{\Sigma} = \hat{\Sigma}_c \cup \hat{\Sigma}_r \cup \hat{\Sigma}_\iota$ comme suit :

$$\hat{\Sigma}_c = \{\langle a \mid a \in \Sigma\}$$
$$\hat{\Sigma}_r = \{a\rangle \mid a \in \Sigma\}$$
$$\hat{\Sigma}_\iota = \varnothing$$

Nous présentons maintenant la construction $\mathsf{enc} : \mathcal{T}_\Sigma \to \hat{\Sigma}^*_{\mathsf{bi}}$ (encodage) :

$$\mathsf{enc}(t) = \langle a \cdot \mathsf{enc}(t|_1) \dots \mathsf{enc}(t|_m) \cdot a\rangle \text{ avec } m \in \mathbb{N}, a \in \Sigma_m \text{ et } t = a(t|_1, \dots t|_m)$$

Cette construction produit une linéarisation correspondant à un parcours en profondeur de l'arbre. Il est facile de voir que $\mathsf{enc}(t)$ est un mot bien imbriqué. Vous pouvez noter que pour tout $t \in \mathcal{T}_\Sigma$, t n'est pas égal à $\mathsf{fcns}(\mathsf{enc}(t))$, et cela même si nous renommons les lettres. Cela vient du fait que les transformations sont de natures différentes : fcns produit un codage *first-child-next-sibling*, changeant la structure de l'arbre. Il aurait été possible de considérer l'inverse de ce codage pour enc, mais la construction présentée est plus simple et plus naturelle. Par contre, il n'aurait pas été possible d'adapter enc en préservant la structure de l'arbre, car un mot bien imbriqué est composé d'une suite d'arbres.

Nous appelons $\hat{\Sigma}^*_{\mathsf{cr}}$ l'ensemble des mots $w \in \hat{\Sigma}^*_{\mathsf{bi}}$ tels que $w = cur$ avec $c \in \hat{\Sigma}_c$, $r \in \hat{\Sigma}_r$ et $u \in \hat{\Sigma}^*_{\mathsf{bi}}$. Pour tout $t \in \mathcal{T}_\Sigma$ nous avons $\mathsf{enc}(t) \in \hat{\Sigma}^*_{\mathsf{cr}}$. Comme $\hat{\Sigma}_\iota = \varnothing$, le mot u peut se décomposer de la manière suivante :

$$u = u_1 \dots u_l \text{ avec } l \in \mathbb{N} \text{ et } u_i \in \hat{\Sigma}^*_{\mathsf{cr}} \text{ pour tout } i \in [1, l]$$

Exemple 1.2.7

Soit (Σ, ar) un alphabet gradué tel que $\Sigma = \{f, g, a, b\}$, $\mathsf{ar}(f) = 2$, $\mathsf{ar}(g) = 1$, $\mathsf{ar}(a) = 0$ et $\mathsf{ar}(b) = 0$. Nous avons $\hat{\Sigma}_c = \{\langle f, \langle g, \langle a, \langle b\}$ et $\hat{\Sigma}_r = \{f\rangle, g\rangle, a\rangle, b\rangle\}$. Nous considérons les arbres $t = g(f(a, b))$ et $t' = f(g(a), g(b))$. La construction enc produit les mots imbriqués $\langle g\langle f\langle aa\rangle\langle bb\rangle f\rangle g\rangle$ et $\langle f\langle g\langle aa\rangle g\rangle\langle g\langle bb\rangle g\rangle f\rangle$.

1.3 Automates finis et automates d'arbres

Nous allons maintenant présenter plusieurs modèles d'automates. Nous appelons *Automate* une machine possédant des états (en nombre fini ou infini) et un ensemble de transitions permettant de se déplacer d'un état à un autre. Chaque transition est étiquetée par une lettre d'un alphabet d'entrée, ceci définissant le langage de l'automate.

1.3.1 Automates finis

Le modèle d'*automates finis* (*finite state automata* ou AF) est un modèle ancien étudié depuis longtemps, et très pratique pour modéliser de nombreuses problématiques, par exemple la vérification de systèmes informatique (voir [Bü60], [Elg61]). Une présentation complète et exhaustive peut-être trouvée dans [Sak09]. Dans cette section nous fixons les notations et la sémantique liées aux AF.

Définition 1.3.1: Automate fini

Soit Σ un alphabet. Un *automate fini* A sur Σ est un 4-uplet (Q, I, F, δ) avec :

- Q un ensemble fini d'états,
- $I \subseteq Q$ un ensemble d'états initiaux,
- $F \subseteq Q$ un ensemble d'états finaux,
- $\delta \subseteq Q \times \Sigma \times Q$ un ensemble fini de transitions.

Pour la suite de la section nous considérons l' AF $A = (Q, I, F, \delta)$ sur l'alphabet Σ. Pour toute transition $\tau = (p, a, q)$ de δ, nous définissons les fonctions $\mathsf{source}(\tau) = p$, $\mathsf{etiq}(\tau) = a$ et $\mathsf{cible}(\tau) = q$.

Calcul Un *calcul* de A sur un mot $w = a_1 \dots a_n \in \Sigma^*$ avec $n \in \mathbb{N}$, d'un état p de Q à un état q de Q est une séquence de transitions $\rho = (\tau_1, \dots, \tau_n)$ telle que pour tout $i \in [1, n-1]$ nous avons $\mathsf{cible}(\tau_i) = \mathsf{source}(\tau_{i+1})$, $\mathsf{source}(\tau_1) = p$, $\mathsf{cible}(\tau_n) = q$ et pour tout $i \in [1, n]$ nous avons $\mathsf{etiq}(\tau_i) = a_i$. Si $w = \epsilon$ alors nous avons $p = q$. Un calcul est acceptant si $p \in I$ et $q \in F$. Nous pouvons aussi définir ce calcul des deux façons suivantes :

$$p_1 \xrightarrow{a_1} p_2 \xrightarrow{a_2} \cdots \xrightarrow{a_n} p_{n+1} \text{ ou } p_1 \xrightarrow{a_1 a_2 \dots a_n} {}^* p_{n+1}$$

A noter qu'il peut y avoir plusieurs calculs correspondants à la deuxième écriture. Nous appelons $\mathsf{Cal}_w(A)$ l'ensemble des calculs de A sur w, et $\mathsf{ACal}_w(A)$ l'ensemble des calculs acceptants de A sur w.

Accessibilité Un état q de A est accessible depuis un état p de A s'il existe un mot $w \in \Sigma^*$ et un calcul $p \xrightarrow{w}{}^* q \in \mathsf{Cal}_w(A)$. Nous disons aussi que p est co-accessible depuis q. De plus l'état q est accessible s'il existe un état initial p_i tel que q est accessible depuis p_i, et co-accessible s'il existe un état final p_f tel que q est co-accessible depuis p_f.

Langage Le langage de A, que nous noterons $\mathcal{L}(A)$, est défini comme suit :

$$\mathcal{L}(A) = \{w \in \Sigma^* \mid \mathsf{ACal}_w(A) \neq \varnothing\}$$

Autrement dit, le langage de A est l'ensemble des mots w pour lesquels il existe un calcul acceptant.

Ambiguïté Le degré d'*ambiguïté* de A, dénoté par $\mathsf{amb}(A)$ est défini par :

$$\mathsf{amb}(A) = \sup\{\#\mathsf{ACal}_w(A) \mid w \in \mathcal{L}(A)\}$$

Nous disons que A est *finiment ambigu* si $\mathsf{amb}(A) < \infty$ et infiniment ambigu sinon.

Définition 1.3.2: Déterminisme

Un AF $A = (Q, I, F, \delta)$ sur un alphabet Σ est déterministe si $\#I = 1$ et pour toutes transitions $(q, a, p), (q, a, p') \in \delta$ nous avons $p = p'$.

Les automates finis déterministes sont équivalents aux automates finis non-déterministes, c'est à dire que pour tout AF A nous pouvons construire un AF A' déterministe tel que le langage de A est égal à celui de A'. Cependant la procédure de déterminisation peut produire automate fini avec un nombre d'états exponentiel dans le nombre d'états de l' AF en entrée. Par exemple, étant donnés un alphabet $\Sigma = \{a, b\}$ et un langage $L = \{uav \mid u, v \in \Sigma^* \text{ et } |v| = k-1\}$, il existe un automate fini reconnaissant L avec $k+1$ états, ce qui n'est pas le cas pour les automates finis déterministes, comme statué ci-dessous (voir [HMU03]) :

Théorème 1.3.3

Soient $\Sigma = \{a, b\}$ un alphabet et $L = \{uav \mid u, v \in \Sigma^* \text{ et } |v| = k-1\}$, alors tout automate fini déterministe reconnaissant ce langage a au moins 2^k états.

A noter qu'un automate déterministe possède une ambiguïté de 1, et la notion d'ambiguïté peut être vue comme une généralisation du déterminisme. Un automate avec une ambiguïté de 1 n'est par contre pas nécessairement déterministe.

Exemple 1.3.4

Soit $\Sigma = \{a, b\}$ un alphabet et $A = (Q, I, F, \delta)$ un AF sur Σ tel que :

- $Q = \{p, q\}$, $I = \{p\}$, $F = \{q\}$,
- $\delta = \{(p, a, p), (p, b, q), (q, a, q), (q, b, p)\}$.

Cet automate reconnaît les mots $w \in \Sigma^*$ tels que $|w|_b$ est impair. L'état est utilisé pour retenir l'information concernant la parité du nombre de b. Si nous sommes dans l'état p, nous avons rencontré jusqu'à présent un nombre pair de b, alors que dans l'état q le nombre de b rencontré est impair. Plus précisément, le langage de A est le suivant :

$$\mathcal{L}(A) = \{w \in \Sigma^* \mid |w|_b \text{ est pair } \}$$

Cet automate est aussi déterministe. Nous avons représenté A à la figure 1.3. Les cercles représentent les états, les flèches d'un cercle vers un autre représentent une transition. Un cercle doublement entouré représente un état final et un cercle avec une flèche entrante représente un état initial.

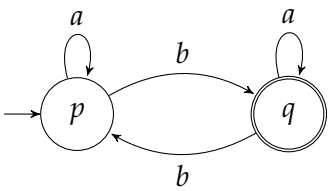


FIGURE 1.3 – $\mathcal{L}(A) = \{w \in \Sigma^* \mid |w|_b \text{ est pair } \}$

1.3.2 Automates d'arbres

Tout comme pour les automates finis, les automates d'arbres sont un modèle ancien et ont été longuement étudiés. Ce modèle a été introduit en 1965 de manière indépendante par John

Doner dans [Don65] et par James W. Thatcher dans [Tha65]. Une présentation complète des AA peut-être trouvée dans [CDG+07].

Définition 1.3.5: Automate d'arbres

Un *automate d'arbres* A sur l'alphabet gradué (Σ, ar) est un 3-uplet (Q, I, δ) avec :

- Q un ensemble fini d'états,
- $I \subseteq Q$ un ensemble d'états initiaux,
- $\delta \subseteq \bigcup_{m\in\mathbb{N}} \delta_m$ un ensemble fini de transitions avec $\delta_m \subseteq Q \times \Sigma_m \times Q^m$.

Cette définition correspond à celle des automates d'arbres *top-down*, tels qu'ils sont décrits dans [GS84]). Ce nom est dû à la façon de lire les arbres reconnus par l'automate. Dans un automate d'arbres *top-down*, nous lisons les arbres de haut en bas, de la racine aux feuilles.

Pour la suite de la section nous considérons l' AA $A = (Q, I, \delta)$ sur l'alphabet gradué (Σ, ar). Pour toute transition $\tau = (q, f, p_1, \ldots, p_m)$ de δ avec $m \in \mathbb{N}$, nous définissons les fonctions $\mathsf{racine}(\tau) = p$, $\mathsf{etiq}(\tau) = f$, $\mathsf{ar}(\tau) = m$, et $\mathsf{fils}_i(\tau) = p_i$ pour tout $i \in [1, m]$.

Calcul Notons $\mathcal{X}_n = \{x_1, \ldots, x_n\}$ un ensemble de variables avec $n \in \mathbb{N}$, q un état, $S = p_1 \ldots p_n$ une séquence d'états et $t \in \mathcal{T}_\Sigma(X_n)$ un contexte d'arbre. Nous définissons un (q, S)-calcul ρ sur le contexte t comme étant un contexte d'arbre sur $\mathcal{T}_\delta(X_n)$ tel que :

- si $t = x_i \in X_n$ avec $i \in [1, n]$ alors $q = p_i$ et $\rho = x_i$,
- si $t = f(t|_1, \ldots, t|_m)$ avec $f \in \Sigma_m$ alors $\rho = \tau(\rho_1, \ldots, \rho_n)$ avec $\tau = (q, f, q_1, \ldots, q_m)$ et ρ_i un (q_i, S)-calcul de $t|_i$ pour $i \in [1, m]$.

Nous pouvons aussi décrire ρ par $p \xrightarrow{t}{}^* S$. Un (q, ϵ)-calcul est aussi appelé un q-calcul. Un q-calcul est acceptant si $q \in I$. Nous appelons $\mathsf{Cal}_t(A)$ l'ensemble des q-calculs de A sur t, et $\mathsf{ACal}_t(A)$ l'ensemble des q-calculs acceptants de A sur t.

Accessibilité Un état q de A est accessible depuis un état p de A s'il existe un $(p, (q))$-calcul de A. Nous disons aussi que p est co-accessible depuis q. De plus l'état q est accessible s'il existe un état initial p_i tel que q est accessible depuis p_i, et co-accessible s'il existe un q-calcul de A.

Langage Le langage de A, que nous appellerons $\mathcal{L}(A)$, est l'ensemble défini comme suit :

$$\mathcal{L}(A) = \{t \in \mathcal{T}_\Sigma \mid \mathsf{ACal}_t(A) \neq \varnothing\}$$

Autrement dit, le langage de A est l'ensemble des arbres t tel qu'il existe un q-calcul acceptant sur t.

Ambiguïté Le degré d'*ambiguïté* de A, dénoté par $\mathsf{amb}(A)$ est défini par :

$$\mathsf{amb}(A) = \sup\{\#\mathsf{ACal}_t(A) \mid t \in \mathcal{L}(A)\}$$

Nous disons que A est *finiment ambigu* si $\mathsf{amb}(A) < \infty$ et infiniment ambigu si $\mathsf{amb}(A) = \infty$.

Exemple 1.3.6

Soit (Σ, ar) un alphabet gradué avec $\Sigma_2 = \{f\}$ et $\Sigma_0 = \{a, b\}$ et $A = (Q, I, \delta)$ un AA tel que :

- $Q = \{p, q\}$, $I = \{q\}$,
- $\delta = \{(p, f, p, p), (p, f, q, q), (q, f, p, q), (q, f, q, p), (q, b), (p, a)\}$.

Cet automate reconnaît les arbres t de $\mathcal{T}_\Sigma$ tels que $|t|_b$ est impair. Nous avons représenté dans la Figure 1.4 un arbre $t = f(f(a, a), f(a, b))$ et un q-calcul acceptant de A sur t.

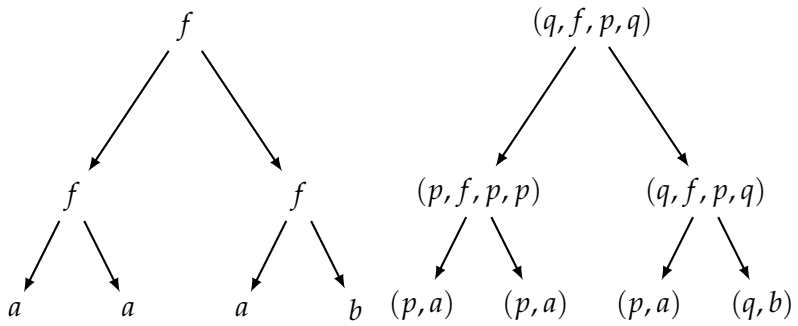


FIGURE 1.4 – Un arbre $t = f(f(a, a), f(a, b))$ et un q-calcul acceptant de A sur t.

1.4 Automates à pile visible

1.4.1 Définition

Les *automates à pile visible* (*visibly pushdown automata* ou APV) sont une restriction des automates à pile, que nous ne définirons pas ici. Une description complète de ce modèle peut être trouvée dans [HMU03]. Un APV est donc un automate fini sur un alphabet structuré auquel nous avons ajouté une pile, et chaque transition a la possibilité d'empiler ou dépiler un symbole, et ne peut être traversée que si le bon symbole à dépiler est au sommet de la pile. Elle est appelée *visible* car restreinte par le mot lu en entrée : un symbole est empilé lorsqu'un ouvrant de l'alphabet structuré est lu, un symbole est dépilé lorsqu'un fermant est lu et aucune action n'est effectuée sur la pile lors de la lecture d'un interne. Ce comportement est différent de celui des automates à piles, où un même mot peut produire des piles de hauteurs différentes, selon le chemin emprunté. Ce modèle est plus expressif que les automates finis, c'est à dire qu'il existe des langages reconnus par un APV et pas par un AF, par exemple le langage $\{a^n b^n \mid n \in \mathbb{N}\}$ (voir [HMU03]). Les APV ont été présenté pour la première fois en 1983 par Burchard von Braunmühl et Rutger Verbeek sous le nom de *Input-Driven Automata* dans [vBV83]. Ils ont été réintroduits depuis par Rajeev Alur et Parthasarathy Madhusudan en 2004 dans [AM04].

Définition 1.4.1: Automate à pile visible

Soit Δ un alphabet structuré. Un *automate à pile visible* A sur l'alphabet structuré Δ est un 5-uplet $(Q, I, F, \delta, \Gamma)$ avec :

- Q un ensemble fini d'états,
- $I \subseteq Q$ un ensemble d'états initiaux,
- $F \subseteq Q$ un ensemble d'états finaux,
- Γ un alphabet de pile et $\perp \notin \Gamma$ un symbole de pile vide,
- $\delta = \delta_c \cup \delta_r \cup \delta_r^{\perp} \cup \delta_\iota$ un ensemble fini de transitions avec :
  - $\delta_c \subseteq Q \times \Delta_c \times \Gamma \times Q$,
  - $\delta_r \subseteq Q \times \Gamma \times \Delta_r \times Q$,
  - $\delta_r^{\perp} \subseteq Q \times \{\perp\} \times \Delta_r \times Q$,
  - $\delta_\iota \subseteq Q \times \Delta_\iota \times Q$,

Un APV possède quatre ensembles de transitions δ_c, δ_r, $\delta_r^{\perp}$ et δ_ι, dépendants du type d'action à effectuer sur la pile. Intuitivement, lorsque nous franchissons une transition de δ_c, un symbole de Γ est empilé. Sur une transition de δ_r, le symbole au sommet de la pile est dépilé. Enfin, sur une transition de δ_ι, aucune action n'est effectuée sur la pile. Les transitions de $\delta_r^{\perp}$ ne peuvent être franchies que lorsque la pile est vide.

Pour la suite de la section nous considérons l'APV $A = (Q, I, F, \delta, \Gamma)$ sur l'alphabet structuré Δ. Pour toute transition $\tau = (p, a, q) \in \delta_\iota$, $\tau = (p, a, \gamma, q) \in \delta_c$, $\tau = (p, \gamma, a, q) \in \delta_r$ ou $\tau = (p, \perp, a, q) \in \delta_r^{\perp}$, nous définissons les fonctions $\mathsf{source}(\tau) = p$, $\mathsf{etiq}(\tau) = a$, $\mathsf{cible}(\tau) = q$ et $\mathsf{spile}(\tau) = \gamma$ ou $\mathsf{spile}(\tau) = \perp$ si $\tau \in \delta_\iota \cup \delta_r^{\perp}$.

Pile et configuration Nous définissons une *pile* comme un mot de Γ^*, et notons par $\perp$ la pile vide. Une *configuration* de A est une paire composée d'un état et d'une pile, c'est à dire de $Q \times \Gamma^*$. Nous appellerons *configuration initiale* une configuration de $I \times \{\perp\}$ et *configuration finale* une configuration de $F \times \Gamma^*$.

Calcul Un calcul de A sur un mot $w = a_1 \dots a_n \in \Sigma^*$ avec $n \in \mathbb{N}$, d'une configuration $(p, \sigma) \in Q \times \Gamma^*$ vers une configuration $(q, \sigma') \in Q \times \Gamma^*$ est une séquence de transitions $\rho = (\tau_1, \dots, \tau_n) \in \delta^*$ telle que pour tout $i \in [1, n-1]$ nous avons $\mathsf{cible}(\tau_i) = \mathsf{source}(\tau_{i+1})$, $\mathsf{source}(\tau_1) = p$, $\mathsf{cible}(\tau_n) = q$, et pour tout $i \in [1, n]$ nous avons $\mathsf{etiq}(\tau_i) = a_i$ et il existe $\sigma_1, \dots, \sigma_{n+1} \in \Gamma^*$ avec $\sigma_1 = \sigma$, $\sigma_{n+1} = \sigma'$ et :

- Si $a_i \in \Delta_\iota$ alors $\sigma_i = \sigma_{i+1}$,
- Si $a_i \in \Delta_c$ alors il existe $\gamma \in \Gamma$ tel que $\sigma_{i+1} = \sigma_i \gamma$,
- Si $a_i \in \Delta_r$ alors il existe $\gamma \in \Gamma$ tel que $\sigma_i = \sigma_{i+1} \gamma$,
- Si $a_i = \perp$ alors $\sigma_i = \sigma_{i+1} = \perp$.

Si $w = \epsilon$ alors $(p, \sigma) = (q, \sigma')$. Un calcul est initialisé si (p, σ) est une configuration initiale, et acceptant si initialisé et (q, σ') est une configuration finale. Nous pouvons aussi écrire un calcul des deux façons suivantes :

$$(p_1, \sigma_1) \xrightarrow{a_1} (p_2, \sigma_2) \xrightarrow{a_2} \cdots \xrightarrow{a_n} (p_{n+1}, \sigma_{n+1}) \text{ ou } (p_1, \sigma_1) \xrightarrow{a_1 a_2 \dots a_n}{}^{*} (p_{n+1}, \sigma_{n+1})$$

avec $\mathsf{etiq}(\tau_i) = a_i$, $\mathsf{source}(\tau_i) = p_i$ et $\mathsf{cible}(\tau_i) = p_{i+1}$ pour tout $i \in [1, n]$. Nous appelons $\mathsf{Cal}_w(A)$ l'ensemble des calculs de A sur w, et $\mathsf{ACal}_w(A)$ l'ensemble des calculs acceptants de A sur w. Soit $\rho : (p, \sigma) \xrightarrow{u}{}^*(q, \sigma')$ un calcul de A. Nous appelons $\mathsf{premier}(\rho)$ la configuration (p, σ) et $\mathsf{dernier}(\rho)$ la configuration (q, σ').

Concaténation de calculs Soit deux calculs ρ, ρ' de A définis comme suit :

$$\rho : (p_1, \sigma_1) \xrightarrow{a_1} \dots \xrightarrow{a_m} (p_{m+1}, \sigma_{m+1})$$
$$\rho' : (p'_1, \sigma'_1) \xrightarrow{a'_1} \dots \xrightarrow{a'_m} (p'_{m+1}, \sigma'_{m'+1})$$

tels que $(p_{m+1}, \sigma_{m+1}) = (p'_1, \sigma'_1)$. La concaténation de ρ et ρ', dénotée par $\rho\rho'$ est le calcul $(p_1, \sigma_1) \xrightarrow{a_1} \dots \xrightarrow{a_m} (p_{m+1}, \sigma_{m+1}) \xrightarrow{a'_1} \dots \xrightarrow{a'_m} (p'_{m+1}, \sigma'_{m'+1})$. Pour tout $i \in \mathbb{N}$, nous appelons ρ^i le calcul $\Pi_{j=1}^{i}(\rho)$.

Configuration acceptante Une configuration κ de A est *acceptante* si elle est finale et il existe un calcul acceptant menant à elle. Les configurations acceptantes sont les configurations finales atteignables depuis une configuration initiale. Il est important de distinguer ces configurations finales des autres car contrairement à un AF (où état égal configuration), toutes les piles ne sont pas atteignables.

Accessibilité Une configuration κ de A est accessible depuis une configuration κ' s'il existe un calcul $\kappa' \xrightarrow{u}{}^* \kappa \in \mathsf{Cal}_u(A)$. Nous disons aussi que κ' est co-accessible depuis κ. De plus la configuration κ est accessible s'il existe une configuration initiale κ_i telle que κ est accessible depuis κ_i, et co-accessible s'il existe une configuration **acceptante** p_f telle que κ est co-accessible depuis κ_f.

Langage Le langage de A, que nous appellerons $\mathcal{L}(A)$, est l'ensemble défini comme suit :

$$\mathcal{L}(A) = \{w \in \Sigma^* \mid \mathsf{ACal}_w(A) \neq \varnothing\}$$

Autrement dit le langage de A est l'ensemble des mots w pour lesquels il existe un calcul acceptant.

Langage bien imbriqué Le langage de A est *bien imbriqué* si pour tout mot $w \in \mathcal{L}(A)$ nous avons $w \in \Sigma^*_{\mathsf{bi}}$. Nous disons alors que A est *bien imbriqué*. Nous appelons APVbi l'ensemble des APV bien imbriqués. A noter qu'un APVbi n'utilise pas de transition de $\delta_r^{\perp}$, car cela implique qu'il peut accepter des mots avec des fermants sans correspondance.

Remarque 1.4.2

Si A est bien imbriqué alors ses configurations finales (f, σ) sont accessibles seulement si $\sigma = \perp$.

Langage linéaire Le langage de A est *linéaire* si pour tout mot $w \in \mathcal{L}(A)$ nous avons $w \in \Sigma^*_{\mathsf{lin}}$. Nous disons alors que A est *linéaire*. Nous appelons APVlin l'ensemble des APV linéaires.

Ambiguïté Le degré d'*ambiguïté* de A, dénoté par $\mathsf{amb}(A)$ est défini par :

$$\mathsf{amb}(A) = \sup\{\#\mathsf{ACal}_w(A) \mid w \in \mathcal{L}(A)\}$$

Nous disons que A est *finiment ambigu* si $\mathsf{amb}(A) < \infty$ et infiniment ambigu sinon.

Définition 1.4.3: Déterminisme

Un APV $A = (Q, I, F, \delta, \Gamma)$ sur un alphabet structuré Δ est déterministe si $\#I = 1$ et :

- Pour toutes transitions $(q, c, \gamma, p), (q, c, \gamma', p') \in \delta_c$ nous avons $p = p'$ et $\gamma = \gamma'$,
- Pour toutes transitions $(q, \gamma, r, p), (q, \gamma, r, p') \in \delta_r \cup \delta_r^{\perp}$ nous avons $p = p'$.
- Pour toutes transitions $(q, a, p), (q, a, p') \in \delta_\iota$ nous avons $p = p'$,

Comme pour les automates finis, les APV généraux sont équivalents aux APV déterministes (voir [AM04]) mais sont exponentiellement plus concis. A noter que pour les transitions de dépilement nous n'avons pas de condition sur le symbole de pile, car celui-ci est fixé par la configuration du calcul. Ceci nous assure qu'il n'y a qu'un seul calcul par mot accepté par l'automate à pile visible. Nous montrerons comment déterminer un automate à pile visible dans la section suivante.

Exemple 1.4.4

Soit $\Delta = \Delta_c \cup \Delta_r \cup \Delta_\iota$ un alphabet structuré tel que $\Sigma_c = \{a\}$, $\Sigma_r = \{b\}$ et $\Sigma_\iota = \varnothing$ et $A = (Q, I, F, \delta, \Gamma)$ un APV tel que :

- $Q = \{p_i, p_1, p_2, p_f\}$, $I = \{p_i\}$, $F = \{p_i, p_f\}$, $\Gamma = \{\gamma_1, \gamma_2\}$,
- $\delta = \delta_\iota \cup \delta_c \cup \delta_r \cup \delta_r^{\perp}$ avec :
  - $\delta_\iota = \varnothing$,
  - $\delta_c = \{(p_i, a, \gamma_1, p_1), (p_1, a, \gamma_2, p_1), (p_1, a, \gamma_2, p_2)\}$,
  - $\delta_r = \{(p_2, \gamma_2, b, p_2), (p_2, \gamma_1, b, p_f), (p_1, \gamma_1, b, p_f)\}$,
  - $\delta_r^{\perp} = \varnothing$.

Le langage reconnu par A est l'ensemble des suites de a puis de b tel que leur nombre est le même. Plus précisément nous avons :

$$\mathcal{L}(A) = \{a^n b^n \mid n \in \mathbb{N}\}$$

A est bien imbriqué et linéaire. De la même manière que pour les AF, un APV peut être représenté de manière graphique. L'éventuel symbole de pile est écrit à droite de la lettre sur la transition, l'expression $(a \mid \gamma)$ symbolisant un empilement de γ et l'expression $(a \mid \gamma^{-1})$ un dépilement, comme vu à la Figure 1.5.

1.4.2 Propriétés des automates à pile visible

Nous allons maintenant présenter plusieurs propriétés importantes des automates à pile visible. Nous allons étudier plusieurs résultats liés à la pile de départ d'un calcul dans la première partie. La deuxième partie établira une propriété particulière des automates à pile visible

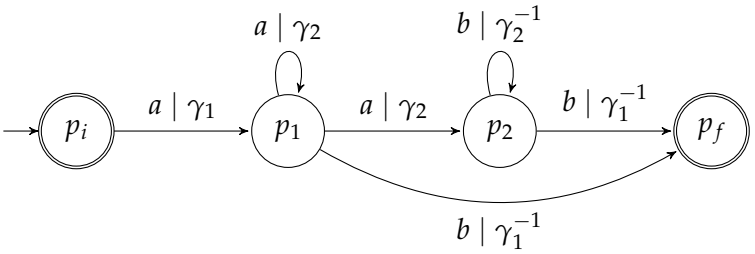


FIGURE 1.5 – $\mathcal{L}(A) = \{a^n b^n \mid n \in \mathbb{N}\}$

reconnaissant des langages bien imbriqués. Dans la troisième partie nous introduirons les langages de pile. Enfin, nous parlerons de déterminisation dans la dernière partie.

Modification de la pile Nous allons introduire deux Lemmes. Le premier statue qu'il est possible d'ajouter des symboles au fond de toutes les piles des configurations d'un calcul sans altérer les transitions empruntées par celui-ci. Le deuxième lemme quant à lui permet de complètement supprimer la pile de la première configuration d'un calcul sur un mot bien imbriqué, et ceci sans altérer les transitions empruntées par celui-ci. Ceci est possible car ces symboles ne seront pas touchés par le calcul.

Lemme 1.4.5

Soit A un APV sur un alphabet structuré Δ et $(p_0, \sigma_0) \xrightarrow{a_1} (p_1, \sigma_1) \cdots \xrightarrow{a_n} (p_n, \sigma_n)$ un calcul de A sur un mot $w \in \Delta^*$ avec aucun fermant sans correspondance. Pour tout $\sigma \in \Gamma^*$, $(p_0, \sigma\sigma_0) \xrightarrow{a_1} (p_1, \sigma\sigma_1) \cdots \xrightarrow{a_n} (p_n, \sigma\sigma_n)$ est un calcul de A sur w.

Démonstration. Par induction sur la longueur de w. □

Exemple 1.4.6

Nous considérons l'APV A de l'Exemple 1.4.4. Le calcul suivant :

$$\rho = (p_1, \gamma_2) \xrightarrow{a} (p_2, \gamma_2\gamma_2) \xrightarrow{b} (p_2, \gamma_2) \xrightarrow{b} (p_2, \bot)$$

est un calcul de A sur le mot abb. En appliquant le Lemme 1.4.5 nous obtenons que :

$$\rho' = (p_1, \gamma_1\gamma_2) \xrightarrow{a} (p_2, \gamma_1\gamma_2\gamma_2) \xrightarrow{b} (p_2, \gamma_1\gamma_2) \xrightarrow{b} (p_2, \gamma_1)$$

est aussi un calcul de A sur aab, et emprunte les mêmes transitions que ρ. Notez que cela permet de compléter ρ en un calcul acceptant $\rho_f = (p_i, \bot) \xrightarrow{a} (p_1, \gamma_1) \xrightarrow{a} \rho' \xrightarrow{b} (p_f, \bot)$. C'est un mécanisme que nous utiliserons souvent par la suite.

Lemme 1.4.7

Soit A un APV sur un alphabet structuré Δ et $(p_0, \sigma_0) \xrightarrow{a_1} (p_1, \sigma_1) \cdots \xrightarrow{a_n} (p_n, \sigma_n)$ un calcul de A sur un mot bien imbriqué $w \in \Delta^*_{\mathsf{bi}}$. $(p_0, \sigma'_0) \xrightarrow{a_1} (p_1, \sigma'_1) \cdots \xrightarrow{a_n} (p_n, \sigma'_n)$ est un calcul de A sur w avec $\sigma_i = \sigma_0 \sigma'_i$ pour tout $i \in [1, n]$.

Démonstration. Nous notons $A = (Q, I, F, \Gamma, \delta)$. Nous prouvons ce lemme par induction sur la structure de w. Le cas où $w = \epsilon$ étant trivial, nous considérons les autres possibilités :

- Si $a_1 \in \Delta_\iota$ alors nous avons $(p_0, a_1, p_1) \in \delta_\iota$ et par hypothèse d'induction nous déduisons que $(p_1, \sigma'_1) \xrightarrow{a_2} (p_2, \sigma'_2) \cdots \xrightarrow{a_n} (p_n, \sigma'_n) \in \mathsf{Cal}_{a_2 \dots a_n}(A)$ avec $\sigma_i = \sigma_1 \sigma'_i$ pour tout $i \in [1, n]$. Comme $\sigma_0 = \sigma_1$ nous pouvons conclure que $(p_0, \sigma'_0) \xrightarrow{a_1} (p_1, \sigma'_1) \cdots \xrightarrow{a_n} (p_n, \sigma'_n) \in \mathsf{Cal}_w(A)$ avec $\sigma_i = \sigma_0 \sigma'_i$ pour tout $i \in [1, n]$.
- Si $a_1 \in \Delta_c$ alors il existe $j \in [1, n]$ tel que $a_j \in \Delta_r$ et $a_2 \dots a_{j-1}, a_{j+1} a_n \in \Delta^*_{\mathsf{bi}}$ alors il existe $\gamma \in \Gamma$ tel que $(p_0, a_1, \gamma, p_1) \in \delta_c$ et $(p_{j-1}, \gamma, a_j, p_j) \in \delta_r$ et par hypothèse d'induction nous déduisons que :

$$(p_1, \sigma''_1) \xrightarrow{a_2} (p_2, \sigma''_2) \cdots \xrightarrow{a_{j-1}} (p_{j-1}, \sigma''_{j-1}) \in \mathsf{Cal}_{a_2 \dots a_{j-1}}(A)$$
$$(p_j, \sigma'''_j) \xrightarrow{a_{j+1}} (p_{j+1}, \sigma'''_{j+1}) \cdots \xrightarrow{a_n} (p_n, \sigma'''_n) \in \mathsf{Cal}_{a_{j+1} \dots a_n}(A)$$

  avec $\sigma_i = \sigma_1 \sigma''_i$ pour tout $i \in [1, j-1]$ et $\sigma_i = \sigma_j \sigma'''_i$ pour tout $i \in [j+1, n]$. Nous pouvons donc conclure que $(p_0, \sigma'_0) \xrightarrow{a_1} (p_1, \sigma'_1) \cdots \xrightarrow{a_n} (p_n, \sigma'_n) \in \mathsf{Cal}_w(A)$ avec $\sigma_i = \sigma_0 \sigma'_i$ pour tout $i \in [j+1, n]$, $\sigma'_i = \sigma''_i \gamma$ pour tout $i \in [1, j-1]$ et $\sigma'_i = \sigma'''_i$ pour tout $i \in [1, n]$. □

Exemple 1.4.8

Nous considérons l'APV A de l'Exemple 1.4.4. Le calcul suivant :

$$\rho = (p_1, \gamma_1) \xrightarrow{a} (p_2, \gamma_1 \gamma_2) \xrightarrow{b} (p_2, \gamma_1)$$

est un calcul de A sur le mot ab. En appliquant le Lemme 1.4.7 nous obtenons que :

$$\rho = (p_1, \bot) \xrightarrow{a} (p_2, \gamma_2) \xrightarrow{b} (p_2, \bot)$$

est aussi un calcul de A sur aab, et emprunte les mêmes transitions que ρ'.

Calculs sur des mots bien imbriqués Nous allons construire un ensemble listant toutes les paires d'états séparant des mots bien imbriquées. Nous allons ensuite montrer que cet ensemble est constructible en temps polynomial dans le nombre d'états de l'automate à pile visible.

Définition 1.4.9

Soit $A = (Q, I, F, \delta, \Gamma)$ un APV sur l'alphabet structuré Δ. Nous appelons BI_A le sous-ensemble de $Q \times Q$ tel que pour tout $p, q \in Q$ nous avons $(p, q) \in \mathsf{BI}_A$ si et seulement s'il existe $w \in \Delta^*_{\mathsf{bi}}$ tel que $(p, \bot) \xrightarrow{w}{}^* (q, \bot) \in \mathsf{Cal}_w(A)$.

Lemme 1.4.10

Soit $A = (Q, I, F, \delta, \Gamma)$ un APV sur un alphabet structuré Δ. L'ensemble BI_A peut être construit en temps polynomial dans la taille de Q.

Démonstration. Nous construisons BI_A de la façon suivante :

- Pour tout $p \in Q$ nous avons $(p, p) \in \mathsf{BI}_A$,
- Pour tout $(p, a, p') \in \delta_\iota$ et $(p', q) \in \mathsf{BI}_A$ nous avons $(p, q) \in \mathsf{BI}_A$,
- Pour tout $(p, c, \gamma, p') \in \delta_c$, $(q', \gamma, r, s) \in \delta_r$ et $(p', q'), (s, q) \in \mathsf{BI}_A$ nous avons $(p, q) \in \mathsf{BI}_A$.

Nous considérons en premier l'implication $\Rightarrow$.

- Si $w = \epsilon$, alors $p = q$. Par construction de BI_A nous avons bien $(p, p) \in \mathsf{BI}_A$.
- Si $w = au$ avec $a \in \Delta_\iota$ et $u \in \Delta^*_{\mathsf{bi}}$, alors il existe $p' \in Q$ tel que $(p, a, p') \in \delta_\iota$, $(p', \sigma) \xrightarrow{u}{}^*(q, \sigma) \in \mathsf{Cal}_u(A)$. Par le Lemme 1.4.7 et par hypothèse d'induction, nous déduisons que $(p', q) \in \mathsf{BI}_A$. Par construction de BI_A nous pouvons donc conclure que $(p, q) \in \mathsf{BI}_A$.
- Si $w = curv$ avec $c \in \Delta_c$, $r \in \Delta_r$ et $u, v \in \Delta^*_{\mathsf{bi}}$, alors il existe $p', q' \in Q$, $\gamma \in \Gamma$ tels que : $(p, c, \gamma, p') \in \delta_c$, $(p', \sigma\gamma) \xrightarrow{u}{}^*(q', \sigma\gamma) \in \mathsf{Cal}_u(A)$, $(q', \gamma, r, s) \in \delta_r$ et $(s, \sigma) \xrightarrow{v}{}^*(q, \sigma) \in \mathsf{Cal}_v(A)$. Par le Lemme 1.4.7 et par hypothèse d'induction, nous déduisons que $(p', q'), (s, q) \in \mathsf{BI}_A$. Par construction de BI_A nous pouvons donc conclure que $(p, q) \in \mathsf{BI}_A$.

Nous montrons maintenant le sens $\Leftarrow$.

- Si $(p, q) \in \mathsf{BI}_A$ car $p = q$, alors nous avons $(p, \bot) \in \mathsf{Cal}_\epsilon(A)$.
- Si $(p, q) \in \mathsf{BI}_A$ car il existe $(p, a, p') \in \delta_\iota$ et $(p', q) \in \mathsf{BI}_A$, alors par hypothèse d'induction il existe $w \in \Delta^*_{\mathsf{bi}}$ tel que $(p', \bot) \xrightarrow{w}{}^*(q, \bot) \in \mathsf{Cal}_w(A)$. Nous pouvons donc en conclure que $(p, \bot) \xrightarrow{a} (p', \bot) \xrightarrow{w}{}^*(q, \bot) \in \mathsf{Cal}_{aw}(A)$.
- Si $(p, q) \in \mathsf{BI}_A$ car il existe des mots $(p, c, \gamma, p') \in \delta_c$, $(q', \gamma, r, s) \in \delta_r$ et $(p', q'), (s, q) \in \mathsf{BI}_A$, alors par hypothèse d'induction il existe $u, v \in \Delta^*_{\mathsf{bi}}$ tels que $(p', \bot) \xrightarrow{u}{}^*(q', \bot) \in \mathsf{Cal}_u(A)$ et $(s, \bot) \xrightarrow{v}{}^*(q, \bot) \in \mathsf{Cal}_u(A)$. Par le Lemme 1.4.5 nous pouvons donc en conclure que $(p, \bot) \xrightarrow{c} (p', \gamma) \xrightarrow{u}{}^*(q', \gamma) \xrightarrow{r} (s, \bot) \xrightarrow{v}{}^*(q, \bot) \in \mathsf{Cal}_{curv}(A)$ □

Exemple 1.4.11

Nous considérons l'APV A de l'Exemple 1.4.4. De base nous avons les couples suivants $(p_i, p_i), (p_1, p_1), (p_2, p_2), (p_f, p_f)$ dans BI_A. Comme (p_i, a, γ_1, p_1), $(p_1, a, \gamma_2, p_2) \in \delta_c$ et $(p_1, \gamma_1, b, p_f), (p_2, \gamma_2, b, p_2) \in \delta_r$ nous avons aussi $(p_i, p_f), (p_1, p_2) \in \mathsf{BI}_A$. Nous pouvons conclure que $\mathsf{BI}_A = \{(p_i, p_i), (p_1, p_1), (p_2, p_2), (p_f, p_f), (p_i, p_f), (p_1, p_2)\}$.

Langage de pile Dans cette dernière partie nous allons introduire les langages de pile. Cette notion a été originalement introduite dans le cadre de l'étude des automates à pile, nous allons la présenter pour les automates à pile visible.

Définition 1.4.12

Étant donné un APV $A = (Q, I, F, \Gamma, \delta)$ et un état q de Q, nous appelons pile_A^q l'ensemble des mots de pile σ tels que (q, σ) est accessible et co-accessible. Cet ensemble est le *langage de pile* de A à l'état q.

Cet ensemble de mots de pile est un langage régulier sur l'alphabet Γ, c'est à dire qu'il existe un automate fini le reconnaissant. Plus formellement nous avons :

Proposition 1.4.13

Soit $A = (Q, I, F, \Gamma, \delta)$ un APV. Pour tout état $q \in Q$ il existe un AF acc_A^q tel que $\mathcal{L}(\mathsf{acc}_A^q) = \mathsf{pile}_A^q$ et peut être construit en temps polynomial.

Ce résultat est bien connu pour les automates à pile et a été prouvé de nombreuses fois, notamment dans [RS97]. Nous utiliserons les langages de pile pour décider si un état est utile ou non. En effet, si pile_A^q est vide alors il n'existe aucun calcul acceptant passant par q, et cet état peut alors être supprimé.

Exemple 1.4.14

Nous considérons l'APV A de l'Exemple 1.4.4. L'automate fini $\mathsf{acc}_A^{p_2}$ est représenté à la Figure 1.6.

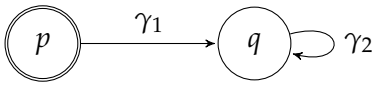


FIGURE 1.6 – L'automate $\mathsf{acc}_A^{p_2}$

Déterminisation Nous allons maintenant montrer comment construire un automate à pile visible déterministe équivalent à un APV donné. La procédure repose sur la classique construction par sous-ensembles ([HMU03]), avec une composante supplémentaire. Chaque état de l'automate à pile visible déterminisé stocke un ensemble de paires d'états de l'automate d'entrée. Le deuxième état de chaque paire a la même utilité que dans la construction par sous-ensemble. Le premier état quant à lui permet de se souvenir où était le calcul quand le symbole au sommet de la pile a été empilé. De cette manière, lorsqu'il sera dépilé, l'automate sera capable de vérifier quelles sont les transitions de dépilement valides. En effet, afin de déterminiser un automate à pile visible, il est parfois nécessaire de fusionner plusieurs transitions d'empilement en une seule, et celles-ci peuvent empiler des symboles différents, et ne couvrant pas nécessairement l'ensemble des symboles de piles possibles. Cette information supplémentaire permet de traiter ce cas particulier. De plus, la pile est utilisée pour stocker l'ensemble de paires courant lorsqu'un symbole est empilé, afin de pouvoir le retrouver lorsque le fermant correspondant sera rencontré.

Nous présentons maintenant la construction.

Définition 1.4.15

Soit $A = (Q, I, F, \delta, \Gamma)$ un APV sur un alphabet structuré Δ. Nous appelons $\det(A)$ le APV $A' = (Q', I', F', \delta', \Gamma')$ sur Δ défini comme suit :

$\circ\ Q' = 2^{Q \times Q} \qquad I' = 2^{I \times I} \qquad F' = \{S \in Q' \mid \exists (p,q) \in S \text{ et } q \in F\} \qquad \Gamma' = Q' \times \Delta_c$

$\circ\ \delta' = \delta'_c \cup \delta'_r \cup \delta'^{\perp}_r \cup \delta'_\iota$ avec :

$$- \delta'_c = \left\{ (S, c, (S, c), S') \;\middle|\; \begin{array}{l} S \in Q', c \in \Delta_c, \\ S' = \{(q', q') \mid \exists (p, q) \in S, (q, c, \gamma, q') \in \delta_c\} \end{array} \right\}$$

$$- \delta'_r = \left\{ (S, (S'', c), r, S') \;\middle|\; \begin{array}{l} S, S'' \in Q', c \in \Delta_c, r \in \Delta_r, \\ S' = \left\{ (p, q') \middle| \begin{array}{l} \exists (p'', q'') \in S, (p, p') \in S'', \\ (p', c, \gamma, p'') \in \delta_c, (q'', \gamma, r, q') \in \delta_r \end{array} \right\} \end{array} \right\}$$

$$- \delta'^{\perp}_r = \left\{ (S, r, \perp, S') \;\middle|\; \begin{array}{l} S \in Q', r \in \Delta_r, \\ S' = \{(q, q') \mid \exists (q, p) \in S, (p, \perp, r, q') \in \delta_r\} \end{array} \right\}$$

$$- \delta_\iota = \left\{ (S, a, S') \;\middle|\; \begin{array}{l} S \in Q', a \in \Delta_\iota, \\ S' = \{(q, q') \mid \exists (q, p) \in S, (p, a, q') \in \delta_\iota\} \end{array} \right\}$$

À noter qu'il s'agit ici de la construction présentée dans [Ser11]. Il ne s'agit pas de la construction originale, présentée dans [AM04]. Nous avons choisi de présenter celle-là car elle est plus succincte, l'autre utilisant des triplets d'état et pas simplement des paires. La partie difficile de la construction concerne les transitions fermantes. En effet, le premier état d'une paire correspond à l'endroit où le calcul pouvait être quand le symbole au sommet de la pile a été empilé. Comme nous traitons les transitions fermantes, ce symbole vient d'être dépilé, et le premier état de chaque paire doit maintenant faire référence au symbole au-dessous. Pour cela nous pouvons utiliser les informations mémorisées dans le symbole dépilé. La situation est résumée à la Figure 1.7.

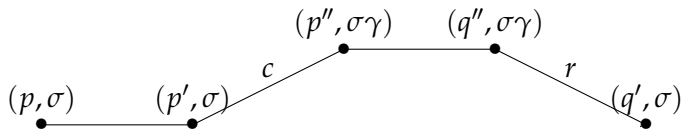


FIGURE 1.7 – Construction des transitions de dépilement de $\det(A)$.

Nous n'allons pas montrer la correction de cette construction, pour cela il convient de se référer à [AM04] et [Ser11]. Nous présentons cependant ses propriétés principales dans le théorème et les deux lemmes suivants.

Théorème 1.4.16: [AM04]

Soit A un APV sur l'alphabet structuré Δ. $\mathsf{determiniser}(A)$ est un APV déterministe et $L(A) = L(\mathsf{determiniser}(A))$.

Lemme 1.4.17: [AM04]

Soient A un APV sur l'alphabet structuré Δ, $w \in \Delta^*$ un mot et w' le plus long suffixe bien imbriqué de w. L'unique calcul de $\mathsf{det}(A)$ sur le mot w atteint l'état S défini par :

$$S = \{(p,q) \in Q \times Q \mid \exists\, (i,\bot) \xrightarrow{w} (p'',\sigma) \xrightarrow{w'} (q'',\sigma) \in \mathsf{Cal}(A) \text{ avec } i \text{ un état initial}\}$$

Lemme 1.4.18: [AM04]

Soient A un APV sur l'alphabet structuré Δ et $w \in \Delta^*$ un mot. S'il existe un calcul initialisé ρ' de $\mathsf{det}(A)$ sur w, alors il existe un calcul initialisé ρ de A sur w.

Le premier lemme est une propriété des transitions de l'automate déterminisé, et fait directement référence à la Figure 1.7. Le deuxième lemme quand à lui est une propriété de langages utilisée pour montrer que l'APV original et l'APV déterminisé ont le même langage.

Cette construction peut produire beaucoup d'états inutiles, c'est à dire des états qu'aucun calcul acceptant ne traverse. Il serait possible de l'optimiser en ajoutant les ensembles de paires d'états au fur et à mesure, en partant de ceux présents. Mais cela ne serait pas suffisant, car cela implique seulement que pour tout état S il existe une configuration (S,σ) accessible. Ces configurations peuvent ne pas être co-accessible, et donc ne pas faire partie d'un calcul acceptant.

Nous allons utiliser les langages de piles définis dans cette section. Pour chaque état S de $\mathsf{det}(A)$ nous calculons le langage de pile de S et nous testons sa vacuité, qui est une propriété décidable en temps polynomial ([AM04]). Si le langage est vide, cela signifie qu'il n'existe pas de configuration (S,σ) accessible et co-accessible, et nous pouvons donc supprimer S. Par le Lemme 1.4.13 concernant les langages de pile nous obtenons ainsi une procédure exécutable en temps polynomial pour supprimer tous les états inutiles de $\mathsf{det}(A)$.

Pour plus de simplicité, nous appelons $\mathsf{determiniser}(A)$ le résultat de la suppression des états inutiles appliquée à $\mathsf{det}(A)$. À noter que les Lemmes 1.4.17 et 1.4.18 sont toujours corrects car nous n'avons pas ajouté de nouveaux calculs à $\mathsf{determiniser}(A)$.

Chapitre 2

Coût et transduction

Dans ce chapitre, nous allons nous intéresser à des extensions des modèles présentés dans le chapitre précédent. Le premier modèle est celui des automates à coût, c'est à dire un automate muni d'un semi-anneau avec lequel il opère. Chaque transition est équipée d'une valeur du semi-anneau, et il est ensuite possible de calculer le coût d'un calcul et d'une entrée en utilisant les opérateurs du semi-anneau. Un automate à coût associe donc à une entrée un élément du semi-anneau. Nous considérerons principalement le semi-anneau des entiers naturels dans ce manuscrit, c'est à dire le semi-anneau sur les entiers naturels avec les opérateurs d'addition et de multiplication.

Le deuxième modèle considéré est celui des transducteurs. Un transducteur est un automate où chaque transition est équipée d'un mot ou d'un contexte d'arbre de sortie. Il est alors possible de calculer le mot ou l'arbre de sortie d'un calcul en concaténant les mots des transitions. Là où un automate reconnaît un langage, nous disons qu'un transducteur reconnaît une transformation.

Nous allons aussi montrer comment construire un automate d'arbres depuis un automate à pile visible et inversement, puis nous étendrons ces constructions aux automates à coûts et aux transducteurs. Nous utiliserons ces résultats plus tard pour appliquer des résultats des automates d'arbres aux automates à pile visible, et inversement.

Plan du chapitre Nous allons définir dans la Section 2.1 les automates à coûts et les transducteurs dans la Section 2.2. Nous comparerons les différents modèles dans la Section 2.3.

2.1 Automates à coûts

Dans cette section nous présentons les notions de coûts sur des automates. Un automate à coûts est un automate muni d'un semi-anneau $\mathcal{S} = (E, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ auquel nous aurions ajouté un élément de E sur chaque transition (son coût). Chaque mot (ou arbre) accepté par l'automate produit donc un coût, définissant ainsi une série formelle.

Les *automates finis à coûts* sur un semi-anneau $\mathcal{S}$ ont été mis en relation avec les séries formelles pour la première fois en 1961 par Marcel-Paul Schützenberger dans [Sch61] et développées par Samuel Eilenberg ([Eil76]), Arto Salomaa et Matti Soittola ([SS78]). Une présentation complète peut-être trouvée dans [Sak09] et dans [BR88].

Les automates d'arbres ont été étendus aux *automates d'arbres à coûts* sur un semi-anneau $\mathcal{S}$ pour la première fois en 1982 par Jean Berstel et Christophe Reutenauer dans [BR82], où les séries formelles sur des arbres sont introduites. Il existe un modèle d'automate d'arbres à coûts un peu différent, basés sur l'utilisation de polynômes, et présenté par Hemult Seidl

dans [Sei94b] et repris par Björn Borchardt, Zoltán Fülöp, Zsolt Gazdag et Andreas Maletti en 2005 dans [BFGM05]. Dans ce manuscrit nous nous baserons sur les travaux originaux de Jean Berstel et Christophe Reutenauer, bien que nous présenterons aussi certains résultats prouvés sur le modèle de Helmut Seidl.

Enfin, les *automates à pile visible à coûts* sur un semi-anneau $\mathcal{S}$ sont définis de manière similaire aux aux automates finis à coûts. Comme pour les automates finis, la différence principale reste la pile. Il existe de nombreux travaux sur les automates à pile à coût, ceux-ci ayant été introduits pour la première fois en 1961 par Marcel-Paul Schützenberger dans [Sch61].

Définition 2.1.1

Soit $\mathcal{S} = (E, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ un semi-anneau. Un AF à coûts sur $\mathcal{S}$ ou $\mathcal{S}$-AF sur l'alphabet Σ (resp. un APV à coûts sur $\mathcal{S}$ ou $\mathcal{S}$-APV sur l'alphabet Δ) est une paire de (A, λ) avec $A = (Q, I, F, \delta)$ un AF (resp. $A = (Q, I, F, \delta, \Gamma)$ un APV) et $\lambda : \delta \to E\backslash\{\mathbf{0}\}$ une fonction de coût.

Définition 2.1.2

Soit $\mathcal{S} = (E, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ un semi-anneau. Un AA à coûts sur $\mathcal{S}$ ou $\mathcal{S}$-AA sur l'alphabet gradué (Σ, ar) est une paire de (A, λ) avec $A = (Q, I, \delta)$ un AA et $\lambda : \delta \to E\backslash\{\mathbf{0}\}$ une fonction de coût.

Pour la suite de cette section nous considérons un automate à coûts $S = (A, \lambda)$ sur le semi-anneau $\mathcal{S} = (E, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ où A peut-être un AF, un APV ou un AA indistinctement .

Coût d'un calcul Afin d'expliciter le coût d'un calcul devons distinguer le type de A.

- Si A est un AF (resp. un APV) alors nous définissons le coût $\mathsf{cout}(\rho)$ d'un calcul $\rho = (\tau_1, \tau_2, \ldots, \tau_n) \in \delta^*$ d'un état ϕ vers un état ϕ (resp. d'une configuration ϕ vers une configuration ϕ) avec $n \in \mathbb{N}$ comme le produit des coûts des transitions de ρ :

  $$\mathsf{cout}(\rho) = \mathbf{1} \otimes \lambda(\tau_1) \otimes \ldots \otimes \lambda(\tau_n)$$

  De plus nous pouvons écrire ρ par :

  $$\rho = \phi_1 \xrightarrow{a_1, \lambda(\tau_1)} \phi_2 \xrightarrow{a_2, \lambda(\tau_2)} \ldots \xrightarrow{a_{n-1}, \lambda(\tau_{n-1})} \phi_n \xrightarrow{a_n, \lambda(\tau_n)} \phi_{n+1}$$
  $$\rho : \phi_1 \xrightarrow{w, \lambda(\tau_1) \otimes \cdots \otimes \lambda(\tau_n)} {}^* \phi_{n+1}$$

  où la séquence de transition forme un calcul de A.

- Si A est un AA alors nous définissons le coût $\mathsf{cout}(\rho)$ d'un (q, R)-calcul $\rho \in \mathcal{T}_\delta(X)$ avec $\mathcal{X}$ un ensemble de variables, q un état et R une séquence d'états comme le produit des coûts des transitions de ρ :

  $$\mathsf{cout}(\rho) = \begin{cases} \lambda(\tau) \otimes \mathsf{cout}(\rho|_1) \otimes \cdots \otimes \mathsf{cout}(\rho|_m) & \text{si } \rho = \tau(\rho|_1, \ldots, \rho|_m) \text{ avec } m \in \mathbb{N}, \\ \mathbf{1} & \text{si } \rho = x \text{ avec } x \in \mathcal{X}. \end{cases}$$

  Nous pouvons aussi décrire ρ par $q \xrightarrow{t, \mathsf{cout}(\rho)} {}^* R$.

Coût d'un automate à coûts Le coût $\mathsf{cout}(s)$ de $s \in \mathcal{L}(A)$ est défini comme la somme des coûts des calculs acceptants sur s :

$$\mathsf{cout}(s) = \oplus_{\rho \in \mathsf{ACal}_s(S)}(\mathsf{cout}(\rho))$$

Enfin, le coût de S, dénoté par $\mathsf{cout}(S)$, est défini par :

$$\mathsf{cout}(S) = \sup\{\mathsf{cout}(w) \mid w \in \mathcal{L}(A)\}$$

Pour tout entier $k \in \mathbb{N}$, si $\mathsf{cout}(S) < k$, alors le coût de S est k-borné. S'il existe un entier $k \in \mathbb{N}$ tel que $\mathsf{cout}(S) < k$ alors le coût de S est *fini*, sinon le coût de S est *infini*.

Fonction La *fonction* définie par S et dénotée par $\mathsf{fonction}(S)$ est l'ensemble des paires de $(\mathcal{L}(A) \times E)$ définie par :

$$\mathsf{fonction}(S) = \{(s, e) \in (\mathcal{L}(A) \times E) \mid s \in \mathcal{L}(A), e \in \mathsf{cout}(s) \text{ et } e \neq \mathbf{0}\}$$

Autrement dit, la fonction de S est l'ensemble des paires de mots ou arbres reconnus par A et des coûts produits par ces mots avec un coût différent de $\mathbf{0}$.

Exemple 2.1.3

Nous considérons le semi-anneau $\mathcal{S} = (\mathbb{N}, +, \times, 1, 0)$ avec $+$ la somme et $\times$ le produit sur les entiers. Nous avons représenté à la Figure 2.1 un $\mathcal{S}$-AF sur l'alphabet $\Sigma = \{a\}$. Il existe trois calculs acceptants sur le mot *aaa* :

$$p \xrightarrow{a,1} p \xrightarrow{a,1} p \xrightarrow{a,2} q$$
$$p \xrightarrow{a,1} p \xrightarrow{a,2} q \xrightarrow{a,3} q$$
$$p \xrightarrow{a,2} q \xrightarrow{a,3} q \xrightarrow{a,3} q$$

Ces trois calculs ont pour coût respectif $2, 6$ et 18. Le coût de *aaa* est de 26. De manière plus générale, le coût du mot a^n avec $n \in \mathbb{N}$ un entier est supérieur à n. Nous pouvons en déduire que le coût du $\mathcal{S}$-AF est infini.

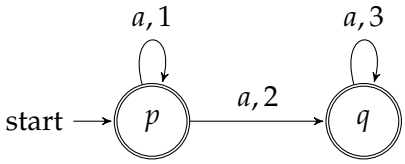


FIGURE 2.1 – Le coût du mot *aaa* est de 26.

2.2 Transducteurs

Les transducteurs sont des automates produisant une sortie sous forme de mot ou d'arbre. De la même manière que pour les automates à coûts, nous allons présenter plusieurs formalismes sur les modèles d'automates présentés en Section 1.3.

Nous présentons en premier les transducteurs finis. Ce modèle est basé sur les automates finis produisant des mots sur chaque transition, la sortie produite le long d'un calcul étant la concaténation de la sortie de chaque transition. Ce modèle est étudié depuis très longtemps, on le retrouve notamment pour la première fois dans les travaux de Seymour Ginsburg dans [Gin66], ceux de Samuel Eilenberg dans [Eil74] et ceux de Jean Berstel et Luc Boasson dans [BB79]. Une description complète des transducteurs finis peut être trouvée dans [Sak09].

Nous présenterons ensuite un modèle appelé *transducteur d'arbres* et où chaque transition produit un contexte d'arbre. En utilisant une substitution d'arbres il est alors possible de générer un arbre de sortie pour chaque arbre défini par le langage du de l'automate d'arbre sous-jacent. Ce modèle a été proposé pour la première fois par William C. Rounds dans [Rou68], puis par James W. Thatcher dans [Tha70]. Une description complète du modèle peut-être trouvée dans [CDG+07].

Le dernier modèle présenté est basé sur les automates à pile visible appelé *transducteur à pile visible*. Comme pour les transducteurs finis, chaque transition peut produire un mot de sortie, y compris les transitions empilant ou dépilant un symbole sur la pile. Une présentation complète des transducteurs à pile visible peut-être trouvée dans la thèse de Frédéric Servais ([Ser11]). Une autre version de ce résultat est disponible dans [RS08].

Définition 2.2.1

Un transducteur fini ou TF de l'alphabet Σ vers l'alphabet Σ' (resp. transducteur à pile visible ou TPV de l'alphabet structuré Δ vers l'alphabet structuré Δ') est une paire $T = (A, \lambda)$ composée d'un AF $A = (Q, I, F, \delta)$ sur Σ (resp. d'un APV $A = (Q, I, F, \delta, \Gamma)$ sur Δ et d'une fonction $\lambda : \delta \to \Sigma'^*$ (resp. $\lambda : \delta \to \Delta'^*$) .

À noter que ce modèle pourrait être vu comme un AF (resp. APV) à coûts sur le semi-anneau $(2^{\Sigma'^*}, \cup, \cdot, \varnothing, \{\epsilon\})$ (resp. $(2^{\Delta'^*}, \cup, \cdot, \varnothing, \{\epsilon\})$) où $\cdot$ est l'opérateur de concaténation défini pour tout $E_1, E_2 \in E$ de la manière suivante :

$$E_1 \cdot E_2 = \{w_1 w_2 \mid w_1 \in E_1, w_2 \in E_2\}$$

Définition 2.2.2

Un transducteur d'arbres ou TA de l'alphabet gradué (Σ, ar) vers l'alphabet (Σ', ar') est une paire $T = (A, \lambda)$ composée d'un d'un AA $A = (Q, I, \delta)$ sur (Σ, ar) et pour tout entier $n \in \mathbb{N}$ et toute transition $\tau \in \delta_n$ nous avons $\lambda(\tau) \in \mathcal{T}_{\Sigma'}(\mathcal{X}_n)$ avec $\mathcal{X}_n = \{x_1, \ldots, x_n\}$ un ensemble ordonné de variables.

Il existe un autre modèle de transducteur d'arbre plus traditionnel et présenté par exemple dans [CDG+07]. Nous lui préférons un autre modèle, présenté par Helmut Seidl dans [Sei94a]. En effet, dans ce manuscrit nous parlons de transducteurs d'arbres principalement au Chapitre 5 où nous présentons les résultats démontrés dans [Sei94a].

Pour la suite de cette section nous considérons le transducteur $T = (A, \lambda)$ où A peut être un AF, un APV ou un AA indistinctement .

Sortie d'un calcul Afin d'expliciter la sortie d'un calcul devons distinguer le type de A.

- Si A est un AF (resp. un APV) alors nous définissons la sortie $\mathsf{sortie}(\rho)$ d'un calcul $\rho = (\tau_1, \tau_2, \ldots, \tau_n) \in \delta^*$ d'un état ϕ vers un état ϕ (resp. d'une configuration ϕ vers une configuration ϕ) avec $n \in \mathbb{N}$ comme concaténation des sorties des transitions de ρ :

  $$\mathsf{sortie}(\rho) = \lambda(\tau_1) \ldots \lambda(\tau_n)$$

  De plus nous pouvons écrire ρ par :

  $$\rho = \phi_1 \xrightarrow{a_1,\lambda(\tau_1)} \phi_2 \xrightarrow{a_2,\lambda(\tau_2)} \ldots \xrightarrow{a_{n-1},\lambda(\tau_{n-1})} \phi_n \xrightarrow{a_n,\lambda(\tau_n)} \phi_{n+1}$$
  $$\rho : \phi_1 \xrightarrow{w,\lambda(\tau_1)\cdots\lambda(\tau_n)} {}^*\phi_{n+1}$$

  où la séquence de transition forme un calcul de A.

- Si A est un AA alors nous définissons la sortie $\mathsf{sortie}(\rho)$ d'un (q, R)-calcul $\rho \in \mathcal{T}_\delta(X)$ avec $\mathcal{X}$ un ensemble de variables, q un état et R une séquence d'états comme la substitution des sorties des transitions de ρ :

  $$\mathsf{sortie}(\rho) = \begin{cases} \lambda(\tau)[\mathsf{sortie}(\rho|_1), \ldots, \mathsf{sortie}(\rho|_m)] & \text{si } \rho = \tau(\rho|_1, \ldots, \rho|_m) \text{ avec } m \in \mathbb{N}, \\ x & \text{si } \rho = x \text{ avec } x \in \mathcal{X}. \end{cases}$$

  Nous pouvons aussi décrire ρ par $q \xrightarrow{t,\mathsf{sortie}(\rho)} {}^*R$.

Sortie d'un transducteur La sortie $\mathsf{sortie}(s)$ de $s \in \mathcal{L}(A)$ est définie comme suit :

$$\mathsf{sortie}(s) = \bigcup_{\rho \in \mathsf{ACal}_s(A)} (\mathsf{sortie}(\rho))$$

Autrement dit, la sortie de s est l'union des sorties des calculs acceptants sur s.

Transformation Soit T un transducteur $\mathcal{A}$. La transformation définie par T et dénotée par $\mathsf{trans}(T)$ est l'ensemble des paires de $(\mathcal{L}(A) \times E)$ définie par :

$$\mathsf{trans}(T) = \{(s, e) \in (\mathcal{L}(A) \times E) \mid s \in \mathcal{L}(A) \text{ et } e \in \mathsf{sortie}(s)\}$$

Autrement dit, la transformation de T est l'ensemble des paires de mots ou arbres reconnus par A et des mots ou arbres produits.

Norme La norme $\mathsf{nor}(s)$ de $s \in \mathcal{L}(A)$ est définie comme suit :

$$\mathsf{nor}(s) = \#\mathsf{sortie}(s)$$

La norme de T, dénotée par $\mathsf{nor}(T)$, est définie par :

$$\mathsf{nor}(T) = \sup\{\mathsf{nor}(s) \mid s \in \mathcal{L}(A)\}$$

Pour tout entier $k \in \mathbb{N}$ si $\mathsf{nor}(T) < k$ alors la norme de T est k-bornée. S'il existe un tel que $\mathsf{nor}(T) < k$ alors la norme de T est *finie*, sinon la norme de T est *infinie*.

Exemple 2.2.3

Nous présentons à la Figure 2.2 un TF sur l'alphabet $\Sigma = \{a\}$ vers l'alphabet $\Sigma = \{b, c\}$. Il existe trois calculs sur le mot *aaa* :

$$p \xrightarrow{a,bc} p \xrightarrow{a,bc} p \xrightarrow{a,cb} q$$
$$p \xrightarrow{a,bc} p \xrightarrow{a,cb} q \xrightarrow{a,bc} q$$
$$p \xrightarrow{a,cb} q \xrightarrow{a,bc} q \xrightarrow{a,bc} q$$

Ces trois calculs ont pour sorties respectives $bcbccb, bccbbc$ et $cbbcbc$. Ces trois mots sont différents, la sortie de *aaa* est donc $\{bcbccb, bccbbc, cbbcbc\}$, et sa norme est donc de 3. De manière plus générale, le nombre de sorties de a^n avec $n \in \mathbb{N}$ un entier est égal à n. Nous pouvons en déduire que la norme du TF est infinie.

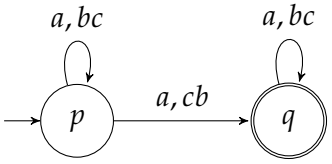


FIGURE 2.2 – $\mathcal{L}(T) = a^*$.

Exemple 2.2.4

Nous avons représenté à la Figure 2.3 un TPV T sur l'alphabet Δ vers l'alphabet Δ' avec $\Delta_c = \{c\}$, $\Delta_r = \{r\}$, $\Delta_\iota = \{a, b\}$ et $\Delta' = \Delta$. Ce TPV déterministe aplatit le mot en entrée en supprimant toutes les lettres de Δ_c et de Δ_r sauf les premières. Ainsi, le mot *cabcacbrrr* produit le mot *cababr*.

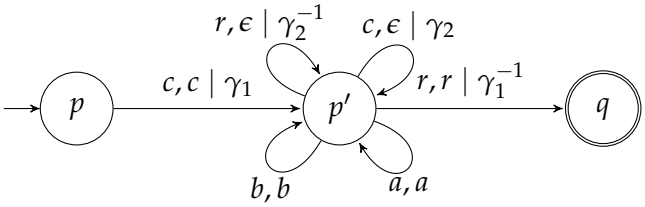


FIGURE 2.3 – Ce TPV aplatit l'entrée en supprimant les lettre c et r sauf les premières.

Exemple 2.2.5

Soit $\mathcal{X}_2$ un ensemble de 2 variables $\{x_1, x_2\}$. Nous notons T le TA $T = (A, \lambda)$ de (Σ, ar) vers (Σ, ar) avec $A = (Q, I, \delta)$, $\Sigma_0 = \{a, b\}$, $\Sigma_2 = \{f\}$, $Q = I = \{p\}$, $\delta = \{(p, f, p, q), (p, f, q, p), (q, f, q, q), (q, f, p, p), (q, a), (q, b)\}$ et :

- $\lambda((p, f, p, q)) = f(x_1, x_1)$,
- $\lambda((p, f, q, p)) = f(x_1, x_1)$,
- $\lambda((q, f, p, p)) = f(x_2, x_2)$,
- $\lambda((q, f, q, q)) = f(x_2, x_2)$,
- $\lambda((p, a)) = a$,
- $\lambda((q, b)) = b$.

Ce TA n'accepte que les arbres où le nombre de a est impair. De plus, il produit deux fois son sous-arbre gauche si le nombre de a est impair, et deux fois le sous-arbre droit si le nombre de a est pair. Un exemple de calcul est donné sur la Figure 2.4. A noter que chaque copie est associée au même sous-calcul. Il ne peut pas y avoir deux sous-calculs différents pour une même variable. De plus le transducteur a parcouru entièrement l'arbre avant de produire sa sortie, c'est la raison pour laquelle nous sommes capable de vérifier que le nombre de a est impair : les sous-arbres de droite sont effacés dans la sortie mais ont été quand même parcouru dans le calcul.

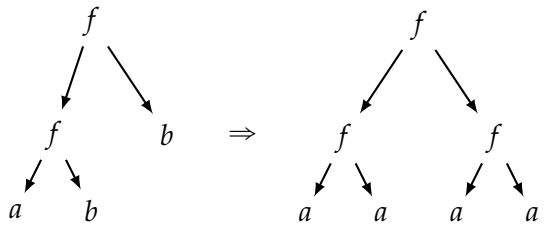


FIGURE 2.4 – $\lambda(f(f(a, b), b)) = f(f(a, a), f(a, a))$.

2.3 Équivalence des automates d'arbres et des automates à pile visible

Dans cette section, nous allons étudier les relations entre les automates à pile visible bien imbriqués et les automates d'arbres de plusieurs manières. Nous allons ainsi présenter deux algorithmes pour transformer dans un sens un automate à pile visible A vers un automate d'arbres A' reconnaissant le langage $\mathsf{fcns}(\mathcal{L}(A))$, et dans l'autre sens un automate d'arbres A' vers un automate à pile visible A reconnaissant le langage $\mathsf{enc}(\mathcal{L}(A'))$. Puis nous étendrons ces fonctions aux automates à coûts et aux transducteurs, montrant ainsi qu'elles induisent une bijection entre les relations reconnues par l'automate d'origine et l'automate produit.

Notez qu'ici nous n'allons considérer que les APVbi. Ce choix a été fait car les constructions sont plus simples et le cas bien imbriqué nous suffit pour la suite du manuscrit. Il existe

cependant une construction traitant du cas général dans [AM04], que nous n'utiliserons pas car elle n'induit pas une bijection entre les calculs de l'automate d'origine et de l'automate produit. Il est aussi proposée dans [AM04] une construction produisant un APV à partir d'un AA, mais elle ne traite pas du cas général. En effet, elle ne transforme que les AA lisant en entrée des *stack-tree*, une classe d'arbres où les symboles de l'alphabet gradué sont partitionnés en plusieurs sous-ensembles, de manière similaire aux alphabets structurés. Ces constructions pourraient être modifiées afin de résoudre ces deux problèmes, mais nous préférons proposer nos propres constructions afin de pouvoir plus facilement les adapter aux automates à coûts et aux transducteurs.

2.3.1 Construire un automate d'arbre depuis un automate à pile visible

Nous allons traiter en premier de la construction produisant un automate d'arbres à partir d'un automate à pile visible bien imbriqué. Comme un APV lit des mots et un AA des arbres, il est nécessaire de transformer les entrées de l'APV en entrée compatible avec celles d'un AA, ce que fera la fonction fcns présentée au premier chapitre.

Nous présentons maintenant la construction APV2AA. Intuitivement, elle produit à partir d'un APVbi A un AA A_{va} tel que $\mathcal{L}(A_{\mathsf{va}}) = \{\mathsf{fcns}(w) \mid w \in \mathcal{L}(A)\}$.

Définition 2.3.1

Soit $A = (Q, I, F, \delta, \Gamma)$ un APVbi sur l'alphabet Δ. L' AA $\mathsf{APV2AA}(A) = (Q', I', \delta')$ sur l'alphabet structuré $(\tilde{\Delta}, \mathsf{ar})$ est défini par :

- $Q' = \mathsf{BI}_A \times (\Gamma \cup \{\bot\})$,
- $I' = (\mathsf{BI}_A \cap (I \times F)) \times \{\bot\}$,
- $\delta' = \left\{ ((p,q,\gamma),(c,r),(p',q',\gamma'),(q'',q,\gamma)) \;\middle|\; \begin{array}{l} (p,c,\gamma',p') \in \delta_c, \\ (q',\gamma',r,q'') \in \delta_r, \\ \gamma \in \Gamma \text{ et } (p,q),(p',q'), \\ (q'',q) \in \mathsf{BI}_A \end{array} \right\}$
  $\cup \quad \{((p,q,\gamma),a,(p',q,\gamma)) \mid (p,a,p') \in \delta_\iota, \gamma \in \Gamma \text{ et } (p,q),(p',q) \in \mathsf{BI}_A\}$
  $\cup \quad \{((p,p,\gamma),\top) \mid \gamma \in \Gamma \text{ et } p \in Q\}$

Quand il existe un calcul $\rho : (p,\gamma) \xrightarrow{u}{}^*(q,\gamma)$ de A, cela signifie qu'il existe un (p,q,γ)-calcul de $\mathsf{APV2AA}(A)$ sur un arbre $t = \mathsf{fcns}(u)$. De plus comme u est bien imbriqué nous savons que $u = cwrv$ ou $u = av$ ou $u = \epsilon$ avec $v, w \in \Delta_{\mathsf{bi}}^*$, c un ouvrant et r un fermant. Chaque transition de l'automate d'arbres est associé à une de ces décompositions. Par exemple, si $u = cwrv$, alors ρ se décompose par :

$$\rho : (p,\gamma) \xrightarrow{c} (p',\gamma\gamma') \xrightarrow{w}{}^* (q',\gamma\gamma') \xrightarrow{r} (q'',\gamma) \xrightarrow{v}{}^* (q,\gamma)$$

La situation est représentée à la Figure 2.5. Le calcul correspondant de $\mathsf{APV2AA}(A)$ est égal à $\tau(\rho_1,\rho_2)$ avec $\tau = ((p,q,\gamma),(c,r),(p',q',\gamma'),(q'',q,\gamma))$, ρ_1 un (p',q',γ')-calcul et ρ_2 un (q'',q,γ)-calcul. Il en va de même pour les autres transitions. A noter que nous mémorisons un symbole de pile afin de préserver une bijection entre les calculs de A et $\mathsf{APV2AA}(A)$. En effet, sans ce symbole nous pourrions confondre le couple $(p,c,\gamma,q) \in \delta_c$ et $(p',\gamma,r,q') \in \delta_r$ avec le couple $(p,c,\gamma',q) \in \delta_c$ et $(p',\gamma',r,q') \in \delta_r$.

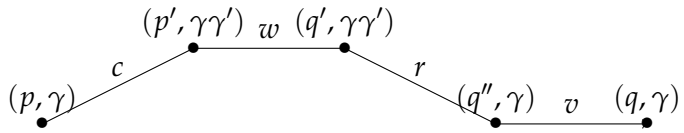


FIGURE 2.5 – Construction des transitions d'empilement.

Exemple 2.3.2

Soit A l' APVbi de la Figure 2.6. APV2AA produit l' AA $A' = (Q', I', \delta')$ avec :

- $Q' = \left\{ \begin{array}{l} (p_i, p_f, \bot), (p_1, p_2, \gamma_1), (p_1, p_2, \gamma_2), (p_1, p_1, \gamma_1), \\ (p_2, p_2, \gamma_1), (p_2, p_2, \gamma_2), (p_f, p_f, \bot) \end{array} \right\}$,
- $I' = \{(p_i, p_f, \bot)\}$,
- $\delta' = \left\{ \begin{array}{l} ((p_i, p_f, \bot), (a, b), (p_1, p_1, \gamma_1), (p_f, p_f, \bot)), \\ ((p_i, p_f, \bot), (a, b), (p_1, p_2, \gamma_1), (p_f, p_f, \bot)), \\ ((p_1, p_2, \gamma_1), (a, b), (p_2, p_2, \gamma_2), (p_2, p_2, \gamma_1)), \\ ((p_1, p_2, \gamma_1), (a, b), (p_1, p_2, \gamma_2), (p_2, p_2, \gamma_1)), \\ ((p_1, p_2, \gamma_2), (a, b), (p_2, p_2, \gamma_2), (p_2, p_2, \gamma_2)), \\ ((p_1, p_2, \gamma_2), (a, b), (p_1, p_2, \gamma_2), (p_2, p_2, \gamma_1)), \\ ((p_1, p_2, \gamma_1), (a, b), (p_2, p_2, \gamma_2), (p_2, p_2, \gamma_1)), \\ ((p_f, p_f, \bot), \bot), \\ ((p_1, p_1, \gamma_1), \bot), \\ ((p_2, p_2, \gamma_1), \bot), \\ ((p_2, p_2, \gamma_2), \bot) \end{array} \right\}$

A noter que nous avons supprimé les états n'étant pas accessibles et co-accessibles.

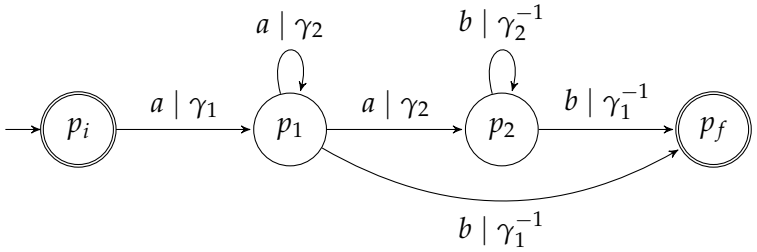


FIGURE 2.6 – $\mathcal{L}(A) = \{a^n b^n \mid n \in \mathbb{N}\}$

Nous allons maintenant montrer le résultat suivant :

Théorème 2.3.3

Soit A un APVbi et A_{va} l' AA APV2AA(A). Nous avons :

- $\mathcal{L}(A_{\mathsf{va}}) = \{\mathsf{fcns}(w) \mid w \in \mathcal{L}(A)\}$
- Pour tout $w \in \mathcal{L}(A)$ il existe une bijection entre $\mathsf{ACal}_w(A)$ et $\mathsf{ACal}_{\mathsf{fcns}(w)}(A_{\mathsf{va}})$.

Soit $A = (Q, I, F, \Gamma, \delta)$ un APVbi et A_{va} l' AA APV2AA(A) (va pour *visible* vers *arbre*). Nous allons maintenant définir une fonction liant les calculs de A avec ceux de A_{va}. Le but de cette fonction est d'établir comment les calculs de A sont produits à partir de A_{va}. Soient Φ_{va} une fonction de $\mathsf{Cal}(A_{\mathsf{va}}) \times \Gamma^*$ vers $\mathsf{Cal}(A)$, $\sigma \in \Gamma^*$ une pile et $\rho \in \mathsf{Cal}(A_{\mathsf{va}})$ un calcul défini comme suit :

$$\rho = \tau(\rho|_1, \ldots, \rho|_m) \text{ avec } m \in [0, 2]$$

Nous définissons $\Phi_{\mathsf{va}}(\rho, \sigma) \in \mathsf{Cal}(A)$ comme étant le calcul suivant :

$$\Phi_{\mathsf{va}}(\rho, \sigma) = \begin{cases} (p, \sigma\gamma) \xrightarrow{c} \Phi_{\mathsf{va}}(\rho|_1, \sigma\gamma) \xrightarrow{r} \Phi_{\mathsf{va}}(\rho|_2, \sigma\gamma) & \text{si } \tau = ((p, q'', \gamma), (c, r), (p', q', \gamma'), (q, q'', \gamma)) \\ (p, \sigma\gamma) \xrightarrow{a} \Phi_{\mathsf{va}}(\rho|_1, \sigma) & \text{si } \tau = ((p, q, \gamma), a, (p', q, \gamma)) \\ (p, \sigma\gamma) & \text{si } \tau = ((p, p, \gamma), \top) \end{cases}$$

Exemple 2.3.4

Nous considérons l' APV A et l'AA de l'Exemple 2.3.2. Nous avons représenté à la Figure 2.7 un calcul ρ sur l'arbre $(a, b)((a, b)(\bot, \bot), \bot) = \mathsf{fcns}(aabb)$. La fonction $\Phi_{\mathsf{va}}(\rho, \bot)$ produit le calcul ρ' sur le mot $aabb$ défini comme suit :

$$\rho' = (p_i, \bot) \xrightarrow{a} (p_1, \gamma_1) \xrightarrow{a} (p_2, \gamma_1\gamma_2) \xrightarrow{b} (p_2, \gamma_1) \xrightarrow{b} (p_f, \bot)$$

qui est bien un calcul de A sur $aabb$.

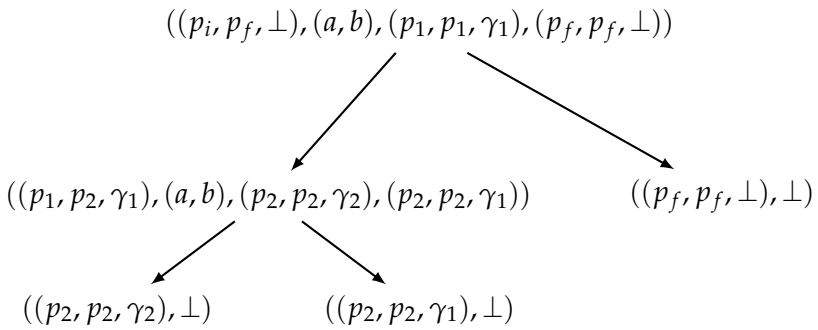


FIGURE 2.7 – Un calcul sur l'arbre $(a, b)((a, b)(\bot, \bot), \bot)$.

Nous allons maintenant montrer plusieurs propriétés sur la fonction Φ_{va}. Le Lemme 2.3.5 montre que la fonction construit bien un calcul de l' APVbi de départ. Nous montrons ensuite que cette fonction est surjective au Lemme 2.3.6 puis nous montrons qu'elle est injective au Lemme 2.3.7.

Lemme 2.3.5

Soit A un APVbi sur un alphabet structuré Δ et $t \in \mathcal{T}_{\tilde{\Delta}}$ un arbre. Pour tout (p, q, γ)-calcul ρ de APV2AA(A) sur t et pour toute pile σ, il existe un mot $u = \mathsf{fcns}(t)$ tel que $\Phi_{\mathsf{va}}(\rho, \sigma)$ est un calcul de A sur u de $(p, \sigma\gamma)$ à $(q, \sigma\gamma)$.

Démonstration. Nous notons $A = (Q, I, F, \Gamma, \delta)$ et $\mathsf{APV2AA}(A) = A_{\mathsf{va}} = (Q', I', \delta')$. Nous allons prouver ce lemme par induction sur la structure de t.

- si $t = \top$, alors $u = \epsilon$. Par définition $\Phi_{\mathsf{va}}(\rho, \sigma)$ est un calcul de A sur ϵ, le résultat est donc vérifié.
- si $t = a(t|_1)$ avec $a \in \Delta_\iota$, alors il existe $((p, q, \gamma), a, (p', q, \gamma)) \in \delta'$ et un (p', q, γ)-calcul ρ_1 de A sur $t|_1$. Par hypothèse d'induction, il existe un mot $v \in \Delta^*_{\mathsf{bi}}$ tel que $t|_1 = \mathsf{fcns}(v)$ et $\Phi_{\mathsf{va}}(\rho_1, \sigma)$ est un calcul de A sur v. Par définition de δ', il existe $(p, a, p') \in \delta_\iota$ et donc $\Phi_{\mathsf{va}}(\rho, \sigma) = (p, \sigma\gamma) \xrightarrow{a} \Phi_{\mathsf{va}}(\rho_1, \sigma\gamma)$ est un calcul de A sur $u = av$.
- si $t = (c, r)(t|_1, t|_2)$ avec $c \in \Delta_c$ et $r \in \Delta_r$, alors il existe $((p, q, \gamma), (c, r), (p', q', \gamma'), (q'', q, \gamma)) \in \delta'$, un (p', q', γ')-calcul ρ_1 de A sur $t|_1$ et un (q'', q, γ)-calcul ρ_2 de A sur $t|_2$. Par hypothèse d'induction, il existe deux mots $w, v \in \Delta^*_{\mathsf{bi}}$ tels que $t|_1 = \mathsf{fcns}(w)$, $t|_2 = \mathsf{fcns}(v)$, $\Phi_{\mathsf{va}}(\rho_1, \sigma\gamma)$ est un calcul de A sur w. et $\Phi_{\mathsf{va}}(\rho_2, \sigma)$ est un calcul de A sur v. Par définition de δ', il existe $(p, c, \gamma', p') \in \delta_c$ et $(q', \gamma', r, q'') \in \delta_r$, donc $\Phi_{\mathsf{va}}(\rho, \sigma) = (p, \sigma\gamma) \xrightarrow{c} \Phi_{\mathsf{va}}(\rho_1, \sigma\gamma) \xrightarrow{r} \Phi_{\mathsf{va}}(\rho_2, \sigma\gamma)$ est un calcul de A sur $u = cwrv$. A noter que la présence de γ' dans l'état nous permet d'éviter que $\Phi_{\mathsf{va}}(\rho, \sigma)$ puisse produire plusieurs calculs différents. □

Lemme 2.3.6

Soit A un APVbi sur un alphabet structuré Δ et $u \in \Delta^*_{\mathsf{bi}}$ un mot bien imbriqué. Pour tout calcul $\rho' : (p, \sigma\gamma) \xrightarrow{u}{}^* (q, \sigma\gamma)$ de A, il existe un calcul (p, q, γ)-calcul ρ de $\mathsf{APV2AA}(A)$ sur $\mathsf{fcns}(u)$ tel que $\Phi_{\mathsf{va}}(\rho, \sigma) = \rho'$.

Démonstration. Nous notons $A = (Q, I, F, \Gamma, \delta)$ et $\mathsf{APV2AA}(A) = A_{\mathsf{va}} = (Q', I', \delta')$. Nous faisons la preuve par induction sur la structure imbriquée de u.

- Si $u = \epsilon$ alors $q = p$. Par construction de δ', nous avons $((p, p, \gamma)) \in \delta'$, et donc il existe un (p, q, γ)-calcul ρ de A_{va} sur $\top$ tel que $\Phi_{\mathsf{va}}(\rho, \sigma) = \rho'$.
- Si $u = av$ avec $v \in \Delta^*_{\mathsf{bi}}$ et $a \in \Delta_\iota$ alors ρ' se décompose ainsi :

  $$(p, \sigma\gamma) \xrightarrow{a} (p', \sigma\gamma) \xrightarrow{v}{}^* (q, \sigma\gamma)$$

  Par hypothèse d'induction, nous savons qu'il existe un (p', q, γ)-calcul ρ_1 de A_{va} sur $\mathsf{fcns}(v)$ tel que $\Phi_{\mathsf{va}}(\rho_1, \sigma) = (p', \sigma\gamma) \xrightarrow{v}{}^* (q, \sigma\gamma)$. De plus par construction de δ', nous avons $((p, q, \gamma), a, (p', q, \gamma)) \in \delta'$, et donc il existe un (p, q, γ)-calcul ρ de A_{va} sur $\mathsf{fcns}(u) = a(\mathsf{fcns}(v))$ tel que $\Phi_{\mathsf{va}}(\rho, \sigma) = \rho'$.
- Si $u = cwrv$ avec $w, v \in \Delta^*_{\mathsf{bi}}$, $c \in \Delta_c$ et $r \in \Delta_r$ alors ρ' se décompose ainsi :

  $$(p, \sigma\gamma) \xrightarrow{c} (p', \sigma\gamma\gamma') \xrightarrow{w}{}^* (q', \sigma\gamma\gamma') \xrightarrow{r} (q'', \sigma\gamma) \xrightarrow{v}{}^* (q, \sigma\gamma)$$

  Par hypothèse d'induction, nous savons qu'il existe un (p', q', γ')-calcul ρ_1 et un (q'', q, γ)-calcul ρ_2 de A_{va} sur $\mathsf{fcns}(w)$ et $\mathsf{fcns}(v)$ tels que $\Phi_{\mathsf{va}}(\rho_1, \sigma\gamma) = (p', \sigma\gamma\gamma') \xrightarrow{w}{}^* (q', \sigma\gamma\gamma')$ et $\Phi_{\mathsf{va}}(\rho_2, \sigma) = (q'', \sigma\gamma) \xrightarrow{v}{}^* (q, \sigma\gamma)$. Nous avons $((p, q, \gamma), (c, r), (p', q', \gamma'), (q'', q, \gamma)) \in \delta'$ par construction de δ', et il existe donc un (p, q, γ)-calcul ρ de A_{va} sur l'arbre $\mathsf{fcns}(u) = (c, r)(\mathsf{fcns}(w), \mathsf{fcns}(v))$ tel que $\Phi_{\mathsf{va}}(\rho, \sigma) = \rho'$. □

Lemme 2.3.7

Soit $A = (Q, I, F, \Gamma, \delta)$ un APVbi sur un alphabet structuré Δ et $u \in \Delta_{\mathsf{bi}}^*$ un mot bien imbriqué. Pour tout (p, q, γ)-calculs ρ, ρ' de $\mathsf{APV2AA}(A)$ sur $\mathsf{fcns}(u)$ et $\sigma \in \Gamma^*$, si $\Phi_{\mathsf{va}}(\rho, \sigma) = \Phi_{\mathsf{va}}(\rho', \sigma)$ alors $\rho = \rho'$.

Démonstration. Nous notons $\mathsf{APV2AA}(A) = A_{\mathsf{va}} = (Q', I', \delta')$. Nous faisons la preuve par induction sur la structure imbriquée de u.

- Si $u = \epsilon$ alors $\rho = (p, \sigma\gamma)$. Par définition de Φ_{va} nous avons donc $\rho = \rho' = ((p, p, \gamma), \top)$.
- Si $u = av$ avec $v \in \Delta_{\mathsf{bi}}^*$ et $a \in \Delta_\iota$ alors $\Phi_{\mathsf{va}}(\rho, \sigma) = \Phi_{\mathsf{va}}(\rho', \sigma)$ se décomposent ainsi :

  $$(p, \sigma\gamma) \xrightarrow{a} (p', \sigma\gamma) \xrightarrow{v}{}^* (q, \sigma\gamma)$$

  Par définition de Φ_{va} nous avons donc $\rho = \tau(\rho|_1)$ et $\rho' = \tau'(\rho'|_1)$ avec $\Phi_{\mathsf{va}}(\rho|_1, \sigma) = \Phi_{\mathsf{va}}(\rho'|_1, \sigma) = (p', \sigma\gamma) \xrightarrow{v}{}^* (q, \sigma\gamma)$. Par hypothèse d'induction, nous avons $\rho|_1 = \rho'|_1$. De plus, par construction il existe une seule transition $((p, q, \gamma), a, (p', q, \gamma)) \in \delta'$, donc $\tau = \tau'$ et donc $\rho = \rho'$.
- Si $u = cwrv$ avec $w, v \in \Delta_{\mathsf{bi}}^*$, $c \in \Delta_c$ et $r \in \Delta_r$ alors ρ' se décompose ainsi :

  $$(p, \sigma\gamma) \xrightarrow{c} (p', \sigma\gamma\gamma') \xrightarrow{w}{}^* (q', \sigma\gamma\gamma') \xrightarrow{r} (q'', \sigma\gamma) \xrightarrow{v}{}^* (q, \sigma\gamma)$$

  Par définition de Φ_{va} nous avons $\rho = \tau(\rho|_1, \rho|_2)$ et $\rho' = \tau'(\rho'|_1, \rho'|_2)$ avec $\Phi_{\mathsf{va}}(\rho|_1, \sigma\gamma) = \Phi_{\mathsf{va}}(\rho'|_1, \sigma\gamma) = (p', \sigma\gamma\gamma') \xrightarrow{w}{}^* (q', \sigma\gamma\gamma')$ et $\Phi_{\mathsf{va}}(\rho|_2, \sigma) = \Phi_{\mathsf{va}}(\rho'|_2, \sigma) = (q'', \sigma\gamma) \xrightarrow{v}{}^* (q, \sigma\gamma)$ Par hypothèse d'induction, nous avons $\rho|_1 = \rho'|_1$ et $\rho|_2 = \rho'|_2$. De plus, il existe une seule transition $((p, q, \gamma), (c, r), (p', q', \gamma'), (q'', q, \gamma)) \in \delta'$, donc $\tau = \tau'$ et donc $\rho = \rho'$. □

Nous pouvons maintenant combiner ces résultats afin de montrer le Théorème 2.3.3.

Démonstration du Théorème 2.3.3. Les Lemmes 2.3.5, 2.3.6 et 2.3.7 assurent que pour tout mot $u \in \Delta_{\mathsf{bi}}^*$, $\mathsf{ACal}_u(A)$ est en bijection avec $\mathsf{ACal}_{\mathsf{fcns}(u)}(A_{\mathsf{va}})$. De plus, par le Lemme 2.3.5, pour tout arbre $t \in \mathcal{L}(A_{\mathsf{va}})$ il existe $u \in \Delta_{\mathsf{bi}}^*$ tel que $t = \mathsf{fcns}(u)$, ceci assurant la propriété souhaitée sur tous les calculs acceptants de A et A_{va}. □

2.3.2 Construire un automate à pile visible depuis un automate d'arbre

Nous allons maintenant construire un automate à pile visible depuis un automate d'arbres. Pour cela nous allons utiliser la fonction enc définie au premier chapitre, et permettant de construire un mot bien imbriqué depuis un arbre. Nous définissons maintenant les fonctions anc et suiv nous permettant d'obtenir des informations sur les relations de parenté entre les transitions d'un automate d'arbre.

Ancêtres d'une transition Soit $A = (Q, I, \delta)$ un AA sur l'alphabet gradué (Σ, ar). Nous appelons Qd_A l'ensemble des paires $(Q \cup \delta) \times \mathbb{N}$. Pour tout $\tau \in \delta$ nous définissons la fonction $\mathsf{anc} : \delta \to \mathsf{Qd}_A$ (ancêtres), comme suit :

$$\mathsf{anc}(\tau) = \{(\tau', n) \in \mathsf{Qd}_A \mid \tau' \in \delta, n \leqslant \mathsf{ar}(\tau'), \mathsf{fils}_n(\tau') = \mathsf{racine}(\tau)\} \cup \{(p, 0) \mid p \in I, p = \mathsf{racine}(\tau)\}$$

Enfin, pour tout $(x, m) \in \mathsf{Qd}_A$, nous considérons la fonction $\mathsf{suiv} : \mathsf{Qd}_A \to \mathsf{Qd}_A$ (suivant), formellement définie par :

$$\mathsf{suiv}((x, m)) = (x, m + 1)$$

Ces fonctions sont utiles pour connaître les ancêtres possibles d'une transition sur le calcul d'un arbre. Nous allons nous en servir dans la construction AA2APV, que nous allons présenter maintenant. Intuitivement, elle produit à partir d'un AA A un APV A_{av} tel que $\mathcal{L}(A_{\mathsf{av}}) = \{\mathsf{enc}(t) \mid t \in \mathcal{L}(A_{\mathsf{av}})\}$.

Définition 2.3.8

Soit $A = (Q, I, \delta)$ un AA sur l'alphabet (Σ, ar). L'APV $\mathsf{AA2APV}(A) = (Q', I', F', \delta', \Gamma')$ sur l'alphabet structuré $\hat{\Sigma}$ est défini par :

- $Q' = \{(p,i) \in \mathsf{Qd}_A \mid p \in I, i \in \{0,1\}\} \cup \{(\tau,i) \in \mathsf{Qd}_A \mid \tau \in \delta, i \in \{0,\dots,\mathsf{ar}(\tau)\}\}$,
- $I' = \{(p,0) \in \mathsf{Qd}_A \mid p \in I\}$,
- $F' = \{(p,1) \in \mathsf{Qd}_A \mid p \in I\}$,
- $\Gamma' = Q' \times \Sigma$,
- $\delta' = \delta'_c \cup \delta'_r$ avec :
  - $\delta'_c = \{(\psi, \langle a, (\psi,a), (\tau',0)) \mid \mathsf{etiq}(\tau') = a, \psi \in \mathsf{anc}(\tau')\}$
  - $\delta'_r = \{((\tau', \mathsf{ar}(\tau')), (\psi,a), a\rangle, \mathsf{suiv}(\psi)) \mid \mathsf{etiq}(\tau') = a, \psi \in \mathsf{anc}(\tau')\}$

Cette construction simule le parcours en profondeur d'un calcul en stockant dans l'état de l'APV la transition utilisée par l' AA. Lorsque nous empilons un symbole, nous débutons le parcours d'une nouvelle transition τ de notre calcul. Si l'on suppose que cette transition possède une arité de m, alors l'APV devra encore passer m fois par cette transition afin de parcourir tous ses fils. L'entier stocké dans l'état permet de retenir cette information. Ainsi, si nous sommes dans l'état (τ, n) avec $n < m$, alors la prochaine transition du calcul à explorer est celle étiquetant le fils n de τ. Inversement, lorsque nous dépilons un symbole, nous finissons du parcours de la transition τ. Il est donc nécessaire de remonter à la transition supérieure de notre calcul. Nous incrémentons le compteur de 1, pour signifier le passage au fils suivant.

Exemple 2.3.9

Nous considérons l' AA $A = (Q, I, F)$ sur (Σ, ar) avec $\Sigma_2 = \{f\}$, $\Sigma_0 = \{a\}$, $Q = I = \{p\}$ et $\delta = \{(p,f,p,p), (p,a)\}$ avec $(p,f,p,p) = \tau$ et $(p,a) = \tau'$. La construction APV2AA produit l' APVbi A' représenté à la Figure 2.8.

Nous allons maintenant prouver :

Théorème 2.3.10

Soit A un AA et A_{av} l' APVbi $\mathsf{AA2APV}(A)$. Nous avons :

- $\mathcal{L}(A_{\mathsf{av}}) = \{\mathsf{enc}(t) \mid t \in \mathcal{L}(A)\}$
- Pour tout $t \in \mathcal{L}(A)$ il existe une bijection entre $\mathsf{ACal}_t(A)$ et $\mathsf{ACal}_{\mathsf{enc}(t)}(A_{\mathsf{av}})$.

Afin de prouver ce théorème nous allons tout d'abord montrer que le domaine de l'automate d'arbre construit est bien un sous-ensemble du co-domaine de la fonction enc. Pour cela nous allons tout d'abord montrer au Lemme 2.3.11 une propriété des calculs de la construction AA2APV.

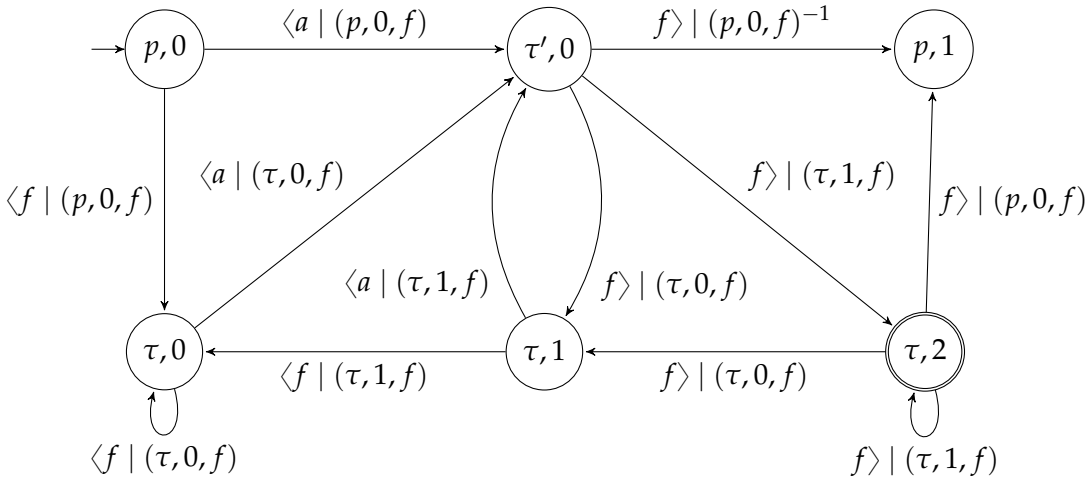


FIGURE 2.8 – Un APVbi produit par la fonction APV2AA.

Lemme 2.3.11

Soit A un AA sur l'alphabet gradué (Σ, ar). Pour tout calcul $((x,n),\bot) \xrightarrow{w}{}^*((y,m),\bot) \in \mathsf{Cal}_w(\mathsf{AA2APV}(A))$ tel que $w \in \hat{\Sigma}^*_{\mathsf{cr}}$ nous avons $x = y$ et $m = n+1$.

Démonstration. Comme $w \in \hat{\Sigma}^*_{\mathsf{cr}}$, il existe $c \in \hat{\Sigma}_c$, $r \in \hat{\Sigma}_r$ et $u \in \hat{\Sigma}^*_{\mathsf{bi}}$ tel que $w = cur$. Par construction il existe $a \in \Sigma$ et $x' \in Q \cup \delta$ tels que $((x,n), \langle a, ((x,n),a), (x',0)) \in \delta'_c$. Par définition de δ', les transitions pouvant dépiler le symbole $((x,m), \langle a)$ sont de la forme $((y',k), ((x,n),a), a\rangle, (x, n+1)) \in \delta'_r$ avec $y' \in Q \cup \delta$ et $k = 1$ si $y' \in Q$ et $k = \mathsf{ar}(y')$ sinon. Nous avons donc bien $x = y$ et $m = n+1$. □

Lemme 2.3.12

Soient $A = (Q, I, \delta)$ un AA sur l'alphabet gradué (Σ, ar) et $A_{\mathsf{av}} = (Q', I', F', \delta', \Gamma')$ l'APVbi $\mathsf{AA2APV}(A)$ sur l'alphabet structuré $\hat{\Sigma}$. Pour tout mot $w \in \mathcal{L}(A_{\mathsf{av}})$, il existe $t \in \mathcal{T}_\Sigma$ tel que $w = \mathsf{enc}(t)$.

Démonstration. Il convient de noter que pour tout mot $w \in \mathcal{L}(A_{\mathsf{av}})$, nous avons $w \in \hat{\Sigma}^*_{\mathsf{cr}}$. En effet, comme $F' \cap I' = \varnothing$, nous avons $w \neq \epsilon$. Comme $\hat{\Sigma}_\iota = \varnothing$ nous pouvons en déduire que pour tout calcul $((p,0),\bot) \xrightarrow{w}{}^*((q,1),\sigma) \in \mathsf{ACal}_w(A_{\mathsf{av}})$ il existe $\tau, \tau' \in \delta$ tels que ce calcul est égal à :

$$((p,0),\bot) \xrightarrow{\langle a} ((\tau,0),((p,0),a)) \xrightarrow{u}{}^*((\tau',\mathsf{ar}(\tau')),\sigma\cdot((q,0),b)) \xrightarrow{b\rangle} ((q,1),\sigma)$$

Avec $u \in \hat{\Sigma}^*$ et $w = \langle aub\rangle$. Par définition de δ' l'état $(q,0)$ est le seul état pouvant empiler $((q,0),b))$, et ne peut être rencontré qu'en position initiale. Nous avons donc $p = q$, $a = b$ et $\sigma = \bot$. Nous concluons que $w = cur$ avec $c = \langle a, r = a\rangle$ et $u \in \hat{\Sigma}^*_{\mathsf{bi}}$.

Soit $w \in \hat{\Sigma}^*_{\mathsf{cr}}$ un mot et $((x,n),\bot) \xrightarrow{w}{}^*((y,m),\bot)$ un calcul de A_{av} sur w. Par le Lemme 2.3.11 nous avons $x = y$ et $m = n+1$. Nous montrons maintenant par induction sur la structure de w qu'il existe $t \in \mathcal{T}_\Sigma$ tel que $w = \mathsf{enc}(t)$. Comme $w \in \hat{\Sigma}^*_{\mathsf{cr}}$, il existe $l \in \mathbb{N}$ tel que $w = cu_1 \dots u_l r$ avec $c \in \hat{\Sigma}_c$, $r \in \hat{\Sigma}_r$ et $u_i \in \hat{\Sigma}^*_{\mathsf{cr}}$ pour tout $i \in [1,l]$. Par construction de δ' et par le Lemme 2.3.11,

il existe $x' \in Q \cup \delta$ et $a \in \Sigma$ tels que :

$$((x,n),\langle a,((x,n),a),(x',0))\in\delta'_c \qquad ((x',l),((x,n),a),a\rangle,(x,n+1))\in\delta'_r$$
$$((x',0),((x,n),a))\xrightarrow{u_1}{}^*((x',1),((x,n),a))\xrightarrow{u_2}{}^*\cdots\xrightarrow{u_l}{}^*((x',l),((x,n),a))\in \mathsf{Cal}_{u_1\dots u_l}(A_{\mathsf{av}})$$

Avec $c=\langle a, r=a\rangle$. Par hypothèse d'induction et par le Lemme 1.4.7 il existe $t_i \in \mathcal{T}_\Sigma$ pour tout $i \in [1,l]$ tels que $u_i = \mathsf{enc}(t_i)$ pour tout $i \in [1,l]$. Par définition de δ', $l = \mathsf{ar}(a)$, nous en déduisons que $w = \langle a \cdot \mathsf{enc}(t_1)\dots\mathsf{enc}(t_l)\cdot a\rangle = \mathsf{enc}(a(t_1,\dots,t_l))$. Comme pour tout mot $w \in \mathcal{L}(A_{\mathsf{av}})$, nous avons $w \in \hat{\Sigma}^*_{\mathsf{cr}}$, nous pouvons conclure qu'il existe $t \in \mathcal{T}_\Sigma$ tel que $w = \mathsf{enc}(t)$ avec $c = \langle a$ et $r = a\rangle$. □

Soit $A = (Q, I, \delta)$ un AA et A_{av} l' APVbi AA2APV(A). Nous allons maintenant définir une fonction liant les calculs de A avec ceux de A_{av}. Le but de cette fonction est d'établir comment les calculs de A_{av} sont produits à partir de A. Soient Φ_{av} une fonction de $\mathsf{Cal}(A)\times\mathsf{Qd}_A\times\Gamma^*$ vers $\mathsf{Cal}(A_{\mathsf{av}})$, une paire $\psi \in \mathsf{Qd}_A$, $\sigma \in \Gamma^*$ une pile et $\rho \in \mathsf{Cal}(A)$ le calcul $\rho = \tau(\rho|_1,\dots,\rho|_m)$. Nous définissons $\Phi_{\mathsf{TA}}(\rho,\psi,\sigma) \in \mathsf{Cal}(A_{\mathsf{av}})$ comme étant le calcul suivant :

$$\Phi_{\mathsf{av}}(\rho,\psi,\sigma) = (\psi,\sigma)\xrightarrow{\langle a}\Phi_{\mathsf{av}}(\rho_1,(\tau,0),\sigma(\psi,a))\cdots\Phi_{\mathsf{av}}(\rho_m,(\tau,m-1),\sigma(\psi,a))\xrightarrow{a\rangle}(\mathsf{suiv}(\psi),\sigma)$$

Exemple 2.3.13

Nous considérons l' AA $A = (Q, I, F)$, l' APVbi A' défini à l'Exemple 2.3.9 et le calcul $\rho = \tau[\tau',\tau']$ de A. La fonction $\Phi_{\mathsf{va}}(\rho,p,\perp)$ produit le calcul ρ' défini comme suit :

$$\begin{aligned}\rho' =&((p,0),\perp)\xrightarrow{\langle f}((\tau,0),(p,0,f))\xrightarrow{\langle a}((\tau',0),(p,0,f)(\tau,0,f))\xrightarrow{a\rangle}\\ &((\tau,1),(p,0,f))\xrightarrow{\langle a}((\tau',0),(p,0,f)(\tau,1,f))\xrightarrow{a\rangle}((\tau,2),(p,0,f))\xrightarrow{f\rangle}((p,1),\perp)\end{aligned}$$

qui est bien un calcul de A' sur $\langle f\langle aa\rangle\langle aa\rangle f\rangle$.

Nous allons maintenant prouver au Lemme 2.3.14 que la fonction construit bien un calcul de l' AA de départ. Nous montrons ensuite que cette fonction est surjective au Lemme 2.3.15 puis nous montrons qu'elle est injective au Lemme 2.3.16.

Lemme 2.3.14

Soit A un AA sur un alphabet (Σ,ar) et $t = a(t|_1,\dots t|_m) \in \mathcal{T}_\Sigma$ un arbre. Pour tout calcul $\rho = \tau(\rho|_1,\dots,\rho|_m)$ de A sur t, toute paire $\psi \in \mathsf{anc}(\tau)$ et toute pile σ, $\Phi_{\mathsf{av}}(\rho,\psi,\sigma)$ est un calcul de APV2AA(A) sur $\mathsf{enc}(t)$.

Démonstration. Nous notons $A = (Q', I', \delta')$ et APV2AA$(A) = A_{\mathsf{av}} = (Q, I, \delta)$. Nous allons prouver ce lemme par induction sur la structure de t. Par hypothèse d'induction, pour tout $i \in [1,m]$, $\Phi_{\mathsf{av}}(\rho_i,(\tau,i-1),\sigma(\psi,a))$ est un calcul de A_{av} sur le mot $\mathsf{enc}(t|_i)$. Par construction, nous avons $\psi \in \mathsf{anc}(\tau)$ et $\mathsf{etiq}(\tau) = f$. Nous pouvons donc en conclure que $(\psi,\langle a,(\psi,a),(\tau',0))\in\delta'_c$ et $((\tau',\mathsf{ar}(\tau')),(\psi,a),a\rangle,\mathsf{suiv}(\psi))\in\delta'_r$, et le calcul suivant :

$$(\psi,\sigma)\xrightarrow{\langle a}\Phi_{\mathsf{av}}(\rho_1,(\tau,0),\sigma(\psi,a))\cdots\Phi_{\mathsf{av}}(\rho_m,(\tau,m-1),\sigma(\psi,a))\xrightarrow{a\rangle}(\mathsf{suiv}(\psi),\sigma)$$

est un calcul de A_{av} sur $\mathsf{enc}(t)$. Nous pouvons donc conclure que $\Phi_{\mathsf{av}}(\rho,\psi,\sigma)$ est un calcul de A_{av} sur $\mathsf{enc}(t)$. □

Lemme 2.3.15

Soit A un AA sur un alphabet (Σ, ar) et $t = a(t|_1, \ldots, t|_m) \in \mathcal{T}_\Sigma$ un arbre. Pour tout calcul $\rho' : (\psi, \sigma) \xrightarrow{\langle a} ((\tau, 0), \sigma(\psi, a)) \xrightarrow{\mathsf{enc}(t|_1)\cdots\mathsf{enc}(t|_m)}{}^*((\tau, m-1), \sigma(\psi, a)) \xrightarrow{a\rangle} (\mathsf{suiv}(\psi), \sigma)$ de A, il existe un calcul $\rho = \tau(\rho|_1, \ldots, \rho|_m)$ de $\mathsf{APV2AA}(A)$ sur t tel que $\Phi_{\mathsf{av}}(\rho, \sigma) = \rho'$.

Démonstration. Nous notons $A = (Q, I, F, \Gamma, \delta)$ et $\mathsf{APV2AA}(A) = A_{\mathsf{va}} = (Q', I', \delta')$, et $\sigma' = \sigma(\psi, a)$. Nous faisons la preuve par induction sur la structure de t. Pour tout $i \in [1, l]$ nous avons :

$$(\psi_{i-1}, \sigma') \xrightarrow{\mathsf{enc}(t|_i)}{}^*(\psi_i, \sigma') \in \mathsf{Cal}_{\mathsf{enc}(t|_i)}(A_{\mathsf{av}})$$

avec $\psi_i = (\tau, i)$. Comme $\mathsf{enc}(t|_i) \in \hat{\Sigma}^*_{\mathsf{cr}}$ par le Lemme 2.3.11 il existe $\tau_i \in \delta$, $n_i \in \mathbb{N}$ et $a_i \in \Sigma_{n_i}$ tels que nous avons $t_i = a_i(t_1|_1, \ldots, t_i|_{n_i})$, $\psi_i \in \mathsf{anc}(\tau_i)$ et :

$$(\psi_{i-1}, \sigma') \xrightarrow{\langle a_i} ((\tau_i, 0), \sigma'(\psi_i, a_i)) \xrightarrow{\mathsf{enc}(t_i|_1)\cdots\mathsf{enc}(t_i|_{n_i})}{}^*((\tau_i, n_i), \sigma'(\psi_i, a_i)) \xrightarrow{a_i\rangle} (\psi_i, \sigma') \in \mathsf{Cal}_{\mathsf{enc}(t|_i)}(A_{\mathsf{av}})$$

Nous pouvons donc appliquer l'hypothèse d'induction et montrer que pour tout $i \in [1, l]$ il existe un calcul $\rho|_i$ de A sur $t|_i$. Enfin, par définition de δ', nous avons :

$$(\psi, \langle a, (\psi, a), (\tau, 0)) \in \delta'_c \qquad ((\tau, l), (\psi, a), a\rangle, \mathsf{suiv}(\psi)) \in \delta'_r$$

Nous en déduisons qu'il existe $\tau \in \delta$ telle que $\tau(\rho|_1, \ldots \rho|_l)$ est un calcul de A sur t. □

Lemme 2.3.16

Soient A un AA sur un alphabet (Σ, ar) et $t = a(t|_1, \ldots, t|_m) \in \mathcal{T}_\Sigma$ un arbre. Pour tout calculs $\rho = \tau(\rho|_1, \ldots, \rho|_m)$,$\rho' = \tau'(\rho'|_1, \ldots, \rho'|_m)$ de A sur t, toute paire $\psi \in \mathsf{anc}(\tau)$ si $\Phi_{\mathsf{av}}(\rho, \psi, \sigma) = \Phi_{\mathsf{av}}(\rho', \psi, \sigma)$ alors $\rho = \rho'$.

Démonstration. Nous notons $A = (Q, I, F, \Gamma, \delta)$ et $\mathsf{APV2AA}(A) = A_{\mathsf{va}} = (Q', I', \delta')$, et $\sigma' = \sigma(\psi, a)$. Nous faisons la preuve par induction sur la structure de t. Par le Lemme 2.3.12, ρ et ρ' sont sur le mot $\mathsf{enc}(t)$. Les calculs $\Phi_{\mathsf{av}}(\rho, \psi, \sigma) = \Phi_{\mathsf{av}}(\rho', \psi, \sigma)$ se décomposent donc ainsi :

$$(\psi, \sigma) \xrightarrow{\langle a} \Phi_{\mathsf{av}}(\rho_1, (\tau, 0), \sigma(\psi, a)) \cdots \Phi_{\mathsf{av}}(\rho_m, (\tau, m-1), \sigma(\psi, a)) \xrightarrow{a\rangle} (\mathsf{suiv}(\psi), \sigma)$$

Par définition de Φ_{av} pour tout $i \in [1, m]$ nous avons $\Phi_{\mathsf{av}}(\rho|_i, (\tau, i-1), \sigma(\psi, a)) = \Phi_{\mathsf{av}}(\rho'|_i, (\tau, i-1), \sigma(\psi, a))$. Par hypothèse d'induction, nous avons $\rho|_1 = \rho'|_1$. De plus, comme $\Phi_{\mathsf{av}}(\rho, \psi, \sigma) = \Phi_{\mathsf{av}}(\rho', \psi, \sigma)$ nous avons $\tau = \tau'$ et donc $\rho = \rho'$. □

Nous pouvons maintenant combiner ces résultats afin de montrer le Théorème 2.3.10.

Démonstration du Théorème 2.3.10. Les Lemmes 2.3.14, 2.3.15 et 2.3.16 assurent que pour tout arbre $t \in \mathcal{T}_\Sigma$, $\mathsf{ACal}_t(A)$ est bijectif avec $\mathsf{ACal}_{\mathsf{enc}(t)}(A_{\mathsf{av}})$. De plus, par le Lemme 2.3.12, pour tout arbre $u \in \mathcal{L}(A_{\mathsf{av}})$ il existe $t \in \mathcal{T}_\Sigma$ tel que $u = \mathsf{enc}(t)$, ceci assurant la propriété souhaitée sur tous les calculs acceptants de A et A_{av}. □

2.3.3 Extension aux automates à coûts

Nous avons présenté dans les sections précédentes les constructions APV2AA et AA2APV, que nous allons maintenant étendre aux automates à coûts. Pour cela, nous allons expliquer comment les coûts sont ajoutés aux transitions, et montrerons que les coûts calculés respectent la bijection entre les calculs de l'automate d'origine et l'automate produit. Nous demandons à chaque fois que le semi-anneau étudié soit commutatif, car les constructions échangent l'ordre dans lequel les coûts sont lus. Nous étendons dans un premier temps la construction APV2AA.

Définition 2.3.17

Soient $\mathcal{S} = (E, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ un semi-anneau commutatif et $S = (A, \lambda)$ un $\mathcal{S}$-APVbi sur l'alphabet structuré Δ. Le $\mathcal{S}$-AA $\mathsf{APV2AA}(S) = (A', \lambda')$ est défini par :

- $A' = \mathsf{APV2AA}(A) = (Q', I', \delta')$,
- $\lambda'(\tau) = \begin{cases} \lambda((p,a,p')) & \text{si } \tau = ((p,q,\gamma),a,(p',q)), \\ \lambda((p,c,\gamma',p')) \otimes \lambda((q',\gamma',r,q'')) & \text{si } \tau = ((p,p',\gamma),(c,r),(p',q',\gamma'),(q'',q)), \\ \mathbf{1} & \text{si } \tau = ((p,p,\gamma),\top). \end{cases}$

  Pour tout $\tau \in \delta'$.

Chaque couple d'ouvrant/fermant (c,r) est remplacé par un nœud de l'arbre dans A_{va}, c'est pourquoi nous faisons le produit des coûts de c et de r. C'est ici que la commutativité est importante. Nous souhaitons maintenant montrer le théorème suivant :

Théorème 2.3.18

Soit $\mathcal{S}$ un semi-anneau commutatif, $S = (A, \lambda)$ un $\mathcal{S}$-APVbi et $S_{\mathsf{va}} = (A_{\mathsf{va}}, \lambda_{\mathsf{va}})$ l' AA $\mathsf{APV2AA}(S)$. Nous avons :

- $\mathsf{fonction}(S_{\mathsf{va}}) = \{(\mathsf{fcns}(w), e) \mid (w,e) \in \mathsf{fonction}(S)\}$
- Pour tout $w \in \mathcal{L}(A)$ il existe une bijection entre $\mathsf{ACal}_w(A)$ et $\mathsf{ACal}_{\mathsf{fcns}(w)}(A_{\mathsf{va}})$.

Pour montrer ce résultat nous donnons la propriété suivante de la fonction Φ_{va}. De ce résultat découle directement la preuve du Théorème 2.3.18.

Lemme 2.3.19

Soient $\mathcal{S}$ un semi-anneau commutatif et $S = (A, \lambda)$ un $\mathcal{S}$-APVbi sur l'alphabet structuré Δ. Pour tout calcul acceptant ρ de A nous avons $\mathsf{cout}(\Phi_{\mathsf{va}}(\rho, \bot)) = \mathsf{cout}(\rho)$.

Démonstration. Par construction de λ' et la commutativité de $\mathcal{S}$. □

Nous étendons maintenant la construction APV2AA. Le principe est le même, nous expliquons comment les coûts sont calculés puis nous établissons une propriété sur la fonction Φ_{av}.

Définition 2.3.20

Soient $\mathcal{S} = (E, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ un semi-anneau commutatif et $S = (A, \lambda)$ un $\mathcal{S}$-AA sur l'alphabet gradué Σ. Le $\mathcal{S}$-APV $\mathsf{AA2APV}(S) = (A', \lambda')$ est défini par :

- $A' = \mathsf{AA2APV}(A) = (Q', I', F', \delta', \Gamma')$,
- $\lambda'(\tau) = \begin{cases} \lambda(\tau') & \text{si } \tau = (S, \langle a, (S, a), (\tau', 0)), \\ \mathbf{1} & \text{si } \tau = ((\tau', \mathsf{ar}(\tau')), (S, a), a\rangle, \mathsf{suiv}(S)), \end{cases}$

  Pour tout $\tau \in \delta'$.

Chaque nœud d'un arbre lu par A est coupé en deux pour être lu par $\mathsf{AA2APV}(S)$, un ouvrant et un fermant. C'est la raison pour laquelle le coût de l'ouvrant est égal au coût du nœud original et le coût du fermant est égal à l'élément neutre de $\otimes$. À noter que l'inverse aurait aussi été possible. Nous souhaitons maintenant montrer le théorème suivant :

Théorème 2.3.21

Soient $\mathcal{S}$ un semi-anneau commutatif, $S = (A, \lambda)$ un $\mathcal{S}$-AA et $S_{\mathsf{av}} = (A_{\mathsf{av}}, \lambda_{\mathsf{av}})$ l' APVbi $\mathsf{AA2APV}(S)$. Nous avons :

- $\mathsf{fonction}(S_{\mathsf{av}}) = \{(\mathsf{fcns}(t), e) \mid (t, e) \in \mathsf{fonction}(S)\}$,
- Pour tout $t \in \mathcal{L}(A)$ il existe une bijection entre $\mathsf{ACal}_t(A)$ et $\mathsf{ACal}_{\mathsf{enc}(t)}(A_{\mathsf{av}})$.

Pour montrer ce résultat nous donnons la priorité suivante de la fonction Φ_{av}. De ce résultat découle directement la preuve du Théorème 2.3.21.

Lemme 2.3.22

Soient $\mathcal{S}$ un semi-anneau commutatif et $S = (A, \lambda)$ un $\mathcal{S}$-AA sur l'alphabet (Σ, ar). Pour tout calcul acceptant ρ de A et tout $\psi \in \mathsf{Qd}_A$ nous avons $\mathsf{cout}(\Phi_{\mathsf{va}}(\rho, \psi, \perp)) = \mathsf{cout}(\rho)$.

Démonstration. Par construction de λ' et la commutativité de $\mathcal{S}$. □

2.3.4 Extension aux transducteurs

Nous considérons maintenant les transducteurs. Nous allons nous servir de ces résultats au Chapitre 5. pour étendre aux transducteurs à pile visible les résultats de Helmut Seidl concernant les transducteurs d'arbres ([Sei94a]). Cette fois-ci nous ne traiterons pas de la construction AA2APV, car nous n'en avons pas besoin pour le reste de ce manuscrit. Il serait cependant possible de le faire sur une sous-classe des transducteurs d'arbres. En effet dans le cas général, ce modèle peut copier ou changer l'ordre des sous-arbres produits, ce qu'un transducteur à pile visible ne peut pas faire. Il faudrait donc considérer une classe de transducteurs d'arbres sans copies et sans échanges, ce que nous avons fait dans [CFR+13].

Comme un transducteur d'arbres produit des arbres et non des mots, nous allons devoir transformer les sorties des transducteurs à pile visible pour les adapter à ce modèle.

Nous n'allons cependant pas traiter l'intégralité des transducteurs à pile visible, seulement ceux correspondant à une restriction dont nous donnons l'intuition maintenant.

L'automate produit par APV2AA possède trois types de transitions (voir page 40). Si nous souhaitons l'étendre aux transducteurs, il est nécessaire que ces transitions produisent des contextes d'arbre, ce qui n'est pas toujours le cas.

Le premier type de transition correspond aux paires d'ouvrants et fermants en correspondance. Sur un transducteur à pile visible, deux mots w_1 et w_2 sont produits entre et après ces deux transitions. Pour chacune de ces paires, nous devons donc vérifier qu'il est possible de construire un contexte, afin de pouvoir substituer aux variables de celui-ci les arbres $\mathsf{fcns}(w_1)$ et $\mathsf{fcns}(w_2)$. Cela nécessite deux choses : tout d'abord, les mots u et v produits respectivement par l'ouvrant et le fermant doivent produire un mot bien imbriqué, afin que nous puissions appliquer la fonction fcns dessus. Ensuite, nous devons remplacer par des variables deux feuilles de l'arbre produit par fcns (que nous appellerons n_1 et n_2 pour l'exemple), afin que pour tous mots w_1 et w_2 nous ayons $\mathsf{fcns}(uw_1vw_2) = \mathsf{fcns}(uv)[n_1 \leftarrow \mathsf{fcns}(w_1)][n_2 \leftarrow \mathsf{fcns}(w_2)]$. Ceci nous permet de construire un contexte t tel que $\mathsf{fcns}(uw_1vw_2) = t[\mathsf{fcns}(w_1), \mathsf{fcns}(w_2)]$. Cette opération n'est pas toujours réalisable. Par exemple, si nous considérons les mots $u = c$, $v = br$ et $w_1 = w_2 = a$ avec c un ouvrant, r un fermant et a, b des internes alors $\mathsf{fcns}(uv) = (c, r)(b(\top), \top)$ et $\mathsf{fcns}(uw_1vw_2) = (c, r)(a(b(\top)), a(\top))$. Clairement nous ne pouvons pas modifier $\mathsf{fcns}(uv)$ pour obtenir l'égalité souhaitée avec $\mathsf{fcns}(uw_1vw_2)$, car cela nécessiterait de modifier l'intérieur de l'arbre. Nous verrons plus tard que nous pouvons avoir l'égalité entre $\mathsf{fcns}(uw_1vw_2)$ et $\mathsf{fcns}(uv)[n_1 \leftarrow \mathsf{fcns}(w_1)][n_2 \leftarrow \mathsf{fcns}(w_2)]$ si la première lettre de v est un fermant. Nous allons présenter une restriction des transducteurs à pile visible appelée *bien formée* basée sur cette idée.

Le deuxième type de transition correspond aux internes. Cette fois encore, les mots u produits par ces transitions doivent être bien imbriqués. Encore une fois, nous allons remplacer une feuille n de $\mathsf{fcns}(u)$ afin que pour tout mot w nous ayons $\mathsf{fcns}(uw) = \mathsf{fcns}(u)[n \leftarrow \mathsf{fcns}(w)]$.

Enfin, le dernier type de transition correspond aux $\top$, qui vont simplement produire le même symbole $\top$.

Nous présentons maintenant les transducteurs à pile visible bien formés :

Définition 2.3.23

Soit $T = (A, \lambda)$ un TPV de Δ vers Δ' avec $A = (Q, I, F, \delta, \Gamma)$. Nous disons que T est *bien formé* si pour toutes transitions $\tau = (p, a, q) \in \delta_\iota$ et pour toutes transitions $\tau' = (p', c, \gamma, q') \in \delta_c$ et $\tau'' = (q'', \gamma, r, p'') \in \delta_r$ telles que $(q', q'') \in \mathsf{BI}_A$ nous avons :

- $\lambda(\tau) \in \Delta'^*_{\mathsf{bi}}$,
- $\lambda(\tau')\lambda(\tau'') \in \Delta'^*_{\mathsf{bi}}$ et $\lambda(\tau'') = ru$ avec $r \in \Delta_r$ et $u \in \Delta'^*$.

A noter que cette restriction empêche *d'aplatir* le mot de sortie. Aplatir un mot sur un alphabet structuré consiste à supprimer tous les ouvrants et fermants de la sortie, afin de ne garder que les internes. Ici la restriction $\lambda(\tau'') = ru$ implique qu'il reste au moins un ouvrant et un fermant par couple. Ceci est tout à fait logique, les transducteurs d'arbres n'étant pas capable d'effectuer cette opération.

Nous présentons maintenant la fonction $\mathsf{noeudmot} : \Delta^* \to \mathbb{N}^*$. Nous allons l'utiliser afin de calculer les positions où remplacer la feuille par une variable. Pour tout $u \in \Delta^*$, $\mathsf{noeudmot}(u)$ est une position définie comme suit :

$$\mathsf{noeudmot}(u) = \begin{cases} \epsilon & \text{si } u = \epsilon \\ 1.\mathsf{noeudmot}(v) & \text{si } u = av \text{ avec } a \in \Delta_\iota \text{ et } v \in \Delta^*, \\ 1.\mathsf{noeudmot}(v) & \text{si } u = cv \text{ avec } c \in \Delta_c \text{ sans correspondance dans } v \\ 2.\mathsf{noeudmot}(v) & \text{si } u = cwrv \text{ avec } c \in \Delta_c, r \in \Delta_r, w \in \Delta'^*_{\mathsf{bi}}. \end{cases}$$

Soient u et v deux mots tels que uv est bien imbriqué. Si l'on parcourt l'arbre $\mathsf{fcns}(uv)$ en profondeur, alors $\mathsf{noeudmot}(u)$ représente la position correspondant au $|u|+1$-ème noeud rencontré.

Le fait suivant établit les propriétés principales de noeudmot dont nous avons besoin pour construire les sorties du transducteur d'arbre. Ils reposent sur le fait qu'étant donnés deux mots u et v, $\mathsf{noeud}(uv)$ est une feuille étiquetée par $\top$ et si la première lettre de v est un fermant alors $\mathsf{noeudmot}(u)$ est aussi une feuille étiquetée par $\top$. Nous pouvons donc les remplacer par des variables sans supprimer certains noeuds de l'arbre produit par $\mathsf{fcns}(uv)$.

Fait 2.3.24

Soient Δ un alphabet structuré et des mots $uv, w_1, w_2 \in \Delta_{\mathsf{bi}}^*$. Nous avons :

$$\mathsf{fcns}(w_1w_2) = \mathsf{fcns}(w_1)[\mathsf{noeudmot}(w_1) \leftarrow \mathsf{fcns}(w_2)]$$

De plus si $v = rv'$ avec $r \in \Delta_r$ et $v' \in \Delta^*$ alors nous avons :

$$\mathsf{fcns}(uw_1vw_2) = \mathsf{fcns}(uv)[\mathsf{noeudmot}(u) \leftarrow \mathsf{fcns}(w_1)][\mathsf{noeudmot}(uv) \leftarrow \mathsf{fcns}(w_2)]$$

Soit $\mathcal{X}_2 = \{x_1, x_2\}$ un ensemble de variables ordonnées . Pour tous mots $uv, w \in \Delta_{\mathsf{bi}}^*$ nous appelons $\mathsf{con}(u, v)$ le contexte $\mathsf{fcns}(uv)[\mathsf{noeudmot}(u) \leftarrow x_1][\mathsf{noeudmot}(uv) \leftarrow x_2]$ et $\mathsf{con}(w)$ le contexte $\mathsf{fcns}(w)[\mathsf{noeudmot}(w) \leftarrow x_1]$.

Exemple 2.3.25

Soit Δ un alphabet structuré avec $\Delta_c = \{c\}$, $\Delta_r = \{r\}$ et $\Delta_\iota = \{a, b\}$ et $\mathcal{X} = \{x_1, x_2\}$ un ensemble de variables ordonnées. Nous considérons le mot $cacrbr$. Nous avons $t = \mathsf{fcns}(cacrbr) = (c,r)(a((c,r)(\top, b(\top)))), \top)$, $t' = \mathsf{con}(ca, r) = (c,r)(a(x_1), x_2)$, $n_1 = \mathsf{noeudmot}(ca) = 1.1$ et $n_2 = \mathsf{noeudmot}(car) = 2$. Par conséquent, nous avons $t'|_{n_1} = x_1$ et $t'|_{n_2} = x_2$, et donc $t'[\mathsf{fcns}(crb)][\mathsf{fcns}(\epsilon)] = t$. La situation est représentée à la Figure 2.9. À noter que pour $t'' = \mathsf{con}(ca, br) = (c,r)(a(x_1), x_2)$ nous n'avons pas $t'[\mathsf{fcns}(cr)][\mathsf{fcns}(\epsilon)] = t$ car le noeud b a été remplacé par x_1 dans t'' lors de l'application de con. Ceci se produit car le mot br ne commence pas par une lettre de Δ_r.

Nous pouvons maintenant étendre la fonction APV2AA aux biTPV bien formés. Pour cela nous construisons des contextes d'arbres en utilisant la fonction fcns et la fonction noeudmot précédemment définie.

Définition 2.3.26

Soient $T = (A, \lambda)$ un biTPV bien formé sur l'alphabet structuré Δ vers l'alphabet structuré Δ' et $\mathcal{X}_2 = \{x_1, x_2\}$ un ensemble de variables ordonnées. Le TA $\mathsf{APV2AA}(T) = (A', \lambda')$ sur l'alphabet $\tilde{\Delta}$ vers l'alphabet $\tilde{\Delta}'$ est défini par :

- $A' = \mathsf{APV2AA}(A) = (Q', I', \delta')$,
- $\lambda'(\tau) = \begin{cases} \mathsf{con}(\lambda((p,c,\gamma',p')), \lambda((q',\gamma',r,q''))) & \text{si } \tau = ((p,p',\gamma),(c,r),(p',q',\gamma'),(q'',q)), \\ \mathsf{con}(\lambda((p,a,p'))) & \text{si } \tau = ((p,q,\gamma),a,(p',q)), \\ \top & \text{si } \tau = ((p,p,\gamma),\top). \end{cases}$

  Pour tout $\tau \in \delta'$.

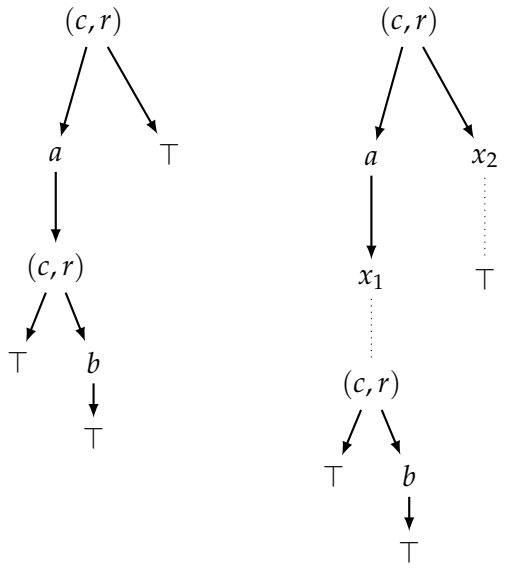


FIGURE 2.9 – Les arbres produits par $\mathsf{fcns}(cacrbr)$ et $\mathsf{con}(ca, r)[\mathsf{fcns}(crb)][\mathsf{fcns}(\epsilon)]$.

Le Fait 2.3.24 implique que pour tout calcul acceptant ρ de A nous avons :

$$\mathsf{fcns}(\mathsf{sortie}(\Phi_{\mathsf{va}}(\rho, \bot))) = \mathsf{sortie}(\rho)$$

Nous pouvons donc en déduire la propriété suivante de la construction APV2AA :

Théorème 2.3.27

Soient $T = (A, \lambda)$ un biTPV bien formé et $T_{\mathsf{va}} = (A_{\mathsf{va}}, \lambda_{\mathsf{va}}) = \mathsf{APV2AA}(T)$. Nous avons :

- $\mathsf{trans}(T_{\mathsf{va}}) = \{(\mathsf{fcns}(w), \mathsf{fcns}(e)) \mid (w, e) \in \mathsf{trans}(T)\}$
- Pour tout $w \in \mathcal{L}(A)$ il existe une bijection entre $\mathsf{ACal}_w(A)$ et $\mathsf{ACal}_{\mathsf{fcns}(w)}(A_{\mathsf{va}})$.

Chapitre 3

Émonder un automate à pile visible

Tout état d'un automate émondé a une utilité, c'est à dire qu'il existe un calcul acceptant le traversant. Une des conséquences importantes de l'émondage est la garantie qu'un calcul peut être toujours complété en un calcul acceptant, et la plupart des opérations algorithmiques traitant de langage ne sont pas intéressantes à exécuter sur la partie non émondée d'un automate. Par exemple, l'appartenance d'un mot à un langage est décidée plus efficacement sur un automate émondé. D'un point de vue théorique, de nombreux résultats reposent sur le fait que les automates finis peuvent être émondés. Par exemple, décider si le coût d'un automate fini à multiplicités est fini repose sur l'utilisation de propriétés pour automates émondés (voir [MS77, DS08]), ce qui sera aussi le cas dans le cas des automates à pile visible, comme nous le verrons aux Chapitres 4 et 5. Bien qu'émonder un automate de mots ou d'arbres soit une chose aisée (voir [Sak09] pour les AF et [CDG$^+$07] pour les AA), nous allons voir qu'émonder un automate à pile visible ne représente pas la même difficulté. En effet, dans le cas des AA et AF il suffit de vérifier l'accessibilité et la co-accessibilité de tous les états, ce qui n'est pas suffisant dans le cas des APV, la pile ajoutant une complexité supplémentaire : il est possible qu'un état soit accessible et co-accessible en considérant une configuration bien particulière, mais soit inaccessible pour toutes les autres.

Il existe plusieurs moyens d'émonder un automate à pile visible hérités d'autres modèles. Il est par exemple possible d'émonder un automate à pile en temps exponentiel en analysant les langages générés par la pile ([Bea84]). Il est aussi possible d'émonder un automate à pile visible en passant par les transducteurs d'arbres. Nous verrons que ces méthodes présentent des inconvénients, et nous allons présenter dans ce chapitre une procédure originale. Premièrement, nous allons étudier le cas des APV bien imbriqués en présentant deux procédures, la réduction et la co-réduction, qui combinées amènent le résultat souhaité. Puis nous étendrons ce résultat aux APV par l'utilisation de deux autres procédures, une transformant un APV en APVbi (que nous émonderons) et l'autre permettant de revenir à un APV. Nous allons montrer que l'APV ainsi obtenu est bien émondé et possède le même langage que celui de départ. Puis, nous présenterons une construction donnant un APV émondé et déterministe, la procédure originale ne préservant pas le déterminisme. Enfin, nous montrerons aussi comment étendre ces résultats aux APV à coûts et aux TPV.

Plan du chapitre À la Section 3.1 nous allons présenter une définition d'émondage des APV prenant en compte la pile. Nous décrirons dans la Section 3.2 d'autres résultats d'émondage sur différents modèles liés aux APV. Nous étudierons une procédure pour émonder un APVbi à la Section 3.3 et nous l'étendrons aux APV généraux à la Section 3.4. Puis nous présenterons une procédure d'émondage des APV déterministes à la Section 3.5, des APV à coûts et des TPV à la Section 3.6. Enfin, nous résumerons les résultats obtenus à la Section 3.7.

3.1 Préliminaires

Dans le cas des automates de mots et d'arbres, un automate émondé est défini comme suit :

Définition 3.1.1

Soit $A = (Q, I, F, \delta)$ un AF sur l'alphabet Σ. A est émondé si et seulement si pour tout état q de Q il existe des mots $u, v \in \Sigma^*$ et un calcul $p \xrightarrow{u}{}^* q \xrightarrow{v}{}^* s \in \mathsf{ACal}_{uv}(A)$.

Définition 3.1.2

Soit $A = (Q, I, \delta)$ un AA sur l'alphabet gradué (Σ, ar). A est émondé si et seulement si pour tout état q de Q il existe $(p, (q))$-calcul et un q-calcul de A avec $p \in I$.

Émonder consiste donc à supprimer les états ne faisant partie d'aucun calcul acceptant. Nous allons donner une définition d'émondage pour automates à pile visible. Dans ce but, nous définissons une caractérisation de l'émondage pour AF et AA utilisant l'accessibilité des états, puis nous l'étendrons aux APV en utilisant l'accessibilité des configurations. Si un automate A est émondé, alors chaque calcul peut-être complété en un calcul acceptant. Autrement dit :

Propriété A est émondé si et seulement si :

Tout état est accessible est co-accessible,

Tout état est co-accessible si est co-accessible,

Tout état est accessible.

En fait, il est assez simple d'émonder un automate fini en temps polynomial, il suffit de réaliser un parcours en profondeur de l'automate depuis les états initiaux et noter les états accessibles, puis de réaliser la même opération depuis les états finaux afin de noter les états co-accessibles. La procédure pour les automates d'arbres fonctionne de manière similaire.

Dans le cas des automates à pile visible, une configuration est co-accessible si elle co-accessible depuis une configuration acceptante, c'est à dire une configuration finale accessible. Il est possible d'atteindre un état avec plusieurs piles différentes : par exemple, étant donné un état p il existe peut-être une configuration (p, σ) faisant partie d'un calcul acceptant et une autre (p, σ') pour laquelle ça n'est pas le cas. Pour cette raison l'émondage doit prendre en compte la pile dans sa définition. Nous donnons maintenant une définition d'émondage des APV utilisant cette idée et la propriété pour les automates d'arbres et les automates finis.

Définition 3.1.3

Soit A un APV. Nous considérons les trois conditions suivantes :

1. Toute configuration accessible est co-accessible,
2. Toute configuration co-accessible est accessible,
3. Pour tout état p de A il existe une configuration (p, σ) accessible.

Nous dirons que A est *émondé* si les Conditions 1, 2 et 3 sont vraies, *faiblement réduit* si la Condition 1 est vraie, *faiblement co-réduit* si la Condition 2 est vraie, *réduit* si les Conditions 1 et 3 sont vraies et *co-réduit* si les Conditions 2 et 3 sont vraies.

Nous ne pouvons pas appliquer la procédure d'émondage pour les automates d'arbres et les automates finis. Le problème est le suivant : lorsqu'un symbole est empilé, nous ne savons pas s'il sera possible de le dépiler par la suite. S'il existe une transition pour le faire, alors il est nécessaire de suivre le bon chemin menant à elle, sous peine d'arriver à une impasse. Dans la construction que nous allons présenter, quand un symbole est empilé une transition de dépilement est choisie (de manière non-déterministe si elles sont multiples) en se basant sur deux critères. Le premier, le symbole empilé doit être le même que celui dépilé, ceci étant simple à vérifier. Ensuite, il faut vérifier que la transition peut-être atteinte en ayant lu un mot bien imbriqué, ceci assurant que la transition dépilera bien le bon symbole. Pour cela nous utiliserons l'ensemble BI_A défini à la Définition 1.4.9.

Nous allons reformuler la propriété de co-réduction dans le cas des APVbi. En effet, dans le cas général, un calcul peut-être accepté alors que la pile contient encore des symboles, ce qui arrive quand un mot n'étant pas bien imbriqué est accepté. Dans le cas des APVbi, nous avons l'assurance que la pile est vide lorsqu'un état final est atteint, ceci impliquant qu'il n'y a qu'une seule configuration acceptante par état final. Ceci nous permet de définir une notion de co-réduction symétrique à la réduction, synthétisée comme suit :

Proposition 3.1.4

> Soit A un APVbi tel que pour tout final f est accessible. Alors A est *faiblement co-réduit* si toute configuration co-accessible depuis la configuration $(f, \bot)$ avec f final est accessible.

Démonstration. Par la remarque 1.4.2 et l'hypothèse sur A, l'ensemble des configurations finales accessibles de A est $\{(f, \bot) \mid f \text{ est un état final}\}$. □

3.2 État de l'art

Automates d'arbres Il est très simple d'émonder un automate d'arbres, et nous avons montré à la Section 2.3 que ce modèle est équivalent aux automates à pile visible. Il serait donc possible de transformer un APV en AA, d'émonder le résultat, puis de revernir vers un APV. Cette méthode pose problème car les transformations que nous avons proposés ne préservent pas le langage de l'APV initial et ne traite que le cas bien imbriqué. Nous pourrions aussi utiliser les constructions proposées dans [AM04], qui traitent du cas général et préservent le langage. Cependant elles ne préservent pas les calculs et en ajoutent de nouveaux menant à des impasses. Dans tous les cas nous ne pouvons pas utiliser directement les constructions existantes, et il est nécessaire de les modifier afin de préserver l'émondage. Nous verrons cependant que la construction utilisée pour transformer un APV en un AA est similaire à la construction permettant de réduire un APV, ceci expliquant pourquoi il est plus facile d'émonder un automate d'arbre qu'un automate à pile visible : en construisant un AA nous réduisons l'APV en entrée.

Automates à pile Il existe un résultat de réduction sur les automates à pile, présenté par Danièle Girault-Beauquier dans [Bea84]. Nous allons présenter ici ce résultat sur les APV.

Définition 3.2.1

Soient $\hat{A} = (\hat{Q}, \hat{I}, \hat{F}, \hat{\Gamma}, \hat{\delta})$ un APV sur l'alphabet structuré Δ et $A = (Q, I, F, \delta)$ un AF déterministe sur l'alphabet $\hat{\Gamma}$. La construction $\mathsf{canon}(\hat{A}, A)$ produit $\bar{A} = (\bar{Q}, \bar{I}, \bar{F}, \bar{\Gamma}, \bar{\delta})$ l'APV sur l'alphabet structuré Δ tel que :

- $\bar{Q} = \hat{Q} \times Q$
- $\bar{I} = \hat{I} \times I$
- $\bar{F} = \hat{F} \times Q$
- $\bar{\Gamma} = Q \times Q \times \hat{\Gamma}$
- $\bar{\delta} = \bar{\delta}_c \cup \bar{\delta}_r \cup \bar{\delta}_r^{\perp} \cup \bar{\delta}_\iota$ avec :
  - $\bar{\delta}_c = \{((p,q), c, (q,q',\gamma), (p',q')) \mid (p,c,\gamma,p') \in \hat{\delta}_c \text{ et } (q,\gamma,q') \in \delta\}$
  - $\bar{\delta}_r = \{((p,q'), (q,q',\gamma), r, (p',q)) \mid (p,\gamma,r,p') \in \hat{\delta}_r \text{ et } (q,\gamma,q') \in \delta\}$
  - $\bar{\delta}_r^{\perp} = \{((p,q), \perp, r, (p',q)) \mid (p,\perp,r,p') \in \hat{\delta}_r^{\perp} \text{ et } q \in Q\}$
  - $\bar{\delta}_\iota = \{((p,q), a, (p',q)) \mid (p,a,p') \in \hat{\delta}_\iota \text{ et } q \in Q\}$

Cette construction permet de modifier le comportement de la pile, comme statué dans la proposition suivante :

Proposition 3.2.2

Soient $\hat{A} = (\hat{Q}, \hat{I}, \hat{F}, \hat{\Gamma}, \hat{\delta})$ un APV sur l'alphabet structuré Δ et $A = (Q, I, F, \delta)$ un AF déterministe sur l'alphabet $\hat{\Gamma}$. Pour tout calcul $((i,i'), \perp) \xrightarrow{u} ((p,q), \sigma)$ de $\mathsf{canon}(\hat{A}, A)$ avec $(i,i') \in \hat{I} \times I$, si $p \in F$ alors $\sigma \in \mathcal{L}(A)$.

Cette propriété est garantie par la construction $\mathsf{canon}(\hat{A}, A)$ en simulant en parallèle un calcul sur $\hat{A}$ et un calcul sur A, assurant ainsi que les empilements et dépilements de $\hat{A}$ sont autorisés par A. Notez qu'empiler un symbole sur $\bar{A}$ fait avancer le calcul sur A, et dépiler le fait reculer. Pour mémoriser la progression du calcul de A, un état supplémentaire est mémorisé dans $\bar{A}$, et nous stockons dans la pile l'état cible et source de la transition de δ afin de pouvoir remonter la transition lorsque le symbole sera dépilé.

Nous allons utiliser cette construction et la Proposition 1.4.13 pour construire un APV réduit. En effet, nous savons que pour tout état $q \in \hat{Q}$, il existe un AF $\mathsf{acc}_{\hat{A}}^q$ tel que :

$$\mathcal{L}(\mathsf{acc}_{\hat{A}}^q) = \{\sigma \mid (q,\sigma) \text{ est accessible et co-accessible dans } \hat{A}\}$$

C'est à dire, $\mathsf{acc}_{\hat{A}}^q$ définit les seules configuration utiles : toute configuration (q,σ) telle que $\sigma \notin \mathsf{acc}_{\hat{A}}^q$ ne fait partie d'aucun calcul acceptant de $\hat{A}$, nous ne voulons donc pas la conserver, et la construction canon nous réalise ce comportement. Nous utilisons $\mathsf{acc}_{\hat{A}}^q$ pour filtrer les piles interdites, assurant que les configurations rencontrées sont accessibles et co-accessibles.

Théorème 3.2.3

Soit $\hat{A} = (\hat{Q}, \hat{I}, \hat{F}, \hat{\Gamma}, \hat{\delta})$ un APV, et $\bar{A}$ l'APV $\mathsf{canon}(\hat{A}, A)$ avec $A = \Pi_{q \in \hat{Q}}(\mathsf{acc}_{\hat{A}}^q)$. Nous avons $\mathcal{L}(\hat{A}) = \mathcal{L}(\bar{A})$ et $\bar{A}$ est réduit.

Pour tout état q, $\mathsf{acc}^q_{\hat{A}}$ permet de filtrer les configurations de l'état q de $\hat{A}$. En considérant le produit pour tout $q \in \hat{Q}$, nous sommes sûrs que les configurations de tous les états sont filtrées, assurant ainsi la réduction de $\bar{A}$. Cette construction repose sur la Définition 3.2.1, utilisant un AF déterministe. Il est donc nécessaire de déterminiser A, amenant une complexité exponentielle à la procédure complète.

Le déterminisme de A permet à canon de considérer qu'il n'y a qu'un seul calcul pour tout préfixe de mot de pile de $\mathcal{L}(A)$, $\bar{A}$ ne doit donc simuler qu'un seul calcul de A à la fois. Si A n'est plus déterministe, il peut y avoir des calculs différents associés à un même mot, et il faut tous les simuler afin d'assurer que $\bar{A}$ est correctement construit. Ceci est une construction par sous-ensemble, menant à un complexité exponentielle. Dans tous les cas il semble difficile de modifier directement canon afin d'obtenir une procédure polynomiale.

Automates de mots imbriqués Les automates de mots imbriqués (*nested word automata* ou NWA) ont été présentés pour la première fois en 2009 dans [AM09], et sont des automates lisant des mots avec une relation de parenthésage sur les positions du mot, par exemple $abdbefa$ avec la relation $\{(-\infty,7)(1,6),(2,4)\}$ décrite à la Figure 3.1. Les positions $-\infty$ (resp. $+\infty$) sont utilisées afin de simuler des positions en relation avec une position située avant (resp. après) le mot. Chaque position ne peut apparaître qu'une seule fois dans la relation, dans chaque couple l'élément de gauche est toujours plus petit que l'élément de droite et il ne peut pas y avoir de croisements (par exemple, $(1,3)$ et $(2,4)$ ne peuvent pas apparaître ensemble dans une même relation). Cette notion de relation est donc très similaire aux mots imbriqués, dans le sens qu'une lettre apparaissant à gauche dans la relation serait un ouvrant, à droite un fermant, et n'apparaissant pas un interne. Sur l'exemple précédent les positions 1 et 2 seraient des ouvrants, les positions 7, 4 et 6 des fermants et les positions 3 et 5 des internes. Le mot imbriqué correspondant à cette relation serait donc par exemple $\langle a\langle bdb\rangle ef\rangle a\rangle$, les lettres de la forme $\langle\alpha$ définissant un ouvrant et celles de la forme $\alpha\rangle$ un fermant.

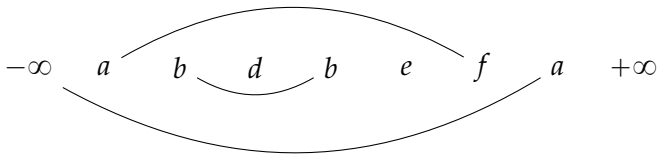


FIGURE 3.1 – Le mot $abdbefa$ et la relation $\{(-\infty,7)(1,6),(2,4)\}$.

Un NWA est un automate de mots possédant trois types de transitions appelées *ouvrantes*, *fermantes* et *internes*. Les transitions du type ouvrante ou fermante sont similaires aux transitions ouvrantes et fermantes des APV dans le sens qu'elles disposent d'un élément supplémentaire appelé *état hiérarchique*. Chaque transition ouvrante émet un *état hiérarchique*, et chaque transition fermante en réceptionne un. Lorsqu'un NWA lit un mot u sur une relation $\{(i_1,j_1),\ldots,(i_n,j_n)\}$, une transition ouvrante est utilisée à la position i_k et une transition fermante est utilisée à la position j_k. La contrainte supplémentaire est sur l'état hiérarchique, il doit être le même à l'émission et à la réception. Encore une fois cette notion est très similaire aux APV, les états hiérarchiques remplaçant les symboles de pile. Lorsqu'un symbole de pile est empilé sur un APV, un état hiérarchique est émit sur un NWA, et quand un symbole de pile est dépilé sur un APV, un état hiérarchique est réceptionné sur un NWA. Il existe donc une transformation assez évidente permettant de construire un APV depuis un NWA et inversement (voir [AM09]), et il est assez naturel de se demander si les problèmes étudiés sur les NWA apportent des solutions aux problèmes étudiés sur les APV.

La question abordée par Rajeev Alur et Parthasarathy Madhusudan est celle de la clôture par préfixe sur les NWA. Étant donné un automate A sur un alphabet Σ reconnaissant le langage L, peut-on construire un automate B tel que $\mathcal{L}(B) = \{u \in \Sigma^* \mid \text{ il existe } v \in \Sigma^* \text{ tel que } uv \in L\}$? Pour résoudre ce problème, il suffit de construire un automate faiblement réduit et de rendre tous les états finaux. En effet, un calcul commencé depuis un état initial dans un automate réduit pourra toujours rejoindre un état final. Chaque mot lu depuis un état initial est donc un préfixe de mot accepté par l'automate. En rendant finaux tous les états, tous les préfixes du langage seront acceptés. Nous expliquons maintenant la construction en se basant sur les APV.

Définition 3.2.4

Soit $A = (Q, I, F, \Gamma, \delta)$ un APV sur l'alphabet structuré Δ. La construction $\mathsf{prefix}(A)$ produit $B = (Q', I', F', \Gamma, \delta')$ l'APV sur l'alphabet structuré Δ tel que :

$$
\begin{aligned}
\circ\ Q' &= \left\{ (p,q,3) \;\middle|\; \text{il existe } u \in \Delta_{\mathsf{bi}}^* \text{ tel que } (p,\bot) \xrightarrow{u}{}^*\, (q,\bot) \in \mathsf{Cal}_u(A) \right\} \cup \\
&\quad \left\{ (p,q,2) \;\middle|\; \begin{array}{l} \text{il existe } u \in \Delta^* \text{ tel que } (p,\bot) \xrightarrow{u}{}^*\, (q,\sigma) \in \mathsf{Cal}_u(A) \\ \text{et } u \neq vrw \text{ avec } v \in \Delta_{\mathsf{bi}}^*, w \in \Delta^* \text{ et } r \in \Delta_r \end{array} \right\} \cup \\
&\quad \left\{ (p,q,1) \;\middle|\; \text{il existe } u \in \Delta^* \text{ tel que } (p,\bot) \xrightarrow{u}{}^*\, (q,\sigma) \in \mathsf{Cal}_u(A) \right\} \\
\circ\ I' &= I \times F \times \{1\} \\
\circ\ F' &= Q \times Q \times [3] \\
\circ\ \Gamma' &= Q \times Q \times [3] \times \Gamma \\
\circ\ \delta' &= \delta'_c \cup \delta'_r \cup \delta'^{\bot}_r \cup \delta'_\iota \text{ avec :} \\
&\quad -\ \delta'_c = \left\{ ((p,q,n), c, (q,n,\gamma), (p',q',n')) \;\middle|\; \begin{array}{l} (p,c,\gamma,p') \in \delta_c, \\ (q',\gamma,r,s) \in \delta_r, \\ (s,q), (p',q') \in \mathsf{BI}_A, \\ n' \in \{2,3\} \text{ si } n \in \{1,2\}, \\ n = 3 \text{ si } n' = 3 \end{array} \right\} \\
&\quad -\ \delta'_r = \left\{ ((p,p,3), (q',n',\gamma), r, (p',q',n')) \;\middle|\; (p,\gamma,r,p') \in -\delta_r \right\} \\
&\quad -\ \delta'^{\bot}_r = \left\{ ((p,q,1), \bot, r, (p',q',1)) \;\middle|\; (p,\bot,r,p') \in -\delta^{\bot}_r \right\} \\
&\quad -\ \delta'_\iota = \left\{ ((p,q,n), a, (p',q',n)) \;\middle|\; (p,a,p') \in \delta_\iota \right\}
\end{aligned}
$$

Cette construction produit un APV B où les états sont des triplets (p,q,n) avec p,q des états de A et n un entier tels qu'il existe un calcul de p à q sur un mot u, la forme de u dépendant de n. Si $n = 3$ alors u est bien imbriqué, si $n = 2$ alors u peut contenir des ouvrants sans correspondance, et si $n = 1$ alors u peut aussi contenir des fermants sans correspondance. Le deuxième état permet de retenir quel est l'état à atteindre afin de terminer le calcul, ceci assurant la réduction de B. L'entier permet de s'assurer que les transitions traversées correspondent au mot lu, par exemple si $n = 2$ on ne veut pas lire de transition fermante correspondant au symbole ouvrant ayant provoqué le passage à 2.

Il est possible d'adapter cette construction (en modifiant les états finaux pour ne conserver que les états de $F \times F \times [3]$)) afin de réduire un APV. Cependant, comme dans le cas général la co-réduction n'est pas symétrique avec la réduction, il n'est pas possible de l'utiliser afin d'émonder complètement un APV. Nous allons cependant voir que la procédure présentée dans les prochaines sections coïncide avec cette construction en plusieurs aspects.

3.3 Émonder un automate à pile visible bien imbriqué

Dans cette section, étant donné un APVbi A, nous allons présenter la construction du APVbi $\mathsf{emonder}_{\mathsf{bi}}(A)$, reconnaissant le même langage que A et étant émondé. Premièrement nous allons définir l' APVbi réduit $\mathsf{reduire}(A)$ équivalent à A. Ensuite nous construirons $\mathsf{coreduire}(A)$ et enfin nous combinerons ces deux constructions afin de produire $\mathsf{emonder}_{\mathsf{bi}}(A)$. La construction $\mathsf{coreduire}(A)$ préservant la réduction de A, nous obtiendrons ainsi un APV émondé.

3.3.1 Construction de $\mathsf{freduire}(A)$ et $\mathsf{reduire}(A)$

Étant donné un automate à pile visible bien imbriqué A, nous allons décrire la construction du APV $\mathsf{freduire}(A)$ faiblement réduit. L'APV $\mathsf{reduire}(A)$ est ensuite obtenu en supprimant tous les états inaccessibles. Pour cela pour chaque état p de A nous calculons le langage de pile de p et nous testons sa vacuité, qui est une propriété décidable en temps polynomial ([AM04]). Si le langage est vide, cela signifie qu'il n'existe pas de configuration (p, σ) accessible, et nous pouvons donc supprimer p. Par le Lemme 1.4.13 concernant les langages de pile nous obtenons ainsi une procédure exécutable en temps polynomial pour obtenir un automate à pile visible réduit depuis un automate à pile visible faiblement réduit.

Définition 3.3.1

Soit $A = (Q, I, F, \Gamma, \delta)$ un APVbi sur l'alphabet structuré Δ, nous définissons l' APVbi $\mathsf{freduire}(A) = (Q', I', F', \Gamma', \delta')$ sur Δ, avec :

$$\circ\ Q' = \mathsf{BI}_A \qquad I' = \mathsf{BI}_A \cap (I \times F) \qquad F' = \{(f,f) \mid f \in F\} \qquad \Gamma' = \Gamma \times Q$$

$$\circ\ \delta' = \delta'_c \cup \delta'_r \cup \delta'_\iota \text{ avec :}$$

$$- \ \delta'_c = \left\{ ((p,q), c, (\gamma,q), (p',q')) \;\middle|\; \begin{array}{l}(p,q),(p',q') \in Q', (p,c,\gamma,p') \in \delta_c, \\ (q',\gamma,r,s) \in \delta_r \text{ et } (s,q) \in Q'\end{array} \right\}$$

$$- \ \delta'_r = \{ ((q',q'),(\gamma,q),r,(p,q)) \mid (p,q) \in Q', (q',r,\gamma,p) \in \delta_r \}$$

$$- \ \delta'_\iota = \{ ((p,q),a,(p',q)) \mid (p,q),(p',q) \in Q', (p,a,p') \in \delta_\iota \}$$

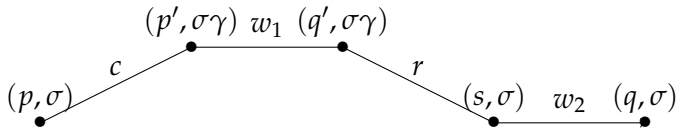


FIGURE 3.2 – Construction des transitions d'empilement avec w_1 et w_2 des mots bien imbriqués.

Intuitivement, les états (et la pile) de $\mathsf{freduire}(A)$ étendent ceux de A avec un état supplémentaire de A. Ce composant est utilisé par $\mathsf{freduire}(A)$ lors de la simulation d'un calcul de A, pour stocker l'état que le calcul doit atteindre avant de dépiler le symbole au sommet de la pile. Les états sources des transitions de dépilement sont de la forme (q,q) pour assurer que l'état cible est atteint lorsque le symbole est dépilé. Pour obtenir un APV faiblement réduit il doit exister pour chaque transition d'empilement une transition de dépilement permettant d'atteindre l'état cible. Cette condition est décrite dans la Figure 3.2.

Exemple 3.3.2

Soit A l' APVbi sur l'alphabet structuré Δ et A_{red} l' APVbi reduire(A) représentés à la Figure 3.3. Nous considérons le calcul suivant de A :

$$\rho = (1, \bot) \xrightarrow{c} (2, \gamma) \xrightarrow{c_2} (2, \gamma\gamma_2) \in \mathsf{Cal}_{cc_2}(A)$$

Ce calcul ne peut pas être complété en un calcul acceptant. En effet, le seul moyen de rejoindre un état final est de dépiler γ, mais pour cela il faut dépiler γ_2 et aller à l'état 3, état impossible à quitter car il aurait fallu empiler γ_1 en premier. De manière générale, un symbole γ_1 doit se trouver sur la pile avant d'empiler γ_2, ce qui n'est pas le cas pour le mot cc_2. Par conséquent, il n'y a pas d'équivalent au calcul ρ dans reduire(A).

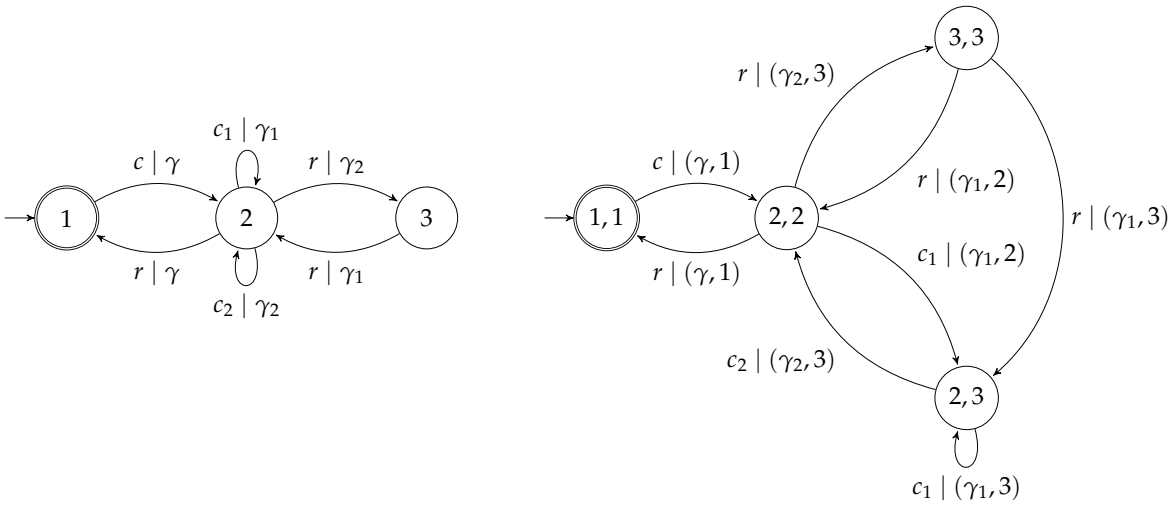


FIGURE 3.3 – À gauche l'APV A, à droite reduire(A)

Nous allons maintenant montrer que les APVbi construits par freduire respectent certaines bonnes propriétés. Le Lemme 3.3.3 établit que l'état supplémentaire mémorisé afin de se souvenir de l'endroit où dépiler reste le même après avoir lu un mot bien imbriqué. C'est un lemme technique que nous utiliserons pour les preuves des lemmes suivants. Nous montrerons ensuite que la construction induit une bijection entre les calculs acceptants de l'APV en entrée et l'APV réduit (et donc une égalité de langage) dans les Lemmes 3.3.5 et 3.3.7, résultat synthétisé dans le Théorème 3.3.9. Enfin, le Théorème 3.3.10 établit que freduire est bien faiblement réduit, et la Proposition 3.3.11 établit que la construction préserve la co-réduction.

Lemme 3.3.3

Soit A un APVbi sur l'alphabet structuré Δ et A_{fred} l' APVbi freduire(A). Pour tout $w \in \Delta^*_{\mathsf{bi}}$ et tout $\rho' : ((p, q), \bot) \xrightarrow{w}{}^* ((p', q'), \bot) \in \mathsf{Cal}(A_{\mathsf{fred}})$, nous avons $q = q'$.

Démonstration. Nous notons $A = (Q, I, F, \Gamma, \delta)$ et $A_{\mathsf{fred}} = (Q', I', F', \Gamma', \delta')$. Nous prouvons ce résultat par induction sur la structure de w. Le résultat est évident quand $w = \varepsilon$. Nous supposons maintenant que $|w| > 0$, il y a deux cas :

- Si $w = aw_1$, avec $a \in \Delta_\iota$, $w_1 \in \Delta_{\mathsf{bi}}^*$ alors nous pouvons décomposer le calcul ρ' comme suit :

$$\rho' : ((p,q),\bot) \xrightarrow{a} ((p_1,q_1),\bot) \xrightarrow{w_1} {}^*((p',q'),\bot)$$

  Par hypothèse d'induction appliquée sur le suffixe de ρ' sur w_1, nous avons $q_1 = q'$. Comme $a \in \Delta_\iota$, et par définition de δ'_ι, nous avons $q = q_1$.

- Si $w = cw_1rw_2$, avec $c \in \Delta_c$, $r \in \Delta_r$ et $w_1, w_2 \in \Delta_{\mathsf{bi}}^*$ alors nous pouvons décomposer ρ' selon la décomposition de w. Nous avons :

$$\rho' : ((p,q),\bot) \xrightarrow{c} ((p_1,q_1),\gamma') \xrightarrow{w_1} {}^*((p'_1,q'_1),\gamma') \xrightarrow{r} ((p'',q''),\bot) \xrightarrow{w_2} {}^*((p',q'),\bot)$$

  Par hypothèse d'induction appliquée sur w_1 et w_2 nous obtenons $q_1 = q'_1$ et $q'' = q'$. Par définition des transitions de δ'_r nous avons $p'_1 = q'_1$ et $\gamma' = (\gamma, q')$ pour $\gamma \in \Gamma$. La transition d'empilement utilisée sur c empile le symbole γ' sur la pile. Par la définition des transitions de δ'_c, nous déduisons que $q = q'$. □

Soit A un APVbi et A_{fred} l' APVbi freduire(A). Nous allons maintenant définir une fonction liant les calculs de A avec ceux de A_{fred}. Le but de cette fonction est d'établir comment les calculs de A_{fred} sont produits à partir de A. Soit Φ_{fred} une fonction de $\mathsf{Cal}(A_{\mathsf{fred}})$ vers $\mathsf{Cal}(A)$ et $\rho' \in \mathsf{Cal}(A_{\mathsf{fred}})$ un calcul défini comme suit :

$$\rho' : ((p_0,q_0),\sigma_0) \xrightarrow{a_1} ((p_1,q_1),\sigma_1) \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} ((p_{n-1},q_{n-1}),\sigma_{n-1}) \xrightarrow{a_n} ((p_n,q_n),\sigma_n)$$

Nous définissons $\Phi_{\mathsf{fred}}(\rho') \in \mathsf{Cal}(A)$ comme étant le calcul suivant :

$$\Phi_{\mathsf{fred}}(\rho') : (p_0,\pi_1(\sigma_0)) \xrightarrow{a_1} (p_1,\pi_1(\sigma_1)) \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} (p_{n-1},\pi_1(\sigma_{n-1})) \xrightarrow{a_n} (p_n,\pi_1(\sigma_n))$$

Nous rappelons que la fonction π_1 est la projection sur le premier élément du couple, et ici nous l'étendons naturellement aux séquences de couples. Nous allons maintenant montrer que $\Phi_{\mathsf{fred}}(\rho')$ est bien un calcul de A, puis nous montrerons que restreinte aux calculs acceptants de A_{fred} la fonction Φ_{fred} est bijective.

Lemme 3.3.4

> Soit A un APVbi sur l'alphabet structuré Δ et A_{fred} l' APVbi freduire(A). Pour tout mot $w \in \Delta_{\mathsf{bi}}^*$ et tout calcul $\rho \in \mathsf{Cal}_w(A_{\mathsf{fred}})$, nous avons $\Phi_{\mathsf{fred}}(\rho) \in \mathsf{Cal}_w(A)$.

Démonstration. Nous notons $A = (Q, I, F, \Gamma, \delta)$ et $A_{\mathsf{fred}} = (Q', I', F', \Gamma', \delta')$. Nous prouvons ce résultat par induction sur la structure de w. Le résultat est évident quand $w = \varepsilon$. Nous supposons maintenant que $|w| > 0$, il y a deux cas :

- Si $w = aw_1$ avec $a \in \Delta_\iota$, $w_1 \in \Delta_{\mathsf{bi}}^*$ alors nous pouvons décomposer ρ' de la manière suivante :

$$\rho' : ((p_1,q_1),\sigma') \xrightarrow{a} ((p_2,q_2),\sigma') \xrightarrow{w_1} {}^*((p_3,q_3),\sigma')$$

  Le résultat est évident en appliquant l'hypothèse d'induction sur le suffixe de ρ' sur w_1, et en utilisant la définition de δ'_ι, assurant que $(p_1, a, p_2) \in \delta_\iota$.

- Si $w = cw_1rw_2$ avec $c \in \Delta_c$, $r \in \Delta_r$ et $w_1, w_2 \in \Delta_{\mathsf{bi}}^*$ alors nous pouvons décomposer ρ' comme suit :

$$\rho' : ((p,q),\sigma') \xrightarrow{c} ((p_1,q_1),\gamma') \xrightarrow{w_1} {}^*((p'_1,q'_1),\sigma'\gamma') \xrightarrow{r} ((p'',q''),\sigma') \xrightarrow{w_2} {}^*((p',q'),\sigma')$$

  Nous pouvons premièrement appliquer l'hypothèse d'induction sur les mots w_1 et w_2. Ensuite, l'existence de transitions d'empilement et de dépilement dans A_{fred} assure l'existence des transitions suivantes dans A : $(p, c, \pi_1(\gamma'), p_1) \in \delta_c$ et $(p'_1, \pi_1(\gamma'), r, p'') \in \delta_r$. Ceci nous permet donc d'affirmer l'existence du calcul de A souhaité. □

Les lemmes suivants prouvent l'injectivité et la surjectivité de Φ_{fred}. Cela nous servira pour montrer que la construction freduire produit une bijection sur les calculs acceptants.

Lemme 3.3.5

Soit A un APVbi sur l'alphabet structuré Δ et A_{fred} l' APVbi freduire(A). Pour tout mot $w \in \Delta^*_{\mathsf{bi}}$, s'il existe un calcul $\rho : (p, \bot) \xrightarrow{w}{}^*(q, \bot)$ dans $\mathsf{Cal}(A)$, alors il existe un calcul $\rho' : ((p, q), \bot) \xrightarrow{w}{}^*((q, q), \bot) \in \mathsf{Cal}(A_{\mathsf{fred}})$ tel que $\Phi_{\mathsf{fred}}(\rho') = \rho$.

Démonstration. Nous notons $A = (Q, I, F, \Gamma, \delta)$ et $A_{\mathsf{fred}} = (Q', I', F', \Gamma', \delta')$. Nous prouvons ce résultat par induction sur la structure de w. Le résultat est évident si $w = \varepsilon$. Nous supposons maintenant que $|w| > 0$, il y a deux cas :

- Si $w = aw_1$ avec $a \in \Delta_\iota$, $w_1 \in \Delta^*_{\mathsf{bi}}$ alors nous pouvons décomposer le calcul ρ de la manière suivante :
$$\rho : (p, \bot) \xrightarrow{a} (p_1, \bot) \xrightarrow{w_1}{}^* (q, \bot)$$
Par hypothèse d'induction appliquée sur le suffixe ρ_1 de ρ sur w_1, il existe un calcul ρ'_1 de A' sur w_1 de la forme $\rho'_1 : ((p_1, q), \bot) \xrightarrow{w_1} ((q, q), \bot)$, tel que $\Phi_{\mathsf{fred}}(\rho'_1) = \rho_1$. De plus, par l'existence de ρ, nous avons $(p, q), (p_1, q) \in \mathsf{BI}$. Par définition de δ'_ι, Nous pouvons en déduire l'existence du calcul $((p, q), \bot) \xrightarrow{a} ((p_1, q), \bot)$ de A', ceci concluant ce cas.
- Si $w = cw_1rw_2$, avec $c \in \Delta_c$, $r \in \Delta_r$ et $w_1, w_2 \in \Delta^*_{\mathsf{bi}}$ alors nous pouvons décomposer le calcul ρ comme suit :
$$\rho : (p, \bot) \xrightarrow{c} (p_1, \gamma) \xrightarrow{w_1}{}^* (q_1, \gamma) \xrightarrow{r} (p_2, \bot) \xrightarrow{w_2}{}^* (q, \bot)$$
Nous appelons ρ_1 (resp. ρ_2) la restriction sur le calcul ρ sur le mot w_1 (resp. w_2). Par hypothèse d'induction appliquée sur w_1 et w_2, il existe des calculs ρ'_1 et ρ'_2 de la forme suivante :
$$\rho'_1 : ((p_1, q_1), \bot) \xrightarrow{w_1}{}^* ((q_1, q_1), \bot)$$
$$\rho'_2 : ((p_2, q), \bot) \xrightarrow{w_2}{}^* ((q, q), \bot)$$
tels que $\Phi_{\mathsf{fred}}(\rho'_1) = \rho_1$ et $\Phi_{\mathsf{fred}}(\rho'_2) = \rho_2$. Par l'existence de ρ il existe deux transitions $(p, c, \gamma, p_1) \in \delta_c$ et $(q_1, \gamma, r, p_2) \in \delta_r$. Nous avons donc $((p, q), c, (\gamma, q), (p_1, q_1)) \in \delta'_c$ et $((q_1, q_1), (\gamma, q), r, (p_2, q)) \in \delta'_r$ par construction de δ' car $(p, q) \in \mathsf{BI}_A$. Ceci nous permet donc d'affirmer l'existence du calcul de A_{fred} souhaité. □

Corollaire 3.3.6

Soit A un APVbi sur l'alphabet structuré Δ et A_{fred} l' APVbi freduire(A). Restreint sur $\mathsf{ACal}(A_{\mathsf{fred}})$, Φ_{fred} est surjective vers $\mathsf{ACal}(A)$.

Démonstration. C'est une conséquence directe du Lemme 3.3.5 : par définition de A_{fred}, tout calcul acceptant de A_{fred} est de la forme $((p, q), \bot) \xrightarrow{w}{}^* ((q, q), \bot) \in \mathsf{ACal}_w(A_{\mathsf{fred}})$ avec p un état initial de A et q un état final de A. Par conséquent, la fonction Φ_{fred} restreinte sur $\mathsf{ACal}(A_{\mathsf{fred}})$ est bien surjective vers $\mathsf{ACal}(A)$. □

Lemme 3.3.7

Soit A un APVbi sur l'alphabet structuré Δ et A_{fred} l' APVbi freduire(A). Pour tout mot $w \in \Delta_{\mathsf{bi}}^*$ et tous calculs $\rho'_1, \rho'_2 \in \mathsf{Cal}_w(A_{\mathsf{fred}})$, si $\Phi_{\mathsf{fred}}(\rho'_1) = \Phi_{\mathsf{fred}}(\rho'_2)$ et dernier$(\rho'_1) =$ dernier(ρ'_2), alors $\rho'_1 = \rho'_2$.

Démonstration. Nous notons $A = (Q, I, F, \Gamma, \delta)$ et $A_{\mathsf{fred}} = (Q', I', F', \Gamma', \delta')$. Nous prouvons ce résultat par induction sur la structure de w. Ce résultat est évident quand $w = \varepsilon$. Nous supposons maintenant que $|w| > 0$, il y a deux cas :

- Si $w = aw_1$ avec $a \in \Delta_\iota$, $w_1 \in \Delta_{\mathsf{bi}}^*$ alors par le Lemme 3.3.3, nous pouvons décomposer les calculs ρ'_1 et ρ'_2 comme suit : (nous utilisons l'hypothèse $\Phi_{\mathsf{fred}}(\rho'_1) = \Phi_{\mathsf{fred}}(\rho'_2)$)

  $$\forall i \in \{1,2\}, \rho'_i : ((p, q_i), \bot) \xrightarrow{a} ((p', q_i), \bot) \xrightarrow{w_1}{}^* ((p'', q_i), \bot)$$

  Nous pouvons appliquer l'hypothèse d'induction sur les suffixes de ρ'_1 et ρ'_2 sur w_1 et w_2, comme nous avons dernier$(\rho'_i) =$ dernier(ρ'_i) pour $i \in [1,2]$, impliquant que $q_1 = q_2$. Nous pouvons en conclure que $\rho'_1 = \rho'_2$.
- Si $w = cw_1rw_2$ avec $c \in \Delta_c$, $r \in \Delta_r$ et $w_1, w_2 \in \Delta_{\mathsf{bi}}^*$ alors par le Lemme 3.3.3 et le fait que $\Phi_{\mathsf{fred}}(\rho'_1) = \Phi_{\mathsf{fred}}(\rho'_2)$, pour tout $i \in [1,2]$ nous pouvons décomposer les calculs ρ'_1 et ρ'_2 comme suit :

  $$\rho'_i : ((p, q_i), \bot) \xrightarrow{c} ((p', q'_i), (\gamma, q_i)) \xrightarrow{w_1}{}^* ((p'', q'_i), (\gamma, q_i)) \xrightarrow{r} ((s, q_i), \bot) \xrightarrow{w_2}{}^* ((s', q_i), \bot)$$

  Premièrement, l'hypothèse dernier$(\rho'_1) =$ dernier(ρ'_2) assure que l'on puisse appliquer l'hypothèse d'induction sur les calculs sur w_2, et déduire en particulier que $q_1 = q_2 = q$).

  Ceci implique que les symboles (γ, q_1) et (γ, q_2) sont les mêmes. De plus, par la forme des transitions de δ'_r, nous pouvons conclure que $p'' = q'_i$ pour $i \in \{1,2\}$. Ces deux calculs ont donc le même dernier état, et par le Lemme 1.4.7 nous pouvons appliquer l'hypothèse d'induction. □

Corollaire 3.3.8

Soit A un APVbi sur l'alphabet structuré Δ et A_{fred} l' APVbi freduire(A). Restreint sur $\mathsf{ACal}(A_{\mathsf{fred}})$, la fonction Φ_{fred} est injective.

Démonstration. C'est un corollaire du Lemme 3.3.7 puisque les dernières configurations des calculs acceptants de A_{fred} sont de la forme $((q,q), \bot)$ avec q un état final. Donc, si deux calculs acceptants ρ'_1, ρ'_2 ont la même image image par Φ_{fred}, ils doivent vérifier que dernier$(\rho'_1) =$ dernier$(\rho'_2) = (q, \bot)$ avec q un état final. De plus par définition des états initiaux et finaux de A_{fred}, tout calcul acceptant de A_{fred} est de la forme $((p,q), \bot) \xrightarrow{w}{}^* ((q,q), \bot) \in \mathsf{ACal}_w(A_{\mathsf{fred}})$ avec p un état initial de A et q un état final de A. Par conséquent, restreint sur $\mathsf{ACal}(A_{\mathsf{fred}})$, la fonction Φ_{fred} est injective. □

Le théorème suivant est une conséquence directe du Lemme 3.3.4 et des Corollaires 3.3.6 et 3.3.8.

Théorème 3.3.9

Soient A un APVbi sur l'alphabet structuré Δ , $A_{\mathsf{fred}} = \mathsf{freduire}(A)$ et $A_{\mathsf{red}} = \mathsf{reduire}(A)$. A_{fred} et A_{red} sont construits en temps polynomial. De plus pour tout mot $w \in \Delta^*_{\mathsf{bi}}$, il existe une bijection entre $\mathsf{ACal}_w(A_{\mathsf{fred}})$ et $\mathsf{ACal}_w(A)$, et une bijection entre $\mathsf{ACal}_w(A_{\mathsf{red}})$ et $\mathsf{ACal}_w(A)$, et donc en particulier nous avons $\mathcal{L}(A) = \mathcal{L}(A_{\mathsf{fred}}) = \mathcal{L}(A_{\mathsf{red}})$.

Dans le cas de A_{red}, il existe une bijection entre $\mathsf{ACal}_w(A_{\mathsf{red}})$ et $\mathsf{ACal}_w(A)$ car nous avons produit A_{red} depuis A_{fred} en supprimant les états inaccessibles, par conséquent des états n'étant pas traversés par un calcul acceptant. La bijection entre $\mathsf{ACal}_w(A_{\mathsf{fred}})$ et $\mathsf{ACal}_w(A)$ est donc préservée.

Nous allons maintenant nous intéresser aux propriétés du APVbi que la construction freduire produit. Bien évidemment, nous allons montrer qu'elle produit un APVbi faiblement réduit, car c'est le but de la construction. Nous allons aussi montrer qu'elle préserve la co-réduction de l' APVbi original. Ceci nous sera utile pour émonder l' APVbi. Dans la section suivante, nous allons présenter une construction pour co-réduire, que nous allons pouvoir combiner avec freduire grâce à sa préservation de la co-réduction.

Théorème 3.3.10

Soient A un APVbi sur l'alphabet structuré Δ , $A_{\mathsf{fred}} = \mathsf{freduire}(A)$ et $A_{\mathsf{red}} = \mathsf{reduire}(A)$. A_{fred} est faiblement réduit et A_{red} est réduit.

Démonstration. Nous notons $A = (Q, I, F, \Gamma, \delta)$ et $A_{\mathsf{fred}} = (Q', I', F', \Gamma', \delta')$. Nous allons prouver que A_{fred} est faiblement réduit en montrant que toute configuration accessible de A_{fred} est aussi co-accessible depuis une configuration finale. Comme A_{red} est obtenu en supprimant les états non accessibles de A_{fred}, nous pouvons conclure que A_{red} est aussi réduit.

Soit $\kappa = ((p,q),\sigma)$ une configuration accessible de A_{fred}. Il existe donc un mot $w \in \Delta^*$ et un calcul sur w de A_{fred} de la forme $((i,f),\bot) \xrightarrow{w}{}^*\kappa$ entre la configuration initiale $((i,f),\bot)$ et κ. Nous allons montrer par induction sur la taille de la pile σ que nous pouvons atteindre une configuration finale depuis κ.

Si $|\sigma| = 0$, par le Lemme 3.3.3, nous obtenons $q = f$. En particulier, ceci implique que $q \in F$. Comme $(p,q) \in \mathsf{BI}_A$, il existe un mot $w' \in \Delta^*$ et un calcul $(p,\bot) \xrightarrow{w'}{}^*(q,\bot)$ sur w' de A, et par le Lemme 3.3.5 il existe un calcul $((p,q),\bot) \xrightarrow{w'}{}^*((q,q),\bot)$ de A_{fred}. Ceci conclut ce cas car $q \in F$, et donc nous avons $(q,q) \in F'$.

Soit $n \in \mathbb{N}$ un entier. Nous supposons maintenant que la propriété est vraie quand $|\sigma| \leqslant n$ et nous montrons qu'elle est aussi vraie quand $|\sigma| = n+1$. Nous notons par (γ, q') le symbole au sommet de la pile σ, c'est à dire $\sigma = \sigma'.(\gamma,q')$, et nous examinons la dernière position dans le calcul ρ empilant ce symbole. Nous dénotons par c la lettre d'appel associée. Plus précisément, il existe deux mots $u \in \Delta^*$, $v \in \Delta^*_{\mathsf{bi}}$ et une décomposition de ρ comme suit :

$$\rho : ((i,f),\bot) \xrightarrow{u}{}^*((p',q'),\sigma') \xrightarrow{c} ((p'',q''),\sigma) \xrightarrow{v}{}^*((p,q),\sigma)$$

Par le Lemme 3.3.5, nous obtenons $q'' = q$. Considérant la transition d'appel associée à c, et par définition de δ'_c, il existe une transition $((q,q),(\gamma,q'),r,(s,q')) \in \delta'_r$ avec $r \in \Delta_r$. De plus, comme $(p,q) \in \mathsf{BI}$, il existe un mot $w' \in \Delta^*$ et un calcul $(p,\bot) \to^* (q,\bot)$ sur w' de A. Par le Lemme 3.3.3 nous avons un calcul $((p,q),\bot) \xrightarrow{w'}{}^*((q,q),\bot)$ de A_{fred}, et donc par le Lemme 1.4.5 un calcul $((p,q),\sigma) \to^* ((q,q),\sigma)$ de A'.

Comme $\sigma = \sigma'.(\gamma, q')$ la transition de dépilement définie au-dessus peut-être utilisée pour atteindre la configuration $((s, q'), \sigma')$ telle que $|\sigma'| = n$. Le résultat suit par hypothèse d'induction. □

Proposition 3.3.11

Soit A un APVbi sur l'alphabet structuré Δ. Si A est co-réduit, alors $\mathsf{reduire}(A)$ est également co-réduit.

Démonstration. Nous notons $A = (Q, I, F, \Gamma, \delta)$ et $A_{\mathsf{fred}} = (Q', I', F', \Gamma', \delta')$. Nous allons prouver que si A est co-réduit, alors A_{fred} est aussi co-réduit. Comme $\mathsf{reduire}(A)$ est obtenu en supprimant les états non accessibles de A_{fred}, nous pouvons conclure que $\mathsf{reduire}(A)$ est aussi co-réduit.

Soit $\kappa = ((p, q), \sigma)$ et $\kappa_i = ((p_i, q_i), \bot)$ deux configurations de A_{fred} telles que (p_i, q_i) est un état initial de A_{fred} et il existe deux mots $w, w' \in \Delta^*$ et deux calculs $\rho : \kappa \xrightarrow{w}{}^*((f, f), \sigma')$ sur w et $\kappa_i \xrightarrow{w'}{}^*((f, f), \sigma')$ sur w' de A_{fred} où (f, f) est un état final de A_{fred}. Comme A_{fred} est un APVbi et par la Remarque 1.4.2, nous avons $\sigma' = \bot$. Nous allons montrer par induction sur la taille de la pile σ que κ peut-être atteint depuis un état initial.

Si $|\sigma| = 0$, alors par le Lemme 3.3.3, nous avons $q = f$. Par construction de l'ensemble BI_A, $(p, \bot) \xrightarrow{w}{}^*(f, \bot)$ est un calcul de A. Comme A est co-réduit, il existe un calcul $(i, \bot) \xrightarrow{u}{}^*(p, \bot)$ de A avec $u \in \Delta^*$ et $i \in I$. Donc par les Lemmes 3.3.5 et 3.3.3 nous concluons que :

$$((i, f), \bot) \xrightarrow{u}{}^*((p, f), \bot) \xrightarrow{w}{}^*((f, f), \bot) \in \mathsf{Cal}(A_{\mathsf{fred}})$$

Soit $n \in \mathbb{N}$ un entier. Nous supposons maintenant que la propriété est vraie quand $|\sigma| \leqslant n$ et nous montrons qu'elle est aussi vraie quand $|\sigma| = n + 1$. Nous allons examiner la dernière position dans le calcul ρ dépilant le symbole au sommet de la pile. Nous notons r la lettre associée à la transition. Plus précisément, il existe deux mots $u \in \Delta_{\mathsf{bi}}^*$, $v \in \Delta^*$ et une décomposition de ρ telle que :

$$((p, q), \sigma) \xrightarrow{u}{}^*((q'', q''), \sigma) \xrightarrow{r}((p', q'), \sigma') \xrightarrow{v}{}^*((f, f), \bot) \in \mathsf{Cal}(A_{\mathsf{fred}})$$

Avec $\sigma = \sigma'(\gamma, q')$ et $\gamma \in \Gamma$. Par le Lemme 3.3.3 nous avons $q'' = q$ et par le Lemme 3.3.7, la projection de ρ est un calcul dans A :

$$(p, \bar{\sigma}) \xrightarrow{u}{}^*(q, \bar{\sigma}) \xrightarrow{r}(p', \bar{\sigma}') \xrightarrow{v}{}^*(f, \bot) \in \mathsf{Cal}(A)$$

avec $\bar{\sigma} = \pi_1(\sigma)$ et $\bar{\sigma}' = \pi_1(\sigma')$. Comme A est co-réduit, la configuration $(p, \bar{\sigma})$ est accessible en utilisant un calcul ρ_1. Nous décomposons ρ_1 afin d'exhiber la transition ayant empilé le dernier symbole de $\bar{\sigma}$:

$$\rho_1 : (i, \bot) \xrightarrow{v'}{}^*(p'', \bar{\sigma}') \xrightarrow{c}(s, \bar{\sigma}) \xrightarrow{u'} (p, \bar{\sigma}) \in \mathsf{Cal}(A)$$

Avec $c \in \Delta_c$, $v' \in \Delta^*$ et $u' \in \Delta_{\mathsf{bi}}^*$. Par les Lemmes 3.3.5 et 3.3.3, nous pouvons déduire que :

$$((s, q), \bot) \xrightarrow{u'}{}^*((p, q), \bot) \xrightarrow{u}{}^*((q, q), \bot) \in \mathsf{Cal}(A_{\mathsf{fred}})$$

et donc par le Lemme 1.4.5 :

$$((s, q), \sigma) \xrightarrow{u'}{}^*((p, q), \sigma) \xrightarrow{u}{}^*((q, q), \sigma) \in \mathsf{Cal}(A_{\mathsf{fred}})$$

Nous considérons maintenant la transition d'empilement sur c utilisée dans le calcul ρ_1, elle est de la forme $t = (p'', c, \gamma, s)$. Par l'existence du calcul ci-dessus dans A_{fred}, nous déduisons l'existence de la transition $((p'', q'), c, (\gamma, q'), (s, q))$ de δ'_c. Nous pouvons étendre le calcul ci-dessus A_{fred} de la manière suivante :

$$((p'', q'), \sigma') \xrightarrow{c} ((s, q), \sigma) \xrightarrow{u'}{}^{*} ((p, q), \sigma) \xrightarrow{u}{}^{*} ((q, q), \sigma) \xrightarrow{rv}{}^{*} ((f, f), \bot) \in \mathsf{Cal}(A_{\mathsf{fred}})$$

Et le résultat suit par hypothèse d'induction car $|\sigma'| = n$. □

Nous avons montré dans cette section que freduire produit un APVbi faiblement réduit et reduire produit un APVbi réduit. De plus elles induisent une bijection sur les calculs acceptants entre l' APVbi construit et le l' APVbi initial, préservant donc le langage. Enfin nous avons montré qu'elles préservent la co-réduction. Dans la section suivante, nous allons prouver des résultats similaires sur la co-réduction, et nous allons les combiner afin d'émonder un APVbi.

3.3.2 Construction de coreduire(A)et fcoreduire(A)

Étant donné un APVbi A, nous décrivons la construction du APV fcoreduire(A), qui est faiblement co-réduit. L'APV coreduire(A) est alors obtenu de manière similaire à reduire(A).

Définition 3.3.12

Soit $A = (Q, I, F, \Gamma, \delta)$ un APVbi sur l'alphabet structuré Δ, nous définissons l' APVbi fcoreduire$(A) = (Q', I', F', \Gamma', \delta')$ sur l'alphabet structuré Δ avec :

$$\begin{array}{lll}
\circ\ Q' & = & \mathsf{BI}_A \qquad I' = \{(i,i) \mid i \in I\}) \qquad F' = \mathsf{BI}_A \cap (I \times F) \qquad \Gamma' = \Gamma \times Q \\
\circ\ \delta' & = & \delta'_c \cup \delta'_r \cup \delta'_\iota \text{ avec :} \\
\quad - \ \delta'_c & = & \left\{ \ ((p,q), c, (\gamma, p), (q', q')) \ \middle| \ (p,q) \in Q', (q, c, \gamma, q') \in \delta_c \ \right\} \\
\quad - \ \delta'_r & = & \left\{ \ ((p,q), (\gamma, p'), r, (p', q')) \ \middle| \ \begin{array}{l} (p,q), (p',q') \in Q', (q, \gamma, r, q') \in \delta_r, \\ (s, c, \gamma, p) \in \delta_c \text{ et } (p', s) \in Q' \end{array} \right\} \\
\quad - \ \delta'_\iota & = & \left\{ \ ((p,q), a, (p, q')) \ \middle| \ (p,q), (p,q') \in Q', (q, a, q') \in \delta_\iota \ \right\}
\end{array}$$

De la même manière que pour freduire, les états (et la pile) de fcoreduire(A) étendent ceux de A avec un état additionnel de A. Cette information est utilisée pour mémoriser l'état q atteint juste avant que le symbole au sommet de la pile ait été empilé. Pour obtenir APVbi faiblement co-réduit il doit exister pour chaque transition de dépilement une transition d'empilement permettant de passer par l'état source q. Par la Proposition 3.1.4, nous savons que la propriété de réduction est symétrique à la propriété de co-réduction, et il en va de même pour les constructions associées : là où freduire mémorise un état supplémentaire pour pouvoir avancer, fcoreduire mémorise un état pour pouvoir reculer.

Pour prouver que cette construction produit bien ce que l'on souhaite, nous allons présenter une deuxième construction, dual, produisant le dual d'un APVbi.

Définition 3.3.13

Soit $A = (Q, I, F, \Gamma, \delta)$ un APVbi sur l'alphabet structuré Δ, nous définissons l' APVbi $\mathsf{dual}(A) = (Q', I', F', \Gamma', \delta')$ sur l'alphabet structuré Δ avec :

$$
\begin{array}{lllll}
\circ\ Q' & = & Q \qquad I' = F \qquad F' = I \qquad \Gamma' = \Gamma & & \\
\circ\ \delta' & = & \delta'_c \cup \delta'_r \cup \delta'_\iota \text{ avec :} & & \\
\quad - \ \delta'_c & = & \{ \ (q, r, \gamma, p) & \mid & (p, \gamma, r, q) \in \delta_r \ \} \\
\quad - \ \delta'_r & = & \{ \ (q, \gamma, c, p) & \mid & (p, c, \gamma, q) \in \delta_c \ \} \\
\quad - \ \delta'_\iota & = & \{ \ (q, a, p) & \mid & (p, a, q) \in \delta_\iota \ \}
\end{array}
$$

Cette construction permet de reconnaître le miroir du langage d'un APVbi, c'est à dire que nous avons :

Lemme 3.3.14

Soit A un APVbi sur l'alphabet structuré Δ. Pour tout $a_1 \dots a_n \in \Delta^*$ nous avons :

$$(p, \sigma) \xrightarrow{a_1 \dots a_n}{}^* (p', \sigma') \in \mathsf{Cal}_{a_1 \dots a_n}(A) \Leftrightarrow (p', \sigma') \xrightarrow{a_n \dots a_1}{}^* (p, \sigma) \in \mathsf{Cal}_{a_n \dots a_1}(\mathsf{dual}(A))$$

De plus elle possède une propriété intéressante liée à freduire et fcoreduire : Comme la construction freduire est symétrique à la construction fcoreduire, appliquer fcoreduire est la même chose qu'appliquer dual puis freduire, et ré-appliquer dual. Nous allons maintenant montrer comment fcoreduire dual $\circ$ freduire $\circ$ dual sont reliées.

Lemme 3.3.15

Soit A un APVbi sur l'alphabet structuré Δ, $A_{\mathsf{fcor}} = \mathsf{fcoreduire}(A)$ et $A_{\mathsf{dual}} = \mathsf{dual} \circ \mathsf{freduire} \circ \mathsf{dual}(A)$. Pour tout mot $w \in \Delta^*_{\mathsf{bi}}$ sur w nous avons :

$$((p, q), \sigma) \xrightarrow{w}{}^* ((p', q'), \sigma') \in \mathsf{Cal}_w(A_{\mathsf{fcor}}) \Leftrightarrow ((q, p), \sigma) \xrightarrow{w}{}^* ((q', p'), \sigma') \in \mathsf{Cal}_w(A_{\mathsf{dual}})$$

Le couple d'état de A est inversé dans A_{dual} car fcoreduire mémorise l'état supplémentaire dans la première composante, alors que freduire le mémorise dans la deuxième. Les symboles de piles sont cependant les mêmes. Nous disposons aussi de la propriété suivante, venant directement de la symétrie des constructions freduire et fcoreduire :

Lemme 3.3.16

Soit A un APVbi sur l'alphabet structuré Δ. Si A est réduit (resp. co-réduit) alors $\mathsf{dual}(A)$ est co-réduit (resp. réduit). Si A est faiblement réduit (resp. faiblement co-réduit) alors $\mathsf{dual}(A)$ est faiblement co-réduit (resp. faiblement réduit).

Soient A un APVbi et A_{fcor} le APVbi $\mathsf{fcoreduire}(A)$. Nous allons maintenant définir une fonction liant les calculs de A avec ceux de A_{fcor}. Le but de cette fonction est d'établir comment les

calculs de A_{fcor} sont produits à partir de A. Soit Φ_{fcor} une fonction de $\mathsf{Cal}(A_{\mathsf{fcor}})$ vers $\mathsf{Cal}(A)$ et $\rho' \in \mathsf{Cal}(A_{\mathsf{fcor}})$ un calcul défini comme suit :

$$\rho' : ((p_0, q_0), \sigma_0) \xrightarrow{a_1} ((p_1, q_1), \sigma_1) \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} ((p_{n-1}, q_{n-1}), \sigma_{n-1}) \xrightarrow{a_n} ((p_n, q_n), \sigma_n)$$

Nous définissons $\Phi_{\mathsf{fcor}}(\rho') \in \mathsf{Cal}(A)$ comme étant le calcul suivant :

$$\Phi_{\mathsf{fcor}}(\rho') : (q_0, \pi_1(\sigma_0)) \xrightarrow{a_1} (q_1, \pi_1(\sigma_1)) \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} (q_{n-1}, \pi_1(\sigma_{n-1})) \xrightarrow{a_n} (q_n, \pi_1(\sigma_n))$$

Nous pouvons maintenant prouver les résultats nous intéressant en utilisant dual. Le Théorème 3.3.17 statut que la fonction Φ_{fcor} induit une bijection entre les calculs des APVbi. Nous montrerons ensuite dans le Théorème 3.3.18 que fcoreduire produit des APVbi faiblement co-réduit et coreduire produit des APVbi co-réduit. Enfin, nous montrerons que fcoreduire préserve la réduction et le déterminisme dans les Lemmes 3.3.11 et 3.3.20. Nous nous servirons du Lemme 3.3.20 à la Section 3.5 pour montrer que l'on peut construire un APV à la fois déterministe et émondé.

Théorème 3.3.17

> Soient A un APVbi sur l'alphabet structuré Δ, A_{fcor} le APVbi $\mathsf{fcoreduire}(A)$ et A_{cor} le APVbi $\mathsf{coreduire}(A)$. A_{fcor} et A_{cor} sont construits en temps polynomial. De plus pour tout mot $w \in \Delta^*_{\mathsf{bi}}$, la fonction Φ_{fcor} induit une bijection entre $\mathsf{ACal}_w(A_{\mathsf{fcor}})$ et $\mathsf{ACal}_w(A)$, et une bijection entre $\mathsf{ACal}_w(A_{\mathsf{cor}})$ et $\mathsf{ACal}_w(A)$, et donc en particulier nous avons $\mathcal{L}(A) = \mathcal{L}(A_{\mathsf{fcor}}) = \mathcal{L}(A_{\mathsf{cor}})$.

Démonstration. Soit A_{dual} l' APVbi produit par $\mathsf{dual} \circ \mathsf{freduire} \circ \mathsf{dual}(A)$. Par les Lemmes 3.3.14, 3.3.3 et 3.3.5 nous avons que tout calcul $((p_0, q_0), \sigma_0) \xrightarrow{a_1} ((p_1, q_1), \sigma_1) \xrightarrow{a_2} \cdots \xrightarrow{a_n} ((p_n, q_n), \sigma_n)$ est un calcul acceptant de A_{dual} si et seulement si $(p_0, \pi_1(\sigma_0)) \xrightarrow{a_1} (p_1, \pi_1(\sigma_1)) \xrightarrow{a_2} \ldots \xrightarrow{a_n} (p_n, \pi_1(\sigma_n))$ est un calcul de A. Par le Lemme 3.3.15 nous pouvons en déduire que Φ_{fcor} satisfait la propriété souhaitée. Comme A_{cor} est obtenu en supprimant les états non accessibles de A_{fred}, nous concluons qu'il existe aussi une bijection entre $\mathsf{ACal}_w(A_{\mathsf{cor}})$ et $\mathsf{ACal}_w(A)$ □

Théorème 3.3.18

> Soient A un APVbi sur l'alphabet structuré Δ, A_{fcor} le APVbi $\mathsf{fcoreduire}(A)$ et A_{cor} le APVbi $\mathsf{coreduire}(A)$. A_{fcor} est faiblement co-réduit et A_{cor} est co-réduit.

Démonstration. Par le Théorème 3.3.10, $\mathsf{freduire} \circ \mathsf{dual}(A)$ est faiblement réduit. Donc par le Lemme 3.3.16, $\mathsf{dual} \circ \mathsf{freduire} \circ \mathsf{dual}(A)$ est faiblement co-réduit. Par le Lemme 3.3.15 nous avons donc que A_{fcor} est faiblement co-réduit. De plus, comme A_{cor} est obtenu en supprimant les état non accessibles de A_{fcor}, nous concluons que A_{cor} est co-réduit. □

Proposition 3.3.19

> Soit A un APVbi réduit sur l'alphabet structuré Δ, alors $\mathsf{coreduire}(A)$ est aussi réduit.

Démonstration. Par le Lemme 3.3.16, nous savons que si A est réduit, alors $\mathsf{dual}(A)$ est co-réduit. Par la Proposition 3.3.11, $\mathsf{freduire} \circ \mathsf{dual}(A)$ est aussi co-réduit. Par le Lemme 3.3.16 nous déduisons que $\mathsf{dual} \circ \mathsf{freduire} \circ \mathsf{dual}(A)$ est réduit. Par le Lemme 3.3.15 nous pouvons en conclure que $\mathsf{coreduire}(A)$ l'est aussi. □

Proposition 3.3.20

Soit A un APVbi déterministe sur l'alphabet structuré Δ, alors $A_{\mathsf{fcor}} = \mathsf{fcoreduire}(A)$ et $A_{\mathsf{cor}} = \mathsf{coreduire}(A)$ sont aussi déterministes.

Démonstration. Nous allons montrer ce résultat sur A_{fcor}. Comme A_{cor} est obtenu en supprimant les états inutiles de A_{fcor}, nous pouvons conclure que A_{cor} est aussi déterministe. Premièrement, par définition nous avons $I' = \{(p,p) \mid p \in I\}$, et donc $\#I' = \#I = 1$. Deuxièmement, nous considérons les transitions de A_{fcor}, et analysons la manière dont elles sont construites. Soit τ une transition de A_{fcor} menant de l'état (p,q) vers l'état (p',q') sur la lettre a de Δ :

- Si $a \in \Delta_\iota$, alors $\tau = ((p,q),a,(p',q'))$. Par construction de A_{fcor}, $(q,a,q') \in \delta_\iota$. Comme A est déterministe, il n'y a pas de transition $(q,a,q'') \in \delta_\iota$ avec $q'' \neq q'$. Nous concluons qu'il n'existe pas d'état $p'' \in Q$ tel que $((p,q),a,(p'',q'')) \in \delta'_\iota$ avec $(p',q') \neq (p'',q'')$ car par définition de δ'_ι nous avons $p = p' = p''$.
- Si $a \in \Delta_c$, alors $\tau = ((p,q),a,(\gamma,p),(p',q'))$ avec $\gamma \in \Gamma$. Par construction de A_{fcor} nous avons $(q,a,\gamma,q') \in \delta_c$. Comme A est déterministe, il n'y a pas de transition $(q,a,\gamma',q'') \in \delta_c$ avec $q'' \neq q'$ ou $\gamma' \neq \gamma$. Nous pouvons donc en conclure qu'il n'existe pas d'états $p'', p''' \in Q$ tels que $((p,q),(\gamma',p'''),a,(p'',q'')) \in \delta'_r$ avec $(p',q') \neq (p'',q'')$ ou $(\gamma,p) \neq (\gamma',p''')$ car par définition de δ'_c nous avons $p = p'''$, $p' = q'$ et $p'' = q''$.
- Si $a \in \Delta_r$, alors $\tau = ((p,q),(\gamma,p'),a,(p',q'))$ avec $\gamma \in \Gamma$. Par construction de A_{fcor} nous avons $(q,\gamma,a,q') \in \delta_r$. Comme A est déterministe, il n'y a pas de transition $(q,\gamma,a,q'') \in \delta_r$ avec $q'' \neq q'$. Nous pouvons donc en conclure qu'il n'existe pas d'état $p'' \in Q$ tel que $((p,q),(\gamma,p'),a,(p'',q'')) \in \delta'_c$ avec $(p',q') \neq (p'',q'')$ car par définition de δ'_r nous avons $p' = p''$. □

3.3.3 La procédure complète

Nous définissons la construction $\mathsf{emonder}_{\mathsf{bi}}$ par $\mathsf{emonder}_{\mathsf{bi}} = \mathsf{coreduire} \circ \mathsf{reduire}$. La Proposition 3.3.19 assure que la construction coreduire préserve la réduction, et donc par les Théorèmes 3.3.9, 3.3.10, 3.3.17 et 3.3.18 nous obtenons le résultat suivant :

Théorème 3.3.21

Soit A un APVbi sur l'alphabet structuré Δ, et A_{emonder} le APVbi $\mathsf{emonder}_{\mathsf{bi}}(A)$. A_{emonder} est émondé et est construit en temps polynomial. De plus, pour tout $w \in \Delta^*$, il existe une bijection entre $\mathsf{ACal}_w(A)$ et $\mathsf{ACal}_w(A_{\mathsf{emonder}})$, et donc $\mathcal{L}(A) = \mathcal{L}(A_{\mathsf{emonder}})$.

3.4 Émonder un automate à pile visible

Nous allons maintenant présenter une construction pour transformer un APV A en un APVbi utilisant un alphabet particulier. Dans une seconde étape, nous émondons l' APVbi en utilisant la méthode décrite dans la Section 3.3 pour les APVbi. Enfin depuis le nouveau APVbi nous construisons un APV utilisant l'alphabet original reconnaissant le langage $\mathcal{L}(A)$ et qui est toujours émondé. Il n'est pas évident de proposer des procédures pour transformer un APV vers un APVbi et inversement, et qui en plus de cela sont compatibles avec les constructions d'émondage.

3.4.1 Des automates à pile visible arbitraires vers les bien imbriqués...

Les mots de Δ^* sont différents des mots bien imbriqués car il permettent des lettres sans correspondances. Nous corrigeons ce problème en étendant l'alphabet Δ et en modifiant les mots de Δ^*. D'un coté, une nouvelle lettre est ajoutée pour fermer les lettres de Δ_c sans correspondance. De l'autre coté, un symbole interne est créé pour remplacer chacune des lettres de Δ_r sans correspondance.

Soit $\Delta = \Delta_c \uplus \Delta_r \uplus \Delta_\iota$ un alphabet structuré Nous introduisons l'alphabet structuré $\Delta^{\mathsf{ext}} = \Delta_c^{\mathsf{ext}} \uplus \Delta_r^{\mathsf{ext}} \uplus \Delta_\iota^{\mathsf{ext}}$ défini par :

$$\Delta_c^{\mathsf{ext}} = \Delta_c \qquad \Delta_r^{\mathsf{ext}} = \Delta_r \uplus \{\bar{r}\} \qquad \Delta_\iota^{\mathsf{ext}} = \Delta_\iota \uplus \{a_r \mid r \in \Delta_r\}$$

avec $\bar{r}$ et $\{a_r \mid r \in \Delta_r\}$ des nouveaux symboles non présents dans Δ.

Nous allons définir de manière inductive la fonction ext transformant un mot de Δ vers un mot de Δ^{ext} comme suit, étant donnés $a \in \Delta_\iota$, $r \in \Delta_r$, $c \in \Delta_c$ et $n \in \mathbb{N}$:

- $\mathsf{ext}(\epsilon, n) = \epsilon$,
- $\mathsf{ext}(aw, n) = a\,\mathsf{ext}(w, n)$,
- $\mathsf{ext}(rw, n) = r\,\mathsf{ext}(w, n-1)$ si $n > 0$,
- $\mathsf{ext}(rw, 0) = a_r\,\mathsf{ext}(w, 0)$,
- $\mathsf{ext}(cw, n) = \begin{cases} cw_1 r\,\mathsf{ext}(w_2, n) & \text{si } \exists w_1 \in \Delta_{\mathsf{bi}}^*, w_2 \in \Delta^*, r \in \Delta_r \text{ et } w = w_1 r w_2, \\ c\,\mathsf{ext}(w, n+1)\,\bar{r} & \text{sinon.} \end{cases}$

Étant donné un mot $w \in \Delta^*$, nous appelons $\mathsf{ext}(w)$ le mot $\mathsf{ext}(w, 0)$. Cette fonction remplace chaque lettre r de Δ_r sans ouvrant par le symbole interne a_r et ajoute un suffixe de la forme $\{\bar{r}\}^*$ dans le but d'ajouter une correspondance pour chaque lettre de Δ_c sans fermant. Il est possible de montrer par induction sur la structure de w que $\mathsf{ext}(w)$ est un mot bien imbriqué sur l'alphabet Δ^{ext}.

Intuitivement, $\mathsf{ext}(w, n)$ produit un mot bien imbriqué le long d'un calcul en prenant en compte la hauteur de la pile via l'entier n.

Exemple 3.4.1

Nous notons w le mot $rrccar$. Nous avons $\mathsf{ext}(w, 1) = r\,\mathsf{ext}(rccar, 0) = ra_r\,\mathsf{ext}(ccar, 0) = ra_r c\,\mathsf{ext}(car, 1)\bar{r} = ra_r ccar\,\mathsf{ext}(\epsilon, 1)\bar{r} = ra_r ccar\bar{r}$.

Nous définissons aussi une variante de ext où le suffixe de $\{\bar{r}\}^*$ n'est pas ajouté, et nous appelons cette fonction ext_ι :

$$\mathsf{ext}_\iota(w, n) = w' \text{ tel que } \mathsf{ext}(w, n) = w'w'' \text{ avec } w' \in (\Delta^{\mathsf{ext}} \backslash \{\bar{r}\})^* \text{ et } w'' \in \{\bar{r}\}^*$$

Étant donné un mot $w \in \Delta^*$, nous définissons aussi $\mathsf{ext}_\iota(w)$ comme étant le résultat de la fonction $\mathsf{ext}_\iota(w, 0)$.

Soit A un APV, nous présentons maintenant la construction etendre transformant un APV sur l'alphabet Δ vers un APVbi sur l'alphabet Δ^{ext}. Intuitivement, cette construction ajoute un suffixe au APV lisant des mots de la forme $\bar{r}^n$ avec $n \in \mathbb{N}_{>0}$ et remplace les transitions dépilant sur pile vide par des transitions sans action sur la pile, implémentant de cette manière la fonction ext. Pour compléter ceci, l'APV doit avoir certaines bonnes propriétés, introduites dans la définition suivante :

Définition 3.4.2

Soit $A = (Q, I, F, \Gamma, \delta)$ un APV sur l'alphabet structuré Δ. Nous dirons que A est *conforme en pile* s'il existe une partition de Q telle que $Q = Q_\perp \uplus Q_{\not\perp}$ et une partition de Γ telle que $\Gamma_\perp \uplus \Gamma_{\not\perp}$ et :

1. Pour tout calcul initialisé de A menant vers une configuration $(p, \sigma), \sigma = \perp \Leftrightarrow p \in Q_\perp$,
2. Pour tout calcul initialisé de A menant vers une configuration $(p, \sigma), \sigma \in \{\perp\} \cup \{\gamma\sigma \mid \gamma \in \Gamma_\perp, \sigma \in \Gamma^*_{\not\perp}\}$,
3. Pour tout $\tau \in \delta_r$, $\mathsf{source}(\tau) \notin F$.

Nous montrerons dans la Sous-section 3.4.3 que pour tout APV il est possible de construire en temps polynomial un APV conforme en pile équivalent. Nous présentons maintenant la construction etendre, opérant sur les APV conforme en pile. Cette construction permet de transformer un APV en un APVbi que nous pourrons émonder avec la procédure définie dans la section précédente.

Définition 3.4.3

Soit $A = (Q, I, F, \Gamma, \delta)$ un APV conforme en pile sur l'alphabet Δ avec les partitions $Q = Q_\perp \uplus Q_{\not\perp}$ et $\Gamma = \Gamma_\perp \uplus \Gamma_{\not\perp}$. L' APVbi $\mathsf{etendre}(A) = (Q', I', F', \Gamma', \delta')$ sur l'alphabet Δ^{ext} est défini par :

$$\circ\ Q' = Q \cup \{\bar{f}_\perp, \bar{f}_{\not\perp}\} \qquad I' = I \qquad F' = (F \cap Q_\perp) \cup \{\bar{f}_\perp\} \qquad \Gamma' = \Gamma$$

$\circ\ \delta' = \delta'_c \cup \delta'_r \cup \delta'_\iota$ avec :

$$\begin{aligned}
-\ \delta'_c &= \delta_c \\
-\ \delta'_r &= \delta_r \cup \{ (p, \bar{r}, \gamma, \bar{f}_\int) \mid \int \in \{\perp, \not\perp\}, p \in F \cap Q_{\not\perp}, \gamma \in \Gamma_\int \} \\
&\quad \cup \{ (\bar{f}_{\not\perp}, \bar{r}, \gamma, \bar{f}_\int) \mid \int \in \{\perp, \not\perp\}, \gamma \in \Gamma_\int \} \\
-\ \delta'_\iota &= \delta_\iota \cup \{ (p, a_r, q) \mid (p, r, \perp, q) \in \delta_r^\perp, p \in Q_\perp \}
\end{aligned}$$

Intuitivement, $\mathsf{etendre}(A)$ possède deux parties : La première, composée des états de A, peut lire des mots de la forme $(\Delta^{\mathsf{ext}} \backslash \{\bar{r}\})^*$ et remplace chaque transition dépilant sur pile vide par une transition ne faisant rien sur la pile, et la deuxième composée de deux nouveaux états $\{\bar{f}_\perp, \bar{f}_{\not\perp}\}$, lit des mots de la forme $\{\bar{r}\}^+$. Cette intuition est formalisée pour n'importe quel APV sur l'alphabet Δ^{ext} par la définition suivante :

Définition 3.4.4

Soient Δ un alphabet structuré et $A = (Q, I, F, \Gamma, \delta)$ un APV sur l'alphabet Δ^{ext}. Nous définissons deux sous-ensembles de Q comme suit :

$\mathsf{piege}(A) = \{q \in Q \mid \exists p, (p, \bar{r}, \gamma, q) \in \delta_r\}$

$\mathsf{bordure}(A) = \{p \notin \mathsf{piege}(A) \mid \exists \tau \in \delta$ tel que $\mathsf{source}(\tau) = p$ et $\mathsf{cible}(\tau) \in \mathsf{piege}(A)\}$

Les états de $\mathsf{bordure}(A)$ sont appelés *états de bord* de A.

Les états de bord sont les états n'appartenant pas à $\mathsf{piege}(A)$ et permettant d'atteindre un état de $\mathsf{piege}(A)$. Il est possible de vérifier que dans le cas du APV $\mathsf{etendre}(A)$, nous avons $\mathsf{piege}(\mathsf{etendre}(A)) = \{\bar{f}_{\not\perp}, \bar{f}_\perp\}$ et $\mathsf{bordure}(\mathsf{etendre}(A)) = F \cap Q_{\not\perp}$.

Exemple 3.4.5

Soit $A = (Q, I, F, \Gamma, \delta)$ un APV sur l'alphabet structuré Δ et A_{ext} l'APV $\mathsf{etendre}(A)$. Ces deux APV sont représentés à la Figure 3.4. Les parties en pointillés sont les parties de A modifiées par etendre. A est conforme en pile pour $Q = Q_\perp \uplus Q_{\not\perp}$ et $\Gamma = \Gamma_\perp \uplus \Gamma_{\not\perp}$ avec $Q_\perp = \{1, 4\}$, $Q_{\not\perp} = \{2, 3, 5, 6\}$, $\Gamma_\perp = \{\gamma\}$ et $\Gamma_{\not\perp} = \{\gamma'\}$. Nous verrons plus tard que A est produit par la construction confpile. De plus nous avons $\mathsf{bordure}(A_{\mathsf{ext}}) = \{3\}$ et $\mathsf{piege}(A_{\mathsf{ext}}) = \{\bar{f}_{\not\perp}, \bar{f}_\perp\}$

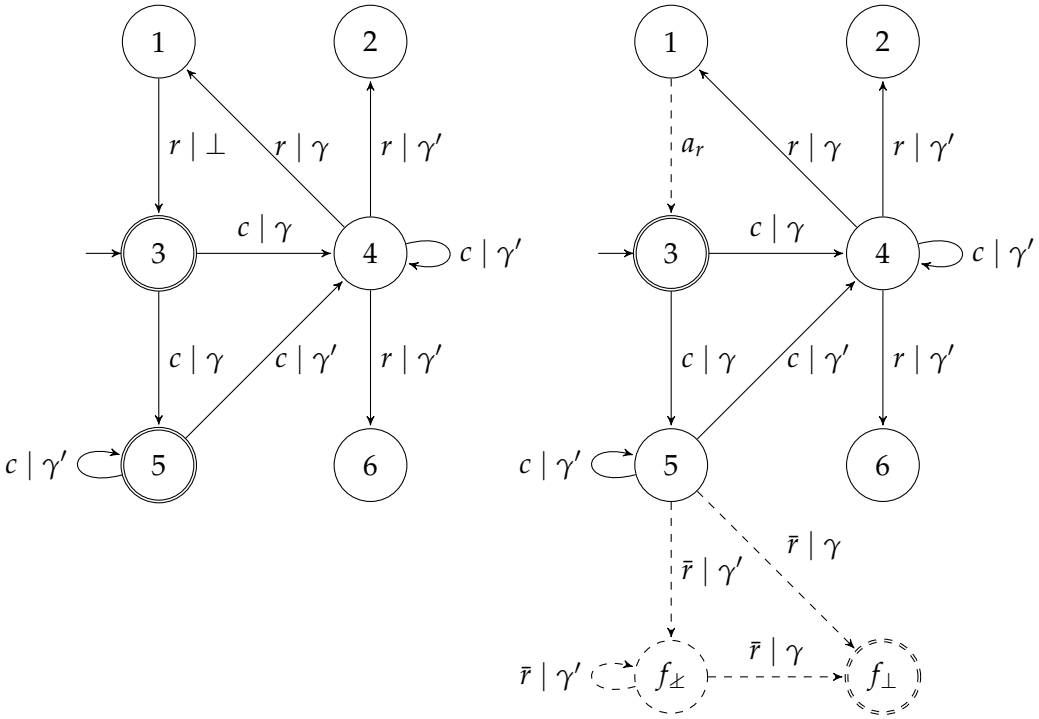


FIGURE 3.4 – L'APV A est à gauche et l'APV $A_{\mathsf{ext}} = \mathsf{etendre}(A)$ est à droite.

Dans la suite de cette section, nous allons montrer que pour tout APV A et tout mot $w \in \mathcal{L}(A)$, il existe une bijection entre les calculs acceptants de A sur w et ceux de $\mathsf{etendre}(A)$ sur $\mathsf{ext}(w)$. Pour cela nous allons montrer en premier lieu dans le Lemme 3.4.6 qu'il existe un calcul initialisé sur un mot u de A si et seulement s'il existe un calcul initialisé sur le mot $\mathsf{ext}_\iota(w)$ de A_{ext}. Pour montrer ce résultat nous avons besoin d'étendre la fonction ext_ι aux calculs. Soient $n, m \in \mathbb{N}$ des entiers, $a_1 \dots a_m \in \Delta^*$ un mot et $\rho : (p_1, \sigma_1) \xrightarrow{a_1} {}^*(p_2, \sigma_2) \xrightarrow{a_m} (q_{m+1}, \sigma_{m+1})$ un calcul de A, $\mathsf{ext}_\iota(\rho, n)$ est défini de la manière suivante :

$$\mathsf{ext}_\iota(\rho, n) : (p_1, \sigma_1) \xrightarrow{b_1} {}^*(p_2, \sigma_2) \xrightarrow{b_m} (q_{m+1}, \sigma_{m+1}) \qquad \text{avec } b_1 \dots b_m = \mathsf{ext}_\iota(a_1 \dots a_m, n)$$

Nous définissons $\mathsf{ext}_\iota(\rho)$ de la même manière que pour les mots.

Dans un deuxième temps, nous allons montrer dans le Lemme 3.4.7 que pour toute pile σ il existe un unique chemin sur $\{\bar{r}\}$ dépilant σ. Enfin, nous combinons ces deux résultats dans le Théorème 3.4.8 afin d'obtenir le résultat souhaité.

Lemme 3.4.6

Soient A un APV conforme en pile sur l'alphabet structuré Δ, A_{ext} l'APV $\mathsf{etendre}(A)$ et $w \in \Delta^*$ un mot. ρ est un calcul initialisé de A sur w si et seulement si $\mathsf{ext}_\iota(\rho)$ est un calcul initialisé de A_{ext} sur $\mathsf{ext}_\iota(w)$.

Démonstration. Par définition, $\mathsf{ext}_\iota(w)$ est obtenu depuis w en remplaçant chaque lettre de Δ_r sans ouvrant par un symbole interne. La construction $\mathsf{etendre}$ implémente correctement cette transformation grâce au point (1) de la définition de conforme en pile. □

Lemme 3.4.7

Soient A un APV conforme en pile sur l'alphabet structuré Δ, A_{ext} l'APV $\mathsf{etendre}(A)$ et ρ un calcul initialisé de A_{ext} sur un mot $w \in \Delta^*$ menant vers la configuration (p, σ) telle que p est un état de bord de A_{ext}, alors il existe exactement un calcul ρ' sur $\bar{r}^{|\sigma|}$ tel que $\rho\rho'$ est un calcul acceptant de A_{ext}.

Démonstration. Nous notons $A = (Q, I, F, \Gamma, \delta)$, $Q = Q_\perp \uplus Q_{\not\perp}$ et $\Gamma = \Gamma_\perp \uplus \Gamma_{\not\perp}$. Par construction de A_{ext}, le mot w ne contient aucune occurrence de $\bar{r}$. Par le Lemme 3.4.6, il existe un calcul ρ_1 de A tel que $\rho = \mathsf{ext}_\iota(\rho_1)$. Comme observé ci-dessus nous avons $\mathsf{bordure}(A_{\mathsf{ext}}) = F \cap Q_{\not\perp}$. Donc, par le point (2) de la définition de conforme en pile, nous avons $\sigma \in \{\gamma\sigma' \mid \gamma \in \Gamma_\perp, \sigma' \in \Gamma_{\not\perp}^*\}$, et par construction il existe un unique calcul de A_{ext} menant de p vers $\bar{f}_\perp$ sur $\bar{r}^{|\sigma|}$ dépilant tous les symboles de la pile. □

Théorème 3.4.8

Soient A un APV conforme en pile sur l'alphabet structuré Δ, A_{ext} l'APV $\mathsf{etendre}(A)$ et $w \in \Delta^*$ un mot, il existe une bijection entre les calculs de $\mathsf{ACal}_w(A)$ et les calculs $\mathsf{ACal}_{\mathsf{ext}(w)}(A_{\mathsf{ext}})$, et donc $\mathcal{L}(A_{\mathsf{ext}}) = \mathsf{ext}(\mathcal{L}(A))$.

Démonstration. Nous notons $A = (Q, I, F, \Gamma, \delta)$, $Q = Q_\perp \uplus Q_{\not\perp}$ et $\Gamma = \Gamma_\perp \uplus \Gamma_{\not\perp}$. Premièrement supposons que w est un mot où toute lettre de Δ_c possède un fermant, alors $\mathsf{ext}_\iota(w) = \mathsf{ext}(w)$. Soit ρ un calcul acceptant de A sur w menant vers la configuration (p, σ). Comme $\mathsf{ext}_\iota(w) = \mathsf{ext}(w)$, nous concluons que $\sigma = \perp$ et donc, par le point (1) de la définition de conforme en pile, nous avons $p \in F \cap Q_\perp$. Le résultat suit par le Lemme 3.4.6.

Sinon s'il existe une lettre de Δ_c dans w sans fermant. Nous définissons l'ensemble ECal_w comme suit :

$$\mathsf{ECal}_w = \{\rho \in \mathsf{Cal}_{\mathsf{ext}_\iota(w)}(A_{\mathsf{ext}}) \mid \rho \text{ est initialisé est termine dans un état de bord}\}$$

Comme les états de bord de A_{ext} sont construits à partir des états acceptants de A, par le Lemme 3.4.6 pour tout calcul $\rho \in \mathsf{Cal}_w(A)$ nous avons $\rho \in \mathsf{ACal}_w(A)$ si et seulement si $\mathsf{ext}_\iota(\rho) \in \mathsf{ECal}_w$. Le résultat suit par le Lemme 3.4.7. □

3.4.2 ... Et vice-versa.

Nous allons définir l'inverse la fonction $\mathsf{etendre}$, appelée $\mathsf{retracter}$, produisant la dernière étape de la procédure. Cependant, nous devons déterminer le piège parmi les états de l' APVbi pour le supprimer et recouvrer le langage original. Ça n'est pas toujours possible, et nous introduisons une propriété appelée *rétractable*, permettant d'appliquer la construction $\mathsf{retracter}$:

Définition 3.4.9

Soient Δ un alphabet structuré et $A = (Q, I, F, \Gamma, \delta)$ un APV sur l'alphabet Δ^{ext}. Nous considérons les conditions suivantes :

1. Il existe un APV B sur Δ tel que $\mathcal{L}(A) = \mathsf{ext}(\mathcal{L}(B))$,
2. Nous avons $\mathsf{piege}(A) \cap I = \varnothing$,
3. Pour toute transition τ dans δ, $\mathsf{lettre}(\tau) = \bar{r}$ si et seulement si $\mathsf{cible}(\tau) \in \mathsf{piege}(A)$,
4. Pour toute transition τ dans δ telle que $\mathsf{source}(\tau) \in \mathsf{piege}(A)$, $\mathsf{lettre}(\tau) = \bar{r}$,
5. Pour tout calcul initialisé de A terminant dans un état de bord il existe un entier $n \in \mathbb{N}_{>0}$ et exactement un calcul ρ' sur $\bar{r}^n$ tel que $\rho\rho'$ est un calcul acceptant.
6. Pour toute transition $\tau \in \delta_r$ telle que $\mathsf{source}(\tau) \in \mathsf{bordure}(A)$, $\mathsf{lettre}(\tau) = \bar{r}$.

Nous appelons A *rétractable* s'il remplit les conditions $(1), \ldots, (5)$ et *fortement rétractable* s'il est rétractable et remplit la condition (6).

La propriété de rétractabilité garantit que retracter peut être appliqué. Dans la suite nous allons montrer que nos constructions préservent la rétractabilité et donc le piège. Par exemple celui de l' APVbi produit par $\mathsf{freduire} \circ \mathsf{etendre}$ est $\{(\bar{f}_\perp, \bar{f}_\perp), (\bar{f}_\not\perp, \bar{f}_\not\perp)\}$. Nous avons besoin de la rétractabilité forte car la construction reduire ne préserve pas la rétractabilité. Nous allons d'abord montrer que la construction etendre assure la rétractabilité forte :

Théorème 3.4.10

Soit A un APV conforme en pile, alors $\mathsf{etendre}(A)$ est fortement rétractable.

Démonstration. La propriété (1) est une conséquence du Théorème 3.4.8. Les propriétés (2), (3), (4) sont des conséquences de la construction, la propriété (5) est une conséquence du Lemme 3.4.7 et la propriété (6) est une conséquence de la propriété (3) de la définition de conforme en pile. □

Nous définissons maintenant la construction retracter :

Définition 3.4.11

Soient Δ un alphabet structuré et $A = (Q, I, F, \Gamma, \delta)$ un APVbi rétractable sur l'alphabet Δ^{ext}, nous définissons l'APV $\mathsf{retracter}(A) = (Q', I', F', \Gamma', \delta')$ sur l'alphabet Δ par :

$$\circ\ Q' = Q \backslash \mathsf{piege}(A) \qquad I' = I \qquad F' = (F \backslash \mathsf{piege}(A)) \cup \mathsf{bordure}(A) \qquad \Gamma' = \Gamma$$

$\circ\ \delta' = \delta'_c \uplus \delta'_r \uplus \delta'^{\perp}_r \uplus \delta'_\iota$ avec :

- $\delta'_c = \delta_c$
- $\delta'_r = \{\ \tau \in \delta_r \mid \mathsf{lettre}(\tau) \neq \bar{r}\ \}$
- $\delta'^{\perp}_r = \{\ (p, r, \perp, q) \mid (p, a_r, q) \in \delta_\iota\ \}$
- $\delta'_\iota = \{\ \tau \in \delta_\iota \mid \mathsf{lettre}(\tau) \in \Delta_\iota\ \}$

Cette construction fonctionne comme suit : elle remplace toutes les transitions sur la lettre a_r par des transitions dépilant la pile vide sur la lettre r, et supprime les transitions dépilantes

sur $\bar{r}$. Notez que les états finaux de $\mathsf{retracter}(A)$ sont les états de bord de A et les états finaux de A qui ne sont pas dans $\mathsf{piege}(A)$.

Nous montrons maintenant que pour tout APV A, la construction retracter induit une bijection entre les calculs de A et les calculs de $\mathsf{retracter}(A)$. Premièrement, le Lemme 3.4.12 établit qu'il existe un calcul de A sur le mot u débutant sur une pile σ si et seulement s'il existe un calcul de $\mathsf{retracter}(A)$ sur le mot $\mathsf{ext}_\iota(w, |\sigma|)$. Nous utilisons ce résultat dans la démonstration du Théorème 3.4.13 afin de montrer le résultat souhaité.

Lemme 3.4.12

> Soient A un APV conforme en pile sur l'alphabet structuré Δ et $w \in \Delta^*$ un mot. Le calcul ρ appartient à $\mathsf{Cal}_w(\mathsf{retracter}(A))$ avec $\mathsf{premier}(\rho) = (p, |\sigma|)$ si et seulement si le calcul initialisé $\mathsf{ext}_\iota(\rho, |\sigma|)$ appartient à $\mathsf{Cal}_{\mathsf{ext}_\iota(w,|\sigma|)}(A)$.

Démonstration. Nous considérons en premier le sens $\Rightarrow$. Il convient d'observer que $w \in \Delta^*$ assure, par les conditions (2) et (3) de la définition de la rétractabilité, que les configurations (p_i, σ_i) composant le calcul ρ vérifient la propriété $p_i \notin \mathsf{piege}(A)$ pour tout $i \in [1, |w| + 1]$. Par conséquent, par définition des états et des transitions de $\mathsf{retracter}(A)$ nous en concluons que $\mathsf{ext}_\iota(\rho)$ appartient à $\mathsf{Cal}_{\mathsf{ext}_\iota(w,|\sigma|)}(A)$.

Nous considérons maintenant le sens $\Leftarrow$. Par la définition de la fonction ext_ι, quand $\mathsf{ext}_\iota(\rho)$ lit une lettre $a_r \in \Delta^{\mathsf{ext}}$ alors la pile est vide. Par conséquent, par définition des états et des transitions de $\mathsf{retracter}(A)$ nous en concluons que ρ appartient à $\mathsf{Cal}_w(\mathsf{retracter}(A))$.

□

Théorème 3.4.13

> Soient Δ un alphabet structuré et A un APV rétractable sur Δ^{ext}. Pour tout mot $w \in \Delta^*$, il existe une bijection entre $\mathsf{ACal}_w(\mathsf{retracter}(A))$ et $\mathsf{ACal}_{\mathsf{ext}(w)}(A)$, et en particulier $\mathcal{L}(A) = \mathsf{ext}(\mathcal{L}(\mathsf{retracter}(A)))$.

Démonstration. Soit $w \in \Delta^*$ un mot. Nous distinguons deux cas. si w est un mot tel que toute lettre de Δ_c est fermée, alors $\mathsf{ext}(w) = \mathsf{ext}_\iota(w)$. Par le Lemme 3.4.12 et le point (2) de la définition de la rétractabilité, ext amène une bijection entre l'ensemble des calculs initialisés de $\mathsf{retracter}(A)$ sur w et l'ensemble des calcul initialisés de A sur $\mathsf{ext}(w)$. De plus, un calcul initialisé ρ de $\mathsf{retracter}(A)$ sur w est acceptant si et seulement si $\mathsf{ext}(\rho)$ est acceptant dans A. En effet, comme toute lettre de Δ_c de w est fermée, la configuration (p, σ) atteinte par ρ vérifie $\sigma = \bot$, et donc $p \notin \mathsf{bordure}(A)$ par la condition (5) de la définition de rétractable. Alors p est un état final de $\mathsf{retracter}(A)$ si et seulement si p est un état final de A. Sinon, il existe une lettre de Δ_c de w non fermée. Nous définissons l'ensemble RCal_w comme suit :

$$\mathsf{RCal}_w = \{\rho \in \mathsf{Cal}_{\mathsf{ext}_\iota(w)}(A) \mid \rho \text{ est initialisé et termine dans un état de bord}\}$$

Comme pour le cas précédent, nous pouvons montrer par la condition (2) de la définition de la rétractabilité et par le Lemme 3.4.12 que ext_ι introduit une bijection entre $\mathsf{ACal}_w(\mathsf{retracter}(A))$ et RCal_w. Et donc par le point (5) de la définition de rétractable, nous avons une bijection entre $\mathsf{ACal}_w(\mathsf{retracter}(A))$ et $\mathsf{ACal}_{\mathsf{ext}(w)}(A)$.

Enfin par la condition (1) de la définition de la rétractabilité, tout mot accepté par A est de la forme $\mathsf{ext}(w)$ pour un certain mot $w \in \Delta^*$, et donc la bijection précédente implique $\mathcal{L}(A) = \mathsf{ext}(\mathcal{L}(\mathsf{retracter}(A)))$. □

Nous allons maintenant montrer que retracter préserve l'émondage. La partie compliquée de cette preuve est celle concernant la co-réduction. Un APVbi est co-réduit si tout calcul finissant dans une configuration acceptante est accessible. Dans le cas des APVbi, il suffit de considérer les configurations finales de la forme $(p, \bot)$ avec p un état final car tous les calculs acceptants finissent sur une pile vide. Nous devons donc nous assurer que la construction retracter prend en compte cette complexité supplémentaire, car elle produit un APV à partir d'un APVbi.

Proposition 3.4.14

> Soient Δ un alphabet structuré et A un APV rétractable sur Δ^{ext}, si A est émondé alors $\mathsf{retracter}(A)$ l'est aussi.

Démonstration. Soit A_{red} l'APV $\mathsf{retracter}(A)$. Pour la partie réduction, nous considérons un mot $w \in \Delta^*$ et un calcul initialisé $\rho : (p, \bot) \xrightarrow{w}{}^*(q, \sigma) \in \mathsf{Cal}_w(A)$. Par le Lemme 3.4.12 $\hat{\rho} = \mathsf{ext}_\iota(\rho)$ est un calcul de A sur $\hat{w} = \mathsf{ext}_\iota(w)$. Il convient d'observer que $\hat{\rho}$ est initialisé. Comme A est réduit, il existe un calcul $\hat{\rho}'$ sur w' tel que $\hat{\rho}\hat{\rho}'$ est un calcul acceptant de A. Par construction nous décomposons w' par $w' = w_1 w_2$ avec $w_1 \in \Delta^*$, $w_2 \in \{\bar{r}\}^*$ et $\hat{\rho}'$ par $\hat{\rho}_1 : (q, \sigma) \xrightarrow{w_1}{}^*(q', \sigma')$ sur w_1 et $\hat{\rho}_2 : (q', \sigma') \xrightarrow{w_2}{}^*(p', \bot)$ sur w_2. Si $w_2 = \varepsilon$, alors $\sigma' = \bot$ et par construction q' est un état final de A_{ret}. Par le Lemme 3.4.12, il existe un calcul ρ_1 tel que $\hat{\rho}_1 = \mathsf{ext}(\rho_1, |\sigma|)$ et $\rho\rho_1$ est un calcul acceptant de A_{ret}. Si $w_2 \neq \varepsilon$, alors $q' \in \mathsf{bordure}(A)$ et donc q' est un état final de A_{ret}. De même, par le Lemme 3.4.12 il existe un calcul ρ_1 tel que $\hat{\rho}_1 = \mathsf{ext}(\rho_1, |\sigma|)$ et $\rho\rho_1$ est un calcul acceptant de A_{ret}.

Pour la partie co-réduction, nous considérons deux calculs $\rho : (p, \sigma) \xrightarrow{w} (q, \sigma')$ et $\rho' : (i, \bot) \xrightarrow{w'} (q, \sigma')$ de A_{ret} avec q un état final et i un état initial. Nous distinguons deux cas :

- Si $\sigma' = \bot$, alors par la condition (5) de la définition de la rétractabilité, l'état q ne peut pas être dans $\mathsf{bordure}(A)$. Par définition de A_{ret}, ceci implique que q est un état final de A. Par le Lemme 3.4.12, $\hat{\rho} = \mathsf{ext}_\iota(\rho, |\sigma|)$ est un calcul de A sur $\hat{w} = \mathsf{ext}_\iota(w, |\sigma|)$, alors comme A est un APVbi et est co-reduit, il existe un calcul $\hat{\rho}''$ de A sur le mot $\hat{w}''$ tel que $\hat{\rho}''\hat{\rho}$ est un calcul acceptant de A sur $\hat{w}''\hat{w}$. Donc par le Lemme 3.4.12 il existe un calcul ρ'' de A_{ret} tel que $\hat{\rho}'' = \mathsf{ext}_\iota(\rho'')$ est un calcul sur w'' avec $\hat{w}'' = \mathsf{ext}(w'')$ et $\rho''\rho$ est un calcul acceptant de A_{ret} sur $w''w$.
- Si $\sigma' \neq \bot$, alors $q \in \mathsf{bordure}(A)$. Par le Lemme 3.4.12, $\hat{\rho} = \mathsf{ext}_\iota(\rho, |\sigma|)$ et $\hat{\rho}' = \mathsf{ext}_\iota(\rho')$ sont des calculs de A sur $\mathsf{ext}_\iota(w, |\sigma|)$ et $\mathsf{ext}_\iota(w')$ respectivement. Il convient d'observer que $\hat{\rho}'$ est initialisé. Alors par le point (5) de la rétractabilité il existe exactement un calcul $\bar{\rho}$ sur $\bar{r}^+$ tel que $\hat{\rho}'\bar{\rho}$ est un calcul acceptant de A et donc $\hat{\rho}\bar{\rho}$ est aussi un calcul de A. Comme A est co-réduit il existe un calcul $\hat{\rho}''$ de A sur le mot $\hat{w}''$ tel que $\hat{\rho}''\hat{\rho}\bar{\rho}$ est un calcul acceptant de A. Par le Lemme 3.4.12, il existe un calcul ρ'' de A_{ret} tel que $\hat{\rho}'' = \mathsf{ext}_\iota(\rho'')$ sur w'' avec $\hat{w}'' = \mathsf{ext}(w'')$. Donc par le Lemme 3.4.12, $\rho''\rho$ est un calcul acceptant de A_{ret} sur $w''w$. □

Nous allons maintenant montrer que nos différentes constructions préservent la rétractabilité, ceci dans le but de pouvoir les composer et appliquer à la fin la construction retracter. Nous allons donc montrer que reduire préserve la rétractabilité forte dans le Lemme 3.4.15 et coreduire préserve la rétractabilité dans le Lemme 3.4.16.

Proposition 3.4.15

Soit A un APV fortement rétractable sur l'alphabet structuré Δ, alors l'APV $A_{\mathsf{red}} = \mathsf{reduire}(A)$ est fortement rétractable.

Démonstration. Premièrement notons que par construction nous avons :

$$\text{Pour tout } (p,q) \in Q', \text{ si } (p,q) \in \mathsf{piege}(A_{\mathsf{red}}) \text{ alors } p \in \mathsf{piege}(A) \tag{3.1a}$$
$$\text{Pour tout } (p,q) \in Q', \text{ si } (p,q) \in \mathsf{bordure}(A_{\mathsf{red}}) \text{ alors } p \in \mathsf{bordure}(A) \tag{3.1b}$$

L'Énoncé (3.1a) est une conséquence de la construction reduire. Concernant l'Énoncé (3.1b) le fait que $p \notin \mathsf{piege}(A)$ est amené par la condition (3) de la Définition 3.1.3 appliquée sur (p,q), et le fait qu'il existe une transition sur $\bar{r}$ de l'état p vers un état de $\mathsf{piege}(A)$ vient de la construction reduire.

La condition (1) de la définition de la rétractabilité est évidente car $\mathcal{L}(A) = \mathcal{L}(A_{\mathsf{red}})$. Les conditions (2), (3) et (4) peuvent être montrées en utilisant l'Énoncé (3.1a), et la condition (6) en utilisant l'Énoncé (3.1b).

Pour prouver la condition (5), nous considérons un calcul initialisé ρ de A_{red} sur un mot w menant à la configuration $((q,q),\sigma)$ telle que $(q,q) \in \mathsf{bordure}(A_{\mathsf{red}})$. Par le Lemme 3.3.4, $\rho' = \Phi_{\mathsf{fred}}(\rho)$ [1] est un calcul de A menant à la configuration $(q, \pi_1(\sigma))$. Comme $q \in \mathsf{bordure}(A)$ (par l'Énoncé (3.1b)) et A est rétractable, $\pi_1(\sigma)$ n'est pas vide et σ ne l'est pas non plus. On peut donc décomposer ρ de cette manière :

$$\rho : ((i,f),\bot) \xrightarrow{w'}{}^{*} ((p',q'),\sigma') \xrightarrow{c} ((p,q),\sigma) \xrightarrow{w''}{}^{*} ((q,q),\sigma)$$

avec $w' \in \Delta^*$, $w'' \in \Delta_{\mathsf{bi}}^*$, $c \in \Delta_c$ et $\sigma = \sigma' \cdot (\gamma, q')$ avec $\gamma \in \Gamma$. De plus, par définition de la réduction, il existe une transition $(q, r, \gamma, q'') \in \delta_r$. Comme $q \in \mathsf{bordure}(A)$, par la condition (6) de la rétractabilité nous avons $r = \bar{r}$, et donc il existe une transition $((q,q), \bar{r}, (\gamma, q'), (q'', q')) \in \delta'_r$ et $\rho_1 : ((q,q),\sigma) \xrightarrow{\bar{r}} ((q'',q'),\sigma')$ est un calcul de A_{red}. Comme A_{red} est réduit, il est possible de compléter le calcul initialisé $\rho\rho_1$ en un calcul acceptant. Par la condition (1) les mots acceptés sont de la forme $\Delta^*\bar{r}^*$. Il existe donc un calcul $\bar{\rho} : ((q'',q'),\sigma') \xrightarrow{\bar{r}^{|\sigma'|}} ((f,f),\bot)$ tel que $\rho\rho_1\bar{\rho}$ est un calcul acceptant de A_{red}. Enfin, l'unicité de l'extension ρ vient de la bijection induite par la fonction Φ_{fred}. □

Proposition 3.4.16

Soit A un APV rétractable sur l'alphabet structuré Δ, alors $A_{\mathsf{cor}} = \mathsf{coreduire}(A)$ est rétractable.

Démonstration. Premièrement notons que par construction nous avons :

$$\text{Pour tout } (p,q) \in Q', \text{ si } (p,q) \in \mathsf{piege}(A_{\mathsf{cor}}) \text{ alors } q \in \mathsf{piege}(A) \tag{3.2a}$$
$$\text{Pour tout } (p,q) \in Q', \text{ si } (p,q) \in \mathsf{bordure}(A_{\mathsf{cor}}) \text{ alors } q \in \mathsf{bordure}(A) \tag{3.2b}$$

Les Énoncés (3.2a) et (3.2b) peuvent être prouvés de manière similaire aux Énoncés (3.1a) et (3.1b).

1. Voir à la page 63 pour la définition de Φ_{fred}.

La condition (1) de la définition de rétractabilité est évidente car $\mathcal{L}(A) = \mathcal{L}(A_{\mathsf{cor}})$. Les conditions (2), (3) et (4) peuvent être facilement prouvées en utilisant l'Énoncé (3.2a).

Pour prouver la condition (5), nous établissons premièrement la propriété suivante venant de la définition de coreduire : pour tout calcul initialisé ρ de A_{cor} menant à une configuration $((p,q),\sigma)$, nous définissons ρ' le calcul $\Phi_{\mathsf{fcor}}(\rho)$ [2] de A menant à la configuration $(q,\bar{\sigma})$ avec $\bar{\sigma} = \pi_1(\sigma)$, s'il existe une transition $(q,r,\gamma,q') \in \delta_r$ telle que le calcul $\rho' \xrightarrow{r} (q',\bar{\sigma}')$ existe dans A avec $\bar{\sigma} = \bar{\sigma}'\gamma$, alors il existe une transition $((p,q),r,(\gamma,p'),(p',q')) \in \delta'_r$ telle que le calcul $\rho \xrightarrow{r} ((p',q'),\sigma')$ existe dans A_{cor}, avec $\pi_1(\sigma') = \bar{\sigma}'$.

Alors, la condition (5) suit facilement de cette propriété, l'Énoncé (3.2b) et la rétractabilité de A. L'unicité vient de la bijection Φ_{fcor}. □

3.4.3 La construction confpile

Nous avons vu dans la Sous-section 3.4.1 comment transformer un APV en un APVbi en utilisant la construction etendre. Cependant etendre a besoin d'un APV conforme en pile en entrée. Dans cette section, nous allons présenter la construction confpile transformant un APV quelconque en un APV conforme en pile équivalent.

Nous allons premièrement donner une intuition de la construction et ensuite montrer le lien avec la définition de conforme en pile (Définition 3.4.2). Notre construction distingue trois types de configurations le long d'un calcul : les configurations pour lesquelles la pile est vide, celles où le symbole au sommet de la pile sera dépilé avant de terminer le calcul et celles où le symbole au sommet de la pile ne sera pas dépilé avant de terminer le calcul. Lorsqu'un symbole est empilé il est donc nécessaire de deviner si ce symbole sera dépilé dans le futur ou non. De plus pour maintenir une information correcte tout le long du calcul nous modifierons la pile pour mémoriser quel était sa forme avant sa modification. Nous définissons donc en premier lieu l'ensemble de symboles spéciaux $\mathbb{S}$ comme suit :

$$\mathbb{S} = \{\bot, \top, \circ\}$$

Intuitivement cet ensemble nous permettra de distinguer les trois types de configurations. Nous présentons maintenant la construction confpile.

Définition 3.4.17

Soit un APV $A = (Q, I, F, \Gamma, \delta)$ sur l'alphabet structuré Δ, nous définissons l'APV $\mathsf{confpile}(A) = (Q', I', F', \Gamma', \delta')$ sur l'alphabet structuré Δ par :

$$\begin{array}{lcl}
\circ\ Q' & = & (Q \times T) \qquad I' = I \times \{\bot\} \qquad F' = F \times \{\bot, \circ\} \qquad \Gamma' = \Gamma \times T \\
\circ\ \delta' & = & \delta'_c \cup \delta'_r \cup \delta'^{\bot\prime}_r \cup \delta'_\iota \text{ avec :} \\
\quad - \ \delta'_c & = & \left\{ ((p,\int),c,(\gamma,\int),(q,\int')) \;\middle|\; \begin{array}{l}(p,c,\gamma,q) \in \delta_c \text{ et} \\ \int'{=}\top \text{ ou } (\int'{=}\circ \wedge \int{\neq}\top)\end{array} \right\} \\
\quad - \ \delta'_r & = & \{ ((p,\top),(\gamma,\int),r,(q,\int)) \mid (p,\gamma,r,q) \in \delta_r \} \\
\quad - \ \delta'^{\bot\prime}_r & = & \{ ((p,\bot),\bot,r,(q,\bot))) \mid (p,\bot,r,q) \in \delta_r \} \\
\quad - \ \delta'_\iota & = & \{ ((p,\int),a,(q,\int)) \mid (p,a,q) \in \delta_\iota \}
\end{array}$$

La sémantique de cette construction est la suivante : supposons qu'un calcul initialisé atteigne l'état $(p,\int)$, alors la forme de la pile dépend de $\int$. Si $\int = \bot$ alors la pile est vide. Si $\int = \circ$

2. Voir à la page 70 pour la définition de Φ_{fcor}.

alors aucun des symboles dans la pile ne sera dépilé avant la fin du calcul. Enfin, si $\int = \top$, alors la pile est composée de symboles qui ne seront jamais dépilés, suivis de symboles qui seront dépilés avant la fin du calcul.

Exemple 3.4.18

Soient $A = (Q, I, F, \Gamma, \delta)$ un APV sur l'alphabet structuré Δ et A_{sc} l'APV $\mathsf{confpile}(A)$. Ces deux APV sont représentés dans la Figure 3.5. Notez que l'état $(1, \top)$ n'est pas final, car si un calcul passe par cet état cela signifie qu'il doit finir de dépiler ses symboles avant de terminer.

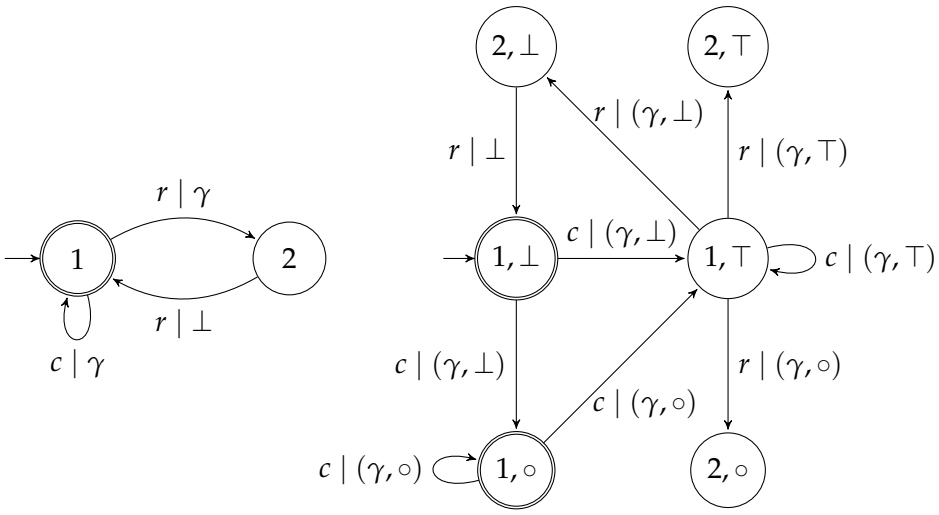


FIGURE 3.5 – À gauche l'APV A, à droite l'APV $\mathsf{confpile}(A)$.

Nous allons maintenant montrer dans le Lemme 3.4.19 que pour tout APV A l'information stockée dans l'état de $\mathsf{confpile}(A)$ se propage correctement le long de calculs sur des mots bien imbriqués.

Lemme 3.4.19

Soient A un APV sur l'alphabet structuré Δ et $\rho : ((p, \int), \sigma) \xrightarrow{w} ((p', \int'), \sigma)$ un calcul de $\mathsf{confpile}(A)$ sur un mot bien imbriqué w, alors $\int = \int'$.

Démonstration. Nous prouvons ce résultat par induction sur la structure de w :

- Si $w = \epsilon$ alors le résultat est évident.
- Si $w = au$ avec $a \in \Delta_\iota$ et u un mot bien imbriqué, alors par hypothèse d'induction il existe un calcul $\rho' : ((p'', \int''), \sigma') \xrightarrow{u} ((p', \int'), \sigma)$ tel que $\int'' = \int'$. Par construction il existe une transition $((p, \int), a, (p'', \int'')) \in \delta'_\iota$ telle que $\int = \int''$, et donc $\int = \int'$.
- Si $w = cu_1ru_2$ avec $c \in \Delta_c$, $r \in \Delta_r$ et u_1, u_2 deux mots bien imbriqués, par hypothèse d'induction il existe deux calculs $\rho' : ((p_1, \int_1), \sigma_1) \xrightarrow{u_1} ((p_2, \int_2), \sigma_2)$ et $\rho'' : ((p_3, \int_3), \sigma_3) \xrightarrow{u_2} ((p', \int'), \sigma)$ tels que $\int' = \int_3$ et $\int_1 = \int_2$. Par construction de δ'_c il existe deux transitions $((p, \int), a, (\gamma, \int), (p_1, \int_1)) \in \delta'_c$ et $((p_2, \int_2), r, (\gamma, \int_3), (p_3, \int_3)) \in \delta'_r$ avec $\int = \int_3$. Nous concluons que $\int' = \int_3$ et donc $\int = \int'$. □

Pour tout symbole $\int \in \mathbb{S}$ nous définissons l'ensemble $\mathbb{S}^*_\int$ comme suit :

$$\begin{array}{rcl} \mathbb{S}^*_\bot & = & \{\bot\} \\ \mathbb{S}^*_\top & = & \{\gamma\sigma\sigma' \mid \gamma \in \Gamma \times \{\bot\}, \sigma \in (\Gamma \times \{\circ\})^*, \sigma' \in (\Gamma \times \{\top\})^*\} \\ \mathbb{S}^*_\circ & = & \{\gamma\sigma \mid \gamma \in \Gamma \times \{\bot\}, \sigma \in (\Gamma \times \{\circ\})^*\} \end{array}$$

Nous décrivons dans le Lemme 3.4.20 comment les états sont conformes avec la pile dans $\mathsf{confpile}(A)$.

Lemme 3.4.20

Soient A un APV sur l'alphabet structuré Δ et un calcul ρ de $\mathsf{confpile}(A)$ sur $a_1 \dots a_n \in \Delta^*$ tel que $\rho : ((p_0, \int_0), \sigma_0) \xrightarrow{a_1} \cdots \xrightarrow{a_n} ((p_n, \int_n), \sigma_n)$. Si $\sigma_0 \in \mathbb{S}^*_{\int_0}$, alors pour tout $i \in [1, k]$, $\sigma_i \in \mathbb{S}^*_{\int_i}$.

Démonstration. Soit ρ un calcul de $A_{\mathsf{sc}} = \mathsf{confpile}(A)$. Nous procédons par induction sur la longueur de ρ. Si $\rho : ((q, \int), \sigma)$, alors par hypothèse $\sigma \in \mathbb{S}^*_\int$. Sinon il existe une lettre $a \in \Delta$ tel que ρ se décompose en $\rho' \xrightarrow{a} ((q, \int'), \sigma')$. Nous appelons $((p, \int), \sigma)$ la configuration $\mathsf{dernier}(\rho')$. Par hypothèse d'induction sur ρ' nous avons $\sigma \in \mathbb{S}^*_\int$. Nous montrons maintenant que $\sigma' \in \mathbb{S}^*_{\int'}$. Par définition il existe une transition τ, allant de $((p, \int), \sigma)$ vers $((q, \int'), \sigma')$. Nous distinguons quatre cas, selon le type de τ :

- Si $\tau = ((p, \int), a, (q, \int')) \in \delta_\iota$ avec $a \in \Delta_\iota$, par définition $\int' = \int$ et $\sigma = \sigma'$, et donc $\sigma' \in \mathbb{S}^*_{\int'}$.
- Si $\tau = ((p, \int), a, (\gamma, \int), (q, \int')) \in \delta_c$ avec $a \in \Delta_c$ et $\int' = \top$, alors par définition de $\mathbb{S}$ nous avons $\mathbb{S}^*_\int \cdot \{(\gamma, \int)\} \subseteq \mathbb{S}^*_{\int'}$. Comme $\sigma' = \sigma \cdot (\gamma, \int)$, nous concluons que $\sigma' \in \mathbb{S}^*_{\int'}$.
- Si $\tau = ((p, \int), a, (\gamma, \int), (q, \int')) \in \delta_c$ avec $a \in \Delta_c$ et $\int' = \circ$, alors par construction $\int \in \{\bot, \circ\}$ et par définition de $\mathbb{S}$ nous avons $\mathbb{S}^*_\int \cdot \{(\gamma, \int)\} \subseteq \mathbb{S}^*_{\int'}$. Comme $\sigma' = \sigma \cdot (\gamma, \int)$, nous concluons que $\sigma' \in \mathbb{S}^*_\circ$.
- Si $\tau = ((p, \int), (\gamma, \int), a, (q, \int')) \in \delta_r$ avec $a \in \Delta_r$, alors par construction $\int = \top$ et par définition de $\mathbb{S}$ nous avons $\mathbb{S}^*_\int \cdot \{(\gamma, \int)\} \subseteq \mathbb{S}^*_{\int'}$. Comme $\sigma = \sigma' \cdot (\gamma, \int)$, nous concluons que $\sigma' \in \mathbb{S}^*_{\int'}$.
- Si $\tau = ((p, \int), a, \bot, (q, \int')) \in \delta^\bot_r$ avec $a \in \Delta_r$ alors par construction $\int = \int' = \bot$ et donc $\sigma' = \bot \in \mathbb{S}^*_\bot$. □

Définition 3.4.21

Soient A un APV sur l'alphabet structuré Δ et $\rho : ((p_0, \int_0), \sigma_0) \xrightarrow{a_1} \cdots \xrightarrow{a_n} ((p_n, \int_n), \sigma_n)$ un calcul de $\mathsf{confpile}(A)$ sur $w \in \Delta^*$. Alors ρ est un calcul conforme en pile si $\sigma_0 \in S^*_{\int_0}$ et $\int_n \neq \top$.

Intuitivement, par la définition précédente, la contrainte $\int_n \neq \top$ assure qu'aucun symbole de la pile ne sera dépilé avant d'atteindre une configuration finale. Ceci est formalisé dans le lemme suivant.

Lemme 3.4.22

Soient A un APV sur l'alphabet structuré Δ et ρ un calcul conforme en pile de l'APV $\mathsf{confpile}(A)$ sur $w \in \Delta^*$ tel que $\rho : ((p, \int), \sigma) \xrightarrow{w}{}^*((p', \int'), \sigma')$ et $\sigma \neq \bot$. Alors w se décompose en $w_1 r w_2$ avec $w_1 \in \Delta_{\mathsf{bi}}^*$ et $r \in \Delta_r$ si et seulement si $\int = \top$.

Démonstration. Nous considérons en premier l'implication $\Rightarrow$. Si $\int \neq \top$, supposons que w se décompose en $w_1 r w_2$. Par le Lemme 3.4.19, nous pouvons aussi décomposer ρ comme suit :

$$\rho : ((p, \int), \sigma) \xrightarrow{w_1}{}^*((q, \int), \sigma) \xrightarrow{r} ((q', \int''), \sigma'') \xrightarrow{w_2}{}^*((p', \int'), \sigma')$$

$\int$ ne peut pas être égal à $\circ$ parce qu'il n'y a pas de transition $\tau \in \delta'_r$ telle que $\mathsf{source}(\tau) = (q, \circ)$ et $\mathsf{lettre}(\tau) \in \Delta_r$. $\int$ est aussi différent de $\bot$ car selon le Lemme 3.4.20, $\sigma \in \mathbb{S}_{\int}^*$ mais $\sigma \neq \bot$, ce qui est une contradiction. Nous concluons que $\int = \top$.

Nous considérons maintenant l'implication $\Leftarrow$. Si $\int = \top$, supposons que w ne se décompose pas en $w_1 r w_2$, alors $w = w_0 c_1 w_1 \cdots c_n w_n$, pour $n \in \mathbb{N}$, $c_i \in \Delta_c$ et $w_i \in \Delta_{\mathsf{bi}}^*$ pour tout $i \in \{0, \ldots n\}$. Nous pouvons décomposer ρ comme suit :

$$\rho : ((p_0, \int_0), \sigma_0) \xrightarrow{w_0}{}^*((p'_0, \int'_0), \sigma'_0) \xrightarrow{c_1} ((p_1, \int_1), \sigma_1) \xrightarrow{w_1}{}^* \cdots \xrightarrow{c_n} ((p_n, \int_n), \sigma_n) \xrightarrow{w_n}{}^*((p'_n, \int'_n), \sigma'_n)$$

avec $((p, \int), \sigma) = ((p_0, \int_0), \sigma_0)$ et $((p', \int'), \sigma') = ((p'_n, \int'_n), \sigma'_n)$. Par construction des transitions de $\mathsf{confpile}(A)$, si $\int'_i = \top$ alors $\int_{i+1} = \top$, et par le Lemme 3.4.19 nous avons $\int_i = \int'_i$, et donc comme $\int = \int_0 = \top$ nous pouvons conclure que $\int' = \int'_n = \top$. Ceci est une contradiction car ρ est un calcul conforme en pile et $\int'$ doit être différent de $\top$. □

Nous allons maintenant prouver le théorème suivant :

Théorème 3.4.23

Soit A un APV sur l'alphabet structuré Δ, alors $A_{\mathsf{sc}} = \mathsf{confpile}(A)$ peut-être construit en temps polynomial et :

- Pour tout mot $w \in \Delta^*$ il existe une bijection entre $\mathsf{ACal}_w(A)$ et $\mathsf{ACal}_w(A_{\mathsf{sc}})$, et donc $\mathcal{L}(A) = \mathcal{L}(A_{\mathsf{sc}})$.
- A_{sc} est conforme en pile.

Pour prouver ce théorème nous avons besoin d'indiquer comment confpile construit ses calculs. Soient $A = (Q, I, F, \Gamma, \delta)$ un APV sur l'alphabet structuré Δ et $A_{\mathsf{sc}} = (Q', I', F', \Gamma', \delta')$ l'APV $\mathsf{confpile}(A)$. Nous définissons une fonction Φ_{sc} des calculs de A_{sc} vers les calculs de A. Soit ρ un calcul de A_{sc} défini comme suit :

$$\rho : ((q_0, \int_0), \sigma_0) \xrightarrow{a_1} ((q_1, \int_1), \sigma_1) \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} ((q_{n-1}, \int_{n-1}), \sigma_{n-1}) \xrightarrow{a_n} ((q_n, \int_n), \sigma_n)$$

Nous définissons $\Phi_{\mathsf{sc}}(\rho')$ ainsi :

$$\Phi_{\mathsf{sc}}(\rho) : (q_0, \pi_1(\sigma_0)) \xrightarrow{a_1} (q_1, \pi_1(\sigma_1)) \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} (q_{n-1}, \pi_1(\sigma_{n-1})) \xrightarrow{a_n} (q_n, \pi_1(\sigma_n))$$

Nous allons maintenant montrer que Φ_{sc} induit bien une bijection entre les calculs acceptants de A et de A_{sc}. Pour cela, nous allons montrer dans la Proposition 3.4.24 que pour tout calcul ρ de A_{sc}, $\Phi_{\mathsf{sc}}(\rho)$ est bien un calcul de A, puis nous prouverons que cette fonction est injective dans le Lemme 3.4.25 et surjective dans le Lemme 3.4.26.

Proposition 3.4.24

Soient A un APV sur l'alphabet structuré Δ et A_{sc} l'APV $\mathsf{confpile}(A)$. Pour tout calcul conforme en pile ρ de A_{sc}, $\Phi_{\mathsf{sc}}(\rho')$ est un calcul de A.

Démonstration. Soit ρ un calcul de A_{sc}. Nous faisons la preuve par induction sur la longueur de ρ. Si $\rho : ((q, \int), \sigma)$ alors par construction $(q, \pi_1(\sigma))$ est un calcul de A. Sinon ρ se décompose par :

$$((p, \int), \sigma) \xrightarrow{a} \rho' \text{ avec } \rho' : ((p', \int'), \sigma') \xrightarrow{w}{}^{*} ((p'', \int''), \sigma'')$$

Par le Lemme 3.4.20, nous pouvons appliquer l'hypothèse d'induction sur ρ' et donc $\Phi_{\mathsf{sc}}(\rho')$ est un calcul de A. Nous montrons maintenant que $\Phi_{\mathsf{sc}}(((p, \int), \sigma) \xrightarrow{a} \rho')$ est aussi un calcul de A. Par définition il existe une transition τ, menant de $((p, \int), \sigma)$ vers ρ'. Il y a quatre cas possibles :

- Si $\tau = ((p, \int), a, (p', \int')) \in \delta'_\iota$ avec $a \in \Delta_\iota$, alors $\sigma = \sigma'$. Par construction, $(p, a, p') \in \delta_\iota$ mène de $\Phi_{\mathsf{sc}}((p, \int), \sigma))$ à $\Phi_{\mathsf{sc}}(\rho')$.
- Si $\tau = ((p, \int), a, (\gamma, \int), (p', \int')) \in \delta'_c$ avec $a \in \Delta_c$, alors $\sigma(\gamma, \int) = \sigma'$. Par construction, $(p, a, \gamma, p') \in \delta_c$ mène de $\Phi_{\mathsf{sc}}((p, \int), \sigma))$ à $\Phi_{\mathsf{sc}}(\rho')$.
- Si $\tau = ((p, \int), a, (\gamma, \int), (p', \int')) \in \delta'_r$ avec $a \in \Delta_r$, alors $\sigma = \sigma'(\gamma, \int)$. Par construction $(p, a, \gamma, p') \in \delta_r$ mène de $\Phi_{\mathsf{sc}}((p, \int), \sigma))$ à $\Phi_{\mathsf{sc}}(\rho')$.
- Si $\tau = ((p, \int), a, \bot, (p', \int')) \in \delta'_r$ avec $a \in \Delta_r$, alors $\sigma = \sigma = \bot$. Par construction $(p, a, \bot, p') \in \delta_r$ mène de $\Phi_{\mathsf{sc}}((p, \int), \sigma))$ à $\Phi_{\mathsf{sc}}(\rho')$. □

Lemme 3.4.25

Soient A un APV sur l'alphabet structuré Δ et A_{sc} l'APV $\mathsf{confpile}(A)$. La fonction Φ_{sc} des calculs conformes en pile de $\mathsf{confpile}(A)$ vers les calculs de A est injective.

Démonstration. Soient ρ_1 et ρ_2 deux calculs conformes en pile de A_{sc} sur le même mot w. Nous montrons par induction sur la longueur de w que si $\Phi_{\mathsf{sc}}(\rho_1) = \Phi_{\mathsf{sc}}(\rho_2)$, alors $\rho_1 = \rho_2$. Si $w = \varepsilon$ alors $\rho_1 : ((p_1, \int_1), \sigma_1)$ et $\rho_2 : ((p_2, \int_2), \sigma_2)$. Comme $\Phi_{\mathsf{sc}}(\rho_1) = \Phi_{\mathsf{sc}}(\rho_2)$ nous en déduisons que $p_1 = p_2$ et $\pi_1(\sigma_1) = \pi_1(\sigma_2)$, et donc $|\sigma_1| = |\sigma_2|$. Notez que comme ρ_1 et ρ_2 sont des calculs conformes en pile nous avons $\int_i \in \{\circ, \bot\}$ et $\sigma_i \in \mathrm{S}_i^*$ pour $i \in [1, 2]$. Nous pouvons conclure que $\int_1 = \int_2$ et $\sigma_1 = \sigma_2$ car $|\sigma_1| = |\sigma_2|$. Sinon ρ_1 et ρ_2 se décomposent comme suit :

$$((p_i, \int_i), \sigma_i) \xrightarrow{a} \rho'_i \text{ avec } \rho'_i : ((p'_i, \int'_i), \sigma'_i) \xrightarrow{w}{}^{*} ((p''_i, \int''_i), \sigma''_i) \text{ avec } i \in [1, 2]$$

Par le Lemme 3.4.20, nous pouvons appliquer l'hypothèse d'induction sur ρ'_1 et ρ'_2, et donc nous avons $\rho'_1 = \rho'_2$. Nous montrons maintenant que $((p_1, \int_1), \sigma_1) \xrightarrow{a} \rho'_1 = ((p_2, \int_2), \sigma_2) \xrightarrow{a} \rho'_2$. Premièrement notez que si $\Phi_{\mathsf{sc}}(((p_1, \int_1), \sigma_1) \xrightarrow{a} \rho'_1) : \Phi_{\mathsf{sc}}(((p_2, \int_2), \sigma_2) \xrightarrow{a} \rho'_2)$, alors $p_1 = p_2$ et $\pi_1(\sigma_1) = \pi_1(\sigma_2)$, ceci amenant $|\sigma_1| = |\sigma_2|$. Par définition il existe deux transitions τ_1 et τ_2, menant de $((p_i, \int_i), \sigma_i)$ à ρ'_i pour $i \in [1, 2]$:

- Si $\tau_i = ((p_i, \int_i), a, (p'_i, \int'_i)) \in \delta'_\iota$ avec $a \in \Delta_\iota$ alors $\sigma_i = \sigma'_i$ et $\int_i = \int'_i$. Nous concluons que $\int_1 = \int_2$ et $\sigma_1 = \sigma_2$.
- Si $\tau_i = ((p_i, \int_i), a, (\gamma_i, \int_i), (p'_i, \int'_i)) \in \delta'_c$ avec $a \in \Delta_c$ alors $\gamma_1 = \gamma_2$. Supposons que $\int_1 \neq \int_2$. Si $\int_1 = \bot$ et $\int_1 \neq \bot$, alors par la propriété de conforme en pile nous avons $|\sigma_1| \neq |\sigma_2|$, ce qui est une contradiction. Si $\int_1 = \top$ et $\int_2 = \circ$, alors par le Lemme 3.4.22, γ_1 est dépilé dans ρ'_1 mais pas γ_2, ce qui est une contradiction. Nous en déduisons que $\int_1 = \int_2$, et comme $\sigma'_1 = \sigma'_2 = \sigma_i(\gamma_i, \int_i)$, nous concluons que $\sigma_1 = \sigma_2$.

- Si $\tau_i = ((p_i, \int_i), a, (\gamma_i, \int_i'), (p_i', \int_i')) \in \delta_r'$ avec $a \in \Delta_r$, alors $\int_1 = \int_2 = \top$ et $\gamma_1 = \gamma_2$. Comme $\sigma_1'(\gamma_i, \int_i') = \sigma_2'(\gamma_i, \int_i') = \sigma_i$ et $\int_1' = \int_2'$, nous concluons que $\sigma_1 = \sigma_2$.
- Si $\tau_i = ((p_i, \int_i), a, \bot, (p_i', \int_i')) \in \delta_r'$ avec $a \in \Delta_r$ alors $\sigma_i = \sigma_i' = \bot$ et $\int_i = \int_i'$. □

Lemme 3.4.26

Soient A un APV sur l'alphabet structuré Δ et A_{sc} l'APV $\mathsf{confpile}(A)$. La fonction Φ_{sc} des calculs conformes en pile de $\mathsf{confpile}(A)$ vers les calculs de A est surjective.

Démonstration. Soit ρ un calcul de A. Nous faisons la preuve par induction sur la longueur de ρ. Nous considérons en premier que $\rho : (p, \sigma)$ avec $\sigma = (\gamma_i)_{1 \leqslant i \leqslant l}$. Si $\sigma = \bot$, alors il existe un calcul conforme en pile ρ' de A_{sc} de la forme $((p, \bot), \bot)$ tel que $\Phi_{\mathsf{sc}}(\rho') = \rho$. Sinon il existe un calcul conforme en pile ρ' de A_{sc} de la forme $((p, \circ), \sigma')$ tel que $\Phi_{\mathsf{sc}}(\rho') = \rho$ et $\sigma' = (\gamma_i, \int_i)_{1 \leqslant i \leqslant l}$ avec $\int_0 = \bot$ et $\int_i = \circ$ pour $i \in \{2, \ldots, l\}$. Nous considérons maintenant que $\rho \neq (p, \sigma)$, il peut donc se décomposer de la manière suivante :

$$(p, \sigma) \xrightarrow{a} \rho' \text{ avec } \rho' : (p', \sigma') \xrightarrow{w}{}^{*} (p'', \sigma'')$$

Par le Lemme 3.4.20, nous pouvons appliquer l'hypothèse d'induction sur ρ', et donc il existe un calcul conforme en pile ρ'' de A_{sc} tel que $\rho' = \Phi_{\mathsf{sc}}(\rho'')$. Nous montrons maintenant qu'il existe une configuration $((p, \int), \bar{\sigma})$ telle que $\Phi_{\mathsf{sc}}(((p, \int), \bar{\sigma}) \xrightarrow{a} \rho'') = \rho$. Nous appelons $((p', \int'), \bar{\sigma}')$ la configuration $\mathsf{premier}(\rho'')$. Par définition il existe une transition τ menant de (p, σ) à (p', σ'). Nous distinguons quatre cas :

- Si $\tau = (p, a, p') \in \delta_\iota$ avec $a \in \Delta_\iota$ alors $\sigma' = \sigma$. Par construction il existe une transition $((p, \int), a, (p', \int)) \in \delta_\iota'$, et donc $((p, \int), \bar{\sigma}) \xrightarrow{a} \rho''$ est un calcul conforme en pile de A_{sc} avec $\bar{\sigma} = \bar{\sigma'}$ et $\int = \int'$.
- Si $\tau = (p, a, \gamma, p') \in \delta_c$ avec $a \in \Delta_c$ et $\gamma \in \Gamma$ alors $\sigma' = \sigma\gamma$. Notez que par définition, $\bar{\sigma}' \neq \bot$ et donc $\int' \neq \bot$. Par construction il existe $((p, \int), a, (\gamma, \int)(p', \int')) \in \delta_c'$ avec $\int' \in \{\top, \circ\}$.
  - Si $\int' = \top$ alors $((p, \int), \bar{\sigma}) \xrightarrow{a} \rho''$ est un calcul conforme en pile de A_{sc} avec $\bar{\sigma}' = \bar{\sigma}(\gamma, \int)$. De plus nous avons $\int = \bot$ si $\bar{\sigma} = \bot$, $\int = \circ$ s'il existe $r \in \Delta_r$ tel que $w = w_1 r w_2$ et $\int = \top$ sinon.
  - Si $\int' = \circ$, alors $\int \in \{\bot, \circ\}$ et par définition de $\mathbb{S}_\circ^*$, $((p, \int), \bar{\sigma}) \xrightarrow{a} \rho''$ est un calcul conforme en pile de A_{sc} avec $\bar{\sigma}' = \bar{\sigma}(\gamma, \int)$. De plus nous avons $\int = \bot$ si $\bar{\sigma} = \bot$ et $\int = \circ$ sinon.
- Si $\tau = (p, a, \gamma, p')$ avec $a \in \Delta_r$ et $\gamma \in \Gamma$ alors $\sigma = \sigma'\gamma$. Par construction il existe une transition $((p, \int), (\gamma, \int'), a, (p', \int')) \in \delta_r'$ avec $\int = \top$. Donc $((p, \int), \bar{\sigma}) \xrightarrow{a} \rho''$ est un calcul conforme en pile de A_{sc} avec $\bar{\sigma} = \bar{\sigma}'(\gamma, \int')$ et $\int = \top$.
- Si $\tau = (p, a, \bot, p')$ avec $a \in \Delta_r$ alors $\sigma' = \sigma = \bot$. Notez que par définition, $\bar{\sigma}' = \bot$ et donc $\int' = \bot$. Par construction il existe une transition $((p, \int), a, \bot, (p', \int')) \in \delta_r'$ avec $\int = \int' = \bot$. Donc $((p, \int), \bar{\sigma}) \xrightarrow{a} \rho''$ est un calcul de A_{sc} avec $\bar{\sigma} = \bar{\sigma}' = \bot$ et $\int = \bot$. □

Nous pouvons prouver le Théorème 3.4.23 :

Démonstration. [*Théorème 3.4.23*] Pour le premier point il convient de noter que tout calcul $\rho \in \mathsf{ACal}(A_{\mathsf{sc}})$ est un calcul conforme en pile de A_{sc}. Donc par Proposition 3.4.24 et par les Lemmes 3.4.25 et 3.4.26, ρ est un calcul acceptant de A_{sc} sur w si et seulement si $\Phi_{\mathsf{sc}}(\rho)$ est un calcul acceptant de A sur w. Pour le second point, les conditions (1) et (2) de la définition de conforme en pile sont des conséquences du Lemme 3.4.20, avec $Q'_{\not\perp} = (Q \times \{\circ\}) \cup (Q \times \{\top\})$, $Q'_{\perp} = (Q \times \{\bot\})$, $\Gamma'_{\not\perp} = (\Gamma \times \{\circ\}) \cup (\Gamma \times \{\top\})$ et $\Gamma'_{\perp} = (\Gamma \times \{\bot\})$. La condition (3) est une conséquence directe de la construction. □

3.4.4 La procédure complète

Nous pouvons maintenant montrer comment émonder un APV en combinant les constructions présentées jusqu'à présent. Nous considérons la construction emonder définie par :

$$\mathsf{emonder}(A) = \mathsf{retracter} \circ \mathsf{emonder}_{\mathsf{bi}} \circ \mathsf{etendre} \circ \mathsf{confpile}(A)$$

et formulons ses principales propriétés.

Théorème 3.4.27

Soient A un APV sur l'alphabet Δ et A_{em} l'APV $\mathsf{emonder}(A)$. L'APV A_{em} peut être construit en temps polynomial et satisfait :

1. il y a une bijection entre les calculs de $\mathsf{ACal}(A)$ et les calculs de $\mathsf{ACal}(A_{\mathsf{em}})$, et donc $\mathcal{L}(A) = \mathcal{L}(A_{\mathsf{em}})$,
2. A_{emonder} est émondé.

Démonstration. Il est possible de prouver ce résultat en utilisant les théorèmes et propositions listés dans la table de la Section 3.7. □

3.5 Automate à pile visible déterministe et émondé

Nous avons prouvé dans la section précédente qu'il est toujours possible, étant donné un APV, de construire un APV équivalent émondé. De plus, nous avons montré à la Section 1.4.2 qu'il est toujours possible de construire un APV avec le même langage et déterministe. Dans cette section, nous allons montrer comment construire un APV à la fois émondé et déterministe. Ceci n'est pas un corollaire immédiat des précédents résultats, car la construction reduire introduit du non-déterminisme, et la construction d'un APV déterministe présentée dans [AM04] ne préserve pas la co-réduction.

3.5.1 La déterminisation préserve la réduction et la rétractabilité

Nous prouvons maintenant que determiniser préserve les propriétés de faible réduction et de rétractabilité.

Proposition 3.5.1

Soit A un APV sur l'alphabet structuré Δ. Si A est faiblement réduit, alors $A_{\mathsf{det}} = \mathsf{determiniser}(A)$ est faiblement réduit.

Démonstration. Soit ρ' un calcul initialisé de A_{det}. Nous devons montrer que ρ' peut être complété dans un calcul acceptant de A_{det}. Nous appelons w_1 le mot associé au calcul ρ'.

Par le Lemme 1.4.18, Il existe un calcul initialisé ρ de A sur le mot w_1. Comme A est faiblement co-réduit, le calcul ρ peut-être complété par un calcul ρ_2 sur un mot w_2 dans un calcul acceptant de A. Par conséquent, nous avons $w_1 w_2 \in \mathcal{L}(A)$. Par la correction de la procédure de déterminisation nous avons $w_1 w_2 \in \mathcal{L}(A_{\mathsf{det}})$. Il existe donc un calcul ρ'' de A_{det} sur $w_1 w_2$. Comme A_{det} est déterministe, le préfixe de ρ'' sur w_1 doit être exactement ρ'. Ceci prouve le résultat. □

Proposition 3.5.2

Soit A un APV sur l'alphabet structuré Δ. Si A est rétractable, alors A_{det} le APV $\mathsf{determiniser}(A)$ est rétractable.

Démonstration. Nous notons $A_{\mathsf{det}} = (Q', I', F', \delta', \Gamma')$. Par le Lemme 1.4.17 nous avons :

$$\text{Pour tout } S \in Q', \text{ si } S \in \mathsf{piege}(A_{\mathsf{det}}) \text{ alors } \exists (p,q) \in S \text{ tel que } q \in \mathsf{piege}(A) \tag{3.3a}$$

$$\text{Pour tout } S \in Q', \text{ si } S \in \mathsf{bordure}(A_{\mathsf{det}}) \text{ alors } \exists (p,q) \in S \text{ tel que } q \in \mathsf{bordure}(A) \tag{3.3b}$$

Nous prouvons chaque condition de la Définition 3.4.9 : la condition (1) est évidente car nous avons $\mathcal{L}(A_{\mathsf{det}}) = \mathcal{L}(A)$. Les conditions $(2) - (4)$ sont des conséquences de l'Énoncé (3.3a). Pour la condition (5), nous définissons un calcul initialisé ρ de A_{det} sur le mot w avec $\mathsf{dernier}(\rho) = (S, \sigma)$ tel que S est un état de bord de A_{det}. Il existe un calcul ρ' de A sur w avec $\mathsf{dernier}(\rho) = (q, \sigma')$ et $(p,q) \in S$. Par l'Énoncé (3.3b), q est un état de bord. Donc par la définition de la rétractabilité, il existe un calcul $\bar{\rho}$ de A sur $\bar{r}^k$ tel que $\rho'\bar{\rho}$ est un calcul acceptant de A avec $k \in \mathbb{N}$. Par conséquent, nous avons $w\bar{r}^k \in \mathcal{L}(A)$. Par la correction de la procédure de déterminisation nous avons $w\bar{r}^k \in \mathcal{L}(A_{\mathsf{det}})$. Il existe donc un calcul acceptant ρ'' de A_{det} sur ce mot. Comme A_{det} est déterministe, le préfixe ρ'' sur w est exactement ρ. Ceci prouve le résultat. □

3.5.2 Construction d'un APV déterministe et émondé.

Nous pouvons maintenant montrer comment construire un APV émondé et déterministe en utilisant une combinaison des constructions présentées jusqu'à présent. Nous considérons la composition suivante :

$$\mathsf{det\text{-}em} = \mathsf{retracter} \circ \mathsf{coreduire} \circ \mathsf{determiniser} \circ \mathsf{reduire} \circ \mathsf{etendre} \circ \mathsf{confpile}$$

Cette composition permet de construire un APV à la fois déterministe et émondé :

Théorème 3.5.3

Soit A un APV sur l'alphabet structuré Δ. L'APV $\mathsf{det\text{-}em}(A)$ est déterministe, émondé et satisfait $\mathcal{L}(A) = \mathcal{L}(\mathsf{det\text{-}em}(A))$.

Démonstration. Nous pouvons prouver ce résultat en utilisant les résultats listés dans la table de la Section 3.7. □

3.5.3 Une borne inférieure pour les entrées déterministes

Nous pourrions tenter d'émonder un APV déterministe en temps polynomial, tout en préservant sa nature déterministe. Nous allons prouver maintenant que ce n'est pas possible et qu'il n'existe pas de procédure émondant un APV en temps polynomial et préservant le déterminisme. Pour cela nous allons présenter un famille d'automates à pile visible déterministes qu'il n'est pas possible d'émonder en temps polynomial tout en préservant le déterminisme.

Soit $\Delta = \Delta_c \cup \Delta_r \cup \Delta_\iota$ un alphabet structuré tel que $\Delta_c = \{c_1, c_2\}$, $\Delta_r = \{r\}$, $\Delta_\iota = \{a\}$ et $\Gamma = \{\gamma_1, \gamma_2\}$. Pour tout $k \in \mathbb{N}$ nous appelons $A_k = (Q, I, F, \Gamma, \delta)$ l'APV défini comme suit :

- $Q = \{q\} \cup \{q_i \mid i \in \{0, \ldots, k\}\}$
- $I = \{q\}$
- $F = \{q_k\}$
- $\delta = \delta_c \cup \delta_r \cup \delta_r^{\perp} \cup \delta_\iota$ avec :

$$
\begin{array}{lcll}
- \ \delta_c & = & \{(q, c_1, \gamma_1, q), (q, c_2, \gamma_2, q)\} & \\
- \ \delta_r & = & \{(q_i, r, \gamma, q_{i+1}) \mid i \in \{0, \ldots, k-2\}, \gamma \in \Gamma\} & \cup \\
 & & \{(q_{k-1}, r, \gamma_1, q_k)\} & \cup \\
 & & \{(q_k, r, \gamma, q_k) \mid \gamma \in \Gamma\} & \\
- \ \delta_r^{\perp} & = & \varnothing & \\
- \ \delta_\iota & = & \{(q, a, q_0)\} &
\end{array}
$$

L'APV A_k est représenté sur la figure Figure 3.6. Il est facile de montrer que pour tout $k \in \mathbb{N}$ nous avons :

$$\mathcal{L}(A_k) = \{uc_1vaw \mid u, v \in \Delta_c^*, r \in \Delta_r^*, |v| = k-1, k \leqslant |w| \leqslant |u| + k\}$$

Nous allons maintenant montrer que pour tout $k \in \mathbb{N}$ il n'existe pas de APV $\bar{A}_k$ déterministe et émondé tel que $\mathcal{L}(A_k) = \mathcal{L}(\bar{A}_k)$ et la taille de $\bar{A}_k$ est polynomiale dans la taille de A_k.

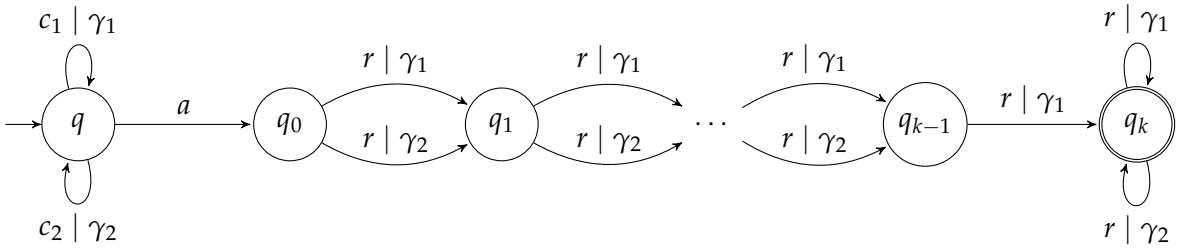


FIGURE 3.6 – L'APV A_k.

Lemme 3.5.4

Soient $k \in \mathbb{N}_{>0}$ un entier et $\rho : (q, \perp) \xrightarrow{w}{}^*(q', \sigma)$ un calcul initialisé de A_k tel que $w = w'a$ avec $w' \in \Delta^*$. Il existe un calcul ρ' de A_k tel que $\rho\rho'$ est un calcul acceptant de A_k si et seulement si $w' = uc_1v$ avec $u, v \in \Delta_c^*$ et $|v| = k-1$.

Démonstration. Premièrement notez que par construction de A_k nous avons $q' = q_0$. Le seul moyen d'atteindre un état final depuis (q_0, σ) est de considérer la pile $\sigma = \sigma_1\gamma_1\sigma_2$ avec $u, v \in \Gamma^*$ et $|\sigma_2| = k-1$. Comme le seul moyen d'empiler le symbole γ_1 (resp. γ_2) est de lire la lettre c_1 (resp. c_2) nous pouvons conclure qu'il existe ρ' de A_k tel que $\rho\rho'$ est acceptant si et seulement si ρ est sur un mot $w' = uc_1v$ avec $u, v \in \Delta_c^*$ et $|v| = k-1$. □

Pour tout $k \in \mathbb{N}$ nous considérons $\bar{A}_k = (\bar{Q}, \bar{I}, \bar{F}, \bar{\Gamma}, \bar{\delta})$ comme étant un APV réduit et déterministe tel que $\mathcal{L}(\bar{A}_k) = \mathcal{L}(A_k)$.

Lemme 3.5.5

Pour tout calcul initialisé $\rho : (p, \perp) \xrightarrow{w'}{}^*(p', \sigma)$ de $\bar{A}_k$ tel qu'il existe une transition $(p', a, p'') \in \bar{\delta}_\iota$, alors $w' = uc_1v$ avec $u, v \in \Delta_c^*$ et $|v| = k-1$.

Démonstration. Comme a est une lettre interne, $(p, \perp) \xrightarrow{w'}{}^*(p', \sigma) \xrightarrow{a}(p'', \sigma)$ est un calcul de $\bar{A}_k$ sur $w'a$. Comme $\bar{A}_k$ est réduit, par définition il existe un calcul ρ' de $\bar{A}_k$ tel que $\rho\rho'$ est un calcul acceptant de $\bar{A}_k$. Par le Lemme 3.5.4 et comme $\mathcal{L}(A_k) = \mathcal{L}(\bar{A}_k)$ nous concluons que $w' = uc_1v$ avec $u, v \in \Delta_c^*$ et $|v| = k-1$. □

Lemme 3.5.6

L'APV $\bar{A}_k$ a au moins 2^k états.

Démonstration. Afin de prouver ce résultat nous définissons l' AF $B_k = (Q', I', F', \delta')$ sur l'alphabet $\Sigma = \{c_1, c_2\}$ et $Q' = \{p, p' \in \bar{Q} \mid \exists (p, c, \gamma, p') \in \bar{\delta}_c\} \cup \{p \in \bar{Q} \mid \exists (p, a, p') \in \bar{\delta}_\iota\}$, $I' = \bar{I}$, $F' = \{p \in \bar{Q} \mid (p, a, p') \in \bar{\delta}_\iota\}$, et $\delta' = \{(p, c, p') \mid (p, c, \gamma, p') \in \bar{\delta}_c\}$.

Premièrement notez que B_k est défini par restriction sur $\bar{A}_k$, où les transitions de B_k sont les transitions d'empilement de $\bar{A}_k$ dans lesquelles le symbole de pile est supprimé. De plus comme $\bar{A}_k$ est déterministe, B_k l'est aussi. Les états finaux de B_k sont les états de $\bar{A}_k$ avec une transition sortante sur a (et par hypothèse sur le langage de $\bar{A}_k$, ont au moins un transition d'empilement entrante). Donc par le Lemme 3.5.5, nous avons $\mathcal{L}(B_k) = \{uc_1v \in \Delta^* \mid u, v \in \Delta_c^* \text{ et } |v| = k-1\}$. Par le Théorème 1.3.3, B_k a au moins 2^k états. Comme B_k est défini comme une restriction de $\bar{A}_k$, celui-ci possède aussi a au moins 2^k états. □

Comme conséquence immédiate du Lemme 3.5.6 nous obtenons le résultat souhaité :

Théorème 3.5.7

Il n'existe pas de procédure émondant un APV en temps polynomial et préservant le déterminisme.

3.6 Application aux automates à pile visible avec coûts

Nous montrons dans cette section que notre procédure d'émondage peut être appliquée aux APV avec coûts, comme énoncé dans le théorème suivant :

Théorème 3.6.1

Soient $\mathcal{S} = (E, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ un semi-anneau et S un $\mathcal{S}$-APV. Il est possible de construire en temps polynomial un $\mathcal{S}$-APV émondé S' tel que $\mathsf{fonction}(S') = \mathsf{fonction}(S)$.

Il est donc possible d'émonder un APV à coûts en préservant la fonction de celui-ci. Nous utiliserons ce résultat dans les prochains chapitres afin de ne considérer que des automates émondés. A noter les TF et TPV peuvent être vus comme des automates à coûts sur le semi-anneau des ensembles de mots, et les résultats que nous montrons dans cette section s'appliquent aussi à ces modèles. Nous étendons maintenant les constructions présentées aux sections précédentes aux APV à coûts, en précisant à chaque fois comment les coûts sont calculés pour chaque transition. Nous ne parlerons pas de déterminisation car cela nécessiterait des outils plus complexes. En effet, déterminiser un automate à pile visible à coût tout en préservant son coût n'est pas toujours possible et nous n'aborderons pas ce sujet ici. Le Lemme 3.6.7 résume les propriétés de ces constructions. Le Théorème 3.6.1 en est une conséquence directe.

Définition 3.6.2

Soient $\mathcal{S} = (E, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ un semi-anneau et $S = (A, \lambda)$ un $\mathcal{S}$-APVbi sur l'alphabet structuré Δ, nous définissons le $\mathcal{S}$-APVbi $\mathsf{red}(S) = (A', \lambda')$ sur Δ, avec :

$$\circ\ A' = \mathsf{reduire}(A) = (Q, I, F, \Gamma, \delta)$$

$$\circ\ \lambda'(\tau) = \begin{cases} \lambda((p,a,p')) & \text{si } \tau = ((p,q),a,(p',q')) \in \delta_\iota, \\ \lambda((p,c,\gamma,p')) & \text{si } \tau = ((p,q),c,(\gamma,q),(p',q')) \in \delta_c, \\ \lambda((p,\gamma,r,p')) & \text{si } \tau = ((q,q),(\gamma,q'),r,(p',q')) \in \delta'_r. \end{cases}$$

Définition 3.6.3

Soient $\mathcal{S} = (E, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ un semi-anneau et $S = (A, \lambda)$ un $\mathcal{S}$-APVbi sur l'alphabet structuré Δ, nous définissons le $\mathcal{S}$-APVbi $\mathsf{cor}(S) = (A', \lambda')$ sur Δ, avec :

$$\circ\ A' = \mathsf{coreduire}(A) = (Q, I, F, \Gamma, \delta)$$

$$\circ\ \lambda'(\tau) = \begin{cases} \lambda((q,a,q')) & \text{si } \tau = ((p,q),a,(p',q')) \in \delta_\iota, \\ \lambda((q,c,\gamma,q')) & \text{si } \tau = ((p,q),c,(\gamma,p),(q',q')) \in \delta_c, \\ \lambda((q,\gamma,r,q')) & \text{si } \tau = ((p,q),(\gamma,p'),r,(p',q')) \in \delta_r. \end{cases}$$

Définition 3.6.4

Soit $\mathcal{S} = (E, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ un semi-anneau et $S = (A, \lambda)$ un $\mathcal{S}$-APV sur l'alphabet structuré Δ tel que A est conforme en pile, $\mathsf{etendre}(S) = (A', \lambda')$ sur Δ^{ext} est défini par :

$$\circ\ A' = \mathsf{etendre}(A) = (Q, I, F, \Gamma, \delta)$$

$$\circ\ \lambda'(\tau) = \begin{cases} \lambda((p,a,q)) & \text{si } \tau = (p,a,q) \in \delta_\iota \text{ et } a \in \Delta_\iota, \\ \lambda((p,\bot,r,q)) & \text{si } \tau = (p,a_r,q) \in \delta_\iota \text{ et } a_r \in \Delta_r, \\ \lambda((p,c,\gamma,q)) & \text{si } \tau = (p,c,\gamma,q) \in \delta_c, \\ \lambda((p,\gamma,r,q)) & \text{si } \tau = (p,\gamma,r,q) \in \delta_r \text{ et } r \neq \bar{r}, \\ \mathbf{1} & \text{si } \tau = (\bar{f}_\bot,\gamma,\bar{r},\bar{f}_\int) \in \delta_r \text{ ou } \tau = (p,\gamma,\bar{r},\bar{f}_\int) \in \delta_r. \end{cases}$$

Définition 3.6.5

Soient $\mathcal{S} = (E, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ un semi-anneau et $S = (A, \lambda)$ un $\mathcal{S}$-APVbi sur l'alphabet structuré Δ^{ext} tel que A est rétractable, $\mathsf{retracter}(S) = (A', \lambda')$ sur Δ est défini par :

$$\circ\ A' = \mathsf{retracter}(A) = (Q, I, F, \Gamma, \delta)$$

$$\circ\ \lambda'(\tau) = \begin{cases} \lambda((p,a,q)) & \text{si } \tau = (p,a,q) \in \delta_\iota, \\ \lambda((p,a_r,q)) & \text{si } \tau = (p,\bot,r,q) \in \delta_\iota \\ \lambda((p,c,\gamma,q)) & \text{si } \tau = (p,c,\gamma,q) \in \delta_c, \\ \lambda((p,\gamma,r,q)) & \text{si } \tau = (p,\gamma,r,q) \in \delta_r. \end{cases}$$

Définition 3.6.6

Soient $\mathcal{S} = (E, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ un semi-anneau et $S = (A, \lambda)$ un $\mathcal{S}$-APV sur l'alphabet structuré Δ, nous définissons le $\mathcal{S}$-APV $\mathsf{confpile}(S) = (A', \lambda')$ sur Δ avec :

$$\begin{aligned}
\circ\ A' &= \mathsf{confpile}(A) = (Q, I, F, \Gamma, \delta) \\
\circ\ \lambda'(\tau) &= \begin{cases}
\lambda((p,a,q)) & \text{si } \tau = ((p,\int), a, (q,\int)) \in \delta_\iota \\
\lambda((p,\bot,r,q)) & \text{si } \tau = ((p,\bot),\bot,r,(q,\bot)) \in \delta_\iota \text{ et } r \in \Delta_r, \\
\lambda((p,c,\gamma,q)) & \text{si } \tau = ((p,\int), c, (\gamma,\int), (q,\int')) \in \delta_c, \\
\lambda((p,\gamma,r,q)) & \text{si } \tau = ((p,\top),(\gamma,\int),r,(q,\int)) \in \delta_r
\end{cases}
\end{aligned}$$

Lemme 3.6.7

Soient $\mathcal{S}$ un semi-anneau et S un $\mathcal{S}$-APV. Considérons les $\mathcal{S}$-APV $S_{\mathsf{sc}} = \mathsf{confpile}(S)$, $S_{\mathsf{ext}} = \mathsf{etendre}(S_{\mathsf{sc}})$, $S_{\mathsf{red}} = \mathsf{reduire}(S_{\mathsf{ext}})$, $S_{\mathsf{cor}} = \mathsf{coreduire}(S_{\mathsf{red}})$ et $S_{\mathsf{ret}} = \mathsf{confpile}(A_{\mathsf{cor}})$. Nous avons les égalités $\mathsf{fonction}(S) = \mathsf{fonction}(S_{\mathsf{sc}}) = \mathsf{fonction}(S_{\mathsf{ext}}) = \mathsf{fonction}(S_{\mathsf{red}}) = \mathsf{fonction}(S_{\mathsf{cor}}) = \mathsf{fonction}(S_{\mathsf{cor}})$.

Démonstration. Par définition le poids d'un calcul d'un $\mathcal{S}$-APV dépend uniquement du calcul de l'APV sous-jacent. De plus, il est possible de vérifier que toutes les bijections des calculs produites par les différentes constructions présentées jusqu'à présent préservent les transitions, *i.e.* chaque transition de l'APV que nous construisons est produite à partir d'une transition de l'APV originel. □

3.7 Résumé

Nous concluons ce chapitre en résumant toutes les constructions présentées, ainsi que leurs principales propriétés.

Algorithme	Profil	Nécessite	Préserve	Assure
reduire	APVbi → APVbi		Rétractabilité forte (Proposition 3.4.15) Co-réduction (Proposition 3.3.11)	Réduction (Théorème 3.3.9)
coreduire	APVbi → APVbi		Rétractabilité (Proposition 3.4.16) Réduction (Proposition 3.3.19) Déterminisme (Proposition 3.3.20)	Co-réduction (Théorème 3.3.17)
determiniser	APV → APV		Réduction (Proposition 3.5.1) Rétractabilité (Proposition 3.5.2)	Déterminisme (Théorème 1.4.16)
etendre	APV → APVbi	Conformité en pile		Rétractabilité forte (Théorème 3.4.10)
retracter	APVbi → APV	Rétractabilité	Réduction (Proposition 3.4.15) Co-réduction (Proposition 3.4.16)	
confpile	APV → APV			Conformité en pile (Théorème 3.4.23)

En utilisant ces constructions nous avons défini trois procédures permettant respectivement d'émonder un APVbi, d'émonder un APV, et d'émonder et déterminiser un APV :

$$\begin{aligned}
\mathsf{emonder}_{\mathsf{bi}} &= \mathsf{coreduire} \circ \mathsf{reduire} \\
\mathsf{emonder} &= \mathsf{retracter} \circ \mathsf{emonder}_{\mathsf{bi}} \circ \mathsf{etendre} \circ \mathsf{confpile} \\
\mathsf{det\text{-}em} &= \mathsf{retracter} \circ \mathsf{coreduire} \circ \mathsf{determiniser} \circ \mathsf{reduire} \circ \mathsf{etendre} \circ \mathsf{confpile}
\end{aligned}$$

Chapitre 4

Finitude et K-borne des automate à pile visible à coûts

Nous avons présenté dans le Chapitre 2 l'idée d'ajouter des coûts sur les transitions d'un automate. C'est une notion plutôt standard, et qui a été étudiée intensivement par le passé dans le cas des automates de mots, par exemple dans [Sch61], [Eil76] et [SS78], mais aussi sur les automates d'arbres, par exemple dans [BR82] et [Sei94b]. Ce modèle présente aussi un intérêt pratique car il permet de représenter des comportements quantitatifs, comme par exemple des mouvements de flux.

Dans ce chapitre nous allons étudier plusieurs modèles d'automates à coûts sur des semi-anneaux différents.

Nous allons en premier considérer les APV à coûts sur le semi-anneau des entiers naturels muni des opérateurs de multiplication et d'addition. Plusieurs questions se posent naturellement. Premièrement l'infinité du coût, c'est à dire décider si le coût de l'automate est fini ou non. Ce problème a déjà été étudié dans le cas des automates de mots dans [MS77] par Arnaldo Mandel et Imre Simon. Cependant nous considérerons dans ce manuscrit la version de la preuve présentée dans [SdS08]. Jacques Sakarovitch et Rodrigo De Souza présentent deux propriétés, et montrent que le coût d'un automate de mots est infini si et seulement s'il satisfait l'une des deux propriétés. Nous allons étendre ce résultat aux APV en présentant une extension de ces propriétés, et montrerons qu'il est possible de décider de leur présence en temps polynomial. Un deuxième problème intéressant est celui de décider étant donné un entier k si le coût de l'automate est inférieur à k. Nous étudierons deux résultats différents dépendants de la nature de k. Nous montrerons ainsi que le problème est EXPTIME-complet si k est donné en entrée, et peut être résolu en temps polynomial si k est fixé. Dans le premier cas nous utiliserons une procédure de saturation d'un ensemble fini de matrices, et dans le deuxième cas nous étendrons un algorithme proposé par Helmut Seidl dans [Sei90] pour décider si l'ambiguïté d'un automate d'arbres est bornée par k.

Dans une deuxième partie, nous exposerons une extension de ces résultats aux automates d'arbres à coûts sur le semi-anneau des entiers, en utilisant l'équivalence entre ce modèle et celui des APV avec coûts présentée dans le Chapitre 2.

Enfin, nous parlerons du semi-anneau Max-plus dans la troisième partie, c'est-à-dire le semi-anneau des entiers équipé des opérateurs max et $+$.

Plan du chapitre Nous allons tout d'abord présenter le semi-anneau des entiers naturels et les résultats existants dans la Section 4.1. Dans la Section 4.2, nous donnerons une caractérisation du coût infini des APV à coûts sur ce semi-anneau et étudierons les problèmes de borne dans la Section 4.3. Enfin, nous étendrons ces résultats à la Section 4.4 aux automates d'arbres à coûts

et nous traiterons du semi-anneau Max-plus dans la Section 4.5.

4.1 État de l'art pour le semi-anneau des entiers naturels

Dans cette section, nous allons traiter du semi-anneau des entiers naturels et de problèmes de bornes liés à celui-ci.

Semi-anneau des entiers naturels Le semi-anneau des entiers naturels $\mathcal{N}$ est le semi-anneau $(\mathbb{N}, +, \times, 0, 1)$ avec $+$ l'addition et $\times$ la multiplication.

Par exemple, les APV (resp. AA) à coûts sur le semi-anneau des entiers naturels seront appelés $\mathcal{N}$-APV (resp. $\mathcal{N}$-AA). Nous considérons les deux problèmes suivants :

Étant donné un $\mathcal{N}$-APV S, existe-t-il un entier k tel que les coûts de tous les mots acceptés par S sont inférieurs à k ?

Étant donnés un entier k et un $\mathcal{N}$-APV S, existe-t-il un mot accepté avec un coût supérieur à k ?

Nous allons maintenant étudier le premier problème pour les automates finis et d'arbres.

4.1.1 Décider si le coût d'un automate fini est infini

Dans [MS77], Arnaldo Mandel et Imre Simon donnent une caractérisation des semi-groupes finiments engendrés par le semi-anneau des matrices $n \times n$ d'entiers naturels de taille finie. Dans [DS08], Rodrigo De Souza repense ce résultat pour les automates finis à coûts en présentant une propriété directement adaptée du résultat original. Elle est nécessaire et suffisante à l'infinité du coût d'un $\mathcal{N}$-AF $S = (A, \lambda)$ émondé. Cette propriété est la suivante :

Propriété S1 S a la propriété **(S1)** s'il existe $u \in \Sigma^*$ tel que :

$$\rho_1 : p \xrightarrow{u,m_1}{}^* p, \rho_2 : p \xrightarrow{u,m_2}{}^* q, \rho_3 : q \xrightarrow{u,m_3}{}^* q \in \mathsf{Cal}_u(S)$$

avec $\rho_1 \neq \rho_2$ ou $m_1 > 1$

Intuitivement, cette propriété permet de produire un coût non borné de deux manières différentes. Comme S est émondé, il existe deux calculs $\rho_i : p_i \xrightarrow{v,m}{}^* p \in \mathsf{Cal}_v(A)$ et $\rho_f : q \xrightarrow{w,m'}{}^* q \in \mathsf{Cal}_v(A)$ tels que p_i est initial et p_f est final. Si $m_1 > 1$ et $\rho_1 = \rho_2$, alors pour tout $k \in \mathbb{N}$ nous avons $\mathsf{cout}(\rho_i \rho_1^k \rho_f) = m m_1^k m'$ et donc le coût de S est infini. Si $\rho_1 \neq \rho_2$, alors pour tout $k \in \mathbb{N}$ nous avons $\mathsf{cout}(v u^k w) \geqslant mkm'$ car les calculs $\rho_1^i \rho_2 \rho_3^{k-i-1}$ avec $i \in [1, k-1]$ sont tous différents, et donc le coût de S est infini.

Cette propriété nous intéresse car elle caractérise simplement l'infinité du coût d'un automate fini. Nous allons étendre ce résultat aux $\mathcal{N}$-APV à la Section 4.2.

4.1.2 Décider si le coût d'un automate d'arbres est infini

Dans [Sei94a], Helmut Seidl présente un modèle d'automate d'arbres à coûts avec des polynômes sur le semi-anneau des entiers. Toutes les transitions de l'automate sont équipées d'un polynôme où chaque variable est associée à un des fils (par exemple le polynôme $2 \times x_1 \times x_2 + x_1^2 \times x_2$ et la transition (q, a, p_1, p_2), où la variable x_1 est liée au fils p_1 et la variable x_2 au fils p_2). Pour obtenir le coût d'un calcul il suffit de composer les polynômes de toutes les transitions et d'évaluer le résultat. Le coût de l'automate est simplement l'ensemble des coûts de ses calculs

acceptants. Ce modèle est donc différent car dans notre cas le coût d'un automate est défini à partir du coût des mots, eux-mêmes calculés à partir de ceux de l'ensemble des calculs sur ce mot. lieu nécessaire de calculer le coût du mot avant de calculer celui de l'automate. Sur ce modèle il décide ensuite si l'ensemble de coûts défini est fini ou non. Étant donné un $\mathcal{N}$-AA $S = (A, \lambda)$, nous appelons $\mathsf{coutmax}(S)$ la valeur définie comme suit :

$$\mathsf{coutmax}(S) = \sup_{\rho \in \mathsf{ACal}(A)} \mathsf{cout}(\rho)$$

De plus, nous pouvons définir l'ambiguïté de A et le coût de S de la manière suivante :

$$\mathsf{amb}(A) = \sup_{w \in \mathcal{L}(A)} \sum_{\rho \in \mathsf{ACal}_w(A)} 1$$

$$\mathsf{cout}(S) = \sup_{w \in \mathcal{L}(A)} \sum_{\rho \in \mathsf{ACal}_w(A)} \mathsf{cout}(\rho)$$

D'après ces équations il est facile de voir que $\mathsf{cout}(S) \geqslant \max(\mathsf{amb}(A), \mathsf{coutmax}(S))$ car $\mathsf{cout}(\rho) > 0$ et $cout(S) \leqslant \mathsf{amb}(A) * \mathsf{coutmax}(S)$. Nous pouvons donc conclure que :

$$\max(\mathsf{amb}(A), \mathsf{coutmax}(S)) \leqslant \mathsf{cout}(S) \leqslant \mathsf{amb}(A) * \mathsf{coutmax}(S)$$

Nous obtenons donc que $\mathsf{cout}(S) < +\infty$ si et seulement si $\mathsf{amb}(A) < +\infty$ ou $\mathsf{coutmax}(S) < +\infty$. Nous savons grâce à [Sei94b] et [Sei89] qu'il est possible de décider en temps polynomial si $\mathsf{coutmax}(S)$ est fini et si $\mathsf{amb}(A)$ est fini respectivement. Grâce à la construction APV2AA produisant un automate d'arbre depuis un automate à pile visible et présentée au Chapitre 2 nous obtenons donc une procédure pour décider si le coût d'un $\mathcal{N}$-APV est fini ou non,

Nous allons cependant proposer une méthode directe basée sur les automates à pile visible et un algorithme plus intuitif que ce qui est présenté dans [Sei94b] et [Sei89] .

4.2 Décidabilité du coût infini des automates à pile visible sur $\mathcal{N}$

Dans cette section, nous allons considérer dans un premier temps les automates à pile visible réduits et bien imbriqués. Nous caractérisons l'infinité de ce modèle en définissant deux propriétés sur les coûts, puis nous définissons un algorithme dans PTIME basé sur ces propriétés pour décider si le coût d'un $\mathcal{N}$-APVbi réduit est fini.

4.2.1 Caractérisation

Soit $S = (A, \lambda)$ un $\mathcal{N}$-APVbi avec $A = (Q, I, F, \Gamma, \delta)$. Nous introduisons deux propriétés sur les $\mathcal{N}$-APVbi, **(S1)** et **(S2)**.

Propriété S1 S a la propriété **(S1)** s'il existe $u \in \Delta_{\mathsf{bi}}^*$ tel que :

$$\rho_1 : (p, \bot) \xrightarrow{u, m_1} {}^*(p, \bot), \rho_2 : (p, \bot) \xrightarrow{u, m_2} {}^*(q, \bot), \rho_3 : (q, \bot) \xrightarrow{u, m_3} {}^*(q, \bot) \in \mathsf{Cal}_u(A)$$

avec $\rho_1 \neq \rho_2$ ou $m_1 > 1$ (voir la Figure 4.1).

Pour plus de simplicité dans les preuves, nous donnons deux propriétés équivalentes.

Propriété S1.a S a la propriété **(S1.a)** s'il existe $u \in \Delta_{\mathsf{bi}}^*$ tel que :

$$\rho_1 : (p, \bot) \xrightarrow{u, m_1} {}^*(p, \bot) \in \mathsf{Cal}_u(A)$$

avec $m_1 > 1$.

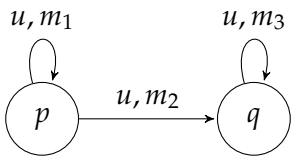


FIGURE 4.1 – La propriété **(S1)**.

Propriété S1.b S a la propriété **(S1.b)** s'il existe $u \in \Delta^*_{\mathsf{bi}}$ tel que :

$\rho_1 : (p, \bot) \xrightarrow{u,n_1}{}^*(p, \bot), \rho_2 : (p, \bot) \xrightarrow{u,m_2}{}^*(q, \bot), \rho_3 : (q, \bot) \xrightarrow{u,m_3}{}^*(q, \bot) \in \mathsf{Cal}_u(A)$

avec $\rho_1 \neq \rho_2$.

Intuitivement, les propriétés **S1.a** et **S1.b** sont équivalentes à la propriété **S1** car il est possible de fusionner les calculs ρ_1 et ρ_2 en un seul calcul si $\rho_1 = \rho_2$.

Propriété S2 S a la propriété **(S2)** s'il existe $uv, w \in \Delta^*_{\mathsf{bi}}$ tels que $u \notin \Delta^*_{\mathsf{bi}}$ et :

$\rho_1 : (p, \bot) \xrightarrow{u,m_1}{}^*(p, \sigma_1), \rho_2 : (p, \bot) \xrightarrow{u,m_2}{}^*(q, \sigma_2), \rho_3 : (q, \bot) \xrightarrow{u,m_3}{}^*(q, \sigma_3) \in \mathsf{Cal}_u(A)$

$\rho : (q, \bot) \xrightarrow{w,m}{}^*(q', \bot) \in \mathsf{Cal}_w(A)$

$\rho'_3 : (q', \sigma_3) \xrightarrow{v,m'_3}{}^*(q', \bot), \rho'_2 : (q', \sigma_2) \xrightarrow{v,m'_2}{}^*(p', \bot), \rho'_1 : (p', \sigma_1) \xrightarrow{v,m'_1}{}^*(p', \bot) \in \mathsf{Cal}_v(A)$

avec $\rho_1 \neq \rho_2$ ou $\rho'_1 \neq \rho'_2$ ou $m_1 > 1$ ou $m'_1 > 1$ (voir la Figure 4.2).

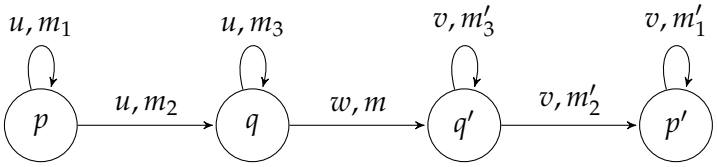


FIGURE 4.2 – La propriété **(S2)**.

Pour plus de simplicité dans les preuves, nous donnons deux propriétés équivalentes.

Propriété S2.a S a la propriété **(S2.a)** s'il existe $uv, w \in \Delta^*_{\mathsf{bi}}$ tels que $u \notin \Delta^*_{\mathsf{bi}}$ et :

$\rho_1 : (p, \bot) \xrightarrow{u,m_1}{}^*(p, \sigma_1) \in \mathsf{Cal}_u(A)$

$\rho : (p, \bot) \xrightarrow{w,m}{}^*(p', \bot) \in \mathsf{Cal}_w(A)$

$\rho'_1 : (p', \sigma_1) \xrightarrow{v,m'_1}{}^*(p', \bot) \in \mathsf{Cal}_v(A)$

avec $m_1 > 1$ ou $m'_1 > 1$.

Propriété S2.b S a la propriété **(S2.b)** s'il existe $uv, w \in \Delta^*_{\mathsf{bi}}$ tels que $u \notin \Delta^*_{\mathsf{bi}}$ et :

$\rho_1 : (p, \bot) \xrightarrow{u,m_1}{}^*(p, \sigma_1), \rho_2 : (p, \bot) \xrightarrow{u,m_2}{}^*(q, \sigma_2), \rho_3 : (q, \bot) \xrightarrow{u,m_3}{}^*(q, \sigma_3) \in \mathsf{Cal}_u(A)$

$\rho : (q, \bot) \xrightarrow{w,m}{}^*(q', \bot) \in \mathsf{Cal}_w(A)$

$\rho'_3 : (q', \sigma_3) \xrightarrow{v,m'_3}{}^*(q', \bot), \rho'_2 : (q', \sigma_2) \xrightarrow{v,m'_2}{}^*(p', \bot), \rho'_1 : (p', \sigma_1) \xrightarrow{v,m'_1}{}^*(p', \bot) \in \mathsf{Cal}_v(A)$

avec $\rho_1 \neq \rho_2$ ou $\rho'_1 \neq \rho'_2$.

Intuitivement, les propriétés **S2.a** et **S2.b** sont équivalentes à la propriété **S2** car il est possible de fusionner les calculs ρ_1 et ρ_2 en un seul calcul si $\rho_1 = \rho_2$ et de fusionner les calculs ρ'_1 et ρ'_2 en un seul calcul si $\rho'_1 = \rho'_2$.

Nous allons montrer dans cette section que ces deux propriétés sur les $\mathcal{N}$-APVbi caractérisent l'infinité du coût.

Théorème 4.2.1

> Soit S un $\mathcal{N}$-APVbi réduit sur l'alphabet structuré Δ. Le coût de S est infini si et seulement si S satisfait l'une des propriétés **(S1)** et **(S2)**.

Pour prouver ce résultat, nous allons montrer en premier au Lemme 4.2.2 que si nous avons l'une des propriétés, alors le coût est infini. Dans une seconde partie, nous montrerons aux Lemmes 4.2.4 et 4.2.5 que si le coût est infini, alors le $\mathcal{N}$-APVbi satisfait l'une des propriétés.

Lemme 4.2.2

> Soit S un $\mathcal{N}$-APVbi réduit sur l'alphabet structuré Δ. Si S satisfait **(S1)** ou **(S2)**, alors le coût de S est infini.

Démonstration. Nous notons $S = (A, \lambda)$ avec $A = (Q, I, F, \Gamma, \delta)$. Nous distinguons quatre cas, et montrons que pour chacun de ces cas le coût de S est infini :

- Soient $u \in \Delta^*_{\mathsf{bi}}$ un mot et ρ_1 un calcul choisis comme dans la propriété **(S1.a)**. Comme S est réduit, il existe $\sigma \in \Gamma^*$ tel que (p, σ) est accessible (par un mot $u_i \in \Delta^*$) et co-accessible (par un mot $u_f \in \Delta^*$). Par le Lemme 1.4.5 nous en déduisons que $\hat{\rho}_1 = (p, \sigma) \xrightarrow{u, m_1}{}^*(p, \sigma) \in \mathsf{Cal}_u(A)$. Pour tout $k \in \mathbb{N}$, $\hat{\rho}_1^k$ est un calcul de A sur u^k, de plus le coût $\mathsf{cout}(\hat{\rho}_1^k) = m^k$ croit vers $+\infty$ quand k tend vers $+\infty$. Nous avons donc $u_i u^k u_f \in \mathcal{L}(A)$ et $\mathsf{cout}(u_i u^k u_f) \geqslant m^k$.
- Soient $u \in \Delta^*_{\mathsf{bi}}$ et ρ_1, ρ_2, ρ_3 des calculs choisis comme dans la propriété **(S1.b)**. Comme S est réduit, il existe $\sigma \in \Gamma^*$ tel que (p, σ) est accessible (par un mot $u_i \in \Delta^*$). Par le Lemme 1.4.5 nous en déduisons que $\hat{\rho}_1 = (p, \sigma) \xrightarrow{u, m_1}{}^*(p, \sigma)$, $\hat{\rho}_2 = (p, \sigma) \xrightarrow{u, m_2}{}^*(q, \sigma)$, $\hat{\rho}_3 = (q, \sigma) \xrightarrow{u, m_3}{}^*(q, \sigma) \in \mathsf{Cal}_u(A)$ et donc (q, σ) est co-accessible (par un mot $u_f \in \Delta^*$). Pour tout $k \in \mathbb{N}_{>0}$, et $i, j \in \mathbb{N}$ des entiers tels que $i + j + 1 = k$, $\hat{\rho}_1^i \hat{\rho}_2 \hat{\rho}_3^j$ est un calcul sur le mot u^k, et comme $\rho_1 \neq \rho_2$, tous ces calculs sont différents quand i, j parcourent l'ensemble des entiers tels que $i + j + 1 = k$. De plus quand k tend vers $+\infty$ nous obtenons un nombre arbitrairement grand de calculs sur u^k. Nous avons donc $u_i u^k u_f \in \mathcal{L}(A)$ et $\mathsf{cout}(u_i u^k u_f) \geqslant k$.
- Soient $uv, w \in \Delta^*_{\mathsf{bi}}$ et ρ_1, ρ, ρ'_1 des calculs choisis comme dans la propriété **(S2.a)**. Comme S est réduit, il existe $\sigma \in \Gamma^*$ tel que (p, σ) est accessible (par un mot $u_i \in \Delta^*$). De plus, comme $uv \in \Delta^*_{\mathsf{bi}}$ nous savons que u et v ne contiennent pas de fermant sans correspondance. En appliquant itérativement le Lemme 1.4.5 nous en déduisons donc que pour tout $k \in \mathbb{N}_{>0}$ nous avons

  $$\begin{aligned}
  \hat{\rho}_1^k &= (p, \sigma) \xrightarrow{u^k, m_1^k}{}^*(p, \sigma\sigma_1^k) \in \mathsf{Cal}_{u^k}(A) \\
  \hat{\rho}_k &= (p, \sigma\sigma_1^k) \xrightarrow{w, m}{}^*(p', \sigma\sigma_1^k) \in \mathsf{Cal}_w(A) \\
  \hat{\rho}_1'^k &= (p', \sigma\sigma_1^k) \xrightarrow{v^k, m_1'^k}{}^*(p', \sigma) \in \mathsf{Cal}_{v^k}(A)
  \end{aligned}$$

  Donc (p', σ) est accessible et co-accessible (par un mot $u_f \in \Delta^*$) comme A est émondé. Pour tout $k \in \mathbb{N}$, $\hat{\rho}_1^k \hat{\rho}_k \hat{\rho}_1'^k$ est un calcul de A sur $u^k w v^k$, de plus le coût $\hat{\rho}_1^k \hat{\rho}_k \hat{\rho}_1'^k$ croit vers $+\infty$ quand k tend vers $+\infty$. Nous avons donc $u_i u^k w v^k u_f \in \mathcal{L}(A)$ et $\mathsf{cout}(u_i u^k w v^k u_f) \geqslant m_1^k m_1'^k$.

◦ Soient $uv, w \in \Delta_{\mathsf{bi}}^*$ et $\rho_1, \rho_2, \rho_3, \rho, \rho'_3, \rho'_2, \rho'_1$ des calculs choisis comme dans la propriété **(S2.b)**. Comme S est réduit, il existe $\sigma \in \Gamma^*$ tel que (p, σ) est accessible (par un mot $u_i \in \Delta^*$). De plus, comme $uv \in \Delta_{\mathsf{bi}}^*$, nous savons que u et v ne contiennent pas de fermant sans correspondance. En appliquant itérativement le Lemme 1.4.5 nous en déduisons donc que pour tout $i, j \in \mathbb{N}_{>0}$ nous avons

$$\begin{aligned}
\hat{\rho}_1^{i,j} &= (p, \sigma) \xrightarrow{u^{i+j+1}, m_1^i m_2 m_3^j}{}^*(q, \sigma\sigma_1^i\sigma_2\sigma_3^j) \in \mathsf{Cal}_{u^{i+j+1}}(A) \\
\hat{\rho}_{i,j} &= (q, \sigma\sigma_1^i\sigma_2\sigma_3^j) \xrightarrow{w, m}{}^*(q', \sigma\sigma_1^i\sigma_2\sigma_3^j) \in \mathsf{Cal}_{w}(A) \\
\hat{\rho}_1'^{i,j} &= (q', \sigma\sigma_1^i\sigma_2\sigma_3^j) \xrightarrow{v^{i+j+1}, m_1'^i m_2' m_3'^j}{}^*(p', \sigma) \in \mathsf{Cal}_{v^{i+j+1}}(A)
\end{aligned}$$

Donc (p', σ) est accessible et co-accessible (par un mot $u_f \in \Delta^*$) comme A est émondé. Pour tous entiers $k \in \mathbb{N}_{>0}$ et $i, j \in \mathbb{N}$ tels que $i + j + 1 = k$, $\hat{\rho}_1^{i,j}\hat{\rho}_{i,j}\hat{\rho}_1'^{i,j}$ est un calcul sur le mot $u^k w v^k$, et comme $\rho_1 \neq \rho_2$ ou $\rho'_1 \neq \rho'_2$, tous ces calculs sont différents quand i, j parcourent l'ensemble des entiers tels que $i + j + 1 = k$. De plus quand k tend vers $+\infty$ nous obtenons un nombre arbitrairement grand de calculs sur $u^k w v^k$. Nous avons donc $u_i u^k w v^k u_f \in \mathcal{L}(A)$ et $\mathsf{cout}(u_i u^k u_f) \geqslant k$. □

La preuve de la réciproque (un coût infini implique la présence de l'une des deux propriétés) repose sur deux Lemmes techniques 4.2.4 et 4.2.5 que nous présentons maintenant.

Soit $S = (A, \lambda)$ un $\mathcal{N}$-APVbi réduit avec $A = (Q, I, F, \Gamma, \delta)$. Nous définissons des constantes L_S et N_S et une fonction $\psi_S : \mathbb{N} \to \mathbb{N}$:

$$\begin{aligned}
L_S &= \max(\{m \in \mathbb{N}_{>0} \mid \text{ il existe } \tau \in \delta \text{ telle que } \lambda(\tau) = m\}) \\
N_S &= ((\#Q)^2 L_S)^2 \#\Gamma \\
\psi_S(K) &= \#Q(N_S K)^{2^{\#Q}} \text{ pour tout } K \in \mathbb{N}
\end{aligned}$$

La propriété **(S1)** permet d'augmenter le coût en parcourant un mot bien imbriqué. Le Lemme 4.2.4 établit que si S ne satisfait pas **(S1)**, alors un mot bien imbriqué u avec un coût plus grand que $\psi_S(K)$ possède un sous-mot bien imbriqué v avec un coût plus grand que K, et la hauteur de u est strictement plus grande que celle de v (v est imbriqué dans u, c'est à dire il existe $c \in \Delta_c$ et $r \in \Delta_r$ tels que $u = v_1 c v r v_2$). Le Lemme 4.2.5 applique alors itérativement le Lemme 4.2.4 le long d'un pompage vertical pour prouver qu'un mot avec un grand coût a une grande hauteur, et donc montrer que la propriété **(S2)** est satisfaite.

Soit $w \in \Delta_{\mathsf{bi}}^*$. Étant donnés deux entiers $i \in [1, |w| + 1]$ et $j \in [i, |w| + 1]$ tels que $w(i, j) \in \Delta_{\mathsf{bi}}^*$, nous définissons une matrice, notée par $\mathsf{induit}_{i,j}^w$, représentant comment le coût est modifié par le sous-mot $w(i, j)$. Formellement, $\mathsf{induit}_{i,j}^w$ est un élément de $\mathbb{N}^{Q \times Q}$, et pour tous $p, q \in Q$, nous avons :

$$\mathsf{induit}_{i,j}^w(p, q) = \sum_{m \in M} (m) \text{ avec } M = \{m \in \mathbb{N} \mid (p_i, \bot) \xrightarrow{w(1,i), m_i}{}^*(p, \sigma) \xrightarrow{w(i,j), m}{}^*(q, \sigma) \xrightarrow{w(i\ldots), m_f}{}^*(q_f, \bot) \in \mathsf{ACal}_w(A)\}$$

Finalement, nous définissons $s_{i,j}^w \in \mathbb{N}$ comme étant :

$$s_{i,j}^w = \sum_{p,q \in Q} \mathsf{induit}_{i,j}^w(p, q)$$

D'après les propriétés élémentaires des matrices et des produits matriciels, nous avons :

Lemme 4.2.3

Soient $w \in \Delta_{\mathsf{bi}}^*$ un mot et trois entiers $i, j, k \in \mathbb{N}$ tels que $1 \leqslant i \leqslant j \leqslant k \leqslant |w|$ et $w(i,j), w(j,k) \in \Delta_{\mathsf{bi}}^*$. Nous avons $\mathsf{induit}_{i,k}^w = \mathsf{induit}_{i,j}^w \times \mathsf{induit}_{j,k}^w$, $s_{i,j}^w \leqslant s_{i,k}^w$ et $s_{i,k}^w \leqslant s_{i,j}^w . s_{j,k}^w$.

Lemme 4.2.4

Soit S un $\mathcal{N}$-APVbi réduit sur l'alphabet structuré Δ avec un coût infini mais ne satisfaisant pas **(S1)**. Pour tout $w \in \mathcal{L}(A)$ et $K, i, j \in \mathbb{N}_{>0}$ tels que $i \leqslant j \leqslant |w|$, $w(i,j) \in \Delta_{\mathsf{bi}}^*$ et $s_{i,j}^w \geqslant \psi_S(K)$ il existe $i', j' \in \mathbb{N}_{>0}$ tels que $i < i' \leqslant j' < j$, $w(i', j') \in \Delta_{\mathsf{bi}}^*$, $h_w(i') = h_w(i) + 1$ et $s_{i',j'}^w \geqslant K$.

Démonstration. Nous notons $S = (A, \lambda)$ avec $A = (Q, I, F, \Gamma, \delta)$. Nous définissons l'ensemble P des positions de w à la même hauteur que la position i :

$$P = \{k \in [1, j] \mid w(i,k) \in \Delta_{\mathsf{bi}}^*\}$$

Notez que comme $w(i,i) = \epsilon \in \Delta_{\mathsf{bi}}^*$ nous avons $i \in P$ et $s_{i,i}^w \leqslant \#Q$. Pour chaque $k \in P$, nous définissons l'ensemble X_k de la manière suivante :

$$X_k = \{q \in Q \mid (p_i, \bot) \xrightarrow{w(i,k)}{}^* (q, \sigma) \xrightarrow{w(k,|w|)}{}^* (q_f, \bot) \in \mathsf{ACal}_w(A)\}$$

X_k est l'ensemble des états atteignables après avoir lu le sous-mot $w(1,k)$ (le long d'un calcul acceptant sur w). Pour tout $k, l \in P$ tels que $k < l$ nous avons :

1. $\forall p \in X_k, \exists q \in X_l$ tel que $(p, \bot) \xrightarrow{w(k,l)}{}^* (q, \bot) \in \mathsf{Cal}_{w(k,l)}(A)$
2. $\forall q \in X_l, \exists q \in X_k$ tel que $(p, \bot) \xrightarrow{w(k,l)}{}^* (q, \bot) \in \mathsf{Cal}_{w(k,l)}(A)$
3. $s_{i,k}^w \leqslant s_{i,l}^w$
4. $s_{i,l}^w \leqslant s_{i,k}^w \times s_{k,l}^w$

Les propriétés (3) et (4) sont des conséquences directes du Lemme 4.2.3. Soit C l'ensemble défini par :

$$C = \{k \in P \mid \forall l \in P, l < k \Rightarrow s_{i,l}^w < s_{i,k}^w\}$$

L'ensemble C filtre l'ensemble P en ne gardant que les positions où le coût a augmenté. Pour tout $k \in [1, \#C]$ nous appelons i_k le k-ième élément de C dans l'ordre croissant. Notez que $s_{i,i_l}^w < s_{i,i_k}^w$ pour tout $l \in [1, \#C]$ et $k \in [l+1, \#C]$.

Nous allons maintenant montrer par contradiction qu'il existe $k \in [1, \#C - 1]$ tel que $s_{i_k,i_{k+1}}^w \geqslant N_S K$. Nous supposons donc que cette propriété est fausse, c'est à dire :

$$\text{Pour tout } k \in [1, \#C - 1], s_{i_k,i_{k+1}}^w < N_S K \qquad (\dagger)$$

Cette situation est représentée à la Figure 4.3. La propriété (†) combinée avec (4) implique que $s_{i,i_{k+1}}^w < s_{i,i_k}^w \times N_S K$ pour tout $k \in [1, \#C - 1]$, et donc $s_{i,i_{\#C}}^w < s_{i,i_1}^w \times (N_S K)^{\#C-1}$. Par définition, nous avons $s_{i,i_1}^w = s_{i,i}^w \leqslant \#Q$ et $s_{i,i_{\#C}}^w = s_{i,j}^w \geqslant \psi_S(K) = \#Q(N_S K)^{2^{\#Q}}$. Nous pouvons donc conclure que $\#C > 2^{\#Q}$, ceci assurant qu'il existe deux indices $k \in [1, \#C]$ et $l \in [1, \#C]$ tels que $X_{i_k} = X_{i_l}$. Nous notons $X = X_{i_k}$ et $u = w(i_k, i_l)$.

Supposons que les deux propriétés suivantes soient correctes en même temps :

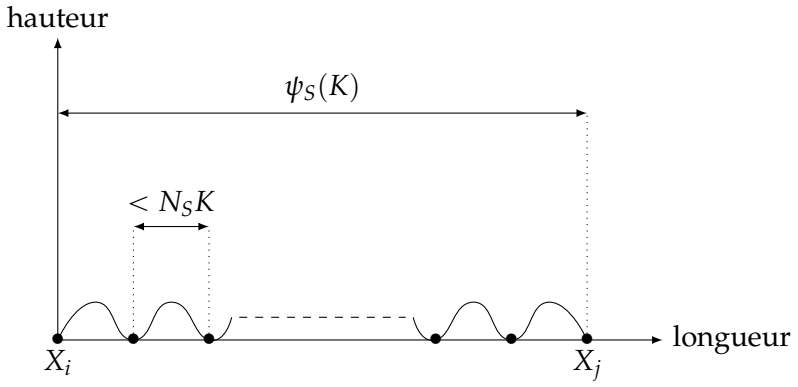


FIGURE 4.3 – $s^w_{i_k,i_{k+1}} < N_S K$

(a) Pour tout calcul

$(p,\perp)\xrightarrow{u,m}{}^*(q,\perp) \in \mathsf{Cal}_u(A)$

tel que $p,q \in X$ nous avons $m = 1$.

(b) Pour tous calculs

$\rho : (p,\perp)\xrightarrow{u,m}{}^*(q,\perp) \in \mathsf{Cal}_u(A) \quad \rho' : (p',\perp)\xrightarrow{u,m'}{}^*(q',\perp) \in \mathsf{Cal}_u(A)$

tels que $p,q,p',q' \in X$, si $p = p'$ ou $q = q'$ alors $\rho = \rho'$.

Par la propriété (b), il n'existe qu'un unique calcul $(p,\perp)\xrightarrow{u,m}{}^*(q,\perp) \in \mathsf{Cal}_u(A)$ pour tout état $p \in X$, et par la propriété (a) nous avons $m = 1$. De cela nous pouvons déduire que $\mathsf{induit}^w_{i_k,i_l}$ est une matrice de permutation, et par le Lemme 4.2.3, nous obtenons que $s^w_{i,i_k} = s^w_{i,i_l}$, ce qui est une contradiction. Les propriétés (a) et (b) ne peuvent donc pas être satisfaites en même temps. Nous distinguons deux cas :

(a) **n'est pas satisfaite et** (b) **est satisfaite.** Il existe donc :

$(p,\perp)\xrightarrow{u,m}{}^*(q,\perp) \in \mathsf{Cal}_u(A)$

avec $p,q \in X$ et $m > 1$. Comme (b) est satisfaite, par les propriétés (1) et de (2) de X il existe un entier $n \in \mathbb{N}$ tel que :

$(q,\perp)\xrightarrow{u^n,m'}{}^*(p,\perp) \in \mathsf{Cal}_{u^n}(A)$ Il existe donc le calcul suivant :

$(p,\perp)\xrightarrow{u^{n+1},mm'}{}^*(p,\perp) \in \mathsf{Cal}_{u^{n+1}}(A)$

avec $m'm > 1$. En conséquence S satisfait la propriété **(S1.a)**, et c'est une contradiction avec l'hypothèse de départ.

(b) **n'est pas satisfaite.** Il existe donc :

$(p,\perp)\xrightarrow{u,m}{}^*(q,\perp) \in \mathsf{Cal}_u(A) \quad (p',\perp)\xrightarrow{u,m'}{}^*(q',\perp) \in \mathsf{Cal}_u(A)$

avec $p,p',q,q' \in X$. Notez que les calculs sont distincts mais q et q' peuvent être égaux. Par les propriétés (1) et de (2) de X pour tout $y \in [1,3]$ il existe des états $p_y \in Q$ et des entiers $n_y, n'_y, m_y, m'_y \in \mathbb{N}$ tels que :

$(p_1,\perp)\xrightarrow{u^{n'_1},m'_1}{}^*(p_1,\perp)\xrightarrow{u^{n_1},m_1}{}^*(p,\perp) \in \mathsf{Cal}_{u^{n'_1+n_1}}(A)$

$(q,\perp)\xrightarrow{u^{n_2},m_2}{}^*(p_2,\perp)\xrightarrow{u^{n'_2},m'_2}{}^*(p_2,\perp) \in \mathsf{Cal}_{u^{n_2+n'_2}}(A)$

$(q',\perp)\xrightarrow{u^{n_3},m_3}{}^*(p_3,\perp)\xrightarrow{u^{n'_3},m'_3}{}^*(p_3,\perp) \in \mathsf{Cal}_{u^{n_3+n'_3}}(A)$

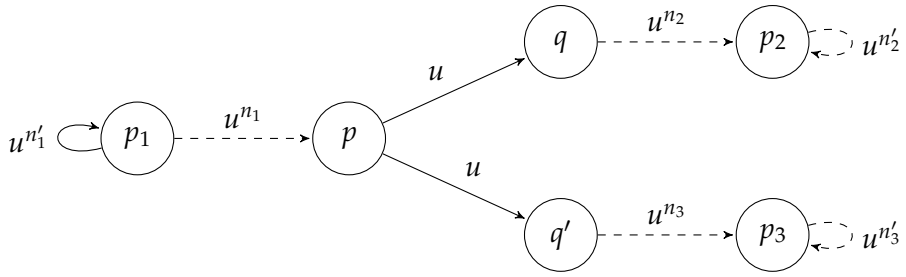


FIGURE 4.4 – La propriété (b) n'est pas satisfaite.

La situation globale est représentée à la Figure 4.4.

Si $p_1 \neq p_2$, alors la propriété **(S1.b)** est satisfaite en choisissant une bonne itération des boucles sur p_1 et p_2. La situation est similaire si $p_1 \neq p_3$. Si $p_1 = p_2 = p_3$, alors la propriété **(S1.b)** est satisfaite dans S car il y a deux boucles différentes autour des états q, et autour des états q'. Dans tous les cas nous avons une contradiction avec les hypothèses de départ.

Nous avons donc montré que si la propriété (†) est satisfaite, alors S satisfait la propriété **(S1)**, ceci contredisant l'hypothèse. Il existe donc $k \in [1, \#C - 1]$ tel que $s^w_{i_k,i_{k+1}} \geqslant N_S K$. Soient i'' et j'' deux entiers tels que $j'' = i_{k+1}$ et $i'' = \max(\{l \in P \mid l < j''\})$. Par définition de C, nous avons $s^w_{i,i_k} = s^w_{i,i''}$. Cela signifie que la somme des coûts des calculs acceptants sur w entre les positions i et i_k d'un coté, et entre i et i'' de l'autre, sont égaux. Ceci entraîne que $s^w_{i_k,i_{k+1}} = s^w_{i'',i_{k+1}}$. De plus, par notre choix de i'' et j'', il n'y a pas de position $x \in P$ telle que $i'' < x < j''$. Par la définition de P, référençant toutes les positions à la même hauteur que i, il y a deux cas possibles, soit $w(i'', j'') \in \Delta_\iota$, soit $w(i'', j'') = cur$, avec $c \in \Delta_c$, $r \in \Delta_r$ et $u \in \Delta^*_{\mathsf{bi}}$. La propriété $s^w_{i'',j''} \geqslant N_S K$ exclut le premier cas. Nous pouvons donc considérer les positions $i' = i'' + 1$ et $j' = j'' - 1$ satisfaisant les conditions $i < i' < j' \leqslant j$, $w(i', j') \in \Delta^*_{\mathsf{bi}}$ et $h_w(i') = h_w(i) + 1$, comme représenté à la Figure 4.5.

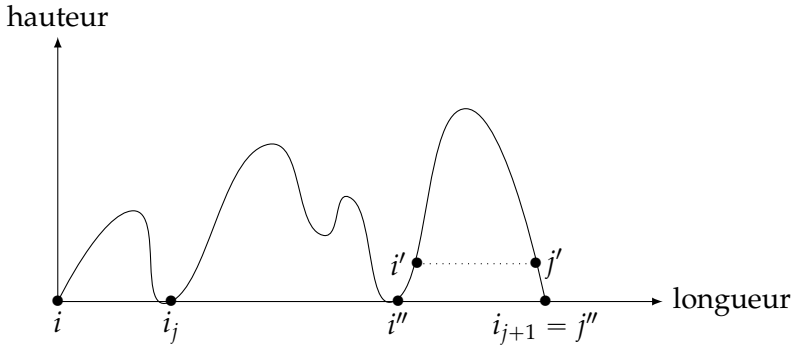


FIGURE 4.5 – $s^w_{i'',j''} \geqslant N_S K$

La lettre lue entre les positions i'' et i' est un ouvrant correspondant avec le fermant lu entre les positions j' et j''. Par conséquent le coût produit entre ces deux paires de positions est borné par $(\#Q^2 L_S)^2 \#\Gamma$. Comme $s^w_{i'',j''} \geqslant N_S K = (\#Q^2 L_S)^2 \#\Gamma K$, nous obtenons finalement que $s^w_{i',j'} \geqslant K$ comme attendu. □

Lemme 4.2.5

Soit S un $\mathcal{N}$-APVbi réduit sur l'alphabet structuré Δ. Si le coût de S est infini, alors il satisfait l'une des deux propriétés **(S1)** et **(S2)**.

Démonstration. Nous notons $S = (A, \lambda)$ avec $A = (Q, I, F, \Gamma, \delta)$. Pour prouver ce résultat, nous supposons que le coût de S est infini mais que s'il ne satisfait pas **(S1)** alors nous montrons qu'il satisfait **(S2)**.

Soit $w \in \mathcal{L}(A)$ un mot tel que $\mathrm{cout}(w) \geqslant \psi_S^H(1)$ où $H = 2^{\#Q^2}$. Nous considérons la séquence (finie) $(x_k, y_k)_k \in (\mathbb{N} \times \mathbb{N})^{\mathbb{N}_{>0}}$ définie par $(x_1, y_1) = (1, |w|)$ et, pour $k \in \mathbb{N}_{>1}$, (x_k, y_k) est défini si et seulement si (x_k, y_k) est la plus petite paire dans l'ordre lexicographique telle que :

- $w(x_k, y_k) \in \Delta_{\mathsf{bi}}^*$,
- $x_k \in [x_{k-1} + 1, y_{k-1} - 1]$,
- $y_k \in [x_k, y_{k-1} - 1]$, $h_w(x_k) = h_w(x_{k-1}) + 1$,
- soit $K_{\max} \in \mathbb{N}$ maximal tel que $s^w_{x_{k-1}, y_{k-1}} \geqslant \psi_S(K_{\max})$ alors $s^w_{x_k, y_k} \geqslant K_{\max}$.

La situation globale est représentée à la Figure 4.6.

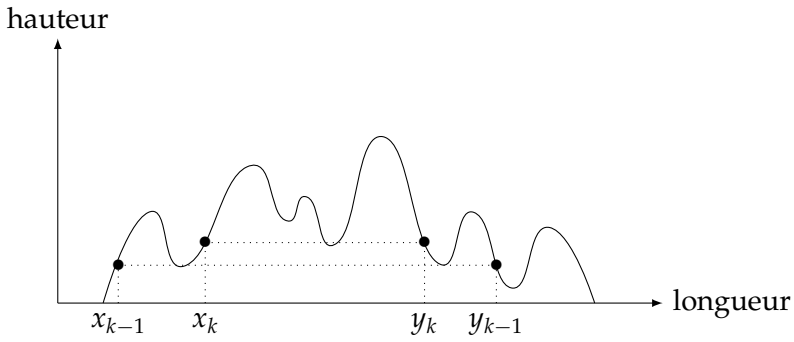


FIGURE 4.6 – $s^w_{x_k, y_k} \geqslant K_{\max}$ et $s^w_{x_{k-1}, y_{k-1}} \geqslant \psi_S(K_{\max})$

Comme h_w est fini cela implique que la séquence $(x_k, y_k)_k$ est finie, et nous la représentons comme $(x_k, y_k)_{k \in [1,L]}$. Notez que $s^w_{x_L, y_L} < \psi_S(1)$. En effet, par le Lemme 4.2.4 s'il existait un entier $K \in \mathbb{N}$ tel que $s^w_{x_L, y_L} \geqslant \psi_S(K)$ alors $s^w_{x_{L+1}, y_{L+1}}$ serait défini. Pour chaque entier $k \in [1, L]$ nous définissons une paire d'ensemble d'états $X_k \subseteq Q \times Q$ comme suit :

$$X_k = \{(p, p') \in Q \times Q \mid (p_i, \bot) \xrightarrow{w(1,x_k)}{}^* (p, \sigma) \xrightarrow{w(x_k,y_k)} (p', \sigma) \xrightarrow{w(y_k,|w|)}{}^* (p_f, \bot) \in \mathsf{ACal}_w(A)\}$$

Pour tout $k, l \in [1, L]$ tels que $k < l$ nous avons :

1. Pour tout $(p, p') \in X_k$ il existe $(q, q') \in X_l$ et $\sigma \in \Gamma^*$ tel que :
$$(p, \bot) \xrightarrow{w(x_k,x_l)}{}^* (q, \sigma) \in \mathsf{Cal}_{w(x_k,x_l)}(A) \quad (q', \sigma) \xrightarrow{w(y_l,y_k)}{}^* (p', \bot) \in \mathsf{Cal}_{w(y_l,y_k)}(A)$$
2. Pour tout $(q, q') \in X_k$ il existe $(p, p') \in X_l$ et $\sigma \in \Gamma^*$ tel que :
$$(p, \bot) \xrightarrow{w(x_k,x_l)}{}^* (q, \sigma) \in \mathsf{Cal}_{w(x_k,x_l)}(A) \quad (q', \sigma) \xrightarrow{w(y_l,y_k)}{}^* (p', \bot) \in \mathsf{Cal}_{w(y_l,y_k)}(A)$$
3. $s^w_{x_l, y_l} \leqslant s^w_{x_k, y_k}$

Soit C l'ensemble défini par :

$$C = \{k \in [1, L] \mid \forall l \in \mathbb{N}, l < k \Rightarrow s^w_{x_l,y_l} > s^w_{x_k,y_k}\}$$

L'ensemble C filtre les éléments de la séquence $(x_k, y_k)_{k\in[1,L]}$ en ne gardant que les positions où le coût a augmenté strictement. Pour tout $k \in [1, \#C]$ nous appelons i_k le k-ième élément de C dans l'ordre croissant. Nous avons :

$$s^w_{x_{i_1},y_{i_1}} = s^w_{x_1,y_1} = \mathsf{cout}(w) \geqslant \psi^H_S(1)$$

En appliquant inductivement le Lemme 4.2.4 nous obtenons que $s_{x_{i_h},y_{i_h}} \geqslant \psi^{H-h}_S(1)$ pour $h \in [1, \min\{H, \#C\}]$. Comme nous avons $s^w_{x_{i_{\#C}},y_{i_{\#C}}} = s^w_{x_L,y_L} < \psi_S(1)$, ceci amène que $\#C > H$. Il existe donc deux entiers différents k et l tels que $X_{i_k} = X_{i_l}$. Le reste de la preuve suit les mêmes lignes que celle du Lemme 4.2.4. Nous appelons $X = X_{i_k}$, $u = w(x_{i_k}, x_{i_l})$, $w' = w(x_{i_l}, y_{i_l})$ et $v = w(y_{i_l}, y_{i_k})$.

Supposons que les deux propriétés suivantes soient vraies en même temps :

(a) Pour tous calculs :
$(p, \bot) \xrightarrow{u,m_u}{}^*(q, \sigma) \in \mathsf{Cal}_u(A) \quad (q', \sigma) \xrightarrow{v,m_v}{}^*(p', \bot) \in \mathsf{Cal}_v(A)$
tels que $(p, p'), (q, q') \in X$ alors $(m, m') = (1, 1)$.

(b) Pour tous calculs :
$\rho_1 : (p_1, \bot) \xrightarrow{u,m_q}{}^*(q_1, \sigma_1) \in \mathsf{Cal}_u(A) \quad \rho_2 : (p_2, \bot) \xrightarrow{u,m_s}{}^*(q_2, \sigma_2) \in \mathsf{Cal}_u(A)$
$\rho'_1 : (q'_1, \sigma_1) \xrightarrow{v,m'_q}{}^*(p'_1, \bot) \in \mathsf{Cal}_v(A) \quad \rho'_2 : (q'_2, \sigma_2) \xrightarrow{v,m'_s}{}^*(p'_2, \bot) \in \mathsf{Cal}_v(A)$
tels que (p_1, p'_1), (q_1, q_1''), (p_2, p'_2), $(q_2, q'_2) \in X$, si $(p_1, p'_1) = (p_2, p'_2)$ ou $(q_1, q'_1) = (q_2, q'_2)$ alors $\rho_1 = \rho_2$ et $\rho'_1 = \rho'_2$.

Dans ce cas, nous pouvons observer que $\mathsf{induit}^u_{x_k,x_l}$ et $\mathsf{induit}^u_{y_l,y_k}$ sont des matrices de permutation et donc par la propriété (3) nous obtenons $s^u_{x_k,y_k} = s^u_{x_l,y_l}$, ce qui est une contradiction. Les propriétés (a) et (b) ne peuvent donc pas être satisfaites en même temps. Nous distinguons deux cas :

(a) **n'est pas satisfaite et** (b) **est satisfaite.** Il existe donc :
$(p, \bot) \xrightarrow{u,m_u}{}^*(q, \sigma) \in \mathsf{Cal}_u(A) \quad (q', \sigma) \xrightarrow{v,m_v}{}^*(p', \bot) \in \mathsf{Cal}_v(A)$
avec $(p, p'), (q, q') \in X$ et $m_u > 1$ ou $m_v > 1$. Comme (b) est satisfaite, par les propriétés (1) et (2) de X il existe un entier $n \in \mathbb{N}$ tel que :
$(q, \bot) \xrightarrow{u^n,m'_u}{}^*(p, \sigma') \in \mathsf{Cal}_{u^n}(A) \quad (p', \sigma') \xrightarrow{v^n,m'_v}{}^*(q', \bot) \in \mathsf{Cal}_{v^n}(A)$
Par le Lemme 1.4.5 il existe donc le calcul suivant :
$(p, \bot) \xrightarrow{u^{n+1},m_u m'_u}{}^*(p, \sigma\sigma') \xrightarrow{w'}{}^*(p', \sigma\sigma') \xrightarrow{v^{n+1},m_v m'_v}{}^*(p', \bot) \in \mathsf{Cal}_{u^{n+1}w'v^{n+1}}(A)$
avec $m'_u m_u > 1$ ou $m'_v m_v > 1$. En conséquence S satisfait la propriété **(S2.a)**.

(b) **n'est pas satisfaite.** Il existe donc :
$(p, \bot) \xrightarrow{u,m_q} (q, \sigma_q) \quad \in \mathsf{Cal}_u(A) \quad (p, \bot) \xrightarrow{u,m_s} (s, \sigma_s) \quad \in \mathsf{Cal}_u(A)$
$(q', \sigma_q) \xrightarrow{v,m'_q}{}^*(p', \bot) \in \mathsf{Cal}_v(A) \quad (s', \sigma_s) \xrightarrow{v,m'_s}{}^*(p', \bot) \in \mathsf{Cal}_v(A)$
avec (p, p'), $(q, q'), (s, s') \in X$. Notez que les calculs sont distincts mais (q, q') et (s, s') peuvent être égaux. Par les propriétés (1) et (2) de X pour tout $y \in [1, 3]$ il existe des états $p_y, p'_y, q_y, q'_y, s_y, s'_y \in Q$ et des entiers $n_y, n'_y, m_y, m'_y, o_y, o'_y \in \mathbb{N}$ tels que :

$(p_1, \bot) \xrightarrow{u^{n'_1}, o_1}{}^* (p_1, \sigma'_1) \xrightarrow{u^{n_1}, m_1}{}^* (p, \sigma_1) \in \mathsf{Cal}_{u^{n'_1+n_1}}(A)$

$(q, \bot) \xrightarrow{u^{n_2}, m_2}{}^* (p_2, \sigma_2) \xrightarrow{u^{n'_2}, o_2}{}^* (p_2, \sigma'_2) \in \mathsf{Cal}_{u^{n_2+n'_2}}(A)$

$(s, \bot) \xrightarrow{u^{n_3}, m_3}{}^* (p_3, \sigma_3) \xrightarrow{u^{n'_3}, o_3}{}^* (p_3, \sigma'_3) \in \mathsf{Cal}_{u^{n_3+n'_3}}(A)$

$(p', \sigma_1) \xrightarrow{v^{n_1}, m'_1}{}^* (p'_1, \sigma'_1) \xrightarrow{v^{n'_1}, o'_1}{}^* (p'_1, \bot) \in \mathsf{Cal}_{v^{n_1+n'_1}}(A)$

$(p'_2, \sigma'_2) \xrightarrow{v^{n'_2}, o'_2}{}^* (p'_2, \sigma_2) \xrightarrow{v^{n_2}, m'_2}{}^* (q', \bot) \in \mathsf{Cal}_{v^{n'_2+n_2}}(A)$

$(p'_3, \sigma'_3) \xrightarrow{v^{n'_3}, o'_3}{}^* (p'_3, \sigma_3) \xrightarrow{v^{n_3}, m'_3}{}^* (s', \bot) \in \mathsf{Cal}_{v^{n'_3+n_3}}(A)$

La situation globale est représentée à la Figure 4.7.

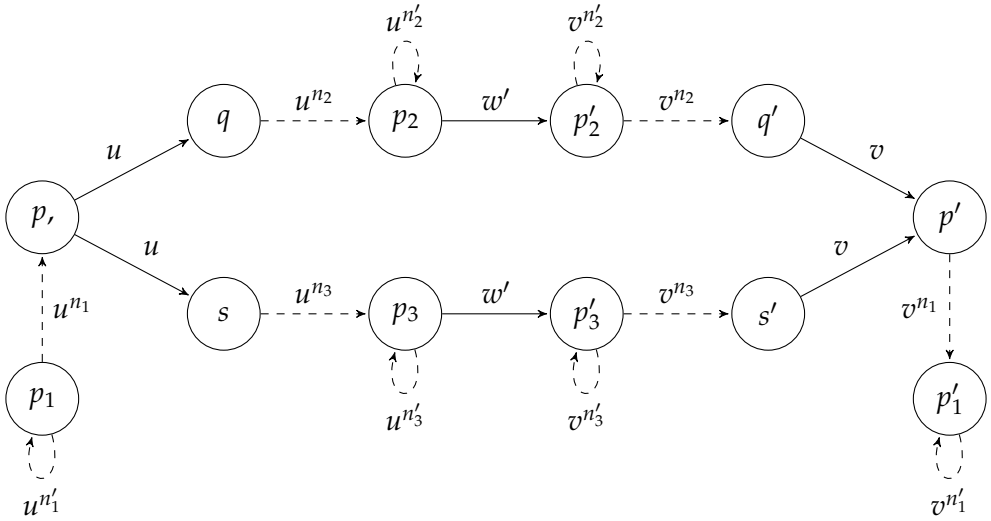


FIGURE 4.7 – S ne satisfait pas la propriété (b).

Si $(p_1, p'_1) \neq (p_2, p'_2)$, alors la propriété **(S2.b)** est satisfaite en choisissant une bonne itération des boucles sur p_1, p_2, p'_2 et p'_1. La situation est similaire si $(p_1, p'_1) \neq (p_3, p'_3)$. Si $(p_1, p'_1) = (p_2, p'_2) = (p_3, p'_3)$, alors la propriété **(S2.b)** est satisfaite dans S car il y a des boucles différentes autour des états q, s, q' et s'. □

4.2.2 La décision du coût infini est dans PTIME

Nous montrons dans cette partie comment déterminer en PTIME la présence de l'une de ces propriétés. Dans ce but nous présentons maintenant la procédure estinfini, représenté à la Figure 4.8. Cet algorithme utilise trois types de règles d'inférence utilisées comme une procédure de saturation : les quatre premières règles construisent un ensemble $\mathcal{S}_1$ de tuples composés de six états et un booléen, permettant de décider la présence de **(S1)**. Les six états représentent la source et la cible de trois calculs sur le même mot bien imbriqué et le booléen mémorise une multiplicité supérieure à 1 sur le premier calcul ou le fait que le premier calcul est différent du second. Les quatre suivantes construisent l'ensemble $\mathcal{S}_2$ de tuples composés de douze états et d'un booléen permettant de détecter la propriété **(S2)**. Cette construction est basée sur $\mathcal{S}_1$: les états permettent d'identifier deux ensembles de trois calculs sur les mots u et v, tels que les calculs du second ensemble dépilent les symboles empilés par les calculs du premier ensemble, assurant ainsi que $uv \in \Delta^*_{\mathsf{bi}}$. L'information stockée dans le booléen dépend de l'un de ces deux ensembles de calcul, de manière similaire à $\mathcal{S}_1$. Enfin, les dernières règles construisent

l'ensemble $\mathcal{S}_3$ permettant d'assurer qu'un tuple de $\mathcal{S}_1$ satisfait la propriété **(S1)** (règle 3.1), ou qu'un tuple de $\mathcal{S}_2$ satisfait la propriété **(S2)** (règle 3.2).

$$\frac{p_i \in Q \text{ pour tout } i \in [1,3]}{(p_1,p_1,p_2,p_2,p_3,p_3,p_1 \neq p_2) \in \mathcal{S}_1} \quad (1.1)$$

$$\frac{\tau_i = (p_i,a,q_i) \in \delta_\iota \text{ pour tout } i \in [1,3]}{(p_1,q_1,p_2,q_2,p_3,q_3,(\tau_1 \neq \tau_2 \vee \lambda(\tau_1) > 1)) \in \mathcal{S}_1} \quad (1.2)$$

$$\frac{(p_1,q_1',p_2,q_2',p_3,q_3',B') \in \mathcal{S}_1, (q_1',q_1,q_2',q_2,q_3',q_3,B'') \in \mathcal{S}_1}{(p_1,q_1,p_2,q_2,p_3,q_3,B' \vee B'') \in \mathcal{S}_1} \quad (1.3)$$

$$\frac{\begin{array}{c}(p_1',q_1',p_2',q_2',p_3',q_3',B') \in \mathcal{S}_1,\\ \tau_i = (p_i,c,\gamma_i,p_i') \in \delta_c, \tau_i' = (q_i',\gamma_i,r,q_i) \in \delta_r \text{ pour tout } i \in [1,3]\end{array}}{(p_1,q_1,p_2,q_2,p_3,q_3,B' \vee (\tau_1 \neq \tau_2 \vee \tau_1' \neq \tau_2' \vee \lambda(\tau_1) > 1 \vee \lambda(\tau_1') > 1)) \in \mathcal{S}_1} \quad (1.4)$$

$$\frac{(p_1,q_1,p_2,q_2,p_3,q_3,B') \in \mathcal{S}_1, (q_3',p_3',q_2',p_2',q_1',p_1',B'') \in \mathcal{S}_1}{(p_1,q_1,p_2,q_2,p_3,q_3,q_3',p_3',q_2',p_2',q_1',p_1',B' \vee B'') \in \mathcal{S}_2} \quad (2.1)$$

$$\frac{\tau_i = (p_i,c,\gamma_i,q_i) \in \delta_c, \quad \tau_i' = (q_i',\gamma_i,r,p_i') \in \delta_r, \text{ pour tout } i \in [1,3]}{(p_1,q_1,p_2,q_2,p_3,q_3,q_3',p_3',q_2',p_2',q_1',p_1',\tau_1 \neq \tau_2 \vee \tau_1' \neq \tau_2' \vee \lambda(\tau_1) > 1 \vee \lambda(\tau_1') > 1) \in \mathcal{S}_2} \quad (2.2)$$

$$\frac{\begin{array}{c}(p_1,p_1'',p_2,p_2'',p_3,p_3'',q_3'',p_3',q_2'',p_2',q_1'',p_1',B') \in \mathcal{S}_2,\\ (p_1'',q_1,p_2'',q_2,p_3'',q_3,q_3'',q_3'',q_2'',q_2'',q_1',q_1'',B'') \in \mathcal{S}_2\end{array}}{(p_1,q_1,p_2,q_2,p_3,q_3,q_3',p_3',q_2',p_2',q_1',p_1',B' \vee B'') \in \mathcal{S}_2} \quad (2.3)$$

$$\frac{(p,p,p,q,q,q,\top) \in \mathcal{S}_1}{\top \in \mathcal{S}_3} \quad (3.1) \qquad \frac{\begin{array}{c}(q,q') \in \mathsf{BI}_A,\\ (p,p,p,q,q,q,q',q',q',p',p',p',\top) \in \mathcal{S}_2\end{array}}{\top \in \mathcal{S}_3} \quad (3.2)$$

FIGURE 4.8 – Règles d'inférence utilisées par estinfini pour décider le coût infini d'un $\mathcal{N}$-APVbi réduit.

Proposition 4.2.6

Soit S un $\mathcal{N}$-APVbi réduit sur l'alphabet structuré Δ. L'algorithme estinfini produit l'élément $\top \in \mathcal{S}_3$ si et seulement si le coût de S est infini.

Démonstration. Nous notons $S = (A,\lambda)$ avec $A = (Q,I,F,\Gamma,\delta)$. Nous procédons successivement avec $\mathcal{S}_1$, $\mathcal{S}_2$ et $\mathcal{S}_3$.

Nous prouvons que tout tuple $(p_1,q_1,p_2,q_2,p_2,q_3,B) \in \mathcal{S}_1$ si et seulement si la propriété suivante est correcte :

$$\begin{array}{r}\exists u \in \Delta_{\mathsf{bi}}^* \text{ et des calculs } \rho_i : (p_i,\bot) \xrightarrow{u}{}^* (q_i,\bot) \text{ de } A \text{ pour tout } i \in [1,3]\\ \text{tels que } B = (\rho_1 \neq \rho_2 \vee \langle \rho_1 \rangle > 1)\end{array} \quad (4.1)$$

Nous prouvons en premier le sens $\Rightarrow$. Soit $(p_1,q_1,p_2,q_2,p_3,q_3,B)$ un tuple de $\mathcal{S}_1$. Ce tuple a pu être ajouté à $\mathcal{S}_1$ de quatre manières différentes :

règle 1.1 Nous avons $p_i = q_i$ pour tout $i \in [1,3]$. De plus $\rho_i : (p_i,\bot) \in \mathsf{Cal}_\epsilon(A)$, $\rho_1 \neq \rho_2$ si et seulement si $p_1 \neq p_2$ et $\mathrm{cout}(\rho_1) = 1$ par définition. Donc $(p_1,p_1,p_2,p_2,p_3,p_3,B)$ satisfait la Propriété 4.1 avec $B = (p_1 \neq p_2)$.

règle 1.2 Il existe $\tau_i = (p_i, a, q_i) \in \delta_\iota$ pour tout $i \in [1,3]$. Nous avons $\rho_i : (p_i, \bot) \xrightarrow{a} (q_i, \bot) \in \mathsf{Cal}_a(A)$ avec $\rho_1 \neq \rho_2$ si et seulement si $\tau_1 \neq \tau_2$ et $\mathsf{cout}(\rho_1) > 1$ si et seulement si $\lambda(\tau_i) > 1$. Donc $(p_1, q_1, p_2, q_2, p_3, q_3, B)$ satisfait la Propriété 4.1 avec $B = \lambda(\tau_1) > 1 \vee (\tau_1 \neq \tau_2)$.

règle 1.3 Il existe $(p_1, q'_1, p_2, q'_2, p_3, q'_3, B'), (q'_1, q_1, q'_2, q_2, q'_3, q_3, B'') \in \mathcal{S}_1$. Par hypothèse d'induction il existe $u, v \in \Delta^*_{\mathsf{bi}}$ tels que $\rho'_i : (p_i, \bot) \xrightarrow{u}{}^*(q'_i, \bot) \in \mathsf{Cal}_u(A)$ et $\rho''_i : (q'_i, \bot) \xrightarrow{v}{}^*(q_i, \bot) \in \mathsf{Cal}_v(A)$ pour tout $i \in [1,3]$ avec $B' = (\rho'_1 \neq \rho'_2 \vee \langle \rho'_1 \rangle > 1)$ et $B'' = (\rho''_1 \neq \rho''_2 \vee \langle \rho''_1 \rangle > 1)$. Nous avons donc $\rho_i : \rho'_i \rho''_i \in \mathsf{Cal}_{uv}(A)$, avec $\rho_1 \neq \rho_2$ si et seulement si $\rho'_1 \neq \rho'_2$ ou $\rho''_1 \neq \rho''_2$ et $\mathsf{cout}(\rho_1) > 1$ si et seulement si $\mathsf{cout}(\rho'_1) > 1$ ou $\mathsf{cout}(\rho''_1) > 1$. Donc $(p_1, q_1, p_2, q_2, p_3, q_3, B)$ satisfait la Propriété 4.1 avec $B = B' \vee B''$.

règle 1.4 Il existe $(p'_1, q'_1, p'_2, q'_2, p'_3, q'_3, B') \in \mathcal{S}_1$, $\tau_i = (p_i, c, \gamma, p'_i) \in \delta_c$ et $\tau'_i = (q'_i, \gamma, r, q_i) \in \delta_r$ pour tout $i \in [1,3]$. Par hypothèse d'induction il existe $u \in \Delta^*_{\mathsf{bi}}$ tel que $\rho'_i : (p'_i, \bot) \xrightarrow{u}{}^*(q_i, \bot) \in \mathsf{Cal}_u(A)$ pour tout $i \in [1,3]$ avec $B' = (\rho'_1 \neq \rho'_2 \vee \langle \rho'_1 \rangle > 1)$. Par le Lemme 1.4.5 nous avons donc $\rho_i : (p_i, \bot) \xrightarrow{c} (p'_i, \gamma) \xrightarrow{u}{}^* (q'_i, \gamma) \xrightarrow{r} (q_i, \bot) \in \mathsf{Cal}_{cur}(A)$ avec $\rho_1 \neq \rho_2$ si et seulement si $\rho'_1 \neq \rho'_2$ ou $\tau_1 \neq \tau_2$ ou $\tau'_1 \neq \tau'_2$ et $\mathsf{cout}(\rho_1) > 1$ si et seulement si $\mathsf{cout}(\rho'_1) > 1$ ou $\lambda(\tau_1) > 1$ ou $\lambda(\tau'_1) > 1$. Donc $(p_1, q_1, p_2, q_2, p_3, q_3, B)$ satisfait la Propriété 4.1 avec $B = B' \vee (\lambda(\tau_1) > 1 \vee \lambda(\tau'_1) > 1 \vee \tau_1 \neq \tau_2 \vee \tau'_1 \neq \tau'_2)$.

Nous montrons maintenant le sens $\Leftarrow$ par induction sur la structure de u.

- Si $u = \epsilon$ alors pour tout $i \in [1,3]$ nous avons $p_i = q_i$, $\rho_1 \neq \rho_2$ si et seulement si $p_1 \neq p_2$ et $\mathsf{cout}(\rho_1) = 1$ par définition. La Propriété 4.1 est donc vérifiée car par la règle 1.1 nous avons $(p_1, p_1, p_2, p_2, p_3, p_3, B) \in \mathcal{S}_1$ avec $B = (p_1 \neq p_2)$.
- Si $u = a$ avec $\in \Delta_\iota$ alors pour tout $i \in [1,3]$ il existe $\tau_i = (p_i, a, q_i) \in \delta_\iota$ avec $\rho_1 \neq \rho_2$ si et seulement si $\tau_1 \neq \tau_2$ et $\mathsf{cout}(\rho_1) > 1$ si et seulement si $\lambda(\tau_i) > 1$. La Propriété 4.1 est donc vérifiée car par la règle 1.2 nous avons $(p_1, p_1, p_2, p_2, p_3, p_3, B) \in \mathcal{S}_1$ avec $B = \lambda(\tau_1) > 1 \vee (\tau_1 \neq \tau_2)$.
- Si $u = vw$ avec $v, w \in \Delta^*_{\mathsf{bi}}$ alors pour tout $i \in [1,3]$ il existe $\rho'_i : (p_i, \bot) \xrightarrow{v}{}^*(p'_i, \bot) \in \mathsf{Cal}_v(A)$ et $\rho''_i : (q'_i, \bot) \xrightarrow{w}{}^*(q_i, \bot) \in \mathsf{Cal}_w(A)$ avec $\rho_1 \neq \rho_2$ si et seulement si $\rho'_1 \neq \rho'_2$ ou $\rho''_1 \neq \rho''_2$ et $\mathsf{cout}(\rho_1) > 1$ si et seulement si $\mathsf{cout}(\rho'_1) > 1$ ou $\mathsf{cout}(\rho''_1) > 1$. Par hypothèse d'induction nous avons $(p_1, p'_1, p_2, p'_2, p_3, p'_3, B'), (q'_1, q_1, q'_2, q_2, q'_3, q_3, B'') \in \mathcal{S}_1$ avec $B' = (\rho'_1 \neq \rho'_2 \vee \langle \rho'_1 \rangle > 1)$ et $B'' = (\rho''_1 \neq \rho''_2 \vee \langle \rho''_1 \rangle > 1)$. La Propriété 4.1 est donc vérifiée car par la règle 1.3 nous avons $(p_1, p_1, p_2, p_2, p_3, p_3, B) \in \mathcal{S}_1$ avec $B = B' \vee B''$.
- Si $u = cvr$ avec $c \in \Delta_c$, $r \in \Delta_r$ et $v \in \Delta^*_{\mathsf{bi}}$ alors pour tout $i \in [1,3]$ il existe $\tau_i = (p_i, c, \gamma, p'_i) \in \delta_c$, $\tau'_i = (q'_i, \gamma, r, q_i) \in \delta_r$ et $\rho'_i : (p'_i, \gamma) \xrightarrow{v}{}^*(q'_i, \gamma) \in \mathsf{Cal}_v(A)$ avec $\rho_1 \neq \rho_2$ si et seulement si $\rho'_1 \neq \rho'_2$ ou $\tau_1 \neq \tau_2$ ou $\tau'_1 \neq \tau'_2$ et $\mathsf{cout}(\rho_1) > 1$ si et seulement si $\mathsf{cout}(\rho'_1) > 1$ ou $\lambda(\tau_1) > 1$ ou $\lambda(\tau'_1) > 1$. Par le Lemme 1.4.7 et par hypothèse d'induction nous avons $(p'_1, q'_1, p'_2, q'_2, p'_3, q'_3, B') \in \mathcal{S}_1$ avec $B' = (\rho'_1 \neq \rho'_2 \vee \langle \rho'_1 \rangle > 1)$. La Propriété 4.1 est donc vérifiée car par la règle 1.4 nous avons $(p_1, p_1, p_2, p_2, p_3, p_3, B) \in \mathcal{S}_1$ avec $B = B' \vee (\lambda(\tau_1) > 1 \vee \lambda(\tau'_1) > 1 \vee \tau_1 \neq \tau_2 \vee \tau'_1 \neq \tau'_2)$.

Nous prouvons que tout tuple $(p_1, q_1, p_2, q_2, p_3, q_3, q'_3, p'_3, q'_2, p'_2, q'_1, p'_1, B) \in \mathcal{S}_2$ si et seulement si la propriété suivante est correcte :

$$\begin{aligned} &\exists uv \in \Delta^*_{\mathsf{bi}} \text{ et des calculs } \rho_i : (p_i, \bot) \xrightarrow{u}{}^*(q_i, \sigma_i) \\ &\text{et } \rho'_i : (q'_i, \sigma_i) \xrightarrow{v}{}^*(p'_i, \bot) \text{ de } A \text{ pour } i \in [1,3] \\ &\text{et } B = (\rho_1 \neq \rho_2 \vee \rho'_1 \neq \rho'_2 \neq \vee \langle \rho_1 \rangle > 1 \vee \langle \rho'_1 \rangle > 1) \end{aligned} \tag{4.2}$$

Nous prouvons en premier le sens $\Rightarrow$. Soit $c = (p_1, q_1, p_2, q_2, p_3, q_3, q'_3, p'_3, q'_2, p'_2, q'_1, p'_1, B)$ un tuple de $\mathcal{S}_2$. Ce tuple a pu être ajouté à $\mathcal{S}_2$ de quatre manières différentes :

règle 2.1 Il existe $(p_1, q_1, p_2, q_2, p_3, q_3, B')$ et $(q'_3, p'_3, q'_2, p'_2, q'_1, p'_1, B'') \in \mathcal{S}_1$. Par la Propriété (4.1) il existe $u, v \in \Delta^*_{\mathsf{bi}}$ et des calculs $\rho'_i : (p_i, \bot) \xrightarrow{u}{}^*(q_i, \bot) \in \mathsf{Cal}_u(A)$ et $\rho''_i : (q'_i, \bot) \xrightarrow{v}{}^*(p'_i, \bot) \in \mathsf{Cal}_v(A)$ pour tout $i \in [1,3]$ avec $B' = (\rho'_1 \neq \rho'_2 \vee \langle \rho'_1 \rangle > 1)$ et $B' = (\rho''_1 \neq \rho''_2 \vee \langle \rho''_1 \rangle > 1)$. Donc $(p_1, q_1, p_2, q_2, p_3, q_3, q'_3, p'_3, q'_2, p'_2, q'_1, p'_1, B)$ satisfait la Propriété 4.2 avec $B = B' \vee B''$.

règle 2.2 Il existe $\tau_i = (p_i, c, \gamma, q_i) \in \delta_c$ et $\tau'_i = (q'_i, \gamma, r, p'_i) \in \delta_r$ pour $i \in [1,3]$. Nous avons donc $\rho_i : (p_i, \bot) \xrightarrow{c} (q_i, \gamma) \in \mathsf{Cal}_c(A)$ et $\rho'_i : (q'_i, \gamma) \xrightarrow{r} (p'_i, \bot) \in \mathsf{Cal}_r(A)$ avec $\rho_1 \neq \rho_2$ ou $\rho'_1 \neq \rho'_2$ si et seulement si $\tau_1 \neq \tau_2$ ou $\tau'_1 \neq \tau'_2$, et $\mathsf{cout}(\rho_1) > 1$ ou $\mathsf{cout}(\rho'_1) > 1$ si et seulement si $\lambda(\tau_1) > 1$ ou $\lambda(\tau'_1) > 1$. Donc $(p_1, q_1, p_2, q_2, p_3, q_3, q'_3, p'_3, q'_2, p'_2, q'_1, p'_1, B)$ satisfait la Propriété 4.2 avec $B = \tau_1 \neq \tau_2 \vee \tau'_1 \neq \tau'_2 \vee \lambda(\tau_1) > 1 \vee \lambda(\tau'_1) > 1$.

règle 2.3 Il existe $(p_1, p''_1, p_2, p''_2, p_3, p''_3, q''_3, p'_3, q''_2, p'_2, q''_1, p'_1, B')$ et $(p''_1, q_1, p''_2, q_2, p''_3, q_3, q'_3, q''_3, q'_2, q''_2, q'_1, q''_1, B'') \in \mathcal{S}_2$ tels que $B'' = B \vee B'$. Par hypothèse d'induction il existe quatre mots $u\tilde{u}, v\tilde{v} \in \Delta^*_{\mathsf{bi}}$ et des calculs $\rho'_i : (p_i, \bot) \xrightarrow{u}{}^*(p''_i, \sigma_i) \in \mathsf{Cal}_u(A)$, $\rho''_i : (p''_i, \bot) \xrightarrow{v}{}^*(q_i, \sigma'_i) \in \mathsf{Cal}_v(A)$, $\tilde{\rho}'_i : (q''_i, \sigma_i) \xrightarrow{\tilde{u}}{}^*(p'_i, \bot) \in \mathsf{Cal}_{\tilde{u}}(A)$ et $\tilde{\rho}''_i = (q'_i, \sigma'_i) \xrightarrow{\tilde{v}}{}^*(q''_i, \bot) \in \mathsf{Cal}_{\tilde{v}}(A)$ pour tout $i \in [1,3]$, avec $B' = (\rho'_1 \neq \rho'_2 \vee \tilde{\rho}'_1 \neq \tilde{\rho}'_2 \vee \langle \rho'_1 \rangle > 1 \vee \langle \tilde{\rho}'_1 \rangle > 1)$ et $B'' = (\rho''_1 \neq \rho''_2 \vee \tilde{\rho}''_1 \neq \tilde{\rho}''_2 \vee \langle \rho''_1 \rangle > 1 \vee \langle \tilde{\rho}''_1 \rangle > 1)$. Nous avons donc $\rho_i : (p_i, \bot) \xrightarrow{u}{}^*(q_i, \sigma_i\sigma'_i) \in \mathsf{Cal}_{uv}(A)$ et $\tilde{\rho}_i : (q'_i, \sigma_i\sigma'_i) \xrightarrow{\tilde{v}\tilde{u}}{}^*(p'_i, \bot) \in \mathsf{Cal}_{\tilde{v}\tilde{u}}(A)$ avec $\rho_1 \neq \rho_2$ ou $\tilde{\rho}_1 \neq \tilde{\rho}_2$ si et seulement si $\rho'_1 \neq \rho'_2$ ou $\rho''_1 \neq \rho''_2$ ou $\tilde{\rho}'_1 \neq \tilde{\rho}'_2$ ou $\tilde{\rho}''_1 \neq \tilde{\rho}''_2$ et $\mathsf{cout}(\rho_1) > 1$ ou $\mathsf{cout}(\tilde{\rho}_1) > 1$ si et seulement si $\mathsf{cout}(\rho'_1) > 1$ ou $\mathsf{cout}(\rho''_1) > 1$ ou $\mathsf{cout}(\tilde{\rho}'_1) > 1$ ou $\mathsf{cout}(\tilde{\rho}''_1) > 1$. Donc $(p_1, q_1, p_2, q_2, p_3, q_3, q'_3, p'_3, q'_2, p'_2, q'_1, p'_1, B)$ satisfait la Propriété 4.2 avec $B = B' \vee B''$.

Nous montrons maintenant le sens $\Leftarrow$ par induction sur la hauteur de σ_1.

- Si $|\sigma_1| = 0$ par la Propriété 4.1 $(p_1, q_1, p_2, q_2, p_3, q_3, B'), (q'_3, p'_3, q'_2, p'_2, q'_1, p'_1, B'') \in \mathcal{S}_1$ avec $B' = (\rho'_1 \neq \rho'_2 \vee \langle \rho'_1 \rangle > 1)$ et $B'' = (\rho''_1 \neq \rho''_2 \vee \langle \rho''_1 \rangle > 1)$. La Propriété 4.2 est donc vérifiée car par la règle 2.1 $(p_1, q_1, p_2, q_2, p_3, q_3, q'_3, p'_3, q'_2, p'_2, q'_1, p'_1, B) \in \mathcal{S}_2$ avec $B = B' \vee B''$.
- Soit $n \in \mathbb{N}$ un entier, nous supposons que la Propriété 4.2 est vraie pour tout $|\sigma_1| < n$ et montrons qu'elle est aussi correcte pour $|\sigma_1| = n$. Comme $|\sigma_1| > 0$, nous avons $u = u_1 c u_2$ et $v = v_2 r v_1$ avec $u_1, v_1 \in \Delta^*_{\mathsf{bi}}$, $u_2, v_2 \in \Delta^*$, $c \in \Delta_c$ et $r \in \Delta_r$. Pour tout $i \in [1,3]$ nous avons $\rho'_i : (p_i, \bot) \xrightarrow{u_1}{}^*(r_i, \bot) \in \mathsf{Cal}_{u_1}(A)$, $\rho''_i : (s_i, \gamma) \xrightarrow{u_2}{}^*(q_i, \sigma_i) \in \mathsf{Cal}_{u_2}(A)$, $\tilde{\rho}''_i = (q'_i, \sigma_i) \xrightarrow{v_2}{}^*(s'_i, \gamma) \in \mathsf{Cal}_{v_2}(A)$, $\tilde{\rho}'_i : (r'_i, \bot) \xrightarrow{v_1}{}^*(p'_i, \bot) \in \mathsf{Cal}_{v_1}(A)$, $\tau_i = (r_i, c, \gamma, s_i) \in \delta_c$ et $\tau'_i \in (s'_i, \gamma, r, r'_i) \in \delta_r$. Donc $(r_1, s_1, r_2, s_2, r_3, s_3, r'_3, s'_3, r'_2, s'_2, r'_1, s'_1, B''') \in \mathcal{S}_2$ par la règle 2.2 avec $B' = (\tau_1 \neq \tau_2 \vee \tau'_1 \neq \tau'_2 \vee \lambda(\tau_1) > 1 \vee \lambda(\tau'_1) > 1)$. Par hypothèse d'induction nous avons $(p_1, r_1, p_2, r_2, p_3, r_3, r'_3, p'_3, r'_2, p'_2, r'_1, p'_1, B) \in \mathcal{S}_2$ avec $B' = (\rho'_1 \neq \rho'_2 \vee \tilde{\rho}'_1 \neq \tilde{\rho}'_2 \vee \langle \rho'_1 \rangle > 1 \vee \langle \tilde{\rho}'_1 \rangle > 1)$ et $(s_1, q_1, s_2, q_2, s_3, q_3, q'_3, s'_3, q'_2, s'_2, q'_1, s'_1, B'') \in \mathcal{S}_2$ avec $B'' = (\rho''_1 \neq \rho''_2 \vee \tilde{\rho}''_1 \neq \tilde{\rho}''_2 \vee \langle \rho''_1 \rangle > 1 \vee \langle \tilde{\rho}''_1 \rangle > 1)$. La Propriété 4.2 est vérifiée car par la règle 2.3 appliquée deux fois $(p_1, q_1, p_2, q_2, p_3, q_3, q'_3, p'_3, q'_2, p'_2, q'_1, p'_1, B) \in \mathcal{S}_2$ avec $B = B' \vee B'' \vee B'''$.

Nous prouvons maintenant que $\top \in \mathcal{S}_3$ si et seulement si le coût de S est infini. Nous montrons en premier le sens $\Rightarrow$. Il y a deux possibilités pour la présence de $\top$ dans $\mathcal{S}_3$:

règle 3.1 Il existe $(p, p, p, q, q, q, \top) \in \mathcal{S}_1$. Par la Propriété 4.1 sur $\mathcal{S}_1$, il existe $u \in \Delta^*_{\mathsf{bi}}$ et des calculs $\rho_1 : (p, \bot) \xrightarrow{u}{}^*(p, \bot)$, $\rho_2 : (p, \bot) \xrightarrow{u}{}^*(q, \bot)$ et $\rho_3 : (q, \bot) \xrightarrow{u}{}^*(q, \bot)$ de A sur u avec $\langle \rho_1 \rangle > 1$ ou $\rho_1 \neq \rho_2$. S satisfait donc la Propriété **S1**, et le coût de S est infini par le Lemme 4.2.2.

règle 3.2 Il existe $(q, q') \in \mathsf{BI}_A$ et $(p, p, p, q, q, q, q', q', q', p', p', p', \top) \in \mathcal{S}_2$. Par la définition de BI_S et la Propriété 4.2 sur $\mathcal{S}_2$, il existe trois mots $uu', w \in \Delta^*_{\mathsf{bi}}$, et des calculs $\rho : (q, \bot) \xrightarrow{w} (q', \bot)$,

$\rho_1 : (p, \bot) \xrightarrow{u} (p, \sigma_1)$, $\rho'_1 : (p', \sigma_1) \xrightarrow{u'} (p', \bot)$, $\rho_2 : (p, \bot) \xrightarrow{u} (q, \sigma_2)$, $\rho'_2 : (q', \sigma_2) \xrightarrow{u'} (p', \bot)$, $\rho_3 : (q, \bot) \xrightarrow{u} (q, \sigma_3)$ et $\rho'_3 : (q', \sigma_3) \xrightarrow{u'} (q', \bot)$ tels que $\langle \rho_1 \rangle > 1$ ou $\langle \rho'_1 \rangle > 1$ ou $\rho_1 \neq \rho_2$ ou $\rho'_1 \neq \rho'_2$. S satisfait donc la Propriété **S2**, et donc le coût de S est infini par le Lemme 4.2.2.

Nous montrons maintenant le sens $\Leftarrow$ Comme le coût de S est infini, par le Lemme 4.2.5, S satisfait la Propriété **S1** ou la Propriété **S2** dans S.

S **satisfait S1.** Il existe $u \in \Delta_{\mathsf{bi}}^*$ et des calculs $\rho_1 : (p, \bot) \xrightarrow{u}{}^*(p, \bot)$, $\rho_2 : (p, \bot) \xrightarrow{u}{}^*(q, \bot)$ et $\rho_3 : (q, \bot) \xrightarrow{u}{}^*(q, \bot)$ de A sur u avec $\langle \rho_1 \rangle > 1$ ou $\rho_1 \neq \rho_2$ Par la propriété 4.1 sur $\mathcal{S}_1$, nous avons $(p, p, p, q, q, q, \top) \in \mathcal{S}_1$, et donc par la règle 3.1, $\top \in \mathcal{S}_3$.

S **satisfait S2.** Il existe trois mots $uu', w \in \Delta_{\mathsf{bi}}^*$, et des calculs $\rho : (q, \bot) \xrightarrow{w} (q', \bot)$, $\rho_1 : (p, \bot) \xrightarrow{u} (p, \sigma_1)$, $\rho'_1 : (p', \sigma_1) \xrightarrow{u'} (p', \bot)$, $\rho_2 : (p, \bot) \xrightarrow{u} (q, \sigma_2)$, $\rho'_2 : (q', \sigma_2) \xrightarrow{u'} (p', \bot)$, $\rho_3 : (q, \bot) \xrightarrow{u} (q, \sigma_3)$ et $\rho'_3 : (q', \sigma_3) \xrightarrow{u'} (q', \bot)$ tels que $\langle \rho_1 \rangle > 1$ ou $\langle \rho'_1 \rangle > 1$ ou $\rho_1 \neq \rho_2$ ou $\rho'_1 \neq \rho'_2$. Par la définition de BI_A et la Propriété 4.2, $(q, q') \in \mathsf{BI}_A$ et $(p, p, p, q, q, q, q', q', q', p', p', p', \top) \in \mathcal{S}_2$, et donc par la règle 3.2 $\top \in \mathcal{S}_3$. □

De la Proposition 4.2.6 et du Lemme 1.4.10, comme la taille des ensembles $\mathcal{S}_1$ $\mathcal{S}_2$ et $\mathcal{S}_3$ est bornée polynomialement par le nombre d'états du $\mathcal{N}$-APVbi considéré, nous obtenons comme conséquence immédiate :

Proposition 4.2.7

> Décider si le coût d'un $\mathcal{N}$-APVbi réduit est infini est en PTIME.

Nous étendons maintenant ces résultats au $\mathcal{N}$-APV. Étant donné un $\mathcal{N}$-APV S, la construction etendre présentée au chapitre précédent nous permet de construire un $\mathcal{N}$-APVbi S' avec le même coût (Lemme 3.6.7). Il est ensuite possible de produire un $\mathcal{N}$-APVbi réduit S'' en utilisant la construction reduire présentée dans le même chapitre. Cet automate possédant le même coût, nous pouvons donc décider si le coût de S est fini en décidant si le coût de S'' est fini. Nous obtenons donc le résultat suivant :

Théorème 4.2.8

> Décider si le coût d'un $\mathcal{N}$-APV est infini est en PTIME.

4.3 Bornes finies du coût des automates à pile visible sur $\mathcal{N}$

4.3.1 Décidabilité du coût K-borné

Nous considérons un $\mathcal{N}$-APVbi $S = (A, \lambda)$ sur l'alphabet structuré Δ, avec $A = (Q, I, F, \Gamma, \delta)$ et un entier $K \in \mathbb{N}_{>0}$ représenté en binaire et nous décrivons un algorithme pour décider si $\mathsf{cout}(S) < K$.

La procédure utilise sur un ensemble de règles d'inférence construisant un sous-ensemble fini $\mathcal{M}$ des matrices de $\mathbb{N}^{Q,Q}$. Pour présenter la sémantique de ces matrices nous avons besoin d'introduire l'opérateur $\mathsf{Extra}_K : \mathbb{N} \to \mathbb{N}$ défini pour tout $z \in \mathbb{N}$ par :

$$\mathsf{Extra}_K(z) = \begin{cases} z & \text{si } z \leqslant K, \\ K & \text{sinon.} \end{cases}$$

Cet opérateur s'étend naturellement aux matrices d'entiers $M \in \mathbb{N}^{Q,Q}$ de la manière suivante :

$$\mathsf{Extra}_K(M) = M' \text{ avec } M'(p,q) = \mathsf{Extra}_K(M(p,q)) \text{ pour tout } p,q \in Q$$

Intuitivement Extra_K permet de borner un entier par une constante donnée. Quand nous souhaitons prouver que le coût d'un $\mathcal{N}$-APVbi est K-borné, il suffit de montrer qu'il existe un mot de coût au moins K. Nous n'avons pas besoin de précisions supplémentaires, le fait de savoir que le coût de ce mot est supérieur ou égal à K est suffisant. C'est là qu'intervient l'opérateur Extra_K. Soit $u \in \Sigma^*_{\mathsf{bi}}$ un mot bien imbriqué. Nous appelons $M_u \in \mathbb{N}^{Q,Q}$ la matrice définie comme suit pour tous états $p,q \in Q$:

$$M_u(p,q) = \mathsf{Extra}_K(\sum_{\rho \in \mu} \mathsf{cout}(\rho)) \text{ avec } \mu = \{(p,\bot) \xrightarrow{u,m}{}^*(q,\bot) \in \mathsf{Cal}_u(A)\}$$

Nous définissons l'ensemble $\mathcal{M}$ comme suit :

$$\mathcal{M} = \{M_u \mid u \in \Sigma^*_{\mathsf{bi}}\}$$

Pour tout $p,q \in Q$, $a \in \Sigma_\iota$, $c \in \Sigma_c$, $r \in \Sigma_r$ et $\gamma \in \Gamma$ nous introduisons les matrices de $\mathbb{N}^{Q,Q}$ suivantes :

$$M_\epsilon(p,q) = \begin{cases} 1 & \text{si } p = q, \\ 0 & \text{sinon.} \end{cases}$$

$$M_a(p,q) = \begin{cases} \mathsf{Extra}_K(\lambda(\tau)) & \text{si } \tau = (p,a,q) \in \delta_\iota, \\ 0 & \text{sinon.} \end{cases}$$

$$M_{c,\gamma}(p,q) = \begin{cases} \mathsf{Extra}_K(\lambda(\tau)) & \text{si } \tau = (p,c,\gamma,q) \in \delta_c, \\ 0 & \text{sinon.} \end{cases}$$

$$M_{r,\gamma}(p,q) = \begin{cases} \mathsf{Extra}_K(\lambda(\tau)) & \text{si } \tau = (p,\gamma,r,q) \in \delta_r, \\ 0 & \text{sinon.} \end{cases}$$

Nous présentons maintenant l'algorithme kborne calculant l'ensemble $\mathcal{M}$. Cet algorithme utilise les règles d'inférences présentées à la Figure 4.9.

$$\frac{}{M_\epsilon \in \mathcal{M}}\ (1) \qquad \frac{a \in \Sigma_\iota}{M_a \in \mathcal{M}}\ (2) \qquad \frac{M_1, M_2 \in \mathcal{M}}{\mathsf{Extra}_K(M_1 \cdot M_2) \in \mathcal{M}}\ (3)$$

$$\frac{M' \in \mathcal{M}, c \in \Sigma_c, r \in \Sigma_r}{\mathsf{Extra}_K(\sum_{\gamma \in \Gamma}(M_{c,\gamma} \cdot M' \cdot M_{r,\gamma})) \in \mathcal{M}}\ (4)$$

FIGURE 4.9 – Règles d'inférence pour décider du coût K-borné d'un $\mathcal{N}$-APVbi

Les règles d'inférences construisent les matrices sur des mots bien imbriqués de longueur croissante. Elles commencent avec les mots de taille 1, et étendent ces mots soit par concaténation, soit en ajoutant une paire d'appel et de retour en correspondance. Nous montrons maintenant le principal résultat de cette section à la Proposition 4.3.1, à savoir que kborne construit correctement $\mathcal{M}$. Nous nous en servirons ensuite pour prouver le Théorème 4.3.3

Proposition 4.3.1

Soit $S = (A, \lambda)$ avec $A = (Q, I, F, \Gamma, \delta)$ un $\mathcal{N}$-APVbi sur l'alphabet structuré Δ et $K \in \mathbb{N}_{>0}$ un entier. L'algorithme kborne construit l'ensemble $\mathcal{M}$ en temps exponentiel.

Démonstration. Nous montrons en premier que kborne construit correctement l'ensemble $\mathcal{M}$. Pour cela nous prouvons que $M \in \mathcal{M}$ si et seulement s'il existe un mot $u \in \Delta_{\mathsf{bi}}^*$ tel que $M = M_u$. Nous montrons d'abord par induction le sens $\Rightarrow$. Il y a quatre manières différentes d'ajouter la matrice M dans $\mathcal{M}$:

Règle 1 Nous avons $M = M_\epsilon$, il existe donc un mot $u = \epsilon$ tel que $M = M_u$.

Règle 2 Nous avons $M = M_a$, il existe donc un mot $u = a$ tel que $M = M_u$.

Règle 3 Il existe deux matrices $M_1, M_2 \in \mathcal{M}$ telles que $M = \mathsf{Extra}_K(M_1 \cdot M_2)$. Par hypothèse d'induction il existe deux mots $w, v \in \Delta_{\mathsf{bi}}^*$ tels que $M_1 = M_w$ et $M_2 = M_v$. Comme $M_{wv} = \mathsf{Extra}_K(M_w \cdot M_v)$, nous pouvons en conclure qu'il existe un mot $u = wv$ tel que $M = M_u$.

Règle 4 Il existe une matrice $M' \in \mathcal{M}$, $c \in \Delta_c$ et $r \in \Delta_r$ tels que $M = \mathsf{Extra}_K(\sum_{\gamma\in\Gamma}(M_{c,\gamma} \cdot M' \cdot M_{r,\gamma}))$. Par hypothèse d'induction il existe $v \in \Delta_{\mathsf{bi}}^*$ tel que $M' = M_v$. Comme $M_{cvr} = \mathsf{Extra}_K(\sum_{\gamma\in\Gamma}(M_{c,\gamma} \cdot M_v \cdot M_{r,\gamma}))$, nous pouvons en conclure qu'il existe un mot $u = cvr$ tel que $M = M_u$.

Nous montrons maintenant le sens $\Leftarrow$ par induction sur la structure imbriquée de u.

- Si $u = \epsilon$ alors nous avons $M_u = M_\epsilon$, et donc par la règle 1 nous avons $M \in \mathcal{M}$.
- Si $u = a$ avec $a \in \Sigma_\iota$ alors nous avons $M_u = M_a$, et par la règle 1 nous avons $M \in \mathcal{M}$.
- Si $u = wv$ avec $w, v \in \Sigma_{\mathsf{bi}}^*$, alors par hypothèse d'induction nous avons $M_w, M_v \in \mathcal{M}$. Comme $M_{wv} = \mathsf{Extra}_K(M_w \cdot M_v)$, par la règle 3 nous en concluons que $M \in \mathcal{M}$.
- Si $u = cvr$ avec $v \in \Sigma_{\mathsf{bi}}^*$, $c \in \Delta_c$ et $r \in \Delta_r$ alors par hypothèse d'induction $M_v \in \mathcal{M}$. Comme $M_{cvr} = \mathsf{Extra}_K(\sum_{\gamma\in\Gamma}(M_{c,\gamma} \cdot M_v \cdot M_{r,\gamma}))$, par la règle 4 nous en concluons que $M \in \mathcal{M}$.

L'appartenance à ExpTime suit le fait que les matrices construites sont de taille $\#Q \times \#Q$ où les entrées sont bornées par K. En effet, la procédure construit au pire $K^{\#Q^2}$ matrices. □

Proposition 4.3.2

Étant donné un $\mathcal{N}$-APVbi S et un entier $K \in \mathbb{N}_{>0}$, décider si $\mathsf{cout}(S) < K$ est ExpTime-complet.

Démonstration. Nous notons $S = (A, \lambda)$ avec $A = (Q, I, F, \Gamma, \delta)$. L'appartenance à ExpTime est amenée par la Proposition 4.3.1. En effet, il suffit de vérifier que parmi les matrices produites par kborne il existe une matrice M telle que :

$$(\sum_{p\in I, q\in F} M(p,q)) \geqslant K$$

Cela nous assure qu'il existe un mot $u \in \mathcal{L}(A)$ avec un coût supérieur ou égal à K. Pour la difficulté, nous pouvons procéder par réduction au problème du vide de l'intersection de K AA déterministes $(A_i)_{1\leqslant i\leqslant K}$ ([CDG+07]) de la manière suivante :

1. Construire l'union disjointe de $(A_i)_{1\leqslant i\leqslant K}$,
2. Construire un APVbi A équivalent en utilisant le Théorème 2.3.10,
3. Construire un $\mathcal{N}$-APVbi S à partir d' A avec un coût de 1 pour chaque transition.

Il est ensuite possible de montrer que l'intersection de $(A_i)_{1\leqslant i\leqslant K}$ est vide si et seulement si $\mathsf{cout}(S) < K$: comme les automates d'arbres sont déterministes et la construction AA2APV induit une bijection sur les calculs, chaque mot accepté par A possède un coût égal à K si et seulement si l'arbre t tel que $u = \mathsf{enc}(t)$ est accepté par tous les AA $(A_i)_{1\leqslant i\leqslant K}$, impliquant donc que leur intersection est non vide. □

Tout comme dans la section précédente nous pouvons construire en temps polynomial un $\mathcal{N}$-APV bien imbriqué depuis un $\mathcal{N}$-APV arbitraire avec le même coût, et nous étendons nos résultats aux $\mathcal{N}$-APV :

Théorème 4.3.3

> Étant donné un $\mathcal{N}$-APV S et un entier $K \in \mathbb{N}_{>0}$, décider si $\mathsf{cout}(S) < K$ est ExpTime-complet.

Calculer le coût d'un $\mathcal{N}$-APV avec un coût fini Considérons maintenant, étant donné un $\mathcal{N}$-APV ayant un coût fini, le problème consistant à calculer ce coût. Nous dérivons du précédent algorithme une procédure pour résoudre ce problème. La procédure explore simplement l'ensemble de matrices comme avant, sans utiliser l'opérateur Extra_K, jusqu'à la saturation des matrices. La terminaison de la procédure repose sur le fait que le coût S est fini et S est réduit. En fait, cela implique que les entrées des matrices sont bornées par $\mathsf{cout}(S)$. En particulier, cela prouve que le nombre de matrices construites est borné par $\mathsf{cout}(S)^{n^2}$, et :

Théorème 4.3.4

> Pour tout $\mathcal{N}$-APV S avec un coût fini, la valeur de $\mathsf{cout}(S)$ peut être calculée en temps $\mathsf{cout}(S)^{O(n^2)}$.

4.3.2 Décidabilité du coût K-borné (pour K fixé)

Nous allons maintenant étudier le problème du coût K-borné où K est un paramètre fixé et l'entrée est seulement le $\mathcal{N}$-APV S et nous demandons si $\mathsf{cout}(S) < K$. L'algorithme de la Section 4.3.1 montre que pour un entier fixé K ce problème peut-être résolu en temps exponentiel ; cependant, en adaptant la technique utilisée dans [Sei90] pour l'ambiguïté des AA nous avons :

Théorème 4.3.5

> Soit $K \in \mathbb{N}_{>0}$ un entier fixé. Pour tout $\mathcal{N}$-APV réduit S, décider si $\mathsf{cout}(S) < K$ est dans PTime.

Pour prouver ce résultat, étant donné un $\mathcal{N}$-APV $S = (A, \lambda)$ avec $A = (Q, I, F, \delta, \Gamma)$ et un entier K fixé nous allons construire une famille de APV $(A_i)_{i\in[1,K]}$ respectant la propriété suivante : A_i est un APV acceptant les mots u tels qu'il y a i différents calculs acceptants $\rho_1, \ldots, \rho_i$ de A sur u vérifiant $\sum_{1\leqslant j\leqslant i} \mathsf{cout}(\rho_j) \geqslant K$. Pour cela A_i simule en parallèle i calculs de A sur le même mot, et pour chacun garde une trace du coût actuel dans l'état en la calculant jusqu'à K.

Nous allons maintenant définir un ensemble de matrices qui nous permettra de différencier deux calculs. Nous appelons $\mathbb{B}$ l'ensemble $\{0, 1\}$, et définissons des matrices de $\mathbb{B}^{[1,K]\times[1,K]}$ de la manière suivante :

$$\text{Pour tout } j,k \in [1,i] \text{ et } (q_1,\dots q_i) \in Q^i, \qquad \mathbf{0}_i^{q_1,\dots,q_i}(j,k) = 0 \text{ si } q_j = q_k \text{ et } 1 \text{ sinon.}$$
$$\text{Pour tout } j,k \in [1,i], \qquad \overline{\mathbf{Id}}_i(j,k) = 0 \text{ si } j = k \text{ et } 1 \text{ sinon.}$$

Pour tout $i \in [1,K]$, chaque état de A_i va stocker une matrice de $M \in \mathbb{B}^{[1,K]\times[1,K]}$ afin de mémoriser si les calculs qu'il simule sont différents ou non. Ainsi, pour tout $k,l \in [1,i]$ nous aurons $M(k,l) = 0$ si et seulement les calculs l et k sont différents. Les matrices $\mathbf{0}_i^{q_1,\dots,q_i}$ et $\overline{\mathbf{Id}}_i$ vont nous servir pour initialiser les états initiaux et finaux. Ainsi $\overline{\mathbf{Id}}_i$ représente un état où tous les calculs sont différents les uns des autres. Nous présentons maintenant la construction kborne produisant cette famille de APV.

Définition 4.3.6

Soient $K \in \mathbb{N}_{>0}$ un entier fixé, $S = (A,\lambda)$ un $\mathcal{N}$-APV avec $A = (Q,I,F,\Gamma,\delta)$ sur l'alphabet structuré Δ. La famille de APV $\mathsf{kborne}(S) = (A_i)_{1\leqslant i\leqslant K} = (Q_i,I_i,F_i,\Gamma_i,\delta_i)$ sur Δ est définie par :

- $Q_i = ([1,K]\times[1,K])^i \times \mathbb{B}^{[1,i],[1,i]}$
- $\Gamma_i = (\Gamma)^i$
- $I_i = \{\ ((q_1,1),\dots,(q_i,1)) \times \mathbf{0}_i^{q_1,\dots,q_i}\} \mid (q_1,\dots,q_i) \in I^i \ \}$
- $F_i = \{\ ((q_1,m_1),\dots,(q_i,m_i)) \times \{\overline{\mathbf{Id}}_i\} \mid (q_1,\dots,q_i) \in F^i, (\sum_{1\leqslant j\leqslant i} m_j) \geqslant K \ \}$
- $\delta_i = \delta_i^c \uplus \delta_i^l \uplus \delta_i^r$ avec δ_i^l et δ_i^r définis de manière similaire :
  - $\delta_i^c = \left\{ \begin{array}{c|c} (((q_1,m_1),\dots,(q_i,m_i),M), & c\in\Sigma_c, \text{pour tout } 1\leqslant j,l\leqslant i, \\ c,(\gamma_1,\dots,\gamma_i), & \tau_j = (q_j,c,\gamma_j,q_j'),\tau_l = (q_l,c,\gamma_l,q_l') \in \delta_c \\ ((q_1',m_1'),\dots,(q_i',m_i'),M')) & m_j' = \mathsf{Extra}_K(\lambda((q_j,c,\gamma_j,q_j')) * m_j) \text{ et} \\ & M'(j,l) = M(j,l) \vee (\tau_j \neq \tau_l) \end{array} \right\}$

Proposition 4.3.7

Soient $K \in \mathbb{N}_{>0}$ un entier fixé, $S = (A,\lambda)$ un $\mathcal{N}$-APV réduit sur l'alphabet structuré Δ et $(A_i)_{1\leqslant i\leqslant K}$ la famille de APV $\mathsf{kborne}(S)$. Pour tout $u \in \Delta^*$ et $M \in \mathbb{B}^{[1,i],[1,i]}$ il existe des entiers $n_j, m_j \in \mathbb{N}$ pour tout $j \in [1,i]$ tels que $n_j = \mathsf{Extra}_K(m_j)$ et :

$$\exists(((q_1,1),\dots,(q_i,1),\mathbf{0}_i^{q_1,\dots,q_i}),\perp^i) \xrightarrow{u}{}^*(((p_1,n_1),\dots,(p_i,n_i),M),(\sigma_1,\dots,\sigma_i)) \in \mathsf{Cal}_u(A_i)$$
$$\Leftrightarrow \exists \rho_j : (q_j,\perp) \xrightarrow{u,m_j}{}^*(p_j,\sigma_j) \in \mathsf{Cal}_u(A) \text{ et } (\rho_j \neq \rho_k) = M(j,k) \text{ pour tout } j,k \in [1,i]$$

Démonstration. Nous notons $A = (Q,I,F,\Gamma,\delta)$. Nous faisons la preuve par induction sur la longueur de u. La propriété est trivialement correcte quand $u = \epsilon$, par la définition de I_i et $\mathbf{0}_i^{q_1,\dots,q_i}$. Nous considérons maintenant que $|u| > 0$, il existe alors une décomposition de u en va avec $v \in \Sigma^*$ et $a \in \Sigma$. Il y a trois cas différents, selon le type de a. Nous considérons le cas $a \in \Sigma_c$.

Nous montrons en premier le sens $\Rightarrow$. Il existe $p_j' \in Q$, $n_j' \in \mathbb{N}_{>0}$ et $M' \in \mathbb{B}^{[1,i],[1,i]}$ pour tout $j \in [1,i]$ tels que :

$$(((p_1',n_1'),\dots,(p_i',n_i'),M'),a,(\gamma_1,\dots,\gamma_i),((p_1,n_1),\dots,(p_i,n_i),M) \in \delta_i^l$$
$$(((q_1,1),\dots,(q_i,1),\mathbf{0}_i^{q_1,\dots,q_i}),\perp^i) \xrightarrow{v}{}^*(((p_1',n_1'),\dots,(p_i',n_i'),M'),(\sigma_1,\dots,\sigma_i)) \in \mathsf{Cal}_v(A_i)$$

Par hypothèse d'induction pour tout $j,k \in [1,i]$ il existe $\rho'_j : (q_j,\bot) \xrightarrow{v,m'_j}{}^*(p'_j,\sigma_j) \in \mathsf{Cal}_v(A)$ avec $\mathsf{Extra}_K(m'_j) = n'_j$ et $(\rho'_j \neq \rho'_k) = M'(j,k)$. Par construction de δ^c_i pour tout $j \in [1,i]$ il existe $(p'_j,a,\gamma_j,p_j) \in \delta_c$. Par construction de M' et $n'_1,\dots,n'_i$ nous en déduisons que pour tout $j,k \in [1,i]$ il existe $\rho_j : (q_j,\bot) \xrightarrow{u,m_j}{}^*(p_j,\sigma_j) \in \mathsf{Cal}_u(A)$ avec $(\rho_j \neq \rho_k) = M(j,k)$.

Nous montrons le sens $\Leftarrow$. Pour tout $j \in [1,i]$ il existe $p'_j \in Q$, $\gamma_j \in \Gamma$ et $n'_j \in \mathbb{N}_{>0}$ tels que $(p'_j,a,\gamma_j,p_j) \in \delta_\iota$ et $\rho'_j : (q_j,\bot) \xrightarrow{v,n'_j}{}^*(p'_j,\sigma_j) \in \mathsf{Cal}_v(A)$. Par hypothèse d'induction nous avons :

$$(((q_1,1),\dots,(q_i,1),\mathbf{0}_i^{q_1,\dots,q_i}),\bot^i) \xrightarrow{v}{}^*(((p'_1,n'_1),\dots,(p'_i,n'_i),M'),(\sigma_1,\dots,\sigma_i)) \in \mathsf{Cal}_v(A_i)$$

avec $\mathsf{Extra}_K(m'_j) = n'_j$ et $(\rho'_j \neq \rho'_k) = M'(j,k)$ pour tout $j,k \in [1,i]$. Par construction de δ^c_i nous avons :

$$(((p'_1,n'_1),\dots,(p'_i,n'_i),M'),a,(\gamma_1,\dots,\gamma_i),((p_1,n_1),\dots,(p_i,n_i),M) \in \delta^\iota_i$$

Nous en déduisons qu'il existe :

$$(((q_1,1),\dots,(q_i,1),\mathbf{0}_i^{q_1,\dots,q_i}),\bot^i) \xrightarrow{u}{}^*(((p_1,n_1),\dots,(p_i,n_i),M),(\sigma_1,\dots,\sigma_i)) \in \mathsf{Cal}_u(A_i)$$

Les cas où $a \in \Sigma_r$ ou $a \in \Sigma_\iota$ sont similaires au premier, le symbole de pile étant la seule différence et ne change pas la preuve. □

Nous pouvons maintenant prouver le Théorème 4.3.5.

Démonstration. [*Théorème 4.3.5*] Nous notons $S = (A,\lambda)$ avec $A = (Q,I,F,\Gamma,\delta)$. Par la Proposition 4.3.7, et par la construction des états finaux de A_i, nous pouvons déduire que pour tout $i \leqslant K$, $\mathcal{L}(A_i) = \varnothing$ si et seulement s'il existe un mot $u \in \Delta^*_{\mathsf{bi}}$ et $\rho_1,\dots\rho_i$ calculs acceptants de A différents deux à deux tels que $\sum_{j\in[1,i]} \mathsf{cout}(\rho_j) \geqslant K$. Il est évident que A_i peut-être construit en temps polynomial en $\#Q$. Finalement nous testons en temps polynomial la vacuité de chacun des K APV A_i. □

4.4 Retour aux arbres

Nous avons montré dans les sections précédentes comment décider si le coût d'un $\mathcal{N}$-APV est fini ou non. En utilisant la construction AA2APV présentée à la Section 2.3 nous allons maintenant montrer comment décider que le coût d'un $\mathcal{N}$-AA est fini.

Pour cela, étant donné un $\mathcal{N}$-AA S sur l'alphabet (Σ,ar) nous énonçons la propriété suivante, caractérisant l'infinité de S :

Propriété N1 S a la propriété **(N1)** s'il existe une variable x_1 et un arbre $\mathcal{T}_\Sigma(\{x_1\})$ tel que :

$\rho_1 : (p,(p))$-calcul, $\rho_2 : (p,(q))$-calcul, $\rho_3 : (q,(q))$-calcul $\in \mathsf{Cal}_\iota(S)$

avec $\mathsf{cout}(\rho_1) > 1$ ou $p \neq q$ (voir la Figure 4.10).

Théorème 4.4.1

> Soit S un $\mathcal{N}$-AA sur l'alphabet (Σ,ar). Le coût de S est infini si et seulement si S vérifie la propriété **(N1)**.

Pour prouver ce théorème nous allons montrer que S satisfait la propriété **(N1)** si et seulement si $S' = \mathsf{AA2APV}(S)$ satisfait les propriétés **(S1)** et **(S2)**. Le $\mathcal{N}$-APV S' est bien imbriqué mais n'est pas nécessairement réduit, or nous avons besoin de la réduction pour appliquer le Théorème 4.2.1. Nous allons donc montrer que si S est émondé alors S' est réduit.

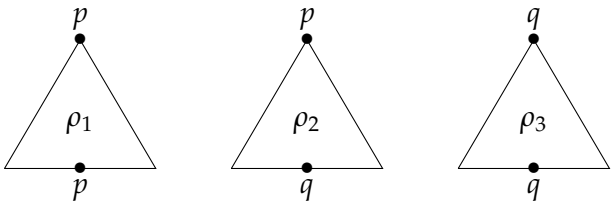


FIGURE 4.10 – La propriété **(N1)**.

Lemme 4.4.2

Soit $A = (Q, I, \delta)$ un AA sur l'alphabet gradué (Σ, ar). Si A est émondé alors le APVbi $\mathsf{SAA2SAPV}(A)$ est réduit.

Démonstration. Nous notons $A' = (Q', I', F', \delta', \Gamma')$ l' APVbi $\mathsf{AA2APV}(A)$. Nous montrons en premier la Condition 1 de la définition 3.1.3. Soit $\rho : ((x,n), \bot) \xrightarrow{u}{}^*((y,m), \sigma)$ un calcul de A' sur $\hat{\Sigma}^*$ avec $(x,n) \in I'$. Nous faisons la preuve par induction sur la taille de σ.

Si $\sigma = \bot$ alors soit $(y,m) \in F$ soit $u = \epsilon$. En effet, comme $\hat{\Sigma}_\iota = \varnothing$, si $u \neq \epsilon$ alors comme $\sigma = \bot$ nous pouvons décomposer ρ par :

$$\rho : ((x,n), \bot) \xrightarrow{c} ((x',n'), \gamma) \xrightarrow{u}{}^* ((x'',n''), \gamma) \xrightarrow{r} ((y',m'), \bot) \xrightarrow{w} ((y,m), \bot)$$

avec $\gamma \in \Gamma'$, $c \in \hat{\Sigma}_c$ et $r \in \hat{\Sigma}_r$. En appliquant le Lemme 2.3.11 nous obtenons que $y' = x$ et $m' = n+1$. L'état (y',m') est donc final et comme pour tout $\tau \in \delta'_c$ nous avons $\mathsf{source}(\tau) \neq (y',m')$ nous concluons que $w = \epsilon$ et $(y,m) \in F$, donc A satisfait la condition 1. Si $u = \epsilon$ comme S est émondé il existe un calcul acceptant $\rho' = \tau(\rho|_1, \dots \rho|_m)$ de S tel que $\tau \in \delta$, $\mathsf{racine}(\tau) = x$ et $m \in \mathbb{N}$. Par le Lemme 2.3.14 $\Phi_{\mathsf{av}}(\rho, (x,n), \bot)$ est un calcul de A' et A' satisfait la condition 1.

Nous supposons qu'il existe $n \in \mathbb{N}_{>0}$ tel que A satisfait la condition 1 si $|\sigma| \leqslant n$, et nous considérons que $|\sigma| = n+1$. Nous pouvons décomposer ρ de la manière suivante :

$$\rho : ((x,n), \bot) \xrightarrow{v}{}^* ((x',n'), \sigma') \xrightarrow{c} ((y',m'), \sigma) \xrightarrow{w} ((y,m), \sigma)$$

avec $c \in \hat{\Sigma}$ et $\sigma = \sigma'((x',n'), c)$. Nous appelons l l'entier défini par 1 si $x \in Q$ et par $\mathsf{ar}(x)$ sinon. Par Le lemme 2.3.11 nous avons $y' = y$ et il existe $k \in \mathbb{N}_{>0}$ tel que $m = n' + k$. Comme A est émondé, pour tout $i \in [m, l-1]$ il existe un calcul ρ'_i de A tel que $(y', i) \in \mathsf{anc}(\rho'_i\langle\epsilon\rangle))$ et $\Phi_{\mathsf{av}}(\rho'_i, (y',i), \sigma)$ est un calcul de A'. De plus par définition de δ' il existe une transition $((y,l), ((x',n'), c), r, (x', n'+1)) \in \delta'_r$, et par hypothèse d'induction A' satisfait la Condition 1.

Nous montrons maintenant la Condition 3 de la définition 3.1.3. Nous considérons un état (x,n) de Q'. Comme A est émondé, il existe un calcul acceptant ρ de A et une position $e \in \mathsf{pos}(\rho)$ telle que si $x \in \delta$ alors $\rho\langle e\rangle = x$ et si $x \in Q$ alors $\mathsf{racine}(\rho\langle e\rangle) = x$. Comme il existe $p \in I$ tel que $\Phi_{\mathsf{av}}(\rho, (p,0), \bot)$ est un calcul acceptant de A', nous concluons que A satisfait la Condition 3. □

Nous pouvons maintenant prouver le Théorème 4.4.1 :

Démonstration du Théorème 4.4.1. Nous montrons en premier l'implication $\Rightarrow$. Comme le coût de S est infini, par le Théorème 2.3.18 le coût de $S' = \mathsf{AA2APV}(S)$ est aussi infini. Par le Lemme 4.4.2 S' est réduit et nous pouvons en déduire par le Théorème 4.2.1 que S' vérifie **(S2)**. S' ne peut pas vérifier la condition **(S1)** car par la définition de la construction AA2APV, pour tout calcul $(p, \bot) \xrightarrow{u}{}^* (q, \bot) \in \mathsf{Cal}_u(S')$ nous avons $p \neq q$. Il est alors facile de vérifier que S satisfait la propriété **(N1)**.

Nous montrons maintenant l'implication $\Leftarrow$. Comme S vérifie la propriété **(N1)**, par la définition de la construction AA2APV, $S' = \mathsf{AA2APV}(S)$ vérifie la propriété **(S2)**. Par le Lemme 4.4.2 S' est réduit et par le Théorème 4.2.1 nous pouvons donc conclure que le coût de S' est infini, et par le Théorème 2.3.18 le coût de S est aussi infini. □

4.5 Décidabilité du coût infini des automates à pile visible sur Max-plus

Nous introduisons dans cette section le semi-anneau Max-plus.

Semi-anneau Max-plus Max-plus est le semi-anneau $(\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)$ avec max le maximum de deux entiers et $+$ l'addition. Nous appellerons ce semi-anneau $\mathcal{MP}$.

La différence entre ce modèle et les $\mathcal{N}$-APV est la manière dont se calcule le coût d'un mot. Dans le cas des $\mathcal{N}$-APV, le coût d'un mot w est influencé par celui de tous ses calculs, car il est nécessaire d'en faire la somme afin de calculer le coût de w. Pour les $\mathcal{MP}$-APV c'est un plus simple, car il n'y a pas besoin de considérer tous les calculs, simplement celui avec le coût le plus élevé.

Le problème nous concernant est celui de l'infinité du coût, c'est à dire déterminer si le coût d'un $\mathcal{MP}$-APV est borné ou non. Nous avons discuté dans la Section 4.1 du travail de Helmut Seidl présenté dans [Sei94b] sur un modèle d'automate d'arbres à coûts utilisant des polynômes, où le coût de l'automate est simplement l'ensemble des coûts de ses calculs acceptants. Il présente notamment une procédure pour décider si cet ensemble est fini lorsque l'on considère le semi-anneau Max-plus. Concernant les $\mathcal{MP}$-AA, le coût de l'automate est égal à celui du mot accepté au coût maximal, et le coût de chaque mot est égal à celui du calcul acceptant au coût maximal sur ce mot. Nous pouvons donc en conclure que dans le cas de Max-plus, le coût de l'automate est égal à celui du calcul acceptant au coût maximal, ce qui revient donc à résoudre le même problème que Helmut Seidl. Il est ensuite assez simple de modifier sa procédure pour déterminer si le coût de l'automate est fini. Par le Théorème 3.4.8 statuant l'équivalence faible entre les APV à coûts et les AA à coûts nous obtenons une procédure pour déterminer si le coût d'un $\mathcal{MP}$-APV est fini ou non.

Nous allons donc adapter sa procédure pour les $\mathcal{MP}$-AA puis donner une caractérisation pour les $\mathcal{MP}$-APV sous forme de deux propriétés nécessaires est suffisantes à l'infinité, et montrer qu'il est possible de décider si un $\mathcal{MP}$-APV les satisfait en temps polynomial.

Premièrement dans la Section 4.5.1 nous présenterons la procédure dans le cas des $\mathcal{MP}$-AA. Puis dans la Sous-section 4.5.2, nous parlerons la caractérisation, et dans la Sous-section 4.5.3 d'un algorithme pour décider de sa présence en temps polynomial.

4.5.1 Automates d'arbres à coûts sur Max-plus

Dans [Sei94b] Helmut Seidl présente une procédure pour résoudre un problème de borne sur des automates d'arbre à coûts équipés de polynômes. Nous allons adapter ce résultat ici aux $\mathcal{MP}$-AA émondés.

Nous présentons une propriété caractérisant le coût infini d'un $\mathcal{MP}$-AA :

Théorème 4.5.1

> Soit $S = (A, \lambda)$ un $\mathcal{MP}$-AA émondé et $A = (Q, I, \delta)$ un AA sur l'alphabet (Σ, ar). Le coût de S est infini si et seulement s'il existe un $(p, (p))$-calcul ρ tel que $\mathsf{cout}(\rho)) > 0$.

S doit être émondé pour assurer le fait que ρ puisse être complété en un calcul acceptant, et l'inégalité $\mathsf{cout}(\rho) > 0$ assure que la boucle sur p produise un coût de plus en plus grand (et donc infini).

Dans son article Hemult Seidl présente une propriété équivalente utilisant un automate d'arbres *parameter-reduced*, où un bit d'information est stocké dans chaque état pour déterminer s'il est possible de produire autre chose que $-\infty$ sur un arbre enraciné à cet état. Il doit faire ça car il suffit de passer par une seule transition avec un coût de $-\infty$ pour obtenir un calcul avec un coût de $-\infty$. Dans notre cas nous n'en avons pas besoin car notre définition d'automate à coûts implique qu'une transition ne pas être étiquetée par l'élément absorbant du semi-anneau, ici $-\infty$. Si nous souhaitons appliquer notre procédure sur un automate d'arbres possédant des transitions avec un coût de $-\infty$, il nous suffit de vérifier que son coût est différent de $-\infty$ puis de supprimer les transitions avec un coût de $-\infty$. Le coût différent de $-\infty$ de l'automate d'arbre initial nous assure que l'automate sans les transitions gênantes défini le même coût.

Démonstration. Nous montrons en premier l'implication $\Rightarrow$. Comme le coût de S est infini il existe un calcul acceptant ρ sur un arbre t tel que $\mathsf{cout}(\rho) > L_S M_S^{\#Q}$ avec $M_S = \max(\{n \in \mathbb{N} \mid \Sigma_n \neq \varnothing\})$ et $L_S = \max(\{m \in \mathbb{N} \mid \tau \in \delta, m = \lambda(\tau)\})$. Nous considérons la décomposition de ρ suivante :

$$\rho = \rho_\epsilon \text{ et } \rho_o = \tau_o(\rho_{o.1}, \ldots, \rho_{o.n}) \text{ pour tout } o \in \mathsf{pos}(t)$$

Comme $\mathsf{cout}(\rho) > L_S M_S^{\#Q}$ nous pouvons en déduire qu'un même état p apparaît au moins deux fois dans une branche du calcul, c'est à dire qu'il existe $o, o' \in \mathsf{pos}(t)$ tels que o est un préfixe strict de o', $\mathsf{racine}(\tau_o) = \mathsf{racine}(\tau_{o'})$ et $\mathsf{cout}(\rho_o) > \mathsf{cout}(\rho_{o'})$. Nous avons donc trouvé un état $p = \mathsf{racine}(\tau_o)$ tel qu'il existe un $(p, (p))$-calcul ρ' et $\mathsf{cout}(\rho') > 0$, ce qui conclut la preuve.

Nous montrons maintenant l'implication $\Leftarrow$. Pour tout $i \in \mathbb{N}$, ρ^i est un $(p, (p))$-calcul de S tel que $\mathsf{cout}(\rho^i) = \mathsf{cout}(\rho)^i$. Comme S est émondé, il existe un $(q, (p))$-calcul ρ_1 de S avec $q \in I$ et un p-calcul ρ_2 de S tels que $\rho_1 \rho^i \rho_2$ est un calcul acceptant de S et $\mathsf{cout}(\rho_1 \rho^i \rho_2) \geqslant \mathsf{cout}(\rho)^i$ pour tout $i \in \mathbb{N}$. Le coût de S est donc infini, ceci concluant la preuve. □

4.5.2 Caractérisation du coût infini d'un automate à pile visible sur Max-plus

Nous allons maintenant étendre les résultats présentés à la Section 4.5.1 aux $\mathcal{MP}$-APV. Nous ne considérerons ici que le cas des APVbi à coût sur le semi-anneau Max-plus, que nous appellerons $\mathcal{MP}$-APVbi. Le résultat peut ensuite être étendu au cas général comme il a été fait pour les $\mathcal{N}$-APV.

Soit $S = (A, \lambda)$ un $\mathcal{MP}$-APVbi sur l'alphabet structuré Δ avec $A = (Q, I, F, \Gamma, \delta)$. Nous introduisons deux propriétés sur les $\mathcal{N}$-APVbi, **(MP1)** et **(MP2)**.

Propriété MP1 S a la propriété **(MP1)** s'il existe $u \in \Delta_{\mathsf{bi}}^*$ tel que :

$$\rho_1 : (p, \bot) \xrightarrow{u, m_1}{}^* (p, \bot) \in \mathsf{Cal}_u(A)$$

avec $m_1 > 0$ (voir la Figure 4.11).

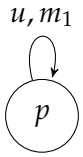


FIGURE 4.11 – La propriété **(MP1)**.

Propriété MP2 S a la propriété **(MP2)** s'il existe $uv, w \in \Delta^*_{\mathsf{bi}}$ tels que $u \notin \Delta^*_{\mathsf{bi}}$ et :

$\rho_1 : (p, \bot) \xrightarrow{u,m_1}{}^*(p, \sigma_1) \in \mathsf{Cal}_u(A)$

$\rho : (p, \bot) \xrightarrow{w,m}{}^*(q, \bot) \in \mathsf{Cal}_w(A)$

$\rho'_1 : (q, \sigma_1) \xrightarrow{v,m'_1}{}^*(q, \bot) \in \mathsf{Cal}_v(A)$

avec $m_1 > 0$ ou $m'_1 > 0$ (voir la Figure 4.12).

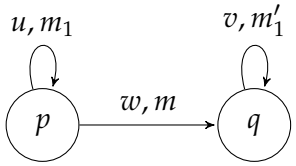


FIGURE 4.12 – La propriété **(MP2)**.

Nous allons montrer dans cette section que ces deux propriétés sur les $\mathcal{MP}$-APVbi caractérisent l'infinité du coût.

Théorème 4.5.2

> Soit S un $\mathcal{MP}$-APVbi émondé sur l'alphabet structuré Δ. Le coût de S est infini si et seulement si S satisfait l'une des propriétés **(MP1)** et **(MP2)**.

Pour prouver ce théorème nous allons procéder de la même manière que pour le Théorème 4.4.1. Nous devons donc montrer dans un premier temps que si S est émondé alors $S' = \mathsf{APV2AA}(S)$ est aussi émondé.

Lemme 4.5.3

> Soit $A = (Q, I, F, \delta, \Gamma)$ un APV sur l'alphabet structuré Δ. Si A est émondé alors $\mathsf{APV2AA}(S)$ l'est aussi.

Démonstration. Nous notons $A' = (Q', I', \delta')$ l' AA produit par $\mathsf{APV2AA}(A)$. Soit (p, q) un état de Q'. Comme S est émondé et par définition des états de A' le calcul suivant est un calcul acceptant de A :

$$\rho : (p_i, \bot) \xrightarrow{u}{}^*(p, \sigma) \xrightarrow{v}{}^*(q, \sigma) \xrightarrow{w}{}^*(p_f, \bot)$$

Avec $p_i \in I$ et $p_f \in F$. Par le Lemme 2.3.6 il existe un $(p_i, q_f, \bot)$-calcul acceptant ρ' de A' tel que $\Phi_{\mathsf{va}}(\rho', \bot) = \rho$, et donc A' est émondé. □

Nous pouvons maintenant montrer le Théorème 4.5.2 :

Démonstration du Théorème 4.5.2. Soit $S = (A, \lambda)$ avec $A = (Q, I, F, \Gamma, \delta)$ un APVbi. Nous appelons S' l' AA APV2AA(S) avec $S' = (A', \lambda')$ et $A' = (Q', I', \delta')$. Comme S est émondé, par le Lemme 4.5.3, S' l'est aussi. Par le Théorème 2.3.18 nous savons que $\mathsf{cout}(S) = \mathsf{cout}(S')$ et donc le coût de S est infini si et seulement si le coût de S' l'est aussi.

Nous montrons le sens $\Rightarrow$. Par le Théorème 4.5.1 comme le coût de S' est infini, il existe un $(p, q, \gamma, (p, q, \gamma))$-calcul ρ' de A' avec $(p, q) \in Q \times Q$ et $\mathsf{cout}(\rho') > 0$. Par le Lemme 2.3.5 cela signifie qu'il existe un calcul $(p, \bot) \xrightarrow{u}{}^* (p, \sigma) \xrightarrow{w}{}^* (q, \sigma) \xrightarrow{v}{}^* (q, \bot) \in \mathsf{Cal}_{uwv}(A)$. Si $\sigma = \bot$ alors S satisfait **(MP1)**, sinon S satisfait **(MP2)**.

Nous montrons maintenant le sens $\Leftarrow$. Comme S satisfait la propriété **(MP1)** ou la propriété **(MP2)**, par le Lemme 2.3.6 cela signifie qu'il existe $\gamma \in \Gamma$ et un $(p, q, \gamma, ((p, q, \gamma)))$-calcul ρ' de A' avec $(p, q) \in Q \times Q$ et $\mathsf{cout}(\rho') > 0$. Nous savons par le Théorème 4.5.1 que le coût de S' est infini, et celui de S l'est aussi. □

4.5.3 La décision du coût infini est dans PTIME

Nous allons maintenant montrer que nous pouvons décider de la présence de **(MP1)** et **(MP2)** en temps polynomial, permettant ainsi de décider de l'infinité du coût d'un $\mathcal{MP}$-APV.

L'algorithme (Figure 4.13) défini trois types de règles d'inférence utilisées comme une procédure de saturation, et fonctionne de la même manière que pour le Lemme 4.2.6. L'ensemble $\mathcal{S}_3$ permet d'assurer qu'un tuple de $\mathcal{S}_1$ satisfait la propriété **(MP1)** (règle 3.1), ou qu'un tuple de $\mathcal{S}_2$ satisfait la propriété **(MP2)** (règle 3.2).

$$\frac{p \in Q}{(p, p, 0) \in \mathcal{S}_1}\ (1.1) \qquad \frac{\tau = (p, a, q) \in \delta_\iota}{(p, q, \lambda(\tau) > 0) \in \mathcal{S}_1}\ (1.2) \qquad \frac{(p, q', B')(q', q, B'') \in \mathcal{S}_1}{(p, q, B' \vee B'') \in \mathcal{S}_1}\ (1.3)$$

$$\frac{(p', q', B') \in \mathcal{S}_1, \tau = (p, c, \gamma, p') \in \delta_c, \tau' = (q', \gamma, r, q) \in \delta_r}{(p, q, B' \vee \lambda(\tau) > 0 \vee \lambda(\tau') > 0) \in \mathcal{S}_1}\ (1.4)$$

$$\frac{(p, q, B') \in \mathcal{S}_1, (q', p', B'') \in \mathcal{S}_1}{(p, q, q', p', B' \vee B'') \in \mathcal{S}_2}\ (2.1) \qquad \frac{\tau = (p, c, \gamma, q) \in \delta_c \tau = (q', \gamma, r, p') \in \delta_r}{(p, q, q', p', \lambda(\tau) > 0 \vee \lambda(\tau') > 0) \in \mathcal{S}_2}\ (2.2)$$

$$\frac{(p, p'', q'', p', B'), (p'', q, q', q'', B'') \in \mathcal{S}_2}{(p, q, q', p', B' \vee B'') \in \mathcal{S}_2}\ (2.3)$$

$$\frac{(p, p, \top) \in \mathcal{S}_1}{(\top) \in \mathcal{S}_3}\ (3.1) \qquad \frac{\begin{array}{c}(q, q') \in \mathsf{BI}_A \\ (p, q, q', p, \top) \in \mathcal{S}_2\end{array}}{(\top) \in \mathcal{S}_3}\ (3.2)$$

FIGURE 4.13 – Règles d'inférence pour décider le coût infini d'un $\mathcal{MP}$-APV.

Théorème 4.5.4

> Soit $S = (A, \lambda)$ un $\mathcal{MP}$-APVbi. Il est décidable en PTIME si le coût de S est infini.

Démonstration. La preuve suit les même lignes que la preuve du Lemme 4.2.6. □

Chapitre 5

Décidabilité de l'infinité de la norme d'un transducteur à pile visible

Dans ce chapitre, nous allons étudier un problème de décidabilité sur le modèle de transducteurs à pile visible.

La norme d'un transducteur est le nombre maximal de sorties différentes qu'un mot peut produire. Cette norme peut être finie ou infinie. Il existe deux problèmes classiques de bornes concernant la norme d'un transducteur. Le premier consiste à décider si, étant donné un entier k, la norme d'un transducteur est bornée par k. Cette question a été étudiée depuis longtemps sur plusieurs modèles, et les applications sont multiples. Par exemple, dans le cas des transducteurs finis ([Web93]) et des transducteurs d'arbres ([Sei94a]) il est possible de tester si deux transducteurs sont équivalents (c'est-à-dire expriment la même transformation) si leurs normes sont bornées par un entier k.

Concernant les transducteurs finis, il a tout d'abord été prouvé dans [Sch76] et [BH77] qu'il est possible de décider si la transformation réalisée est fonctionnelle, c'est à dire s'il y a au plus une sortie par mot d'entrée. Une preuve différente de ce résultat est aussi disponible dans [BCPS03]. Puis, Tat-hung Chan et Oscar H. Ibarra ont montré dans [CI83] la décidabilité de la norme infinie dans le cas où le transducteur fini est lettre à lettre. Le cas général a été finalement prouvé par Andreas Weber dans [Web90], où il présente deux propriétés décidables en temps polynomial caractérisant l'infinité de la norme. Un résultat plus récent est disponible dans [SdS08] par Jacques Sakarovitch et Rodrigo de Souza. Les propriétés sont toujours présentes, mais la preuve de la nécessité est plus élégante et est basée sur la décidabilité du coût fini d'un automate fini à coûts. Dans [Sei92], les résultats de [Web90] sont étendus aux transducteurs d'arbres. La preuve est similaire, et Helmut Seidl propose une extension des propriétés de Andreas Weber caractérisant l'infinité de la norme des transducteurs d'arbres.

Dans le cas des transducteurs à pile visible un certain nombre de résultats est présenté dans [FRR+10]. Premièrement, il est montré que la fonctionnalité est décidable en temps polynomial. Il est ensuite prouvé que l'équivalence des transducteurs à pile visible fonctionnels est un problème EXPTIME-complet. Il est enfin donné un algorithme décidant si, étant donné un entier k, la norme d'un transducteur est bornée par k. Cet algorithme est dans NPTIME.

Dans ce chapitre nous allons parler de l'extension des résultats concernant l'infinité de la norme à une sous classe des biTPV, les transducteurs à pile visibles linéaires. Nous allons présenter un nouveau modèle de transducteur appelé *mot-vers-contexte* et basé sur les automates finis, où les sorties sont des contextes de mots, et montrerons qu'il est équivalent aux biTPV linéaires. Sur ce modèle nous présenterons deux propriétés étendant celles présentées par Andreas Weber pour les automates finis et par Helmut Seidl pour les automates d'arbres et montrerons que ce sont des conditions suffisantes à l'infinité de la norme d'un transducteur

mot-vers-contexte. Nous conjecturons que ces propriétés constituent également une condition nécessaire et discuterons des différentes approches possibles pour le démontrer.

Plan du chapitre Nous allons présenter les travaux d' Andreas Weber sur les transducteurs de mots à la Section 5.1, et ceux de Helmut Seidl sur les transducteurs d'arbres à la Section 5.2. Nous présenterons ensuite les résultats de Jacques Sakarovitch et Rodrigo de Souza à la Section 5.3. Dans la Section 5.4, nous introduirons le nouveau modèle de transducteurs. Enfin nous présenterons nos résultats sur les TPV linéaires émondés à la Section 5.5.

5.1 Caractérisation de la norme infinie d'un transducteur fini

En 1989, Andreas Weber a prouvé dans [Web90] qu'il est possible de décider en temps polynomial si la norme d'un TF est infinie ou non. Pour cela, il présente deux propriétés appelées **(W1)** et **(W2)**. Il montre que la présence de l'une de ces propriétés est nécessaire et suffisante à l'infinité de la norme TF, et il est possible de décider en temps polynomial si le TF les satisfait.

5.1.1 Une caractérisation par les critères (W1) et (W2)

Nous présentons **(W1)** et **(W2)**. Soit $T = (A, \lambda)$ un TF émondé sur l'alphabet Σ.

Propriété (W1) T a la propriété **(W1)** s'il existe un mot $u \in \Sigma^*$ tel que :

$p \xrightarrow{u,u_1}{}^* p \in \mathsf{Cal}_u(A),\ p \xrightarrow{u,u_2}{}^* q \in \mathsf{Cal}_u(A),\ q \xrightarrow{u,u_3}{}^* q \in \mathsf{Cal}_u(A)$

et $u_1u_2 \neq u_2u_3$.

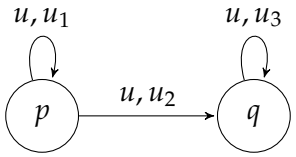


FIGURE 5.1 – La propriété **(W1)**.

La propriété **(W1)** est représentée à la Figure 5.1. Si T satisfait cette propriété il est possible de faire varier la position de u_2 dans la sortie en modifiant le nombre de u_1 et de u_3 produits. Comme $u_1u_2 \neq u_2u_3$, nous obtenons ainsi un nombre non borné de mots différents sur la même entrée.

Propriété (W2) T a la propriété **(W2)** s'il existe des mots $u, v, w \in \Sigma^*$ tels que :

$p \xrightarrow{u,u_1}{}^* p_1 \in \mathsf{Cal}_u(A),\ p_1 \xrightarrow{v,v_1}{}^* p_1 \in \mathsf{Cal}_v(A),\ p_1 \xrightarrow{w,w_1}{}^* p \in \mathsf{Cal}_w(A),$

$p \xrightarrow{u,u_2}{}^* p_2 \in \mathsf{Cal}_u(A),\ p_2 \xrightarrow{v,v_2}{}^* p_2 \in \mathsf{Cal}_v(A),\ p_2 \xrightarrow{w,w_2}{}^* q \in \mathsf{Cal}_w(A),$

$q \xrightarrow{u,u_3}{}^* p_3 \in \mathsf{Cal}_u(A),\ p_3 \xrightarrow{v,v_3}{}^* p_3 \in \mathsf{Cal}_v(A),\ p_3 \xrightarrow{w,w_3}{}^* q \in \mathsf{Cal}_w(A),$

et $|v_1| \neq |v_2|$.

La propriété **(W1)** est représentée à la Figure 5.2. Si T satisfait cette propriété alors il est possible de générer un nombre non borné de mots de différentes tailles sur la même entrée en modifiant combien de fois les mots v_1 et v_2 apparaissent dans la sortie. Dans les sous-sections suivantes nous allons montrer le résultat suivant :

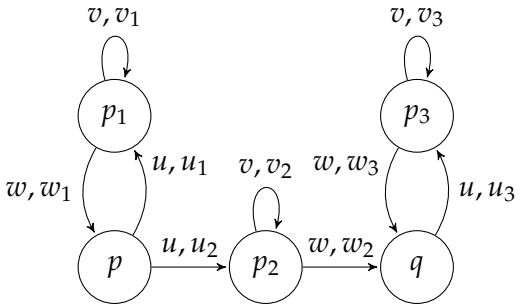


FIGURE 5.2 – La propriété **(W2)**.

Théorème 5.1.1: [Web90]

Soit T un TF émondé. La norme de T est infinie si et seulement si T satisfait l'une des propriétés **(W1)** ou **(W2)**.

5.1.2 Un transducteur satisfaisant (W1) ou (W2) possède une norme infinie

Le but de cette section est de montrer le premier sens du Théorème 5.1.1. Nous définissons la proposition suivante :

Proposition 5.1.2

Soit $T = (A, \lambda)$ un TF émondé. Si T satisfait l'une des propriétés **(W1)** ou **(W2)** alors la norme de T est infinie.

Montrer que la présence de l'une de ces propriétés implique une norme infinie n'est pas difficile. Tout d'abord, comme T est émondé il existe deux calculs $p_i \xrightarrow{u_i}{}^* p \in \mathsf{Cal}_{u_i}(A)$ et $q \xrightarrow{u_i}{}^* q_f \in \mathsf{Cal}_{u_f}(A)$ avec p_i un état initial et q_f un état final. Dans le cas de **(W1)**, il suffit de considérer les sorties $y_{k,n} = u_1^{n-1} u_2 u_3^{k-n}$ produites par le mot u^k pour tout $k \in \mathbb{N}$ et $n \in [1, k]$. Si $|u_1 u_2| \neq |u_2 u_3|$ alors $|u_2| \neq |u_3|$. Il est donc possible de construire des calculs satisfaisants **(W2)** en utilisant les deux calculs $p \xrightarrow{u,u_1} p$ et $q \xrightarrow{u,u_1} q$ plusieurs fois, comme représenté à la figure 5.3.

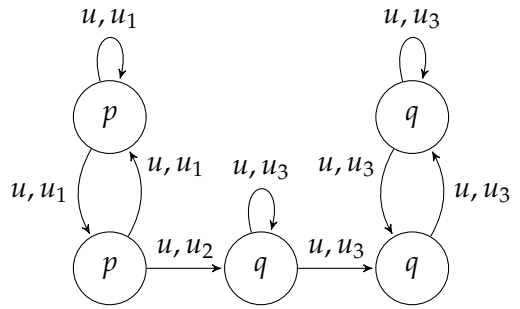


FIGURE 5.3 – Si T satisfait **(W1)** avec $|u_1 u_2| \neq |u_2 u_3|$ alors T satisfait **(W2)**.

Sinon $|u_1 u_2| = l u_2 u_3|$ et il existe donc une position $l \in \mathbb{N}$ telle que $u_1 u_2(l) \neq u_2 u_3(l)$, et ceci implique pour tout $k \in \mathbb{N}$ et $n, n' \in [1, k]$ avec $n \neq n'$ nous avons $y_{k,n} \neq y_{k,n'}$. Nous avons donc $\mathsf{nor}(u_i u^k u_f) \geqslant k$ et nous obtenons que $\mathsf{nor}(T) = +\infty$. Concernant la propriété **(W2)**, il

suffit de considérer les sorties $s_{k,n} = \Pi_{i=1}^{n-1}(u_1 v_1^{k-i} w_1) u_2 v_2^{k-n} w_2 \Pi_{i=n+1}^{k}(u_3 v_3^{k-i} w_3)$ produites par le mot $\Pi_{i=1}^{k}(uv^{k-i}w)$ pour tout $k \in \mathbb{N}$ et $n \in [1,k]$. Pour tout $n' \in [1,k]$ nous avons $|s_{k,n}| - |s_{k,n'}| = (k-n)(|v_2| - |v_1|) - (k-n')(|v_2| - |v_1|)$. Si n est différent de n' alors $|s_{k,n}| \neq |s_{k,n'}|$. Ceci implique que $\mathsf{nor}(u_i \Pi_{i=1}^{k}(uv^{k-i}w)u_f) \geqslant k$, et comme T est émondé nous obtenons que $\mathsf{nor}(T) = +\infty$.

5.1.3 Un transducteur avec une norme infinie satisfait (W1) ou (W2)

Ce sens de la preuve nécessite plus de travail, et nous allons présenter plusieurs résultats intermédiaires. Nous modifions la preuve originale de Andreas Weber afin de la rapprocher de celle de Helmut Seidl dans le cas des transducteurs d'arbres vers arbres. La plupart des arguments utilisés restent cependant les mêmes, seule la structure de la preuve change. L'idée est de décomposer les calculs du transducteur fini de manière finie, puis de montrer que chaque élément de cette décomposition ne peut produire qu'un nombre borné de sorties. Cette décomposition se base sur la notion de composante connexe, que nous présentons.

Définition 5.1.3

Soit $T = (A, \lambda)$ un TF avec $A = (Q, I, F, \delta)$. La composante connexe d'un état $q \in Q$ est l'ensemble des états Q_q tel que pour tout $p \in Q_q$, il existe deux calculs $q \xrightarrow{u}{}^* p \in \mathsf{Cal}_u(T)$ et $p \xrightarrow{v}{}^* q \in \mathsf{Cal}_v(T)$.

Nous considérons maintenant l'ensemble des calculs acceptants sur un mot d'entrée u, et pour chaque position l nous définissons $S_u(l)$ comme l'ensemble des états que peuvent atteindre les calculs à la position l. La décomposition regarde ensuite si l'ensemble $S_u(l)$ se répète à une position ultérieure. Si oui, nous supprimons les états utilisés par les calculs sur u entre ces deux positions. Nous répétons ce processus pour tout l de manière croissante, assurant ainsi que tout calcul possède une représentation finie dépendant uniquement du nombre d'états de l'automate. Formellement,

Définition 5.1.4

Soient $T = (A, \lambda)$ un TF sur l'alphabet Σ, $u = a_1 \dots a_m \in \Sigma^*$ un mot et une position $l \in [1, m]$. Nous appelons $S_u(l)$ l'ensemble d'états pouvant être atteints à la position l par un calcul acceptant sur u :

$$S_u(l) = \{p_l \mid p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} \cdots p_m \in \mathsf{ACal}_u(A)\}$$

Définition 5.1.5

Soient $T = (A, \lambda)$ un TF sur l'alphabet Σ et $u = a_1 \dots a_m \in \Sigma^*$ un mot et une position $l \in [1, m]$. Nous appelons $\mathsf{Next}_u(l)$ l'ensemble défini comme suit :

- $\mathsf{Next}_u(l) = l'$ si $l < l'$, $S_u(l) = S_u(l')$ et $S_u(l) \neq S_u(k)$ pour tout $k > l'$,
- $\mathsf{Next}_u(l) = l$ sinon.

Définition 5.1.6

Soient $T = (A, \lambda)$ un TF sur l'alphabet Σ et $u = a_1 \dots a_m \in \Sigma^*$ un mot. Nous appelons P_u le plus petit sous-ensemble de $[1, m]$ tel que :

- $1 \in P_u$
- $\forall l \in P_u, \mathsf{Next}_u(l) \in P_u$
- $\forall l \in P_u$, si $\mathsf{Next}_u(l) = l$ et $l < |u|$, alors $l + 1 \in P_u$

De plus nous considérons la partition suivante de P_u, $P_u = P_u^= \cup P_u^{\neq}$ avec :

- $P_u^= = \{l \in P_u \mid \mathsf{Next}_u(l) = l\}$
- $P_u^{\neq} = \{l \in P_u \mid \mathsf{Next}_u(l) \neq l\}$

Par définition de $P_u^=$ et de $P_u^{\neq}$, nous avons $\#P_u^= \leqslant 2^n$ et $\#P_u^{\neq} \leqslant 2^n$ avec n le nombre d'états de A. En effet, si $\#P_u^= > 2^n$ alors il existe $l, l' \in \mathbb{N}$ tels que $l \neq l'$, $S_u(l) = S_u(l')$ et $\mathsf{Next}_u(l) = l$, ce qui est en contradiction avec la définition de $\mathsf{Next}_u(l)$. De plus $\#P_u^{\neq} \leqslant 2^n$ car pour tout $l \in P_u^{\neq}$, si $\mathsf{Next}(l) \in P_u^=$ alors $\mathsf{Next}(\mathsf{Next}(l)) = \mathsf{Next}(l)$. Nous pouvons donc borner la taille de $P_u^{\neq}$ avec la taille de $P_u^=$.

Au Lemme 5.1.8 nous présentons formellement la décomposition des calculs de T en utilisant Next et les composantes connexes. Nous rappelons que le but de cette décomposition est d'obtenir une représentation finie d'un calcul, indépendamment de sa longueur. Avant toute chose nous avons besoin d'introduire une notation pour raisonner sur les sorties de T.

Définition 5.1.7

Soit $T = (A, \lambda)$ un TF sur l'alphabet Σ vers Σ' avec $A = (Q, I, F, \delta)$. Pour tout $p, q \in Q$ et $u \in \Sigma^*$ nous appelons $T_{p,q}(u)$ et $T_p(u)$ les ensembles et $\mathsf{nor}_{p,q}(u)$ et $\mathsf{nor}_p(u)$ les entiers définis comme suit :

$$\begin{array}{lcllcl} T_{p,q}(u) & = & \{v \in \Sigma'^* \mid p \xrightarrow{u,v}{}^* q \in \mathsf{Cal}_u(A)\} & \mathsf{nor}_{p,q}(u) & = & \#T_{p,q}(u) \\ T_p(u) & = & \bigcup_{p' \in Q}(T_{p,p'}(u)) & \mathsf{nor}_p(u) & = & \#T_p(u) \end{array}$$

Lemme 5.1.8

Soit $T = (A, \lambda)$ un TF sur l'alphabet Σ avec $A = (Q, I, F, \delta)$ et $u \in \Sigma^*$ un mot. Nous avons :

$$\begin{array}{lll} T_q(u) = & \bigcup\limits_{\substack{i \in P_u^{\neq}, \\ p \in Q_q,\, r \notin Q_q}} & T_{q,p}(u(1,i)) \,.\, T_{p,r}(u(i, \mathsf{Next}(i))) \,.\, T_r(u(\mathsf{Next}(i) \dots)) \\ \cup & \bigcup\limits_{\substack{i \in P_u^=, p \in Q_q, \\ \tau = (p,a,r) \in \delta \\ \text{avec } a = u[i] \text{ et } r \notin Q_q}} & T_{q,p}(u(1,i)) \,.\, \lambda(\tau) \,.\, T_r(u(i+1 \dots)) \\ \cup & \bigcup\limits_{p \in Q_q} & T_{q,p}(u) \end{array}$$

Pour chaque calcul $\rho : p_0 \xrightarrow{a_1,v_1} p_1 \xrightarrow{a_2,v_2} \cdots p_n \in \mathsf{Cal}_u(A)$, nous montrons maintenant pourquoi ρ est toujours compatible avec l'une des trois décompositions.

Nous considérons la plus grande position $l \in [1,n]$ telle que $p_l \in Q_{p_0}$. Par définition, pour tout $l' \in [l+1, |u|+1]$ nous avons $p'_l \notin Q_{p_0}$ et pour tout $l' \in [1,l]$ nous avons $p_{l'} \in Q_{p_0}$. Si $l = n$ alors ρ est compatible avec la troisième décomposition avec $p = p_l$. Sinon si $\mathsf{Next}(l) \neq l$ alors ρ est compatible avec la première décomposition avec $p = p_l$ et $r = p_{\mathsf{Next}(l)}$. Sinon ρ est compatible avec la deuxième décomposition avec $\tau = (p_l, a_l, p_{l+1})$, $p = p_l$ et $r = p_{l+1}$.

Nous sommes donc capable d'exprimer l'intégralité des calculs acceptants sur u via ces décompositions. Il suffit ensuite de borner la taille des ensembles $T_{q,p}$ et $T_{p,r}$, et d'appliquer une induction sur T_r afin de borner le nombre de sorties produites par u. Pour cela nous définissons deux lemmes. Le Lemme 5.1.9 nous permet de borner la taille de $T_{q,p}$, tandis que le Lemme 5.1.10 borne la taille de $T_{p,r}$.

Lemme 5.1.9

> Soit T un TF sur l'alphabet Σ ne satisfaisant pas **(W1)** avec $A = (Q, I, F, \delta)$. Pour tous états $p, q \in Q$ et mot $u \in \Sigma^*$ tels que $q \in Q_p$ nous avons $\mathsf{nor}_{p,q}(u) = 1$.

Si l'on suppose que $\mathsf{nor}_{p,q}(u)) > 1$ alors il existe deux calculs $p \xrightarrow{u,v_1}{}^* q, p \xrightarrow{u,v_2}{}^* q \in \mathsf{Cal}_u(A)$ avec $v_1 \neq v_2$. De plus, comme p et q sont dans la même composante connexe, nous pouvons en déduire qu'il existe un calcul $q \xrightarrow{w,v_3}{}^* p \in \mathsf{Cal}_w(A)$. Nous avons donc deux boucles distinctes sur l'état p, produisant les mots (v_1v_3) et (v_2v_3). Nous pouvons en conclure que T satisfait **(W1)** car $(v_1v_3)(v_1v_3) \neq (v_1v_3)(v_2v_3)$, ce qui est en contradiction avec l'hypothèse de départ concernant **(W1)**.

Pour le deuxième lemme nous avons besoin d'une propriété intermédiaire, appelée **(W1')**. Soit $T = (A, \lambda)$ un TF émondé sur l'alphabet Σ.

Propriété (W1') T a la propriété **(W1')** si il existe un mot $u \in \Sigma^*$ tel que :

$p \xrightarrow{u,u_1}{}^* p \in \mathsf{Cal}_u(A),\ p \xrightarrow{u,u_2}{}^* q \in \mathsf{Cal}_u(A),\ p \xrightarrow{u,u_2}{}^* q \in \mathsf{Cal}_u(A),\ q \xrightarrow{u,u_4}{}^* q \in \mathsf{Cal}_u(A)$

et $u_1 \neq \epsilon \wedge |u_2| = |u_3| \wedge u_2 \neq u_3$

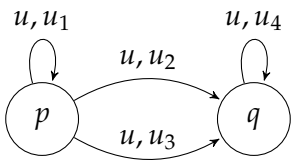


FIGURE 5.4 – La propriété **(W1')**

La propriété **(W1')** est représentée à la Figure 5.4. A noter que si T ne satisfait pas la propriété **(W1)**, alors il ne satisfait pas non plus la propriété **(W1')**. En effet si **(W1)** n'est pas satisfait pour tout $i \in \mathbb{N}$ nous avons $u_1^i u_2 = u_2 u_4^i$ et $u_1^i u_3 = u_3 u_4^i$. Si u_1 est non vide, ces égalités nous disent qu'il existe deux entiers $l, k \in \mathbb{N}$ et des mots $v, w \in \Sigma^*$ tels que v et w sont des préfixes de u_1, $u_2 = u_1^l v$ et $u_3 = u_1^k w$. Si $|u_2| = |u_3|$ alors nous obtenons que $u_2 = u_3$.

Lemme 5.1.10

Soit $T = (A, \lambda)$ un TF sur l'alphabet Σ ne satisfaisant pas **(W2)** et **(W1')** avec $A = (Q, I, F, \delta)$. Il existe un polynôme P indépendant de T tel que pour tout $u \in \Sigma^*$ et $B \subseteq Q$ avec

(1) $\forall p \in B, \exists q \in B, \exists p \xrightarrow{u|v} q \in \mathsf{Cal}_u(A)$,

(2) $\forall q \in B, \exists p \in B, \exists p \xrightarrow{u|v'} q \in \mathsf{Cal}_u(A)$,

nous avons : $\mathsf{nor}_{p,q}(u) \leqslant 2^{P(|A|)}$ pour tout $p, q \in B$.

Pour prouver ce résultat, il convient de noter que par définition de B pour tout $p, q \in B$ il existe des entiers $i, j \in \mathbb{N}$ et des états $p_1, p_2 \in B$ tels que :

- $\eta_1 = p_1 \xrightarrow{u^{i+j+1}|v_1} p_1$
- $\eta_2 = p_1 \xrightarrow{u^{i}|v_2} p$
- $\eta_2' = q \xrightarrow{u^{j}|v_3} p_2$
- $\eta_3 = p_2 \xrightarrow{u^{i+j+1}|v_4} p_2$

En effet, comme B est de taille finie il existe deux calculs :

$$q \xrightarrow{u,v_1'} q_1 \xrightarrow{u,v_2'} \dots \xrightarrow{u,v_n'} q_n \qquad q_m' \xrightarrow{u,v_m''} \dots \xrightarrow{u,v_2''} q_1' \xrightarrow{u,v_1''} p$$

avec $q_1, \dots q_n, q_1', \dots q_m' \in B$ et il existe deux positions $l \in [1, n]$ et $k \in [1, m]$ telles que $q_n = q_l$ et $q_m' = q_k'$. En choisissant une bonne itération des boucles, nous obtenons les calculs η_1, η_2, η_2' et η_3. Nous notons $u = a_1 \dots a_n$ avec $n \in \mathbb{N}$. Soient $\rho_i = p_{i,0} \xrightarrow{a_1} p_{i,1} \xrightarrow{a_2} \cdots p_{i,n} \in \mathsf{Cal}_u(A)$ des calculs pour tout $i \in [1, 3]$. Le *noyau* de $\pi = (\rho_1, \rho_2, \rho_3)$, dénoté par $\mathsf{noyau}(\pi)$ est le triplet de calculs $(\rho_1', \rho_2', \rho_3')$ défini comme suit :

- Si pour tout $k \in [0, n-1]$ et $l \in [k+1, n]$ nous avons $(p_{1,k}, p_{2,k}, p_{3,k}) \neq (p_{1,l}, p_{2,l}, p_{3,l})$ alors $\mathsf{noyau}(\pi) = \pi$.
- Sinon, il existe $k \in [0, n-1]$ et $l \in [k+1, n]$ telles que $(p_{1,k}, p_{2,k}, p_{3,k}) = (p_{1,l}, p_{2,l}, p_{3,l})$ et :
  - Il n'existe pas $k' \in [0, k-1]$ et $l' > k'$ avec $(p_{1,k'}, p_{2,k'}, p_{3,k'}) = (p_{1,l'}, p_{2,l'}, p_{3,l'})$,
  - il n'existe pas $l' > l$ avec $(p_{1,k}, p_{2,k}, p_{3,k}) = (p_{1,l'}, p_{2,l'}, p_{3,l'})$

  Nous appelons $\mathsf{noyau}(\pi)$ le triplet $\mathsf{noyau}(\pi')$ avec $\pi' = (\rho_1', \rho_2', \rho_3')$ et pour tout $i \in [1, 3]$:

$$\rho_i' = p_{i,0} \xrightarrow{a_1} p_{i,1} \xrightarrow{a_2} \cdots p_{i,k} \xrightarrow{a_{l+1}} p_{i,l+1} \xrightarrow{a_{l+2}} \cdots p_{i,n}$$

La fonction noyau supprime donc toutes les boucles synchronisées sur les calculs ρ_1, ρ_2 et ρ_3. Il est assez aisé de voir que le nombre de noyaux différents pouvant être construit par noyau est borné par $2^{|A|^6}$.

Nous considérons maintenant les calculs extraits de B, et supposons qu'il existe deux calculs $\hat{\eta} : p \xrightarrow{u|v} q, \hat{\eta}' : p \xrightarrow{u|v'} q \in \mathsf{Cal}_u(A)$. Nous allons montrer que si $\pi = (\eta_1, \eta_2 \hat{\eta} \eta_2', \eta_3)$ et $\pi' = (\eta_1, \eta_2 \hat{\eta}' \eta_2', \eta_3)$ ont le même noyau alors $v_2 v v_3 = v_2 v' v_3$, ceci impliquant que $v = v'$.

Nous montrons en premier que si π et π' ont le même noyau alors $|v| = |v'|$. Nous appelons etape l'application de la fonction de noyau sans récursion (ne supprimant qu'un seul triplet de boucles). Pour tout $i \in \mathbb{N}_{>0}$ nous appelons $\pi_i = (\bar{\eta}_{1,i}, \bar{\eta}_{2,i}, \bar{\eta}_{3,i})$ et $\pi'_i = (\bar{\eta}'_{1,i}, \bar{\eta}'_{2,i}, \bar{\eta}'_{3,i})$ les triplets produits respectivement par $\mathsf{etape}(\pi_{i-1})$ et par $\mathsf{etape}(\pi'_{i-1})$ avec $\pi_0 = \pi$ et $\pi'_0 = \pi'$. Nous appelons $k, l \in \mathbb{N}$ les entiers tels que $\pi_k = \pi'_l = \mathsf{noyau}(\pi) = \mathsf{noyau}(\pi')$. Comme T ne satisfait pas **(W1)** nous pouvons en déduire que pour tout $i \in \mathbb{N}_{>0}$ nous avons :

$$\begin{aligned} |\mathsf{sortie}(\bar{\eta}_{1,i})| - |\mathsf{sortie}(\bar{\eta}_{2,i})| &= |\mathsf{sortie}(\bar{\eta}_{1,i-1})| - |\mathsf{sortie}(\bar{\eta}_{2,i-1})| \\ |\mathsf{sortie}(\bar{\eta}'_{1,i})| - |\mathsf{sortie}(\bar{\eta}'_{2,i})| &= |\mathsf{sortie}(\bar{\eta}'_{1,i-1})| - |\mathsf{sortie}(\bar{\eta}'_{2,i-1})| \end{aligned}$$

Comme nous avons $\bar{\eta}_{1,0} = \bar{\eta}'_{1,0}$ et $\bar{\eta}_{2,k} = \bar{\eta}'_{2,l}$, nous concluons que $|v_2 v v_3| = |v_2 v' v_3|$ et donc $|v| = |v'|$. Nous distinguons maintenant deux cas :

- Si $v_1 \neq \epsilon$, alors comme T ne satisfait pas **(W1')**, $|v_2 v v_3| = |v_2 v' v_3|$ implique que $v_2 v v_3 = v_2 v' v_3$ et donc $v = v'$.
- Si $v_1 = \epsilon$, alors les mots produits sur les boucles supprimées de η_1 sont vides. Comme T ne satisfait pas **(W1)** nous en déduisons que pour tout $i \in \mathbb{N}_{>0}$ nous avons $\mathsf{sortie}(\bar{\eta}_{2,i}) = \mathsf{sortie}(\bar{\eta}_{2,i-1})$ et $\mathsf{sortie}(\bar{\eta}'_{2,i}) = \mathsf{sortie}(\bar{\eta}'_{2,i-1})$. Par conséquent, comme $\mathsf{sortie}(\pi_k) = \mathsf{sortie}(\pi'_l)$, $\mathsf{sortie}(\pi_0) = v_2 v v_3$ et $\mathsf{sortie}(\pi'_0) = v_2 v' v_3$ nous en déduisons que $v_2 v v_3 = v_2 v' v_3$ et donc $v = v'$.

Nous avons donc prouvé le résultat souhaité. En effet, comme le nombre de noyaux est borné par $2^{|A|^6}$, nous pouvons en déduire que le nombre de mots produits entre p et q sur u est aussi borné par $2^{|A|^6}$. Nous pouvons conclure que $\mathsf{nor}_{p,q}(u) \leqslant 2^{P(|A|)}$ pour tout $p, q \in B$.

Nous pouvons maintenant montrer si T ne satisfait pas **(W1)** et **(W2)** alors la norme de T est finie.

Proposition 5.1.11

> Soit $T = (A, \lambda)$ un TF sur l'alphabet Σ satisfaisant les propriétés **(W1)** et **(W2)** avec $A = (Q, I, F, \delta)$. Il existe un polynôme P' tel que $\mathsf{nor}(T) \leqslant 2^{P'(|T|)}$.

Soit T un transducteur fini avec un seul état initial q_{ini}. Nous notons par $v(n,k)$ le maximum de la norme d'un TF avec au plus n états et k le nombre de composantes connexes de T, c'est à dire $k = \#\{Q_q \mid q \in Q\}$.

Par le Lemme 5.1.9, nous avons montré que $v(n,1) \leqslant n$. Nous donnons maintenant une propriété récursive de $v(n,k)$. Soit $u \in \Sigma^*$ un mot. Par le Lemme 5.1.8 nous avons :

$$\begin{aligned} \mathsf{nor}_{q_{ini}}(u) \leqslant & \sum_{\substack{i \in P_u^{\neq}, \\ p \in Q_{q_{ini}},\, r \notin Q_{q_{ini}}}} \mathsf{nor}_{q_{ini},p}(u[1,i]) \,.\, \mathsf{nor}_{p,r}(u[i,\mathsf{Next}(i)]) \,.\, \mathsf{nor}_r(u[\mathsf{Next}(i),\ldots]) \\ + & \sum_{\substack{i \in P_u^{=}, p \in Q_{q_{ini}}, \\ \tau = (p,a,r) \in \delta \\ \text{avec } a = u[i] \text{ et } r \notin Q_{q_{ini}}}} \mathsf{nor}_{q_{ini},p}(u[1,i]) \,.\, 1 \,.\, \mathsf{nor}_r(u[i,\ldots]) \\ + & \sum_{p \in Q_{q_{ini}}} \mathsf{nor}_{q_{ini},p}(u) \end{aligned}$$

Dans tous les termes, comme $p \in Q_{p_{ini}}$, nous pouvons appliquer le Lemme 5.1.9 pour borner la taille de $\mathsf{nor}_{q_{ini},p}$. De plus dans le premier terme, comme $i \in P_u^{\neq}$, nous avons $S(i) = S(\mathsf{Next}(i))$ et nous pouvons donc appliquer le Lemme 5.1.10 avec $N = S(i)$ pour borner $\mathsf{nor}_{p,r}$ avec $p, r \in S(i)$. De plus dans les deux premiers termes r n'est pas dans $Q_{q_{ini}}$, et donc la norme depuis r

peut-être majorée par le terme $v(n, k-1)$. En conséquence, comme l'inégalité précédente est vrai pour tout mot u nous avons :

$$v(n,k) \leqslant \sum_{\substack{i \in P_u^{\neq}, \\ p \in Q_{q_{ini}},\, r \notin Q_{q_{ini}}}} (2^{P(|T|)} \cdot v(n,k-1)) + \sum_{\substack{i \in P_=, p \in Q_{q_{ini}} \\ (p,a,r) \in \delta \\ \text{avec } a = u[i] \text{ et } r \notin Q_{q_{ini}}}} (v(n,k-1)) + n$$

En utilisant les bornes sur $P_u^{\neq}$ and $P_u^{=}$, nous obtenons :

$$\begin{aligned} v(n,k) &\leqslant 2^n.n^2.2^{P(|T|)}.v(n,k-1) + 2^n.n^2.v(n,k-1) + n \\ &\leqslant v(n,k-1).2^n.n^2.(2^{P(|T|)} + 1) + n \\ &\leqslant v(n,k-1).2^{n+P(n)+2log_2(n)+1} + n \end{aligned}$$

Et donc en utilisant la borne sur $v(n,1)$ nous pouvons en déduire que :

$$v(n,n) \leqslant n.2^{(n-1).(n+P(n)+2log_2(n)+1)} + n$$

Comme il y a au plus n composantes connexes, et comme en plus il faut considérer la somme sur tous les différents états initiaux, nous obtenons finalement la borne suivante sur la norme de T :

$$\mathsf{nor}(T) \leqslant 2^{(n-1).(n+P(n)+2log_2(n)+1)+2.log_2(n)} + n$$

Nous avons donc montré qu'il existe un polynôme P' tel que $\mathsf{nor}(T) \leqslant 2^{P'(|T|)}$.

Nous avons donc montrer la propriété souhaitée, le Théorème 5.1.1 découlant directement de cette proposition et de la Proposition 5.1.2.

5.2 Caractérisation de la norme infinie d'un transducteur d'arbres

En 1994, Helmut Seidl étend les résultats d'Andreas Weber aux transducteurs d'arbres. Il présente une approche similaire, sous la forme de plusieurs propriétés assurant l'infinité de la norme infinie d'un AA. La principale différence vient d'une troisième propriété, absente de la caractérisation d'Andreas Weber. Cette propriété capture la complexité inhérente aux AA. En effet, un automate fini peut être vu comme un automate d'arbre où toutes les transitions ont une arité de 1. La nouvelle propriété permet donc de représenter le coté horizontal d'un AA en plus du coté vertical.

Nous présentons les propriétés **(F0)**, **(F1)** et **(F2)**. Soit $T = (A, \lambda)$ un AA émondé sur l'alphabet gradué (Σ, ar).

Propriété (F0) T a la propriété **(F0)** s'il existe un contexte exact $t \in \mathcal{T}_\Sigma(\{x_1\})$ tel que :

$p \xrightarrow{t,s_1}{}^*(p) \in \mathsf{Cal}_t(T),\ p \xrightarrow{t,s_2}{}^*(q) \in \mathsf{Cal}_t(T),\ q \xrightarrow{t,s_3}{}^*(q) \in \mathsf{Cal}_t(T)$

et $s_1 s_2 \neq s_2 s_3$.

Propriété (F1) T a la propriété **(F1)** s'il existe des contextes $t_1, t_2, t_3 \in \mathcal{T}_\Sigma(\{x_1\})$ tels que :

$p \xrightarrow{t_1,s_{11}}{}^*(p_1) \in \mathsf{Cal}_{t_1}(T),\ p_1 \xrightarrow{t_2,s_{12}}{}^*(p_1) \in \mathsf{Cal}_{t_2}(T),\ p_1 \xrightarrow{t_3,s_{13}}{}^*(p) \in \mathsf{Cal}_{t_3}(T),$
$p \xrightarrow{t_1,s_{21}}{}^*(p_2) \in \mathsf{Cal}_{t_1}(T),\ p_2 \xrightarrow{t_2,s_{22}}{}^*(p_2) \in \mathsf{Cal}_{t_2}(T),\ p_2 \xrightarrow{t_3,s_{23}}{}^*(q) \in \mathsf{Cal}_{t_3}(T),$
$q \xrightarrow{t_1,s_{31}}{}^*(p_3) \in \mathsf{Cal}_{t_1}(T),\ p_3 \xrightarrow{t_2,s_{32}}{}^*(p_3) \in \mathsf{Cal}_{t_2}(T),\ p_3 \xrightarrow{t_3,s_{33}}{}^*(q) \in \mathsf{Cal}_{t_3}(T),$

et $s_{11}s_{12}s_{13}s_{21}s_{23} \neq s_{21}s_{22}s_{23}s_{31}s_{33}$.

Propriété (F2) T a la propriété **(F2)** s'il existe des contextes exacts $t_1 \in \mathcal{T}_\Sigma(\{x_1, x_2\})$, $t_2 \in \mathcal{T}_\Sigma(\{x_1\})$ et $t_3 \in \mathcal{T}_\Sigma$ tels que :

$p \xrightarrow{t_1, s_{11}}{}^*(p, p_1) \in \mathsf{Cal}_{t_1}(T),\ p_1 \xrightarrow{t_2, s_{12}}{}^*(p_1) \in \mathsf{Cal}_{t_2}(T),\ p_1 \xrightarrow{t_3, s_{13}}{}^*() \in \mathsf{Cal}_{t_3}(T),$

$p \xrightarrow{t_1, s_{21}}{}^*(q, p_2) \in \mathsf{Cal}_{t_1}(T),\ p_2 \xrightarrow{t_2, s_{22}}{}^*(p_2) \in \mathsf{Cal}_{t_2}(T),\ p_2 \xrightarrow{t_3, s_{23}}{}^*() \in \mathsf{Cal}_{t_3}(T),$

$q \xrightarrow{t_1, s_{31}}{}^*(q, p_3) \in \mathsf{Cal}_{t_1}(T),\ p_3 \xrightarrow{t_2, s_{32}}{}^*(p_3) \in \mathsf{Cal}_{t_2}(T),\ p_3 \xrightarrow{t_3, s_{33}}{}^*() \in \mathsf{Cal}_{t_3}(T),$

et $s_{11}[s_{21}[x_1, s_{23}], s_{12}s_{13}] \neq s_{21}[s_{31}[x_1, s_{33}], s_{22}s_{23}]$.

Le résultat montré par Helmut Seidl est le suivant :

Théorème 5.2.1: [Sei94a]

> Soit T un AA émondé. La norme de T est infinie si et seulement si T satisfait l'une des propriétés **(F1)** ou **(F2)**.

La première chose à constater est que **(F0)** n'est pas présent dans la caractérisation. En effet, la propriété **(F1)** est suffisamment générale pour englober **(F0)**. La définition de **(F1)** est différente de **(W2)** pour cette raison mais pas seulement. En effet, contrairement aux transducteurs finis où **(W1)** capture tous les cas où il est possible de générer un nombre non borné de mots de même taille mais différents, dans le cas des transducteurs d'arbres la propriété **(F0)** est incomplète est n'englobe pas entièrement toute cette complexité. Enfin, la propriété **(F2)** gère le coté horizontal des AA non présent dans les TF.

La preuve de ce résultat suit les grandes lignes de la preuve du Théorème 5.2.1, la seule grosse différence étant l'intégration de la propriété **(F2)** dans le cheminement, utilisée pour montrer l'équivalent du Lemme 5.1.10 pour les transducteurs d'arbres. Bien évidemment la preuve des différents arguments, même s'ils découlent directement de ceux utilisés dans le cas des transducteurs finis, est bien plus complexe et nécessite un lourd travail de combinatoire des arbres que nous ne présenterons pas ici, et nous renvoyons le lecteur à l'article de référence.

5.3 Nouvelle caractérisation de la norme infinie d'un transducteur fini

Nous avons présenté une caractérisation de la norme infinie d'un transducteur fini par le biais de deux propriétés appelées **(W1)** et **(W2)**. Dans cette section, nous allons parler d'une approche plus récente démontrant le même résultat en utilisant des délais. Intuitivement, si nous considérons les sorties de deux calculs ρ_1, ρ_2 sur le même mot d'entrée, le *délai* entre ρ_1 et ρ_2 est le mot que doit produire ρ_1 pour rattraper $\mathsf{sortie}(\rho_2)$. Pour que cela soit possible $\mathsf{sortie}(\rho_1)$ doit être un préfixe de $\mathsf{sortie}(\rho_2)$, sinon les deux mots sont incompatibles et il devient alors impossible de rattraper le délai. Cette notion est très intéressante car elle permet de représenter finement le comportement d'un transducteur. En stockant des délais dans l'état, il devient alors possible de retenir quel est le nombre de sorties différentes produites jusque là.

5.3.1 Définition de délai

Soit Σ un alphabet. Nous appelons H_Σ l'ensemble de paires de mots de Σ^* où l'un des deux est vide et auquel l'élément $\mathbf{0}$ est ajouté :

$$H_\Sigma = (\Sigma^* \times \{\epsilon\}) \cup (\{\epsilon\} \times \Sigma^*) \cup \mathbf{0}$$

Étant donnés u, v deux mots de Σ^*, le *délai* entre u et v est l'élément de H_Σ dénoté par $\mathsf{delai}(u, v)$ et défini par :

$$\mathsf{delai}(u,v) = \begin{cases} (w, \epsilon) & \text{si } u = vw \text{ avec } w \in \Sigma^*, \\ (\epsilon, w) & \text{si } v = uw \text{ avec } w \in \Sigma^*, \\ \mathbf{0} & \text{sinon.} \end{cases}$$

Intuitivement, $\mathsf{delai}(u, v)$ nous dit à quel point u est en avance ou en retard par rapport à v. Si $\mathsf{delai}(u, v) = \mathbf{0}$ alors u et v sont incompatibles. De plus si $\mathsf{delai}(u, v) = (\epsilon, \epsilon)$ alors $u = v$. La taille d'un délai $d \neq \mathbf{0}$, dénotée par $|d|$, est définie par :

$$|d| = \begin{cases} |w| & \text{si } d = (w, \epsilon) \text{ ou } d = (\epsilon, w) \text{ avec } w \in \Sigma^*, \\ 0 & \text{sinon.} \end{cases}$$

Nous définissons maintenant une fonction de *mise à jour* de délai. Étant donnés un délai $d \in H_\Sigma$ et deux mots $u, v \in \Sigma^*$ nous définissons la fonction $\mathsf{maj}(d, u, v)$ de la manière suivante :

$$\mathsf{maj}(d,u,v) = \begin{cases} \mathsf{delai}(u'u, v'v) & \text{si } d = (u', v') \text{ avec } u', v' \in \Sigma^*, \\ \mathbf{0} & \text{sinon.} \end{cases}$$

Supposons qu'il existe deux mots $u'', v'' \in \Sigma^*$ tels que $\mathsf{delai}(u'', v'') = d$. La fonction maj permet de calculer $\mathsf{delai}(u''u, v''v)$ en ne connaissant que le délai entre u'' et v''. Il est donc facile de montrer que nous avons $\mathsf{maj}(d, u, v) = \mathsf{delai}(u''u, v''v)$ et si $u = v = \epsilon$ alors $\mathsf{maj}(d, u, v) = d$.

Soit $T = (A, \lambda)$ un TF sur l'alphabet Σ vers l'alphabet Σ' et deux calculs de A sur $a_1, \ldots, a_n \in \Sigma^*$ avec $n \in \mathbb{N}$:

$$\rho = p_0 \xrightarrow{a_1,v_1} p_1 \xrightarrow{a_2,v_2} \cdots p_n, \qquad \rho' = p'_0 \xrightarrow{a_1,v'_1} p'_1 \xrightarrow{a_2,v'_2} \cdots p'_n$$

le *délai* entre ρ et ρ' est l'élément de $H_{\Sigma'}$ dénoté par $\mathsf{delai}(\rho, \rho')$ et défini par :

$$\mathsf{delai}(\rho, \rho') = \mathsf{delai}(v_1 \ldots v_n, v'_1, \ldots v'_n)$$

Si $\mathsf{delai}(\rho, \rho') \neq \mathbf{0}$, alors nous définissons le *décalage* $\langle \rho, \rho' \rangle$ entre ρ et ρ' de la manière suivante :

$$\langle \rho, \rho' \rangle = \max(\{|\mathsf{delai}(\rho_l, \rho'_l)| \mid \rho_l = p_0 \xrightarrow{a_1,v_1} p_1 \xrightarrow{a_2,v_2} \cdots p_l, \rho'_l = p'_0 \xrightarrow{a_1,v'_1} p'_1 \xrightarrow{a_2,v'_2} \cdots p'_l \text{ et } l \in [1, n]\})$$

5.3.2 Une nouvelle caractérisation par les critères (C1) et (C2)

Nous présentons maintenant une caractérisation de la norme infinie d'un transducteur fini utilisant les délais définis dans la section précédente. Cette caractérisation est proposée par Rodrigo de Souza et Jacques Sakarovitch dans [SdS08].

Soit $T = (A, \lambda)$ un TF émondé sur l'alphabet Σ avec $A = (Q, I, F, \delta)$. Nous appelons L_T l'entier défini par $L_T = \max(\{|\lambda(\tau)| \mid \tau \in \delta\})$.

Propriété (C1) T a la propriété **(C1)** s'il existe un mot $u \in \Sigma^*$ tel que :

$\rho_1 : p \xrightarrow{u,u_1}{}^* p \in \mathsf{Cal}_u(A)$, $\rho_2 : p \xrightarrow{u,u_2}{}^* q \in \mathsf{Cal}_u(A)$, $\rho_3 : q \xrightarrow{u,u_3}{}^* q \in \mathsf{Cal}_u(A)$

et $u_1u_2 \neq u_2u_3$.

Propriété (C2) T a la propriété **(C2)** s'il existe un mot $u \in \Sigma^*$ tel que :

$\rho_1 : p \xrightarrow{u,u_1}{}^* p \in \mathsf{Cal}_u(A)$, $\rho_2 : p \xrightarrow{u,u_2}{}^* q \in \mathsf{Cal}_u(A)$, $\rho_3 : q \xrightarrow{u,u_3}{}^* q \in \mathsf{Cal}_u(A)$

et $u_1u_2 = u_2u_3 \wedge \langle \rho_1\rho_2, \rho_2\rho_3 \rangle > L_T \#Q^3$.

La propriété **(C1)** est la même que **(W1)**, et la propriété **(C2)** est nouvelle, et permet de cacher la complexité de **(W2)**. En effet l'inégalité $\langle \rho_1\rho_2, \rho_2\rho_3 \rangle > L_T \#Q^3$ implique que la taille de u dépasse $\#Q^3$, ceci permettant d'extraire des boucles synchronisées sur ρ_1, ρ_2 et ρ_3 produisant des mots de longueurs différentes (comme pour **(W2)**).

Théorème 5.3.1: [SdS08]

Soit T un TF émondé. La norme de T est infinie si et seulement si T satisfait l'une des propriétés **(C1)** ou **(C2)**.

5.3.3 Un transducteur satisfaisant (C1) ou (C2) possède une norme infinie

Proposition 5.3.2

Soit $T = (A, \lambda)$ un TF émondé. Si T satisfait l'une des propriétés **(C1)** ou **(C2)** alors la norme de T est infinie.

Pour prouver ce lemme nous montrons ce résultat intermédiaire.

Lemme 5.3.3

Soit $T = (A, \lambda)$ un transducteur fini. Si T ne satisfait pas les propriétés **(W1)** et **(W2)** alors il ne satisfait pas non plus les propriétés **(C1)** et **(C2)**.

Le fait que **(C1)** ne soit pas satisfait est trivial. Si nous supposons que **(C2)** est satisfait alors il existe des calculs $\rho_1 : p \xrightarrow{u,u_1}{}^* p, \rho_2 : p \xrightarrow{u,u_2}{}^* q$ et $\rho_3 : q \xrightarrow{u,u_3}{}^* q \in \mathsf{Cal}_u(A)$ que nous choisissons de tailles minimales tels que $u_1u_2 = u_2u_3$ et $\langle \rho_1\rho_2, \rho_2\rho_3 \rangle > L_T \#Q^3$. Cette hypothèse assurant que $|u| > \#Q^3$ nous permet d'extraire des trois calculs des boucles synchronisées. Plus précisément, il existe des calculs $\rho_{11}, \rho_{12}, \rho_{13}, \rho_{21}, \rho_{22}, \rho_{23}, \rho_{31}, \rho_{32}, \rho_{33}$ de A définis comme suit :

$$\begin{array}{c}
\rho_{11} : p \xrightarrow{u',u'_1}{}^* p_1 \in \mathsf{Cal}_{u'}(A),\ \rho_{12} : p_1 \xrightarrow{v',v'_1}{}^* p_1 \in \mathsf{Cal}_{v'}(A),\ \rho_{13} : p_1 \xrightarrow{w',w'_1}{}^* p \in \mathsf{Cal}_{w'}(A), \\
\rho_{21} : p \xrightarrow{u',u'_2}{}^* p_2 \in \mathsf{Cal}_{u'}(A),\ \rho_{22} : p_2 \xrightarrow{v',v'_2}{}^* p_2 \in \mathsf{Cal}_{v'}(A),\ \rho_{23} : p_2 \xrightarrow{w',w'_2}{}^* q \in \mathsf{Cal}_{w'}(A), \\
\rho_{31} : q \xrightarrow{u',u'_3}{}^* p_3 \in \mathsf{Cal}_{u'}(A),\ \rho_{32} : p_3 \xrightarrow{v',v'_3}{}^* p_3 \in \mathsf{Cal}_{v'}(A),\ \rho_{33} : p_3 \xrightarrow{w',w'_3}{}^* q \in \mathsf{Cal}_{w'}(A)
\end{array}$$

tels que $\rho_1 = \rho_{11}\rho_{12}\rho_{13}$, $\rho_2 = \rho_{21}\rho_{22}\rho_{23}$ et $\rho_3 = \rho_{31}\rho_{32}\rho_{33}$. Pour tout $l \in \mathbb{N}$ nous considérons les calculs suivants :

$$\rho_1^l = \rho_{11}\rho_{12}^l\rho_{13} \qquad \rho_2^l = \rho_{21}\rho_{22}^l\rho_{23} \qquad \rho_3^l = \rho_{31}\rho_{32}^l\rho_{33}$$

Comme T ne satisfait pas **(W1)** nous savons que $\mathsf{sortie}(\rho_1^l\rho_2^l) = \mathsf{sortie}(\rho_2^l\rho_3^l)$. De plus T ne satisfait pas **(W2)** et cela implique que $|v'_1| = |v'_2|$. Nous en déduisons que $\mathsf{delai}(\rho_{11}\rho_{12}, \rho_{21}\rho_{22}) = \mathsf{delai}(\rho_{11}, \rho_{21})$. Nous pouvons conclure que $|\mathsf{delai}(\rho_{11}\rho_{13}, \rho_{21}\rho_{23})| > L_T \#Q^3$ et donc :

$$\langle \rho_{11}\rho_{13}\rho_{21}\rho_{23}, \rho_{21}\rho_{23}\rho_{31}\rho_{33} \rangle > L_T \#Q^3$$

Ce qui est une contradiction avec l'hypothèse de départ sur la minimalité de la longueur des calculs ρ_1, ρ_2 et ρ_3.

La preuve du Lemme 5.3.2 suit directement ce résultat et la Proposition 5.1.2. En effet si T satisfait l'une des propriétés **(C1)** ou **(C2)** alors par le Lemme 5.3.3 T satisfait aussi l'une des propriétés **(W1)** ou **(W2)**. Par la Proposition 5.1.2 nous pouvons donc conclure que la norme de T est infinie.

5.3.4 Un transducteur avec une norme infinie satisfait (C1) ou (C2)

Proposition 5.3.4

> Soit $T = (A, \lambda)$ un TF émondé. Si la norme de T est infinie alors T satisfait l'une des propriétés **(C1)** ou **(C2)**.

Pour prouver cette proposition nous allons utiliser des outils totalement différents de la preuve originale, que nous définissons maintenant.

Soit $T = (A, \lambda)$ un transducteur fini avec $A = (Q, I, F, \delta)$. Nous définissons trois ensembles D^T_{C1}, D^T_{C2} et D^T de paires d'états de T de la manière suivante. Pour toute paire $(p, q) \in Q \times Q$ telle que $p \neq q$ et $\rho_1 : p \xrightarrow{u,u_1}{}^* p, \rho_2 : p \xrightarrow{u,u_2}{}^* q, \rho_3 : q \xrightarrow{u,u_3}{}^* q \in \mathsf{Cal}_u(T)$ nous avons :

$$
\begin{aligned}
(p, q) \in D^T_{C1} &\Leftrightarrow u_1 u_2 \neq u_2 u_3, \\
(p, q) \in D^T_{C2} &\Leftrightarrow u_1 u_2 = u_2 u_3 \wedge \langle \rho_1\rho_2, \rho_2\rho_3 \rangle > L_T \# Q^3, \\
(p, q) \in D^T &\Leftrightarrow u_1 u_2 = u_2 u_3 \wedge \langle \rho_1\rho_2, \rho_2\rho_3 \rangle \leqslant L_T \# Q^3,
\end{aligned}
$$

Si T ne satisfait pas **(C1)** ou **(C2)** alors $D^T_{C1} = D^T_{C2} = \varnothing$. Supposons que D^T soit vide aussi. Nous considérons le $\mathcal{N}$-AF $S = (A, \lambda')$ avec $\lambda'(\tau) = 1$ pour tout $\tau \in \delta$. Nous avons présenté à a page 4.1.1 une propriété caractérisant les $\mathcal{N}$-AF de coût infini. Comme D^T_{C1}, D^T_{C2} et D^T sont vides, nous en déduisons que S ne satisfait pas cette propriété, car il est impossible de trouver les calculs requis. Par conséquent, le coût de S est fini. Notez que pour tout $\rho \in \mathsf{ACal}(S)$ nous avons $\mathsf{cout}(\rho) = 1$, et donc $\mathsf{cout}(S) = \mathsf{amb}(A)$. De $\mathsf{nor}(T) \leqslant \mathsf{amb}(A)$ nous pouvons donc en conclure que la norme de T est finie aussi.

Soit $T = (A, \lambda)$ un TF de Σ vers Σ' avec $A = (Q, I, F, \delta)$. Nous allons présenter une construction produisant un TF T' à partir de T tel que $D^{T'}$ est vide, $\mathsf{trans}(T) = \mathsf{trans}(T')$ et si T ne satisfait pas **(C1)** et **(C2)** alors T' ne satisfait pas **(C1)** et **(C2)**. Le transducteur T' va stocker dans ses états des ensembles de délais. Ces délais seront utilisés pour garder une trace des sorties produites par les autres calculs sur le même mot d'entrée. Afin de garde un nombre d'états finis, nous présentons maintenant deux notions. La première définit un ordre sur les calculs, et nous permet de comparer uniquement un calcul avec ceux plus petits que lui. La deuxième permet de borner la taille d'un délai. En effet pour obtenir $D^{T'} = \varnothing$ il nous suffit de regarder les délais de taille inférieure ou égale à $L_T \# Q^3$.

Ordre sur les calculs Nous supposons qu'il existe un ordre $<_\delta$ sur δ, où deux transitions sont comparables si et seulement si elles ont la même lettre d'entrée. Soient $\rho = p_0 \xrightarrow{a_1,v_1} p_1 \xrightarrow{a_2,v_2} \cdots p_n$, $\rho' = p'_0 \xrightarrow{a_1,v'_1} p'_1 \xrightarrow{a_2,v'_2} \cdots p'_n$ deux calculs de T sur $a_1, \ldots, a_n \in \Sigma^*$ avec $n \in \mathbb{N}$. Nous écrivons $\rho <_T \rho'$ si et seulement s'il existe $i \in [1, n]$ tel que $(p_{i-1}, a_i, p_i) <_\delta (p'_{i-1}, a_i, p'_i)$ et $(p_{i'-1}, a_i, p_{i'}) = (p'_{i'-1}, a_i, p'_{i'})$ pour tout $i' \in [1, i-1]$. A noter que par la définition de $<_\delta$ deux calculs ne sont comparables que s'ils ont la même entrée, et par conséquent l'ordre défini par $<_T$ est partiel.

Délais entre transitions comparables Pour toute transition $\tau = (p, a, q) \in \delta$ et tout état $q' \in Q$ nous définissons l'ensemble $K^{q'}_\tau$ de délais de la manière suivante :

$$
K^{q'}_\tau = \{\mathsf{delai}(\lambda(\tau), \lambda(\tau')) \mid \text{ il existe } p' \in Q \text{ tel que } \tau' = (p', a, q') \in \delta, \tau' <_\delta \tau\}
$$

Cet ensemble représente les délais entre la sortie de τ et les sorties produites par les transitions plus petites que τ.

Délais bornés Soient $N \in \mathbb{N}_{>0}$ une constante entière et ω un symbole tel que $\omega \notin H_{\Sigma'}$. Nous appelons $H^{\omega}_{\Sigma'}$ l'ensemble $H_{\Sigma'} \cup \{\omega\}$. Pour tout délai d de $H^{\omega}_{\Sigma'}$, nous définissons la fonction $\mathsf{borne}_N : H^{\omega}_{\Sigma'} \to H^{\omega}_{\Sigma'}$ comme suit :

$$\mathsf{borne}_N(d) = \begin{cases} d & \text{si } d = \mathbf{0} \text{ ou } |d| \leqslant N, \\ \omega & \text{si } d = \omega \text{ ou } |d| > N. \end{cases}$$ Nous étendons de manière naturelle la fonction borne_N aux sous-ensembles de $2^{H^{\omega}_{\Sigma'}}$.

Nous définissons maintenant le transducteur fini $T' = \mathsf{separe}_N(T) = (A', \lambda')$ de Σ vers Σ' de la manière suivante :

- $A' = (Q', I', F', \delta')$,
- $Q' = Q \times 2^{H^{\omega}_{\Sigma'}{}^{Q}}$,
- $I' = \{(p, D) \in I \times 2^{H^{\omega}_{\Sigma'}} \mid \text{pour tout q } \in Q, D_q = \{(\epsilon, \epsilon)\}\}$,
- $F' = \{(p, D) \in F \times 2^{H^{\omega}_{\Sigma'}} \mid \text{pour tout q } \in Q, (\epsilon, \epsilon) \notin D_q\}$,
- Pour tout $\tau = (p, a, q) \in \delta$ et $D \in 2^{H^{\omega}_{\Sigma'}{}^{Q}}$ nous avons $(p, D) \xrightarrow{a,v} (q, D') \in \delta'$ avec $v = \lambda(\tau)$ et pour tout $q' \in Q$:

$$D'_{q'} = \mathsf{borne}_N(K^q_\tau \cup \bigcup_{p' \in Q,\ d \in D_{p'}} \mathsf{maj}(d, v, \lambda(\tau'))) \qquad \text{avec } \tau' = (p', a, q')$$

Cette construction stocke dans l'état un vecteur d'ensemble de délais. Pour tout $q \in Q$ la composante D_q contient tous les délais du calcul courant avec tous les autres calculs plus petits, sur la même entrée et finissant dans l'état q. Il est très important de retenir dans quels états les délais amènent afin de pouvoir les mettre à jour correctement. Intuitivement l'utilisation de K^q_τ dans la mise à jour permet d'ajouter les délais avec les calculs se différenciant uniquement sur la dernière transition. La deuxième partie de l'union est plus complexe : Étant donné un état $p' \in Q$, elle regarde qu'elles sont les transitions utilisables à partir de l'état p', et met à jour les délais stockés dans D à partir des sorties de ces transitions et de la sortie de τ. Les propriétés assurées par cette construction sont les suivantes :

Lemme 5.3.5

> Soit $T = (A, \lambda)$ un TF et $N \in \mathbb{N}_{>0}$ un entier. Pour tout calcul $\rho = (p_0, D_0) \xrightarrow{a_1,v_1} (p_1, D_1) \xrightarrow{a_2,v_2} \cdots (p_n, D_n) \in \mathsf{Cal}_u(\mathsf{separe}_N(A))$ avec un mot $u = a_1 \dots a_n \in \Sigma^*$ et $n \in \mathbb{N}$ il existe un calcul $\rho' = p_0 \xrightarrow{a_1,v_1} p_1 \xrightarrow{a_2,v_2} \cdots p_n \in \mathsf{Cal}_u(A)$

Lemme 5.3.6

> Soit T un TF et $N \in \mathbb{N}_{>0}$. Pour tout calculs ρ_1, ρ_2 de $\mathsf{separe}_N(T)$ sur la même entrée et la même sortie nous avons $\langle \rho_1, \rho_2 \rangle > N$.

Nous pouvons maintenant prouver la Proposition 5.3.4 en utilisant les Lemmes 5.3.5 et 5.3.6. Soit T un TF et $T' = \mathsf{separe}_N(T)$ avec $N = L_T \# Q^3$. Le Lemme 5.3.5 nous assure que $\mathsf{trans}(T) = \mathsf{trans}(T')$ et si T ne satisfait pas **(C1)** et **(C2)** alors T' ne satisfait pas **(C1)** et **(C2)**. Le Lemme 5.3.6 nous assure que $D^{T''}$ est vide. Nous pouvons donc appliquer le résultat précédemment évoqué sur T' et montrer que si T ne satisfait pas **(C1)** et **(C2)** alors la norme de T' est finie, et donc celle de T est finie aussi.

5.4 Transducteur de mot vers contexte

Cette section va se découper en deux parties. Dans la première, nous allons présenter le modèle des transducteurs mot vers contexte (ou M2C), puis dans la deuxième partie, nous montrerons qu'il existe une construction prenant en entrée un TPV linéaire T et produisant un M2C T' tel que la norme de T' est finie si et seulement si la norme de T est finie.

5.4.1 Définition

Contrairement aux transducteurs de mots où les sorties sont des mots, ici les sorties sont des contextes de mot, nous permettant ainsi de concaténer au milieu du mot et pas seulement à la fin comme pour les TF. Ce modèle permet d'exprimer plus de transformations. Par exemple, étant donné un mot $u = a_1 \dots a_n$, nous pourrions considérer son miroir, c'est à dire le mot $a_n \dots a_1$. Il existe un M2C permettant de transformer un mot en son miroir, mais ça n'est pas le cas pour les TF (voir dans [Sak09] par exemple).

Définition 5.4.1: M2C

> Un M2C T sur un alphabet Σ vers un alphabet Σ' est une paire $T = (A, \lambda)$ composée d'un AF $A = (Q, I, F, \delta)$ sur Σ et d'une fonction de sortie $\lambda : \delta \to \Sigma' \times \Sigma'$.

Pour la suite de la section nous considérons le M2C $T = (A, \lambda)$ sur l'alphabet Σ vers l'alphabet Σ'.

Sortie Nous définissons la sortie $\mathsf{sortie}(\rho)$ d'un calcul $\rho = (\tau_1, \tau_2, \dots, \tau_n) \in \delta^*$ de T sur $w = a_1, \dots, a_n \in \Sigma^*$ avec $n \in \mathbb{N}$ comme la concaténation des contextes sur les sorties des transitions de ρ :

$$\mathsf{sortie}(\rho) = (\epsilon, \epsilon)\lambda(\tau_1) \dots \lambda(\tau_n)$$

Nous pouvons écrire ρ de la manière suivante :

$$\rho = p_1 \xrightarrow{a_1, \lambda(\tau_1)} p_2 \xrightarrow{a_2, \lambda(\tau_2)} \dots \xrightarrow{a_{n-1}, \lambda(\tau_{n-1})} p_n \xrightarrow{a_n, \lambda(\tau_n)} p_{n+1}$$
$$\rho : p_1 \xrightarrow{a_1 \dots a_n, \lambda(\tau_1) \dots \lambda(\tau_n)}{}^* p_{n+1}$$

Sortie d'un transducteur La sortie $\mathsf{sortie}(w)$ de $w \in \mathcal{L}(A)$ est définie comme suit :

$$\mathsf{sortie}(w) = \bigcup_{\rho \in \mathsf{ACal}_w(T)} (\mathsf{sortie}(\rho))[\epsilon]$$

Autrement dit, $\mathsf{sortie}(w)$ est l'union des contextes fermés par ϵ produits par les calculs acceptants sur w :

Transformation La transformation définie par T et dénotée par $\mathsf{trans}(T)$ est définie par :

$$\mathsf{trans}(T) = \{(w, w') \in (\Sigma^* \times \Sigma'^*) \mid w \in \mathcal{L}(T) \text{ et } w' \in \mathsf{sortie}(w)\}$$

Autrement dit, la transformation de T est l'ensemble des paires de mots reconnus et produits par A.

Norme La norme de T, dénotée par $\mathsf{nor}(T)$, est définie par :

$$\mathsf{nor}(T) = \sup\{\#\mathsf{sortie}(w) \mid w \in \mathcal{L}(T)\}$$

Pour tout entier $k \in \mathbb{N}$ si $\mathsf{nor}(T) < k$ alors la norme de T est k-bornée. S'il existe un tel que $\mathsf{nor}(T) < k$ alors la norme de T est *finie*, sinon la norme de T est *infinie*.

Exemple 5.4.2

Nous avons représenté à la Figure 5.5 un M2C sur l'alphabet $\Sigma = \{a, b, c, d\}$ vers l'alphabet $\Sigma' = \{a, b\}$. La norme de T est infinie car pour tout $k \in \mathbb{N}_{>0}$ le mot $\Pi_{i=1}^{k}(ab^ic)d$ produit k mots $\Pi_{j=1}^{i}(a)bb\Pi_{j=i+1}^{k}(a)$ pour tout $i \in [1, k]$.

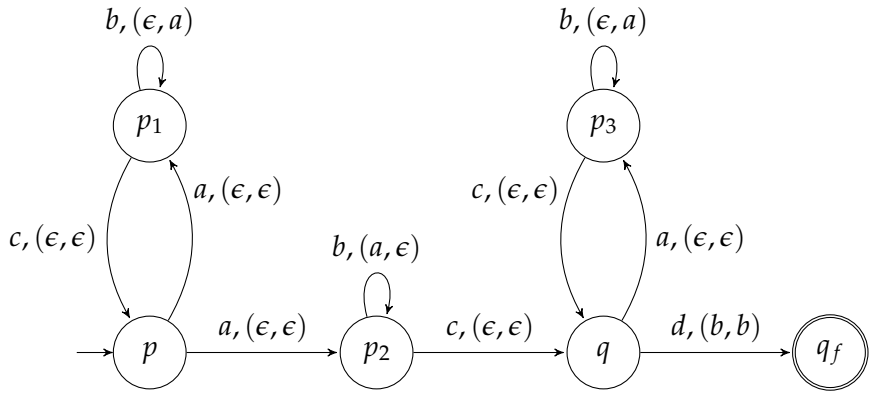


FIGURE 5.5 – Un transducteur mot-vers-contexte avec une norme infinie.

Comme pour les TF, les M2C peuvent être vus comme un automate fini à coûts sur un semi-anneau particulier. Par conséquent, les résultats d'émondage du chapitre 3 s'appliquent aussi ici. A noter que le langage produit par un M2C n'est pas régulier. Par exemple, il est possible de produire le langage a^nb^n, ce que ne peut pas faire un TF. Par contre si nous ne considérons que les contextes de sortie de la forme (w, ϵ) avec w un mot quelconque, alors nous obtenons la puissance des TF. Par conséquent, les M2C sont strictement plus expressifs que les TF.

5.4.2 Équivalence avec les automates à pile visible linéaires

Dans cette section nous considérons uniquement les APV linéaires. Nous allons présenter une construction qui, étant donné un APVlin A, produit un AF A' reconnaissant le langage produit par une fonction bijective que nous allons présenter. Dans une deuxième partie, nous étendrons ce résultat aux TPVlin et M2C, en expliquant comment construire les sorties sur les transitions du M2C construit.

Linéarisation de mots imbriqués Étant donné un alphabet structuré Δ, nous allons définir un alphabet particulier appelé $\tilde{\Delta}$ puis proposer une fonction construisant des mots sur l'alphabet $\tilde{\Delta}$ depuis des mots linéaires sur l'alphabet Δ. Nous construisons l'alphabet $\tilde{\Delta}$ de la manière suivante :

$$\tilde{\Delta} = \{(c, r) \mid c \in \Delta_c, r \in \Delta_r\} \cup \{(a, \top) \mid a \in \Delta_\iota\} \cup \{(\top, a) \mid a \in \Delta_\iota\}$$

avec $\top$ un symbole spécial distinct de Δ. Nous définissons les fonctions $\mathsf{tralin} : \Delta_{\mathsf{lin}}^* \to \tilde{\Delta}^*$ (*Transformation linéaire*), $\mathsf{trag} : \Delta_\iota^* \to \tilde{\Delta}^*$ (*Transformation gauche*) et $\mathsf{trad} : \Delta_\iota^* \to \tilde{\Delta}^*$ (*Transformation droite*) de la manière suivante :

$$\mathsf{tralin}(w) = \begin{cases} \mathsf{trag}(u)\mathsf{trad}(v)(c,r)\mathsf{tralin}(w') & \text{si } w = ucw'rv, c \in \Delta_c, r \in \Delta_r, w' \in \Delta_{\mathsf{lin}}^* \text{ et } u, v \in \Delta_\iota^*, \\ \mathsf{trag}(w) & \text{si } w \in \Delta_\iota^*. \end{cases}$$

$$\mathsf{trag}(w) = (a_1, \top)\dots(a_m, \top) \text{ avec } m \in \mathbb{N} \text{ et } w = a_1 \dots a_m \in \Delta_\iota^*,$$
$$\mathsf{trad}(w) = (\top, a_m)\dots(\top, a_1) \text{ avec } m \in \mathbb{N} \text{ et } w = a_1 \dots a_m \in \Delta_\iota^*.$$

Nous présentons la construction APV2AF. Intuitivement, elle produit à partir d'un APVlin A un AF A' tel que $\mathcal{L}(A') = \{\mathsf{tralin}(w) \mid w \in \mathcal{L}(A)\}$.

Définition 5.4.3

Soit $A = (Q, I, F, \delta, \Gamma)$ un APVlin sur l'alphabet structuré Δ. L' AF $\mathsf{APV2AF}(A) = (Q', I', F', \delta')$ sur l'alphabet $\tilde{\Delta}$ est défini par :

- $Q' = \mathsf{BI}_A \times \{0,1\} \times \Gamma$,
- $I' = (\mathsf{BI}_A \cap (I \times F)) \times \{0\} \times \{\bot\}$,
- $F' = \{(q, q, 0, \gamma) \mid (q,q) \in \mathsf{BI}_A \text{ et } \gamma \in \Gamma \cup \{\bot\}\}$,
- $$\delta' = \left\{ ((p,q,n,\gamma),(c,r),(p',q',0,\gamma')) \;\middle|\; \begin{array}{l}(p,c,\gamma',p') \in \delta_c, (q',\gamma',r,q) \in \delta_r, \\ n \in \{0,1\}, (p,q) \in \mathsf{BI}_A \text{ et } \gamma \in \Gamma \cup \{\bot\}\end{array} \right\}$$
$$\cup \left\{ ((p,q,0,\gamma),(a,\top),(p',q,0,\gamma)) \;\middle|\; \begin{array}{l}(p,a,p') \in \delta_\iota, \\ (p,q) \in \mathsf{BI}_A \text{ et } \gamma \in \Gamma \cup \{\bot\}\end{array} \right\}$$
$$\cup \left\{ ((p,q,n,\gamma),(\top,a),(p,q',1,\gamma)) \;\middle|\; \begin{array}{l}(q',a,q) \in \delta_\iota, n \in \{0,1\}, \\ (p,q) \in \mathsf{BI}_A \text{ et } \gamma \in \Gamma\end{array} \right\}$$

Soit $w = u_1c_1 \dots u_nc_nw'r_nv_n \dots r_1v_1 \in \Delta_{\mathsf{lin}}^*$. Le mot produit par $\mathsf{tralin}(w)$ est de la forme :

$$\mathsf{trag}(u_1)\mathsf{trad}(v_1)(c_1,r_1)\dots\mathsf{trag}(u_n)\mathsf{trad}(v_n)(c_n,r_n)\mathsf{trag}(w')$$

L'entier stocké permet de mémoriser où en est la lecture sur le mot. Nous sommes dans un état 0 avant de lire un mot produit par trad, et dans un état 1 après, (c,r) assurant le passage de 0 à 1. Cela nous assure que l'on ne puisse pas lire un mot u_i après avoir lu un mot v_i, pour tout $i \in [1,n]$. Enfin, nous stockons le symbole de pile dans l'état pour différencier les transitions du type $(p,c,\gamma,q),(p,c,\gamma',q) \in \delta_c$ avec $\gamma \neq \gamma'$. Nous allons montrer plusieurs propriétés de cette construction, le but final étant de lier les calculs de l'APV A avec ceux de l' AF $\mathsf{APV2AF}(A)$. Nous allons maintenant prouver plusieurs propriétés sur la construction, nous permettant de montrer le résultat suivant :

Théorème 5.4.4

Pour tout APVlin A nous avons $\mathcal{L}(\mathsf{APV2AF}(A)) = \{\mathsf{tralin}(w) \mid w \in \mathcal{L}(A)\}$ et pour tout $w \in \mathcal{L}(A)$ il existe une bijection entre $\mathsf{ACal}_w(A)$ et $\mathsf{ACal}_{\mathsf{tralin}(w)}(\mathsf{APV2AF}(A))$.

Exemple 5.4.5

Nous avons représenté dans la Figure 5.6 un APVlin A sur l'alphabet Δ vers l'alphabet $\tilde{\Delta}$ avec $\Delta_c = \{c\}$, $\Delta_r = \{r\}$, $\Delta_\iota = \{b\}$. Ce APV reconnaît les mots de la forme cwr avec $w \in \Delta^*_{\mathsf{lin}}$. Nous avons aussi représenté sur la Figure 5.6 le AF APV2AF(A) reconnaissant les mots de la forme tralin(cwr) avec $w \in \Delta^*_{\mathsf{lin}}$. Notez que nous avons supprimé les états ne faisant pas partis d'un calcul acceptant.

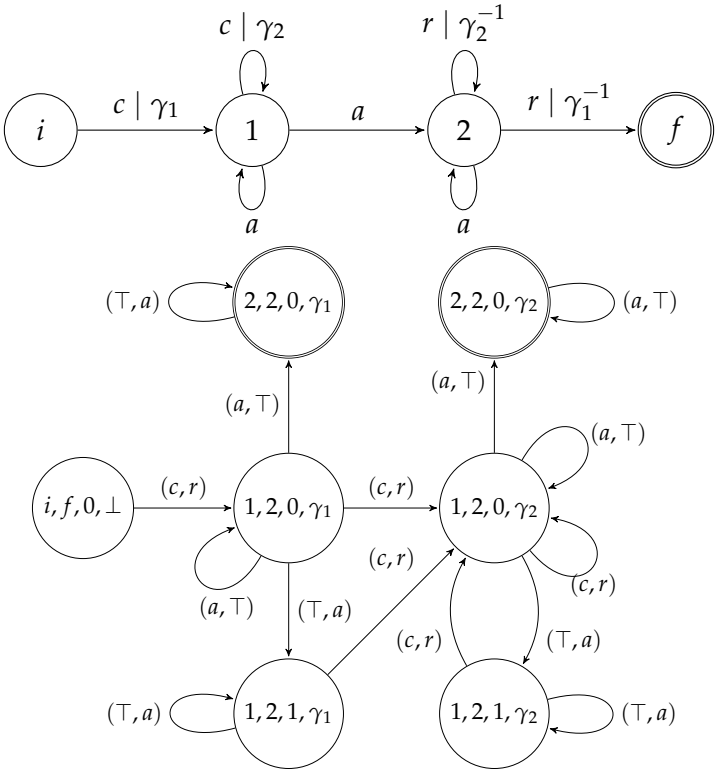


FIGURE 5.6 – $\mathcal{L}(A) = \{cwr \mid w \in \Delta^*_{\mathsf{lin}}\}$.

Pour prouver ce théorème nous montrons que le domaine du M2C construit est bien un sous-ensemble du co-domaine de la fonction tralin.

Lemme 5.4.6

Soit $A = (Q, I, F, \delta, \Gamma)$ un APVlin sur l'alphabet structuré Δ et $A' = (Q', I', F', \delta')$ l'AF APV2AF(A) sur l'alphabet $\tilde{\Delta}$. Pour tout mot $w \in \mathcal{L}(A')$, il existe $u \in \Delta^*_{\mathsf{lin}}$ tel que $w = \mathsf{tralin}(u)$.

Démonstration. Soit $(p, q, n, \gamma) \xrightarrow{w}{}^*(p', p', 0, \gamma')$ un calcul de A' sur $w \in \tilde{\Delta}^*$ avec $(p', p', 0, \gamma')$ un état final, Nous montrons par induction sur la taille de w qu'il existe $u \in \Delta^*_{\mathsf{lin}}$ tel que $w = \mathsf{tralin}(u)$. Si $w = \epsilon$ alors $w = \mathsf{tralin}(\epsilon)$. Si $w = (a, b)v$ avec $(a, b) \in \tilde{\Delta}$ alors il existe $\gamma'' \in \Gamma \cup \{\bot\}$, $p'', q'' \in Q$ tels que :

$$(p, q, n, \gamma) \xrightarrow{(a,b)} (p', q'', m, \gamma'') \xrightarrow{v}{}^* (p', p', 0, \gamma') \in \mathsf{Cal}_w(A)$$

Par hypothèse d'induction, il existe un mot $v' \in \Delta_{\mathsf{lin}}^*$ tel que $v = \mathsf{tralin}(v')$. Il existe donc $v'_1, v'_2 \in \Delta_\iota^*$, $v'_3 \in \Delta_{\mathsf{lin}}^*$ et $(c,r) \in \Delta_c \times \Delta_r$ tels que $v = \mathsf{trag}(v'_1)\mathsf{trad}(v'_2)(c,r)\mathsf{tralin}(v'_3)$ ou $v = \mathsf{trag}(v'_1)$. Nous considérons trois cas différents :

- Si $(a,b) \in \Delta_\iota \times \{\top\}$ alors $(a,b)v = \mathsf{trag}(av'_1)\mathsf{trad}(v'_2)(c,r)\mathsf{tralin}(v'_3)$ ou $(a,b)v = \mathsf{trag}(av'_1)$.
- Si $(a,b) \in \Delta_c \times \Delta_r$ alors $(a,b)v = \mathsf{trag}(\epsilon)\mathsf{trad}(\epsilon)(a,b)\mathsf{tralin}(uv)$.
- Si $(a,b) \in \{\top\} \times \Delta_\iota$ alors par construction de δ', nous avons $m = 1$. Comme le dernier état du calcul est égal à $(p', p', 0, \gamma')$ nous obtenons que $v \neq \epsilon$. Il existe $(a', b') \in \tilde{\Delta}$ tel que $v = (a', b')u'$. Par construction de δ', comme $m = 1$ nous en déduisons que $(a', b') \in \{\top\} \times \Delta_\iota$ ou $(a', b') \in \Delta_c \times \Delta_r$. Nous concluons que $(a,b)v = \mathsf{trag}(\epsilon)\mathsf{trad}(bv'_2)(c,r)\mathsf{tralin}(v'_3)$. □

Soit $A = (Q, I, F, \Gamma, \delta)$ un APVlin et A_{vf} l' AF APV2AF(A). Nous allons maintenant définir une fonction liant les calculs de A avec ceux de A_{vf}. Le but de cette fonction est d'établir comment les calculs de A_{vf} sont produits à partir de A. Soit $\rho \in \mathsf{Cal}(A_{\mathsf{vf}})$ un calcul sur $\mathsf{tralin}(u)$ avec $u \in \Delta_{\mathsf{lin}}^*$. Par définition de Δ_{lin}^*, soit $u = a_1 \dots a_m \in \Delta_\iota^*$, soit $u = wcu'rv$ avec $w = a_1 \dots a_m \in \Delta_\iota^*, v = b_1 \dots b_n \in \Delta_\iota^*$, $(c,r) \in \Delta_c \times \Delta_r$ et $u' \in \Delta_{\mathsf{lin}}^*$. Si $u = a_1 \dots a_m$ alors ρ se décompose ainsi :

$$\rho : (p_1, p_{m+1}, 0, \gamma) \xrightarrow{(a_1,\top)} (p_2, p_{m+1}, 0, \gamma) \xrightarrow{(a_m,\top)} \cdots (p_{m+1}, p_{m+1}, 0, \gamma) \in \mathsf{Cal}_u(A_{\mathsf{vf}})$$

Si $u = wcu'rv$ alors ρ se décompose par $\rho_1\rho_2 \xrightarrow{(c,r)} \rho_3$ avec :

$$\rho_1 : (p_1, q_1, 0, \gamma) \xrightarrow{(a_1,\top)} (p_2, q_1, 0, \gamma) \xrightarrow{(a_2,\top)} * \cdots (p_{n+1}, q_1, 0, \gamma) \in \mathsf{Cal}_w(A_{\mathsf{vf}})$$
$$\rho_2 : (p_{n+1}, q_1, 0, \gamma) \xrightarrow{(\top,b_1)} (p_{n+1}, q_2, 1, \gamma) \xrightarrow{(\top,b_2)} * \cdots (p_{n+1}, q_{m+1}, 1, \gamma) \in \mathsf{Cal}_v(A_{\mathsf{vf}})$$

Soient Φ_{vf} une fonction de $\mathsf{Cal}(A_{\mathsf{vf}}) \times \Gamma^*$ vers $\mathsf{Cal}(A)$, $\sigma \in \Gamma^*$ une pile. Nous définissons maintenant $\Phi_{\mathsf{vf}}(\rho, \sigma) \in \mathsf{Cal}(A)$ comme étant le calcul suivant :

$$\Phi_{\mathsf{vf}}(\rho,\sigma) = \begin{cases} (p_1,\sigma\gamma) \xrightarrow{a_1} (p_1,\sigma\gamma) \xrightarrow{a_m} \cdots (p_m,\sigma\gamma) & \text{si } u \in \Delta_\iota^* \\ (p_1,\sigma\gamma) \xrightarrow{a_1} \dots \xrightarrow{a_n} (p_{n+1},\sigma\gamma) \xrightarrow{c} \Phi_{\mathsf{vf}}(\rho_3,\sigma\gamma) \xrightarrow{r} (q_{m+1},\sigma\gamma) \xrightarrow{b_m} \cdots \xrightarrow{b_1} (q_1,\sigma\gamma) & \text{sinon} \end{cases}$$

Nous allons maintenant montrer plusieurs propriétés sur la fonction Φ_{vf}. Le Lemme 5.4.7 montre que la fonction construit bien un calcul de l' APVbi de départ. Nous montrons ensuite que cette fonction est surjective au Lemme 5.4.8 puis nous montrons qu'elle est injective au Lemme 5.4.9.

Lemme 5.4.7

Soit $A = (Q, I, F, \delta, \Gamma)$ un APVlin sur l'alphabet structuré Δ et $u \in \Delta_{\mathsf{lin}}^*$ un mot linéaire. Pour tout calcul $\rho : (p,q,0,\gamma) \xrightarrow{v} *(p',p',0,\gamma')$ de APV2AF(A) sur $v = \mathsf{tralin}(u)$ et pour toute pile $\sigma^* \in \Gamma$, $\Phi_{\mathsf{vf}}(\rho,\sigma)$ est un calcul de A sur u de la forme $(p,\sigma\gamma) \xrightarrow{u} *(q,\sigma\gamma)$.

Démonstration. Nous allons prouver ce lemme par induction sur la structure de v.

- Si $v = \mathsf{trag}(u)$ alors par définition de δ', ρ se décompose de la manière suivante :

  $$(p_1,q,0,\gamma) \xrightarrow{(a_1,\top)} (p_2,q,0,\gamma) \xrightarrow{(a_2,\top)} \cdots \xrightarrow{(a_n,\top)} (p_{n+1},q,0,\gamma)$$

  avec $p_1 = p$ et $p_{n+1} = q$. Nous avons donc $(p_i, a_i, p_{i+1}) \in \delta_\iota$ pour tout $i \in [1,n]$ et $\Phi_{\mathsf{vf}}(\rho,\sigma) = (p_1,\sigma\gamma) \xrightarrow{a_1} (p_2,\sigma\gamma) \xrightarrow{a_2} *(p_{n+1},\sigma\gamma)$ est bien un calcul de A sur u.

- S'il existe $u_1, u_2 \in \Delta_\iota^*$, $u_3 \in \Delta_{\mathsf{lin}}^*$ et $(c,r) \in \Delta_c \times \Delta_r$ tel que $v = \mathsf{trag}(u_1)\mathsf{trad}(u_2)(c,r)\mathsf{tralin}(u_3)$ alors par définition de δ', ρ se décompose par $\rho_1\rho_2 \xrightarrow{(c,r)} \rho_3$ avec :

$$\begin{aligned}\rho_1 : (p_1, q_1, 0, \gamma) &\xrightarrow{(a_1,\top)} (p_2, q_1, 0, \gamma) \xrightarrow{(a_2,\top)} \cdots (p_{n+1}, q_1, 0, \gamma)\\ \rho_2 : (p_{n+1}, q_1, 0, \gamma) &\xrightarrow{(\top,b_1)} (p_{n+1}, q_2, 1, \gamma) \xrightarrow{(\top,b_2)} \cdots (p_{n+1}, q_{m+1}, 1, \gamma)\\ \rho_3 : (p''q'', 0, \gamma'') &\xrightarrow{u_3}{}^* (p', p', 0, \gamma')\end{aligned}$$

Pour tout $i \in [1,n]$ et $j \in [1,m]$ nous avons $(p_i, a_i, p_{i+1}), (q_{j+1}, b_j, q_j) \in \delta_\iota$, $(p_{n+1}, c, \gamma'', p'') \in \delta_c$ et $(q'', \gamma'', r, q_{m+1}) \in \delta_r$. De plus par hypothèse d'induction, $\Phi_{\mathsf{vf}}(\rho_3, \sigma\gamma) = (p'', \sigma\gamma\gamma'') \xrightarrow{u_3}{}^* (q'', \sigma\gamma\gamma'')$ est un calcul de A. Donc $\Phi_{\mathsf{vf}}(\rho, \sigma) = (p, \sigma\gamma) \xrightarrow{u_1u_2(c,r)u_3}{}^* (q, \sigma\gamma)$ est un calcul de A. □

Lemme 5.4.8

> Soit A un APVlin sur l'alphabet structuré Δ et $u \in \Delta_{\mathsf{lin}}^*$ un mot linéaire. Pour tout calcul $\rho' : (p, \sigma\gamma) \xrightarrow{u}{}^* (q, \sigma\gamma)$ de A, il existe un calcul $\rho : (p, q, n, \gamma) \xrightarrow{\mathsf{tralin}(u)}{}^* (p', p', 0, \gamma')$ de $\mathsf{APV2AA}(A)$ tel que $\Phi_{\mathsf{vf}}(\rho, \sigma) = \rho'$.

Démonstration. Nous notons $A = (Q, I, F, \Gamma, \delta)$ et $\mathsf{APV2AA}(A) = A_{\mathsf{vf}} = (Q', I', F', \delta')$. Nous faisons la preuve par induction sur la structure linéaire de u.

- Si $u \in \Delta_\iota^*$ alors $\rho' = (p_1, \sigma\gamma) \xrightarrow{a_1} (p_2, \sigma\gamma) \xrightarrow{a_m} \cdots (p_{m+1}, \sigma\gamma)$ avec $u = a_1 \dots a_m$, $p = p_1$ et $q = p_{m+1}$. Par définition de δ' nous avons $((p_i, q_{m+1}, 0, \gamma), (a, \top), (p_{i+1}, q_{m+1}, 0, \gamma))$ pour tout $i \in [1,m]$, et donc en appliquant itérativement ϕ_{vf} nous obtenons que $\Phi_{\mathsf{vf}}(\rho, \sigma) = \rho'$ avec

$$\rho = (p_1, p_{m+1}, 0, \gamma) \xrightarrow{(a_1,\top)} (p_2, p_{m+1}, 0, \gamma) \xrightarrow{(a_2,\top)} \cdots \xrightarrow{(a_m,\top)} (p_{m+1}, p_{m+1}, 0, \gamma)$$

- si $u = vcu'rw$ avec $v, w \in \Delta_\iota^*$, $(c,r) \in \Delta_c \times \Delta_r$ et $u' \in \Delta_{\mathsf{lin}}^*$ alors nous avons :

$$\rho' = (p_1, \sigma\gamma) \xrightarrow{a_1} \dots \xrightarrow{a_l} (p_{l+1}, \sigma\gamma) \xrightarrow{c} (p'', \sigma\gamma\gamma') \xrightarrow{u'}{}^* (q'', \sigma\gamma\gamma') \xrightarrow{r} (q_{k+1}, \sigma\gamma) \xrightarrow{b_k} \cdots \xrightarrow{b_1} (q_1, \sigma\gamma)$$

avec $v = a_1 \dots a_l$, $w = b_1 \dots b_k$, $p = p_1$ et $q = q_1$. Par hypothèse d'induction $\rho_3 : (p'', q'', 0, \gamma'') \xrightarrow{\mathsf{tralin}(u')}{}^* (p', p', 0, \gamma')$ est un calcul de A' sur $\mathsf{tralin}(u')$ tel que $\Phi_{\mathsf{vf}}(\rho_3, \sigma\gamma) = (p'', \sigma\gamma\gamma'') \xrightarrow{u'}{}^* (q'', \sigma\gamma\gamma'')$. De plus par définition de δ' pour tout $i \in [1,l]$ et $j \in [1,k]$ nous avons :

$$\begin{aligned}&((p_{l+1}, q_{k+1}, n_{k+1}, \gamma), (c,r), (p'', q'', 0, \gamma'')) \qquad ((p_i, q_1), 0, \gamma), (a_i, \top), (p'_{i+1}, q_1, 0, \gamma))\\ &((p_{l+1}, q_j, n_j, \gamma), (\top, b_j), (p_{l+1}, q_{j+1}, n_{j+1}, \gamma)) \in \delta'\end{aligned}$$

avec $n_j \in [0,1]$. En appliquant itérativement ϕ_{vf} nous obtenons que $\Phi_{\mathsf{vf}}(\rho, \sigma) = \rho'$ avec :

$$\begin{aligned}\rho_1 &: (p_1, q_1, 0, \gamma) \xrightarrow{(a_1,\top)} (p_2, q_1, 0, \gamma) \xrightarrow{(a_l,\top)} \cdots (p_{l+1}, q_1, 0, \gamma)\\ \rho_2 &: (p_{l+1}, q_1, 0, \gamma) \xrightarrow{(b_1,\top)} (p_{l+1}, q_2, 1, \gamma) \xrightarrow{(b_k,\top)} \cdots (p_{l+1}, q_{k+1}, 1, \gamma)\end{aligned}$$

et $\rho : \rho_1\rho_2 \xrightarrow{c,r} \rho_3$. □

Lemme 5.4.9

Soit $A = (Q, I, F, \Gamma, \delta)$ un APVlin sur un alphabet Δ et $u \in \Delta^*_{\mathsf{lin}}$ un mot linéaire. Pour tout $\sigma \in \Gamma^*$ et tout calculs ρ_1 et ρ_2 de $\mathsf{APV2AA}(A)$ sur le mot $v = \mathsf{tralin}(u)$ de la forme :

$$\rho_1 : (p_1, q_1, 0, \gamma_1) \xrightarrow{\mathsf{tralin}(u)}{}^*(p'_1, p'_1, 0, \gamma'_1) \qquad \rho_2 : (p_2, q_2, 0, \gamma_2) \xrightarrow{\mathsf{tralin}(u)}{}^*(p'_2, p'_2, 0, \gamma'_2)$$

si $\Phi_{\mathsf{vf}}(\rho_1, \sigma) = \Phi_{\mathsf{vf}}(\rho_2, \sigma)$ et $\gamma_1 = \gamma_2$ alors $\rho_1 = \rho_2$.

Démonstration. Nous allons prouver ce lemme par induction sur la structure de v.

- Si $v = \mathsf{trag}(u)$ alors par définition de δ', ρ_1 et ρ_2 se décomposent de la manière suivante :

  $$(p_{i,1}, q_i, 0, \gamma_i) \xrightarrow{(a_{i,1}, \top)} (p_{1,2}, q_i, 0, \gamma_i) \xrightarrow{(a_{i,2}, \top)}{}^*(p_{1,n+1}, q_i, 0, \gamma_i)$$

  avec $p_{i,1} = p_i$ et $p_{i,n+1} = q_i = p'_i$ pour tout $i \in [1, 2]$. Comme $\Phi_{\mathsf{vf}}(\rho_1, \sigma) = \Phi_{\mathsf{vf}}(\rho_2, \sigma)$, nous déduisons que $q_1 = q_2$ et $p_{1,j} = p_{2,j}$ pour tout $j \in [1, n]$, et donc de $\gamma_1 = \gamma_2$ vient que $\rho_1 = \rho_2$.
- S'il existe $u_1, u_2 \in \Delta^*_t$, $u_3 \in \Delta^*_{\mathsf{lin}}$ et $(c, r) \in \Delta_c \times \Delta_r$ tels que $v = \mathsf{trag}(u_1)\mathsf{trad}(u_2)(c, r)\mathsf{tralin}(u_3)$ alors par définition de δ', ρ_i se décompose par $\rho_{i,1}\rho_{i,2} \xrightarrow{(c,r)} \rho_{i,3}$ avec :

  $$\begin{aligned} \rho_{i,1} : (p_{i,1}, q_{i,1}, 0, \gamma_i) &\xrightarrow{(a_{i,1}, \top)} (p_{i,2}, q_{i,1}, 0, \gamma_i) \xrightarrow{(a_{i,2}, \top)}{}^* \cdots (p_{i,n+1}, q_{i,1}, 0, \gamma_i) \\ \rho_{i,2} : (p_{i,n+1}, q_{i,1}, 0, \gamma_i) &\xrightarrow{(\top, b_{i,1})} (p_{i,n+1}, q_{i,2}, 1, \gamma_i) \xrightarrow{(\top, b_{i,2})}{}^* \cdots (p_{i,n+1}, q_{i,m+1}, 1, \gamma_i) \\ \rho_{i,3} : (p''_i q''_i, 0, \gamma''_i) &\xrightarrow{\mathsf{tralin}(u_3)}{}^*(p'_i, p'_i, 0, \gamma'_i) \end{aligned}$$

  avec $p_{i,1} = p_i$ et $q_{i,1} = q_i$ pour tout $i \in [1, 2]$. Comme $\Phi_{\mathsf{vf}}(\rho_1, \sigma) = \Phi_{\mathsf{vf}}(\rho_2, \sigma)$ nous déduisons que $p_{i,j} = p_{i,j}$ et $q_{i,l} = q_{i,l}$ pour tout $j \in [1, n]$, $l \in [1, m]$ et $i \in [1, 2]$. De plus, par hypothèse d'induction, nous savons que $\rho_{1,3} = \rho_{2,3}$. Comme $\gamma_1 = \gamma_2$ nous concluons que $\rho_1 = \rho_2$. □

Nous pouvons maintenant combiner ces résultats afin de montrer le Théorème 5.4.4.

Démonstration du Théorème 5.4.4. Les Lemmes 5.4.8, 5.4.9 et la définition des états finaux de A_{vf} assurent que pour tout mot $u \in \Delta^*_{\mathsf{lin}}$, $\mathsf{ACal}_u(A)$ est bijectif avec $\mathsf{ACal}_{\mathsf{tralin}(u)}(A_{\mathsf{vf}})$. De plus, par le Lemme 5.4.6, pour tout mot $w \in \mathcal{L}(A_{\mathsf{vf}})$ il existe $u \in \Delta^*_{\mathsf{lin}}$ tel que $u = \mathsf{tralin}(w)$, ceci assurant la propriété souhaitée sur tous les calculs acceptants de A et A_{vf}. □

5.4.3 Extension aux transducteurs

Nous étendons maintenant les résultats de la section précédente aux transducteurs. Pour cela, nous allons étendre la construction APV2AF aux TPVlin et aux M2C et montrer comment construire les sorties sur les transitions. Concrètement, un TPVlin peut se séparer en deux partie, la partie empilant, et la partie dépilant. Nous allons donc construire un M2C où la sortie produite par la partie empilant sera produite par le mot de gauche, et la partie dépilant par le mot de droite.

Définition 5.4.10

Soient $T = (A, \lambda)$ un TPVlin sur l'alphabet structuré Δ vers l'alphabet structuré Δ'. Le M2C $\mathsf{APV2AF}(T) = (A', \lambda')$ sur l'alphabet $\tilde{\Delta}$ vers l'alphabet Δ' est défini par :

- $A' = \mathsf{APV2AF}(A) = (Q', I', F', \delta')$,
- $\lambda'(\tau) = \begin{cases} (\lambda(p,a,p'),\epsilon) & \text{si } \tau = ((p,q,0,\gamma),(a,\top),(p',q,0,\gamma)), \\ (\epsilon,\lambda(p,a,p')) & \text{si } \tau = ((p,q,n,\gamma),(\top,a),(p,q',1,\gamma)), \\ (\lambda(p,c,\gamma',p'),\lambda(q,\gamma',r,q')) & \text{si } \tau = ((p,q,0,n,\gamma),(c,r),(p',q',0,\gamma)), \end{cases}$

  Pour tout $\tau \in \delta'$.

Nous souhaitons maintenant montrer le résultat suivant :

Théorème 5.4.11

Soient $T = (A, \lambda)$ un TPVlin et $T_{\mathsf{vf}} = (A_{\mathsf{vf}}, \lambda_{\mathsf{vf}})$ le M2C $\mathsf{APV2AF}(T)$. Nous avons :

- $\mathsf{trans}(T_{\mathsf{vf}}) = \{(\mathsf{tralin}(u), v) \mid (u,v) \in \mathsf{trans}(T)\}$,
- Pour tout $u \in \mathcal{L}(A)$ il existe une bijection entre $\mathsf{ACal}_u(A)$ et $\mathsf{ACal}_{\mathsf{tralin}(u)}(A_{\mathsf{vf}})$.

Pour montrer ce résultat nous donnons la propriété suivante de la fonction Φ_{vf}. De ce résultat découle directement la preuve du Théorème 5.4.11.

Lemme 5.4.12

Soit $T = (A, \lambda)$ un TPVlin. Pour tout calcul acceptant ρ de A nous avons l'égalité $\mathsf{sortie}(\Phi_{\mathsf{va}}(\rho, \bot)) = \mathsf{sortie}(\rho)$.

Démonstration. Par construction de λ'. □

5.5 Décidabilité de la norme finie transducteur mot-vers-contexte

Dans cette section nous allons présenter nos résultats concernant la décidabilité de la norme finie des transducteurs mot-vers-contexte. Ce travail est une extension des résultats présentés dans les trois sections précédentes. Nous allons tout d'abord présenter deux propriétés dont nous conjecturons qu'elles caractérisent les transducteurs dont la norme est infinie.

5.5.1 Propriétés

Nous présentons **(G0)** et **(G1)**. Soit $T = (A, \lambda)$ un M2C émondé sur l'alphabet Σ.

Propriété (G0) T a la propriété **(G0)** s'il existe un mot $u \in \Sigma^*$ et un état final q_f tels que :

$p \xrightarrow{u,c_1}{}^* p \in \mathsf{Cal}_u(A)$, $p \xrightarrow{u,c_2}{}^* q \in \mathsf{Cal}_u(A)$, $q \xrightarrow{u,c_3}{}^* q \in \mathsf{Cal}_u(A)$

$q \xrightarrow{u',c}{}^* q_f \in \mathsf{Cal}_v(A)$

et $c_1c_2c[\epsilon] \neq c_2c_3c[\epsilon]$.

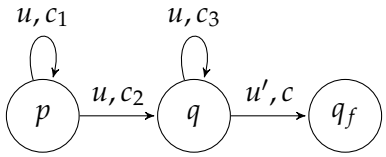


FIGURE 5.7 – La propriété **(G0)**

Propriété (G1) T a la propriété **(G1)** s'il existe des mots $u, v, w \in \Sigma^*$ et un état final q_f tels que :

$p \xrightarrow{u,c_{11}}{}^* p_1 \in \mathsf{Cal}_u(A),\ p_1 \xrightarrow{v,c_{12}}{}^* p_1 \in \mathsf{Cal}_v(A),\ p_1 \xrightarrow{w,c_{13}}{}^* p \in \mathsf{Cal}_w(A),$
$p \xrightarrow{u,c_{21}}{}^* p_2 \in \mathsf{Cal}_u(A),\ p_2 \xrightarrow{v,c_{22}}{}^* p_2 \in \mathsf{Cal}_v(A),\ p_2 \xrightarrow{w,c_{23}}{}^* q \in \mathsf{Cal}_w(A),$
$q \xrightarrow{u,c_{31}}{}^* p_3 \in \mathsf{Cal}_u(A),\ p_3 \xrightarrow{v,c_{32}}{}^* p_3 \in \mathsf{Cal}_v(A),\ p_3 \xrightarrow{w,c_{33}}{}^* q \in \mathsf{Cal}_w(A),$
$q \xrightarrow{u',c}{}^* q_f \in \mathsf{Cal}_v(A)$

et $c_{11}c_{12}c_{13}c_{21}c_{23}c[\epsilon] \neq c_{21}c_{22}c_{23}c_{31}c_{33}c[\epsilon]$.

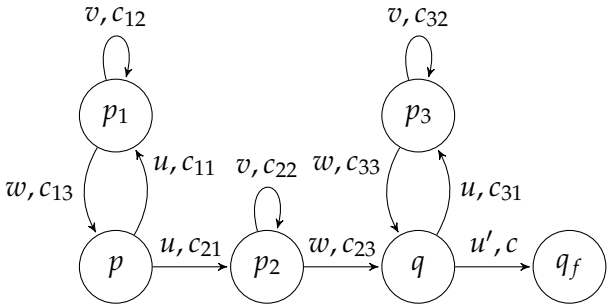


FIGURE 5.8 – La propriété **(G1)**

La propriété **(G0)** est représentée à la Figure 5.7 et la propriété **(G1)** à la Figure 5.8. Notez que la propriété **(G1)** est plus proche de la propriété **(F1)** pour les transducteurs d'arbres que de la propriété **(W1)** pour les transducteurs finis. La Figure 5.5 illustre pourquoi nous avons besoin ce cette complexité : le transducteur représenté ne satisfait pas **(G0)** et pour tout $k \in \mathbb{N}_{>0}$ les sorties des mots $\Pi_{i=1}^{k}(ab^ic)d$ sont de même taille.

De plus, nous avons dû ajouter le calcul permettant d'atteindre l'état final car contrairement aux transducteurs finis et aux transducteurs d'arbres, nous pouvons avoir $c_1c_2 \neq c_2c_3$ et $c_1c_2c[\epsilon] = c_2c_3c[\epsilon]$, comme représenté à la Figure 5.5.

Il n'y a pas d'équivalent à la propriété **(F2)**, qui n'a pas de sens pour les M2C. Il est très probable que cette propriété soit présente dans le cas des TPV généraux. Nous souhaitons montrer le résultat suivant :

Conjecture 5.5.1

> Soit T un M2C émondé. La norme de T est infinie si et seulement s'il satisfait l'une des propriétés **(G0)** ou **(G1)**.

Nous montrerons à la Section 5.5.3 que cette condition est suffisante, c'est à dire que si T satisfait l'une des propriétés alors sa norme est infinie. Puis nous discuterons à la Section 5.5.4 de la nécessité de la condition. Avant cela nous avons besoin d'introduire des éléments de combinatoire des mots.

5.5.2 Combinatoire des mots

Dans cette section nous allons présenter quelques résultats de la combinatoire des mots. Une présentation de la thématique peut être trouvée dans [BK03]. Nous allons tout d'abord présenter les définitions de base et introduire le théorème de Fine et Wilf, l'un des théorèmes fondamentaux de la combinatoire des mots. Enfin, nous prouverons plusieurs lemmes que nous utiliserons dans la Section 5.5.3.

Racine primitive Soit $w \in \Sigma^*$ un mot, la *racine primitive* $\rho(w)$ de w est le mot de longueur minimale $v \in \Sigma^*$ tel que $w \in \{v\}^*$. De plus, si $\rho(w) = w$ alors nous appelons w un mot *primitif*.

Conjugaison Soient $v, w \in \Sigma^*$ deux mots, alors v and w sont *conjugués* s'il existe deux mots $x, y \in \Sigma^*$ tels que $v = xy$ et $w = yx$. Nous notons cette relation entre v et w par $v \sim w$. Nous montrons maintenant que cette relation est transitive :

Lemme 5.5.2

> Soient $u, v, w \in \Sigma^*$ des mots tels que $u \sim v$ and $v \sim w$, alors $u \sim w$.

Démonstration. Par définition de la conjugaison il existe des mots $x, y, x', y' \in \Sigma^*$ tels que $u = xy$, $v = yx = y'x'$ et $w = x'y'$. Nous distinguons deux cas :

Cas 1 $|y| \geqslant |y'|$ **:** Il existe un mot $z \in \Sigma^*$ tel que $y = y'z$ et $x' = zx$, et donc nous avons $u = (xy')(z)$ et $w = (z)(xy')$. Nous pouvons conclure que $u \sim w$.

Cas 2 $|y| < |y'|$ **:** Il existe un mot $z \in \Sigma^*$ tel que $y' = yz$ et $x = zx'$, et donc nous avons $u = (z)(x'y)$ et $w = (x'y)(z)$. Nous pouvons conclure que $u \sim w$. □

Le théorème suivant est un classique de la combinatoire des mots et a été établi par Nathan Fine et Herbert Wilf en 1965 dans [FW65] dans le cadre de l'étude des fonctions périodiques. Il établit que deux mots possédant un sous-mot commun de longueur suffisante définissent des racines primitives conjuguées.

Théorème 5.5.3: Fine et Wilf

> Soient $u, v \in \Sigma^*$ deux mots et $m \in \mathbb{N}_{>0}$ un entier tels qu'il existe un sous-mot commun de u^m et v^m de longueur au moins $|u| + |v| - \mathsf{pgcd}(|u|, |v|)$, alors $\rho(u) \sim \rho(v)$.

Le lemme suivant est un résultat classique de la combinatoire des mots. Il est particulièrement utile pour démontrer par contradiction un résultat incluant des mots primitifs.

Lemme 5.5.4

> Soient $u, v \in \Sigma^*$ des mots tels que $uv = vu$, alors il existe $z \in \Sigma^*$ tel que $u, v \in \{z\}^*$. De plus, si uv est primitif alors $u = \epsilon$ ou $v = \epsilon$.

Démonstration. Soient $u, v \in \Sigma^*$ des mots tels que $uv = vu$. Nous montrons en premier qu'il existe $z \in \Sigma^*$ tel que $u, v \in \{z\}^*$. Si $u = \epsilon$ (resp. $v = \epsilon$) alors $u \in \{v\}^*$ (resp. $v \in \{u\}^*$). Nous supposons maintenant que $u \neq \epsilon$ et $v \neq \epsilon$. Nous prouvons ce cas par induction sur la longueur de $|u| + |v|$, et supposons que $|u| \geqslant |v|$, le cas où $|u| \leqslant |v|$ étant symétrique à celui-ci.

Si $|u| = |v| = 1$, alors $u = v$. Nous pouvons en déduire que $u \in \{v\}^*$.

Nous considérons maintenant le cas où $|u| + |v| > 2$. Comme $uv = vu$ et $|u| \geqslant |v|$, il existe un mot $w \in \Sigma^*$ tel que $u = vw$. Nous considérons l'égalité $vwv = vvw$, conduisant à $wv = vw$. Nous avons $|w| + |v| < |u| + |v|$. Par hypothèse d'induction nous en déduisons qu'il existe $z \in \Sigma^*$ tel que $w \in \{z\}^*$ et $v \in \{z\}^*$. Comme $u = vw$, nous pouvons conclure que $u \in \{z\}^*$.

Pour la deuxième partie du lemme, si nous supposons que $u \neq \epsilon$ et $v \neq \epsilon$, alors il existe $n \in \mathbb{N}_{>1}$ tel que $uv = z^n$. Cela contredisant la primitivité de uv nous avons $u = \epsilon$ ou $v = \epsilon$. □

Nous présentons plusieurs lemmes techniques que nous utiliserons à la section suivante.

Lemme 5.5.5

> Soient $u, v, x \in \Sigma^*$ des mots et $n, m \in \mathbb{N}_{>0}$ des entiers positifs tels que x est primitif et $ux^nv = x^m$, alors il existe $l \in \mathbb{N}$ tel que $u = x^l$.

Démonstration. Comme $ux^nv = x^m$, il existe des mots $w_1, w_2, w_3, w_4 \in \Sigma^*$ et des entiers $l_1, l_2, l_3 \in \mathbb{N}$ tels que $u = x^{l_1}w_1$, $x^n = w_2x^{l_2}w_3$, $v = w_4x^{l_3}$, $w_1w_2, w_3, w_4 \in \{\epsilon, x\}$, $w_1 = \epsilon \Leftrightarrow w_2 = \epsilon$ et $w_3 = \epsilon \Leftrightarrow w_4 = \epsilon$. De $x^n = w_2x^{l_2}w_3$ nous pouvons déduire que soit $|w_2w_3| = |x|$ soit $w_2w_3 = \epsilon$.

Si $w_2w_3 = \epsilon$, comme $w_1 = \epsilon \Leftrightarrow w_2 = \epsilon$ nous pouvons conclure que $u = x^{l_1}$.

Sinon, si $|w_2w_3| = |x|$, nous avons $x = w_1w_2 = w_3w_4$ et donc $w_1 = w_3$, $w_2 = w_4$. De plus puisque $x^n = w_2x^{l_2}w_3$, il existe des mots $w_5, w_6 \in \Sigma^*$ tels que $x = w_2w_5 = w_6w_3$, et de $|w_2w_3| = |x|$ nous obtenons $w_2 = w_6$, $w_5 = w_3$. Nous pouvons donc conclure que $w_2w_3 = x$ et $w_3w_2 = x$. Cela implique par le Lemme 5.5.4 qu'il existe $z \in \Sigma^*$ tel que $w_2, w_3 \in \{z\}^*$. Comme $w_1 = \epsilon \Leftrightarrow w_2 = \epsilon$ et $w_3 = \epsilon \Leftrightarrow w_4 = \epsilon$, nous avons $w_2 \neq \epsilon$ et $w_3 \neq \epsilon$. De cela nous concluons que $x = z^k$ pour $k \in \mathbb{N}$ un entier tel que $k > 1$, ceci contredisant la primitivité de x. □

Lemme 5.5.6

> Soient $u, v, x, y \in \Sigma^*$ des mots et $n, m \in \mathbb{N}_{>0}$ des entiers positifs tels que xy est primitif et $u(xy)^nv = (yx)^m$, alors il existe $k \in \mathbb{N}$ tel que $ux = (yx)^k$ et $xu = (xy)^k$.

Démonstration. Premièrement considérons que $x = \epsilon$. Par le Lemme 5.5.5 nous pouvons en déduire qu'il existe un entier $l \in \mathbb{N}$ tel que $u = y^l$. Comme $xy = y$, nous avons $u = (xy)^l = (yx)^l$, $ux = (xy)^lx$ et $xu = (yx)^lx$. Si $y = \epsilon$, alors par le Lemme 5.5.5 il existe un entier $l \in \mathbb{N}$ tel que $u = x^l$. Comme $xy = x$, nous avons $u = y(xy)^l = (yx)^ly$, $ux = (yx)^lyx = (yx)^{l+1}$ et $xu = xy(xy)^l = (xy)^{l+1}$.

Nous faisons maintenant l'hypothèse que x et y sont différents de ϵ. Comme $u(xy)^nv = (yx)^m$, il existe des mots $w_1, w_2, w_3, w_4 \in \Sigma^*$ et des entiers $l_1, l_3 \in \mathbb{N}$ tels que $u = (yx)^{l_1}w_1$, $v = w_4(yx)^{l_3}$ et $w_1w_2, w_3w_4 \in \{\epsilon, yx\}$. Nous avons donc

$$u(xy)^nv = (yx)^{l_1}w_1(xy)^nw_4(yx)^{l_3} \qquad \text{et} \qquad (yx)^m = (yx)^{l_1}(yx)^{l_2}(yx)^{l_3}$$

Avec $l_2 \in \mathbb{N}$ un entier tel que $m = l_1 + l_2 + l_3$. Comme $u(xy)^nv = (yx)^m$ nous en concluons donc :

$$w_1(xy)^nw_4 = (yx)^{l_2}$$

Nous pouvons en déduire que soit $n = l_2 - 1$ et $|w_1w_4| = |yx|$ soit $n = l_2$ et $w_1w_4 = \epsilon$. Si $w_1w_4 = \epsilon$, alors $xy = yx$. Comme xy est primitif, cela implique par le Lemme 5.5.4 que soit x est vide soit y est vide, ce qui est une contradiction.

Nous supposons maintenant que $|w_1w_4| = |yx|$. Nous avons $yx = w_1w_2 = w_3w_4$, et donc $w_1 = w_3$ et $w_2 = w_4$. Nous distinguons trois cas :

Cas 1 $|w_1| = |y|$ **:** Comme $yx = w_1w_2 = w_3w_4$, nous en concluons que $y = w_1 = w_3$ et $x = w_2 = w_4$. Nous avons donc bien que $u = (yx)^{l_1}y = y(xy)^{l_1}$, $ux = (yx)^{l_1+1}$ et $xu = (xy)^{l_1+1}$.

Cas 2 $|w_1| < |y|$ **:** Comme $yx = w_1w_2$, il existe $w_5 \in \Sigma^*$ tel que $y = w_1w_5$. De plus $|w_1w_4| = |yx|$ et $yx = w_3w_4$, nous pouvons en déduire qu'il existe $w_6 \in \Sigma^*$ tel que $y = w_3w_6$ et $w_4 = w_6x$, et de $w_1 = w_3$ nous avons $w_6 = w_5$. Cela conduit à :

$$w_1(xy)^nw_4 = w_1(xy)^nw_6x = (yx)^{l_2} = w_1w_5(xy)^{l_2-1}x$$

Nous avons $(xy)^nw_6 = w_5(xy)^{l_2-1}$, et puisque $|xy| > |w_6| = |w_5|$ nous déduisons que $(xy)w_6 = w_5(xy)$ par l'égalité $n = l_2 - 1$. De $w_5 = w_6$ le Lemme 5.5.4 nous dit qu'il existe un mot $z \in \Sigma^*$ tel que $(xy) \in \{z\}^*$ et $w_5 \in \{z\}^*$. Comme $|w_5| < |xy|$ et $w_5 \neq \epsilon$, nous pouvons conclure que $(xy) = z^p$ avec $p \in \mathbb{N}$ un entier tel que $p > 1$, ce qui est une contradiction avec la primitivité de xy.

Cas 3 $|w_1| > |y|$ **:** Comme $yx = w_1w_2$, il existe $w_5 \in \Sigma^*$ tel que $x = w_5w_2$ et $w_1 = yw_5$. De plus $|w_1w_4| = |yx|$ et $yx = w_3w_4$, nous pouvons en déduire qu'il existe $w_6 \in \Sigma^*$ tel que $x = w_6w_4$, et de $w_2 = w_4$ nous avons $w_6 = w_5$. Cela conduit à :

$$w_1(xy)^nw_4 = yw_5(xy)^nw_4 = (yx)^{l_2} = y(xy)^{l_2-1}w_6w_4$$

Nous avons $w_5(xy)^n = (xy)^{l_2-1}w_6$ et puisque $|xy| > |w_6| = |w_5|$ nous déduisons que $w_5(xy) = (xy)w_6$ par l'égalité $n = l_2 - 1$. De $w_5 = w_6$ le Lemme 5.5.4 nous dit qu'il existe un mot $z \in \Sigma^*$ tel que $(xy) \in \{z\}^*$ et $w_5 \in \{z\}^*$. Comme $|w_5| < |xy|$ et $w_5 \neq \epsilon$, nous pouvons conclure que $(xy) = z^p$ avec $p \in \mathbb{N}$ un entier tel que $p > 1$, ce qui est une contradiction avec la primitivité de xy. □

Lemme 5.5.7

Soient $u, v, w, x_1, y_1, x_2, y_2 \in \Sigma^*$ des mots et $n, m, l \in \mathbb{N}_{>0}$ des entiers positifs tels que $u(x_1y_1)^nv(x_2y_2)^mw = (y_1x_1)^l$ et $y_1x_1 = y_2x_2$, alors il existe un entier $k \in \mathbb{N}$ tel que $vx_2 = x_1(y_2x_2)^k$ et $y_1v = (y_1x_1)^kx_2$.

Démonstration. Par le Lemme 5.5.6, il existe des entiers $l_1, l_2 \in \mathbb{N}$ tels que nous avons $ux_1y_1 = y_1(x_1y_1)^{l_1}$ et $xy_2w_2 = (x_2y_2)^{l_2}x_2$. De là nous déduisons que

$$u(x_1y_1)^nv(x_2y_2)^mw = y_1(x_1y_1)^{l_1}(x_1y_1)^{n-1}v(x_2y_2)^{m-1}(x_2y_2)^{l_2}x_2$$

Comme $u(x_1y_1)^nv(x_2y_2)^mw = (y_1x_1)^l$ nous concluons que $vx_2 = x_1(y_2x_2)^k$ et $y_1v = (y_1x_1)^kx_2$ avec $k = l - (l_1 + (n-1) + (m-1) + l_2 + 1)$. □

Lemme 5.5.8

Soient u, v et $w \in \Sigma^*$ trois mots, $j \in \mathbb{N}$ un entier et $(z_i)_{i \in \mathbb{N}_{>j-1}}$ une famille de mots sur Σ^* tels que pour tout $i \in \mathbb{N}_{>j-1}$ nous avons $uv^iw \in \{z_i\}^*$ et $|z_i| = |z_{i+1}|$, alors pour tout $i \in \mathbb{N}$ nous avons $uv^iw \in z_j^*$.

Démonstration. Soit $i \in \mathbb{N}_{>0}$ un entier, $z \in \Sigma^*$ un mot, et $p_i(z)$ la propriété suivante :

$$p_i(z) : uv^i w \in z^* \text{ et } |z| \equiv 0 \pmod{|v|}$$

Nous allons maintenant démontrer que :

$$p_i(z) \Rightarrow p_{i-1}(z) \tag{5.1}$$

Comme $uv^i w \in \{z\}^*$, il existe quatre mots $w_1, w_2, w_3, w_4 \in \Sigma^*$ et trois entiers $k_1, k_2, k_3 \in \mathbb{N}$ tels que $uv^{i-1} = z^{k_1} w_1$, $v = w_2 z^{k_2} w_3$, $w = w_4 z^{k_3}$, $w_1 w_2, w_3 w_4 \in \{z, \epsilon\}$, $w_1 = \epsilon \Leftrightarrow w_2 = \epsilon$ et $w_3 = \epsilon \Leftrightarrow w_4 = \epsilon$. Comme $|z| \equiv 0 \pmod{|v|}$, nous avons $w_2 w_3 \in \epsilon$ ou $|w_2 w_3| = |z|$.

Si $|w_2 w_3| = 0$ alors comme $w_1 = \epsilon \Leftrightarrow w_2 = \epsilon$ et $w_3 = \epsilon \Leftrightarrow w_4 = \epsilon$, nous concluons que $w_1 = w_4 = \epsilon$. Cela amène que $uv^i w = z^{k_1+k_2+k_3}$ et $uv^{i-1} w = z^{k_1+k_3}$.

Si $|w_2 w_3| = |z|$ alors $|w_1 w_4| = |z|$, et donc $|w_1| = |w_3|$ et $|w_2| = |w_4|$ comme $|w_1 w_2| = |z|$ et $|w_3 w_4| = |z|$. De $w_1 w_2 = z$ et $w_3 w_4 = z$ nous concluons donc que $w_1 w_4 = z$. Cela amène que $uv^i w = z^{k_1+k_2+k_3+2}$ et $uv^{i-1} w = z^{k_1+k_3+1}$.

Nous pouvons maintenant prouver le lemme en lui même. Nous appelons α l'entier $|z_j|$. Comme $uv^j w \in z_j^*$, nous avons $\alpha \equiv 0 \pmod{|uv^j w|}$. Comme pour tout $i\mathbb{N}_{>j-1}$ nous avons $|z_i| = |z_{i+1}|$, nous pouvons déduire que $\alpha \equiv 0 \pmod{|v|}$ et donc $p_i(z_i)$ est vrai. Soit $i \in \mathbb{N}$ un entier. nous distinguons deux cas :

- **Cas 1** $i < j$ **:** Par l'Équation (5.1) nous avons $p_j(z_j) \Rightarrow p_i(z_j)$, donc $p_i(z_j)$ est vrai et $uv^i w \in z_j^*$.
- **Cas 2** $i \geqslant j$ **:** Par l'Équation (5.1) nous avons $p_i(z_i) \Rightarrow p_j(z_i)$, donc $p_j(z_i)$ est vrai et $uv^i w \in z_j^*$ comme $|z_i| = |z_j| = \alpha$. □

Nous avons traité jusqu'à présent de résultats permettant de comparer des mots égaux, nous allons maintenant parler de résultats permettant de traiter des mots différents.

Lemme 5.5.9

> Soient u, v, z_1, and $z_2 \in \Sigma^*$ des mots tels que $|u| = |v| \neq 0$ et $|z_1| = |z_2|$. Si $u \neq v$ alors il existe $n \in \mathbb{N}$ tel que pour tout $i \in \mathbb{N}_{>n}$ nous avons $uz_1^i \neq z_2^i v$.

Démonstration. Premièrement, nous considérons que $z_1 = z_2 = \epsilon$. Pour tout $i \in \mathbb{N}$, nous avons $u = uz_1^i$ et $v = z_2^i v$, et donc $uz_1^i \neq z_2^i v$. Nous considérons maintenant que $z_1 \neq \epsilon$. Soit $n, i \in \mathbb{N}$ des entiers tels que $|z_1^n| \geqslant |u|$ et $i > 2n$. Nous supposons que $uz_1^i = z_2^i v$. Il existe donc des mots $y_1, y_2 \in \Sigma^*$ tels que $z_2^n = uy_2$ et $z_1^n = y_1 v$. Nous avons $|y_1| = |y_2|$. Comme $i > 2n$, nous pouvons écrire uz_1^i et $z_2^i v$ comme :

$$\begin{aligned} u \cdot z_1^i &= u(y_1 v)(y_1 v)(y_1 v)^{k-2} z_1^{k'} \\ &= (uy_1)(vy_1)(vy_1)^{k-2} v z_1^{k'} \\ z_2^i v &= (uy_2)(uy_2)(uy_2)^{k-2} z_2^{k'} v \end{aligned}$$

avec $k = \lfloor i/n \rfloor$ et $k' = i - nk$. De $u \neq v$ et $|uy_1| = |uy_2|$ nous concluons que $uz_1^i \neq z_2^i v$. □

Lemme 5.5.10

Soient $u, x, v \in \Sigma^*$ trois mots tels que $|u| = |v|$ et pour tout $i \in \mathbb{N}_{>1}$ et $j \in [i-2]$ nous avons $u^j x v^{i-j} \neq u^{j+1} x v^{i-j-1}$, alors pour tout $j' \in [i-1]$ tel que $j' \neq j$ nous avons $u^j x v^{i-j} \neq u^{j'} x v^{i-j'}$.

Démonstration. Premièrement notez que $u \neq \epsilon$ et $v \neq \epsilon$. Pour tout $i \in \mathbb{N}_{>1}$ et $j \in [1, i-2]$, nous appelons $\eta_{i,j}$ le mot $u^j x v^{i-j}$. Il est suffisant de montrer que pour tout $j' \in [j+1, i-2]$ nous avons $\eta_{i,j} \neq \eta_{i,j'}$. Pour tout $j \in [1, i-2]$, nous définissons $W_{i,j} \in [|\eta_{i,j}|]$ comme le plus petit entier tel que $\eta_{i,j}(W_{i,j}) \neq \eta_{i,j+1}(W_{i,j})$. Nous notons $w_1 = ux$ et $w_2 = xv$. Comme nous avons $\eta_{i,j} = u^j w_2 v^{i-j-1} \neq \eta_{i,j+1} = u^j w_1 v^{i-j-1}$ nous pouvons dire que $W_{i,j} + |u| = W_{i,j+1}$. Comme $u \neq \epsilon$ pour tout $j' \in [j+1, i-2]$ nous avons $\eta_{i,j}(W_{i,j}) \neq \eta_{i,j+1}(W_{i,j}) = \eta_{i,j'}(W_{i,j})$ et donc $\eta_{i,j} \neq \eta_{i,j'}$. □

Enfin, nous présentons un dernier résultat concernant la longueur des mots :

Lemme 5.5.11

Soient $u_1, v_1, w_1, u_2, v_2, w_2 \in \Sigma^*$ des mots et $L = \max(|u_1 w_1|, |u_2 w_2|) + 1$. Pour tout $l \geqslant L$:

- Si $|u_1 v_1^L w_1| > |u_2 v_2^L w_2|$ alors $|v_1| \geqslant |v_2|$ et $|u_1 v_1^l w_1| > |u_2 v_2^l w_2|$,
- Si $|u_1 v_1^L w_1| = |u_2 v_2^L w_2|$ alors $|v_1| = |v_2|$ et $|u_1 v_1^l w_1| = |u_2 v_2^l w_2|$.

Démonstration. Nous considérons en premier que $|u_1 v_1^L w_1| > |u_2 v_2^L w_2|$. Si $|v_1| < |v_2|$ alors $|v_2^L| - |v_1^L| \geqslant L$. Comme $L = \max(|u_1 w_1|, |u_2 w_2|) + 1$ nous avons $|u_1 v_1^L w_1| < |u_2 v_2^L w_2|$, ce qui est une contradiction avec l'énoncé du lemme. Nous avons donc $v_1 \geqslant v_2$. Trivialement nous obtenons que pour tout $l > L$ $|u_1 v_1^l w_1| - |u_2 v_2^l w_2| \geqslant |u_1 v_1^{l-1} w_1| - |u_2 v_2^{l-1} w_2|$, et donc $|u_1 v_1^l w_1| > |u_2 v_2^l w_2|$. Nous considérons maintenant que $|u_1 v_1^L w_1| = |u_2 v_2^L w_2|$. Si $|v_1| \neq |v_2|$ alors $||v_2^L| - |v_1^L|| \geqslant L$. Comme $L = \max(|u_1 w_1|, |u_2 w_2|) + 1$ nous en déduisons que $|u_1 v_1^L w_1| \neq |u_2 v_2^L w_2|$, ce qui est une contradiction avec l'énoncé du lemme. Nous avons donc $|v_1| = |v_2|$. Trivialement nous obtenons que pour tout $l > L$ $|u_1 v_1^l w_1| - |u_2 v_2^l w_2| = |u_1 v_1^{l-1} w_1| - |u_2 v_2^{l-1} w_2|$, et donc $|u_1 v_1^l w_1| = |u_2 v_2^l w_2|$. □

5.5.3 Preuve de la condition suffisante

Dans cette section nous allons prouver la Proposition suivante :

Proposition 5.5.12

Soit T un M2C émondé. Si T satisfait l'une des propriétés **(G0)** ou **(G1)** alors sa norme est infinie.

Le Lemme 5.5.13 montre que si un M2C T satisfait **(G0)** alors sa norme est infinie, et le Lemme 5.5.14 montre que si un M2C T satisfait **(G1)** alors sa norme est infinie.

Lemme 5.5.13

Soit T un M2C émondé de Σ vers Σ'. Si T satisfait **(G0)** alors sa norme est infinie.

Démonstration. Nous notons $T = (A, \lambda)$ avec $A = (Q, I, F, \delta)$. Soient $u, u' \in \Sigma^*$ des mots et $c_1, c_2, c_3, c' \in \Sigma'^* \times \Sigma'^*$ des contextes choisis comme dans la propriété **(G0)**. Pour tout $i \in [1,3]$ nous écrivons $c_i = (u_i, v_i)$ et nous appelons u'_1 le mot $c'[\epsilon]$. Pour tout $k \in \mathbb{N}$ et $n \in [1,k]$ nous appelons m_k le mot $u^k u'$ et $y_{k,n}$ le mot $c_1^{n-1} c_2 c_3^{k-n}[u'_1]$. Nous allons montrer que si $c_1c_2[u'_1] \neq c_2c_3[u'_1]$ alors $\mathsf{nor}(T) = +\infty$. Si $|c_1c_2[u'_1]| \neq |c_2c_3[u'_1]|$ alors $|c_1| \neq |c_3|$. Pour tout $k \in \mathbb{N}$ et $n \in [1,k]$ la taille de $y_{k,n}$ est de $|c_1|(n-1) + |c_2| + |c_3|(k-n) + |u'_1|$. Pour tout $n' \in [1,k]$ nous avons $|y_{k,n}| - |y_{k,n'}| = n(|c_1| - |c_3|) + n'(|c_3| - |c_1|)$. Si $n \neq n'$ alors $|y_{k,n}| \neq |y_{k,n'}|$. Nous pouvons conclure que $\mathsf{nor}(m_k) \geqslant k$, et comme T est émondé nous en déduisons que $\mathsf{nor}(T) = +\infty$.

Nous considérons maintenant que $|c_1c_2[u'_1]| = |c_2c_3[u'_1]|$. Pour tout $k \in \mathbb{N}$ et $n, n' \in [1,k]$ nous avons donc $|y_{k,n}| = |y_{k,n'}|$.

Il existe un entier m tel que pour tout $k \in \mathbb{N}$ et $n \in [1, k-1]$ avec $k - n > m$ nous avons $y_{k,n} \neq y_{k,n+1}$. En effet en appliquant le Lemme 5.5.9 avec $z = z' = u_3$ et $z = z' = v_3$ nous obtenons que si $y_{k,n} = y_{k,n+1}$ alors $c_1c_2[u'_1] = c_2c_3[u'_1]$, ce qui est une contradiction avec les hypothèses de départ. Nous définissons maintenant deux entiers l_1 et l_2 de la manière suivante :

$$l_1 = \min(\{u_1^{n-1} u_2 u_3^{k-n}, u_1^n u_2 u_3^{k-n-1}\}),$$
$$l_2 = \min(\{u_1^{n-1} u_2 u_3^{k-n} u'_1, u_1^n u_2 u_3^{k-n-1} u'_1\})$$

Nous considérons la plus petite position l telle que $y_{k,n}(l) \neq y_{k,n+1}(l)$, et distinguons trois cas, selon la position de l.

Si $l \leqslant l_1$ nous pouvons noter en premier lieu que $l > |u_1^n|$. Pour tout $n' \in [n+1, k]$, nous avons $y_{k,n'}(l + i|u_1|) \neq y_{k,n'+1}(l + i|u_1|)$ et $y_{k,n}(l) = y_{k,n'+1}(l)$ avec $i = n' - n$. Nous avons $|u_1| \neq 0$, car sinon $y_{k,n}(1, l_1) = y_{k,n+1}(1, l_1) = u_2 u_3^{k-n-1}$, ce qui est une contradiction avec les hypothèses de départ. Nous pouvons en conclure que $\mathsf{nor}(m_k) \geqslant k$ et comme T est émondé nous en déduisons que $\mathsf{nor}(T) = +\infty$.

Si $l_1 < l \leqslant l_2$ pour tout $n' \in [n+1, k]$, nous avons $y_{k,n'}(l + i|u_1| - i|u_3|) \neq y_{k,n'+1}(l + i|u_1| - i|u_3|)$ et donc $y_{k,n}(l) = y_{k,n'+1}(l)$ avec $i = n' - n$. Nous avons $|u_1| \neq |u_3|$, car sinon $y_{k,n}(l_1, l_2) = y_{k,n+1}(l_1, l_2) = u'_1$, ce qui est une contradiction avec les hypothèses de départ. Nous pouvons en conclure que $\mathsf{nor}(m_k) \geqslant k$ et comme T est émondé nous en déduisons que $\mathsf{nor}(T) = +\infty$.

Enfin, si $l > l_2$ alors par symétrie nous obtenons que $\mathsf{nor}(T) = +\infty$. □

Lemme 5.5.14

Soit T un M2C émondé de Σ vers Σ'. Si T satisfait la propriété **(G1)** alors sa norme est infinie.

Démonstration. Nous notons $T = (A, \lambda)$ avec $A = (Q, I, F, \delta)$. Soient $u, v, w, u' \in \Sigma^*$ des mots et $c_{ij} = (u_{ij}, v_{ij}), c' \in \Sigma'^* \times \Sigma'^*$ des contextes choisis comme dans la propriété **(G1)**, pour tout $i, j \in [1,3]$. Nous appelons u'_1 le mot $c'[\epsilon]$. Nous appelons α_i^n le mot $u_{i1} u_{i2}^n u_{i3}$ et β_i^n le mot $v_{i3} v_{i2}^n v_{i1}$ pour tout $i, j \in [1,3]$. Pour tout $n, m, k \in \mathbb{N}$ nous appelons $p_{n,m,k}$ et $p'_{n,m,k}$ les mots suivants :

$$p_{n,m,k} = c_{11} c_{12}^n c_{13} (c_{11} c_{12}^m c_{13})^k c_{21} c_{22}^n c_{23}[u'_1]$$
$$p'_{n,m,k} = c_{21} c_{22}^n c_{23} (c_{31} c_{32}^m c_{33})^k c_{31} c_{32}^n c_{33}[u'_1]$$

Par le Lemme 5.5.13, si T satisfait **(G0)** alors sa norme est infinie. Nous supposons donc que T ne satisfait pas **(G0)** et :

$$p_{n,m,k} = p'_{n,m,k} \tag{5.2}$$

De cette équation nous pouvons déduire que :

$$|c_{11}c_{13}| = |c_{31}c_{33}| \tag{5.3}$$
$$|c_{12}| = |c_{32}| \tag{5.4}$$

Pour tout $k \in \mathbb{N}$ et tout $n \in [1,k]$ nous appelons m_k et $s_{k,n}$ les mots suivants :

$$m_k = \Pi_{i=1}^{k}(uv^{k-i}w)u'$$
$$s_{k,n} = \Pi_{i=1}^{n-1}(c_{11}c_{12}^{k-i}c_{13})c_{21}c_{22}^{k-n}c_{23}\Sigma_{i=n+1}^{k}(c_{31}c_{32}^{k-i}c_{33})[u_1']$$

Enfin, nous appelons $\mathsf{fw}(n,m)$ l'entier $n+m-\mathsf{pgcd}(n,m)$ pour tout $n,m \in \mathbb{N}$ et L l'entier $\max(\{|\lambda(t)| \mid t \in \delta\})$. Nous faisons maintenant la preuve du lemme. Pour cela nous considérons plusieurs cas :

Cas 1	$\|c_{12}\| \neq \|c_{22}\|$					
Cas 2	$\|c_{12}\| = \|c_{22}\|$	$\|u_{11}u_{12}^{2L}u_{13}\| = \|u_{31}u_{32}^{2L}u_{33}\|$	$\|u_{12}\| = \|u_{22}\|$			
Cas 3	〃	〃	$\|u_{12}\| < \|u_{22}\|$	$\|u_{11}u_{12}u_{13}\| \neq 0$		
Cas 4	〃	〃	〃	$\|u_{11}u_{12}u_{13}\| = 0$		
Cas 5	〃	〃	$\|u_{12}\| > \|u_{22}\|$			
Cas 6	〃	$\|u_{11}u_{12}^{2L}u_{13}\| < \|u_{31}u_{32}^{2L}u_{33}\|$	$\|u_{12}\| = \|u_{22}\|$	$\|u_{12}\| = \|u_{32}\|$		
Cas 7	〃	〃	〃	$\|u_{12}\| < \|u_{32}\|$	$\|u_{11}u_{12}u_{13}\| \neq 0$	$\|v_{33}v_{32}v_{31}\| \neq 0$
Cas 8	〃	〃	〃	〃	〃	$\|v_{33}v_{32}v_{31}\| = 0$
Cas 9	〃	〃	〃	〃	$\|u_{11}u_{12}u_{13}\| = 0$	$\|v_{33}v_{32}v_{31}\| \neq 0$
Cas 10	〃	〃	〃	〃	〃	$\|v_{33}v_{32}v_{31}\| = 0$
Cas 11	〃	〃	$\|u_{12}\| < \|u_{22}\|$	$\|u_{11}u_{12}u_{13}\| \neq 0$	$\|v_{33}v_{32}v_{31}\| \neq 0$	
Cas 12	〃	〃	〃	〃	$\|v_{33}v_{32}v_{31}\| = 0$	
Cas 13	〃	〃	〃	$\|u_{11}u_{12}u_{13}\| = 0$	$\|v_{33}v_{32}v_{31}\| \neq 0$	
Cas 14	〃	〃	〃	〃	$\|v_{33}v_{32}v_{31}\| = 0$	
Cas 15	〃	〃	$\|u_{12}\| > \|u_{22}\|$			
Cas 16	〃	$\|u_{11}u_{12}^{2L}u_{13}\| > \|u_{31}u_{32}^{2L}u_{33}\|$				

Cas 1 :

Nous allons montrer que dans ce cas nous pouvons générer un nombre non borné de mots de tailles différentes. Pour tout $k \in \mathbb{N}$ et tout $n \in [1,k]$ nous avons :

$$|s_{k,n}| = |\Sigma_{i=1}^{n-1}(c_{11}c_{12}^{k-i}c_{13})c_{21}c_{22}^{k-n}c_{23}\Sigma_{i=n+1}^{k}(c_{31}c_{32}^{k-i}c_{33})[u_1']|$$

Par les Équations (5.3) et (5.4) nous obtenons :

$$\begin{aligned}|s_{k,n}| &= (k-1)(|c_{11}c_{13}|) + |c_{21}c_{23}| + (\frac{k(k+1)}{2} - (k-n))(|c_{12}|) + (k-n)(|c_{22}|) \\ &= (k-1)(|c_{11}c_{13}|) + |c_{21}c_{23}| + \frac{k(k+1)}{2}(|c_{12}|) + (k-n)(|c_{22}| - |c_{12}|)\end{aligned}$$

Donc pour tout $n' \in [1,k]$ nous avons $|s_{k,n}| - |s_{k,n'}| = (k-n)(|c_{22}| - |c_{12}|) - (k-n')(|c_{22}| - |c_{12}|)$. Comme $|c_{22}| \neq |c_{12}|$ nous obtenons que si $n \neq n'$ alors $|s_{k,n}| \neq |s_{k,n'}|$. Comme T est émondé nous en déduisons que $\mathsf{nor}(T) = +\infty$.

Cas 2 :

Nous allons montrer que ce cas est en contradiction avec le fait que la propriété **(G0)** n'est pas satisfaite. Comme $|c_{12}| = |c_{22}|$, par les Équations (5.3) et (5.4) pour tout $k \in \mathbb{N}$ et $n,n' \in [1,k]$ nous obtenons que :

$$|s_{k,n}| = |s_{k,n'}| \tag{5.5}$$

Nous savons que $s_{2,2} \neq s_{2,1}$ et par l'Équation 5.5 en déduisons qu'il existe une position $l \in \mathbb{N}$ telle que $s_{2,2}(l) \neq s_{2,1}(l)$. Nous définissons les positions suivantes :

$$l_1 = \min\{|\alpha_1^1 u_{21}|, |\alpha_2^1 u_{31}|\},$$
$$l_2 = \max\{|\alpha_1^1 u_{21}|, |\alpha_2^1 u_{31}|\},$$
$$l_3 = \min\{|\alpha_1^1 \alpha_2^0 u_1' v_{23}|, |\alpha_2^1 \alpha_3^0 u_1' v_{33}|\},$$
$$l_4 = \max\{|\alpha_1^1 \alpha_2^0 u_1' v_{23}|, |\alpha_2^1 \alpha_3^0 u_1' v_{33}|\}.$$

Nous distinguons cinq cas dépendants de l, comme représenté à la Figure 5.9 :

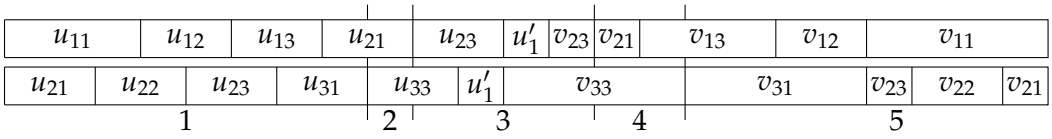


FIGURE 5.9 – $s_{2,2} \neq s_{2,1}$

- Si $l < l_1$ alors nous avons $s_{2,1}(1, l_1) = p'_{1,0,0}(1, l_1)$ et $s_{2,2}(1, l_1) = p_{1,0,0}(1, l_1)$. Comme $p_{1,0,0} \neq p'_{1,0,0}$ nous concluons que c'est une contradiction avec l'Équation (5.2).
- Si $l_1 \leqslant l < l_2$ alors par le Lemme 5.5.11 nous savons que $|u_{12}| = |u_{22}| = |u_{32}|$, nous pouvons donc appliquer le Lemme 5.5.9 (avec $z_1 = u_{22}$, $z_2 = u_{32}$, $u = s_{2,1}(l_1, l_2)$, $v = s_{2,2}(l_1, l_2)$). Il existe donc un entier $n \in \mathbb{N}$ tel que pour tout $i > n$ nous avons $p_{i,0,0} \neq p'_{i,0,0}$, ce qui est une contradiction avec l'Équation (5.2).
- Si $l_2 \leqslant l < l_3$ alors nous avons $s_{2,1}(l_2, l_3) = p'_{1,0,0}(l_2 + |u_{32}|, l_3 + |u_{32}|)$ et $s_{2,2}(l_2, l_3) = p_{1,0,0}(l_2 + |u_{22}|, l_3 + |u_{22}|)$. Comme $|u_{12}| = |u_{22}| = |u_{32}|$ nous concluons que $p_{1,0,0} \neq p'_{1,0,0}$, ce qui est une contradiction avec l'Équation (5.2).
- Si $l_3 \leqslant l < l_4$ alors par le Lemme 5.5.11 nous savons que $|u_{12}| = |u_{22}| = |u_{32}|$, et comme $|c_{12}| = |c_{22}| = |c_{32}|$ nous pouvons donc appliquer le Lemme 5.5.9 (avec $z_1 = v_{22}$, $z_2 = v_{32}$, $u = s_{2,1}(l_3, l_4)$ et $v = s_{2,2}(l_3, l_4)$). Il existe donc un entier $n \in \mathbb{N}$ tel que pour tout $i > n$ nous avons $p_{i,0,0} \neq p'_{i,0,0}$, ce qui est une contradiction avec l'Équation (5.2).
- Si $l_4 \leqslant l$ alors $s_{2,1}(l_4, |s_{2,1}| + 1) = p'_{1,0,0}(l_4 + |u_{32}| + |v_{32}|, |p'_{1,0,0}| + 1)$ et $s_{2,2}(l_4, |s_{2,2}| + 1) = p_{1,0,0}(l_4 + |u_{22}| + |v_{22}|, |p_{1,0,0}| + 1)$. Comme $|u_{12}| = |u_{22}| = |u_{32}|$ et $|c_{12}| = |c_{22}| = |c_{32}|$ nous concluons que $p_{1,0,0} \neq p'_{1,0,0}$, ce qui est une contradiction avec l'Équation (5.2).

Cas 3 :

Ici nous allons utiliser le fait que T ne satisfait pas **(G0)** pour préciser la forme des contextes c_{11}, c_{12}, c_{13}, c_{21}, c_{22}, c_{23}, c_{31}, c_{32}, c_{33} en utilisant le Théorème de Fine et Wilf.

Nous allons commencer par calculer la forme des mots u_{11}, u_{12}, u_{13}, u_{21}, u_{22}, u_{23}, u_{31}, u_{32} et u_{33}. Comme $|u_{11}u_{12}u_{13}| \neq 0$, $|u_{22}| > |u_{12}|$, $|u_{11}u_{12}^{2L}u_{13}| = |u_{31}u_{32}^{2L}u_{33}|$, pour tout $m \geqslant 2L$ il existe des entiers $n, k \in \mathbb{N}$ tels que $|\alpha_2^n| > |\alpha_1^n| + l$, $|(\alpha_1^m)^k| > l$, $|\alpha_2^n| > |\alpha_3^n| + l$ et $|(\alpha_3^m)^k| > l$ avec $l = \mathsf{fw}(|u_{22}|, |\alpha_1^m|)$. De plus par l'Équation 5.2 nous obtenons (voir la Figure 5.10) :

$$\alpha_1^n(\alpha_1^m)^k\alpha_2^n = \alpha_2^n(\alpha_3^m)^k\alpha_3^n$$

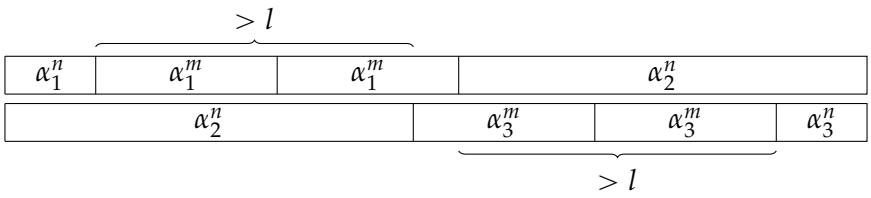


FIGURE 5.10 – $\alpha_1^n(\alpha_1^m)^k\alpha_2^n = \alpha_2^n(\alpha_3^m)^k\alpha_3^n$

Nous pouvons donc appliquer le Théorème 5.5.3 et prouver que pour tout $m \geqslant 2L$ il existe des entiers $a_{1,m}, a_{3,m}, a_2 \in \mathbb{N}_{>0}$ et des mots $x_{m,1}, y_{m,1}, x_{m,2}, y_{m,2} \in \Sigma^*$ tels que :

$$\begin{aligned}
\alpha_1^m &= (x_{m,1}y_{m,1})^{a_{1,m}},\ x_{m,1}y_{m,1} \text{ est la racine primitive de } \alpha_1^m,\\
u_{22} &= (y_{m,1}x_{m,1})^{a_2},\ y_{m,1}x_{m,1} \text{ est la racine primitive de } u_{22},\\
&= (y_{m,2}x_{m,2})^{a_2},\ y_{m,2}x_{m,2} \text{ est la racine primitive de } u_{22},\\
\alpha_3^m &= (x_{m,2}y_{m,2})^{a_{3,m}},\ x_{m,2}y_{m,2} \text{ est la racine primitive de } \alpha_3^m
\end{aligned}$$

Comme pour tout $m' \geqslant 2L$ nous avons $y_{m,1}x_{m,1} = y_{m',1}x_{m',1} = y_{m,2}x_{m,2} = y_{m',2}x_{m',2}$, nous pouvons appliquer le Lemme 5.5.8 et montrer que $x_{m,1}y_{m,1} = x_{0,1}y_{0,1}$ et $x_{m,2}y_{m,2} = x_{0,2}y_{0,2}$. Pour plus de simplicité nous renommons $x_{0,1}$ par x_1, $y_{0,1}$ par y_1, $x_{0,2}$ par x_2 et $y_{0,2}$ par y_2. Nous pouvons ainsi appliquer le Lemme 5.5.2 pour montrer que pour tout $m \geqslant 2L$ il existe des mots $x_3, y_3 \in \Sigma^*$ tels que :

$$\begin{aligned}
&\alpha_1^m \in (x_3y_3)^{a_{1,m}},\ x_3y_3 \text{ est la racine primitive de } \alpha_1^m\\
&\alpha_3^m \in (y_3x_3)^{a_{3,m}},\ y_3x_3 \text{ est la racine primitive de } \alpha_3^m,
\end{aligned}$$

Comme $|u_{11}u_{12}u_{13}| \neq 0$ et $|u_{11}u_{12}^{2L}u_{13}| = |u_{31}u_{32}^{2L}u_{33}|$, pour tout $m \geqslant 2L$ il existe un entier k tel que $|(\alpha_1^m)^k| > |u_{22}^m|$ et $|(\alpha_3^m)^k| > |u_{22}^m|$. De plus, par l'Équation 5.2 nous obtenons :

$$(\alpha_1^m)^k\alpha_2^m = \alpha_2^m(\alpha_3^m)^k$$

Pour tout $m \geqslant 2L$ nous pouvons donc appliquer le Lemme 5.5.7 (avec $v = x_3$, $w = y_3$ et $x = \alpha_2^m$), et montrons qu'il existe un entier $a_{2,m} \in \mathbb{N}_{>0}$ tel que :

$$\alpha_2^m y_3 = (x_3y_3)^{a_{2,m}} \qquad y_3\alpha_2^m = (y_3x_3)^{a_{2,m}}$$

Nous calculons maintenant la forme des mots $v_{11}, v_{12}, v_{13}, v_{21}, v_{22}, v_{23}, v_{31}, v_{32}$ et v_{33}. Notez que comme $|u_{22}| > |u_{12}|$, $|c_{12}| = |c_{22}|$ et $|u_{11}u_{12}^{2L}u_{13}| = |u_{31}u_{32}^{2L}u_{33}|$, par les Équations 5.3 et 5.4 nous obtenons que $|v_{13}v_{12}^{2L}v_{11}| = |v_{33}v_{32}^{2L}v_{31}|$, $|v_{12}| > |v_{22}|$ et $|v_{13}v_{12}v_{11}| \neq 0$. Pour tout $m \geqslant 2L$ il existe donc deux entiers $n, k \in \mathbb{N}$ tels que $|\beta_1^n| > |\beta_2^n| + l$, $|(\beta_3^m)^k| > l$, $|\beta_3^n| > |\beta_2^n| + l$ et $|(\beta_1^m)^k| > l$ avec $l = \mathsf{fw}(v_{12}, \beta_3^m)$. De plus par l'Équation 5.2 nous avons (voir la Figure 5.11) :

$$\beta_3^n(\beta_3^m)^k\beta_2^n = \beta_2^n(\beta_1^m)^k\beta_1^n$$

Nous pouvons donc appliquer le Théorème 5.5.3 et montrer que pour tout $m \geqslant 2L$ il existe des entiers $b_{1,m}, b_{3,m}, b_3, b_1 \in \mathbb{N}_{>0}$ et des mots $x_{m,4}, y_{m,4}, x_{m,5}, y_{m,5} \in \Sigma^*$ tels que :

$$\begin{aligned}
\beta_1^m &= (x_{m,4}y_{m,4})^{b_{1,m}},\ x_{m,4}y_{m,4} \text{ est la racine primitive de } \beta_1^m,\\
v_{32} &= (y_{m,4}x_{m,4})^{b_3},\ y_{m,4}x_{m,4} \text{ est la racine primitive de } v_{32},\\
v_{12} &= (x_{m,5}y_{m,5})^{b_1},\ x_{m,5}y_{m,5} \text{ est la racine primitive de } v_{12},\\
\beta_3^m &= (y_{m,5}x_{m,5})^{b_{3,m}},\ y_{m,5}x_{m,5} \text{ est la racine primitive de } \beta_3^m
\end{aligned}$$

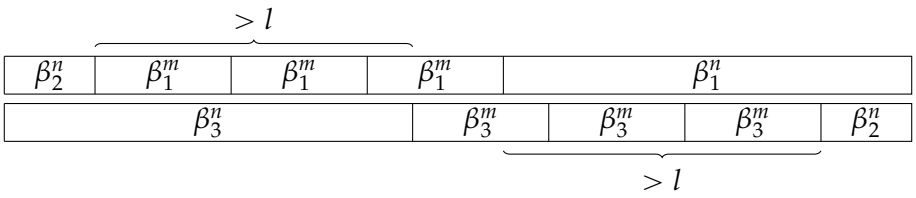


FIGURE 5.11 – $\beta_3^n(\beta_3^m)^k\beta_2^n = \beta_2^n(\beta_1^m)^k\beta_1^n$

Comme pour tout $m' \geqslant 2L$ nous avons $y_{m,4}x_{m,4} = y_{m',4}x_{m',4}$ et $x_{m,5}y_{m,5} = x_{m',5}y_{m',5}$ nous pouvons appliquer le Lemme 5.5.8 et montrer que $x_{m,4}y_{m,4} = x_{0,4}y_{0,4}$ et $y_{m,5}x_{m,5} = y_{0,5}x_{0,5}$. Pour plus de simplicité nous renommons $x_{0,4}$ par x_4, $y_{0,4}$ par y_4, $x_{0,5}$ par x_5 et $y_{0,5}$ par y_5. De plus par $|v_{13}v_{12}v_{11}| \neq 0$ et $|v_{13}v_{12}^{2L}v_{11}| = |v_{33}v_{32}^{2L}v_{31}|$, pour tout $m \geqslant 2L$ il existe un entier $k \in \mathbb{N}$ tel que $|(\beta_1^m)^k| > |\beta_2^m| + l$ et $|(\beta_3^m)^k| > |\beta_2^m| + l$ avec $l = \mathsf{fw}(\beta_1^m, \beta_3^m)$. Par l'Équation 5.2 nous obtenons :

$$(\beta_3^m)^k\beta_2^n = \beta_2^n(\beta_1^m)^k$$

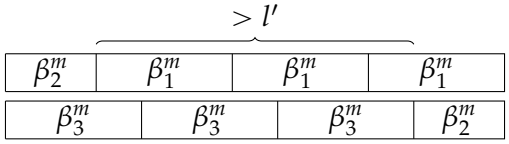


FIGURE 5.12 – $(\beta_3^m)^k\beta_2^n = \beta_2^n(\beta_1^m)^k$

Nous pouvons donc appliquer le Théorème 5.5.3 et montrer qu'il existe des mots $x_6, y_6 \in \Sigma^*$ tels que

$\beta_1^m = (x_6y_6)^{b_{1,m}}$, x_6y_6 est la racine primitive de β_1^m,
$\beta_3^m = (y_6x_6)^{b_{3,m}}$, y_6x_6 est la racine primitive de β_3^m

Comme $|v_{12}| = |v_{32}|$ et $|v_{13}v_{11}| = |v_{33}v_{31}|$ et $|v_{32}| > |v_{22}|$, pour tout entier $m \geqslant 2L$ il existe un entier $k \in \mathbb{N}$ tel que $|(\beta_3^m)^k| > |\beta_2^m| + \beta_1^m$. Nous pouvons donc appliquer le Lemme 5.5.7 (avec $v = x_6$, $w = y_6$ et $x = \beta_2^n$) et montrer que pour tout $m \geqslant 2L$ il existe un entier $b_{2,m} \in \mathbb{N}$ tel que :

$$\beta_2^m x_6 = (y_6x_6)^{b_{2,m}} \qquad x_6\beta_2^m = (x_6y_6)^{b_{2,m}}$$

En utilisant les formes des mots calculées ci-dessus, pour tout $n \in \mathbb{N}$ et $k \in [1, n]$ il existe des entiers $l_n, l'_n \in \mathbb{N}_{>0}$ tels que $s_{k,n} = x_3(y_3x_3)^{l_n}u'_1(y_6x_6)^{l'_n}y_6$. Comme $|u_{12}| = |u_{32}|$ et $|u_{11}u_{13}| = |u_{31}u_{33}|$ nous avons $|x_3(y_3x_3)^{l_n}| - |x_3(y_3x_3)^{l_{n+1}}| = |u_{22}| - |u_{12}|$ et comme $|v_{12}| = |v_{32}|$ et $|v_{13}v_{11}| = |v_{33}v_{31}|$ nous avons $|(y_6x_6)^{l'_n}y_6| - |(y_6x_6)^{l'_{n+1}}y_6| = |v_{12}| - |v_{22}|$. De plus $|c_{12}| = |c_{22}|$, nous en déduisons donc que $|u_{22}| - |u_{12}| = |v_{12}| - |v_{22}|$. Comme $|u_{22}| > |u_{12}|$ et $|v_{12}| > |v_{22}|$ il existe donc un entier $p \in \mathbb{N}_{>0}$ tel que pour tout $n \in \mathbb{N}$ et $k \in [1, n]$ nous avons :

$$s_{k,n} = x_3(y_3x_3)^{l_{n+1}-p}u'_1(y_6x_6)^{l'_{n+1}+p}y_6$$

De plus comme $s_{2,2} \neq s_{2,1}$, nous en déduisons que $s_{k,n} \neq s_{k+1,n}$. Nous pouvons donc appliquer le Lemme 5.5.10 (avec $u = (y_3x_3)^p$, $x = w$ et $v = (y_6x_6)^p$) et montrer que pour tout $k' \in [1, n]$ tel que $k' \neq k$ nous avons $s_{k,n} \neq s_{k',n}$. Nous concluons que pour tout $n \in \mathbb{N}$ nous avons $\mathsf{nor}(m_n) \geqslant n$, et comme T est émondé, $\mathsf{nor}(T) = +\infty$.

Cas 4 :

Nous avons $|u_{11}u_{12}u_{13}| = 0$. Comme $|u_{11}u_{13}| = |u_{31}u_{33}|$, $|u_{12}| = |u_{32}|$ nous en déduisons que $|u_{31}u_{32}u_{33}| = 0$. De plus, similairement au Cas 3 nous avons $|v_{13}v_{11}| = |v_{33}v_{31}|$, $|v_{12}| = |v_{32}|$, $|v_{12}| > |v_{22}|$ et $|v_{13}v_{12}v_{11}| \neq 0$. Nous pouvons donc calculer la forme des mots $v_{11}, v_{12}, v_{13}, v_{21}, v_{22}, v_{23}, v_{31}, v_{32}$ et v_{33} et en déduisons qu'il existe $x_6, y_6 \in \Sigma^*$ tel que pour tout $n \in \mathbb{N}$ et $k \in [1, n]$ il existe un entier $l'_n \in \mathbb{N}_{>0}$ avec $s_{k,n} = u_{21}u_{22}^{k-n}u_{23}u'_1(y_6x_6)^{l'_n}y_6$. Comme $|v_{12}| = |v_{32}|$ et $|v_{13}v_{11}| = |v_{33}v_{31}|$ nous avons $|(y_6x_6)^{l'_n}y_6| - |(y_6x_6)^{l'_{n+1}}y_6| = |v_{12}| - |v_{22}|$. De plus $|c_{12}| = |c_{22}|$ et $|u_{12}| = 0$, nous en déduisons donc que $|u_{22}| = |v_{12}| - |v_{22}|$. Comme $|v_{12}| > |v_{22}|$ il existe donc un entier $p \in \mathbb{N}_{>0}$ tel que pour tout $n \in \mathbb{N}$ et $k \in [1, n]$:

$$s_{k,n} = u_{21}u_{22}^{k-n}u_{23}u'_1(y_6x_6)^{l'_{n+1}-p}y_6$$

De plus comme $s_{2,2} \neq s_{2,1}$, nous en déduisons que $s_{k,n} \neq s_{k+1,n}$. Nous pouvons donc appliquer le Lemme 5.5.10 (avec $u = u_{22}$, $x = u_{23}u'_1$ et $v = (y_6x_6)^p$) et montrer que pour tout $k' \in [1, n]$ tel que $k' \neq k$ nous avons $s_{k,n} \neq s_{k',n}$. Nous pouvons donc conclure que pour tout $n \in \mathbb{N}$ nous avons $\mathsf{nor}(m_n) \geqslant n$, et comme T est émondé, $\mathsf{nor}(T) = +\infty$.

Cas 5 :

La preuve de ce cas suit celles des Cas 3 et 4. En effet, nous avons $|u_{22}| < |u_{12}|$, et donc par $|c_{12}| = |c_{22}|$ nous en concluons que $|v_{22}| > |v_{12}|$. Ce Cas est donc symétrique aux Cas 3 et 4, et il suffit de considérer le même cheminement en échangeant le rôle des mots u_{ij} avec celui des mots $v_{ij'}$ pour tout $i, j \in [1, 3]$ et $j' = 3 - i$.

Cas 6 :

La preuve de ce cas suit celle du Cas 2. En effet, dans cette preuve la seule hypothèse utilisée est $|u_{12}| = |u_{22}| = |u_{32}|$, et le cheminement emprunté est toujours valide ici.

Cas 7 :

Comme $|u_{11}u_{12}^{2L}u_{13}| < |u_{31}u_{32}^{2L}u_{33}|$, $|u_{12}| = |u_{22}|$, $|u_{12}| < |u_{32}|$ et $|c_{12}| = |c_{22}|$, par les Équations 5.3 et 5.4 nous pouvons conclure que $|v_{33}v_{32}^{2L}v_{31}| < |v_{13}v_{12}^{2L}v_{11}|$, $|v_{22}| = |v_{12}|$ et $|v_{32}| < |v_{12}|$. Pour tout $m \geqslant 2L$ il existe donc deux entiers $n, k \in \mathbb{N}$ tels que $|(\alpha_3^m)^k| > |\alpha_1^n(\alpha_1^m)^k u'_1 v u_{23}| + l$ et $|v_{22}^n| > l$ avec $l = \mathsf{fw}(\alpha_3^m, v_{22})$.

Par l'Équation 5.2 nous avons $\alpha_1^n(\alpha_1^m)^k\alpha_2^n[u'_1] = \alpha_2^n(\alpha_3^m)^k\alpha_3^n[u'_1]$, nous pouvons donc appliquer le Théorème 5.5.3 et montrer que pour tout $m \geqslant 2L$ il existe des entiers $a_{3,m}, b_2 \in \mathbb{N}_{>0}$ et des mots $x_{m,4}, y_{m,4} \in \Sigma^*$ tels que :

$$\alpha_3^m = (x_{m,4}y_{m,4})^{a_{3,m}},\ x_{m,4}y_{m,4} \text{ est la racine primitive de } \alpha_3^m,$$
$$v_{22} = (y_{m,4}x_{m,4})^{b_2},\ y_{m,4}x_{m,4} \text{ est la racine primitive de } v_{22}$$

Comme $|u_{11}u_{12}u_{13}| \neq 0$, $|u_{11}u_{12}^{2L}u_{13}| < |u_{31}u_{32}^{2L}u_{33}|$, $|v_{33}v_{32}v_{31}| \neq 0$ et $|v_{33}v_{32}^{2L}v_{31}| < |v_{13}v_{12}^{2L}v_{11}|$ pour tout $m \geqslant 2L$ il existe un entier $k \in \mathbb{N}$ tel que $|(\alpha_1^m)^k| > |\alpha_2^m| + l_1$, $|(\alpha_3^m)^k| > l_1$, $|(\alpha_3^m)^k| > |(\alpha_1^m)^k u'_1| + l_2$, $|(\beta_1^m)^k| > |\beta_2^m| + l_1$ et $|(\beta_3^m)^k| > l_3$ avec $l_1 = \mathsf{fw}|\alpha_1^m|, |\alpha_3^m|)$, $l_2 = \mathsf{fw}(|\beta_1^m|, |\alpha_3^m|)$ et $l_3 = \mathsf{fw}(|\beta_1^m|, |\beta_3^m|)$.

Par l'Équation 5.2 nous avons $(\alpha_1^m)^k\alpha_2^m[u'_1] = \alpha_2^m(\alpha_3^m)^k[u'_1]$ (voir la Figure 5.13), nous pouvons donc appliquer le Lemme 5.5.3 et montrer que pour tout $m \geqslant 2L$ il existe des entiers

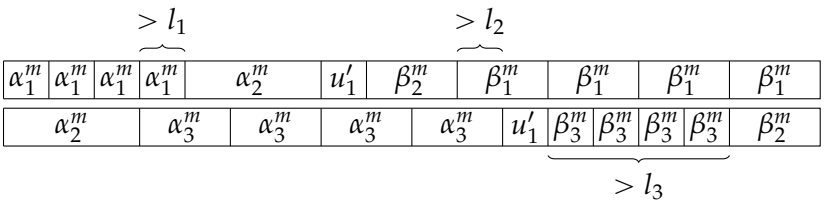


FIGURE 5.13 – $(\alpha_1^m)^k \alpha_2^m[u_1'] = \alpha_2^m(\alpha_3^m)^k[u_1']$

$a_{1,m}, b_{1,m}, b_{3,m} \in \mathbb{N}_{>0}$ et des mots $x_{m,1}$, $y_{m,1}$, $x_{m,2}$, $y_{m,2}$, $x_{m,3}$, $y_{m,3} \in \Sigma^*$ tels que :

$$\begin{aligned}
\alpha_1^m &= (x_{m,1}y_{m,1})^{a_{1,m}},\ x_{m,1}y_{m,1} \text{ est une racine primitive de } \alpha_1^m,\\
\alpha_3^m &= (y_{m,1}x_{m,1})^{a_{3,m}},\ y_{m,1}x_{m,1} \text{ est une racine primitive de } \alpha_3^m\\
&= (y_{m,2}x_{m,2})^{a_{3,m}},\ y_{m,2}x_{m,2} \text{ est une racine primitive de } \alpha_3^m,\\
\beta_1^m &= (x_{m,2}y_{m,2})^{b_{1,m}},\ x_{m,2}y_{m,2} \text{ est une racine primitive de } \beta_1^m\\
&= (x_{m,3}y_{m,3})^{b_{1,m}},\ x_{m,3}y_{m,3} \text{ est une racine primitive de } \beta_1^m,\\
\beta_3^m &= (y_{m,3}x_{m,3})^{b_{3,m}},\ y_{m,3}x_{m,3} \text{ est une racine primitive de } \beta_3^m
\end{aligned}$$

Pour tout $m' \geqslant 2L$ nous avons $y_{m,4}x_{m,4} = y_{m',4}x_{m',4}$ nous pouvons appliquer le Lemme 5.5.8 et montrer que $x_{m,4}y_{m,4} = x_{0,4}y_{0,4}$. Nous pouvons aussi appliquer ce lemme pour montrer que $x_{m,1}y_{m,1} = x_{0,1}y_{0,1}$, $y_{m,1}x_{m,1} = y_{0,1}x_{0,1}$, $x_{m,2}y_{m,2} = x_{0,2}y_{0,2}$, $y_{m,2}x_{m,2} = y_{0,2}x_{0,2}$, $x_{m,3}y_{m,3} = x_{0,3}y_{0,3}$ et $y_{m,3}x_{m,3} = y_{0,3}x_{0,3}$. Pour plus de simplicité nous renommons $x_{0,1}$ as x_1, $y_{0,1}$ par y_1, $x_{0,2}$ par x_2, $y_{0,2}$ par y_2, $x_{0,3}$ par x_3, $y_{0,3}$ et par y_3. Comme $|u_{11}u_{12}u_{13}| \neq 0$ et $|v_{33}v_{32}^{2L}v_{31}| < |v_{13}v_{12}^{2L}v_{11}|$, pour tout $m \in 2L$ il existe un entier $k \in \mathbb{N}$ tel que $|(\alpha_1^m)^k| > |\alpha_2^m|$ et $|(\beta_1^m)^k| > |u_1'(\beta_3^m)^k\beta_2^m|$. Par l'Équation 5.2 nous avons $(\alpha_1^m)^k\alpha_2^m[u_1'] = \alpha_2^m(\alpha_3^m)^k[u_1']$, nous pouvons donc appliquer le Lemme 5.5.7 (avec $v = y_1$, $w = x_1$ et $x = \alpha_2^m$). Pour tout $m \geqslant 2L$ il existe un entier $a_{2,m} \in \mathbb{N}$ tel que :

$$y_1\alpha_2^m = (y_1x_1)^{a_{2,m}} \qquad \alpha_2^m y_1 = (x_1y_1)^{a_{2,m}}$$

Nous appliquons ensuite le même lemme avec $v = x_3$, $w = y_3$ et $x = \beta_2^m$. Pour tout $m \geqslant 2L$ il existe un entier $b_{2,m} \in \mathbb{N}_{>0}$ tel que :

$$x_3\beta_2^m = (x_3y_3)^{b_{2,m}} \qquad \beta_2^m x_3 = (y_3x_3)^{b_{2,m}}$$

Enfin nous utilisons le Lemme 5.5.7 avec $v_1 = y_2$, $w_1 = x_2$, $v_2 = y_3$, $w_2 = x_3$ et $y = u_1'$. Il existe un entier $d \in \mathbb{N}$ tel que :

$$u_1'y_3 = y_2(x_3y_3)^d$$

Finalement nous pouvons donc prouver les égalités suivantes :

$$\begin{aligned}
c_{11}c_{12}c_{13}c_{21}c_{23}[u'_1] &= (x_1y_1)^{a_{1,1}}\alpha_2^0u'_1\beta_2^0(x_3y_3)^{b_{1,1}}\\
&= x_1(y_1x_1)^{a_{1,1}-1}y_1\alpha_2^0u'_1\beta_2^0x_3(y_3x_3)^{b_{1,1}-1}y_3\\
&= x_1(y_1x_1)^{a_{1,1}-1+a_{2,0}}u'_1(y_3x_3)^{b_{1,1}-1+b_{2,0}}y_3\\
&= x_1(y_1x_1)^{a_{1,1}-1+a_{2,0}}u'_1y_3(x_3y_3)^{b_{1,1}-1+b_{2,0}}\\
&= x_1(y_1x_1)^{a_{1,1}-1+a_{2,0}}y_2(x_3y_3)^{b_{1,1}-1+b_{2,0}+d}\\
&= x_1(y_2x_2)^{a_{1,1}-1+a_{2,0}}y_2(x_2y_2)^{b_{1,1}-1+b_{2,0}+d}\\
&= x_1(y_2x_2)^{a_{1,1}-1+a_{2,0}}(y_2x_2)^{b_{1,1}-1+b_{2,0}+d}y_2\\
&= x_1(y_2x_2)^{a_{1,1}-2+a_{2,0}+b_{1,1}+b_{2,0}+d}y_2\\
c_{21}c_{22}c_{23}c_{31}c_{33}[u'_1] &= x_1(y_2x_2)^{a_{2,1}+a_{3,0}-2+b_{3,0}+b_{2,1}+d}y_2
\end{aligned}$$

Comme $|c_{12}| = |c_{22}|$, par les Équations 5.3 et 5.4 nous en déduisons que $|s_{2,1}| = |s_{2,2}|$, et donc $s_{2,1} = s_{2,2}$. C'est une contradiction avec les suppositions de départ.

Cas 8 :

La seule supposition différente du Cas 7 est sur $v_{33}v_{32}v_{31}$, qu'ici nous supposons vide. Il est donc possible de calculer la forme de chacun des mots comme pour le Cas 7, sauf concernant $v_{33}v_{32}v_{31}$. Pour tout $m \geqslant 2L$ il existe donc des entiers $a_{1,m}, b_{1,m}, a_{3,m} \in \mathbb{N}_{>0}$, $a_{2,m} \in \mathbb{N}$ et des mots $x_1, y_1, x_2, y_2 \in \Sigma^*$ tels que :

$$\begin{aligned}
\alpha_1^m &= (x_1y_1)^{a_{1,m}},\ x_1y_1 \text{ est une racine primitive de } \alpha_1^m,\\
\alpha_3^m &= (y_1x_1)^{a_{3,m}},\ y_1x_1 \text{ est une racine primitive de } \alpha_3^m\\
&= (y_2x_2)^{a_{3,m}},\ y_2x_2 \text{ est une racine primitive de } \alpha_3^m,\\
\beta_1^m &= (x_2y_2)^{b_{1,m}},\ x_2y_2 \text{ est une racine primitive de } \beta_1^m
\end{aligned}$$

$$y_1\alpha_2^m = (y_1x_1)^{a_{2,m}} \qquad \alpha_2^m y_1 = (x_1y_1)^{a_{2,m}}$$

Il n'est par contre pas possible de montrer directement la forme des mots $y_1\beta_2^m$ et u'_1y_3 comme dans le Cas 7 car elles dépendent de la forme de β_3^m (qui ici est vide). Il est cependant possible d'appliquer le même raisonnement que pour $y_1\beta_2^m$ afin de calculer la forme de $u'_1\beta_2^m x_2$. Pour tout $m \geqslant 2L$ il existe donc un entier $b_{2,m} \in \mathbb{N}^{>0}$ tel que :

$$x_2u'_1\beta_2^m = (x_2y_2)^{b_{2,m}} \qquad u'_1\beta_2^m x_2 = (y_2x_2)^{b_{2,m}}$$

Nous pouvons donc en déduire que :

$$\begin{aligned}
c_{11}c_{12}c_{13}c_{21}c_{23}[u'_1] &= x_1(y_2x_2)^{a_{1,1}-2+a_{2,0}+b_{1,1}+b_{2,0}}y_2\\
c_{21}c_{22}c_{23}c_{31}c_{33}[u'_1] &= x_1(y_2x_2)^{a_{3,0}-2+a_{2,1}+b_{2,1}}y_2
\end{aligned}$$

Comme $|c_{12}| = |c_{22}|$, par les Équations 5.3 et 5.4 nous en déduisons que $|s_{2,1}| = |s_{2,2}|$, et donc $s_{2,1} = s_{2,2}$. C'est une contradiction avec les suppositions de départ.

Cas 9 :

Encore une fois ce cas est très similaire au Cas 7. Ici la différence est sur le mot $u_{11}u_{12}u_{13}$, que nous supposons vide. Il est donc possible de calculer la forme de chacun des mots comme pour le Cas 7, sauf concernant $u_{11}u_{12}u_{13}$. De plus, comme $|u_{12}| = |u_{22}|$, nous pouvons en déduire que

$|u_{22}| = 0$. Pour tout $m \geqslant 2L$ il existe des entiers $a_{3,m}, b_{1,m}, b_{3,m}, b_{2,m} \in \mathbb{N}_{>0}$, $d \in \mathbb{N}$ et des mots x_2, y_2, x_3, $y_3 \in \Sigma^*$ tels que :

$$\begin{aligned}
\alpha_3^m &= (y_2x_2)^{a_{3,m}}, \, y_2x_2 \text{ est une racine primitive de } \alpha_3^m, \\
\beta_1^m &= (x_2y_2)^{b_{1,m}}, \, x_2y_2 \text{ est une racine primitive de } \beta_1^m \\
&= (x_3y_3)^{b_{1,m}}, \, x_3y_3 \text{ est une racine primitive de } \beta_1^m, \\
\beta_3^m &= (y_3x_3)^{b_{3,m}}, \, y_3x_3 \text{ est une racine primitive de } \beta_3^m
\end{aligned}$$

$$\begin{aligned}
x_3\beta_2^m &= (x_3y_3)^{b_{2,m}} \qquad & \beta_2^m x_3 &= (y_3x_3)^{b_{2,m}} \\
u_1'y_3 &= y_2(x_3y_3)^d \qquad & x_2u_1' &= x_2(y_3x_3)^d
\end{aligned}$$

Nous pouvons donc en déduire que :

$$\begin{aligned}
c_{11}c_{12}c_{13}c_{21}c_{23}[u_1'] &= u_{21}u_{23}(y_2x_2)^{b_{1,1}-1+b_{2,0}}y_2 \\
c_{21}c_{22}c_{23}c_{31}c_{33}[u_1'] &= u_{21}u_{23}(y_2x_2)^{a_{3,0}-1+b_{3,0}+b_{2,1}}y_2
\end{aligned}$$

Comme $|c_{12}| = |c_{22}|$, par les Équations 5.3 et 5.4 nous en déduisons que $|s_{2,1}| = |s_{2,2}|$, et donc $s_{2,1} = s_{2,2}$. C'est une contradiction avec les suppositions de départ.

Cas 10 :

Ce cas est une combinaison du Cas 8 et du Cas 9, c'est à dire que $u_{11}u_{12}u_{13}$ et $v_{33}v_{32}v_{31}$ sont vides. Il est donc possible de calculer la forme de chacun des mots comme pour les cas 8 et 9, sauf concernant $u_{11}v_{12}v_{13}$ et $v_{33}v_{32}v_{31}$. De plus, comme $|u_{12}| = |u_{22}|$, nous pouvons en déduire que $|u_{22}| = 0$. Pour tout $m \geqslant 2L$ il existe donc des entiers $b_{1,m}, a_{3,m}, b_{2,m} \in \mathbb{N}_{>0}$ et des mots x_2, $y_2 \in \Sigma^*$ tels que :

$$\begin{aligned}
\alpha_3^m &= (y_2x_2)^{a_{3,m}}, \, y_2x_2 \text{ est une racine primitive de } \alpha_3^m, \\
\beta_1^m &= (x_2y_2)^{b_{1,m}}, \, x_2y_2 \text{ est une racine primitive de } \beta_1^m
\end{aligned}$$

$$u_1'\beta_2^m x_2 = (y_2x_2)^{b_{2,m}} \qquad x_2u_1'\beta_2^m = (x_2y_2)^{b_{2,m}}$$

Nous pouvons donc en déduire que :

$$\begin{aligned}
c_{11}c_{12}c_{13}c_{21}c_{23}[u_1'] &= u_{21}u_{23}(y_2x_2)^{b_{1,1}-1+b_{2,0}+d}y_2 \\
c_{21}c_{22}c_{23}c_{31}c_{33}[u_1'] &= u_{21}u_{23}(y_2x_2)^{a_{3,0}-1+b_{2,1}+d}y_2
\end{aligned}$$

Comme $|c_{12}| = |c_{22}|$, par les Équations 5.3 et 5.4 nous en déduisons que $|s_{2,1}| = |s_{2,2}|$, et donc $s_{2,1} = s_{2,2}$. C'est une contradiction avec les suppositions de départ.

Cas 11 :

Comme $|u_{11}u_{13}u_{12}| \neq 0$, $|u_{11}u_{12}^{2L}u_{13}| < |u_{31}u_{32}^{2L}u_{33}|$, $|u_{12}| < |u_{22}|$ et $|c_{12}| = |c_{22}|$, par les Équations 5.3 et 5.4 nous pouvons en conclure que $|v_{33}v_{32}v_{31}| \neq 0$, $|v_{33}v_{32}^{2L}v_{31}| < |v_{13}v_{12}^{2L}v_{11}|$ et $|v_{22}| < |v_{12}|$. Pour tout $m \geqslant 2L$ il existe donc deux entiers $n, k \in \mathbb{N}$ tels que $|\alpha_2^n| > |\alpha_1^n| + l_1$, $|(\alpha_1^m)^k| > l_1$, $|\beta_1^n| > |\beta_2^n| + l_2$ et $|(\beta_3^m)^k| > l_2$ avec $l_1 = \mathsf{fw}(|u_{22}|, |\alpha_1^m|)$ et $l_2 = \mathsf{fw}(|v_{12}|, |\beta_3^m|)$.

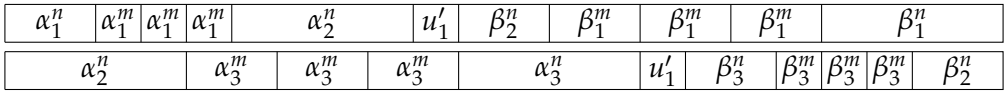


FIGURE 5.14 – $\alpha_1^n(\alpha_1^m)^k\alpha_2^n[u_1'] = \alpha_2^n(\alpha_3^m)^k\alpha_3^n[u_1']$

Par l'Équation 5.2 nous avons $\alpha_1^n(\alpha_1^m)^k\alpha_2^n[u_1'] = \alpha_2^n(\alpha_3^m)^k\alpha_3^n[u_1']$ (voir la Figure 5.14), nous pouvons donc appliquer le Théorème 5.5.3 et montrer qu'il existe $x_{m,4}$, $y_{m,4}$, $x_{m,5}$ et $y_{m,5}$ tels que :

$\alpha_1^m \in (x_{m,4}y_{m,4})^+$, $x_{m,4}y_{m,4}$ est la racine primitive de α_1^m,
$u_{22} \in (y_{m,4}x_{m,4})^+$, $y_{m,4}x_{m,4}$ est la racine primitive de u_{22}
$v_{12} \in (x_{m,5}y_{m,5})^+$, $x_{m,5}y_{m,5}$ est la racine primitive de v_{12}
$\beta_3^m \in (y_{m,5}x_{m,5})^+$, $y_{m,5}x_{m,5}$ est la racine primitive de β_3^m.

Le reste de cette preuve suit la preuve du Cas 7 pour le calcul de la forme des mots α_3^m, β_1^m, u_1', α_2^m et β_2^m.pour tout $m \geqslant 2L$ il existe des entiers $a_{1,m}, b_{1,m}, b_{3,m}, a_{2,m} \in \mathbb{N}_{>0}$, $d, b_{2,m} \in \mathbb{N}$ et des mots $x_{m,1}$, $y_{m,1}$, $x_{m,2}$, $y_{m,2}$, $x_{m,3}$, $y_{m,3} \in \Sigma^*$ tels que :

$\alpha_1^m = (x_{m,1}y_{m,1})^{a_{1,m}}$, $x_{m,1}y_{m,1}$ est une racine primitive de α_1^m,
$\alpha_3^m = (y_{m,1}x_{m,1})^{a_{3,m}}$, $y_{m,1}x_{m,1}$ est une racine primitive de α_3^m
$= (y_{m,2}x_{m,2})^{a_{3,m}}$, $y_{m,2}x_{m,2}$ est une racine primitive de α_3^m,
$\beta_1^m = (x_{m,2}y_{m,2})^{b_{1,m}}$, $x_{m,2}y_{m,2}$ est une racine primitive de β_1^m
$= (x_{m,3}y_{m,3})^{b_{1,m}}$, $x_{m,3}y_{m,3}$ est une racine primitive de β_1^m,
$\beta_3^m = (y_{m,3}x_{m,3})^{b_{3,m}}$, $y_{m,3}x_{m,3}$ est une racine primitive de β_3^m

$$y_1\alpha_2^m = (y_1x_1)^{a_{2,m}} \qquad \alpha_2^m y_1 = (x_1y_1)^{a_{2,m}}$$
$$x_3\beta_2^m = (x_3y_3)^{b_{2,m}} \qquad \beta_2^m x_3 = (y_3x_3)^{b_{2,m}}$$
$$u_1'y_3 = y_2(x_3y_3)^d \qquad x_2u_1' = (x_2y_2)^d x_3$$

Finalement nous pouvons donc prouver les égalités suivantes :

$$c_{11}c_{12}c_{13}c_{21}c_{23}[u_1'] = x_1(y_2x_2)^{a_{1,1}-2+a_{2,0}+b_{1,1}+b_{2,0}+d}y_2$$
$$c_{21}c_{22}c_{23}c_{31}c_{33}[u_1'] = x_1(y_2x_2)^{a_{2,1}+a_{3,0}-2+b_{3,0}+b_{2,1}+d}y_2$$

Comme $|c_{12}| = |c_{22}|$, par les Équations 5.3 et 5.4 nous en déduisons que $|s_{2,1}| = |s_{2,2}|$, et donc $s_{2,1} = s_{2,2}$. C'est une contradiction avec les suppositions de départ.

Cas 12 :

La seule supposition différente du Cas 11 est sur $v_{33}v_{32}v_{31}$, qu'ici nous supposons vide. Il est donc possible de calculer la forme de chacun des mots comme pour le Cas 7, sauf concernant $v_{33}v_{32}v_{31}$. Pour tout $m \geqslant 2L$ il existe donc des entiers $a_{1,m}, b_{1,m}, a_{3,m}, a_{2,m} \in \mathbb{N}_{>0}$ et des mots x_1, y_1, x_2, $y_2 \in \Sigma^*$ tels que :

$\alpha_1^m = (x_1y_1)^{a_{1,m}}$, x_1y_1 est une racine primitive de α_1^m,
$\alpha_3^m = (y_1x_1)^{a_{3,m}}$, y_1x_1 est une racine primitive de α_3^m
$= (y_2x_2)^{a_{3,m}}$, y_2x_2 est une racine primitive de α_3^m,
$\beta_1^m = (x_2y_2)^{b_{1,m}}$, x_2y_2 est une racine primitive de β_1^m

$$y_1\alpha_2^m = (y_1x_1)^{a_{2,m}} \qquad \alpha_2^m y_1 = (x_1y_1)^{a_{2,m}}$$

Il n'est par contre pas possible de montrer directement la forme des mots $y_1\beta_2^m$ et $u_1'y_3$ comme dans le Cas 7 car elles dépendent de la forme de β_3^m (qui ici est vide). Il est cependant possible d'appliquer le même raisonnement que pour $y_1\beta_2^m$ afin de calculer la forme de $u_1'\beta_2^m x_2$. Pour tout $m \geqslant 2L$ il existe donc un entier $b_{2,m} \in \mathbb{N}$ tel que :

$$u_1'\beta_2^m x_2 = (y_2x_2)^{b_{2,m}} \qquad x_2u_1'\beta_2^m = (x_2y_2)^{b_{2,m}}$$

Nous pouvons donc en déduire que :

$$c_{11}c_{12}c_{13}c_{21}c_{23}[u_1'] = x_1(y_2x_2)^{a_{1,1}-2+a_{2,0}+b_{1,1}+b_{2,0}}y_2$$
$$c_{21}c_{22}c_{23}c_{31}c_{33}[u_1'] = x_1(y_2x_2)^{a_{3,0}-2+a_{2,1}+b_{2,1}}y_2$$

Comme $|c_{12}| = |c_{22}|$, par les Équations 5.3 et 5.4 nous en déduisons que $|s_{2,1}| = |s_{2,2}|$, et donc $s_{2,1} = s_{2,2}$. C'est une contradiction avec les suppositions de départ.

Cas 13 :

Encore une fois ce cas est très similaire au Cas 11. Ici la différence est sur le mot $u_{11}u_{12}u_{13}$, que nous supposons vide. Il est donc possible de calculer la forme de chacun des mots comme pour le Cas 11, sauf concernant $u_{11}u_{12}u_{13}$. Pour tout $m \geqslant 2L$ il existe des entiers $a_{3,m}, b_{1,m}, b_{3,m} \in \mathbb{N}_{>0}$, $d, b_{2,m} \in \mathbb{N}$ et des mots $x_2, y_2, x_3, y_3 \in \Sigma^*$ tels que :

$$\begin{aligned}
\alpha_3^m &= (y_2x_2)^{a_{3,m}},\ y_2x_2 \text{ est une racine primitive de } \alpha_3^m,\\
\beta_1^m &= (x_2y_2)^{b_{1,m}},\ x_2y_2 \text{ est une racine primitive de } \beta_1^m\\
&= (x_3y_3)^{b_{1,m}},\ x_3y_3 \text{ est une racine primitive de } \beta_1^m,\\
\beta_3^m &= (y_3x_3)^{b_{3,m}},\ y_3x_3 \text{ est une racine primitive de } \beta_3^m
\end{aligned}$$

$$x_3\beta_2^m = (x_3y_3)^{b_{2,m}} \qquad \beta_2^m x_3 = (y_3x_3)^{b_{2,m}}$$
$$u_1'y_3 = y_2(x_3y_3)^d \qquad x_2u_1' = (x_2y_2)^d x_3$$

Nous en déduisons que pour tout $n \in \mathbb{N}$ et $k \in [1, n]$ il existe un entier $l_n' \in \mathbb{N}_{>0}$ avec $s_{k,n} = u_{21}u_{22}^{k-n}u_{23}u_1'(y_2x_2)^{l_n'}y_2$. Comme nous avons $|s_{k,n}| = |s_{k+1,n}|$ et $|u_{22}| \neq 0$ il existe un entier $p \in \mathbb{N}_{>0}$ tel que pour tout $n \in \mathbb{N}$ et $k \in [1, n]$:

$$s_{k,n} = u_{21}u_{22}^{k-n}u_{23}(y_2x_2)^{l_{n+1}'+p}y_2$$

De plus comme $s_{2,2} \neq s_{2,1}$, nous en déduisons que $s_{k,n} \neq s_{k+1,n}$. Nous pouvons donc appliquer le Lemme 5.5.10 (avec $u = u_{22}$, $x = u_{23}u_1'$ et $v = (y_6x_6)^p$) et montrer que pour tout $k' \in [1, n]$ tel que $k' \neq k$ nous avons $s_{k,n} \neq s_{k',n}$. Nous pouvons donc conclure que pour tout $n \in \mathbb{N}$ nous avons $\mathsf{nor}(m_n) \geqslant n$, et comme T est émondé, $\mathsf{nor}(T) = +\infty$.

Cas 14 :

Ce cas est une combinaison du Cas 12 et du Cas 13, c'est à dire que $u_{11}u_{12}u_{13}$ et $v_{33}v_{32}v_{31}$ sont vides. Il est donc possible de calculer la forme de chacun des mots comme pour les cas 12 et 13, sauf concernant $u_{11}u_{12}u_{13}$ et $v_{33}v_{32}v_{31}$. Pour tout $m \geqslant 2L$ il existe donc des entiers $b_{1,m}, a_{3,m} \in \mathbb{N}_{>0}$, $b_{2,m} \in \mathbb{N}$ et des mots $x_2, y_2 \in \Sigma^*$ tels que :

$$\begin{aligned}
\alpha_3^m &= (y_2x_2)^{a_{3,m}},\ y_2x_2 \text{ est une racine primitive de } \alpha_3^m,\\
\beta_1^m &= (x_2y_2)^{b_{1,m}},\ x_2y_2 \text{ est une racine primitive de } \beta_1^m
\end{aligned}$$

$$u'_1\beta_2^m x_2 = (y_2x_2)^{b_{2,m}} \qquad x_2u'_1\beta_2^m = (x_2y_2)^{b_{2,m}}$$

Nous en déduisons que pour tout $n \in \mathbb{N}$ et $k \in [1,n]$ il existe un entier $l'_n \in \mathbb{N}_{>0}$ avec $s_{k,n} = u_{21}u_{22}^{k-n}u_{23}u'_1(y_2x_2)^{l'_n}y_2$. Comme nous avons $|s_{k,n}| = |s_{k+1,n}|$ et $|u_{22}| \neq 0$ il existe un entier $p \in \mathbb{N}_{>0}$ tel que pour tout $n \in \mathbb{N}$ et $k \in [1,n]$:

$$s_{k,n} = u_{21}u_{22}^{k-n}u_{23}(y_2x_2)^{l'_{n+1}+p}y_2$$

De plus comme $s_{2,2} \neq s_{2,1}$, nous en déduisons que $s_{k,n} \neq s_{k+1,n}$. Nous pouvons donc appliquer le Lemme 5.5.10 (avec $u = u_{22}$, $x = u_{23}u'_1$ et $v = (y_6x_6)^p$) et montrer que pour tout $k' \in [1,n]$ tel que $k' \neq k$ nous avons $s_{k,n} \neq s_{k',n}$. Nous pouvons donc conclure que pour tout $n \in \mathbb{N}$ nous avons $\mathsf{nor}(m_n) \geqslant n$, et comme T est émondé, $\mathsf{nor}(T) = +\infty$.

Cas 15 :

Ce cas suit les lignes des preuves des cas 7 et 8. En effet, comme $|u_{12}| > |u_{22}|$, $|u_{32}| \geqslant |u_{12}|$, $|c_{12}| = |c_{22}| = |c_{32}|$ nous pouvons en déduire que $|v_{22}| > |v_{22}|$. Nous pouvons donc appliquer les mêmes raisonnements afin de calculer la forme des mots v_{22} et α_3^m pour tout $m \geqslant 2L$ puis la forme de tous les autres mots.

Cas 16 :

La preuve de ce cas suit celles des Cas 6, 7, 8, 9, 10, 11, 12, 13 et 14 (quand $|u_{11}u_{12}^{2L}u_{13}| > |u_{31}u_{32}^{2L}u_{33}|$). En effet, nous avons $|u_{11}u_{12}^{2L}u_{13}| < |u_{31}u_{32}^{2L}u_{33}|$, et donc par $|c_{12}| = |c_{22}|$ et $|c_{11}c_{13}| = |c_{31}c_{33}|$ nous en concluons que $|v_{13}v_{12}^{2L}v_{11}| < |v_{33}v_{32}^{2L}v_{31}|$. Ce Cas est donc symétrique aux Ca 6, 7, 8, 9, 10, 11 et 12, 13 et 14. et il suffit de considérer le même cheminement en échangeant le rôle des mots u_{ij} avec celui des mots $v_{ij'}$ pour tout $i,j \in [1,3]$ et $j' = 3-i$. □

5.5.4 Discussion sur la condition nécessaire

Nous avons prouvé que les propriétés **(G0)** et **(G1)** sont des conditions suffisantes à l'infinité de la norme d'un M2C. Nous ne savons pas si elles sont aussi une condition nécessaire, et dans cette section nous allons discuter des différentes idées de preuves pour la conjecture suivante :

Conjecture 5.5.15

> Soit $T = (A, \lambda)$ un M2C émondé. Si T a une norme infinie alors il satisfait l'une des propriétés **(G0)** ou **(G1)**.

Nous allons étudier deux cheminements inspirés de deux résultats existants : celui présenté dans les Sections 5.1 et 5.2, et celui présenté dans la Section 5.3. Le premier, utilisé par Andreas Weber et Helmut Seidl, est basé sur une décomposition des calculs acceptants des automates en utilisant une notion de composante connexe. Le deuxième proposé par Rodrigo de Souza et Jacques Sakarovitch quelques années plus tard utilise des notions de délais. Nous allons montrer dans les sous-sections suivantes comment nous avons essayé d'adapter ces deux approches aux transducteurs mot-vers-contexte.

Décomposition des calculs acceptants Dans le Lemme 5.1.8 de la Section 5.1 nous avons présenté une manière de décomposer les calculs acceptants d'un transducteur fini en utilisant des composantes connexes. Un transducteur mot-vers-contexte étant aussi basé sur un automate fini, il est possible d'appliquer cette décomposition de manière identique. Il nous faut ensuite adapter les Lemmes 5.1.9 et 5.1.10 permettant de borner le nombre de sorties aux M2C. Il existe

une différence majeure entre ce modèle et les TF : un M2C produit des contextes, et un TF des mots. Pour tous mots u et v, si u est différent de v alors pour tout mot w, uw et vw seront aussi différents. Cette propriété n'est pas satisfaite dans le cas des contextes. Nous pouvons avoir deux contextes c et c' différents et un mot w tel que $c[w]$ est égal à $c'[w]$. C'est la raison pour laquelle nous devons considérer la fin du calcul pour les propriétés **(G0)** et **(G1)** et pas pour les propriétés **(W1)** et **(W2)**. Nous devons donc prendre en compte cette différence pour adapter les Lemmes 5.1.9 et 5.1.10. Comme pour les TF, nous devons introduire une notation pour raisonner sur les sorties.

Définition 5.5.16

Soit $T = (A, \lambda)$ un M2C sur l'alphabet Σ vers Σ' avec $A = (Q, I, F, \delta)$. Pour tout $p, q \in Q$ et $u, v \in \Sigma^*$ nous appelons $T_{p,q}(u, v)$ et $T_p(u)$ les ensembles et $\mathsf{nor}_{p,q}(u, v)$ et $\mathsf{nor}_p(u)$ les entiers définis comme suit :

$$
\begin{aligned}
T_{p,q}(u, v) &= \{cc'[\epsilon] \in \Sigma'^* \mid p \xrightarrow{u,c}{}^* q \xrightarrow{v,c'}{}^* q_f \in \mathsf{Cal}_{uv}(A), q_f \in F\} \\
T_p(u) &= \bigcup_{p' \in Q} (T_{p,p'}(u, \epsilon)) \\
\mathsf{nor}_{p,q}(u, v) &= \#T_{p,q}(u, v) \\
\mathsf{nor}_p(u) &= \#T_p(u, \epsilon)
\end{aligned}
$$

Notons que la définition de nor diffère de celle utilisée pour les TF car nous devons ici prendre en compte la fin du calcul. Nous adaptons maintenant le Lemme 5.1.9 aux transducteurs mot-vers-contexte, en utilisant les définitions que nous venons de décrire :

Lemme 5.5.17

Soit $T = (A, \lambda)$ un M2C sur l'alphabet Σ ne satisfaisant pas **(G0)** avec $A = (Q, I, F, \delta)$ Pour tous états $p, q \in Q$ et mots $u, v \in \Sigma^*$ tels que $q \in Q_p$ nous avons $\mathsf{nor}_{p,q}(u, v) \leqslant \mathsf{nor}_q(v)$.

La preuve de ce lemme est presque identique à celle du Lemme 5.1.9. Premièrement comme p et q sont dans la même composante connexe, nous pouvons en déduire qu'il existe un calcul $q \xrightarrow{w,c_3}{}^* p \in \mathsf{Cal}_w(A)$. Nous supposons maintenant que $\mathsf{nor}_{p,q}(u, v) > \mathsf{nor}_q(v)$. Il existe donc trois calculs $p \xrightarrow{u,c_1}{}^* q, p \xrightarrow{u,c_2}{}^* q \in \mathsf{Cal}_u(A)$, $q \xrightarrow{v,c}{}^* q_f \in \mathsf{Cal}_v(A)$ avec $c_1c[\epsilon] \neq c_2c[\epsilon]$ et $q_f \in F$. Nous pouvons en conclure que T satisfait **(G0)** car $(c_3c_1)(c_3c_1)c[\epsilon] \neq (c_3c_1)(c_3c_2)c[\epsilon]$, ce qui est en contradiction avec l'hypothèse de départ. Ce lemme implique que le nombre de mots produits entre deux états dans la même composante connexe est borné par le nombre de mots produits par la suite.

L'étape suivante consiste à adapter le Lemme 5.1.10, mais nous ne sommes pas parvenus à le faire pour l'instant. Nous présentons donc une conjecture et discutons de sa démonstration.

Conjecture 5.5.18

Soit $T = (A, \lambda)$ un M2C sur l'alphabet Σ ne satisfaisant pas **(G0)** et **(G1)** avec $A = (Q, I, F, \delta)$. Il existe un polynôme P tel que pour tout $u, v \in \Sigma^*$ et $B \subseteq Q$ avec

(1) $\forall p \in B, \exists q \in B, \exists p \xrightarrow{u|c} q \in \mathsf{Cal}_u(A)$,

(2) $\forall q \in B, \exists p \in B, \exists p \xrightarrow{u|c} q \in \mathsf{Cal}_u(A)$,

(3) $\forall p \in B, \exists q_f \in F, \exists p \xrightarrow{u|c} q_f \in \mathsf{Cal}_v(A)$,

nous avons : $\mathsf{nor}_{p,q}(u, v) \leqslant \mathsf{nor}_q(v) \cdot 2^{2^{P(|A|)}}$ pour tout $p, q \in B$.

Nous expliquons à présent pourquoi nous ne sommes pas parvenus pour l'instant à démontrer cette conjecture. Nous commençons tout d'abord par extraire des calculs de B, comme nous l'avons fait dans le début de la démonstration du Lemme 5.1.10. Il existe donc des entiers $i, j \in \mathbb{N}$ et des états $p_1, p_2 \in B$ et $q_f, q'_f \in F$ tels que :

- $\eta_1 = p_1 \xrightarrow{u^{i+j+1}|c_1} p_1$
- $\eta_2 = p_1 \xrightarrow{u^i|c_2} p$
- $\eta'_2 = q \xrightarrow{u^j|c_3} p_2$
- $\eta_3 = p_2 \xrightarrow{u^{i+j+1}|c_4} p_2$
- $\eta = p_2 \xrightarrow{v|c} q'_f$

Il est ensuite possible de définir la fonction noyau de la même manière que pour les TF. Nous considérons maintenant les calculs extraits de B, et supposons qu'il existe deux calculs $\hat{\eta} : p \xrightarrow{u|c_5} q, \hat{\eta}' : p \xrightarrow{u|c_6} q \in \mathsf{Cal}_u(A)$. En utilisant des techniques de combinatoire, nous pensons qu'il est possible de montrer que si $\pi_1 = (\eta_1, \eta_2\hat{\eta}\eta'_2, \eta_3)$ et $\pi_2 = (\eta_1, \eta_2\hat{\eta}'\eta'_2, \eta_3)$ ont le même noyau alors $c_2c_5c_3c[\epsilon] = c_2c_6c_3c[\epsilon]$. Intuitivement, comme **(G0)** et **(G1)** ne sont pas satisfaits les contextes produits par les boucles supprimées sont conjugués avec $\mathsf{sortie}(\eta_2\hat{\eta}\eta'_2)$.

Nous ne sommes cependant pas parvenus à déduire de l'égalité $c_2c_5c_3c[\epsilon] = c_2c_6c_3c[\epsilon]$ que pour tout calcul $\eta' = q \xrightarrow{v|c'} q'_f$ si π_1 et π_2 ont le même noyau alors alors nous avons $c_5c'[\epsilon] = c_6c'[\epsilon]$.

Utilisation de délais Nous considérons la technique utilisée dans [SdS08].

Pour suivre ce cheminement, nous devons définir en premier une notion de délais pour contextes. Un solution simple est d'avoir un délai pour le mot de gauche, et un délai pour le mot de droite. Plus formellement, étant donnés $c = (u, v)$ et $c' = (u', v')$ deux contextes de $\Sigma^* \times \Sigma^*$, le *délai* entre c et c' est l'élément de $H_\Sigma \times H_\Sigma$ dénoté par $\mathsf{delai}(c', c)$ et défini par $(\mathsf{delai}(u, u'), \mathsf{delai}(\bar{v}, \bar{v}'))$ avec $\bar{v}$ le miroir du mot v et $\bar{v}'$ le miroir du mot v'. Nous devons considérer les miroirs de v et v' car lorsque deux contextes sont concaténés, le deuxième est ajouté au milieu du premier. Par exemple, le délai entre les contextes (a, aa) et (aa, a) est $((\epsilon, a), (a, \epsilon))$.

Nous devons ensuite définir une notion de taille de délai. Dans le cas des délais de mots, la notion naturelle de taille de délais est égale à la différence entre les tailles des deux mots ayant générés le délai. Dans le cas des contextes de délais, nous pourrions dire que la taille d'un délai $((u, v), (u', v'))$ est égal à $\mid |u| - |v| + |u'| - |v'| \mid$. Par exemple le délai $((\epsilon, a), (a, \epsilon))$ aurait une

taille de 0, ceci correspondant bien avec la différence de taille entre les contextes (a, aa) et (aa, a). Étant donnés deux calculs ρ et ρ' d'un M2C nous pouvons ensuite définir le décalage $\langle \rho, \rho' \rangle$ de la même manière que pour les TF en utilisant cette notion de taille.

L'étape suivante consiste à appliquer la procédure $\mathsf{separe}_N(T)$ afin de construire un transducteur où les sorties produites sur un même mot d'entrée ont un décalage supérieur à N. Pour cela nous devons stocker dans l'état les délais avec une taille de N ou moins. Nous ne pouvons pas faire cela dans le cas des délais de contexte, car nous avons un nombre infini de délais avec une taille inférieure à N. Par exemple, pour tout $i \in \mathbb{N}$ les délais $((a^i, \epsilon), (\epsilon, a^i))$ sont tous de taille 0.

Si nous souhaitons appliquer le cheminement de Rodrigo de Souza et Jacques Sakarovitch nous devons donc trouver une notion de délai permettant d'exprimer la complexité des sorties. L'un des chemins possibles serait de considérer les boucles empruntées par les calculs : quand un délai est long, cela implique que les calculs sous-jacents empruntent des boucles, et les sorties sont composées de motifs se répétant. Il semble donc possible de définir une notion de taille de délai en prenant en compte ces boucles. Nous ne sommes cependant pas encore parvenus à prouver le résultat souhaité en utilisant cette idée.

Conclusion

Dans ce manuscrit, nous avons présenté plusieurs résultats liés à l'étude des automates à pile visible.

Contributions

Tout d'abord, nous avons présenté une nouvelle notion d'émondage pour les automates à pile visible et un algorithme pour émonder un automate arbitraire de ce modèle en temps polynomial. Cette procédure se décompose en plusieurs parties, la réduction, la co-réduction, indépendamment utilisables afin de simplement réduire ou co-réduire un APV. Nous avons ensuite donné une procédure de complexité exponentielle qui, étant donné un automate à pile visible, construit un automate équivalent à la fois émondé et déterministe. Nous avons démontré également que cette complexité exponentielle est optimale. Nous avons ensuite étendu nos différentes constructions d'émondage aux automates à pile visible à coûts.

Dans un deuxième temps, nous nous sommes intéressés aux automates à pile visible à coûts, et avons considéré deux semi-anneaux, celui des entiers naturels et le semi-anneau Max-plus. Nous avons défini des propriétés caractérisant le coût infini, et des algorithmes s'exécutant en temps polynomial pour décider ce problème. Dans le cas des entiers, nous avons aussi prouvé que décider si le coût d'un automate à pile visible à coût est inférieur à un entier donné en entrée est EXPTIME-complet, et peut-être décidé en temps polynomial si l'entier est fixé.

Enfin, nous avons étudié le problème de la norme infinie pour la classe des transducteurs à pile visible linéaires. Nous avons présenté deux propriétés **(G0)** et **(G1)**, et avons montré que si elles sont présentes alors la norme du transducteur est infinie. Puis nous avons traité de pistes potentielles se basant sur les résultats des transducteurs finis et d'arbres pour montrer l'implication réciproque.

Perspectives

Le Chapitre 5 présente un résultat inachevé et plusieurs conjectures concernant l'infinité de la norme d'un transducteur à pile visible linéaire. L'une des perspectives immédiates de ce manuscrit est de montrer que ce résultat est correct, puis en déduire un algorithme pour décider si la norme d'un tel transducteur est infinie.

Il serait ensuite intéressant d'étendre les propriétés **(G0)** et **(G1)** aux transducteurs à pile visibles généraux, et montrer qu'elles caractérisent aussi leur norme infinie. Cela passerait probablement par l'introduction d'une troisième propriété permettant de maîtriser à la fois la verticalité et l'horizontalité induite par la pile. Cette propriété s'inspirerait de la caractérisation proposée par Helmut Seidl concernant les transducteurs d'arbres.

Ce n'est cependant qu'un premier pas vers l'étude des transducteurs à pile visible. Comme montré dans [FRR+10] nous savons déjà résoudre les problèmes de la k-norme et de la fonctionnalité. Les méthodes utilisées pour montrer la décidabilité de la norme finie pourraient

peut-être ensuite être utilisées pour décomposer un transducteur avec une norme de k en k transducteurs fonctionnels, et dont la transformation définie par l'union est égale à celle du transducteur original. En effet, Rodrigo de Souza et Jacques Sakarovitch utilisent dans [SdS08] des délais pour montrer ce résultat, et nous pourrions peut-être faire la même chose dans notre cas. Dans le cas des transducteurs finis ce résultat permet de montrer que l'équivalence entre deux transducteurs finis de norme bornée est décidable. Nous pourrions probablement obtenir un résultat similaire pour les transducteurs à pile visible. Enfin, toujours dans le cas des transducteurs finis, il existe une propriété des transducteurs fonctionnels utilisant des délais caractérisant ceux déterminisables, présentée par exemple dans [BCPS03]. Encore une fois si nous étions capable de définir une notion de délais sur des contextes de mots, nous pourrions probablement adapter ce résultat aux transducteurs à pile visible.

Bibliographie

[AM04] Rajeev Alur and P. Madhusudan. Visibly pushdown languages. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 202–211, 2004.

[AM09] Rajeev Alur and P. Madhusudan. Adding nesting structure to words. *J. ACM*, 56(3), 2009.

[Ama14] Bernd Amann. XSLT. In *Encyclopedia of Social Network Analysis and Mining*, pages 2432–2435. Springer-Verlag New York, 2014.

[BB79] Jean Berstel and Luc Boasson. Transductions and context-free languages. *Ed. Teubner*, pages 1–278, 1979.

[BCPS03] Marie-Pierre Béal, Olivier Carton, Christophe Prieur, and Jacques Sakarovitch. Squaring transducers : an efficient procedure for deciding functionality and sequentiality. *Theor. Comput. Sci.*, 292(1) :45–63, 2003.

[Bea84] Danièle Beauquier. Some results about finite and infinite behaviours of a pushdown automaton. In Jan Paredaens, editor, *ICALP*, volume 172 of *Lecture Notes in Computer Science*, pages 187–195. Springer, 1984.

[BFGM05] Björn Borchardt, Zoltán Fülöp, Zsolt Gazdag, and Andreas Maletti. Bounds for tree automata with polynomial costs. *Journal of Automata, Languages and Combinatorics*, 10(2/3) :107–157, 2005.

[BH77] Meera Blattner and Tom Head. Single-valued a-transducers. *Journal of Computer and System Sciences*, 15(3) :310 – 327, 1977.

[BK03] Jean Berstel and Juhani Karhumäki. Combinatorics on words : a tutorial. *Bulletin of the EATCS*, 79 :178, 2003.

[BR82] Jean Berstel and Christophe Reutenauer. Recognizable formal power series on trees. *Theor. Comput. Sci.*, 18 :115–148, 1982.

[BR88] J. Berstel and C. Reutenauer. *Rational series and their languages*. EATCS monographs on theoretical computer science. Springer-Verlag, 1988.

[Bü60] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6) :66–92, 1960.

[CDG+07] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on : `http://www.grappa.univ-lille3.fr/tata`, 2007. release October, 12th 2007.

[CFR+13] Mathieu Caralp, Emmanuel Filiot, Pierre-Alain Reynier, Frédéric Servais, and Jean-Marc Talbot. Expressiveness of visibly pushdown transducers. In *Proceedings Second International Workshop on Trends in Tree Automata and Tree Transducers, TTATT 2013, Hanoi, Vietnam, 19/10/2013.*, pages 17–26, 2013.

[CGP01] Edmund M. Clarke, Orna Grumberg, and Doron Peled. *Model checking*. MIT Press, 2001.

[Cho62] Noam Chomsky. Context free grammars and pushdown storage. *MIT Research Laboratory in Electronics*, 65 :187 ?194, 1962.

[CI83] Tat-hung Chan and Oscar H. Ibarra. On the finite-valuedness problem for sequential machines. *Theor. Comput. Sci.*, 23 :95–101, 1983.

[Don65] John Doner. Decidability of the weak second-order theory of the two successors. *Notices of the American Mathematical Society*, 12, 1965.

[DS08] Rodrigo De Souza. *Étude structurelle des transducteurs de norme bornée*. PhD thesis, ENST, France, 2008.

[Eil74] Samuel Eilenberg. *Automata, Languages, and Machines*, volume vol. A. Academic Press, Inc., 1974.

[Eil76] Samuel Eilenberg. *Automata, Languages, and Machines*, volume vol. B. Academic Press, Inc., 1976.

[Elg61] Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98(1) :pp. 21–51, 1961.

[FRR+10] Emmanuel Filiot, Jean-François Raskin, Pierre-Alain Reynier, Frédéric Servais, and Jean-Marc Talbot. Properties of visibly pushdown transducers. In *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings*, pages 355–367, 2010.

[FW65] N. J. Fine and H. S. Wilf. Uniqueness theorems for periodic functions. *Proceedings of the American Mathematical Society*, 16(1) :pp. 109–114, 1965.

[Gin66] S. Ginsburg. *The Mathematical Theory of Context-Free Languages*. McGraw-Hill Book Company, 1966.

[GS84] Ferenc Gécseg and Magnus Steinby. *Tree Automata*. Akadéniai Kiadó, Budapest, Hungary, 1984.

[HMU03] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation - international edition (2. ed)*. Addison-Wesley, 2003.

[HWW] Philippe Le Hégaret, Ray Whitmer, and Lauren Wood. W3C : document oject model (DOM). `http://www.w3.org/DOM/`.

[Kle56] S. C. Kleene. Representation of events in nerve nets and finite automata. *Automata Studies*, 1956.

[MB02] W.Scott Means and Michael A. Bodie. The book of sax, 2002.

[MS77] Arnaldo Mandel and Imre Simon. On finite semigroups of matrices. *Theor. Comput. Sci.*, 5(2) :101–111, 1977.

[Rou68] William Chesley Rounds. *Trees, transducers and transformations*. PhD thesis, Stanford University, 1968.

[RS59] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2) :114–125, April 1959.

[RS97] Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of Formal Languages, Vol. 1 : Word, Language, Grammar*. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[RS08] Jean-François Raskin and Frédéric Servais. Visibly pushdown transducers. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B : Logic, Semantics, and Theory of Programming & Track C : Security and Cryptography Foundations*, pages 386–397, 2008.

[Sak09] Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

[Sch61] Marcel-Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4, 1961.

[Sch76] M.P. Schützenberger. Sur les relations rationnelles entre monoides libres. *Theoretical Computer Science*, 3(2) :243 – 259, 1976.

[Sch07] Thomas Schwentick. Automata for xml—a survey. *Journal of Computer and System Sciences*, 73(3) :289 – 315, 2007. Special Issue : Database Theory 2004.

[SdS08] Jacques Sakarovitch and Rodrigo de Souza. On the decidability of bounded valuedness for transducers. In *Mathematical Foundations of Computer Science 2008, 33rd International Symposium, MFCS 2008, Torun, Poland, August 25-29, 2008, Proceedings*, pages 588–600, 2008.

[Sei89] Helmut Seidl. On the finite degree of ambiguity of finite tree automata. *Acta Inf.*, 26(6) :527–542, 1989.

[Sei90] Helmut Seidl. Deciding equivalence of finite tree automata. *SIAM J. Comput.*, 19(3) :424–437, 1990.

[Sei92] Helmut Seidl. Single-valuedness of tree transducers is decidable in polynomial time. *Theor. Comput. Sci.*, 106(1) :135–181, 1992.

[Sei94a] Helmut Seidl. Equivalence of finite-valued tree transducers is decidable. *Mathematical Systems Theory*, 27(4) :285–346, 1994.

[Sei94b] Helmut Seidl. Finite tree automata with cost functions. *Theor. Comput. Sci.*, 126(1) :113–142, 1994.

[Ser11] Frédéric Servais. *Visibly Pushdown Transducers*. PhD thesis, Université Libre de Bruxelles, 2011.

[SS78] Arto Salomaa and Matti Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Texts and Monographs in Computer Science. Springer, 1978.

[Tha65] James W. Thatcher. Generalized finite automata. *Notices of the American Mathematical Society*, 12, 1965.

[Tha70] James W. Thatcher. Generalized2 sequential machine maps. *Journal of Computer and System Sciences*, 4(4) :339 – 367, 1970.

[vBV83] Burchard von Braunmühl and Rutger Verbeek. Input-driven languages are recognized in log n space. In *Fundamentals of Computation Theory, Proceedings of the 1983 International FCT-Conference, Borgholm, Sweden, August 21-27, 1983*, pages 40–51, 1983.

[Web90] Andreas Weber. On the valuedness of finite transducers. *Acta Inf.*, 27(8) :749–780, 1990.

[Web93] Andreas Weber. Decomposing finite-valued transducers and deciding their equivalence. *SIAM J. Comput.*, 22(1) :175–202, 1993.