Contributions à l'étude de la dérivation des expressions rationnelles et à l'étude des systèmes de numération abstraits

Pierre-Yves Angrand

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2012-ENST-009

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Informatique »

présentée et soutenue publiquement par

Pierre-Yves ANGRAND

le 08 mars 2012

Contributions à l'étude de la dérivation des expressions rationnelles et à l'étude des systèmes de numération abstraits

Directeur de thèse : **Jacques SAKAROVITCH**

Jury

M. Patrick BELLOT, Professeur, LTCI, Télécom ParisTech	Président
M. Jean-Marc CHAMPARNAUD, Professeur, LITIS, Université de Rouen	Rapporteur
M. Christian CHOFFRUT, Professeur, LIAFA, Université Paris 7	Rapporteur
M. Sylvain LOMBARDY, Professeur, IGM, Université Paris-Est Marne-la-Vallée	Examinateur
M. Michel RIGO, Professeur, Institute of Math., Université de Liège	Examinateur
M. Jacques SAKAROVITCH, Professeur, LTCI, Télécom ParisTech	Directeur de thèse

TELECOM ParisTech
école de l'Institut Télécom - membre de ParisTech

THÈSE

Remerciements

En préambule de ce mémoire, je tiens à remercier toutes les personnes qui m'ont apportées leur aide et m'ont soutenues pendant toutes ces années. Je souhaite particulièrement remercier mon directeur de thèse, Jacques Sakarovitch pour sa disponibilité, ses conseils et sa patience.

Je souhaite également remercier sincérement Messieurs Champarnaud et Choffrut pour avoir accepté de rapporter ce mémoire. Leurs conseils et corrections ont grandement permis d'améliorer ce dernier.

Je remercie également Messieurs Lombardy et Rigo d'avoir bien voulu participer à mon jury. Leur présence est d'autant plus appréciée que je reprends dans ce mémoire certains de leurs résultats.

Je remercie Monsieur Bellot de s'être intéressé à mon travail et d'avoir accepté de participer à mon jury.

Je remercie également Monsieur Maître, directeur du LTCI et directeur adjoint de l'EDITE, à la fois pour m'avoir acceuilli dans son laboratoire et pour avoir accepté de me donner une année supplémentaire pour finir mon travail.

Je tiens également à exprimer mes remerciements à Hayette Soussou et à tous les membres administratifs de télécom ParisTech qui ont pu m'aider au cours de mes multiples démarches administratives pendant ces années.

Mes derniers remerciements iront finalement à mes proches, mes amis et ma famille pour tous les bons moments passés qui ont permis de rendre le travail plus supportable. Je pense plus particulièrement à mes compagnons de ski, mes camarades de week-ends bretons et de Möllky, mes compères rollistes ainsi que bien évidemment les locataires de la rue Bobillot, de la rue de Crimée, de la rue de l'Ourcq, de la rue Dardenne et de l'avenue du Maine sans qui rien n'aurait été possible.

Table des matières

I Les termes dérivés cassés d'une expression rationnelle 16

1 Notations et préliminaires 17
1.1 Mots et langages et automates 17
1.2 Expressions rationnelles 18
1.3 Automate des positions 21
1.4 Forme normale 23
1.5 Multiplicités 27
1.5.1 Séries formelles 27
1.5.2 $\mathbb{K}$-automates et $\mathbb{K}$-représentations 27

2 Termes dérivés 30
2.1 Dérivation des expressions 31
2.1.1 Expression dérivée d'une expression 31
2.1.2 Nombre d'expressions dérivées 33
2.1.3 Automates des expressions dérivées 33
2.2 Termes dérivés d'une expression 34
2.2.1 Définition et propriétés 34
2.2.2 Automate des termes dérivés 40
2.3 Dérivation et associativité du produit 41
2.3.1 Termes dérivés 42
2.3.2 Expressions dérivées 43

3 Termes dérivés cassés 45
3.1 Définitions et propriétés 46
3.1.1 Cassage d'une expression 46
3.1.2 Termes dérivés cassés 47
3.1.3 Automate des termes dérivés cassés 50
3.1.4 Associativité du produit 51
3.2 Taille de l'automate des termes dérivés cassés 52

3.2.1 Dans le cas général . . . 52
3.2.2 Pour les expressions en forme normale . . . 55
3.2.3 Comparaison avec les termes dérivés . . . 57

4 Construction des automates des termes dérivés et termes dérivés cassés 59
4.1 Concaténation à droite de sous-expressions . . . 61
4.1.1 Sous-expressions sur l'arbre syntaxique . . . 61
4.1.2 Représentation canonique . . . 65
4.2 Calcul de $\mathcal{D}(\mathsf{E})$ et $\mathcal{D}_b(\mathsf{E})$ . . . 70
4.2.1 Calculs sur l'arbre syntaxique décoré . . . 70
4.2.2 Construction de l'automate des termes dérivés . . . 75
4.2.3 Termes dérivés cassés . . . 77

5 Multiplicités 87
5.1 $\mathbb{K}$-expressions rationnelles . . . 88
5.1.1 Définitions et notations . . . 88
5.1.2 Représentation canonique . . . 95
5.2 Termes dérivés . . . 99
5.2.1 Définition . . . 99
5.2.2 Calcul de l'automate . . . 104
5.3 Termes dérivés cassés . . . 108
5.3.1 Définition . . . 108
5.3.2 Calcul de l'automate . . . 112

II Enumération dans les langages rationnels 115

6 Systèmes de numération abstraits et fonctions rationnelles 116
6.1 Systèmes de numération abstraits . . . 117
6.1.1 Définition . . . 117
6.1.2 Enumération . . . 119
6.2 Relations rationnelles . . . 119
6.2.1 Relations et fonctions rationnelles . . . 119
6.2.2 Transducteurs finis . . . 120
6.2.3 Fonction successeur . . . 121

7 Fonctions séquentielles par morceaux 122
7.1 Fonctions séquentielles . . . 123
7.1.1 Séquentialité . . . 123

7.1.2 Caractérisation 124
7.1.3 Décidabilité 127
7.2 Fonctions séquentielles par morceaux 128
7.2.1 Définition 129
7.2.2 Cascades de transducteurs séquentiels 130
7.2.3 coSeqpm et Seqpm 132
7.2.4 Décidabilité 134

8 Fonction successeur de langages rationnels 141
8.1 Produit synchronisé d'automates monocyles 142
8.1.1 Définitions 142
8.1.2 Séquentialité du produit de deux automates monocycles 146
8.1.3 Les langages avec un mot par longueur au plus 149
8.2 Concaténation de transducteurs co-séquentiels 149
8.3 La fonction successeur uniforme 151

9 Séries énumératrices des langages rationnels 161
9.1 Rationalité 162
9.1.1 Représentation de la série énumératrice 163
9.1.2 Calcul de la valeur d'un mot 166
9.1.3 Sous-ensembles reconnaissables d'entiers 168
9.2 Décidabilité 171

Introduction

Les travaux présentés dans ce mémoire s'inscrivent dans la théorie des automates et des langages formels. Dans cette théorie, un alphabet est un ensemble fini de lettres et les mots sont toutes les concaténations successives différentes de lettres de l'alphabet. Les langages sont des ensembles, finis ou infinis de mots. La théorie des automates permet de manipuler ces mots et ces langages.

Ce mémoire peut se diviser en deux parties et donne également deux visions différentes de la manière dont la théorie des automates permet de manipuler les langages. Dans une première partie les automates sont vus comme un moyen de représenter les langages rationnels. La problématique abordée est alors celle de naviguer entre deux modèles, les automates finis et les expressions rationnelles, représentant le même langage. Dans la deuxième partie, nous traitons de problèmes de la théorie des automates issus de la considération des systèmes de numération. Ces systèmes permettent de représenter des ensembles dénombrables par un langage infini. Il s'agit alors d'utiliser les automates, dans leur généralité, afin de manipuler des entiers et en particulier d'être capable de 'compter' sur un langage représentant les entiers.

D'un coté donc, les *automates*, dans leur forme la plus simple, sont des machines finies à travers lesquelles des mots peuvent être lus. Si la lecture d'un de ces mots permet d'atteindre un état d'acceptation de la machine, alors on dit que ce mot est reconnu par l'automate. Les automates, finis, peuvent ainsi permettre de reconnaître les mots de langages qui peuvent être infinis. De tels langages sont appelés des langages reconnaissables. Pour chaque langage reconnaissable, il est possible de construire différents automates qui reconnaissent ce langage. En particulier, il est possible d'avoir des automates dits *déterministes*, c'est-à-dire pour lequel la lecture d'un mot se fait de manière linéaire : il suffit de suivre les lettres consécutives du mot sur un graphe sans autre choix.

De l'autre coté, il est possible de créer des langages en appliquant des successions d'opérations sur les langages. Ces opérations sont l'union de deux langages, le concaténation membres à membres de deux langages et l'étoile d'un langage, c'est-à-dire l'union pour tout entier n des concaténations n fois de ce langage. Ces opérations sont appelées opérations rationnelles. Si l'on prend l'ensemble le plus petit fermé pour ces opérations rationnelles et contenant les langages composés uniquement des lettres de l'alphabet, on obtient une classe de langages appelés langages rationnels. Comme ces langages sont obtenus par une succession d'opérations à partir d'atomes, il est possible des les représenter de manière finie par une formule bien parenthésée représentant ces opérations. Une telle formule est appelée *expression*

rationnelle.

Le théorème fondamental de la théorie des automates, énoncé par Kleene, stipule qu'un langage sur un alphabet est rationnel si, et seulement si, il est reconnaissable par un automate fini sur le même alphabet. Autrement dit, les langages rationnels et les langages reconnaissables sont les mêmes. Cela signifie que les automates finis et les expressions rationnelles représentent les mêmes objets : les langages rationnels. Dès lors le problème du passage d'un modèle à un autre est un problème qui a fait l'objet de beaucoup d'intérêt.

Dans un premier sens, les algorithmes les plus classiques pour calculer une expression rationnelle à partir d'un automate fini – en conservant le langage qu'ils représentent – sont les algorithmes de 'McNaughton et Yamada' [34] et d'élimination d'états (*cf.*[49] et [50] par exemple). Ces deux algorithmes, bien que différents, produisent des résultats similaires. En particulier les deux algorithmes dépendent d'un ordre sur les états et un choix d'ordre différent peut produire des expressions très différentes dans la forme et dans la taille. Un même automate peut donc produire différentes expressions rationnelles. Une autre méthode pour produire des expressions rationnelles est la résolution de systèmes d'équations en utilisant le lemme d'Arden (*cf.* [43, p107–108]).

Dans le deuxième sens, il existe beaucoup plus d'algorithmes pour obtenir un automate fini à partir d'une expression rationnelle. Ces algorithmes peuvent même fournir des automates différents les uns des autres. On peut en particulier évoquer l'automate de Thompson qui est un automate construit à partir de l'expression avec des successions de constructions 'de base' qui correspondent aux opérations rationnelles de l'expression. Cependant cet automate n'en est pas un au sens classique du terme car il possède des *transitions spontanées* c'est-à-dire étiquetées par le mot vide et non une lettre de l'alphabet. Dans les algorithmes notables, l'algorithme le plus répandu est celui de Glushkov [26] qui construit un automate, dit *des positions* dont la taille est celle de l'expression, c'est-à-dire qu'il y a – en plus de l'état qui ne sert que d'état initial – autant d'états que d'occurrences de lettres dans l'expression rationnelle. Cela fait que cet automate semble très proche dans les informations qu'il contient de l'expression rationnelle elle-même. L'automate des positions a également cela de particulier qu'il est possible de le calculer de plusieurs manières, *a priori* différentes. En effet, si la méthode de [26] construit l'automate des positions en calculant les ensembles de lettres qui peuvent suivre une lettre donnée – à partir de l'expression rationnelle –, il existe un algorithme qui le construit avec une construction un peu comme pour l'automate de Thompson : des constructions de base permettent de réaliser la somme et la concaténation de deux automates standards ainsi

que l'étoile standard d'un automate standard – c'est à dire avec un unique état initial qui n'est pas accessible à partir du reste de l'automate –, pour construire l'automate des positions d'une expression, il suffit de construire inductivement cet automate (*cf.* [43, p156–157]).

Bien que l'automate des positions soit un automate intéressant pour les raisons que nous venons d'exposer, d'autres constructions d'automates à partir d'expressions rationnelles ont été considérées, dans le but de construire un automate le plus petit possible. En particulier on peut citer l'automate *follow* de Ilie et Yu [27] qui se trouve être un quotient de l'automate des positions.

Les problématiques de passer d'un modèle à l'autre et réciproquement étant bien étudiées, il apparaît intéressant de regarder la réversibilité de ces opérations. Une première voie, développée dans [13], s'intéresse à la récupération d'une expression rationnelle 'petite' à partir de l'automate des positions d'une expression rationnelle. Une autre problématique envisagée dans [32] la possibilité de retrouver un automate fini à partir d'une expression rationnelle calculée sur cet automate. Une solution à cette problématique, donnée dans [32], utilise l'automate des *termes dérivés cassés* d'une expression.

Pour définir l'automate des termes dérivés cassés, il faut tout d'abord revenir à la définition d'une dérivation pour les expressions rationnelles. Cette dérivation a été introduite par Brzozowski dans [11]. Il s'agit d'une transposition syntaxique de la notion de quotient d'un langage aux expressions rationnelles : la dérivation d'une expression par rapport à une lettre est une expression qui dénote le quotient par rapport à cette lettre du langage dénoté par l'expression de départ. Il se trouve que cette dérivation permet, modulo des identités sur les expressions rationnelles – l'associativité, la commutativité et l'idempotence de la somme –, de construire un automate fini reconnaissant le langage de l'expression. Cet automate est déterministe et sa taille est exponentielle en la taille de l'expression.

Il apparaît donc que cette dérivation ne fournit pas un automate 'petit', en plus de devoir réaliser des opérations contraignantes comme de repérer les expressions égales par commutativité de l'addition. C'est pour résoudre ce problème qu'Antimirov propose dans [5] une variante de la dérivation : cette nouvelle dérivation d'une expression rationnelle n'est plus une expression rationnelle mais un ensemble d'expressions rationnelles. Cette modification des dérivées, que nous appelons *termes dérivés* d'une expression, permet de construire un automate qui est un quotient de l'automate des positions [18], donc plus petit.

Les termes dérivés cassés sont une variante des termes dérivés. Cette variante consiste en l'éclatement des termes dérivés qui commencent par

une somme – c'est-à-dire, si l'expression est une concaténation, si le terme le plus à gauche commence par une somme–. Cette variante permet entre autre d'avoir des automates plus généraux que les automates des termes dérivés qui sont toujours standard. Plus important encore, il est montré dans [32] que si l'on prend un automate – co-déterministe et co-mimimal –, que l'on calcule une expression rationnelle par élimination d'états et que l'on construit le co-quotient minimal de l'automate des termes dérivés cassés de cette expression alors on obtient l'automate de départ. Et ce, quel que soit l'ordre d'élimination des états choisis.

La première partie de ce mémoire se propose d'étudier plus en avant les termes dérivés cassés, en particulier la question de la borne de la taille de l'automate par rapport à l'expression de départ est étudiée. Si dans le cas général la taille de l'automate des termes dérivés cassés peut être le double de celle de l'automate des positions de l'expression, il apparaît que pour une certaine classe d'expressions cette borne devient plus raisonnable. La classe des expressions en question est une classe introduite par Brügemann-Klein dans [10]. Il s'agit des expressions rationnelles sous *forme normale* c'est-à-dire telles que le mot vide n'est jamais 'étoilé'. Dans [10] il est en particulier montré l'intérêt de cette classe qui permet un calcul quadratique de l'automate des positions. Mieux encore, toute expression peut se réduire en temps linéaire en une expression en forme normale ayant le même automate des positions. En résumé les expressions en forme normale permettent de calculer de manière quadratique l'automate des positions de n'importe quelle expression. Dans le sujet qui nous intéresse elles permettent également d'avoir une borne sur la taille de l'automate des termes dérivés cassés d'une expression qui soit la même borne que celle des automates des termes dérivés.

Un autre problème que nous étudierons concerne la compatibilité des opérations de dérivation avec l'associativité de la concaténation des expressions rationnelles. En effet, il est très fréquent de noter les concaténations multiples sans parenthésage explicite car la plupart des opérations classiques sur les expressions rationnelles donnent le même résultat indépendamment du parenthésage. Ce n'est pas le cas pour les termes dérivés (respectivement cassés) et un parenthésage différent peut donner un nombre de termes dérivés (resp. cassés) différent. Nous montrons ainsi que le parenthésage le plus à gauche possible donne un nombre minimal de termes dérivés (resp. cassés). Néanmoins nous notons également que la dérivation (resp. cassée) est compatible avec l'associativité des expressions rationnelles.

Ce mémoire s'intéresse également à la construction de l'automate des termes dérivés cassés. Des travaux ont déjà été réalisés pour la construction de l'automate des termes dérivés. On peut citer en particulier le travail

de [1] qui donne des constructions à la fois pour l'automate des positions, l'automate follow et l'automate des termes dérivés. Ces constructions sont basées uniquement sur des minimisations et des éliminations de transitions spontanées à partir de l'automate de Thompson de l'expression.

Une autre construction de l'automate des termes dérivés est proposée par Champarnaud et Ziadi dans [17]. Leur construction permet de calculer les termes dérivés d'une expression par le calcul de chemins sur l'arbre syntaxique représentant l'expression rationnelle. Cette idée de chemin a été pour la première fois utilisée dans [51] pour calculer l'automate des positions de l'expression. D'autres algorithmes plus récents (*cf.*[28]) améliorent la méthode de construction de Champarnaud et de Ziadi. Dans ce mémoire nous donnons une méthode similaire à celle de [28]. Les différences que nous apportons ne modifient pas le calcul effectif de l'automate mais permettent de donner une méthode simplement adaptable à notre objectif : donner un algorithme pour le calcul de l'automate des termes dérivés cassés. En particulier, on peut noter que l'algorithme de [28] utilise explicitement le fait que l'automate des termes dérivés est un quotient de l'automate des positions. Comme ce n'est pas le cas de l'automate des termes dérivés cassés, nous évitons d'utiliser ce résultat pour le calcul de l'automate des termes dérivés.

Finalement il est possible d'envisager une généralisation de tous les résultats pour les automates à multiplicités. Les automates à multiplicités ne reconnaissent plus seulement un langage mais une série, c'est-à-dire qu'à un mot donné ils associent un élément d'un semi-anneau. De ce point de vue on peut donc voir les automates 'classiques' comme des automates à multiplicités sur le semi-anneau booléen. Si un mot prend pour un automate la valeur 1 il est reconnu, sinon il n'est pas reconnu. Avec la généralisation des automates pour les multiplicités vient également une généralisation des expressions rationnelles pour les multiplicités. Ces expressions rationnelles dénotent également des séries et il a été montré par Schützenberger que les séries dénotées par des expressions rationnelles et les séries qui sont le comportement d'un automate à multiplicités sont les mêmes.

De ce constat le problème du passage d'expressions rationnelles à automates en gardant la série se pose naturellement comme généralisation du cas booléen. Comme dans le cas booléen, des constructions d'automates à partir d'expressions rationnelles existent. Dans [12] l'automate des positions d'une expression rationnelle est généralisée pour les multiplicités. Cet automate est toujours un automate dont les états sont les occurrences de lettres dans l'expression rationnelle. En fait l'automate des positions d'une expression à multiplicités est le même automate que celui de la version booléenne de l'expression pour laquelle chaque transition a un coefficient égal à la multiplicité

des possibilités de faire suivre une lettre donnée par une autre. Dans [31], une autre construction de cet automate est proposée : l'automate des positions est construit inductivement à partir de constructions élémentaires pour la somme, le produit, l'étoile et la multiplication par un scalaire d'automates standard – avec un unique état initial qui n'est pas accessible par le reste de l'automate et dont le poids d'entrée est $1_{\mathbb{K}}$ –. Cet automate peut également être construit, à partir de l'automate de Thompson à multiplicités de l'expression auquel on applique des éliminations de transitions spontanées et des minimisations ([1]). Finalement il est également possible de le construire, comme cela est présenté dans [14], à partir de l'arbre syntaxique d'une expression, par une méthode similaire à celle de la ZPC-structure pour les expressions rationnelles à multiplicités.

Afin d'avoir un automate plus petit, il est également possible de construire, à partir d'une expression rationnelle, l'automate des termes dérivés de l'expression. Les termes dérivés des expressions à multiplicités ont déjà été définies dans [31]. La définition choisie est telle que l'automate des termes dérivés est, comme dans le cas booléen, un quotient de l'automate des positions à multiplicités de l'expression. Dans ce mémoire, nous allons utiliser ces définitions pour donner un algorithme pour construire l'automate des termes dérivés à multiplicités. En plus du calcul de l'automate des termes dérivés à partir de sa définition, il est possible de le construire en utilisant la version à multiplicité de la ZPC-structure [15]. Nous allons présenter comment adapter l'algorithme de construction de l'automate que nous présentons dans le cas booléen pour construire l'automate des termes dérivés d'une expression rationnelle à multiplicités dans $\mathbb{K}$. Cet algorithme utilise des chemins de nœuds de l'arbre syntaxique pour représenter les termes dérivés. L'application de réductions sur ces chemins permet d'identifier ceux qui dénotent la même expression, donc le même état de l'automate des termes dérivés. Finalement nous présenterons une proposition de définition pour les termes dérivés cassés comme une synthèse des travaux réalisés sur les termes dérivés cassés booléens et les termes dérivés sur les expressions à multiplicités. Nous donnerons également un algorithme pour le calcul de ces termes dérivés cassés pour les expressions rationnelles à multiplicités.

Dans une deuxième partie, les travaux de cette thèse s'inscrivent dans la théorie des systèmes de numération abstraits. Un système de numération est un moyen de représenter des nombres par des mots sur un alphabet donné. Nous ne nous intéresserons dans ce mémoire qu'à la représentation d'entiers naturels. Les systèmes de numération les plus connus sont ainsi les

systèmes en base binaire ou en base décimale. Dans ces cas, les alphabets sont les chiffres 0 et 1 et les chiffres de 0 à 9. A partir d'une représentation il est facile de calculer la valeur qu'elle représente en multipliant chaque chiffre par la puissance de 2 ou de 10 adéquate et en faisant la somme de chaque résultat. On peut également noter que l'ensemble des représentations des entiers forme le langage des mots qui ne commencent pas par 0 – sauf le mot 0 lui même –. Si l'on ordonne les représentations dans l'ordre radiciel, c'est-à-dire en prenant d'abord compte la longueur du mot puis l'ordre lexicographique, on se rend compte que la représentation d'un entier n donné est exactement le $n+1$-ième mot dans le langage décrit précédemment. Cette propriété, qui se retrouve dans d'autres systèmes de numération classiques, a motivé la définition des systèmes de numération abstraits par Lecomte et Rigo dans [30] : étant donné un langage, l'entier n est associé au $n+1$-ième mot, dans l'ordre radiciel, de ce langage. Cependant, dans un tel système de numération, le calcul de la valeur d'un mot n'est, a priori, pas aussi direct que dans le cas du système binaire. En effet pour connaître la valeur d'un mot il faut 'compter' tous les mots plus petits que lui, c'est-à-dire les énumérer un à un.

Dans cette partie, les automates seront vus comme des machines qui permettent des opérations sur les mots et donc, par leur représentations, sur les entiers. Pour utiliser les automates dans cet optique, nous utilisons une généralisation des automates, non pas sur un monoïde libre, mais sur un monoïde produit de monoïde libres. Un tel automate est appelé transducteur et peut être vu comme un automate 'avec sortie'. C'est-à-dire que lorsque l'on lit un mot, comme dans un automate classique, dans un transducteur, on obtient non seulement un résultat d'acceptation du mot, mais également un nouveau mot sur un monoïde libre. Les transducteurs réalisent des relations rationnelles qui associent un ou des mots de sortie à un mot d'entrée.

En particulier on peut envisager l'utilisation des transducteurs pour étudier la fonction successeur d'un système de numération abstrait. La fonction successeur est, étant donné un langage, la fonction qui associe le mot suivant dans ce langage ordonné par l'ordre radiciel. C'est-à-dire que la fonction successeur permet de 'compter' les entiers dans un système de numération donné. En effet si l'on donne un entier par sa représentation, la fonction permet de 'réciter' les représentations des entiers qui suivent. Cette fonction est une fonction rationnelle si le langage considéré est rationnel, et elle peut être décrite par un transducteur fini lettre-à-lettre. La fonction successeur d'un langage rationnel n'est pas nécessairement séquentielle ou co-séquentielle. Nous prouverons que la fonction successeur dans un langage rationnel est une fonction co-séquentielle par morceaux.

La séquentialité, pour les fonctions rationnelle, est une notion proche du déterminisme des automates classiques : un transducteur séquentiel est un transducteur dont l'entrée est lu déterministiquement. Les fonctions séquentielles sont les fonctions qui peuvent être réalisées par des transducteurs séquentiels. La notion de séquentialité par morceaux a été définie par Choffrut et Schützenberger sous le nom de fonctions pluri-sousséquentielles dans [21]. Les fonctions séquentielles par morceaux sont les fonctions rationnelles qui peuvent être réalisées par une union finie de transducteurs séquentiels à domaines deux à deux disjoints. Nous étudions dans ce mémoire les fonctions séquentielles par morceaux et, en particulier, nous montrons que ce sont exactement les fonctions qui peuvent être réalisées par des cascades de transducteurs séquentiels : on part d'un mot, on le lit dans un transducteur séquentiel, et, suivant l'état de sortie, on lit l'image dans un nouveau transducteur séquentiel. C'est-à-dire que les cascades réalisent des lectures successives de mots acceptés dans différents transducteurs séquentiels qui dépendent de l'état de sortie à chaque lecture. Ces cascades fournissent naturellement une possibilité de calculer la sortie de la fonction qu'elles réalise en temps linéaire puisqu'il suffit d'un nombre fixé de lectures du mot pour obtenir l'image.

Finalement pour compter dans un système de numération abstrait, il est utile de pouvoir connaître la valeur d'une représentation donnée. Ce problème, pour la théorie des automates, se réduit à une étude de la série énumératrice du système de numération : la série formelle qui, à chaque représentation, associe l'entier qu'elle code plus un. En fait cette série ne dépendant que du langage et de l'ordre sur l'alphabet, elle peut également être appelée série énumératrice du langage. Dans le cas d'un système de numération abstrait rationnel, la série énumératrice est une série rationnelle [39]. Ce résultat a également été montré par Choffrut et Goldwurm dans [20]. Ce dernier article fourni même un cadre plus général et il est montré que différentes séries, qui correspondent à différents ordre en plus du radiciel, sont rationnelles lorsque le langage est rationnel. Il est en outre également montré que si le langage est algébrique, alors les mêmes séries le sont aussi.

Nous montrerons dans le dernier chapitre que le résultat de rationalité de la série énumératrice d'un langage rationnel permet de calculer, par des méthodes classiques, la valeur d'un mot dans un système de numération. Nous donnerons une preuve simple et qui permet de construire une représentation de la série énumératrice d'un langage rationnel à partir de la représentation d'un automate non-ambigu reconnaissant ce langage. La construction de la représentation de la série nous permet ensuite de présenter un calcul simple pour la valeur d'un mot dans le système de numération abstrait.

Enfin nous donnons également une méthode pour construire un automate reconnaissant l'ensemble des mots dont les valeurs sont dans un ensemble reconnaissable de nombres donné – c'est à dire une union finie d'ensemble ultimement périodiques de nombres –. La construction d'un tel automate est également traitée dans [29] et dans [40]. Finalement la rationalité de la série énumératrice nous permet également de montrer qu'il est décidable si une série rationnelle quelconque est une série énumérative d'un système de numération abstrait.

L'étude du nombre de termes dérivés cassés d'une expression rationnelle booléenne et de l'effet du parenthésage des concaténations d'expressions rationnelles sur la dérivation ont été présentés à DCFS09 et ont été publiés dans [2]. Le début de l'étude des fonctions séquentielles par morceaux et des cascades de transducteurs a été présentée aux JM08. La co-séquentialité par morceaux de la fonction successeur d'un langage rationnel est présentée dans [3]. Les résultat concernant la rationalité de la série énumératrice d'un langage rationnel ont été présentés aux JM10 (cf. [4]).

Première partie

Les termes dérivés cassés d'une expression rationnelle

Chapitre 1

Notations et préliminaires

1.1 Mots et langages et automates

Un *alphabet* A est un ensemble fini d'éléments appelés *lettres*. Un *mot* sur l'alphabet A est un élément du *monoïde libre* A^*, c'est-à-dire une suite finie de lettres de A ou l'identité 1_{A^*}, appelée *mot vide* de A^*.

La *longueur* d'un mot u sur un alphabet A, notée $|u|$, est le nombre de lettres qui composent u. Si a est une lettre de A, alors nous notons $|u|_a$ le nombre d'occurrences de a dans u. Par exemple $|abaa| = 4$, $|abaa|_a = 3$ et $|abaa|_b = 1$.

Un *langage* de mots sur l'alphabet A est un sous-ensemble, fini ou infini, de A^*.

Dans ce mémoire, plusieurs sortes d'automates sont définis : les automates sur un alphabet, les automates dont les sorties peuvent être étiquetées par un mot, les $\mathbb{K}$-automates ainsi que les transducteurs. Nous donnons ici la définition des automates booléens les plus simples.

Définition 1. *Un* automate fini $\mathcal{A}$ *sur un alphabet* A *est un quintuplet* $\langle Q, A, E, I, T\rangle$ *où :*

(i) Q *est un ensemble fini dont les éléments sont appelés* états *;*

(ii) E *est un ensemble d'arcs entre des éléments de* Q *étiquetés par* A*, ce sont les* transitions *de* $\mathcal{A}$ *;*

(iii) I *est un sous-ensemble de* Q*, l'ensemble des* états initiaux *;*

(iv) T *est un sous-ensemble de* Q*, l'ensemble des* états finaux.

Dans ce mémoire, nous ne traitons pas le cas des automates infinis et nous appellons automates les automates finis. La Figure 1.1 montre comment nous représentons les automates. Les états initiaux sont repérés par des flèches entrantes, les états finaux par des flèches sortantes.

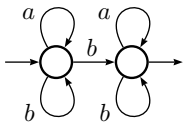


FIGURE 1.1 – Un automate, $\mathcal{B}_1$ sur l'alphabet $\{a, b\}$

Un état p d'un automate $\mathcal{A}$ est dit *acccessible* s'il existe un chemin dans l'automate d'un état initial de I à p. Il est *co-accessible* s'il existe un chemin de p à un élément de T. Un automate $\mathcal{A}$ est *accessible* si tous ses états le sont, *co-accessible* si tous ses états le sont et *émondé* si il est à la fois accessible et co-accessible.

Un *calcul réussi* de $\mathcal{A}$ est un chemin qui commence dans un état initial et qui se termine dans un état final. Le *langage reconnu* par $\mathcal{A}$, noté $|\mathcal{A}|$ est le sous-ensemble de A^* des étiquettes des calculs réussis :

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

Par exemple le langage reconnu par l'automate $\mathcal{B}_1$ de la Figure 1.1 est $|\mathcal{B}_1| = A^*\{b\}A^*$, c'est-à-dire le langage des mots sur A qui contiennent au moins un b.

Un langage de A^* est dit *reconnaissable* s'il existe un automate fini qui le reconnaît.

Un automate est dit *déterministe* s'il a un unique état initial i et si, pour tout état p, il existe, au plus, une unique transition sortante étiquetée par chaque lettre de A. Un automate est dit *ambigu* si il existe un état q et un mot u tel qu'il y ait plus d'un chemin d'un état initial à q étiqueté u. Un automate déterministe est non-ambigu.

Proposition 1. *Il est possible de transformer tout automate fini $\mathcal{A}$ en un automate fini déterministe $\mathcal{B}$ qui reconnaît le même langage. Si l'automate $\mathcal{A}$ a n états, il est possible de construire $\mathcal{B}$ avec au plus 2^n états.*

Les langages reconnaissables sont donc les langages reconnus par les automates finis déterministes. Il peut y avoir plusieurs automates déterministes qui reconnaissent le même langage.

1.2 Expressions rationnelles

Afin de définir les expressions rationnelles et les langages rationnels sur un alphabet, nous définissons tout d'abord les opérations rationnelles sur les sous-ensembles d'un monoïde :

Définition 2. *Les opérations rationnelles sur les sous-ensembles d'un monoïde M sont l'union, le produit d'ensembles et l'étoile définie par :*

$$X^* = \{1_M\} \cup \bigcup_{n>0} X^n \ .$$

La famille des sous-ensembles rationnels *de M, notée* $\mathrm{Rat}\, M$*, est la fermeture des ensembles finis par ces opérations.*

En particulier, nous appelons *langages rationnels* sur A^* les sous-ensembles rationnels de A^*. Les expressions rationnelles sont un moyen de décrire ces langages en notant les opérations rationnelles successives pour obtenir le sous-ensemble rationnel. Plus précisément :

Définition 3. *Une* expression rationnelle E *sur A est une formule bien parenthésée obtenue inductivement par la grammaire suivante dans lesquelles les atomes sont les lettres de A et les expressions prédéfinies* 0 *et* 1 *:*

$$\mathsf{E} \to \{a \mid a \in A\} \mid 0 \mid 1 \mid (\mathsf{E} + \mathsf{E}) \mid (\mathsf{E} \cdot \mathsf{E}) \mid (\mathsf{E}^*) \ .$$

L'ensemble des expression rationnelles sur un alphabet A est noté $\mathsf{RatE(A)}$.

Comme toute expression bien parenthésée, une expression rationnelle E peut être représentée par un arbre syntaxique, que nous notons T_{E}. La *profondeur* d'une expression E, notée $\mathsf{d(E)}$, est la profondeur de l'arbre syntaxique, plus précisément :

$$\begin{gathered}
\mathsf{d}(0) = 0 \ , \ \mathsf{d}(1) = 1 \ , \ \mathsf{d}(a) = 1 \ , \\
\mathsf{d}(\mathsf{E} + \mathsf{F}) = max(\mathsf{d}(\mathsf{E}), \mathsf{d}(\mathsf{F})) + 1 \ , \\
\mathsf{d}(\mathsf{E} \cdot \mathsf{F}) = max(\mathsf{d}(\mathsf{E}), \mathsf{d}(\mathsf{F})) + 1 \ , \\
\mathsf{d}(\mathsf{E}^*) = \mathsf{d}(\mathsf{E}) + 1 \ .
\end{gathered}$$

Exemple 1. La formule $((a+b)^* \cdot b) \cdot (a+b)^*$ est une expression rationnelle de profondeur 5.

Par défaut, lorsque les parenthèses seront absentes, c'est le parenthésage à gauche qui sera utilisé : $\mathsf{E} \cdot \mathsf{F} \cdot \mathsf{G} = ((\mathsf{E} \cdot \mathsf{F}) \cdot \mathsf{G})$. Le justification de ce choix sera donnée par l'analyse de l'influence du parenthésage des produits sur les termes dérivés (et les termes dérivés cassés). La *longueur littérale* de l'expression E, notée $\ell(\mathsf{E})$, est le nombre d'occurrences de lettres de l'alphabet A

dans celle-ci. Le *langage dénoté* par une expression rationnelle E, noté $|\mathsf{E}|$, est défini inductivement par :

$$|0| = \emptyset\,, \quad |1| = \{1_{A^*}\}\,, \quad |a| = \{a\}\ ,$$
$$|(\mathsf{E}+\mathsf{F})| = |\mathsf{E}| \cup |\mathsf{F}|\ ,$$
$$|(\mathsf{E}\cdot\mathsf{F})| = |\mathsf{E}||\mathsf{F}|\ ,$$
$$|(\mathsf{E}^*)| = |\mathsf{E}|^*\ .$$

Exemple 2 (*Ex. 1 cont.*)**.** Le langage dénoté par l'expression rationnelle $((a+b)^*\cdot b)\cdot(a+b)^*$ est le langage A^*bA^* des mots contenant au moins une occurrence de b.

Les langages rationnels sur A^* sont exactement les langages dénotés par une expression rationnelle sur A^*. Il peut y avoir plusieurs expressions rationnelles qui dénotent le même langage rationnel. Deux expressions rationnelles sont dites *équivalentes* si elles dénotent le même langage.

Afin de simplifier les manipulations sur les expressions rationnelles, nous définissons des identités triviales, que nous notons **(T)**, et pour lesquelles les expressions rationnelles sont équivalentes de manière évidente et tous les calculs réalisés par la suite seront réalisés modulo **(T)** :

$$\mathsf{E}+0 \equiv 0+\mathsf{E} \equiv \mathsf{E}\,, \quad \mathsf{E}\cdot 0 \equiv 0\cdot\mathsf{E} \equiv 0\,, \quad \mathsf{E}\cdot 1 \equiv 1\cdot\mathsf{E} \equiv \mathsf{E}\,, \quad 0^* \equiv 1\ .(\mathbf{T})$$

Une expression est dite réduite par ces identités triviales si elle ne contient aucune sous-expression de la forme d'un membre gauche d'une de ces identités. En particulier, ces expressions permettent d'assurer qu'une expression différente de 0 n'a pas pour sous-expression 0, ce qui va nous permettre de traiter ce cas différemment lors de nos preuves par induction sur la profondeur des expressions.

Il apparaît que toute expression rationnelle E peut être réécrite dans une expression réduite équivalente E' et que cette expression réduite est unique indépendamment de l'ordre dans lequel sont appliquées les différentes identités.

Finalement, il est possible de définir inductivement un booléen, appelé le *terme constant* d'une expression rationnelle E, noté $\mathsf{c}(\mathsf{E})$, par :

$$\mathsf{c}(0) = 0\,, \quad \mathsf{c}(1) = 1\,, \quad \forall a \in A \quad \mathsf{c}(a) = 0\ ,$$
$$\mathsf{c}(\mathsf{F}+\mathsf{G}) = \mathsf{c}(\mathsf{F}) \vee \mathsf{c}(\mathsf{G})\,, \quad \mathsf{c}(\mathsf{F}\cdot\mathsf{G}) = \mathsf{c}(\mathsf{F}) \wedge \mathsf{c}(\mathsf{G})\,, \quad \mathsf{c}(\mathsf{F}^*) = 1\ .$$

Proposition 2. *Le terme constant d'une expression* E *est égal à* 1 *si, et seulement si, le mot vide* 1_{A^*} *est dans le langage* $|\mathsf{E}|$.

Nous avons pris soin de donner des définitions sur les expressions rationnelles de manière inductive car nous allons pouvoir ainsi utiliser les objets définis dans des preuves par induction. Ces définitions inductives permettent également de faire les calculs sur l'arbre syntaxique de bas en haut, ce qui sera utilisé lorsque nous donnerons une méthode pour calculer l'automate des termes dérivés.

Finalement nous rappelons le théorème de Kleene qui est le théorème fondamental à la base de la théorie des automates :

Théorème 3 (Kleene)**.** *Pour tout alphabet fini, l'ensemble des langages rationnels est égal à l'ensemble des langages reconnaissables.*

Il est donc possible de représenter un même langage rationnel par un automate fini qui le reconnaît ou par une expression rationnelle qui le dénote. C'est ce théorème qui nous amène à envisager la possibilité de construire des automates à partir d'expressions rationnelles, ou de construire des expressions rationnelles à partir d'automates en gardant le langage représenté.

1.3 Automate des positions

Dans cette section nous allons donner l'exemple le plus classique d'automate construit à partir d'une expression rationnelle. Cet automate, que nous appellerons *l'automate des positions* d'une expressions rationnelle, a été défini simultanément et indépendamment par Glushkov [26] – il est, pour cette raison, souvent appelé *automate de Glushkov* de l'expression – et MacNaughton et Yamada [34]. Cet automate est intrinsèquement très lié à l'expression rationnelle et en particulier il existe plusieurs algorithmes très différents pour le construire à partir de l'expression. Nous décrirons ici la méthode originale pour le construire.

L'automate des positions est *standard*, c'est-à-dire qu'il a un unique état initial i et qu'aucune transition ne pointe vers i. Dans [43], l'automate des positions est également appelé *automate standard* de l'expression et, il est possible de le calculer par une méthode inductive à l'aide de constructions de bases sur les automates standard pour les opérations rationnelles.

Pour construire l'automate, nous devons tout d'abord définir les positions d'une expression rationnelle. Une expression rationnelle E peut être linéarisée - c'est-à-dire que chaque lettre n'apparaît qu'au plus une fois dans E – sur un alphabet plus grand (de la taille de l'expression) en une expression rationnelle $\overline{\mathsf{E}}$ en indiçant les occurrences de chaque lettre. Le nouvel alphabet ainsi créé est appelé *l'ensemble des positions* de l'expression E. Une autre

manière de voir ces positions est sur l'arbre syntaxique : les positions sont les feuilles qui sont étiquetées par une lettre de l'alphabet. L'ensemble des positions est noté $\mathrm{fp}(\mathsf{E})$ et appelé l'ensemble des *feuilles propres* de E.

Définition 4. *Soit* E *une expression rationnelle. L'ensemble* $\mathsf{First}(\mathsf{E})$ *des premières positions de* E*, l'ensemble* $\mathsf{Last}(\mathsf{E})$ *des dernières positions de* E *et l'ensemble* $\mathsf{Follow}(l,\mathsf{E})$ *des positions qui suivent* l *dans* E*, où* l *est une position, sont définis inductivement par :*

$$\begin{aligned}
\mathsf{First}(0) = \mathsf{First}(1) &= \emptyset \ , \\
\mathsf{First}(a) &= \{a\},\ a \in \mathrm{fp}(\mathsf{E}) \ , \\
\mathsf{First}(\mathsf{F}+\mathsf{G}) &= \mathsf{First}(\mathsf{F}) \cup \mathsf{First}(\mathsf{G}) \ , \\
\mathsf{First}(\mathsf{F}\cdot\mathsf{G}) &= \mathsf{First}(\mathsf{F}) \cup \mathsf{c}(\mathsf{F})\mathsf{First}(\mathsf{G}) \ , \\
\mathsf{First}(\mathsf{F}^*) &= \mathsf{First}(\mathsf{F}) \ ,
\end{aligned}$$

$$\begin{aligned}
\mathsf{Last}(0) = \mathsf{Last}(1) &= \emptyset \ , \\
\mathsf{Last}(a) &= \{a\},\ a \in \mathrm{fp}(\mathsf{E}) \ , \\
\mathsf{Last}(\mathsf{F}+\mathsf{G}) &= \mathsf{Last}(\mathsf{F}) \cup \mathsf{Last}(\mathsf{G}) \ , \\
\mathsf{Last}(\mathsf{F}\cdot\mathsf{G}) &= \mathsf{Last}(\mathsf{G}) \cup \mathsf{c}(\mathsf{G})\mathsf{Last}(\mathsf{F}) \ , \\
\mathsf{Last}(\mathsf{F}^*) &= \mathsf{Last}(\mathsf{F}) \ ,
\end{aligned}$$

$$\begin{aligned}
\mathsf{Follow}(l,0) = \mathsf{Follow}(l,1) &= \emptyset \ , \\
\mathsf{Follow}(l,a) &= \emptyset,\ a \in \mathrm{fp}(\mathsf{E}) \ , \\
\mathsf{Follow}(l,\mathsf{F}+\mathsf{G}) &= \mathsf{Follow}(l,\mathsf{F}) \cup \mathsf{Follow}(l,\mathsf{G}) \ , \\
\mathsf{Follow}(l,\mathsf{F}\cdot\mathsf{G}) &= \begin{cases} \mathsf{Follow}(l,\mathsf{F}) \cup \mathsf{c}(\mathsf{F})\mathsf{Follow}(l,\mathsf{G}) \cup \mathsf{First}(\mathsf{G}) & \textit{si } l \in \mathsf{Last}(\mathsf{F}) \\ \mathsf{Follow}(l,\mathsf{F}) \cup \mathsf{c}(\mathsf{F})\mathsf{Follow}(l,\mathsf{G}) & \textit{sinon} \end{cases} \ , \\
\mathsf{Follow}(l,\mathsf{F}^*) &= \begin{cases} \mathsf{Follow}(l,\mathsf{F}) \cup \mathsf{First}(\mathsf{F}) & \textit{si } l \in \mathsf{Last}(\mathsf{F}) \\ \mathsf{Follow}(l,\mathsf{F}) & \textit{sinon} \end{cases} \ .
\end{aligned}$$

C'est-à-dire que $\mathsf{First}(\mathsf{E})$ est l'ensemble des positions qui sont les premières positions d'un mot dans le langage dénoté par l'expression linéarisée $\overline{\mathsf{E}}$. L'ensemble $\mathsf{Last}(\mathsf{E})$ est l'ensemble des positions qui sont les dernières positions d'un mot dans $|\overline{\mathsf{E}}|$. Si l est une position alors $\mathsf{Follow}(l,\mathsf{E})$ est l'ensemble

des positions telles qu'il existe un mot dans $|\overline{\mathsf{E}}|$ pour lequel elles suivent la position l.

Pour la suite, et lorsqu'il n'y aura pas de confusion possible, nous nous autoriserons à parler de positions dans les mots du langage $|\mathsf{E}|$ pour parler des positions correspondantes dans les mots de $|\overline{\mathsf{E}}|$.

Exemple 3 (*Ex. 1 cont.*)**.** Considérons l'expression rationnelle $((a+b)^* \cdot b) \cdot (a+b)^*$. Afin de différencier les feuilles qui représentent une même lettre, nous donnons un indice à chaque occurrence de lettres dans cette expression rationnelle – linéarisation – : $\mathsf{E}_1 = ((a_1 + b_1)^* \cdot b_2) \cdot (a_2 + b_3)^*$.

Nous avons alors :

$$\begin{aligned} \mathsf{First}(\mathsf{E}_1) &= \{a_1, b_1, b_2\} \ , \\ \mathsf{Last}(\mathsf{E}_1) &= \{b_2, a_2, b_3\} \ , \\ \mathsf{Follow}(a_1, \mathsf{E}_1) = \mathsf{Follow}(b_1, \mathsf{E}_1) &= \{a_1, b_1, b_2\} \ , \\ \mathsf{Follow}(a_2, \mathsf{E}_1) = \mathsf{Follow}(b_2, \mathsf{E}_1) = \mathsf{Follow}(b_3, \mathsf{E}_1) &= \{a_2, b_3\} \ . \end{aligned}$$

Il est alors possible de définir l'automate des positions :

Définition 5. L'automate des positions *d'une expression rationnelle* E *est l'automate tel que :*
(i) *les états sont les positions de* E *et l'unique état initial* i *;*
(ii) *les états finaux sont les positions dans* $\mathsf{Last}(\mathsf{E})$ *;*
(iii) *il y a une transition de* i *aux positions* l *étiquetée par* a*, la lettre associée à* l*, si* $l \in \mathsf{First}(\mathsf{E})$*.*
(iv) *il y a une transition entre une position* l *et une position* l' *étiquetée par* a*, la lettre associée à* l'*, si* $l' \in \mathsf{Follow}(l, \mathsf{E})$*.*

Exemple 4 (*Ex. 1 cont.*)**.** L'automate des positions de l'expression $((a + b)^* \cdot b) \cdot (a + b)^*$ est montré sur la Figure 1.2. On remarque en particulier grâce à cet exemple que les transitions entrantes dans un état sont toujours étiquetées par la lettre de la position représentée par l'état.

Proposition 4. *L'automate des positions d'une expression rationnelle* E *est un automate standard à* $\ell(\mathsf{E}) + 1$ *états qui reconnaît* $|\mathsf{E}|$*.*

1.4 Forme normale

Dans la suite, afin de donner une borne au nombre de termes dérivés cassés d'une expression rationnelle, nous sommes amenés à considérer une

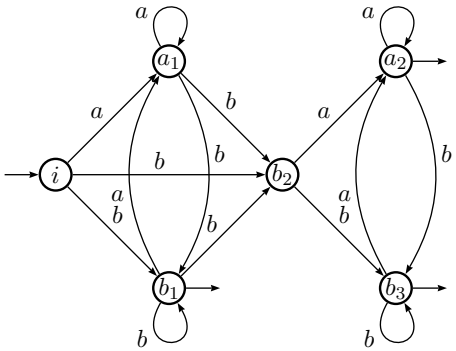


FIGURE 1.2 – L'automate des positions de $((a+b)^* \cdot b) \cdot (a+b)^*$

classe particulière d'expressions rationnelles : les expressions rationnelles en forme normale. Ces expressions ont été définies par Brüggemann-Klein [10] et elles ont l'intérêt de permettre une construction quadratique de l'automate des positions. En outre il est possible de construire à partir d'une expression rationnelle quelconque une expression équivalente en forme normale dont l'automate des positions est identique en temps linéaire. La définition des expressions en forme normale donne donc un algorithme quadratique pour construire l'automate des positions d'une expression.

Nous donnons dans ce mémoire une définition légèrement différente de celle donnée dans [10] des expressions en forme normale mais qui ne change pas les constructions qui permettent d'obtenir une expression en forme normale à partir de n'importe quelle expression rationnelle.

Définition 6 ([10])**.** *Une expression rationnelle* E *est* en forme normale *si, et seulement si, pour toute sous-expression* F^* *de* E*, on a* $\mathsf{c}(\mathsf{F}) = 0$ *.*

Si cette définition décrit les expressions en forme normale, ce qui nous intéresse particulièrement c'est la transformation d'expressions rationnelles quelconques en expressions en forme normale. Pour cela, dans [10], il y a deux opérateurs $\bullet$ et $\circ$ définis par induction. L'opérateur $\bullet$ transforme l'expression en une expression en forme normale en appelant l'opérateur $\circ$ lorsque $\bullet$ est appelé sur des expressions étoilées. L'opérateur $\circ$ transforme, à l'intérieur d'une expression rationnelle, les produits d'expressions rationnelles dont le terme constant est non nul en la somme des mêmes expressions.

Dans cette section nous donnons les définitions inductives originales de [10] puis nous proposons un mode de calcul légèrement différent pour la forme normale d'un expression rationnelle. Cette modification du calcul nous permettra par la suite d'établir des preuves inductives sur les expressions en forme normale.

Les définitions inductives de [10] sont les suivantes :

$$0^\circ = 0\,, \quad 1^\circ = 0\,, \quad \text{for all } a \in A \quad a^\circ = a \ , \tag{1.1}$$

$$(\mathsf{F} + \mathsf{G})^\circ = \mathsf{F}^\circ + \mathsf{G}^\circ \ , \tag{1.2}$$

$$(\mathsf{F} \cdot \mathsf{G})^\circ = \begin{cases} \mathsf{F} \cdot \mathsf{G} & \text{si } \mathsf{c}(\mathsf{F}) = \mathsf{c}(\mathsf{G}) = 0 \ , \\ \mathsf{F}^\circ \cdot \mathsf{G} & \text{si } \mathsf{c}(\mathsf{F}) = 0 \text{ et } \mathsf{c}(\mathsf{G}) = 1 \ , \\ \mathsf{F} \cdot \mathsf{G}^\circ & \text{si } \mathsf{c}(\mathsf{F}) = 1 \text{ et } \mathsf{c}(\mathsf{G}) = 0 \ , \\ \mathsf{F}^\circ + \mathsf{G}^\circ & \text{si } \mathsf{c}(\mathsf{F}) = \mathsf{c}(\mathsf{G}) = 1 \ , \end{cases} \tag{1.3}$$

$$(\mathsf{F}^*)^\circ = \mathsf{F}^\circ \ . \tag{1.4}$$

et :

$$0^\bullet = 0\,, \quad 1^\bullet = 1\,, \quad \text{pour tout } a \in A \quad a^\bullet = a \ , \tag{1.5}$$

$$(\mathsf{F} + \mathsf{G})^\bullet = \mathsf{F}^\bullet + \mathsf{G}^\bullet \ , \tag{1.6}$$

$$(\mathsf{F} \cdot \mathsf{G})^\bullet = \mathsf{F}^\bullet \cdot \mathsf{G}^\bullet \ , \tag{1.7}$$

$$(\mathsf{F}^*)^\bullet = ((\mathsf{F}^\bullet)^\circ)^* \ . \tag{1.8}$$

Il est alors possible d'établir :

Proposition 5 ([10])**.** *Pour toute expression rationnelle* E*, l'expression* $\mathsf{E}^\bullet$ *est en forme normale et est équivalente à* E*.*

Les définitions inductives (1.1)-(1.8) posent le problème suivant : même si les sous-expressions d'une expression en forme normale sont en forme normale, il n'y a pas la même notion d'héritage pour la transformation réalisée par l'opérateur $\circ$. Par exemple :

$$((a^*b^*)^*)^\circ = a + b \qquad \text{alors que} \qquad (c\,(a^*b^*)^*)^\circ = c\,(a^*b^*)^* \ .$$

Cette difficulté est facilement surmontable si l'on se rend compte que dans le calcul de $\mathsf{E}^\bullet$, l'opérateur $\circ$ n'est utilisé qu'en conjonction de l'opérateur $\bullet$ (in (1.8)). Il est alors possible de définir un nouvel opérateur $\square$ inductivement par :

$$\forall \mathsf{E} \in \mathsf{RatE}(\mathsf{A}) \qquad \mathsf{E}^\square = (\mathsf{E}^\bullet)^\circ \ .$$

Il est alors possible de donner une définition inductive de $\square$ grâce aux propriétés suivantes de $\circ$ et de $\bullet$:

Proposition 6 ([2]). *Soit* E *une expression rationnelle. On a :*

$$\mathsf{c}(\mathsf{E}^\circ) = 0 \ , \tag{1.9}$$

$$\mathsf{c}(\mathsf{E}) = 0 \quad \Longrightarrow \quad \mathsf{E} = \mathsf{E}^\circ \qquad \textit{(c'est-à-dire :}\ \mathsf{E} = \mathsf{E}^\circ \Leftrightarrow \mathsf{c}(\mathsf{E}) = 0) \ , \tag{1.10}$$

$$(\mathsf{E}^\circ)^\circ = \mathsf{E}^\circ \qquad \textit{(i.e., } \circ \textit{ est idempotent)} \ , \tag{1.11}$$

$$\mathsf{c}(\mathsf{E}^\bullet) = \mathsf{c}(\mathsf{E}) \ . \tag{1.12}$$

Proposition 7 ([2]). *Soient* F *et* G *deux expressions rationnelles.On a :*

$$(\mathsf{F} + \mathsf{G})^\square = \mathsf{F}^\square + \mathsf{G}^\square \ , \tag{1.13}$$

$$(\mathsf{F} \cdot \mathsf{G})^\square = \begin{cases} \mathsf{F}^\square + \mathsf{G}^\square & \textit{si} \quad \mathsf{c}(\mathsf{F}) = \mathsf{c}(\mathsf{G}) = 1 \\ \mathsf{F}^\bullet \cdot \mathsf{G}^\bullet & \textit{sinon} \end{cases} \ , \tag{1.14}$$

$$(\mathsf{F}^*)^\square = \mathsf{F}^\square \ . \tag{1.15}$$

Démonstration. Des équations (1.2) et (1.6), on a (1.13).
Si $(\mathsf{F} \cdot \mathsf{G})^\square = ((\mathsf{F} \cdot \mathsf{G})^\bullet)^\circ = (\mathsf{F}^\bullet \cdot \mathsf{G}^\bullet)^\circ$; le reste de la preuve de (1.14) demande un examen des cas suivants (en utilisant implicitement le fait que $\mathsf{c}(\mathsf{H}^\bullet) = \mathsf{c}(\mathsf{H})$ pour toute expression H) :

Si $\mathsf{c}(\mathsf{F}) = 0$ ou $\mathsf{c}(\mathsf{G}) = 0$ alors $\mathsf{c}(\mathsf{E}) = 0$ et, l'équation (1.10) donne donc $(\mathsf{F}^\bullet \cdot \mathsf{G}^\bullet)^\circ = \mathsf{F}^\bullet \cdot \mathsf{G}^\bullet$.

Si $\mathsf{c}(\mathsf{F}) = \mathsf{c}(\mathsf{G}) = 1$, alors $(\mathsf{F}^\bullet \cdot \mathsf{G}^\bullet)^\circ = (\mathsf{F}^\bullet)^\circ + (\mathsf{G}^\bullet)^\circ = \mathsf{F}^\square + \mathsf{G}^\square$.

Finalement, $(\mathsf{F}^*)^\square = ((\mathsf{F}^*)^\bullet)^\circ = (((\mathsf{F}^\bullet)^\circ)^*)^\circ = ((\mathsf{F}^\bullet)^\circ)^\circ = (\mathsf{F}^\bullet)^\circ$ par l'équation (1.11). □

Corollaire 8 ([2]). *L'expression* $\mathsf{E}^\square$ *peut être calculée par induction sur la profondeur de* E *en utilisant les équations (1.13)–(1.15) et les cas de base suivants :*

$$0^\square = 0 \, , \quad 1^\square = 0 \, , \qquad \forall a \in A \quad a^\square = a \ .$$

Il est maintenant clair que (1.13)–(1.15), avec (1.5)–(1.7) et (1.8) réécrit en :

$$(\mathsf{F}^*)^\bullet = (\mathsf{F}^\square)^* \ , \tag{1.8'}$$

est une nouvelle description inductive du calcul de $\mathsf{E}^\bullet$, à la différence que toute sous-expression de $\mathsf{E}^\square$ est en forme normale. Par exemple :

$$((a^*b^*)^*)^\square = a + b \qquad \text{et} \qquad (c\,(a^*b^*)^*)^\square = c\,(a+b)^* \ .$$

Cette nouvelle description donne, comme la description originale, un algorithme linéaire pour calculer $\mathsf{E}^\bullet$.

Comme dans la Proposition 6 pour l'opérateur $\circ$, nous pouvons donner des propriétés qui nous seront utiles par la suite pour les opérateurs $\square$ et $\bullet$.

Proposition 9 ([2]). *Soit* E *une expression rationnelle, alors on a :*

$$\mathsf{c}(\mathsf{E}^{\square}) = 0 \ , \tag{1.16}$$

$$\mathsf{c}(\mathsf{E}) = 0 \iff \mathsf{E}^{\bullet} = \mathsf{E}^{\square} \ , \tag{1.17}$$

$$\ell(\mathsf{E}^{\square}) = \ell(\mathsf{E}^{\bullet}) = \ell(\mathsf{E}) \ , \tag{1.18}$$

$$\mathsf{E} \textit{ en forme normale} \implies \mathsf{E} = \mathsf{E}^{\bullet} \ . \tag{1.19}$$

1.5 Multiplicités

Dans cette section nous allons définir le cadre pour l'étude des automates et des expressions rationnelles à multiplicités. Ces notions ne seront utilisées que dans les Chapitres 5 et 8. Toutes les expressions rationnelles et les automates rencontrés en dehors de ces chapitres devront être compris sous le sens booléen défini précédemment. Les définitions pour les séries formelles suivent celles données dans [8].

1.5.1 Séries formelles

Dans la suite on notera $\mathbb{K}$ un semi-anneau quelconque dont l'addition est notée $\oplus$ et la multiplication par une simple concaténation. Une fonction d'un monoïde M dans $\mathbb{K}$ peut être vue comme une somme formelle infinie de monômes dont les variables sont les éléments de M. C'est pour cela que l'on appelle *séries formelles* sur M à coefficients dans $\mathbb{K}$ les fonctions de M dans $\mathbb{K}$.

Soit s une série sur M à coefficients dans $\mathbb{K}$. On note $<s, x>$ le coefficient – ou la multiplicité – de $x \in M$ dans la série s. Le *support* de s, noté $\mathsf{supp}\,(s)$, est l'ensemble des éléments de M dont la multiplicité est différente de $0_{\mathbb{K}}$.

De manière plus précise nous utilisons par la suite des séries formelles sur le monoïde libre engendré par un alphabet. L'ensemble des séries formelles engendrées par le monoïde libre A^*, noté $\mathbb{K}\langle\langle A^* \rangle\rangle$, est un semi-anneau.

Dans ce cadre, la *série caractéristique* d'un langage L est la série s telle que $<s, u> = 1_{\mathbb{K}}$ pour les mots u de L et $<s, u> = 0_{\mathbb{K}}$ sinon.

Exemple 5 (*Ex. 1 cont.*). La série caractéristique du langage A^*bA^* est la série :

$$s = b + ab + ba + bb + aab + \dots$$

1.5.2 $\mathbb{K}$-automates et $\mathbb{K}$-représentations

Les séries formelles permettent d'imaginer des automates qui ne reconnaissent plus seulement un langage mais également un coefficient pour chaque

mot de ce langage, c'est-à-dire une série. Nous donnons une définition simple de tels automates :

Définition 7. *Soit A un alphabet. Un* $\mathbb{K}$-automate *sur A est un quintuplet $\mathcal{A} = \langle Q, A, E, I, T\rangle$ où :*
(i) *Q est un ensemble d'états ;*
(ii) *I est une fonction de Q dans $\mathbb{K}$ appelée fonction initiale. Pour $q \in Q$, si $I(q)$ est non nul alors q est initial avec pour valeur initiale $I(q)$;*
(iii) *T est une fonction de Q dans $\mathbb{K}$ appelée fonction finale.*
(iv) *E est l'ensemble des transitions, c'est-à-dire des arcs orientés entre états étiquetés par des monômes dont le support est une lettre.*

Le *comportement* d'un $\mathbb{K}$-automate $\mathcal{A}$ est la série $|\mathcal{A}|$ telle que le coefficient du mot u soit la somme des coefficients des calculs réussis du mot u dans $\mathcal{A}$. En particulier, si l'on considère les automates booléens définis auparavant comme des $\mathbb{B}$-automates, alors le comportement d'un tel automate est la série caractéristique du langage reconnu par l'automate. Les séries formelles qui sont le comportement de $\mathbb{K}$-automates finis sont appelées *séries $\mathbb{K}$-reconnaissables* .

Exemple 6 (*Ex. 1 cont.*)**.** Prenons $(\mathbb{Z}, +, \times)$ comme semi-anneau. Nous notons le monôme $1_{\mathbb{Z}}a$ par l'étiquette a. Le $\mathbb{Z}$-automate de la Figure 1.3 est tel qu'un mot u est l'étiquette d'autant de calculs que le nombre d'occurrences de b dans u. En effet, la transition du milieu, étiquetée par b peut être traversée une et une seule fois et chaque b implique un calcul différent. En outre chaque chemin a pour valeur 1. L'automate $\mathcal{A}_1$ 'compte' donc le nombre de b et réalise la série dont le support est A^*bA^* et telle que $\langle s, u\rangle = |u|_b$.

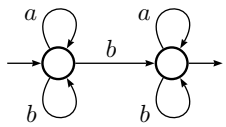


FIGURE 1.3 – Un $\mathbb{Z}$ automate, $\mathcal{A}_1$ sur l'alphabet $\{a, b\}$

Une $\mathbb{K}$-*représentation* de dimension n est un triplet (λ, μ, ν), où λ et ν sont respectivement, un vecteur ligne de dimension n et un vecteur colonne de dimension n sur $\mathbb{K}$, et μ est un morphisme de A^* dans $\mathbb{K}^{Q\times Q}$.

La série $s \in \mathbb{K}\langle\langle A^*\rangle\rangle$ réalisée par (λ, μ, ν) est définie par : pour tout $u \in A^*$, $\langle s, u\rangle = \lambda \cdot \mu(u) \cdot \nu$. Les $\mathbb{K}$-représentations sont en fait des objets similaires aux $\mathbb{K}$-automates : il est possible d'associer une représentation à chaque $\mathbb{K}$-automate : un automate dont les états sont $Q = \{q_1, \ldots, q_n\}$ est représenté par (λ, μ, ν) de dimension n si :

(i) λ_i est la valeur initiale de l'état q_i,
(ii) ν_i est la valeur finale de l'état q_i,
(iii) $(\mu(a))_{i,j}$ est le poids de la transition entre q_i et q_j étiquetée par a (si le poids est nul il n'y a pas de transition).

Chapitre 2

Termes dérivés

Dans ce chapitre nous donnons des définitions qui permettent de construire des automates finis à partir d'expressions rationnelles. La notion particulière abordée ici est la dérivation. Nous rappelons ainsi la première définition de dérivation sur les expressions rationnelles, donnée par Brzozowski dans [11]. Nous présentons ensuite une variante plus récente, les termes dérivés d'une expression, définie par Antimirov dans [5]. Le but de ces définitions est de nous permettre, dans le chapitre suivant, de définir les termes dérivés cassés, nouvelle variante, et dont la motivation sera trouvée dans [32] et développée dans le chapitre suivant.

Nous avons déjà vu l'automate des positions de [26] qui permet de construire un automate standard de la même taille que l'expression. La notion de dérivation est une autre manière de construire un automate à partir d'une expression rationnelle. La notion de dérivation d'expressions a été introduite en premier lieu dans [11]. La dérivation d'une expression est une construction qui permet de faire passer les calculs de quotients de langages rationnels au niveau des expressions rationnelles.

L'idée de la dérivation est de construire, pour une expression rationnelle E dénotant L et étant donné un mot u, une expression rationnelle qui dénote le langage $u^{-1}L$, quotient de L par u. L'importance de cette construction est qu'elle permet d'obtenir, sous certaines conditions, notamment l'associativité et la commutativité des sommes, un nombre fini de ces expressions dérivées. A partir de ce point il est possible de construire un automate fini déterministe qui reconnaît L.

Afin de s'épargner la complexité des calculs des expressions modulo la commutativité de la somme, Antimirov introduit une variante de cette dérivation dans [5]. Cette variante ne va plus construire une expression dérivée

par lettre, comme c'était le cas précédemment, mais un ensemble d'expressions rationnelles. L'automate que permet de construire cette variante, que nous appelons *automate des termes dérivés*, est un automate qui reconnaît toujours le même langage. L'automate alors construit n'est pas nécessairement déterministe et sa taille est plus petite que celle de l'automate des positions. En fait il a même été montré dans [18] que c'est un quotient de l'automate des positions.

En plus des définitions et propriétés intéressantes des dérivées et de termes dérivés d'expressions rationnelles, nous allons, dans ce chapitre, donner un résultat sur le comportement des opérations de dérivations alors définies par rapport au parenthésage des produits d'expressions rationnelles. En particulier si les termes dérivés sont invariants modulo l'associativité du produit, les dérivées ne le sont pas. Ces résultats permettent également de justifier notre choix de parenthésage à gauche par défaut.

2.1 Dérivation des expressions

Dans cette section nous rappelons la définition des expressions dérivées d'une expression rationnelle et de l'automate qui en découle.

2.1.1 Expression dérivée d'une expression

Définition 8 ([11])**.** *Soient* E *une expression rationnelle sur* A *et* a *une lettre de* A*.* L'expression dérivée[1] *de* E *par rapport à* a*, notée* $[a]^{-1}(\mathsf{E})$*, est l'expression rationnelle définie inductivement par :*

$$\begin{aligned}
[a]^{-1}(0) = [a]^{-1}(1) = [a]^{-1}(\mathrm{b}) = 0 \qquad & \forall b \in A,\ b \neq a \ , \\
[a]^{-1}(\mathrm{a}) = 1 \ , & \\
[a]^{-1}(\mathsf{F} + \mathsf{G}) = [a]^{-1}(\mathsf{F}) + [a]^{-1}(\mathsf{G}) \ , & \\
[a]^{-1}(\mathsf{F} \cdot \mathsf{G}) = ([a]^{-1}(\mathsf{F})) \cdot \mathsf{F} + \mathsf{c}(\mathsf{F})[a]^{-1}(\mathsf{G}) \ , & \\
[a]^{-1}(\mathsf{F}^*) = ([a]^{-1}(\mathsf{F})) \cdot \mathsf{F}^* \ , &
\end{aligned}$$

où il faut comprendre que $\mathsf{c}(\mathsf{F})[a]^{-1}(\mathsf{G})$ *vaut* $[a]^{-1}(\mathsf{G})$ *si le terme constant de* F *est non nul et* 0 *sinon.*

Nous avons choisi de noter $[a]^{-1}(\mathsf{E})$ l'expression dérivée par rapport à a de l'expression rationnelle E afin de n'avoir aucune ambiguïté avec les autres opérations de dérivation que nous définissons par la suite.

1. Appelée en anglais *derivative* dans [11].

Exemple 7. Posons $\mathsf{K}_1 = (b \cdot a^*) \cdot ((a+b)^*)$. Pour simplifier l'écriture, nous posons $\mathsf{K}'_1 = (a+b)^*$ et donc $\mathsf{K}_1 = (b \cdot a^*) \cdot \mathsf{K}'_1$. Les expressions dérivées par rapport à a et b de K_1 sont :

$$\begin{aligned}
[a]^{-1}(\mathsf{K}_1) &= [a]^{-1}(\mathrm{b} \cdot \mathrm{a}^*) \cdot \mathsf{K}'_1 + \mathsf{c}(b \cdot a^*)[a]^{-1}(\mathsf{K}_1\text{'}) \\
&= [a]^{-1}(\mathrm{b} \cdot \mathrm{a}^*) \cdot \mathsf{K}'_1 \\
&= ([a]^{-1}(\mathrm{b}) \cdot a^*) \cdot \mathsf{K}'_1 + \mathsf{c}(b)[a]^{-1}(\mathrm{a}^*) \\
&= (0 \cdot a^*) \cdot \mathsf{K}'_1 = 0 \\
[b]^{-1}(\mathsf{K}_1) &= ([b]^{-1}(\mathrm{b}) \cdot a^*) \cdot \mathsf{K}'_1 \\
&= a^* \cdot \mathsf{K}'_1
\end{aligned}$$

Si l'on dérive ce dernier résultat par a on obtient :

$$\begin{aligned}
[a]^{-1}(\mathrm{a}^* \cdot \mathsf{K}\text{'}_1) &= [a]^{-1}(\mathrm{a}^*) \cdot \mathsf{K}'_1 + \mathsf{c}(a^*)[a]^{-1}(\mathsf{K}_1\text{'}) \\
&= a^* \cdot \mathsf{K}'_1 + \mathsf{K}'_1
\end{aligned}$$

La notation choisie est justifiée par la propriété suivante :

Propriété 10. *Si* E *est une expression rationnelle sur* A *qui dénote le langage* L *et que* a *est une lettre de* A*, alors le langage dénoté par la dérivée* $[a]^{-1}(\mathsf{E})$ *de* E *par rapport à* a *dénote le langage* $a^{-1}L$*.*

Finalement nous pouvons étendre la dérivation aux mots :

Définition 9 ([11])**.** *Soient* E *une expression rationnelle sur* A*,* u *un mot non vide de* A^* *et* a *une lettre de* A*. L'expression dérivée de* E *par rapport à* ua*, notée* $[ua]^{-1}(\mathsf{E})$*, est l'expression rationnelle définie inductivement par :*

$$[ua]^{-1}(\mathsf{E}) = [a]^{-1}([u]^{-1}(\mathsf{E})) \quad .$$

L'ensemble des expressions dérivées d'une expression E *est l'ensemble des expressions dérivées de cette expression* E *par rapport à n'importe quel mot* u *non vide de* A^**.*

Exemple 8 (*Ex. 7 cont.*)**.** Comme nous l'avons déjà calculé :

$$[ba]^{-1}(\mathsf{K}_1) = [a]^{-1}(\mathrm{a}^* \cdot \mathsf{K}_1\text{'}) = a^* \cdot \mathsf{K}'_1 + \mathsf{K}'_1 \quad .$$

et nous avons :

$$\begin{aligned}
[bb]^{-1}(\mathsf{K}_1) &= [b]^{-1}(\mathrm{a}^* \cdot \mathsf{K}_1\text{'}) = \mathsf{K}'_1 \quad , \\
[baa]^{-1}(\mathsf{K}_1) &= [a]^{-1}(\mathrm{a}^* \cdot \mathsf{K}_1 + \mathsf{K}_1\text{'}) \\
&= a^* \cdot \mathsf{K}'_1 + \mathsf{K}'_1 + \mathsf{K}'_1 \quad .
\end{aligned}$$

Propriété 11. *Si* E *est une expression rationnelle sur* A *qui dénote le langage* L *et que* u *est un mot non vide de* A^**, alors le langage dénoté par l'expression dérivée* $[u]^{-1}(\mathsf{E})$ *de* E *par rapport à* u *dénote le langage* $u^{-1}L$*.*

2.1.2 Nombre d'expressions dérivées

La définition donnée peut produire un nombre infini d'expressions dérivées pour une expression rationnelle E donnée, comme le montre l'exemple de l'expression K_1 puisque la dérivation par rapport au mot ba^n donne :

$$[ba^n]^{-1}(\mathsf{K}_1) = a^* \cdot \mathsf{K}_1' + \mathsf{K}_1' + \cdots + \mathsf{K}_1' \ ,$$

avec une somme de n fois K_1'.

Cependant si les calculs sur les expressions rationnelles sont réalisés modulo l'associativité de la somme, la commutativité de la somme et l'idempotence de la somme, alors nous avons le théorème suivant :

Théorème 12 ([11])**.** *L'ensemble des expressions dérivées d'une expression rationnelle est fini modulo les identités* $\mathbf{A_s}$*,* $\mathbf{C}$ *et* $\mathbf{I}$ *et il est calculable.*

Exemple 9 (*Ex. 7 cont.*)**.** Modulo $\mathbf{A_sCI}$, les expressions dérivées de K_1 sont : 0, K_1, $a^* \cdot \mathsf{K}_1$, K_1 et $a^* \cdot \mathsf{K}_1 + \mathsf{K}_1$.

2.1.3 Automates des expressions dérivées

Le théorème précédent permet de calculer un ensemble fini d'expressions dérivées. Cela permet donc de définir l'automate fini suivant dont les états sont les expressions dérivées modulo $\mathbf{A_sCI}$ d'une expression :

Définition 10. *L'automate* $\mathcal{B}_{\mathsf{E}}$ *des expressions dérivées d'une expression rationnelle* E *est l'automate déterministe tel que :*
(i) *l'ensemble des états est l'ensemble des expressions dérivées modulo* $\mathbf{A_sCI}$ *de* E*;*
(ii) *l'état initial est l'expression* E*;*
(iii) *les états finaux sont les états dont le terme constant est non nul;*
(iv) *si* F *est une expression dérivée de* E *alors* $(\mathsf{F}, a, [a]^{-1}(\mathsf{F}))$ *est une transition de* $\mathcal{B}_{\mathsf{E}}$*.*

De la Propriété 11 découle directement :

Proposition 13 ([11])**.** *Si* E *est une expression rationnelle dénotant* L*, alors* $\mathcal{B}_{\mathsf{E}}$ *reconnaît* L*.*

Exemple 10 (*Ex. 7 cont.*)**.** L'automate des expressions dérivées de K_1 est montré sur la Figure 2.1.

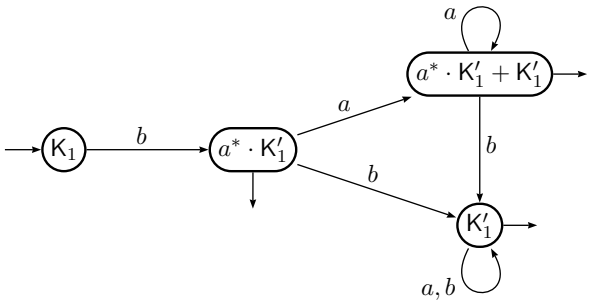


FIGURE 2.1 – Automate des expressions dérivées de K_1

2.2 Termes dérivés d'une expression

L'automate des expressions dérivées d'une expression E reconnaît le langage L dénoté par E, cependant sa taille peut être exponentielle en la taille de l'expression et présuppose également des calculs modulo $\mathbf{A_sCI}$ très coûteux.

Nous décrivons dans cette section une modification de l'opération de dérivation vue dans la section précédente, due à Antimirov dans [5] et qui permet de construire un automate – qui peut ne pas être déterministe – qui ne soit pas plus gros que l'automate standard de cette expression.

L'idée pour atteindre ce but est de remplacer la dérivée d'une expression par un ensemble d'expressions dont chaque élément est un terme de la somme que définit l'expression dérivée. Cela permet, d'une certaine manière, de réaliser 'à la volée' l'opération de modulo $\mathbf{A_sCI}$ là où elle est nécessaire.

2.2.1 Définition et propriétés

Nous allons maintenant définir les termes dérivés d'une expression rationnelle. Ces termes dérivés ont été définis en [5] sous le nom de *partial derivatives*. Nous n'utilisons pas cette terminologie mais celle de [43, I.5.2] pour les raisons évoquées dans cette dernière référence : les 'derivatives' de [11] sont déjà partielles dans le sens où elles sont définies par rapport à une lettre. La terminologie 'terme dérivé' est utilisée car les termes dérivés sont les termes de la somme définie par l'expression dérivée.

Définition 11 ([5])**.** *Soient* E *une expression rationnelle sur* A *et* a *une lettre de* A*. La* dérivation d'une expression *de* E *par rapport à* a*, notée* $\frac{\partial}{\partial a}\mathsf{E}$*, est*

un ensemble d'expressions rationnelles sur A, défini récursivement par :

$$\frac{\partial}{\partial a}\mathsf{0} = \frac{\partial}{\partial a}\mathsf{1} = \emptyset, \qquad \forall b \in A \quad \frac{\partial}{\partial a} b = \begin{cases} \{1\} & \textit{si} \quad b = a\,, \\ \emptyset & \textit{sinon}, \end{cases} ,$$

$$\frac{\partial}{\partial a}(\mathsf{F} + \mathsf{G}) = \frac{\partial}{\partial a}\mathsf{F} \cup \frac{\partial}{\partial a}\mathsf{G} \ , \tag{2.1}$$

$$\frac{\partial}{\partial a}(\mathsf{F} \cdot \mathsf{G}) = \left(\frac{\partial}{\partial a}\mathsf{F}\right) \cdot \mathsf{G} \cup \mathsf{c}(\mathsf{F}) \frac{\partial}{\partial a}\mathsf{G} \ , \tag{2.2}$$

$$\frac{\partial}{\partial a}(\mathsf{F}^*) = \left(\frac{\partial}{\partial a}\mathsf{F}\right) \cdot \mathsf{F}^* \ . \tag{2.3}$$

L'Equation (2.2) doit être vue comme :

$$\frac{\partial}{\partial a}(\mathsf{F} \cdot \mathsf{G}) = \begin{cases} \left(\frac{\partial}{\partial a}\mathsf{F}\right) \cdot \mathsf{G} \cup \frac{\partial}{\partial a}\mathsf{G} \ , & \text{si} \quad \mathsf{c}(\mathsf{F}) = 1\,, \\ \left(\frac{\partial}{\partial a}\mathsf{F}\right) \cdot \mathsf{G} \ , & \text{si} \quad \mathsf{c}(\mathsf{F}) = 0\,. \end{cases}$$

Et l'induction des équations (2.1–2.3) doit être comprise en étendant la dérivation de manière additive – comme toutes les opérations de dérivation, et où, ici, l'addition est l'opération d'union sur les ensembles – et en ajoutant la distribution (à droite) de l'opérateur $\cdot$ sur les ensembles d'expressions :

$$\frac{\partial}{\partial a} X = \bigcup_{\mathsf{E} \in X} \frac{\partial}{\partial a}\mathsf{E}\,, \qquad (X) \cdot \mathsf{F} = \bigcup_{\mathsf{E} \in X} (\mathsf{E} \cdot \mathsf{F}) \ ,$$

où X est un ensemble d'expressions rationnelles.

Nous rappelons également que les équations sont définies, comme toujours, modulo les identités triviales $\mathbf{T}$.

Exemple 11. Soit $\mathsf{E}_1 = (a^* + b^*) \cdot (a \cdot (a^* + b^*))$ sur $\{a, b\}$. Nous posons $\mathsf{F}_1 = (a^* + b^*)$, c'est-à-dire :

$$\mathsf{E}_1 = \mathsf{F}_1 \cdot (a \cdot \mathsf{F}_1) \ .$$

Il vient :

$$\frac{\partial}{\partial a}\mathsf{E}_1 = \left(\frac{\partial}{\partial a}\mathsf{F}_1\right) \cdot (a \cdot \mathsf{F}_1) \cup \mathsf{c}(\mathsf{F}_1)\frac{\partial}{\partial a}(a \cdot \mathsf{F}_1) = \{a^* \cdot (a \cdot \mathsf{F}_1)\,,\, \mathsf{F}_1\} \ .$$
$$\frac{\partial}{\partial b}\mathsf{E}_1 = \left(\frac{\partial}{\partial b}\mathsf{F}_1\right) \cdot (a \cdot \mathsf{F}_1) \cup \mathsf{c}(\mathsf{F}_1)\frac{\partial}{\partial b}(a \cdot \mathsf{F}_1) = \{b^* \cdot (a \cdot \mathsf{F}_1)\} \ .$$

Pour tout mot non vide u de A^*, la dérivation d'une expression E sur A par rapport au mot ua est définie par induction par :

$$\frac{\partial}{\partial ua}\mathsf{E} = \frac{\partial}{\partial a}\left(\frac{\partial}{\partial u}\mathsf{E}\right) \ . \tag{2.4}$$

De cette définition, il apparaît que si $ua = bv$ alors :

$$\frac{\partial}{\partial a}\left(\frac{\partial}{\partial u}\mathsf{E}\right) = \frac{\partial}{\partial v}\left(\frac{\partial}{\partial b}\mathsf{E}\right) \ . \tag{2.5}$$

Définition 12. *Soit* E *une expression rationnelle sur* A*. Les éléments des ensembles* $\frac{\partial}{\partial u}\mathsf{E}$ *pour chaque mot* u *non vide de* A^* *sont appelés les* vrais termes dérivés *de* E *et nous notons* $\mathrm{TD}(\mathsf{E})$ *l'ensemble des vrais termes dérivés de* E *:*

$$\mathrm{TD}(\mathsf{E}) = \bigcup_{f\in A^+} \frac{\partial}{\partial f}\mathsf{E} \ . \tag{2.6}$$

Les termes dérivés *de* E *sont les vrais termes dérivés et l'expression* E *elle-même –qui peut être vue comme la dérivation par rapport au mot vide – et nous notons* $\mathrm{D}(\mathsf{E})$ *leur ensemble :*

$$\mathrm{D}(\mathsf{E}) = \mathrm{TD}(\mathsf{E}) \cup \{\mathsf{E}\} \ .$$

De cette définition il découle que si E est une constante alors $\mathrm{TD}(\mathsf{E}) = \emptyset$ et donc $\mathrm{D}(\mathsf{E}) = \{\mathsf{E}\}$.

Remarque 1. Afin de ne pas faire de différence entre les termes dérivés et les vrais termes dérivés, il est possible d'envisager une dérivation par rapport au mot vide d'une expression rationnelle E dont le résultat serait E. Cette solution n'est pas acceptable pour nous car elle n'est pas compatible avec l'idée que la dérivation d'Antimirov 'sépare' les expressions somme. En fait si l'on devait définir une dérivation par rapport au mot vide elle correspondrait au cassage d'une expression que nous définissons dans le chapitre suivant.

Afin d'être constant dans les définitions pour les différentes dérivations que nous étudions, nous choisissons donc pas non plus de définir une dérivation par rapport au mot vide pour les termes dérivés cassés.

Exemple 12 (*Ex. 11 cont.*)**.** La dérivation de E_1 par rapport aux mots de

longueur 2 donne :

$$\begin{aligned}
\frac{\partial}{\partial aa}\mathsf{E}_1 &= \frac{\partial}{\partial a}\left(\{a^* \cdot (a \cdot \mathsf{F}_1),\, \mathsf{F}_1\}\right) = \{a^* \cdot (a \cdot \mathsf{F}_1),\, \mathsf{F}_1\} \cup \{a^*\} \ , \\
\frac{\partial}{\partial ab}\mathsf{E}_1 &= \frac{\partial}{\partial b}\left(\{a^* \cdot (a \cdot \mathsf{F}_1),\, \mathsf{F}_1\}\right) = \{b^*\} \ , \\
\frac{\partial}{\partial ba}\mathsf{E}_1 &= \frac{\partial}{\partial a}\left(\{b^* \cdot (a \cdot \mathsf{F}_1)\}\right) = \{\mathsf{F}_1\} \ , \\
\frac{\partial}{\partial bb}\mathsf{E}_1 &= \frac{\partial}{\partial b}\left(\{b^* \cdot (a \cdot \mathsf{F}_1)\}\right) = \{b^* \cdot (a \cdot \mathsf{F}_1)\} \ .
\end{aligned}$$

Il n'y a pas de nouveaux termes dérivés trouvés par dérivation par rapport à des mots plus longs et donc :

$$\begin{aligned}
\mathrm{TD}\left(\mathsf{E}_1\right) &= \{a^* \cdot (a \cdot \mathsf{F}_1),\, b^* \cdot (a \cdot \mathsf{F}_1),\, \mathsf{F}_1,\, a^*,\, b^*\} \ , \\
\mathrm{D}\left(\mathsf{E}_1\right) &= \{\mathsf{E}_1,\, a^* \cdot (a \cdot \mathsf{F}_1),\, b^* \cdot (a \cdot \mathsf{F}_1),\, \mathsf{F}_1,\, a^*,\, b^*\} \ .
\end{aligned}$$

La justification de cette variante de la dérivée d'une expression rationnelle réside dans le fait que l'ensemble des termes dérivés est fini et permet de construire – comme nous le verrons par la suite – un automate au moins aussi petit que l'automate standard :

Théorème 14 ([5])**.** *Le nombre de termes dérivés d'une expression rationnelle* E *est majoré par* $\ell(\mathsf{E}) + 1$.

Nous donnons maintenant des propriétés sur l'ensemble des vrais termes dérivés de somme, de produit et d'étoile d'expressions rationnelles :

Proposition 15 ([31])**.** *Soient* F *et* G *deux expression rationnelles. Nous avons :*

$$\mathrm{TD}\left((\mathsf{F} + \mathsf{G})\right) = \mathrm{TD}\left(\mathsf{F}\right) \cup \mathrm{TD}\left(\mathsf{G}\right) \ , \tag{2.7}$$

$$\mathrm{TD}\left((\mathsf{F} \cdot \mathsf{G})\right) = \left(\mathrm{TD}\left(\mathsf{F}\right)\right) \cdot \mathsf{G} \cup \mathrm{TD}\left(\mathsf{G}\right) \ , \tag{2.8}$$

$$\mathrm{TD}\left((\mathsf{F}^*)\right) = \left(\mathrm{TD}\left(\mathsf{F}\right)\right) \cdot \mathsf{F}^* \ . \tag{2.9}$$

Démonstration. Nous allons prouver cette proposition par induction sur la profondeur des expressions rationnelles.

Dans le cas où l'expression E est une somme $\mathsf{F} + \mathsf{G}$, montrons :

$$\forall u \in A^+ \qquad \frac{\partial}{\partial u}\left(\mathsf{F} + \mathsf{G}\right) = \frac{\partial}{\partial u}\mathsf{F} \cup \frac{\partial}{\partial u}\mathsf{G} \ . \tag{2.10}$$

L'équation (2.10) est l'équivalent de (2.1) pour les mots u de longueur supérieure à 1 :

$$\begin{aligned}
&\forall u \in A^+ \,,\ \forall a \in A \\
&\frac{\partial}{\partial ua}(\mathsf{F}+\mathsf{G}) = \frac{\partial}{\partial a}\left(\frac{\partial}{\partial u}(\mathsf{F}+\mathsf{G})\right) = \frac{\partial}{\partial a}\left(\frac{\partial}{\partial u}\mathsf{F} \cup \frac{\partial}{\partial u}\mathsf{G}\right) \\
&\qquad = \frac{\partial}{\partial a}\left(\frac{\partial}{\partial u}\mathsf{F}\right) \cup \frac{\partial}{\partial a}\left(\frac{\partial}{\partial u}\mathsf{G}\right) = \frac{\partial}{\partial ua}\mathsf{F} \cup \frac{\partial}{\partial ua}\mathsf{G} \ .
\end{aligned}$$

En passant à l'union de tous les mots non vides de A^*, cette dernière équation permet d'établir (2.7) par hypothèse d'induction sur F et G.

Dans le cas où E est un produit $\mathsf{F}\cdot\mathsf{G}$, montrons :

$$\forall u \in A^+ \quad \frac{\partial}{\partial u}(\mathsf{F}\cdot\mathsf{G}) = \left(\frac{\partial}{\partial u}\mathsf{F}\right)\cdot\mathsf{G} \cup \left(\bigcup_{\substack{v,w\in A^+\\ vw=u}} \mathsf{c}(\frac{\partial}{\partial v}\mathsf{F})\frac{\partial}{\partial w}\mathsf{G}\right) \cup \mathsf{c}(\mathsf{F})\frac{\partial}{\partial u}\mathsf{G} \ . \tag{2.11}$$

En effet :

$$\begin{aligned}
&\forall u \in A^+ \,, \forall a \in A \\
&\frac{\partial}{\partial ua}(\mathsf{F}\cdot\mathsf{G}) = \frac{\partial}{\partial a}\left(\frac{\partial}{\partial u}(\mathsf{F}\cdot\mathsf{G})\right) \\
&\qquad = \frac{\partial}{\partial a}\left(\left(\frac{\partial}{\partial u}\mathsf{F}\right)\cdot\mathsf{G} \cup \left(\bigcup_{\substack{v,w\in A^+\\ vw=u}} \mathsf{c}(\frac{\partial}{\partial v}\mathsf{F})\frac{\partial}{\partial w}\mathsf{G}\right) \cup \mathsf{c}(\mathsf{F})\frac{\partial}{\partial u}\mathsf{G}\right) \\
&\qquad = \left(\frac{\partial}{\partial a}\left(\frac{\partial}{\partial u}\mathsf{F}\right)\right)\cdot\mathsf{G} \cup \mathsf{c}(\frac{\partial}{\partial u}\mathsf{F})\frac{\partial}{\partial a}\mathsf{G} \\
&\qquad\qquad \cup \left(\bigcup_{\substack{v,w\in A^+\\ vw=u}} \mathsf{c}(\frac{\partial}{\partial v}\mathsf{F})\,\frac{\partial}{\partial a}\left(\frac{\partial}{\partial w}\mathsf{G}\right)\right) \\
&\qquad\qquad \cup \mathsf{c}(\mathsf{F})\,\frac{\partial}{\partial a}\left(\frac{\partial}{\partial u}\mathsf{G}\right) \\
&\qquad = \left(\frac{\partial}{\partial ua}\mathsf{F}\right)\cdot\mathsf{G} \cup \left(\bigcup_{\substack{v,w\in A^+\\ vw=ua}} \mathsf{c}(\frac{\partial}{\partial v}\mathsf{F})\frac{\partial}{\partial w}\mathsf{G}\right) \cup \mathsf{c}(\mathsf{F})\frac{\partial}{\partial ua}\mathsf{G} \ .
\end{aligned}$$

Comme F n'est pas une constante, il existe un mot $v \in A^+$ tel que : $\mathsf{c}(\frac{\partial}{\partial v}\mathsf{F}) = 1$ et donc, par passage à l'union sur tous les mots non vides et par hypothèse d'induction sur F et G, l'équation(2.8) est obtenue.

Dans le cas de l'étoile F^* nous ne calculons pas exactement l'expression pour $\frac{\partial}{\partial f}(\mathsf{F}^*)$, nous montrons la double inclusion :

$$\forall u \in A^+ \qquad \left(\frac{\partial}{\partial u}\mathsf{F}\right)\cdot\mathsf{F}^* \subseteq \frac{\partial}{\partial u}(\mathsf{F}^*) \subseteq \bigcup_{\substack{w\in A^+\\ vw=u}} \left(\frac{\partial}{\partial w}\mathsf{F}\right)\cdot\mathsf{F}^* \ . \qquad (2.12)$$

En effet :

$$\begin{aligned}
&\forall u \in A^+ ,\ \forall a \in A\\
&\quad \frac{\partial}{\partial ua}(\mathsf{F}^*) = \frac{\partial}{\partial a}\left(\frac{\partial}{\partial u}(\mathsf{F}^*)\right)\\
&\qquad\qquad \supseteq \frac{\partial}{\partial a}\left(\left(\frac{\partial}{\partial u}\mathsf{F}\right)\cdot\mathsf{F}^*\right)\\
&\qquad\qquad \supseteq \left(\frac{\partial}{\partial a}\left(\frac{\partial}{\partial u}\mathsf{F}\right) \cup \mathsf{c}(\frac{\partial}{\partial u}\mathsf{F})\frac{\partial}{\partial a}\mathsf{F}\right)\cdot\mathsf{F}^*\\
&\qquad\qquad \supseteq \left(\frac{\partial}{\partial ua}\mathsf{F}\right)\cdot\mathsf{F}^* \ .
\end{aligned}$$

$\forall u \in A^+ , \forall a \in \mathsf{RatE(A)}$

$$\begin{aligned}
\frac{\partial}{\partial ua}(\mathsf{F}^*) &\subseteq \frac{\partial}{\partial a}\left(\bigcup_{\substack{w\in A^+\\ vw=u}} \left(\frac{\partial}{\partial w}\mathsf{F}\right)\cdot\mathsf{F}^*\right) = \bigcup_{\substack{w\in A^+\\ vw=u}} \left(\left(\frac{\partial}{\partial wa}\mathsf{F}\right)\cdot\mathsf{F}^* \cup \mathsf{c}(\frac{\partial}{\partial w}\mathsf{F})\frac{\partial}{\partial a}\mathsf{F}\cdot\mathsf{F}^*\right)\\
&\subseteq \bigcup_{\substack{w\in A^+\\ vw=ua}} \left(\frac{\partial}{\partial w}\mathsf{F}\right)\cdot\mathsf{F}^* \ .
\end{aligned}$$

Lorsqu'on prend l'union pour tous les mots u de A^+ les deux extrémités de la double inclusion sont égales à $\mathrm{TD}(\mathsf{F})\cdot\mathsf{F}^*$, ce qui finit d'établir la proposition. □

Les trois équations de la Proposition 15 permettent de donner une définition alternative et inductive de l'ensemble des vrais termes dérivés – et donc également de l'ensemble des termes dérivés –. C'est pourquoi nous avons distingué dans la terminologie l'opération de dérivation et les termes dérivés,

qui peuvent être calculés différemment – en fait nous verrons même par la suite, que dans le cas des expressions à multiplicités, les deux définitions ne coïncident plus et c'est la définition inductive qu'il convient de choisir –.

Corollaire 16. *L'ensemble* $\mathrm{TD}((\mathsf{E}))$ *des vrais termes dérivés d'une expression rationnelle* E *sur* A *peut être calculé par induction sur la profondeur de* E *en utilisant les équations (2.7)–(2.9) et les trois cas de base suivants :*

$$\mathrm{TD}(\mathsf{0}) = \emptyset\,, \quad \mathrm{TD}(\mathsf{1}) = \emptyset\,, \quad \forall a \in A \quad \mathrm{TD}(a) = \{\mathsf{1}\} \quad .$$

Définition 13. *Soit* E *une expression rationnelle, on appelle* concaténation à droite *de sous-expressions de* E *une expression qui est soit une sous-expression de* E *soit de la forme* $\mathsf{F} \cdot \mathsf{G}$ *où* G *est une sous-expression de* E *et* F *est une concaténation à droite de sous-expressions de* E.

La définition inductive de l'ensemble des termes dérivés induite par le Corollaire 16 donne également une preuve directe de la proposition suivante :

Proposition 17 ([18])**.** *Les termes dérivés d'une expression* E *sont des concaténations à droite de sous-expressions de* E.

Remarque 2. Cette proposition n'est pas donnée en terme de concaténations à droite dans [18] car dans ce papier le produit d'expressions rationnelles est associatif, ce qui, comme nous allons le /ontrer dans la section suivante, est une hypothèse valide mais que nous avons pas retenue dans un soucis de généralité.

2.2.2 Automate des termes dérivés

Comme le Théorème 14 montre qu'il y a un nombre fini de termes dérivés, l'automate que l'on peut construire, comme pour les expressions dérivées, avec les termes dérivés comme états et dont les transitions sont données par l'opération de dérivation est fini :

Définition 14 ([5])**.** L'automate des termes dérivés $\mathcal{D}(\mathsf{E})$ *d'une expression rationnelle* E *est l'automate tel que :*
(i) *l'ensemble des états est l'ensemble* $\mathrm{D}(\mathsf{E})$ *;*
(ii) *l'état initial est l'expression* E *;*
(iii) *les états finaux sont les états dont le terme constant est non nul ;*
(iv) *soient* K_1 *et* K_2 *deux termes dérivés, la transition* $(\mathsf{K}_1, a, \mathsf{K}_2)$ *existe pour une lettre* $a \in A$ *donnée si et seulement si* $\mathsf{K}_2 \in \frac{\partial}{\partial a}\mathsf{K}_1$.

Cette définition se justifie par :

Proposition 18 ([5]). *Le langage reconnu par l'automate* $\mathcal{D}(\mathsf{E})$ *est le langage dénoté par* E.

Exemple 13 (*Ex. 11 cont.*). La Figure 2.2 montre l'automate $\mathcal{D}(\mathsf{E}_1)$ des termes dérivés de E_1.

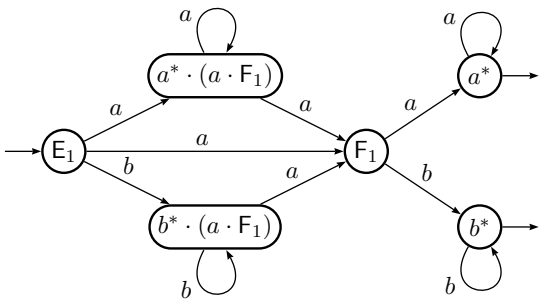


FIGURE 2.2 – $\mathcal{D}(\mathsf{E}_1)$

Comme nous avons vu, l'automate des termes dérivés est au plus aussi grand que l'automate standard. Plus précisément le résultat suivant lie l'automate des termes dérivés avec l'automate des positions :

Proposition 19 ([18]). *L'automate des termes dérivés* $\mathcal{D}(\mathsf{E})$ *de l'expression rationnelle* E *est un quotient de l'automate des positions* E.

2.3 Dérivation et associativité du produit

Il est courant de noter le produit de plusieurs expressions rationnelles E, F et G comme si ce produit était associatif $\mathsf{E}\cdot\mathsf{F}\cdot\mathsf{G}$. Ceci se comprend par le fait que les opérations habituelles sur les expressions rationnelles sont invariantes peut importe le parenthésage choisi.

Dans cette section nous montrons que les termes dérivés sont invariants modulo $\mathbf{A_p}$, c'est-à-dire que les termes dérivés des expressions parenthésées à droite et à gauche sont égaux par l'associativité du produit. Nous allons également montrer que, sans l'associativité du produit, le parenthésage à gauche donne au plus autant de termes dérivés que le parenthésage à droite de l'expression rationnelle.

Finalement une étude similaire montre que les expressions dérivées ne sont pas invariantes modulo $\mathbf{A_p}$ et qu'il n'y a pas de comparaison possible du nombre d'expressions dérivées entre le parenthésage à gauche ou à droite d'un produit donné.

2.3.1 Termes dérivés

Montrons tout d'abord que les termes dérivés conservent l'égalité modulo l'associativité du produit, c'est-à-dire :

Proposition 20. *Soient* E, F *et* G *des expressions rationnelles. Il vient alors*

$$\mathrm{D}\left((\mathsf{E}\cdot\mathsf{F})\cdot\mathsf{G}\right) \equiv_{\mathbf{A_p}} \mathrm{D}\left(\mathsf{E}\cdot(\mathsf{F}\cdot\mathsf{G})\right) \quad .$$

Démonstration. Cette proposition se montre directement par la définition inductive des termes dérivés :

$$\begin{aligned}
\mathrm{D}\left((\mathsf{E}\cdot\mathsf{F})\cdot\mathsf{G}\right) &= \mathrm{TD}\left((\mathsf{E}\cdot\mathsf{F})\cdot\mathsf{G}\right) \cup (\mathsf{E}\cdot\mathsf{F})\cdot\mathsf{G} \ , \\
&= \mathrm{TD}\left(\mathsf{E}\cdot\mathsf{F}\right)\cdot\mathsf{G} \cup \mathrm{TD}\left(\mathsf{G}\right) \cup (\mathsf{E}\cdot\mathsf{F})\cdot\mathsf{G} \ , \\
&= \left(\mathrm{TD}\left(\mathsf{E}\right)\cdot\mathsf{F}\right)\cdot\mathsf{G} \cup \mathrm{TD}\left(\mathsf{F}\right)\cdot\mathsf{G} \cup \mathrm{TD}\left(\mathsf{G}\right) \cup (\mathsf{E}\cdot\mathsf{F})\cdot\mathsf{G} \ , \\
&\equiv_{\mathbf{A_p}} \mathrm{TD}\left(\mathsf{E}\right)\cdot(\mathsf{F}\cdot\mathsf{G}) \cup \mathrm{TD}\left(\mathsf{F}\right)\cdot\mathsf{G} \cup \mathrm{TD}\left(\mathsf{G}\right) \cup \mathsf{E}\cdot(\mathsf{F}\cdot\mathsf{G}) \ , \\
&\equiv_{\mathbf{A_p}} \mathrm{D}\left(\mathsf{E}\cdot(\mathsf{F}\cdot\mathsf{G})\right) \quad .
\end{aligned}$$

□

Remarque 3. Si l'on veut être plus précis c'est en fait l'opération de dérivation elle-même qui commute modulo $\mathbf{A_p}$ et pas uniquement l'ensemble des termes dérivés.

Pourtant, si l'on ne considère pas que les expressions sont identiques par associativité du produit, les ensembles de termes dérivés d'une même expression avec un parenthésage droit ou un parenthésage gauche peuvent ne pas avoir le même cardinal comme le montre l'exemple suivant :

Exemple 14. Soit l'expression produit non (totalement) parenthésée suivante :

$$a\,b(c(a\,b))^*$$

nous observons le résultat de la dérivation de deux expressions obtenues respectivement par parenthésage à droite et par parenthésage à gauche : $a\,(b(c(a\,b))^*)$ et $(a\,b)\,(c(a\,b))^*$:

$$\begin{aligned}
&\mathrm{D}\left(a\,(b(c(a\,b))^*)\right) = \{a\,(b(c(a\,b))^*)\,,\ b(c(a\,b))^*, (c(a\,b))^*\,,\ (a\,b)\,(c(a\,b))^*\} \ . \\
&\mathrm{D}\left((a\,b)\,(c(a\,b))^*\right) = \{(a\,b)\,(c(a\,b))^*\,,\ b(c(a\,b))^*, (c(a\,b))^*\} \ .
\end{aligned}$$

Cet exemple est un cas particulier du théorème :

Théorème 21 ([2])**.** *Soient* E*,* F *et* G *des expressions rationnelles, alors :*

$$\operatorname{card}(\mathrm{D}((\mathsf{E}\cdot\mathsf{F})\cdot\mathsf{G})) \leqslant \operatorname{card}(\mathrm{D}(\mathsf{E}\cdot(\mathsf{F}\cdot\mathsf{G}))) \quad .$$

Démonstration. Par la définition inductive de l'ensemble des termes dérivés, donnée par la Proposition 15, nous avons :

$$\mathrm{D}((\mathsf{E}\cdot\mathsf{F})\cdot\mathsf{G}) = (\mathrm{TD}(\mathsf{E})\cdot\mathsf{F})\cdot\mathsf{G} \cup \mathrm{TD}(\mathsf{F})\cdot\mathsf{G} \cup \mathrm{TD}(\mathsf{G}) \cup (\mathsf{E}\cdot\mathsf{F})\cdot\mathsf{G} \text{ et}$$
$$\mathrm{D}(\mathsf{E}\cdot(\mathsf{F}\cdot\mathsf{G})) = \mathrm{TD}(\mathsf{E})\cdot(\mathsf{F}\cdot\mathsf{G}) \cup \mathrm{TD}(\mathsf{F})\cdot\mathsf{G} \cup \mathrm{TD}(\mathsf{G}) \cup \mathsf{E}\cdot(\mathsf{F}\cdot\mathsf{G}) \quad .$$

Posons :

$$X = \mathrm{TD}(\mathsf{F})\cdot\mathsf{G} \quad ,$$
$$Y = \mathrm{TD}(\mathsf{G}) \quad .$$

Il y a évidemment une bijection entre les éléments de $(\mathrm{TD}(\mathsf{E})\cdot\mathsf{F})\cdot\mathsf{G}$ et ceux de $\mathrm{TD}(\mathsf{E})\cdot(\mathsf{F}\cdot\mathsf{G})$ qui ont donc le même cardinal.

Il suffit donc, pour montrer le théorème, de montrer que :

$$\mathrm{TD}(\mathsf{E})\cdot(\mathsf{F}\cdot\mathsf{G}) \cap (X \cup Y) = \emptyset \quad \text{et} \quad \mathsf{E}\cdot(\mathsf{F}\cdot\mathsf{G}) \notin (X \cup Y) \quad .$$

Car alors, en effet, $\mathrm{D}((\mathsf{E}\cdot\mathsf{F})\cdot\mathsf{G})$ a au plus autant d'éléments que $\mathrm{D}(\mathsf{E}\cdot(\mathsf{F}\cdot\mathsf{G}))$.

Comme F n'est pas 1 – sinon E ne serait pas réduite par les identités triviales – une expression produit avec pour terme de droite $\mathsf{F}\cdot\mathsf{G}$ ne peut pas être dans X dont les éléments sont des produits avec pour terme de droite G.

D'un autre côté, comme les termes dérivés de G sont des concaténations de sous-expressions de G et comme $\mathsf{F}\cdot\mathsf{G}$ ne peut pas être une sous-expression de G, l'intersection avec Y est également vide.

Comme l'intersection de l'ensemble des expressions qui sont un produit dont le terme de droite est $\mathsf{F}\cdot\mathsf{G}$ avec X et Y est vide, les deux conditions posées ci-dessus sont vérifiées et donc le théorème est prouvé. □

Remarque 4. C'est ce résultat qui nous a fait choisir le parenthésage à gauche comme parenthésage par défaut pour les produits d'expressions rationnelles. En effet les concaténations à droite sont parenthésées naturellement à gauche.

2.3.2 Expressions dérivées

L'exemple suivant permet de montrer que, contrairement aux termes dérivés, les expressions dérivées ne commutent pas à l'associativité du produit.

Exemple 15. Soit l'expression $a^*\,a\,a^*$, on a :

$$[a]^{\text{-}1}((\text{a}^*\text{a})\text{a}) = (a^*a + 1)\cdot a \text{ et}$$
$$[a]^{\text{-}1}(\text{a}^*(\text{aa}^*)) = a^*(aa^*) + a^* \quad .$$

Ces deux expressions ne sont pas égales modulo $\mathbf{A_p}$.

En déroulant l'exemple précédent nous pouvons voir que $(a^*a)a^*$ a plus d'expressions dérivées que $a^*(aa^*)$. L'Exemple 14 montre que ce résultat est inversé pour l'expression $a\,b\,(c\,(a\,b))^*$.

Il n'y a donc pas, dans le cas des expressions dérivées modulo $\mathbf{A_sCI}$ un choix évident pour réduire la taille de l'automate, entre un parenthésage à droite et un parenthésage à gauche.

Chapitre 3

Termes dérivés cassés

Nous avons vu dans le chapitre précédent que les termes dérivés permettent de construire un automate fini plus petit que l'automate standard et qui est même un quotient de celui-ci. Dans ce chapitre nous introduisons une nouvelle variante de la dérivation, la dérivation cassante, qui casse les sommes les plus à gauche d'une concaténation.

La motivation pour cette nouvelle variante n'est pas exclusivement la construction d'un automate petit à partir d'une expression, bien que nous montrons que, sous certaines conditions, la borne sur l'ensemble des termes dérivés tient aussi pour les *termes dérivés cassés*. La motivation vient de [32] – ou plus précisément [33], la version corrigée du précédent –. Il y est montré qu'il est possible en utilisant les termes dérivés cassés, de reconstruire un automate à partir d'une expression calculée par élimination sur celui-ci. Plus précisément, si $\mathcal{A}$ est un automate co-déterministe et co-minimal et si E est une expression rationnelle calculée par élimination d'états, quelque soit l'ordre d'élimination choisi, sur $\mathcal{A}$, alors $\mathcal{A}$ est le co-quotient minimal de l'automate des termes dérivés cassés de E.

Dans ce chapitre nous présentons donc la définition des termes dérivés cassés et de l'automate qu'il représente. Nous présentons également des résultats qui donnent des bornes sur la taille de l'automate des termes dérivés cassés d'une expression donnée. En particulier si une expression est en forme normale, alors son automate des termes dérivés cassés a la même borne de taille que son automate des termes dérivés. L'automate des termes dérivés cassés n'est cependant pas un quotient de l'automate des positions et, pour des expressions qui ne sont pas en forme normale, il est également possible qu'il soit deux fois plus grand que ce dernier.

3.1 Définitions et propriétés

3.1.1 Cassage d'une expression

Posons tout d'abord quelques notations sur les ensembles d'expressions rationnelles : soit X un ensemble d'expression rationnelles, $(X)_{\mathsf{p}}$ est la *partie propre* de X – c'est à dire $X \setminus \{\mathsf{1}\}$ – et δ_X est un booléen valant 1 si l'expression $\mathsf{1}$ appartient à X. En particulier il vient :

$$\forall X \subseteq \mathsf{RatE}(\mathsf{A}) \qquad \frac{\partial}{\partial a} X_{\mathsf{p}} = \frac{\partial}{\partial a} X \;, \quad \mathsf{c}(X_{\mathsf{p}}) \vee \delta_X = \mathsf{c}(X) \;\; , \tag{3.1}$$

$$\text{et} \qquad X_{\mathsf{p}} \cup \delta_X\{\mathsf{1}\} = X \;\; . \tag{3.2}$$

Le cassage d'expressions rationnelles est en fait une opération qui décompose une expression en un ensemble d'expressions dont le facteur le plus à gauche n'est pas une somme :

Définition 15 ([2])**.** *L'ensemble* $\mathrm{B}(\mathsf{E})$ *des termes cassés l'expression rationnelle* E *sur* A *est l'ensemble d'expressions défini inductivement par :*

$$\mathrm{B}(\mathsf{0}) = \{\mathsf{0}\} \;, \quad \mathrm{B}(\mathsf{1}) = \{\mathsf{1}\} \;, \quad \forall a \in A \quad \mathrm{B}(a) = \{a\} \;,$$

$$\mathrm{B}(\mathsf{F} + \mathsf{G}) = \mathrm{B}(\mathsf{F}) \cup \mathrm{B}(\mathsf{G}) \;\; , \tag{3.3}$$

$$\mathrm{B}(\mathsf{F} \cdot \mathsf{G}) = (\mathrm{B}(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G} \cup \delta_{\mathrm{B}(\mathsf{F})}\mathrm{B}(\mathsf{G}) \;\; , \tag{3.4}$$

$$\mathrm{B}(\mathsf{F}^*) = \{\mathsf{F}^*\} \;\; . \tag{3.5}$$

L'ensemble $\mathrm{B}(\mathsf{E})$ peut également être appelé le *cassage* de E, ou, lorsque l'on parle plus précisément de la fonction B, on dit *fonction de cassage.*

Par définition, le *cassage* d'une expression est additif :

$$\forall X \subseteq \mathsf{RatE}(\mathsf{A}) \qquad \mathrm{B}(X) = \bigcup_{\mathsf{E} \in X} \mathrm{B}(\mathsf{E}) \;\; .$$

Il est immédiat de vérifier, par exemple par induction, que cette opération est idempotente :

$$\mathrm{B}(\mathrm{B}(\mathsf{E})) = \mathrm{B}(\mathsf{E}) \;\; .$$

Il est aussi possible de comparer le langage d'une expression E et le langage des expressions résultant du cassage de E :

$$\bigcup_{X \in \mathrm{B}(\mathsf{E})} |X| = |\mathsf{E}| \;\; .$$

Il apparaît également immédiatement que le cassage d'une expression rationnelle E est un ensemble de concaténations droites de sous-expressions de E.

Exemple 16 (*Ex. 11 cont.*)**.** Nous continuons d'utiliser l'expression $\mathsf{E}_1 = \mathsf{F}_1 \cdot (a \cdot \mathsf{F}_1)$ avec $\mathsf{F}_1 = (a^* + b^*)$. Le cassage de E_1 et de F_1 donne les ensembles de termes cassés suivants :

$$\mathrm{B}(\mathsf{E}_1) = \{a^* \cdot (a \cdot \mathsf{F}_1),\, b^* \cdot (a \cdot \mathsf{F}_1)\}\,, \quad \mathrm{B}(\mathsf{F}_1) = \{a^*,\, b^*\}\ .$$

3.1.2 Termes dérivés cassés

Nous définissons maintenant la dérivation cassante par rapport à une lettre (respectivement à un mot) qui va nous permettre de définir les termes dérivés cassés :

Définition 16. *La dérivation cassante d'une expression rationnelle* E *sur* A *par rapport à une lettre* a *est définie comme le cassage de la dérivation de* E *par rapport à* a *:*

$$\frac{\partial_{\mathsf{b}}}{\partial a}\mathsf{E} = \mathrm{B}\left(\frac{\partial}{\partial a}\mathsf{E}\right)\ .$$

La dérivation cassante par rapport à un mot ua*, avec* $u \in A^+$*, est définie comme pour les autres dérivations :*

$$\frac{\partial_{\mathsf{b}}}{\partial ua}\mathsf{E} = \frac{\partial_{\mathsf{b}}}{\partial a}\left(\frac{\partial_{\mathsf{b}}}{\partial u}\mathsf{E}\right)\ .$$

Toute expression qui appartient à $\dfrac{\partial_{\mathsf{b}}}{\partial u}\mathsf{E}$*, pour un mot* u *quelconque de* A^+*, est appelée vrai terme dérivé cassé de* E*. Nous notons* $\mathrm{TBD}(\mathsf{E})$ *l'ensemble des vrais termes dérivés cassés de* E *:*

$$\mathrm{TBD}(\mathsf{E}) = \bigcup_{u \in A^+} \frac{\partial_{\mathsf{b}}}{\partial u}\mathsf{E}\ . \tag{3.6}$$

L'ensemble $\mathrm{BD}(\mathsf{E})$ *des termes dérivés cassés de* E *est l'union de l'ensemble* $\mathrm{TBD}(\mathsf{E})$ *des vrais termes dérivés cassés de* E *et de l'ensemble* $\mathrm{B}(\mathsf{E})$ *des termes cassés de* E *:* $\mathrm{BD}(\mathsf{E}) = \mathrm{TBD}(\mathsf{E}) \cup \mathrm{B}(\mathsf{E})$*.*

Par induction sur la profondeur de E et en utilisant la définition du cassage de E, on a :

$$\forall a \in A \quad \frac{\partial}{\partial a}\mathrm{B}(\mathsf{E}) = \frac{\partial}{\partial a}\mathsf{E}\,,$$

il en découle directement que :

$$\forall u \in A^+ \quad \frac{\partial_{\mathsf{b}}}{\partial u}\mathsf{E} = \mathrm{B}\left(\frac{\partial}{\partial u}\mathsf{E}\right)\ .$$

Ce qui implique la propriété fondamentale suivante :

Propriété 22. *Les termes dérivés cassés (respectivement les vrais termes dérivés cassés) d'une expression rationnelle* E *sont obtenus en prenant les termes cassés des termes dérivés (resp. des vrais termes dérivés) de* E *:*

$$\forall \mathsf{E} \in \mathsf{RatE}(\mathsf{A}) \qquad \mathrm{BD}(\mathsf{E}) = \bigcup_{\mathsf{K} \in \mathrm{D}(\mathsf{E})} \mathrm{B}(\mathsf{K}) \ , \quad \mathrm{TBD}(\mathsf{E}) = \bigcup_{\mathsf{K} \in \mathrm{TD}(\mathsf{E})} \mathrm{B}(\mathsf{K}) \ . \tag{3.7}$$

En particulier, si une expression E est constante, alors $\mathrm{TBD}(\mathsf{E}) = \emptyset$. De même, cela montre que l'ensemble des termes dérivés cassés est fini.

Exemple 17 (*Ex. 11 cont.*)**.** L'ensemble des vrais termes dérivés cassés de E_1 est :

$$\begin{aligned} \mathrm{TBD}(\mathsf{E}_1) &= \mathrm{B}(\{a^* \cdot (a \cdot \mathsf{F}_1),\, b^* \cdot (a \cdot \mathsf{F}_1),\, \mathsf{F}_1,\, a^*,\, b^*\}) \ , \\ &= \{a^* \cdot (a \cdot \mathsf{F}_1),\, b^* \cdot (a \cdot \mathsf{F}_1),\, a^*,\, b^*\} \ . \end{aligned}$$

L'ensemble des termes dérivés cassés de E_1 est donc :

$$\mathrm{BD}(\mathsf{E}_1) = \mathrm{TBD}(\mathsf{E}_1) \cup \mathrm{B}(\mathsf{E}_1) = \{a^* \cdot (a \cdot \mathsf{F}_1),\, b^* \cdot (a \cdot \mathsf{F}_1),\, a^*,\, b^*\} \ .$$

La Propriété 22 donne une manière de calculer l'ensemble des vrais termes dérivés cassés directement à partir de l'ensemble des vrais termes dérivés. Cela permet donc d'avoir, comme précédemment, une manière de calculer l'ensemble des termes dérivés cassés inductive sans passer par la dérivation cassée :

Proposition 23. *Soient* F *et* G *deux expression rationnelles. Nous avons :*

$$\mathrm{TBD}(\mathsf{F} + \mathsf{G}) = \mathrm{TBD}(\mathsf{F}) \cup \mathrm{TBD}(\mathsf{G}) \ , \tag{3.8}$$

$$\mathrm{TBD}(\mathsf{F} \cdot \mathsf{G}) = (\mathrm{TBD}(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G} \cup \mathrm{TBD}(\mathsf{G}) \cup \delta_{\mathrm{TBD}(\mathsf{F})}\mathrm{B}(\mathsf{G}) \ , \tag{3.9}$$

$$\mathrm{TBD}(\mathsf{F}^*) = (\mathrm{TBD}(\mathsf{F})) \cdot \mathsf{F}^* \ . \tag{3.10}$$

Démonstration. Cet énoncé est une application directe de (3.7) au calcul inductif de l'ensemble des vrais termes dérivés (Proposition 15, Equations (2.7)–(2.9)). L'équation (3.8) est triviale, développons (3.9) :

$$\begin{aligned} \mathrm{TBD}(\mathsf{F} \cdot \mathsf{G}) &= \mathrm{B}(\mathrm{TD}(\mathsf{F} \cdot \mathsf{G})) = \mathrm{B}\big((\mathrm{TD}(\mathsf{F})) \cdot \mathsf{G} \cup \mathrm{TD}(\mathsf{G})\big) \\ &= \mathrm{B}((\mathrm{TD}(\mathsf{F})) \cdot \mathsf{G}) \cup \mathrm{B}(\mathrm{TD}(\mathsf{G})) \\ &= (\mathrm{TBD}(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G} \cup \delta_{\mathrm{TBD}(\mathsf{F})}\mathrm{B}(\mathsf{G}) \cup \mathrm{TBD}(\mathsf{G}) \ . \end{aligned}$$

Pour établir (3.10), notons tout d'abord que pour tout ensemble X d'expressions rationnelles, nous avons :

$$\left(\mathrm{B}\left(X\right)\right)_{\mathsf{p}} \subseteq \mathrm{B}\left(\left(X\right)_{\mathsf{p}}\right) \quad ,$$

et donc, par (3.2) : $\mathrm{B}\left(\left(X\right)_{\mathsf{p}}\right) \cup \delta_{\mathrm{B}(X)}\{1\} = \mathrm{B}\left(X\right)$. Nous avons alors :

$$\begin{aligned}\mathrm{TBD}\left(\mathsf{F}^*\right) &= \mathrm{B}\left(\mathrm{TD}\left(\mathsf{F}^*\right)\right) = \mathrm{B}\left(\left(\mathrm{TD}\left(\mathsf{F}\right)\right)\cdot\mathsf{F}^*\right)\\ &= \left(\mathrm{B}\left(\left(\mathrm{TD}\left(\mathsf{F}\right)\right)_{\mathsf{p}}\right)\right)\cdot\mathsf{F}^* \cup \delta_{\mathrm{B}(\mathrm{TD}(\mathsf{F}))}\mathrm{B}\left(\mathsf{F}^*\right)\\ &= \mathrm{B}\left(\left(\left(\mathrm{TD}\left(\mathsf{F}\right)\right)_{\mathsf{p}} \cup \{1\}\right)\cdot\mathsf{F}^*\right) = \left(\mathrm{TBD}\left(\mathsf{F}\right)\right)\cdot\mathsf{F}^* \quad .\end{aligned}$$

□

Alors qu'il n'est pas nécessaire de donner une définition inductive directe de l'ensemble des termes dérivés car il suffit de rajouter l'expression elle même à l'ensemble des vrais termes dérivés – définis inductivement –, il est utile de le faire pour les termes dérivés cassés :

Proposition 24. *Soient* F *et* G *deux expression rationnelles. Nous avons :*

$$\mathrm{BD}\left(\mathsf{F}+\mathsf{G}\right) = \mathrm{BD}\left(\mathsf{F}\right) \cup \mathrm{BD}\left(\mathsf{G}\right) \quad , \tag{3.11}$$

$$\mathrm{BD}\left(\mathsf{F}\cdot\mathsf{G}\right) = \left(\mathrm{BD}\left(\mathsf{F}\right)\right)_{\mathsf{p}}\cdot\mathsf{G} \cup \mathrm{TBD}\left(\mathsf{G}\right) \cup \delta_{\mathrm{BD}(\mathsf{F})}\mathrm{B}\left(\mathsf{G}\right) \quad , \tag{3.12}$$

$$\mathrm{BD}\left(\mathsf{F}^*\right) = \left(\mathrm{TBD}\left(\mathsf{F}\right)\right)_{\mathsf{p}}\cdot\mathsf{F}^* \cup \{\mathsf{F}^*\} \quad . \tag{3.13}$$

Démonstration. Une nouvelle fois le cas de la somme, (3.11), est trivial ; nous développons donc (3.12) :

$$\begin{aligned}\mathrm{BD}\left(\mathsf{F}\cdot\mathsf{G}\right) &= \mathrm{TBD}\left(\mathsf{F}\cdot\mathsf{G}\right) \cup \mathrm{B}\left(\mathsf{F}\cdot\mathsf{G}\right)\\ &= \left(\mathrm{TBD}\left(\mathsf{F}\right)\right)_{\mathsf{p}}\cdot\mathsf{G} \cup \left(\mathrm{B}\left(\mathsf{F}\right)\right)_{\mathsf{p}}\cdot\mathsf{G} \cup \mathrm{TBD}\left(\mathsf{G}\right) \cup \delta_{\mathrm{TBD}(\mathsf{F})}\mathrm{B}\left(\mathsf{G}\right) \cup \delta_{\mathrm{B}(\mathsf{F})}\mathrm{B}\left(\mathsf{G}\right)\\ &= \left(\mathrm{BD}\left(\mathsf{F}\right)\right)_{\mathsf{p}}\cdot\mathsf{G} \cup \mathrm{TBD}\left(\mathsf{G}\right) \cup \delta_{\mathrm{BD}(\mathsf{F})}\mathrm{B}\left(\mathsf{G}\right) \quad .\end{aligned}$$

Pour l'Equation (3.13), l'écriture directe donne :

$$\mathrm{BD}\left(\mathsf{F}^*\right) = \mathrm{TBD}\left(\mathsf{F}\right)\cdot\mathsf{F}^* \cup \{\mathsf{F}^*\} \quad .$$

Comme $1\cdot\mathsf{F}^* \equiv \mathsf{F}^*$ modulo les identités triviales, il est possible de prendre la partie propre de $\mathrm{TBD}\left(\mathsf{F}\right)$ dans l'expression ci-dessus et, donc, de la transformer en Equation (3.13). □

Enfin, comme pour les termes dérivés, cela permet une définition inductive directe de l'ensemble des termes dérivés cassés (resp. des vrais termes dérivés cassés).

Corollaire 25. *Les ensembles* $\mathrm{BD}(\mathsf{E})$ *et* $\mathrm{TBD}(\mathsf{E})$ *des termes dérivés cassés et des vrais termes dérivés cassés d'une expression rationnelle* E *sur* A *peuvent être calculés par induction sur la profondeur de* E *en utilisant les équations (3.8)–(3.13) et les cas de base suivants :*

$$\begin{aligned} \mathrm{TBD}(\mathsf{0}) &= \emptyset\,, \quad & \mathrm{TBD}(\mathsf{1}) &= \emptyset\,, \quad & \forall a \in A \quad \mathrm{TBD}(a) &= \{\mathsf{1}\}\ , \\ \mathrm{BD}(\mathsf{0}) &= \{\mathsf{0}\}\,, \quad & \mathrm{BD}(\mathsf{1}) &= \{\mathsf{1}\}\,, \quad & \forall a \in A \quad \mathrm{BD}(a) &= \{a, \mathsf{1}\}\ . \end{aligned}$$

Il apparait alors évident que nous avons, comme pour les termes dérivés :

Proposition 26. *Les termes dérivés cassés d'une expression* E *sont des concaténations à droite de sous-expressions de* E.

3.1.3 Automate des termes dérivés cassés

Comme pour les expressions dérivées et les termes dérivés, les termes dérivés cassés permettent la définition d'un automate fini :

Définition 17 ([32])**.** *L'automate* $\mathcal{D}_b(\mathsf{E})$ *des termes dérivés cassés d'une expression rationnelle* E *est tel que :*
(i) *l'ensemble des états est l'ensemble* $\mathrm{BD}(\mathsf{E})$ *;*
(ii) *les états initiaux sont les termes cassés* $\mathrm{B}(\mathsf{E})$ *de* E *;*
(iii) *les états finaux sont les états dont le terme constant est non nul ;*
(iv) *soient* K_1 *et* K_2 *deux termes dérivés cassés, la transition* $(\mathsf{K}_1, a, \mathsf{K}_2)$ *existe pour une lettre* $a \in A$ *donnée si et seulement si* $\mathsf{K}_2 \in \frac{\partial_b}{\partial a}\mathsf{K}_1$.

Et nous avons :

Proposition 27. *Le langage reconnu par l'automate* $\mathcal{D}_b(\mathsf{E})$ *est le langage dénoté par* E.

Exemple 18 (*Ex. 11 cont.*)**.** La Figure 3.1 montre l'automate $\mathcal{D}_b(\mathsf{E}_1)$ des termes dérivés cassés de E_1.

Remarque 5. Le fait que l'automate des termes dérivés cassés d'une expression puisse avoir plusieurs états initiaux montre en soi que ce n'est pas, contrairement à l'automate des termes dérivés, un quotient de l'automate standard.

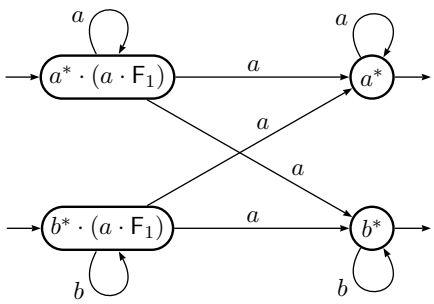


FIGURE 3.1 – $\mathcal{D}_b(\mathsf{E}_1)$

.

La motivation de la définition des termes dérivés cassés et de l'automate associé vient de la problématique même du passage d'une expression rationnelle à un automate, et vice-versa, en effet :

Théorème 28 ([32])**.** *Soient $\mathcal{A}$ un automate fini co-déterministe et co-minimal et E une expression rationnelle obtenue par élimination d'états sur l'automate $\mathcal{A}$* dans un ordre quelconque.

Le co-quotient minimal de l'automate $\mathcal{D}_b(\mathsf{E})$ des termes dérivés cassés de E est isomorphe à $\mathcal{A}$.

Ce théorème est important car il montre que – dans les conditions énoncées – l'élimination d'états ne se contente pas d'encoder le langage reconnu par l'automate dans l'expression mais également, d'une certaine manière, la structure de l'automate lui-même. La dérivation cassante permet donc de 'décoder' les informations ainsi placées dans l'expression.

3.1.4 Associativité du produit

Comme pour les termes dérivés et les expressions dérivées, il est intéressant de regarder l'effet du parenthésage des produits d'expressions sur le nombre de termes dérivés cassés.

Comme l'opération de cassage des expressions rationnelles 'entre' dans le terme le plus à gauche d'une concaténation, peu importe le parenthésage, il en résulte que le cassage commute par l'associativité du produit. Il en découle que la dérivation cassante également :

Proposition 29. *Soient E, F et G des expressions rationnelles :*

$$\mathrm{BD}\left((\mathsf{E} \cdot \mathsf{F}) \cdot \mathsf{G}\right) \equiv_{\mathbf{A_p}} \mathrm{BD}\left(\mathsf{E} \cdot (\mathsf{F} \cdot \mathsf{G})\right) \quad .$$

Comme les termes dérivés cassés d'une expression rationnelle E sont, comme les termes dérivés, des concaténations à droite de sous-expressions de E, une preuve similaire à celle du Théorème 21 permet d'établir :

Théorème 30 ([2])**.** *Soient* E*,* F *et* G *des expressions rationnelles.*

$$\operatorname{card}(\mathrm{BD}((\mathsf{E}\cdot\mathsf{F})\cdot\mathsf{G})) \leqslant \operatorname{card}(\mathrm{BD}(\mathsf{E}\cdot(\mathsf{F}\cdot\mathsf{G})))\quad .$$

3.2 Taille de l'automate des termes dérivés cassés

Pour que l'automate des termes dérivés cassés soit un objet manipulable, il faut entre autres pouvoir donner une borne sur sa taille. C'est ce que nous établissons dans cette section.

Nous montrons tout d'abord une borne sur la taille de l'automate des termes dérivés cassés dans le cas général – ou presque général –. Cette borne correspond à deux fois la taille maximale de l'automate des termes dérivés. Ensuite nous donnons une borne dans le cas d'une expression en forme normale. Comme nous l'avons vu précédemment une expression peut être transformée en une expression équivalente en forme normale en temps linéaire, ce qui rend ces expressions particulières intéressantes et quasi-générales. Pour une expression en forme normale, l'automate des termes dérivés cassés est au plus aussi grand que l'automate standard – comme l'automate des termes dérivés –. Enfin nous analysons les tailles de l'automate des termes dérivés et de l'automate des termes dérivés cassés d'une même expression pour montrer qu'elles sont incomparables.

3.2.1 Dans le cas général

Commençons par un exemple :

Exemple 19. Soit l'expression constante $1+1^*+(1^*)^*+((1^*)^*)^*$. Ses termes cassés sont :

$$\mathrm{B}(1+1^*+(1^*)^*+((1^*)^*)^*) = \{1, 1^*, (1^*)^*, ((1^*)^*)^*\}\quad .$$

Cet exemple donne une manière facile de créer une expression sans aucune lettre et dont l'ensemble des termes dérivés cassés, est arbitrairement grand car l'ensemble des termes cassés est lui-même aussi arbitrairement grand. Cependant cette explosion est une conséquence du cassage de l'expression initiale elle même et non de la dérivation car l'expression de l'exemple n'a aucun vrai terme dérivé – et donc aucun vrai terme dérivé cassé –. Il apparaît

donc raisonnaible de restreindre dans le cas *général* l'utilisation de l'étoile sur l'expression $\mathsf{1}$.

Nous appelons *expression constante étoilée* une expression constante dont une sous-expression – donc constante également – est de la forme E^*. Une expression constante qui n'est pas étoilée est donc une somme de $\mathsf{1}$.

L'exemple suivant nous montre également que, même sans considérer les expressions constantes étoilées, le nombre de termes dérivés cassés d'une expression E peut dépasser la limite $\ell(\mathsf{E}) + 1$ des termes dérivés :

Exemple 20. Soit $\mathsf{G}_k = ((a^*)^* + \mathsf{1})^k$, pour tout entier positif k. Plus précisément – puisque le choix du parenthésage a une importance – : nous posons $\mathsf{G}_1 = ((a^*)^* + \mathsf{1})$ et $\mathsf{G}_{k+1} = (\mathsf{G}_1 \cdot \mathsf{G}_k)$. Nous avons $\ell(\mathsf{G}_k) = k$ et :

$$\mathrm{B}(\mathsf{G}_1) = \{(a^*)^*,\, \mathsf{1}\}\,, \qquad \text{et} \qquad \mathrm{BD}(\mathsf{G}_1) = \{(a^*)^*, a^*(a^*)^*, \mathsf{1}\}\ ,$$

ensuite, pour tout k, $\mathrm{B}(\mathsf{G}_{k+1}) = (a^*)^* \cdot \mathsf{G}_k \cup \mathrm{B}(\mathsf{G}_k)$. Par (3.12), et comme $\delta_{\mathrm{BD}(\mathsf{G}_1)} = 1$, nous avons :

$$\begin{aligned}\mathrm{BD}(\mathsf{G}_{k+1}) = \mathrm{BD}(\mathsf{G}_1 \cdot \mathsf{G}_k) &= (\mathrm{BD}(\mathsf{G}_1))_{\mathsf{p}} \cdot \mathsf{G}_k \cup \mathrm{BD}(\mathsf{G}_k) \\ &= (a^*)^* \cdot \mathsf{G}_k \cup (a^*(a^*)^*) \cdot \mathsf{G}_k \cup \mathrm{BD}(\mathsf{G}_k)\ .\end{aligned}$$

Comme toutes les expressions G_k sont deux à deux disjointes, il en va de même pour les $(a^*)^* \cdot \mathsf{G}_k$ et les $(a^*(a^*)^*) \cdot \mathsf{G}_k$.

Et donc : $\mathrm{BD}(\mathsf{G}_k) \cap \big((a^*)^* \cdot \mathsf{G}_k \cup (a^*(a^*)^*) \cdot \mathsf{G}_k\big) = \emptyset$, il en découle :

$$\mathrm{card}\,(\mathrm{BD}(\mathsf{G}_k)) = 2\,\ell(\mathsf{G}_k) + 1\ .$$

L'expression de cet exemple atteint la borne générale des termes dérivés cassés :

Théorème 31 ([2])**.** *Soit* E *une expression rationnelle sans sous-expression constante étoilée. Le nombre de termes dérivés cassés de* E *est borné par deux fois sa longueur littérale plus un :*

$$\mathrm{card}\,(\mathrm{BD}(\mathsf{E})) \leqslant 2\,\ell(\mathsf{E}) + 1\ .$$

Ce théorème est une conséquence de la proposition suivante :

Proposition 32. *Soit* E *une expression rationnelle sans sous-expression constante étoilée. Nous avons :*

$$\mathrm{card}\left((\mathrm{BD}(\mathsf{E}))_{\mathsf{p}}\right) \leqslant 2\,\ell(\mathsf{E})\ , \tag{3.14}$$

$$\textit{si } \ell(\mathsf{E}) \geqslant 1\,, \quad \mathrm{card}\,(\mathrm{TBD}(\mathsf{E})) \leqslant 2\,\ell(\mathsf{E}) - 1\ . \tag{3.15}$$

Comme $\mathrm{card}\,(X) \leqslant \mathrm{card}\left((X)_{\mathsf{p}}\right)+1$ pour tout ensemble d'expressions X, le Théorème 31 correspond exactement à l'équation (3.14). La nécessité d'établir simultanément (3.15) est provoquée par l'apparition de $\mathrm{TBD}\,(\mathsf{F})$ dans la définition inductive de $\mathrm{BD}\,(\mathsf{F}^*)$ (cf. Proposition 24).

Démonstration. Nous prouvons cette proposition par induction sur la profondeur de E à l'aide des formules de la Proposition 24.

Les cas de base $\mathsf{E}=1$ et $\mathsf{E}=a$ sont trivialement vérifiés.

Le cas $\mathsf{E}=0$ n'est pas considéré comme un cas de base car une expression différente de 0 n'a pas 0 comme sous-expression (par les identités triviales **T**).

Nous rappelons que si $\mathsf{E}=\mathsf{F}+\mathsf{G}$ ou $\mathsf{E}=\mathsf{F}\cdot\mathsf{G}$ alors $\ell(\mathsf{E})=\ell(\mathsf{F})+\ell(\mathsf{G})$, et si $\mathsf{E}=\mathsf{F}^*$, alors $\ell(\mathsf{E})=\ell(\mathsf{F})$. De plus, pour la bonne exécution de l'induction, il convient de noter que si E n'a pas de sous-expression constante étoilée, il en est de même pour F et G.

Nous avons donc les hypothèses d'induction suivantes :

$$\mathrm{card}\left((\mathrm{BD}\,(\mathsf{F}))_{\mathsf{p}}\right) \leqslant 2\,\ell(\mathsf{F}) \qquad \text{et} \qquad \mathrm{card}\left((\mathrm{BD}\,(\mathsf{G}))_{\mathsf{p}}\right) \leqslant 2\,\ell(\mathsf{G}) \ .$$

et, si $\ell(\mathsf{F}) \geqslant 1$,

$$\mathrm{card}\,(\mathrm{TBD}\,(\mathsf{F})) \leqslant 2\,\ell(\mathsf{F})-1 \ . \tag{3.16}$$

Cas $\mathsf{E}=\mathsf{F}+\mathsf{G}$: Par la définition inductive (3.11), nous avons $(\mathrm{BD}\,(\mathsf{E}))_{\mathsf{p}} = (\mathrm{BD}\,(\mathsf{F}))_{\mathsf{p}} \cup (\mathrm{BD}\,(\mathsf{G}))_{\mathsf{p}}$ et (3.14) est vérifiée par induction. De même, la définition inductive des vrais termes dérivés cassés (3.8) donne : $\mathrm{TBD}\,(\mathsf{E}) = \mathrm{TBD}\,(\mathsf{F}) \cup \mathrm{TBD}\,(\mathsf{G})$.

Si $\ell(\mathsf{E}) \geqslant 1$, alors on peut supposer sans perte de généralité que $\ell(\mathsf{F}) \geqslant 1$ et alors on a (3.16).

Dans ce cas, ou bien $\ell(\mathsf{G})=0$, et donc $\mathrm{TBD}\,(\mathsf{G})=\emptyset$, ou bien $\ell(\mathsf{G}) \geqslant 1$ et alors $\mathrm{card}\,(\mathrm{TBD}\,(\mathsf{G})) \leqslant 2\,\ell(\mathsf{G})-1$. Dans les deux cas, (3.15) est vérifiée.

Cas $\mathsf{E}=\mathsf{F}\cdot\mathsf{G}$ (qui implique que F et G sont différents de 1) :

De la définition inductive des termes dérivés cassés (3.12), on a :

$$\mathrm{BD}\,(\mathsf{E}) = (\mathrm{BD}\,(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G} \cup \mathrm{TBD}\,(\mathsf{G}) \cup \delta_{\mathrm{BD}(\mathsf{F})}\mathrm{B}\,(\mathsf{G}) \ .$$

Comme $\mathsf{G} \neq 1$, $\Big((\mathrm{BD}\,(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G}\Big)_{\mathsf{p}} = (\mathrm{BD}\,(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G}$ et alors :

$$\begin{aligned}(\mathrm{BD}\,(\mathsf{E}))_{\mathsf{p}} &= (\mathrm{BD}\,(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G} \cup \big(\mathrm{TBD}\,(\mathsf{G}) \cup \delta_{\mathrm{BD}(\mathsf{F})}\mathrm{B}\,(\mathsf{G})\big)_{\mathsf{p}} \\ &\subseteq (\mathrm{BD}\,(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G} \cup (\mathrm{BD}\,(\mathsf{G}))_{\mathsf{p}} \quad ,\end{aligned}$$

donc : $\quad \mathrm{card}\Big((\mathrm{BD}\,(\mathsf{E}))_{\mathsf{p}}\Big) \leqslant \mathrm{card}\Big((\mathrm{BD}\,(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G}\Big) + \mathrm{card}\Big((\mathrm{BD}\,(\mathsf{G}))_{\mathsf{p}}\Big)$,

l'équation (3.14) est vérifiée.

De (3.9), $\mathrm{TBD}\,(\mathsf{E}) = (\mathrm{TBD}\,(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G} \cup \mathrm{TBD}\,(\mathsf{G}) \cup \delta_{\mathrm{TBD}(\mathsf{F})}\mathrm{B}\,(\mathsf{G})$.
Si $1 \in \mathrm{TBD}\,(\mathsf{F})$, alors $\ell(\mathsf{F}) \geqslant 1$ et $\mathrm{TBD}\,(\mathsf{E}) = (\mathrm{TBD}\,(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G} \cup \mathrm{BD}\,(\mathsf{G})$.
Les hypothèses d'induction donnent $\mathrm{card}\Big((\mathrm{TBD}\,(\mathsf{F}))_{\mathsf{p}}\Big) \leqslant 2\,\ell(\mathsf{F}) - 2$ et $\mathrm{card}\,(\mathrm{BD}\,(\mathsf{G})) \leqslant 2\,\ell(\mathsf{G}) + 1$ et donc (3.15).

Si $1 \notin \mathrm{TBD}\,(\mathsf{F})$,
alors $\mathrm{TBD}\,(\mathsf{E}) = (\mathrm{TBD}\,(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G} \cup \mathrm{TBD}\,(\mathsf{G}) \subseteq (\mathrm{BD}\,(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G} \cup \mathrm{TBD}\,(\mathsf{G})$.
Si $\ell(\mathsf{E}) \geqslant 1$ alors,
soit $\ell(\mathsf{G}) \geqslant 1$ et alors $\mathrm{card}\,(\mathrm{TBD}\,(\mathsf{F} \cdot \mathsf{G})) \leqslant 2\,\ell(\mathsf{F}) + 2\,\ell(\mathsf{G}) - 1$,
ou $\ell(\mathsf{G}) = 0$ et alors $\ell(\mathsf{F}) \geqslant 1$ et $\mathrm{TBD}\,(\mathsf{G}) = \emptyset$, et donc :

$$\mathrm{card}\,(\mathrm{TBD}\,(\mathsf{F} \cdot \mathsf{G})) \leqslant 2\,\ell(\mathsf{F}) - 1$$

Dans les deux cas, (3.15) est vérifiée.

Cas $\mathsf{E} = \mathsf{F}^*$: La sous-expression F est étoilée, donc, par hypothèse, elle n'est pas constante et $\ell(\mathsf{F}) \geqslant 1$. Par induction, $\mathrm{card}\,(\mathrm{TBD}\,(\mathsf{F})) \leqslant 2\,\ell(\mathsf{F}) - 1 = 2\,\ell(\mathsf{E}) - 1$.
De (3.13), on a $\mathrm{BD}\,(\mathsf{F}^*) = (\mathrm{TBD}\,(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{F}^* \cup \{\mathsf{F}^*\}$ et donc :

$$\mathrm{card}\Big((\mathrm{BD}\,(\mathsf{E}))_{\mathsf{p}}\Big) = \mathrm{card}\Big((\mathrm{TBD}\,(\mathsf{F}))_{\mathsf{p}}\Big) + 1 \leqslant 2\,\ell(\mathsf{E}) \quad .$$

De (3.10), on a $\mathrm{TBD}\,(\mathsf{F}^*) = \mathrm{TBD}\,(\mathsf{F}) \cdot \mathsf{F}^*$ et donc :

$$\mathrm{card}\,(\mathrm{TBD}\,(\mathsf{F}^*)) = \mathrm{card}\,(\mathrm{TBD}\,(\mathsf{F})) \leqslant 2\,\ell(\mathsf{E}) - 1 \quad .$$

□

3.2.2 Pour les expressions en forme normale

Théorème 33. *Pour une expression rationnelle* E *en forme normale, on a :*

$$\mathrm{card}\,(\mathrm{BD}\,(\mathsf{E})) \leqslant \ell(\mathsf{E}) + 1 \quad .$$

Démonstration. Le théorème est vérifié trivialement pour $\mathsf{E} = \mathsf{0}$, et nous pouvons donc supposer maintenant que E, et donc toutes ses sous-expressions sont différentes de 0.

Comme pour le cas général, nous préférons démontrer le théorème sous la forme suivante :

$$\operatorname{card}\left(\left(\operatorname{BD}(\mathsf{E})\right)_{\mathsf{p}}\right) \leqslant \ell(\mathsf{E}) \ , \tag{3.17}$$

et, comme pour le cas général, l'induction nous amène à démontrer simultanément deux énoncés supplémentaires :

$$\text{si} \quad \mathsf{c}(\mathsf{E}) = 0\,, \quad \operatorname{card}\left(\left(\operatorname{TBD}(\mathsf{E})\right)_{\mathsf{p}}\right) \leqslant \ell(\mathsf{E}) - 1 \ , \tag{3.18}$$

$$\text{si} \quad \mathsf{c}(\mathsf{E}) = 1 \quad \text{et} \quad 1 \in \operatorname{TBD}(\mathsf{E})\ , \quad \operatorname{card}\left(\left(\operatorname{TBD}(\mathsf{E})\right)_{\mathsf{p}}\right) \leqslant \ell(\mathsf{E}) - 1 \ . \tag{3.19}$$

Nous démontrons ces trois conditions par induction sur la profondeur de E en utilisant les définitions inductives des ensembles des termes dérivés cassés et des vrais termes dérivés définis dans les Propositions 23 et 24. Les deux cas de bases, $\mathsf{E} = 1$ et $\mathsf{E} = a$ sont triviaux à vérifier.

Cas $\mathsf{E} = \mathsf{F} + \mathsf{G}$:

$$\operatorname{BD}(\mathsf{E}) = \operatorname{BD}(\mathsf{F}) \cup \operatorname{BD}(\mathsf{G})$$

et (3.17) est vérifiée par induction. De plus :

$$\operatorname{TBD}(\mathsf{E}) = \operatorname{TBD}(\mathsf{F}) \cup \operatorname{TBD}(\mathsf{G}) \subseteq \operatorname{BD}(\mathsf{F}) \cup \operatorname{TBD}(\mathsf{G}) \ ,$$

alors :

$$\left(\operatorname{TBD}(\mathsf{E})\right)_{\mathsf{p}} = \left(\operatorname{TBD}(\mathsf{F})\right)_{\mathsf{p}} \cup \left(\operatorname{TBD}(\mathsf{G})\right)_{\mathsf{p}} \subseteq \left(\operatorname{BD}(\mathsf{F})\right)_{\mathsf{p}} \cup \left(\operatorname{TBD}(\mathsf{G})\right)_{\mathsf{p}} \ .$$

Si $\mathsf{c}(\mathsf{E}) = 0$, alors $\mathsf{c}(\mathsf{F}) = \mathsf{c}(\mathsf{G}) = 0$ et, d'après les hypothèses de récurrence :

$$\operatorname{card}\left(\left(\operatorname{TBD}(\mathsf{E})\right)_{\mathsf{p}}\right) \leqslant \ell(\mathsf{F}) - 1 + \ell(\mathsf{G}) - 1 = \ell(\mathsf{E}) - 2 \ .$$

Si $\mathsf{c}(\mathsf{E}) = 1$, sans perte de généralité, on peut supposer que $\mathsf{c}(\mathsf{F}) = 1$. Dans ce cas, soit $\mathsf{c}(\mathsf{G}) = 0$ ou bien $\mathsf{c}(\mathsf{G}) = 1$. Dans le premier cas, on peut appliquer (3.18) à G. Dans le deuxième cas, si $1 \in \operatorname{TBD}(\mathsf{E})$ alors on peut supposer que 1 est dans $\operatorname{TBD}(\mathsf{G})$ car F et G ont le même rôle et le même terme constant. On peut alors appliquer (3.19) à G.

Dans les deux cas, on a $\operatorname{card}\left(\left(\operatorname{TBD}(\mathsf{G})\right)_{\mathsf{p}}\right) \leqslant \ell(\mathsf{G}) - 1$ et donc :

$$\operatorname{card}\left(\left(\operatorname{TBD}(\mathsf{E})\right)_{\mathsf{p}}\right) \leqslant \ell(\mathsf{F}) + \ell(\mathsf{G}) - 1 = \ell(\mathsf{E}) - 1 \ .$$

Cas $\mathsf{E} = \mathsf{F} \cdot \mathsf{G}$:

$$\mathrm{BD}(\mathsf{E}) = (\mathrm{BD}(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G} \cup \mathrm{TBD}(\mathsf{G}) \cup \delta_{\mathrm{BD}(\mathsf{F})}\mathrm{B}(\mathsf{G}) \subseteq (\mathrm{BD}(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G} \cup \mathrm{BD}(\mathsf{G}) \quad .$$

Comme $\mathsf{G} \neq 1$, (3.17) est vérifiée.

$$\mathrm{TBD}(\mathsf{E}) = (\mathrm{TBD}(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G} \cup \mathrm{TBD}(\mathsf{G}) \cup \delta_{\mathrm{TBD}(\mathsf{F})}\mathrm{B}(\mathsf{G}) \subseteq (\mathrm{BD}(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{G} \cup \mathrm{BD}(\mathsf{G}) \quad .$$

Si $\mathsf{c}(\mathsf{E}) = 0$ alors, ou bien $\mathsf{c}(\mathsf{F}) = 0$ (cas 1), ou bien $\mathsf{c}(\mathsf{F}) = 1$ et $1 \in \mathrm{TD}(\mathsf{F})$ (cas 2) ou bien $\mathsf{c}(\mathsf{F}) = 1$ et $1 \notin \mathrm{TD}(\mathsf{F})$ (cas 3). Pour les cas 1 et 2, par récurrence :

$$\mathrm{card}\left((\mathrm{TBD}(\mathsf{E}))_{\mathsf{p}}\right) \leqslant \mathrm{card}\left((\mathrm{TBD}(\mathsf{F}))_{\mathsf{p}}\right) + \mathrm{card}\left((\mathrm{BD}(\mathsf{G}))_{\mathsf{p}}\right) \leqslant \ell(\mathsf{F}) - 1 + \ell(\mathsf{G}) = \ell(\mathsf{E}) - 1 \quad .$$

Dans le cas 3, nous avons à la fois $\mathsf{c}(\mathsf{G}) = 0$ et $\delta_{\mathrm{TBD}(\mathsf{F})} = 0$ et alors par induction :

$$\mathrm{card}\left((\mathrm{TBD}(\mathsf{E}))_{\mathsf{p}}\right) \leqslant \mathrm{card}\left((\mathrm{BD}(\mathsf{F}))_{\mathsf{p}}\right) + \mathrm{card}\left((\mathrm{TBD}(\mathsf{G}))_{\mathsf{p}}\right) \leqslant \ell(\mathsf{F}) + \ell(\mathsf{G}) - 1 = \ell(\mathsf{E}) - 1 \quad .$$

Cas $\mathsf{E} = \mathsf{F}^*$:

$$\mathrm{BD}(\mathsf{E}) = (\mathrm{TBD}(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{F}^* \cup \{\mathsf{F}^*\} \text{ et } \mathrm{TBD}(\mathsf{E}) = (\mathrm{TBD}(\mathsf{F}))_{\mathsf{p}} \cdot \mathsf{F}^* \quad .$$

Comme E est en forme normale, $\mathsf{c}(\mathsf{F}) = 0$, alors nous avons $\mathrm{card}\left((\mathrm{BD}(\mathsf{E}))_{\mathsf{p}}\right) \leqslant (\ell(\mathsf{F}) - 1) + 1$ et (3.17) est donc vérifiée. Comme $\mathsf{c}(\mathsf{E}) = 1$ et $1 \notin \mathrm{TBD}(\mathsf{E})$ (puisque $\mathsf{F} \neq 1$) les deux énoncés (3.18) et (3.19) sont vérifiés. □

3.2.3 Comparaison avec les termes dérivés

Pour finir, nous montrons que le nombre de termes dérivés cassés est incomparable avec le nombre de termes dérivés de la même expression.

Nous avons déjà vu à travers l'exemple de l'expression $\mathsf{E}_1 = (a^* + b^*) \cdot (a \cdot (a^* + b^*))$ qu'il y a des expressions pour lesquelles l'automate des termes dérivés cassés est plus petit que l'automate des termes dérivés. En fait on peut même étendre à des expressions plus longues, par exemple, $\mathsf{E}_k = (a^* + b^*) \cdot (a \cdot (a^* + b^*))^k$ a $3k + 3$ termes dérivés et $2k + 2$ termes dérivés cassés.

D'un autre côté, si nous posons $\mathsf{H}_k = a \cdot (b_1 + b_2 + ... + b_k)$, l'expression H_k a 3 termes dérivés et $k + 2$ termes dérivés cassés (La Figure 3.2 montre l'automate des termes dérivés – à gauche – et l'automate des termes dérivés cassés – à droite – de l'expression $\mathsf{H}_3 = a \cdot (b_1 + b_2 + b_3)$).

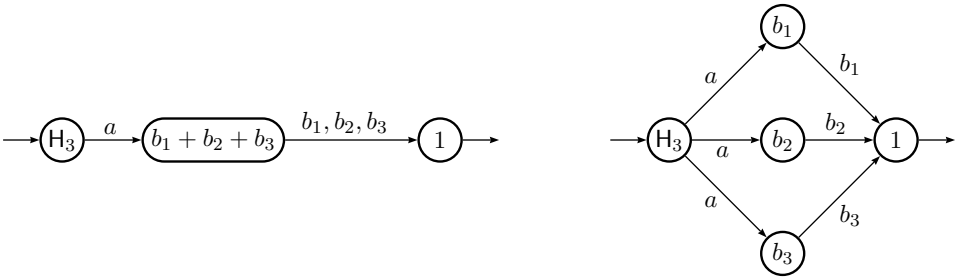


FIGURE 3.2 – $\mathcal{D}(\mathsf{H}_3)$ et $\mathcal{D}_b(\mathsf{H}_3)$

Chapitre 4

Construction des automates des termes dérivés et termes dérivés cassés

Dans ce chapitre, nous expliciterons une construction pour l'automate des termes dérivés et pour l'automate des termes dérivés cassés. La première construction d'un automate des termes dérivés est donnée dans [5] comme l'application directe de la définition de l'automate. Dans cette méthode, on construit l'ensemble des termes dérivés en dérivant tous les termes dérivés connus par chaque lettre de l'alphabet jusqu'à ne plus en trouver de nouveaux. Les transitions sont construites simultanément à chaque étape par la dérivation par rapport à une lettre. Si l'expression de départ est de taille n alors la méthode d'Antimirov a une complexité de $O(n^5)$. Pour les termes dérivés cassés il est possible d'envisager une méthode similaire, avec une complexité identique.

Il existe également une construction de l'automate des termes dérivés qui se base sur des modifications de l'arbre syntaxique de l'expression. Ces modifications font ressortir une représentation des termes dérivés sous forme de chemins dans un graphe. Cette idée de modifier l'arbre syntaxique d'une expression rationnelle en ajoutant des arcs orientés afin de pouvoir obtenir un automate a d'abord été introduite dans [51] où une telle modification de l'arbre, appelée alors *ZPC-structure* permet de construire l'automate des positions de l'expression.

En se basant sur le résultat que l'automate des termes dérivés est un quotient de l'automate des positions, Champarnaud et Ziadi, dans [18], utilisent également la ZPC-structure pour calculer l'ensemble des termes dérivés –

l'automate Follow (cf. [27]) est également un exemple d'automate qui est un quotient de l'automate des positions qui peut être constuit en utilisant la ZPC-structure – . Les termes dérivés sont alors représentés par des chemins sur la ZPC-structure ([17]) et décrits par une liste de sous-expressions dont la concaténation donne le terme dérivé. Une méthode de Paige et Tarjan développée dans [36] permet d'identifier les listes qui représentent la même expression. Cette méthode permet alors d'identifier les termes dérivés avec une complexité en $O(n^3)$. L'automate des termes dérivés est alors construit en utilisant le calcul, quadratique, des transitions de l'automate des positions de [51]. L'automate des termes dérivés est alors construit en complexité cubique.

L'identification des termes dérivés identiques, qui constitue l'étape la plus coûteuse de [18], peut être améliorée. Ainsi dans [17], le marquage des sous-expressions étoilées identiques de l'expression rationnelle permet de réaliser cette identification en complexité $O(n^2)$. L'identification des sous-expressions est poussée encore plus loin dans [28] où toutes les sous-expressions sont marquées de manière à ce que deux sous-expressions isomorphes partagent la même marque. Cette méthode permet d'améliorer encore la complexité du calcul en $O(ln)$ où l est la longueur littérale de l'expression rationnelle.

Dans [1], Allauzen et Mohri proposent un algorithme pour construire l'automate des termes dérivés d'une expression à partir de l'automate de Thompson (cf. [47]) de cette expression. Dans [1] c'est en fait à la fois l'automate des positions, l'automate des termes dérivés et l'automate Follow qui sont construits à partir de l'automate de Thompson de l'expression en utilisant uniquement des opérations de minimisation et d'élimination de transitions spontanées. Plus précisément, l'automate des termes dérivés est le résultat de l'élimination des transitions spontanées sur la minimisation de l'automate obtenu par élimination des transitions spontanées créées lors d'une concaténation dans la construction de l'automate de Thompson. Cette méthode a la même complexité que l'algorithme de [28] mais permet d'unifier la construction de différents automates à partir d'une expression rationnelle.

Pour le calcul de l'automate des termes dérivés, dans ce chapitre, c'est une méthode très similaire à celle de [28] que nous présentons. Le calcul des termes dérivés est réalisé à partir de chemins dans l'arbre syntaxique de l'expression et nous marquons les états de cet arbre pour identifier les termes dérivés. La méthode que nous allons décrire peut être vue comme une généralisation de l'algorithme de [28] dans le cas où les expressions ne sont pas construites modulo l'associativité du produit. Une autre différence réside dans le fait que, afin de pouvoir adapter notre algorithme au calcul de l'automate des termes dérivés cassés, nous ne séparons pas le calcul des

transitions de celui des états de l'automate, c'est-à-dire que nous n'utilisons pas directement le fait que l'automate des termes dérivés est un quotient de l'automate des positions. Ceci est important car l'automate des termes dérivés cassés n'est pas un quotient de l'automate des positions. Dans le cas de l'automate des termes dérivés, cette différence de conception ne change pas les calculs réalisés car dans les deux cas, c'est l'idée de l'algorithme de [51] – pour les transitions de l'automate des positions – qui est utilisé.

Afin de réaliser l'identification des termes dérivés, et des termes dérivés cassés, nous présentons tout d'abord une méthode de marquage de l'arbre syntaxique d'une expression rationnelle. Ce marquage, avec des réductions que nous décrivons, permet de représenter de manière canonique les concaténations – à droite – de sous-expressions et, ainsi, de les identifier. Cette identification nous permet alors de décrire l'algorithme de construction de l'automate des termes dérivés par l'utilisation de chemins sur l'arbre syntaxique. Pour finir, nous détaillons l'algorithme, adapté du précédent, qui permet de construire l'automate des termes dérivés cassés en temps quadratique.

4.1 Concaténation à droite de sous-expressions

Comme nous l'avons vu dans les chapitres précédents, par les propositions 17 et 26, les termes dérivés et les termes dérivés cassés d'une expression E sont des concaténations à droite de sous-expressions de E. Les sous-expressions de E sont identifiables sur l'arbre T_{E} par des nœuds, représentant un sous-arbre.

Dans cette section, nous présentons une manière de représenter des concaténations à droite de sous-expressions par des mots sur les nœuds de l'arbre syntaxique. Nous montrons ensuite comment l'identification des sous-expressions, par le marquage des sous-expressions égales de l'expression, et l'utilisation d'un système de réécriture simple, permet de donner une représentation canonique des concaténations à droite de sous-expressions.

4.1.1 Sous-expressions sur l'arbre syntaxique

Nous avons déjà défini l'arbre syntaxique T_{E} d'une expression rationnelle E sur A. Comme nous travaillons toujours modulo $\mathbf{T}$, les feuilles de T_{E} sont étiquetées soit par $\mathsf{1}$ soit par une lettre de A. Nous avons noté $\mathrm{fp}(\mathsf{E})$ l'ensemble des *feuilles propres* de T_{E}, c'est-à-dire les feuilles étiquetées par une lettre – ces feuilles propres sont souvent également appelées *positions* de E (cf. [26] par exemple) –.

Afin de décrire l'arbre syntaxique formellement, nous utiliserons des notations empruntées aux expressions rationnelles :
(i) si x est un nœud de T_{E}, étiqueté par $+$ et dont les fils gauche et droit sont respectivement y et z, nous notons $x = y + z$;
(ii) de la même manière, nous notons $x = y \cdot z$ un nœud x, étiqueté par $\cdot$ et dont les fils sont y et z ;
(iii) enfin nous notons $x = y^*$ un nœud étiqueté par $*$ dont le fils est le nœud y.

Ces notations sont raisonnables car les nœuds sont utilisés pour représenter des sous-expressions de E et on a naturellement $[\![y + z]\!] = [\![y]\!] + [\![z]\!]$, $[\![y \cdot z]\!] = [\![y]\!] \cdot [\![z]\!]$ et $[\![y^*]\!] = [\![y]\!]^*$. Nous notons également $\phi(x)$ le père du nœud x.

Par simplification, nous noterons a une feuille x étiquetée par a et $\mathsf{1}$ une feuille étiquetée par $\mathsf{1}$ lorsque seule l'étiquette de ce nœud est importante.

Cependant ces notations ne sont pas de même nature que les précédentes, il s'agit juste de simplifications d'écriture lorsque seule l'étiquette d'un nœud est importante. Ces simplifications ne peuvent pas permettre de décrire l'arbre puisque deux feuilles différentes avec la même étiquette ne sont pas identifiables par cette notation.

Exemple 21. Dans cette section nous posons $A = \{a, b, c\}$ et $\mathsf{E}_1 = ((a \cdot b) \cdot (c^*)) + (b \cdot (c^*))$. La Figure 4.1 montre l'arbre syntaxique de E_1. Les nœuds sont notés $x_1, \ldots, x_{11}$ et les feuilles propres de l'arbre sont : $\mathrm{fp}(\mathsf{E}_1) = \{x_6, x_8, x_9, x_{10}, x_{11}\}$.

On a, par exemple : $[\![x_2]\!] = (a \cdot b) \cdot c^*$ et $[\![x_3]\!] = b \cdot c^*$.

Et, avec nos notations, dans $\mathrm{T}_{\mathsf{E}_1}$, on a $x_2 = x_4 \cdot x_5$ et $x_5 = x_{10}^*$.

Comme nous utilisons les nœuds de l'arbre pour représenter les sous-expressions de E, nous pouvons étendre au domaine des nœuds certaines définitions et notations déjà données sur les expressions rationnelles : le *terme constant* $\mathsf{c}(x)$ d'un nœud x de l'arbre syntaxique T_{E} de l'expression E sur A est le booléen défini par :

$$\mathsf{c}(\mathsf{0}) = 0\,, \quad \mathsf{c}(\mathsf{1}) = 1\,, \quad \forall x \in \mathrm{fp}(\mathsf{E}) \quad \mathsf{c}(x) = 0\ ,$$
$$\mathsf{c}(x + y) = \mathsf{c}(x) \vee \mathsf{c}(y)\,, \quad \mathsf{c}(x \cdot y) = \mathsf{c}(x) \wedge \mathsf{c}(y)\,, \quad \mathsf{c}(x^*) = 1\ .$$

Comme il est évident que $\mathsf{c}(x) = \mathsf{c}([\![x]\!])$, cette définition est pertinente. Cette définition donne directement une méthode de calcul des termes constants de tous les nœuds de l'arbre de bas en haut, c'est en fait la méthode habituelle pour calculer le terme constant d'une expression.

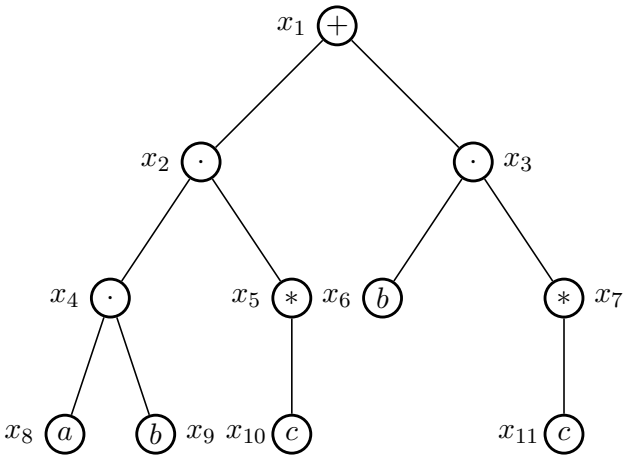


FIGURE 4.1 – L'arbre syntaxique $\mathrm{T}_{\mathsf{E}_1}$

Nous étendons également l'ensemble des *premières positions d'une expression*, défini dans les préliminaires pour la construction de l'automate des positions, à l'ensemble des *premières positions* d'un nœud x :

$$\begin{aligned}
\mathsf{First}(0) = \mathsf{First}(1) &= \emptyset \ , \\
\mathsf{First}(x) &= \{x\}, \quad \forall x \in \mathrm{fp}(\mathsf{E}) \ , \\
\mathsf{First}(y + z) &= \mathsf{First}(y) \cup \mathsf{First}(z) \ , \\
\mathsf{First}(y \cdot z) &= \mathsf{First}(y) \cup \mathsf{c}(y)\mathsf{First}(z) \ , \\
\mathsf{First}(y^*) &= \mathsf{First}(y) \ ,
\end{aligned}$$

De nouveau, nous avons $\mathsf{First}(x) = \mathsf{First}(\llbracket x \rrbracket)$. Nous noterons également $\mathsf{First}(x, a)$ le sous-ensemble de $\mathsf{First}(x)$ des feuilles étiquetées par a.

Exemple 22 (*Ex. 21 cont.*)**.** Dans l'arbre $\mathrm{T}_{\mathsf{E}_1}$, on a : $\mathsf{c}(x_1) = \mathsf{c}(x_2) =$

$\mathsf{c}(x_3) = \mathsf{c}(x_4) = \mathsf{c}(x_6) = \mathsf{c}(x_8) = \mathsf{c}(x_9) = \mathsf{c}(x_{10}) = \mathsf{c}(x_{11}) = 0$ et $\mathsf{c}(x_5) = \mathsf{c}(x_7) = 1$.

Et, par exemple, $\mathsf{First}(x_1) = \{x_8, x_6\}$ et par conséquent : $\mathsf{First}(x_1, a) = \{x_8\}$, $\mathsf{First}(x_1, b) = \{x_6\}$ et $\mathsf{First}(x_1, c) = \emptyset$.

Remarque 6. Pour être précis, le nœud x est lié à l'arbre duquel il est extrait, cependant, en pratique, il pourra être utilisé dans un autre arbre, en particulier l'arbre dont il est la racine, qui est un sous-arbre de l'arbre initial. Comme les fonctions décrites ci-dessus se calculent de bas en haut dans l'arbre, il n'est pas nécessaire de préciser l'arbre dans lequel on effectue le calcul.

Puisque les nœuds de l'arbre syntaxique permettent de représenter les sous-expressions d'une expression E, nous utilisons les mots sur l'alphabet dont les lettres sont les nœuds de T_{E} pour représenter les concaténations à droite de sous-expressions. Les mots sur les nœuds sont définis comme des mots sur l'alphabet dont les lettres sont les nœuds de l'arbre. Afin d'éviter toute confusion, la concaténation sur ces mots est notée :: et le mot $x :: y$ est la concaténation du mot x et du mot y. De même nous choisissons ε comme notation pour le mot vide – pour le distinguer du mot vide 1_{A^*} –.

Les mots sur les nœuds permettent de dénoter une expression, définie inductivement par :

(i) l'expression dénotée par le mot vide est $[\![\varepsilon]\!] = \mathsf{1}$;

(ii) le mot $x_1 :: x_2 :: \cdots :: x_n$ dénote l'expression $[\![x_1 :: x_2 :: \cdots :: x_n]\!] = [\![x_1 :: x_2 :: \cdots :: x_{n-1}]\!] \cdot [\![x_n]\!]$.

De par la définition de l'expression dénotée, les mots sur les nœuds de T_{E} dénotent des concaténations à droite de sous-expressions de E. Cependant cette représentation n'est pas unique et une même concaténation peut être dénotée par plusieurs mots sur les nœuds.

Enfin les notions de terme constant et de premières positions sont étendues aux mots sur les nœuds, de manière à rester cohérent avec les expressions qu'ils dénotent :

(i) $\mathsf{c}(x_1 :: x_2 :: \cdots :: x_n) = \mathsf{c}(x_1) \wedge \mathsf{c}(x_2) \wedge \cdots \wedge \mathsf{c}(x_n)$,

(ii) $\mathsf{First}(\varepsilon) = \emptyset$,

(iii) $\mathsf{First}(x_1 :: x_2 :: \cdots :: x_n) = \mathsf{First}(x_1 :: x_2 :: \cdots :: x_{n-1}) \cup \mathsf{c}(x_1 :: x_2 :: \cdots :: x_{n-1})\mathsf{First}(x_n) = \mathsf{First}(x_1) \cup \mathsf{c}(x_1)\mathsf{First}(x_2 :: \cdots :: x_n)$.

Exemple 23 (*Ex. 21 cont.*)**.**

$$\begin{aligned}
[\![x_8 :: x_9 :: x_5]\!] &= [\![x_8 :: x_9]\!] \cdot [\![x_5]\!] = (a \cdot b) \cdot c^* \ , \\
[\![x_9 :: x_5]\!] &= b \cdot c^* = [\![x_3]\!] \ . \\
\mathsf{c}(x_9 :: x_5) &= \mathsf{c}(x_9) \wedge \mathsf{c}(x_5) = 0 \ , \\
\mathsf{First}(x_9 :: x_5) &= \{x_9\} \ .
\end{aligned}$$

4.1.2 Représentation canonique

En vue de construire l'automate des termes dérivés de E, et comme les termes dérivés – et termes dérivés cassés – sont des concaténations à droite de sous-expressions de E, nous allons les représenter par des mots sur les nœuds de T_{E}. Cependant, pour construire exactement l'ensemble des états de l'automate voulu, il faut être capable d'identifier deux mots sur les nœuds dénotant la même expression. C'est ce que nous allons maintenant faire en proposant une forme canonique pour représenter toute concaténation droite de sous-expressions de E.

Cette idée de représentation canonique des concaténations de sous-expressions rationnelles est sous-jacente dans [28] mais appliquée directement aux mots particuliers qui donnent les termes dérivés. Nous préférons développer une représentation canonique pour toutes les concaténations à droite de sous-expressions afin de pouvoir également appliquer ce résultat pour la construction de l'automate des termes dérivés cassés – qui ne sont pas représentés par les mêmes mots que les termes dérivés –.

Marquer les nœuds

Afin d'avoir une forme canonique pour représenter une concaténation à droite de sous-expressions de E, nous allons 'marquer' les différentes sous-expressions de E. Cette notion de marquage est introduite dans [17] où seules les sous-expressions étoilées sont marquées. Comme dans [28], nous allons repérer tous tous les nœuds représentants des sous-expressions identiques par un même identifiant.

Pour cela nous affectons à chaque nœud x une *marque*, notée $\overline{x}$ telle que, si x et y sont deux nœuds dont les expressions dénotées sont égales – c'est-à-dire que les sous-arbres sont isomorphes –, alors $\overline{x} = \overline{y}$. C'est-à-dire que la marque d'un nœud est directement associée à l'expression rationnelle dénotée par le sous-arbre dont il est la racine. C'est donc une manière de repérer les sous-expressions dans l'arbre syntaxique.

Comme, pour que deux nœuds x et y aient la même marque, il faut, soit que ce soient des feuilles avec la même étiquette, soit qu'ils aient la même étiquette et que leurs fils aient les mêmes marques, il est possible de marquer tout l'arbre en marquant les feuilles en premier puis en remontant dans l'arbre.

Une méthode similaire pour marquer les nœuds de l'arbre est donnée dans [28] par minimisation d'un automate – sans circuits – construit à partir de l'arbre syntaxique : l'arbre syntaxique est transformé en un automate avec un unique état initial dont les transitions sortantes pointent vers les feuilles avec la même étiquette que la feuille en question. Les arêtes de l'arbre sont transformées en transitions ascendantes dont l'étiquette dépend de l'étiquette du père et de la nature du fils (droit ou gauche). L'état final est la racine. En effectuant le quotient de cet automate on récupère les classes d'équivalence des nœuds pour les marques.

Comme les marques représentent les sous-expressions, nous étendons aux marques les définitions proposées précédemment sur les nœuds : $[\![\overline{x}]\!] = [\![x]\!]$ est l'expression rationnelle associée à $\overline{x}$. Il est également naturel d'étendre la définition du terme constant d'un nœud – ou d'une expression – : $\mathsf{c}(\overline{x}) = \mathsf{c}(x)$.

Enfin il est également possible de définir des mots sur les marques de nœuds – les notations sont les mêmes que pour les mots sur les nœuds – et l'expression qu'ils représentent :

$$[\![\overline{x_1} :: x_2 :: \cdots :: \overline{x_n}]\!] = [\![x_1 :: x_2 :: \cdots :: x_n]\!] \ .$$

Il est donc finalement possible de représenter les concaténations droite de sous-expressions par des mots sur les marques.

Exemple 24 (*Ex. 21 cont.*)**.** On pourrait par exemple marquer l'arbre $\mathrm{T}_{\mathsf{E}_1}$ de la manière présentée sur la Figure 4.2.

Exemple 25. La Figure 4.3 montre l'arbre syntaxique de l'expression $((a \cdot b) \cdot c) + (a \cdot (b \cdot c))$. Seules les feuilles x_8 et x_6, x_9 et x_{10} , x_5 et x_{11} partagent la même marque.

Réduction préfixe

Si les marques permettent d'identifier les sous-expressions identiques, elles ne permettent pas directement d'identifier les concaténations de sous-expressions.

On peut noter que si les mots $\overline{x} :: \overline{y}$ et $\overline{z}$ dénotent la même expression, alors il existe sur l'arbre un nœud concaténation marqué $\overline{z}$ dont les fils gauche

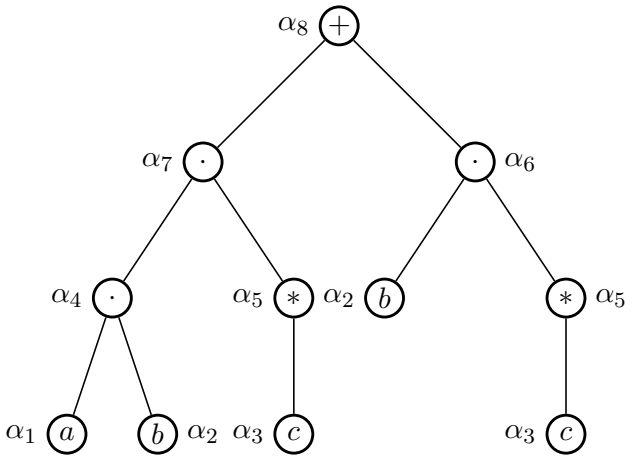


FIGURE 4.2 – Exemple de marquage de l'arbre syntaxique $\mathrm{T}_{\mathsf{E}_1}$

et droit sont marqués respectivement par $\overline{x}$ et $\overline{y}$. Il peut pourtant arriver que $[\![x :: y]\!] = [\![z]\!]$ sans que x et y soient les fils de z ni même qu'ils soient fils d'un même nœud concaténation.

Par exemple, sur la Figure 4.3, $I :: II$ et IV sont des mots de marques qui dénotent la même expression $(a \cdot b)$ et le nœud x_4 est marqué IV et ses fils sont marqués I et II. Par contre, les nœuds x_6 et x_{10} sont aussi marqués respectivement I et II mais ne sont pas les fils d'un même nœud. De même, la même expression $((a \cdot b) \cdot c)$ dénote les mots de marques $I :: II :: III$, $IV :: III$ et VII qui peuvent eux-mêmes être associés à différents mots de nœuds.

C'est pourquoi nous introduisons une réduction sur les mots de marques qui permet d'obtenir une représentation canonique des concaténations à droite de sous-expressions d'une expression rationnelle.

Considérons $\mathcal{R}$ l'ensemble des règles de réécriture sur les mots de marques

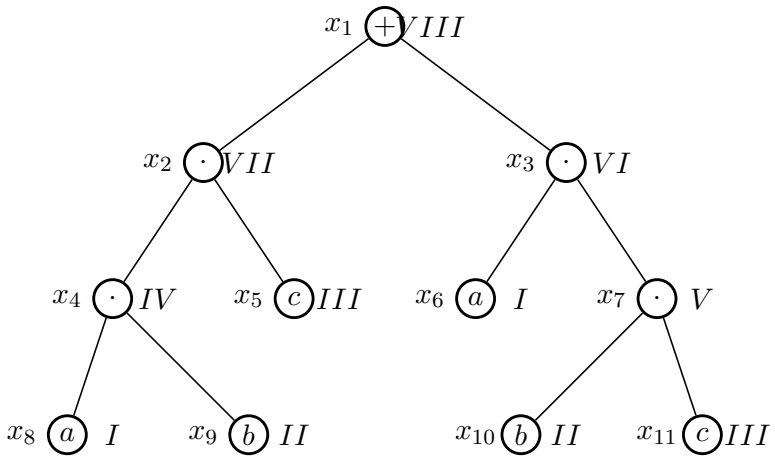


FIGURE 4.3 – Exemple de marquage pour $((a \cdot b) \cdot c) + (a \cdot (b \cdot c))$

$\alpha_1 :: \alpha_2 \to \beta$ avec α_1, α_2 et β telles qu'il existe $y = x_1 \cdot x_2$ avec $\overline{x_1} = \alpha_1$, $\overline{x_2} = \alpha_2$ et $\overline{y} = \beta$ c'est-à-dire des nœuds dont les marques sont respectivement α_1 et α_2 et qui sont fils gauche et droit d'un nœud étiqueté par une concaténation et dont la marque est β.

Notons $\rho\left(\overline{X}\right)$ le résultat de la réécriture préfixe du mot sur les marques $\overline{X}$ par les règles $\mathcal{R}$. Ainsi si les deux premières lettres de $\overline{X}$ correspondent à un nœud concaténation dans l'arbre, on remplace ces deux lettres par la marque du nœud et l'on recommence la réécriture sur le mot ainsi formé. Cette réécriture termine puisqu'elle transforme deux lettres en une seule. De plus on a :

$$\llbracket \rho\left(\overline{X}\right) \rrbracket = \llbracket X \rrbracket \ .$$

Exemple 26 (*Ex. 21 cont.*)**.** Les règles $\mathcal{R}_1$ pour l'arbre $\mathrm{T}_{\mathsf{E}_1}$ sont :

$$\alpha_1 :: \alpha_2 \to \alpha_4 \ , \quad \alpha_2 :: \alpha_5 \to \alpha_6 \ , \quad \alpha_4 :: \alpha_5 \to \alpha_7 \ .$$

Et un exemple de réduction serait :

$$\rho\left(\overline{x_9 :: x_5 :: x_9 :: x_5}\right) = \rho\left(\alpha_2 :: \alpha_5 :: \alpha_2 :: \alpha_5\right) = \alpha_6 :: \alpha_2 :: \alpha_5 \ .$$

Cet exemple montre bien que la réduction est préfixe car $\alpha_2 :: \alpha_5$ n'est réduit qu'en début de mot.

Notons $\overline{1}$ la marque de n'importe quelle feuille étiquetée par 1. Nous appellerons *mot propre sur les marques* un mot dont $\overline{1}$ n'est pas une lettre. Les expressions rationnelles étant, comme toujours, définie modulo les identités triviales **T**, il est possible de transformer un mot non propre sur les marques en un mot propre en éliminant les occurrences de $\overline{1}$ tout en conservant l'expression dénotée.

Pour montrer que $\rho\left(\overline{X}\right)$ est une représentation canonique de $[\![\overline{X}]\!]$ partagée par tous les mots de marques qui dénotent la même expression, nous prouvons le théorème suivant :

Théorème 34. *Soient $\overline{X}$ et $\overline{Y}$ deux mots propres sur les marques de l'arbre syntaxique d'une expression rationnelle.*

$$[\![\overline{X}]\!] = [\![\overline{Y}]\!] \Leftrightarrow \rho\left(\overline{X}\right) = \rho\left(\overline{Y}\right)$$

Démonstration. Prouvons le sens direct par l'absurde : soient $\overline{X} = \overline{x_1} \cdots \overline{x_n}$ et $\overline{Y} = \overline{y_1} \cdots \overline{y_m}$ deux mots propres sur les marques, différents, réduits et tels que $[\![\overline{X}]\!] = [\![\overline{Y}]\!]$.

- Supposons que $n > 1$ et $m > 1$, alors, comme $\overline{X}$ est propre, $[\![\overline{X}]\!] = [\![\overline{x_1} \ldots \overline{x_{n-1}}]\!] \cdot [\![\overline{x_n}]\!]$ donc $[\![\overline{X}]\!]$ est une concaténation dont le terme de droite est $[\![\overline{x_n}]\!]$ et de même $[\![\overline{Y}]\!]$ est une concaténation dont le terme de droite est $[\![\overline{y_m}]\!]$. Il est donc nécessaire que $\overline{x_n} = \overline{y_m}$ et que $\overline{x_1} \cdots \overline{x_{n-1}}$ et $\overline{y_1} \cdots \overline{y_{m-1}}$ soient deux mots, différents, réduits – car un préfixe d'un mot réduit est réduit – et dénotent la même expression. Par hypothèse de récurrence, on peut supposer donc que $n = 1$.
- Si $n = m = 1$ alors $[\![\overline{x_1}]\!] = [\![\overline{y_1}]\!]$ et donc, par définition, $\overline{x_1} = \overline{y_1}$.
- Si $n = 1$ et $m = 2$ alors $[\![\overline{x_1}]\!] = [\![\overline{y_1}]\!] \cdot [\![\overline{y_2}]\!]$. Il existe alors un nœud x_1 tel que $[\![x_1]\!] = [\![\overline{y_1}]\!] \cdot [\![\overline{y_2}]\!]$ et donc x_1 est un nœud concaténation dont le fils gauche représente $[\![\overline{y_1}]\!]$ et le fils droit $[\![\overline{y_2}]\!]$ c'est-à-dire que leurs marques respectives sont $\overline{y_1}$ et $\overline{y_2}$. Contradiction avec l'hypothèse que $\overline{Y}$ est réduit.
- Si $n = 1$ et $m > 2$ alors supposons que $\overline{Y}$ soit un mot de longueur minimale qui dénote la même expression qu'un mot $\overline{X}$ de longueur 1. Dans ce cas $[\![\overline{Y}]\!]$ est une concaténation dont le terme de gauche est $[\![\overline{y_1} \ldots \overline{y_{m-1}}]\!]$, il y a donc un nœud x_1 dans l'arbre dont le fils

gauche dénote cette dernière expression. C'est en contradiction avec l'hypothèse de minimalité.

Comme $\|\rho\left(\overline{X}\right)\| = \|X\|$, la réciproque est directe. □

Nous considérons dans ce chapitre uniquement les mots de nœuds propres car les mots que nous calculons pour représenter les termes dérivés et les termes dérivés cassés sont propres. Cette hypothèse simplifie grandement les réductions pour obtenir un mot de marque canonique pour une expression.

Dans le cas des expressions à multiplicités, les mots qui représentent les termes dérivés et les termes dérivés cassés ne sont pas propres et nous verrons dans le chapitre suivant qu'il faut alors rajouter des règles de transformations, sur les mots de marques, pour obtenir la représentation canonique souhaitée.

4.2 Calcul de $\mathcal{D}(\mathsf{E})$ et $\mathcal{D}_b(\mathsf{E})$

Afin d'établir un algorithme pour calculer l'automate des termes dérivés cassés, nous allons d'abord montrer, dans cette section, comment les mots sur les nœuds et les mots sur les marques permettent de calculer efficacement l'ensemble des termes dérivés et les transitions de l'automate des termes dérivés. Une fois l'algorithme pour les termes dérivés établi, nous détaillerons les modifications qui permettent d'obtenir la construction de l'automate des termes dérivés cassés.

4.2.1 Calculs sur l'arbre syntaxique décoré

Nous présentons ici la méthode de Champarnaud et Ziadi (cf. [17] ou [16] par exemple) qui, inspirée des calculs réalisés sur l'arbre syntaxique pour obtenir l'automate standard (cf. [51]), permet de calculer les termes dérivés d'une expression en complexité quadratique.

Pour ce faire, nous allons définir pour chaque position de l'arbre syntaxique un *chemin* dans celui-ci qui remonte à la racine. Ce chemin permet de collecter quelques nœuds de l'arbre qui définissent alors un mot sur les nœuds qui, nous le verrons par la suite, représente un terme dérivé.

Plus précisément, il est possible de 'décorer' l'arbre syntaxique en ajoutant des arcs orientés, appelés *liens*,

(i) du fils de chaque nœud étiqueté par $*$ à son père ;

(ii) de chaque fils gauche de nœud concaténation au fils droit de ce dernier.

Cette décoration de l'arbre syntaxique est une manière de représenter la *ZPC-structure* décrite dans [51]. La ZPC-structure est constituée de deux

copies orientées de l'arbre syntaxique telles que dans l'une les déplacements sont descendants et dans l'autre ascendants. Il existe des liens entre les deux copies. Dans [51] la ZPC-structure d'une expression rationnelle permet de construire directement l'automate des positions de l'expression.

L'arbre syntaxique décoré que nous avons défini est une manière de voir la ZPC-structure en superposant les deux copies. Par exemple, l'arbre décoré de E_1 est montré sur la Figure 4.4.

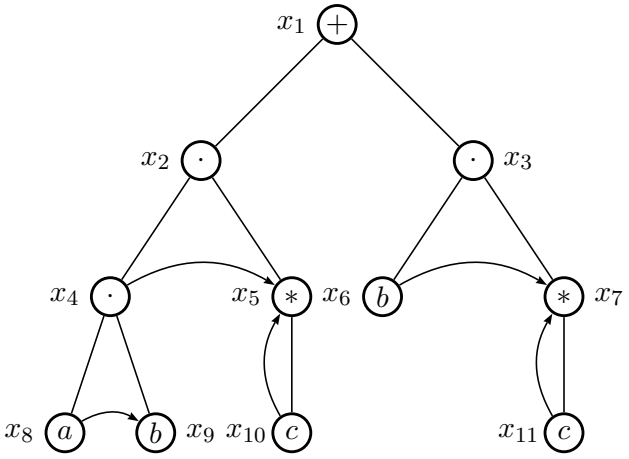


FIGURE 4.4 – Arbre syntaxique décoré de $\mathrm{T}_{\mathsf{E}_1}$

Dans l'arbre décoré, il existe un unique chemin qui part d'un nœud x, qui prend les arcs orientés décrits ci-dessus lorsque c'est possible, et si ce n'est pas possible remonte au père, jusqu'à la racine de l'arbre. Si l'on décrit ce chemin comme un mot dont les lettres sont les nœuds successivement atteints en empruntant un arc orienté (cf. [17]), on obtient le mot $\pi(x)$, que nous définissons formellement ci-dessous.

Soit E une expression rationnelle, T_{E} l'arbre syntaxique qui la représente.

On note r la racine de T_{E}. Le *chemin associé* à un nœud x de T_{E} est le mot sur les nœuds défini par :

$$\pi_r(x) = \begin{cases} \varepsilon & \text{si } x = r, \\ y :: \pi_r(\phi(y)) & \text{si } \phi(x) = x \cdot y, \\ y :: \pi_r(y) & \text{si } y = x^*, \\ \pi_r(\phi(x)) & \text{sinon.} \end{cases}$$

Lorsqu'il n'y a pas de confusion sur l'arbre que l'on considère, on note $\pi(x)$ à la place de $\pi_r(x)$.

Pour tout x, les chemin $\pi(x)$ est donc un mot de nœuds qui sont tous extrémités d'un lien. Comme les expressions – et leurs arbres – sont construites modulo $\mathbf{T}$, le mot $\pi(x)$ est propre pour tout x car les liens pointent soit sur une étoile, soit sur le fils droit d'un nœud concaténation, un tel fils ne peut pas être étiqueté par 1.

Remarque 7. Si la concaténation d'expressions rationnelles est associative, c'est-à-dire que les expressions sont construites modulo $\mathbf{A_p}$, alors l'arbre syntaxique d'une expression est tel qu'aucun nœud concaténation n'a pour fils droit un autre nœud concaténation – les concaténations consécutives sont représentées par un peigne orienté vers la gauche –. En particulier, aucune lettre de $\pi(x)$ ne peut être associée à une concaténation.

C'est pourquoi dans [28] où le produit est associatif, il n'y a pas besoin d'utiliser la réduction préfixe par $\mathcal{R}$ car, dans le calcul des chemins, aucune lettre ne représente une concaténation. En reprenant la Figure 4.3, si le produit est associatif, les deux sous-arbres gauche et droit, de racines respectives x_2 et x_3, représentent la même expression rationnelle et c'est le sous-arbre de racine x_2 qui doit en fait être construit dans les deux cas.

Exemple 27 (*Ex. 21 cont.*)**.** Dans $\mathrm{T}_{\mathsf{E}_1}$ on a les chemins associés suivants :

$$\pi(x_6) = \pi(x_{11}) = x_7, \quad \pi(x_8) = x_9 :: x_5, \quad \pi(x_9) = \pi(x_{10}) = x_5, \quad \pi_{x_4}(x_8) = x_9 \ ,$$

pour lesquels on a, par exemple,

$$[\![\pi(x_8)]\!] = b \cdot c^* \ , \quad [\![\pi(x_6)]\!] = [\![\pi(x_9)]\!] = [\![\pi(x_{10})]\!] = [\![\pi(x_{11})]\!] = c^* \ .$$

Dans l'exemple, x_6 et x_9 ont des chemins différents qui dénotent une même expression rationnelle.

Les chemins associés aux feuilles propres d'un arbre permettent de décrire les termes dérivés de l'expression rationnelle. C'est ce qu'énonce le théorème suivant qui est une réécriture avec nos notations de la Proposition 6 de [17], dont la preuve est également donnée par induction sur la profondeur de l'expression.

Théorème 35 ([17]). *Soit* E *une expression rationnelle dont l'arbre syntaxique* T_{E} *a pour racine* r*. On a :*

$$\frac{\partial}{\partial a}\mathsf{E} = \{[\![\pi_r(l)]\!] \mid l \in \mathsf{First}(r,a)\} \ .$$

Démonstration. Prouvons ce théorème par induction sur la profondeur de l'expression E :

- Si $\mathsf{E} = 1$, alors $\mathsf{First}(r,a) = \emptyset$ et $\frac{\partial}{\partial a}\mathsf{E} = \emptyset$.
- Si $\mathsf{E} = b$, avec $b \neq a$, alors $\mathsf{First}(r,a) = \emptyset$ et $\frac{\partial}{\partial a}\mathsf{E} = \emptyset$
- Si $\mathsf{E} = a$, alors $\mathsf{First}(r,a) = \{r\}$ et $\frac{\partial}{\partial a}\mathsf{E} = [\![\pi(r)]\!] = [\![\varepsilon]\!] = 1$.
- Si $\mathsf{E} = \mathsf{F} + \mathsf{G}$, alors $r = x + y$ avec $[\![x]\!] = \mathsf{F}$ et $[\![y]\!] = \mathsf{G}$. On a donc :

  $$\mathsf{First}(r,a) = \mathsf{First}(x,a) \cup \mathsf{First}(y,a) \ .$$

  Si $l \in \mathsf{First}(x,a)$, alors $\pi_r(l) = \pi_x(l)$ et donc $[\![\pi_r(l)]\!] = [\![\pi_x(l)]\!]$. De même, si $l \in \mathsf{First}(y,a)$ alors $[\![\pi_r(l)]\!] = [\![\pi_y(l)]\!]$ et donc par induction :

  $$\{[\![\pi_r(l)]\!] \mid l \in \mathsf{First}(r,a)\} = \frac{\partial}{\partial a}\mathsf{F} \cup \frac{\partial}{\partial a}\mathsf{G} \ .$$

- Si $\mathsf{E} = \mathsf{F} \cdot \mathsf{G}$, alors $r = x \cdot y$ avec $[\![x]\!] = \mathsf{F}$ et $[\![y]\!] = \mathsf{G}$. On a :

  $$\mathsf{First}(r,a) = \mathsf{First}(x,a) \cup \mathsf{c}(x)\mathsf{First}(y,a) \ .$$

  Si $l \in \mathsf{First}(x,a)$, alors $\pi_r(l) = \pi_x(l) :: y$ et donc $[\![\pi_r(l)]\!] = [\![\pi_x(l)]\!] \cdot \mathsf{G}$. Si $l \in \mathsf{First}(y,a)$ alors $\pi_r(l) = \pi_y(l)$ et donc $[\![\pi_r(l)]\!] = [\![\pi_y(l)]\!]$. Alors, par induction :

  $$\{[\![\pi_r(l)]\!] \mid l \in \mathsf{First}(r,a)\} = \frac{\partial}{\partial a}\mathsf{F} \cdot \mathsf{G} \cup \mathsf{c}(\mathsf{F})\frac{\partial}{\partial a}\mathsf{G} \ .$$

- Si $\mathsf{E} = \mathsf{F}^*$, alors $r = x^*$ avec $[\![x]\!] = \mathsf{F}$. On a :

  $$\mathsf{First}(r,a) = \mathsf{First}(x,a) \ .$$

  Si $l \in \mathsf{First}(x,a)$, alors $\pi_r(l) = \pi_x(l) :: r$ et alors $[\![\pi_r(l)]\!] = [\![\pi_x(l)]\!] \cdot \mathsf{E}$. Par induction :

  $$\{[\![\pi_r(l)]\!] \mid l \in \mathsf{First}(r,a)\} = \frac{\partial}{\partial a}\mathsf{F} \cdot \mathsf{E} \ .$$

□

Afin de pouvoir dériver par rapport à un mot non vide, et comme nous venons de voir que les dérivations par rapport à une lettre sont des mots de nœuds, nous 'dérivons' les mots de nœuds :

Théorème 36. *Soit* E *une expression rationnelle et* X *un mot sur les nœuds :*

$$\frac{\partial}{\partial a}[\![X]\!] = \{[\![\pi(l)]\!] \mid l \in \mathsf{First}(X, a)\} \ .$$

Démonstration. Par induction sur la longueur de X. Si $X = x_1$ alors ce théorème est une instance du précédent pour l'expression rationnelle $[\![x_1]\!]$. Pour $X = x_1...x_n$, nous avons par définition :

$$\mathsf{First}(X, a) = \mathsf{First}(x_1 :: \cdots :: x_{n-1}, a) \cup \mathsf{c}(x_1 :: \cdots :: x_{n-1})\mathsf{First}(x_n, a) \ .$$

L'arbre syntaxique représentant $[\![X]\!]$, dont la racine est notée p, peut être représenté, récursivement, par la concaténation de l'arbre représentant $[\![x_1 :: \cdots :: x_{n-1}]\!]$, dont nous noterons la racine q, et du sous-arbre de T_{E} dont la racine est x_n.

Si $l \in \mathsf{First}(x_1 :: \cdots :: x_{n-1}, a)$, alors $[\![\pi_p(l)]\!] = [\![\pi_q(l)]\!] \cdot [\![x_n]\!]$ et donc

$$\begin{aligned}\{[\![\pi(l)]\!] \mid l \in \mathsf{First}(X, a)\} = {} & \{[\![\pi(l)]\!] \cdot [\![x_n]\!] \mid l \in \mathsf{First}(x_1 :: \cdots :: x_{n-1}, a)\} \\ & \cup \mathsf{c}(x_1 :: \cdots :: x_{n-1})\{[\![\pi(l)]\!] \mid l \in \mathsf{First}(x_n, a)\} \ .\end{aligned}$$

C'est à dire :

$$\begin{aligned}\{[\![\pi(l)]\!] \mid l \in \mathsf{First}(X, a)\} &= \left[\frac{\partial}{\partial a}[\![x_1 :: \cdots :: x_{n-1}]\!]\right] \cdot [\![x_n]\!] \cup \mathsf{c}(x_1 :: \cdots :: x_{n-1})\frac{\partial}{\partial a}[\![x_n]\!] \\ &= \frac{\partial}{\partial a}[\![X]\!] \ .\end{aligned}$$

□

En particulier, une instance de ce théorème donne, pour une feuille propre h :

$$\frac{\partial}{\partial a}[\![\pi(h)]\!] = \{[\![\pi(l)]\!] \mid l \in \mathsf{First}(\pi(h), a)\} \ .$$

L'application du Théorème 35 avec cette dernière équation donne l'équivalent du théorème 9 de [18] :

Théorème 37 ([18])**.** *Soit* E *une expression rationnelle, on a :*

$$\mathrm{TD}\,(\mathsf{E}) = \{[\![\pi(l)]\!] \mid l \in \mathrm{fp}(\mathsf{E})\}$$

Exemple 28 (*Ex. 21 cont.*)**.**

$$\begin{aligned}\mathrm{TD}\,(\mathsf{E}_1) &= \{[\![\pi(l)]\!] \mid l \in \mathrm{fp}(\mathsf{E}_1)\} \\ &= \{[\![\pi(x_6)]\!], [\![\pi(x_8)]\!], [\![\pi(x_9)]\!], [\![\pi(x_{10})]\!], [\![\pi(x_{11})]\!]\} \\ &= \{b \cdot c^*, c^*\}\end{aligned}$$

4.2.2 Construction de l'automate des termes dérivés

Nous avons vu dans les sous-sections précédentes toutes les briques qui permettent maintenant de construire l'automate des termes dérivés. Cette construction se divise en deux étapes. La première étape est la construction de l'ensemble des états de l'automate. Cette étape utilise le résultat du Théorème 37 et l'identification canonique des concaténations à droite de sous-expressions décrite dans la section précédente et coïncide avec le calcul des états réalisé par la méthode présentée dans [28]. La deuxième étape est le calcul des transitions de l'automate. Cette étape utilise le Théorème 35 et reprend la méthode de calcul des transitions de l'automate des positions de [51].

Soit E une expression rationnelle dont l'arbre syntaxique T_{E} a pour racine r. Les états de l'automate $\mathcal{D}(\mathsf{E})$ des termes dérivés de E sont l'expression elle-même $\llbracket r \rrbracket$ et les $\llbracket \pi(l) \rrbracket$ pour toute feuille propre l de T_{E}. Il est possible que pour deux feuilles l et l', $\llbracket \pi(l) \rrbracket = \llbracket \pi(l') \rrbracket$. Cependant, il est possible de représenter de manière unique $\llbracket \pi(l) \rrbracket$ en prenant la réduction du mot marqué : $\rho\left(\overline{\pi(l)}\right) = \rho\left(\overline{\pi(l')}\right)$.

Il est donc possible de considérer que les états de $\mathcal{D}(\mathsf{E})$ ne sont plus les termes dérivés mais les mots réduits sur les marques qui les représentent. L'ensemble des états est construit en calculant les réductions par $\mathcal{R}$ des mots de marques résultant du calcul des chemins pour chaque feuille. L'état initial est $\rho\left(\overline{r}\right)$ et les états finaux sont ceux dont le terme constant est non nul $\mathsf{c}(q) = 1$ – le terme constant a été défini pour les mots de marques –.

Notons $\theta(q)$ l'ensemble des feuilles propres l de T_{E} dont la réduction du mot marqué associé au chemin est q. C'est-à-dire que si l et l' sont dans $\theta(q)$ alors : $\rho\left(\overline{\pi(l)}\right) = \rho\left(\overline{\pi(l')}\right) = q$.

Nous choisissons de définir $\mathsf{First}(q) = \mathsf{First}(\pi(l))$ où l est un représentant arbitrairement choisi dans $\theta(q)$. Il faut noter que, même si l et l' sont dans $\theta(q)$, les chemins $\pi(l)$ et $\pi(l')$ peuvent être différents et il n'y a alors pas nécessairement égalité de $\mathsf{First}(\pi(l))$ et $\mathsf{First}(\pi(l'))$. Le choix d'un représentant pour définir la fonction First pour toute la classe q est justifiée par la proposition suivante – qui découle directement du fait que si deux feuilles ont des chemins associés à la même expression, alors la dérivation par rapport à une lettre de ces chemins est nécessairement identique – :

Proposition 38. *Soit q un état de l'automate des termes dérivés. Si l et l' sont dans $\theta(q)$ alors :*

$$\{\llbracket \pi(k) \rrbracket \mid k \in \mathsf{First}(\pi(l), a)\} = \{\llbracket \pi(k) \rrbracket \mid k \in \mathsf{First}(\pi(l'), a)\} \quad .$$

Enfin nous pouvons décrire de manière constructive l'ensemble des transitions de l'automate $\mathcal{D}(\mathsf{E})$: soient q_i et q_j deux états, on a :

$$\begin{aligned} \llbracket q_j \rrbracket \in \frac{\partial}{\partial a} \llbracket q_i \rrbracket &\Leftrightarrow \exists l \in \mathsf{First}(\llbracket q_i \rrbracket, a), \quad \llbracket \pi(l) \rrbracket = \llbracket q_j \rrbracket \ , \\ &\Leftrightarrow \exists l \in \mathsf{First}(q_i, a), \quad l \in \theta(q_j) \ . \end{aligned}$$

C'est à dire qu'il y a une transition entre un état q_i et un état q_j étiquetée par a si et seulement si, il existe dans T_{E} une feuille propre l étiquetée par a telle que $l \in \mathsf{First}(q_i)$ et $l \in \theta(q_j)$.

Pour résumer, pour construire l'automate des termes dérivés d'une expression rationnelle E nous suivons les différentes étapes suivantes :

1. calcul pour tout nœud x de l'arbre syntaxique par un parcours de bas en haut :
   - du terme constant $\mathsf{c}(x)$;
   - de l'ensemble des feuilles propres $\mathsf{First}(x)$;
   - de la marque $\overline{x}$;
2. calcul du chemin $\pi(l)$ associé à toute feuille propre l de T_{E} et réduction préfixe de leur version marquée : $\rho\left(\overline{\pi(l)}\right)$;
3. par identification lexicographique, toutes les feuilles l et l' telles que $\rho\left(\overline{\pi(l)}\right) = \rho\left(\overline{\pi(l')}\right)$ sont regroupées dans une même classe. Les classes de cette étape donnent les états de l'automate, il faut garder la composition de chaque classe $\theta(q)$ de feuilles ;
4. pour chaque état q, calcul de $\mathsf{First}(\rho\left(\overline{\pi(l)}\right))$ pour un représentant $l \in \theta(q)$ donné ;
5. chaque état q dont le terme constant $\mathsf{c}(q)$ est non nul est identifié comme un état final ;
6. pour chaque paire d'états (q_i, q_j) calcul des intersections $\mathsf{First}(q_i, a) \cap \theta(q_j)$. Si l'intersection est non vide, alors il y a une transition de q_i à q_j étiquetée a.

Si n est la taille de l'expression rationnelle E et l sa longueur littérale alors, le point 1 est réalisé par un parcours en profondeur sur l'arbre, donc en $O(n)$. Le calcul du chemin pour une feuille donnée est également en $O(n)$ et la réduction préfixe est linéaire. L'étape 2 a donc un complexité de $O(ln)$. L'identification lexicographique des états – étape 3 – est également en $O(ln)$. Finalement les étapes suivantes sont également en $O(ln)$ et l'algorithme obtenu est donc en $O(ln)$, comme l'algorithme de [28].

Exemple 29 (*Ex. 21 cont.*)**.** Nous avons vu que les trois états de $\mathcal{D}(\mathsf{E}_1)$ sont α_8 – le marque de la racine de $\mathrm{T}_{\mathsf{E}_1}$ – qui est initial, $\alpha_2 :: \alpha_5$, qui représente $b \cdot c^*$ et α_5, qui représente c^* et est final.

La Figure 4.5 montre l'automate des termes dérivés de E_1 ainsi construit :

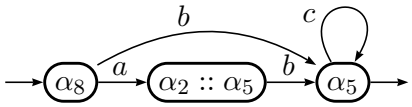


FIGURE 4.5 – L'automate $\mathcal{D}(\mathsf{E}_1)$

Le calcul de l'ensemble des états de l'automate des termes dérivés que nous avons décrit est, à l'associativité du produit près, la méthode exposée dans [28].

En revanche, la méthode de calcul des transitions diffère, conceptuellement, de celle de [28]. Dans [28] les transitions sont calculées presque indépendamment du calcul des états de l'automate. En fait ce sont les transitions de l'automate des positions qui sont calculées et c'est en réalisant le quotient de l'automate des positions, grâce aux classes d'équivalence des états, que sont obtenues les transitions de l'automate des termes dérivés. Dans les faits, la méthode de [28] réalise les mêmes calculs que ceux que dans la méthode que nous présentons car, l'automate des positions est calculé par la méthode exposée dans [51] qui utilise les mêmes chemins sur l'arbre pour construire l'automate des positions que ceux pour calculer les termes dérivés.

Comme l'automate des termes dérivés cassés n'est pas un quotient de l'automate des positions, nous avons choisi de proposer une méthode de calcul pour les termes dérivés qui s'adapte facilement aux autres cas que nous allons considérer – termes dérivés cassés et multiplicités –. C'est pourquoi nous ne séparons pas le calcul des états de celui des transitions.

4.2.3 Termes dérivés cassés

Nous allons maintenant décrire comment modifier la méthode décrite ci-dessus afin de calculer l'automate $\mathcal{D}_b(\mathsf{E})$ des termes dérivés cassés de l'expression rationnelle E.

Cassage d'un mot de nœuds

Comme nous avons représenté les vrais termes dérivés par un mot de nœuds de l'arbre syntaxique – décrit par un chemin sur celui-ci –, et que les vrais termes dérivés cassés sont le cassage des vrais termes dérivés, nous allons nous intéresser à la réalisation d'une sorte de cassage sur les mots de nœuds – qui donne donc, à partir d'un mot de nœuds, un ensemble de mots –. Ce cassage doit représenter le cassage de l'expression dénotée.

Exemple 30. Nous utiliserons maintenant comme exemple courant l'expression $\mathsf{E}_2 = (a^* + a \cdot (b + 1)) \cdot (a + b)$ dont l'arbre syntaxique est présenté sur la Figure 4.6.

Pour tout nœud x de l'arbre T_{E}, définissons l'ensemble des nœuds suivants :

$$\omega(x) = \begin{cases} \{x\} & \text{si } x \text{ est une feuille ou si } x = y^*, \\ \omega(y) & \text{si } x = y \cdot z, \\ \omega(y) \cup \omega(z) & \text{si } x = y + z. \end{cases}$$

C'est-à-dire que $\omega(x)$ est l'ensemble des nœuds obtenus à partir de x dans le sous-arbre dont il est racine en :
(i) continuant dans chaque branche d'un nœud étiqueté $+$;
(ii) continuant dans la branche gauche d'un nœud $\cdot$;
(iii) s'arrêtant sur tout autre nœud – feuille ou nœud $*$ –.

Exemple 31 (*Ex. 30 cont.*)**.**

$$\omega(x_1) = \omega(x_2) = \{x_4, x_9\}\,, \quad \omega(x_3) = \{x_6, x_7\}\,, \quad \omega(x_{10}) = \{x_{11}, x_{12}\}\ .$$

Nous allons maintenant introduire le cassage $\mathrm{B}(X)$ d'un mot X sur les nœuds. Comme un mot sur les nœuds représente une concaténation, l'idée est de 'casser' le premier terme de cette concaténation, c'est-à-dire la première lettre du mot. Si ce cassage donne le mot vide, alors il faut casser également la deuxième lettre et ainsi de suite.

Nous donnons donc une définition inductive :
(i) si $X = \varepsilon$, alors $\mathrm{B}(X) = \{\varepsilon\}$;
(ii) si $X = x_1 :: \cdots :: x_n$ alors :

$$\begin{aligned} \mathrm{B}(X) = &\{y :: \pi_{x_1}(y) :: x_2 ... x_n \mid y \in \omega(x_1), [\![y]\!] \neq 1\} \\ &\cup \{\mathrm{B}(\pi_{x_1}(y) :: x_2 ... x_n) \mid y \in \omega(x_1), [\![y]\!] = 1\} \end{aligned} \tag{4.1}$$

Exemple 32 (*Ex. 30 cont.*)**.**

$$\begin{aligned}\mathrm{B}(x_1) &= \{x_4 :: \pi_{x_1}(x_4), x_9 :: \pi_{x_1}(x_9)\} = \{x_4 :: x_3, x_9 :: x_{10} :: x_3\} \ , \\ \mathrm{B}(\pi(x_9)) &= \mathrm{B}(x_{10} :: x_3) \\ &= \{x_{11} :: \pi_{x_{10}}(x_{11}) :: x_3, \mathrm{B}(\pi_{x_{10}}(x_{12}) :: x_3)\} \\ &= \{x_{11} :: x_3, x_6, x_7\} \ .\end{aligned}$$

Exemple 33 (*Ex. 30 cont.*)**.** Comme pour les termes dérivés, il est possible de voir les opérations qui nous intéressent pour le calcul des termes dérivés cassés de manière graphique sur l'arbre syntaxique.

Ainsi, pour réaliser le cassage des chemins – que nous avons définis à l'aide de liens sur l'arbre dans le cas des termes non cassés –, nous allons introduire la notion de lien secondaires – tous les liens précédemment définis existent sous l'appellation lien primaires –.

Ces liens secondaires partent de chaque nœud duquel part un lien primaire, et, si y est le nœud destination de ce lien primaire, alors les liens secondaires pointent vers les $\omega(y)$. En particulier, les liens secondaires qui partent d'un nœud sous une étoile se confondent avec le lien primaire.

Les chemins cassés sont alors définis comme les chemins qui :

(i) partent du nœud qui nous intéresse ;
(ii) remontent vers la racine ;
(iii) empruntent un lien secondaire dès que possible jusqu'à en avoir emprunté un ne pointant pas vers 1 ; à partir de là :
(iv) empruntent les liens primaires dès que possible.

Comme pour les termes dérivés, chaque chemin est représenté par le mot composé par les extrémités de liens primaires et secondaires. L'ensemble des mots correspondant aux liens secondaires qui partent d'un nœud x forme l'ensemble des chemins cassés de x : $\mathrm{B}(x)$.

En reprenant l'exemple $\mathsf{E}_2 = (a^* + a \cdot (b + 1)) \cdot (a + b)$, la Figure 4.7 montre les liens primaires (en continu) et secondaires (en pointillés).

La définition donnée ci-dessus est justifiée par le théorème suivant :

Théorème 39. *Soit X un mot sur les nœuds de l'arbre syntaxique T_{E} d'une expression rationnelle E. Nous avons :*

$$\mathrm{B}(\llbracket X \rrbracket) = \bigcup_{Y \in \mathrm{B}(X)} \llbracket Y \rrbracket = \llbracket \mathrm{B}(X) \rrbracket$$

Démonstration. Par induction sur la profondeur de $\llbracket X \rrbracket$. Nous supposons que X est propre. Nous avons déjà vu qu'il est facile de transformer un mot en mot propre.

Si $[\![X]\!] = 1$, alors, comme X est propre, $X = \varepsilon$ et $\mathrm{B}(X) = \{\varepsilon\}$ et vérifie bien $\mathrm{B}([\![X]\!]) = \bigcup_{Y \in \mathrm{B}(X)} [\![Y]\!]$.

Si $[\![X]\!] = a$ **ou** $[\![X]\!] = \mathsf{F}^*$ alors $X = x_1$ où x_1 est un nœud étiqueté par a ou par une étoile. On a alors $\omega(x_1) = \{x_1\}$ et donc $\mathrm{B}(X) = \{x_1\}$ ce qui termine ce cas.

Si $[\![X]\!] = \mathsf{F} + \mathsf{G}$: alors $X = x_1$, où $x_1 = y_1 + y_2$, avec y_1 dénotant F et y_2 dénotant G. Dans ce cas, on a $\omega(x_1) = \omega(y_1) \cup \omega(y_2)$. Et comme pour tout $z \in \mathrm{T}_{y_i}$, on a $\pi_{x_1}(z) = \pi_{y_i}(z)$, alors $\mathrm{B}(X) = \mathrm{B}(y_1) \cup \mathrm{B}(y_2)$ et donc $\mathrm{B}([\![X]\!]) = \mathrm{B}([\![y_1]\!]) \cup \mathrm{B}([\![y_2]\!])u$.

Si $[\![X]\!] = \mathsf{F} \cdot \mathsf{G}$: on distingue deux cas suivant la longueur de X :

soit $X = x_1 :: \cdots :: x_n$, avec $n \geqslant 2$. De la définition du cassage, on a :

$$\begin{aligned}\mathrm{B}(x_1 :: x_2 :: ... :: x_n) &= \mathrm{B}(x_1) :: (x_2 :: ... :: x_n) \\ &\quad \cup\ \delta_{\mathrm{B}(x_1)} :: \mathrm{B}(x_2 :: ... :: x_n) \\ &= \mathrm{B}(x_1 :: ... :: x_{n-1}) :: x_n \\ &\quad \cup\ \delta_{\mathrm{B}(x_1 :: ... :: x_{n-1})} \mathrm{B}(x_n)\end{aligned}$$

,

alors, par récurrence, $\mathrm{B}(X) = \mathrm{B}\left([\![x_1 :: \cdots :: x_{n-1}]\!]\right) \cdot [\![x_n]\!] \cup \delta_{\mathrm{B}([\![x_1 :: \cdots :: x_{n-1}]\!])} \mathrm{B}\left([\![x_n]\!]\right)$.

soit $X = x_1$, alors comme $[\![X]\!]$ est une concaténation, le nœud x_1 est une concaténation : $x_1 = y_1 \cdot y_2$ et $\omega(x_1) = \omega(y_1)$. De plus, pour $z \in \mathrm{T}_{y_1}$, $\pi_{x_1}(z) = \pi_{y_1}(z) :: y_2$. On a alors :

$$\begin{aligned}\mathrm{B}(X) = &\{z :: \pi_{y_1}(z) :: y_2 \mid z \in \omega(y_1), [\![z]\!] \neq 1\} \\ &\cup\ \{\mathrm{B}(\pi_{y_1}(z) :: y_2) \mid z \in \omega(y_1), [\![z]\!] = 1\}\end{aligned}$$

ce qui correspond au premier cas avec $X = y_1 :: y_2$.

□

La Propriété 22 permet d'appliquer le théorème ci-dessus aux résultats obtenus pour les termes dérivés et d'obtenir directement :

Théorème 40. *Soit* E *une expression rationnelle dont l'arbre syntaxique* T_{E} *a pour racine* r *et soit* X *un mot sur les nœuds :*

$$\frac{\partial_{\mathsf{b}}}{\partial a}[\![X]\!] = \bigcup_{l \in \mathsf{First}(r,X)} \{[\![X]\!] \mid X \in \mathrm{B}(\pi(l))\} \quad .$$

et :

Théorème 41.

$$\mathrm{TBD}\,(\mathsf{E}) = \bigcup_{l \in \mathrm{fp}(\mathsf{E})} \{[\![X]\!] \mid X \in \mathrm{B}(\pi(l))\} \quad .$$

Construction de l'automate des termes dérivés cassés

Comme dans le cas des termes dérivés, nous venons de montrer que les vrais termes dérivés cassés peuvent être calculés par un mot sur les nœuds de l'arbre. Il est donc possible de représenter de manière unique les états de l'automate des termes dérivés cassés – il faut rajouter le cassage $\mathrm{B}(r)$ de l'expression elle-même, qui sera également l'ensemble des états initiaux – avec des mots réduits sur les marques.

Cependant la méthode diffère un peu de celle des termes dérivés car le cassage du chemin d'une feuille propre l peut donner plusieurs mots de marques réduits différents. Il ne faut donc plus identifier les états de l'automate a une classe de feuilles dont les chemins marqués et réduits sont identique mais directement les identifier aux différents chemins dont la réduction est identique.

C'est-à-dire que pour chaque état q on associe $\theta_b(q)$ l'ensemble des mots X tels qu'il existe une feuille propre l telle que $X \in \mathrm{B}(\pi(l))$ et $\rho\left(\overline{X}\right) = q$. En particulier, dans ce cas, $\mathsf{First}(q)$ est alors défini comme le First d'un représentant, c'est-à-dire qu'on choisit un X arbitrairement dans $\theta_b(q)$ et $\mathsf{First}(q) = \mathsf{First}(X)$.

L'application des théorèmes de la sous-section précédente permet de montrer directement, en appliquant les résultats montrés pour les termes dérivés, qu'il y a une transition dans $\mathcal{D}_b(\mathsf{E})$ de q_i à q_j étiquetée par a si et seulement si il existe une feuille $l \in \mathsf{First}(q_i, a)$ et un mot sur les nœud $X \in \mathrm{B}(\pi(l)) \cap \theta_b(q_j)$.

En résumé pour calculer l'automate des termes dérivés cassés d'une expression rationnelle E nous suivons les différentes étapes suivantes :

1. calcul pour tout nœud x de l'arbre syntaxique – par un parcours de bas en haut – :
   - du terme constant $\mathsf{c}(x)$;
   - de l'ensemble des feuilles propres $\mathsf{First}(x)$;
   - de la marque associée $\overline{x}$;
2. calcul pour tout nœud x de l'arbre – par un parcours de haut en bas – de l'ensemble $\omega(x)$;
3. calcul du chemin $\pi(l)$ associé à toute feuille propre l de T_{E} ; calcul du cassage $\mathrm{B}(\pi(l))$ et réduction préfixe des mots ainsi obtenus, pour

chaque mot X il faut retenir l'ensemble des feuilles l telles que $X \in \mathrm{B}(\pi(l))$;

4. identification des mots X et X' obtenus à l'étape précédente, tels que $\rho\left(\overline{X}\right) = \rho\left(\overline{X'}\right)$. Cette étape donne les états de l'automate, il faut garder la composition de chaque classe $\theta_b(q)$ de mot ;
5. pour chaque état q, calcul de $\mathsf{First}(X)$ pour un représentant $X \in \theta_b(q)$ donné ;
6. chaque état q dont le terme constant $\mathsf{c}(q)$ est non nul est identifié comme un état final ;
7. pour chaque paire d'états (q_i, q_j), il y a une transition étiquetée par a si on trouve une feuille de $\mathsf{First}(q_i, a)$ telle qu'il existe $X \in \mathrm{B}(\pi(l)) \cap \theta_b(q_j)$.

Par rapport à l'algorithme donné pour les termes dérivés, l'algorithme des termes dérivés cassés se distingue par le fait que chaque chemin peut engendrer plusieurs chemins cassés. Cependant ces chemins sont limités par le nombre de liens secondaires, ou plutôt par le nombre de nœuds dans lesquels arrive un lien secondaire, puisqu'un chemin cassé peut être vu comme un chemin 'normal' partant d'un nœud extrémité d'un lien secondaire. Il en résulte que l'algorithme reste quadratique mais en $O(n^2)$.

Exemple 34. Prenons l'exemple très simple de l'expression $\mathsf{E}_3 = (a + b) \cdot (c + d)$ dont l'arbre est présenté sur la Figure 4.8.

Sur cet exemple nous n'avons pas a nous préoccuper de marques particulières puisque tous les sous-arbres sont différents : les noms des nœuds sont les marques.

Il existe un unique lien primaire entre x_2 et x_3. On a $\omega(x_1) = \omega(x_2) = \{x_4, x_5\}$ et $\omega(x_3) = \{x_6, x_7\}$. Il y a donc des liens secondaires entre x_2 et x_6 d'une part et x_7 d'autre part. De plus les mots $x_4 :: x_3$ et $x_5 :: x_3$ sont les états initiaux de l'automate des termes dérivés cassés.

On a également $\pi(x_4) = \pi(x_5) = x_3$ et $\pi(x_6) = \pi(x_7) = \varepsilon$. Le cassage du mot x_3 donne $\mathrm{B}(x_3) = \{x_6, x_7\}$. Les termes dérivés cassés sont donc $\{x_4 :: x_3, x_5 :: x_3, x_6, x_7, \varepsilon\}$.

Comme $\mathsf{First}(x_4 :: x_3) = \mathsf{First}(x_5 :: x_3) = \{x_6, x_7\}$, on peut construire l'automate des termes dérivés cassés de la Figure 4.9.

Remarque 8. Dans la version 1.4 de la plateforme de manipulation d'automates Vaucanson, le calcul de l'automate des termes dérivés et le calcul de l'automate des termes dérivés cassés se font par un algorithme hybride entre l'algorithme de [5] et l'algorithme de [18] : les dérivations par rapport à chaque lettre de l'alphabet sont effectuées pour chaque terme dérivé (cassé)

jusqu'à ce qu'aucun nouveau ne soit créé, comme dans la méthode d'Antimirov. Les termes dérivés (cassés) sont néanmoins représentés par une liste de nœuds de l'arbre comme dans [18]. Le premier élément de la liste est alors décomposé tant qu'il s'agit d'un produit afin de garantir l'unicité de la décomposition – raisonnement similaire à celui que nous adoptons dans la première section de ce chapitre –. L'identification des termes dérivés (cassés) égaux est alors réalisée par une reconnaissance lexicographique de la liste des éléments de la liste un à un.

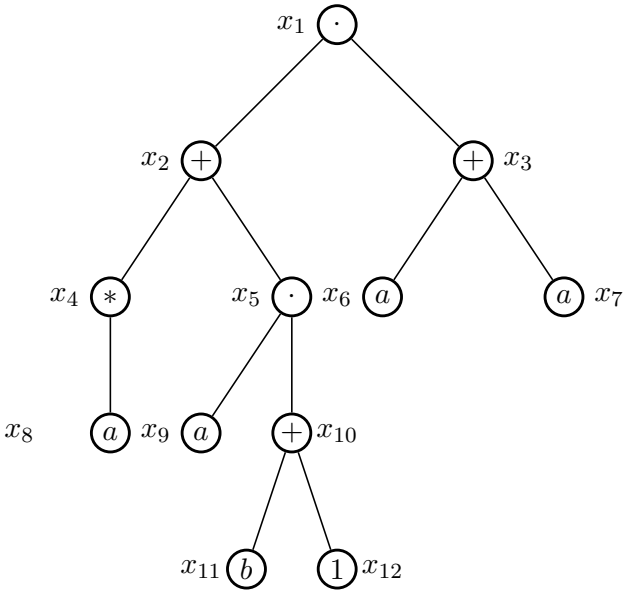


FIGURE 4.6 – l'arbre syntaxique $\mathrm{T}_{\mathsf{E}_2}$

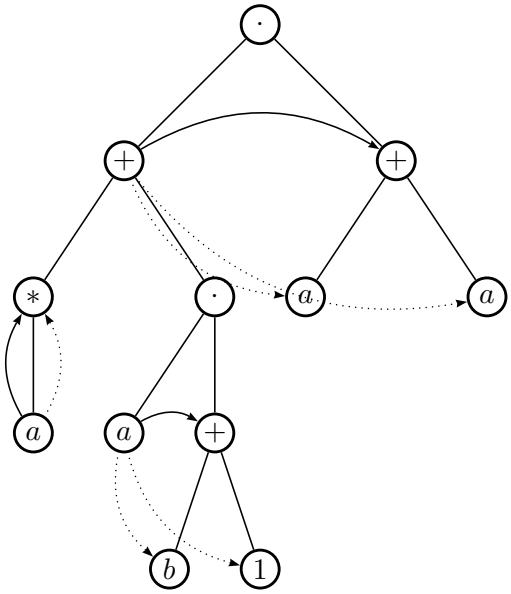


FIGURE 4.7 – l'arbre syntaxique décoré T_{E_2}

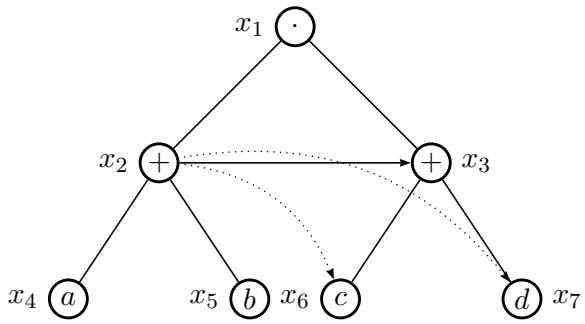


FIGURE 4.8 – L'arbre syntaxique $\mathrm{T}_{\mathsf{E}_3}$

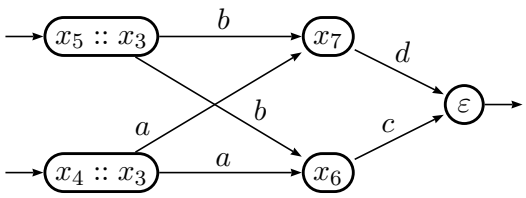


FIGURE 4.9 – L'automate des termes dérivés cassés de E_3

Chapitre 5

Multiplicités

Le but de ce chapitre est de généraliser les études et les résultats précédemment présentés au cas des expressions rationnelles à multiplicités dans un semi-anneau $\mathbb{K}$. La première chose qu'il faut faire, pour manipuler les expressions rationnelles à multiplicités, est de définir le cadre dans lequel on peut les considérer. En effet, l'étoile d'un scalaire de $\mathbb{K}$ n'étant pas nécessairement définie, il faut s'assurer tout d'abord que la notion d'étoile soit cohérente dans le semi-anneau $\mathbb{K}$. Ensuite il faut également s'assurer qu'étant donné une expression sur $\mathbb{K}$, les étoiles des scalaires soient définies.

Les termes dérivés d'une expression rationnelle à multiplicités ont été définis dans [31] afin de pouvoir construire un $\mathbb{K}$-automate qui soit un quotient de l'automate des positions de l'expression. Comme pour le cas booléen, les termes dérivés sont définis conjointement à une opération de dérivation. Cependant la relation entre termes dérivés et cette dérivation n'est pas aussi directe que dans le cas booléen. En effet la dérivation à multiplicités ne produit pas des ensembles d'expressions rationnelles mais des combinaisons linéaires – c'est-à-dire des ensembles dont chaque élément est pondéré par un élément de $\mathbb{K}$ – d'expression rationnelles. Par le jeu des coefficients qui peuvent s'annuler, un terme dérivé peut 'disparaître' de la combinaison linéaire obtenue par dérivation. C'est pourquoi, pour les expressions à multiplicités, les termes dérivés sont définis par une induction sur des ensembles d'expressions rationnelles. Ce sont les états de l'automate et les combinaisons linéaires obtenues par dérivation permettent d'obtenir les transitions de l'automate des termes dérivés.

Dans ce chapitre nous donnons un algorithme pour construire l'automate des termes dérivés d'une expression rationnelle à multiplicités. Cet algorithme utilise, comme dans le cas booléen, des chemins de nœuds de l'arbre

pour représenter les termes dérivés. Un marquage de ces nœuds et des règles de réduction permettent d'identifier les termes dérivés. Les transitions sont également calculées par des calculs sur l'arbre.

Finalement nous proposons une définition pour les termes dérivés cassés. Cette définition introduit le cassage d'une expression à multiplicités sous forme d'une combinaison linéaire d'expressions rationnelles. Les termes dérivés cassés permettant la construction d'un automate, nous donnons également un algorithme pour calculer cet automate. Cet algorithme est une synthèse des résultats pour les termes dérivés cassés booléens et pour les termes dérivés à multiplicités.

5.1 $\mathbb{K}$-expressions rationnelles

Afin de généraliser nos résultats, il nous faut d'abord donner une définition des expressions rationnelles à multiplicités. En particulier il nous faut énoncer les identités triviales qui seront utilisées. Certains choix sont justifiés lors de la définition des termes dérivés.

5.1.1 Définitions et notations

Lorsque l'on traite des séries rationnelles, plus particulièrement de l'étoile de celles-ci, il faut définir une distance, au moins implicitement, sur le semi-anneau des coefficients. Dans tout ce chapitre, $\mathbb{K}$ est donc un semi-anneau topologique, c'est-à-dire que $\mathbb{K}$ est muni d'une topologie pour laquelle l'addition et la multiplication sont continues, et A est un alphabet. L'addition sur $\mathbb{K}$ est notée $\oplus$ et la multiplication est notée par une simple concaténation. Par extension, $\oplus$ note également l'addition dans les séries à coefficients dans $\mathbb{K}$. Le semi-anneau $\mathbb{K}$ n'est, *a priori*, pas commutatif. Sur le semi-anneau topologique $\mathbb{K}$ pour tout élément k, la famille $\{k^n\}_{n\in\mathbb{N}}$ peut être, ou ne pas être, sommable. Lorsqu'elle est sommable, nous appellerons *étoile de* k sa somme, notée k^* :

$$k^* = \sum_{n\in\mathbb{N}} k^n \ .$$

Lorsqu'elle n'est pas sommable, nous dirons que l'étoile de k n'est pas définie.

Si $\mathbb{K}$ est un semi-anneau topologique *fort*, c'est-à-dire que le produit de deux familles sommables est sommable, alors il est possible de munir le semi-anneau des séries formelles $\mathbb{K}\langle\langle A^*\rangle\rangle$ de la topologie produit dérivée de la topologie de $\mathbb{K}$ – ce qui est plus intéressant que d'utiliser la topologie produit dérivée de la topologie discrète sur $\mathbb{K}$ – (cf. [43, III.1.3]).

Les opérations rationnelles peuvent alors être définies pour les séries rationnelles. Soient s et t deux séries :

$$\begin{aligned} <s \oplus t, w> &= <s, w> \oplus <t, w> \ , \\ <st, w> &= \sigma_{uv=w} <s, u><t, v> \ , \\ s^* &= \sigma_{n \in \mathbb{N}} s^n \ . \end{aligned}$$

L'étoile de la série s n'est définie que si la série est propre, c'est-à-dire si le coefficient de $1_{\mathbb{K}}$ dans la série s est nul.

Expressions rationnelles à multiplicités

Définition 18. *Une* expression rationnelle sur A^* à multiplicités *dans un semi-anneau* $\mathbb{K}$*, ou* $\mathbb{K}$-expression rationnelle*, est une formule bien parenthésée obtenue inductivement par la grammaire :*

$$\mathsf{E} \to \{a \mid a \in A\} \mid 0 \mid 1 \mid (\mathsf{E} + \mathsf{E}) \mid (\mathsf{E} \cdot \mathsf{E}) \mid (\mathsf{E}^*) \mid (k\mathsf{E}), k \in \mathbb{K} \mid (\mathsf{E}k), k \in \mathbb{K} \ .$$

On note $\mathbb{K}\,\mathsf{RatE}\,A^*$ l'ensemble des $\mathbb{K}$-expressions rationnelles sur A^*.

L'unité du semi-anneau $\mathbb{K}$ est notée $1_{\mathbb{K}}$ pour éviter toute confusion avec le mot vide 1_{A^*} et l'expression $\mathsf{1}$. Pour éviter également des confusions entre des scalaires et l'expression $\mathsf{1}$ (par exemple dans l'expression $(\mathsf{21})$, nous nous autorisons à noter l'expression $\mathsf{1}$ par 1_{A^*} (qui donne (21_{A^*})) même si nous sommes conscient que le mot vide et l'expression $\mathsf{1}$ ne sont pas les mêmes objets.

Exemple 35. Les formules suivantes sont des expressions rationnelles :

$$((3a) + (2b)) \qquad ((a \cdot ((a^*2) + 1_{A^*}))3) \ .$$

Remarque 9. Les expressions rationnelles à multiplicités sont souvent (cf. [41] ou [14] par exemple) définies par :

$$\mathsf{E} \to \{a \mid a \in A\} \mid 0 \mid 1 \mid k \in \mathbb{K} \mid (\mathsf{E} + \mathsf{E}) \mid (\mathsf{E} \cdot \mathsf{E}) \mid (\mathsf{E}^*) \ .$$

C'est-à-dire que les multiplicités sont considérées directement comme des atomes et les multiplications scalaires à gauche et à droite sont alors confondues avec la concaténation. Nous ne prenons pas ce formalisme car dans nos constructions les scalaires et les lettres ne jouent pas le même rôle et les multiplications à droite et à gauche ne sont pas traités comme la concaténation.

Le parenthésage des expressions rationnelles pourra être implicite lorsqu'il n'y a pas de confusion (par exemple les multiplicités à gauche et à droite sont appliquées à la plus petite expression à laquelle elles sont accolées –. De plus, les concaténations à droite consécutives pourront être écrites sans parenthèses.

Nous appelons $\mathbb{K}$-*expression unitaire* (à gauche et à droite) une expression rationnelle qui n'est ni $(k\mathsf{E})$, ni $(\mathsf{E}k)$ – donc soit une somme, soit une concaténation, soit une étoile, soit un atome –.

Exemple 36. L'expression rationnelle $3(a{\cdot}b)$ n'est pas unitaire. En revanche, l'expression $(3a \cdot b)$ l'est (avec le parenthésage explicite, la multiplicité 3 s'applique à a et non à $a \cdot b$).

Comme dans le cas booléen, nous définissons des identités dites *triviales* modulo lesquelles les $\mathbb{K}$-expressions rationnelles sont calculées. Ces identités sont les identités **(T)** du cas booléen auxquelles on ajoute des identités pour les multiplications à gauche et à droite par un scalaire :

$$\begin{gathered}(0_{\mathbb{K}}\mathsf{E}) \equiv (\mathsf{E}0_{\mathbb{K}}) \equiv 0 \ , \quad (1_{\mathbb{K}}\mathsf{E}) \equiv (\mathsf{E}1_{\mathbb{K}}) \equiv \mathsf{E} \\ (1k) \equiv (k1) \ , \quad ((k1) \cdot \mathsf{E}) \equiv (k\mathsf{E}) \ , \quad (\mathsf{E} \cdot (k1)) \equiv (\mathsf{E}k) \ , \\ (k(k'\mathsf{E})) \equiv (kk'\mathsf{E}) \ , \quad ((k\mathsf{E})k') \equiv (k(\mathsf{E}k')) \ , \quad ((\mathsf{E}k)k') \equiv (\mathsf{E}kk') \ .\end{gathered}$$

La considération de ces identités permet à nouveau d'éviter d'avoir 0 comme sous-expression d'une expression non nulle. En outre, les trois dernières identités permettent de représenter toute $\mathbb{K}$-expression rationnelle par un triplet (k, E, k') où k et k' sont des éléments de $\mathbb{K}$ et E est une expression unitaire.

Exemple 37. L'expression $2(a \cdot 31_{A^*})3$ se réduit par identités triviales en $2(a9)$.

Remarque 10. Il est courant d'utiliser également comme identité triviale la commutativité des multiplicatition par un scalaire à gauche et à droite : $(ak) \equiv (ka)$. Nous ne la retenons pas car les définitions que nous donnerons des termes dérivés ne commutent pas à cette identité.

Il est également possible de définir une profondeur sur les $\mathbb{K}$-expressions afin de réaliser des raisonnements inductifs sur celles-ci. La *profondeur* d'une expression rationnelle à multiplicités dans $\mathbb{K}$ est définie comme celle d'une expression booléenne, augmenté de :

$$\mathsf{d}(\mathsf{E}k) = \mathsf{d}(k\mathsf{E}) = \mathsf{d}(\mathsf{E}) + 1 \ .$$

Rappelons que l'addition sur le semi-anneau $\mathbb{K}$ est notée $\oplus$, la multiplication implicitement et que les neutres sont $0_{\mathbb{K}}$ et $1_{\mathbb{K}}$.

Définition 19. *Le* terme constant *d'une expression rationnelle est un élément de* $\mathbb{K}$ *défini par :*

$$\begin{gathered}
\mathsf{c}(0) = 0_{\mathbb{K}}, \qquad \mathsf{c}(1) = 1_{\mathbb{K}}, \qquad \forall a \in A \;\; \mathsf{c}(a) = 0_{\mathbb{K}} \;\; , \\
\mathsf{c}((k\mathsf{E})) = k\mathsf{c}(\mathsf{E}), \qquad \mathsf{c}((\mathsf{E}k)) = \mathsf{c}(\mathsf{E})k \;\; , \\
\mathsf{c}((\mathsf{F} + \mathsf{G})) = \mathsf{c}(\mathsf{F}) \oplus \mathsf{c}(\mathsf{G}), \qquad \mathsf{c}((\mathsf{F} \cdot \mathsf{G})) = \mathsf{c}(\mathsf{F})\mathsf{c}(\mathsf{G}) \;\; , \\
\mathsf{c}((\mathsf{E}^*)) = \mathsf{c}(\mathsf{E})^* \textit{ si le membre de droite est défini.} \;\; .
\end{gathered}$$

On dit qu'une expression rationnelle est *valide* si son terme constant est défini – et alors toutes ses sous-expressions sont valides –.

Les expressions rationnelles à multiplicités dans $\mathbb{K}$ sont une manière de représenter les séries rationnelles à coefficients dans $\mathbb{K}$. La série *dénotée* par l'expression rationnelle E est définie par :

$$\begin{gathered}
|0| = 0 \;\; , |1| = 1_{A^*} \;\; , |a| = a \;\; , \\
|(k\mathsf{E})| = k|\mathsf{E}| \;\; , |(\mathsf{E}k)| = |\mathsf{E}|k \;\; , \\
|(\mathsf{E} + \mathsf{F})| = |\mathsf{E}| \oplus |\mathsf{F}| \;\; , \\
|(\mathsf{E} \cdot \mathsf{F})| = |\mathsf{E}||\mathsf{F}| \;\; , \\
|(\mathsf{E}^*)| = |\mathsf{E}|^* \;\; .
\end{gathered}$$

Comme dans le cas booléen, où le terme constant exprime l'appartenance du mot vide au langage reconnu par une expression rationnelle, le terme constant d'une expression rationnelle à multiplicité est égal à la multiplicité du mot vide 1_{A^*} dans la série rationnelle qu'elle dénote.

Il faut bien faire la différence entre le monôme $k\mathsf{E}$ – que nous pourrons également noter $[k\mathsf{E}]$ pour éviter toute confusion – et l'expression rationnelle $(k\mathsf{E})$. Cette distinction est d'autant plus importante par la suite lorsque les combinaisons linéaires d'expressions sont définies.

L'ensemble des séries rationnelles sur A^* a coefficients dans $\mathbb{K}$ est la clôture rationnelle de l'ensemble des polynômes sur A^* à coefficients dans $\mathbb{K}$.

Proposition 42. *Une série formelle est* rationnelle *si, et seulement si, elle est dénotée par une* $\mathbb{K}$*-expression rationnelle valide.*

Le théorème suivant, relie, comme le théorème de Kleene dans le cas booléen, les séries rationnelles avec les séries reconnaissable (*cf.* [43, Th. III.3.5]).

Théorème 43 (Kleene-Schützenberger)**.** *Une série de* $\mathbb{K}\langle\langle A^* \rangle\rangle$ *est rationnelle si, et seulement si, elle est le comportement d'un* $\mathbb{K}$*-automate fini sur* A^**.*

Combinaisons linéaires de $\mathbb{K}$-expressions

Par la suite, pour définir les termes dérivés et les termes dérivés cassés, nous avons besoin d'introduire les combinaisons linéaires à gauche d'expressions rationnelles. Ces combinaisons forment un $\mathbb{K}$-module, c'est-à-dire que l'addition est commutative et la multiplication à gauche par un scalaire de $\mathbb{K}$ est distributive :

$$k\mathsf{E} \oplus k'\mathsf{F} = k'\mathsf{F} \oplus k\mathsf{E}, \qquad k\mathsf{E} \oplus k'\mathsf{E} = [k \oplus k']\mathsf{E} \ .$$

Nous ajoutons également à ce module une multiplication à droite par un scalaire et une concaténation à droite par une expression :

$$\begin{aligned} ([k\mathsf{E}] \cdot \mathsf{F}) &\equiv k(\mathsf{E} \cdot \mathsf{F}), \qquad ([k\mathsf{E}]k') \equiv k(\mathsf{E}k') \ , \\ ([\mathsf{E} \oplus \mathsf{E}'] \cdot \mathsf{F}) &\equiv (\mathsf{E} \cdot \mathsf{F}) \oplus (\mathsf{E}' \cdot \mathsf{F}), \qquad ([\mathsf{E} \oplus \mathsf{E}']k) \equiv (\mathsf{E}k) \oplus (\mathsf{E}'k) \ . \end{aligned}$$

Nous ne définissons pas une multiplication sur l'ensemble des combinaisons linéaires car, comme la concaténation des expressions rationnelles n'est pas associative, cette multiplication ne serait alors pas associative non plus. L'ensemble des combinaisons linéaires à gauche ne forme donc pas une semi-algèbre.

Arbre syntaxique

Comme nous l'avons déjà fait remarquer, toute $\mathbb{K}$ expression rationnelle peut être représentée par un triplet (k, E, k'), avec E unitaire. Afin de rester proche du modèle booléen, nous choisissons ces représentations en triplets pour *l'arbre syntaxique* d'une $\mathbb{K}$-expression. C'est-à-dire que les nœuds de l'arbre sont des triplets (k, u, k') où u représente un sous-arbre dont la racine est une somme, une concaténation, une étoile ou un atome – *i.e.* un sous-arbre représentant une expression unitaire –. Un tel u est appelé le u-nœud – pour nœud unitaire – du nœud (k, u, k'). Il faut noter que si nous avons choisi cette représentation de l'arbre syntaxique d'une expression pour sa facilité d'écriture et afin de repérer facilement les expressions unitaires, elle a le désavantage de déconnecter la profondeur de l'expression définie précédemment avec celle de l'arbre. La profondeur de l'expression serait celle de l'arbre si nous avions choisi de représenter les multiplications par des scalaires par des nœuds sur l'arbre.

Les *feuilles propres* de l'arbre syntaxique T_{E} d'une expression E sont les u-nœuds de T_{E} étiquetés par une lettre de A. Leur ensemble est toujours noté $\mathrm{fp}(\mathsf{E})$.

Notons $[\![(k, u, k')]\!]$ l'expression rationnelle représentée par le sous-arbre dont la racine est le nœud $x = (k, u, k')$. Notons dans la continuité $[\![u]\!]$ l'expression rationnelle unitaire représentée par u. Ainsi, on a :

$$[\![(k, u, k')]\!] = (k([\![u]\!]k')) \ .$$

Exemple 38. Posons $\mathsf{F}_1 = (2a3 \cdot b)$ et l'expression rationnelle :

$$\mathsf{E}_1 = 3\,((2\mathsf{F}_1 2) + (\mathsf{F}_1^* 2))\,3 \ .$$

L'arbre syntaxique de E_1 est montré sur la Figure 5.1. Nous noterons pour la suite u_i le u-nœud associé au nœud x_i. Par exemple u_3 est étiqueté par $*$ et l'expression associée est $[\![u_3]\!] = \mathsf{F}_1^*$.

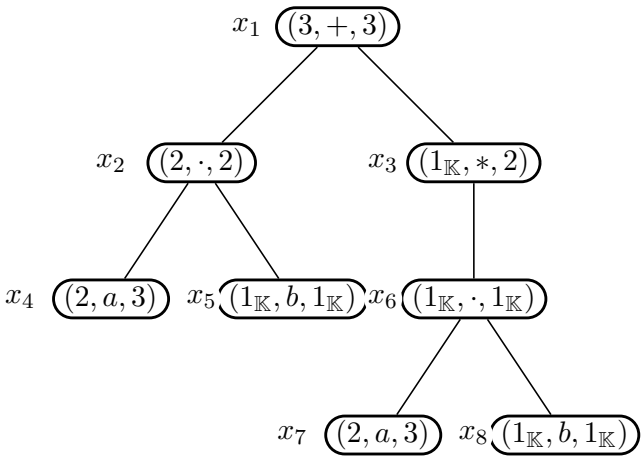


FIGURE 5.1 – L'arbre syntaxique de E_1

Un *u-nœud pondéré* est un triplet (k, u, k') où k et k' sont des éléments de $\mathbb{K}$. Il est important de noter que si un nœud x de l'arbre est bien un u-nœud pondéré, la réciproque n'est pas vraie. C'est-à-dire que nous nous autorisons à donner des coefficients aux u-nœuds qui n'apparaissent pas nécessairement dans l'arbre syntaxique de l'expression – seule l'expression unitaire doit apparaître dans l'arbre –.

Sur les u-nœuds pondérés il est possible d'étendre la définition d'expression rationnelle associée et de définir un terme constant naturellement :

(i) $[\![(k, u, k')]\!] = (k([\![u]\!]k'))$,

(ii) $\mathsf{c}((k, u, k')) = \mathsf{c}([\![(k, u, k')]\!])$.

Exemple 39 (*Ex. 38 cont.*)**.** Par exemple le u-nœud pondéré $(2, u_6, 2)$ n'est pas un nœud de l'arbre $\mathrm{T}_{\mathsf{E}_1}$. Il représente pourtant la même expression que le nœud x_2 : $[\![(2, u_6, 2)]\!] = 2\mathsf{F}_1 2$.

Nous étendons également la notion de premières feuilles d'un nœud – ou d'une expression – pour les expressions à multiplicités :

Définition 20. *La fonction* First *associe à tout u-nœud pondéré de l'arbre syntaxique* T_{E} *de l'expression* E *une combinaison linéaire à gauche de feuilles propres définie inductivement par :*

$$\begin{aligned}
\mathsf{First}(0) = \mathsf{First}(1) &= 0_{\mathbb{K}} \ , \\
\mathsf{First}((k, u, k')) &= \begin{cases} ku & \text{si } u \in \mathrm{fp}(\mathsf{E}) \\ k\mathsf{First}(u) & \text{sinon.} \end{cases} \ , \\
\mathsf{First}(x + y) &= \mathsf{First}(x) \oplus \mathsf{First}(y) \ , \\
\mathsf{First}(x \cdot y) &= \mathsf{First}(x) \oplus \mathsf{c}(x)\mathsf{First}(y) \ , \\
\mathsf{First}(x^*) &= \mathsf{c}(x)^*\mathsf{First}(x) \ ,
\end{aligned}$$

où (k, u, k')*,* x *et* y *sont des u-nœuds pondérés.*

Exemple 40 (*Ex. 38 cont.*)**.** On a par exemple :

$$\mathsf{First}(x_1) = 12u_4 \oplus 6u_7 \ .$$

Comme nous le verrons par la suite, les termes dérivés et les termes dérivés cassés d'expressions à multiplicités sont des concaténations à droite pondérées de sous-expressions unitaires. C'est pourquoi nous considérons les mots dont les lettres sont des u-nœuds pondérés. Le mot ε représente le mot vide sur cet alphabet. Sur ces mots, à l'instar des mots de nœuds booléens, nous étendons les fonctions terme constant et First :

(i) $\mathsf{c}(x_1 :: x_2 :: \ldots :: x_n) = \mathsf{c}(x_1)\mathsf{c}(x_2)\ldots\mathsf{c}(x_n)$,
(ii) $\mathsf{First}(\varepsilon) = 0_{\mathbb{K}}$,
(iii) $\mathsf{First}(x_1 :: \ldots :: x_n) = \mathsf{First}(x_1 :: \ldots :: x_{n-1}) \oplus \mathsf{c}(x_1 :: \ldots :: x_{n-1})\mathsf{First}(x_n)$.

Pour simplifier les écritures, nous nous autorisons à simplifier la notation des u-nœuds pondérés dont l'expression unitaire est 1. Ces expressions apparaîtront beaucoup dans les mots que nous calculerons et, par le jeu des expressions triviales, représentent sur l'expression dénotée par un mot, non pas une concaténation, mais une multiplication à droite par un scalaire. Ainsi, nous simplifions le u-nœud pondéré $(k', 1, k'')$ en k, avec $k = k'k''$. Un tel u-nœud pondéré sera naturellement appelé une multiplicité, ou *poids*.

Exemple 41 (*Ex. 38 cont.*)**.** Par exemple on a :

$$\begin{aligned}
[\![3 :: x_5 :: 2 :: 3]\!] &= (3(b6)) \\
[\![3 :: x_8 :: (1_{\mathbb{K}}, x_3, 1_{\mathbb{K}}) :: 2 :: 3]\!] &= ((3b) \cdot \mathsf{F}_1^*)6 \\
[\![2 :: 3]\!] &= (61_{A^*}) \\
[\![x_3 :: 2 :: 3]\!] &= \mathsf{F}_1^* 6 \ .
\end{aligned}$$

Et on a :

$$\begin{aligned}
\mathsf{First}(3 :: x_5 :: 2 :: 3) &= 3u_5 \\
\mathsf{First}(3 :: x_8 :: (1_{\mathbb{K}}, x_3, 1_{\mathbb{K}}) :: 2 :: 3) &= 3u_8 \\
\mathsf{First}(2 :: 3) &= 0 \\
\mathsf{First}(x_3 :: 2 :: 3) &= 2u_7 \ .
\end{aligned}$$

5.1.2 Représentation canonique

Dans le cas booléen, les termes dérivés étaient représentés par des mots sur les nœuds car ils étaient des concaténations à droite de sous-expressions. Pour les expressions à multiplicités, nous allons montrer par la suite que les termes dérivés, et les termes dérivés cassés, sont des concaténations de sous-expressions unitaires pondérées. Il apparaît donc que les termes dérivés pourront être représentés par des mots sur les u-nœuds pondérés. Nous allons donc développer dans cette sous-section une méthode similaire à celle du cas booléen, avec des marques sur les nœuds et u-nœuds, qui permet de les identifier et d'avoir une représentation canonique d'une concaténation de sous-expressions unitaires pondérées par un mot sur les u-nœuds pondérés. Plusieurs particularités des multiplicités rendent la tâche plus complexe que dans le cas booléen.

Dans le cas booléen, il était possible de ne s'intéresser qu'aux mots propres car il suffisait 'd'effacer' les lettres 1 du mot sans que cela ne change l'expression associée. Ceci n'est pas aussi direct dans le cas à multiplicités car seules les lettres $(1_{\mathbb{K}}, 1, 1_{\mathbb{K}})$ peuvent être effacées. Les autres poids non égaux à l'unité doivent être traités différemment. En particulier il est nécessaire, avant de faire tout autre calcul, de transformer le mot afin de se 'conformer' aux identités triviales introduites par les multiplicités. La première opération est alors de regrouper les poids successifs.

Les marques

L'arbre syntaxique de l'expression E est 'marqué' de telle manière que chaque u-nœud u a une marque $\overline{u}$ telle que $\overline{u} = \overline{v}$ si $[\![u]\!] = [\![v]\!]$, c'est-à-dire que u et v sont racines d'arbres isomorphes. La définition de marque s'étend aux u-nœuds pondérés par :

$$\overline{(k, u, k')} = (k, \overline{u}, k') \quad .$$

Dans un souci de simplicité, nous continuons d'appeler *poids* les marques des poids (les u-nœuds pondérés dont l'expression unitaire est l'unité). Les mots sur les marques sont les mots dont les lettres sont des marques de u-nœuds pondérés.

Exemple 42 (*Ex. 38 cont.*)**.** Notons α_i la marque du u-nœud u_i. On a alors les égalités suivantes :

$$\alpha_4 = \alpha_7 \qquad \alpha_5 = \alpha_8$$

de plus comme les poids sont identiques, on a $\overline{x_4} = \overline{x_7}$ et $\overline{x_5} = \overline{x_8}$ et donc on a également :

$$\alpha_2 = \alpha_6 \quad .$$

Il est important de remarquer que l'on n'a néanmoins pas $\overline{x_2} = \overline{x_6}$ car les poids diffèrent.

Les identités triviales

Dans un premier temps, pour transformer un mot sur les marques en un mot canonique pour l'expression associée, nous allons le transformer de la même manière que les identités triviales permettent de transformer des expressions. Il faut noter que, contrairement aux réductions vues dans le cas booléen – et dont l'adaptation aux multiplicités suit –, les réductions que nous effectuons ici sont totalement indépendantes de l'expression E.

Nous allons ainsi tout d'abord transformer un mot sur les marques afin qu'il soit cohérent avec les identités :

$$((\mathsf{E}k)k') = (\mathsf{E}kk') \qquad \text{et} \quad (k(k'\mathsf{E})) = (kk'\mathsf{E}) \ .$$

Pour cela il suffit de remplacer tous poids consécutifs en leur produit :

$$\alpha_1 :: \ldots :: k :: k' :: \cdots \to^{\rho_1} \alpha_1 :: \ldots :: kk' :: \ldots \ .$$

Exemple 43 (*Ex. 38 cont.*)**.** Par exemple $3 :: \overline{x_5} :: 2 :: 3$ se réduit par ρ_1 en $3 :: \overline{x_5} :: 6$.

Dans un deuxième temps, nous rendons cohérent le mot pour l'identité :

$$((k1) \cdot (k'\mathsf{E})) = (kk'\mathsf{E}) \ .$$

Pour cela il faut rentrer le coefficient dans le premier terme de la concaténation à droite :

$$k :: (k_1, \overline{u_1}, k_1') :: \cdots \to^{\rho_2} (kk_1, \overline{u_1}, k_1') :: \ldots \ .$$

Exemple 44 (*Ex. 38 cont.*)**.** Par exemple $3 :: \overline{x_5} :: 6$ se réduit par ρ_2 en $(3, \alpha_5, 1_{\mathbb{K}}) :: 6$.

Finalement, nous transformons l'expression pour la rendre cohérente avec :

$$((\mathsf{E}k) \cdot (k'1) = (\mathsf{E}kk')$$

Comme les mots sur les marques représentent des concaténations à droite, le coefficient en fin de mot ne doit être rentré dans la marque précédente que dans le cas où celle-ci est également la première marque du mot :

$$(k_1, \overline{u_1}, k_1') :: k \to^{\rho_3} (k_1, \overline{u_1}, k_1'k) \ .$$

Exemple 45 (*Ex. 38 cont.*)**.** Par exemple $(3, \alpha_5, 1_{\mathbb{K}}) :: 6$ se réduit par ρ_3 en $(3, \alpha_5, 6)$. En revanche, $(3, \alpha_5, 1_{\mathbb{K}}) :: 6 :: (2, \alpha_1, 3)$ ne se réduit pas par ρ_3.

Après avoir réalisé successivement les transformations ρ_1, ρ_2 et ρ_3, un mot de marque $\overline{X}$ quelconque est donc remplacé par un mot de marque $\overline{Y}i = \rho_3(\rho_2(\rho_1(\overline{X})))$ tel que :

(i) il n'y a pas deux poids consécutifs dans $\overline{Y}$;
(ii) $\overline{Y}$ ne commence pas par un poids ;
(iii) si $\overline{Y}$ est de longueur 2 alors ses lettres ne sont pas des poids ;
(iv) $\overline{X}$ et $\overline{Y}$ représentent la même expression.

Réduction

Lorsque le mot a été transformé pour être cohérent avec les identités triviales, nous allons réaliser les réductions qui permettent d'avoir une représentation canonique des concaténations de sous-expressions. Contrairement aux transformations précédentes, les règles de réductions présentées maintenant sont dépendantes de l'expression rationnelle, ou plus particulièrement de son arbre syntaxique, que l'on considère.

Soit $\mathcal{R}$ l'ensemble des règles de réécriture sur les mots de marques :

$$(k_1, \alpha_1, k_1') :: (k_2, \alpha_2, k_2') \to (1_{\mathbb{K}}, \overline{u}, 1_{\mathbb{K}})$$

avec α_1, α_2 et u tels qu'il existe $v = x_1 \cdot x_2$ où $x_1 = (k_1, u_1, k_1')$ et $x_2 = (k_2, u_2, k_2')$ et avec $\overline{u_1} = \alpha_1$, $\overline{u_2} = \alpha_2$ et $\overline{v} = \overline{u}$. C'est-à-dire des nœuds dont les marques sont respectivement α_1 et α_2 et qui sont fils gauche et droit d'un u-nœud étiqueté par une concaténation et dont la marque est $\overline{u}$.

Exemple 46 (*Ex. 38 cont.*)**.** Pour E_1 il y a une seule règle dans $\mathcal{R}$ car les deux concaténations amènent la même règle sur les marques :

$$(2, \alpha_4, 3) :: (1_{\mathbb{K}}, \alpha_5, 1_{\mathbb{K}}) \to (1_{\mathbb{K}}, \alpha_2, 1_{\mathbb{K}}) \ .$$

Notons ρ_4 la réduction préfixe par l'ensemble des règles $\mathcal{R}$. La réduction complète est notée ρ : pour tout mot sur les marques $\overline{X}$, on a $\rho\overline{X} = \rho_3(\rho_4(\rho_3(\rho_2(\rho_1(\overline{X})))))$. Le dernière occurrence de ρ_3 vient du fait que la réduction préfixe ρ_4 réduit la taille du mot et il est possible qu'il devienne de longueur 2 et il faut alors à nouveau vérifier si la deuxième marque est un poids. Avec cet ajout, il apparaît que la réécriture ρ est idempotente. En outre toutes les transformations ont été choisies dans le souci de conserver l'expression rationnelle dénotée :

$$[\![\rho\overline{X}]\!] = [\![\overline{X}]\!] \ .$$

La réduction ρ permet l'écriture de manière canonique de concaténations à droite de sous-expressions unitaires pondérées comme le prouve le théorème suivant :

Théorème 44. *Soient $\overline{X}$ et $\overline{Y}$ deux mots sur les marques. On a :*

$$[\![\overline{X}]\!] = [\![\overline{Y}]\!] \Leftrightarrow \rho\left(\overline{X}\right) = \rho\left(\overline{Y}\right) \ .$$

Démonstration. Comme la réduction est idempotente, nous allons montrer que si $\overline{X}$ et $\overline{Y}$ sont réduits par ρ, alors ils sont égaux si, et seulement si,

ils dénotent la même expression. La réciproque est triviale, montrons donc que s'ils dénotent la même expression, alors ils sont égaux. Soient $\overline{X} = \overline{x_1} :: \ldots :: \overline{x_n}$ et $\overline{Y} = \overline{y_1} :: \ldots :: \overline{y_m}$ deux mots réduits sur les marques, avec $x_i = (k_i, u_i, k'_i)$ et $y_i = (h_i, v_i, h'_i)$.

Par induction sur n et sur m : **Cas** $n = 1$:

$m = 1$: $\|\overline{X}\| = \|\overline{Y}\|$ si, et seulement si, $\|\overline{x_1}\| = \|\overline{y_1}\|$, c'est-à-dire $\|u_1\| = \|v_1\|$, $k_1 = h_1$ et $k'_1 = h'_1$, soit $\overline{X} = \overline{Y}$.

$m = 2$: comme $\overline{Y}$ est réduit, ni y_1, ni y_2 ne sont des poids. Dans ce cas $\|\overline{X}\| = \|\overline{Y}\|$ si, et seulement si, $\|\overline{x_1}\| = \|\overline{y_1}\| \cdot \|\overline{y_2}\|$. Ce qui implique $k_1 = k'_1 = 1_{\mathbb{K}}$ et il existe x tel que $\overline{x} = \overline{x_1}$ et $x = y_1 \cdot y_2$. Ce cas est impossible puisque $\overline{Y}$ a été réduit par ρ_4.

$m > 2$: Ce cas se réduit inductivement au cas précédent qui est impossible.

Cas $n > 1$ **et** $m > 1$: par identification sur l'expression rationnelle $\|\overline{X}\|$, on a $\|\overline{X}\| = \|\overline{Y}\|$ seulement si $\overline{x_n} = \overline{y_m}$ et l'on peut donc revenir au premier cas. □

5.2 Termes dérivés

Dans cette section nous rappelons la définition des termes dérivés et de la dérivation qui leur est associée dans le cas des expressions rationnelles à multiplicités. Cette définition a été donnée dans [31]. Par rapport au cas booléen, un certain nombre de choix sont à faire si l'on souhaite étendre la définition des termes dérivés, en particulier en ce qui concerne le traitement des expressions avec des poids à gauche et à droite. La définition qui a été retenue l'a été afin d'avoir, comme dans le cas booléen, l'automate des termes dérivés quotient de l'automate des positions. Nous verrons entre autre que cette condition amène à considérer différemment les multiplications à gauche et à droite, ce qui justifie notre choix de ne pas construire les expressions modulo la commutativité des multiplicités sur les atomes.

5.2.1 Définition

Définition 21 ([31]). *Soit* E *une expression* $\mathbb{K}$*-rationnelle et* a *une lettre de* A*. La* $\mathbb{K}$-derivation *de* E *par rapport à* a*, notée* $\dfrac{\partial}{\partial a}\mathsf{E}$*, est la combinaison linéaire à gauche d'expressions rationnelles à coefficients dans* $\mathbb{K}$*, définie par*

les formules suivantes :

$$\frac{\partial}{\partial a}\,0 = \frac{\partial}{\partial a}\,1 = 0\,, \quad \frac{\partial}{\partial a}\,b = \begin{cases} 1 & si \quad b = a \\ 0 & sinon \end{cases}\,,$$

$$\frac{\partial}{\partial a}(k\,\mathsf{E}) = k\,\frac{\partial}{\partial a}\,\mathsf{E}\,, \quad \frac{\partial}{\partial a}(\mathsf{E}\,k) = \left(\left[\frac{\partial}{\partial a}\,\mathsf{E}\right]\,k\right)\,,$$

$$\frac{\partial}{\partial a}(\mathsf{E}+\mathsf{F}) = \frac{\partial}{\partial a}\,\mathsf{E} \oplus \frac{\partial}{\partial a}\,\mathsf{F}\,, \tag{5.1}$$

$$\frac{\partial}{\partial a}(\mathsf{E}\cdot\mathsf{F}) = \left(\left[\frac{\partial}{\partial a}\,\mathsf{E}\right]\cdot\mathsf{F}\right) \oplus \mathsf{c}(\mathsf{E})\,\frac{\partial}{\partial a}\,\mathsf{F}\,, \tag{5.2}$$

$$\frac{\partial}{\partial a}(\mathsf{E}^*) = \mathsf{c}(\mathsf{E})^*\,\left(\left[\frac{\partial}{\partial a}\,\mathsf{E}\right]\cdot(\mathsf{E}^*)\right)\,. \tag{5.3}$$

La dérivation d'une combinaison linéaire est définie par linéarité :

$$\frac{\partial}{\partial a}\left(\bigoplus_{i\in I} k_i\,\mathsf{E}_i\right) = \bigoplus_{i\in I} k_i\,\frac{\partial}{\partial a}\,\mathsf{E}_i\,. \tag{5.4}$$

Exemple 47 (*Ex. 38 cont.*)**.** La dérivation de E_1 par rapport à a donne :

$$\begin{aligned}
\frac{\partial}{\partial a}\,\mathsf{F}_1 &= 2(\frac{\partial}{\partial a}(a3)\cdot b) \\
&= 2((1_{A^*}3)\cdot b) = [2(3b)] \\
\frac{\partial}{\partial a}\,\mathsf{E}_1 &= 3\left(\frac{\partial}{\partial a}((2\mathsf{F}_1 2) + (\mathsf{F}_1^* 2))\right)3 \\
&= 3\left(2(\frac{\partial}{\partial a}\,\mathsf{F}_1 2) \oplus ((\frac{\partial}{\partial a}\,\mathsf{F}_1\cdot\mathsf{F}_1^*)2\right)3 \\
&= [12(3b2)] \oplus [6(((3b)\cdot\mathsf{F}_1^*)6)]\,.
\end{aligned}$$

Nous rappelons que $[12(3b2)]$ est le monôme de coefficient 12 et d'expression rationnelle $(3b2)$, ce qui est différent de $[36(b2)]$ qui est le monôme de coefficient 36 de l'expression $(b2)$.

Cette dérivation par rapport à une lettre permet une dérivation par rapport à un mot :

Définition 22. *La dérivation de* E *par rapport à un mot non vide* ua *de* A^* *est définie par induction par :*

$$\frac{\partial}{\partial ua}\,\mathsf{E} = \frac{\partial}{\partial a}\left(\frac{\partial}{\partial u}\,\mathsf{E}\right)\,.$$

Exemple 48 (*Ex. 38 cont.*)**.** La dérivation de E_1 par rapport à ab et aba donnent :

$$\begin{aligned}\frac{\partial}{\partial ab}\mathsf{E}_1 &= \frac{\partial}{\partial b}12(3b2)\oplus 6(((3b)\cdot\mathsf{F}_1)6)\\ &= 36(21_{A^*})\oplus 18(\mathsf{F}_1 6)\\ \frac{\partial}{\partial aba}\mathsf{E}_1 &= \frac{\partial}{\partial a}36(21_{A^*})\oplus 18(\mathsf{F}_1 6)\\ &= 36((3b)6)\ .\end{aligned}$$

Contrairement au cas booléen, la définition de termes dérivés à partir des termes qui apparaissent dans les dérivations par rapport à un mot et la définition inductive de l'ensemble des termes dérivés ne donnent pas le même résultat. Dans le but de garder, dans le cas à multiplicités, l'automate des termes dérivés comme quotient de l'automate des positions de l'expression, nous prenons la définition inductive des termes dérivés :

Définition 23 ([31])**.** *L'ensemble* $\mathrm{TD}(\mathsf{E})$ *des* vrais termes dérivés *de l'expression* E *est défini inductivement par les règles suivantes :*

$$\begin{aligned}\mathrm{TD}(\mathsf{0})=\mathrm{TD}(\mathsf{1})=\emptyset\ ,\quad \forall a\in A,\ \mathrm{TD}(a)=\{\mathsf{1}\}\ ,\\ \forall k\in\mathbb{K},\ \mathrm{TD}(k\,\mathsf{E})=\mathrm{TD}(\mathsf{E})\ ,\quad \mathrm{TD}(\mathsf{E}\,k)=(\mathrm{TD}(\mathsf{E})\ k)\ ,\\ \mathrm{TD}(\mathsf{E}+\mathsf{F})=\mathrm{TD}(\mathsf{E})\ \cup\ \mathrm{TD}(\mathsf{F})\ ,\\ \mathrm{TD}(\mathsf{E}\cdot\mathsf{F})=(\mathrm{TD}(\mathsf{E})\cdot\mathsf{F})\ \cup\ \mathrm{TD}(\mathsf{F})\ ,\\ \mathrm{TD}(\mathsf{E}^*)=(\mathrm{TD}(\mathsf{E})\cdot(\mathsf{E}^*))\ .\end{aligned}$$

L'expression E n'est pas nécéssairement dans $\mathrm{TD}(\mathsf{E})$ et l'ensemble des *termes dérivés* de E est : $\mathrm{D}(\mathsf{E})=\mathrm{TD}(\mathsf{E})\cup\{\mathsf{E}\}$.

Exemple 49 (*Ex. 38 cont.*)**.** Les vrais termes dérivés de E_1 sont :

$$\begin{aligned}\mathrm{TD}(\mathsf{F}_1)&=\mathrm{TD}((a3)\cdot b)\\ &=\{(1_{A^*}3)\cdot b\}\cup\mathrm{TD}(b)=\{(3b),1_{A^*}\}\\ \mathrm{TD}(\mathsf{E}_1)&\\ &=\mathrm{TD}((2\mathsf{F}_1 2)+(\mathsf{F}_1^*2))\,3\\ &=(\mathrm{TD}(\mathsf{F}_1 2)\,3)\ \cup\ (\mathrm{TD}(\mathsf{F}_1^*2)\,3)\\ &=\{3b6,6,((3b)\cdot\mathsf{F}_1^*)6,(\mathsf{F}_1^*6)\}\ .\end{aligned}$$

La définition inductive des vrais termes dérivés – et celle des termes dérivés qui en découle – permet d'énoncer la proposition suivante :

Proposition 45. *Les termes dérivés d'une expressions rationnelle* E *sont des concaténations à droite de sous-expressions unitaires de* E *pondérées.*

Une induction sur la profondeur de l'expression permet de relier la dérivation avec l'ensemble des termes dérivés :

Théorème 46 ([31])**.** *Soit* $\mathrm{D}(\mathsf{E}) = \{\mathsf{K}_1, ..., \mathsf{K}_n\}$ *l'ensemble des termes dérivés de* E*. Pour toute lettre* a *de* A*, il existe une matrice carrée* $a\mu$*, de dimension* n *et à coefficients dans* $\mathbb{K}$ *telle que :*

$$\forall i \leqslant n \qquad \frac{\partial}{\partial a}\mathsf{K}_i = \bigoplus_j a\mu_{i,j}\,\mathsf{K}_j\ .$$

Définition 24. L'automate des termes dérivés *d'une* $\mathbb{K}$*-expression* E *est l'automate* $\mathcal{D}(\mathsf{E})$ *dont les états sont les termes dérivés.*
(i) *La fonction initiale vaut* 1 *pour* E *et* 0 *pour tous les autres termes dérives.*
(ii) *La fonction finale est le terme constant du terme dérivé.*
(iii) *Il y a une transition de* K_i *à* K_j *étiqueté par* ka *si* $k = a\mu_{i,j} \neq 0_{\mathbb{K}}$*.*

Exemple 50 (*Ex. 38 cont.*)**.** La Figure 5.2 représente l'automate des termes dérivés de E_1.

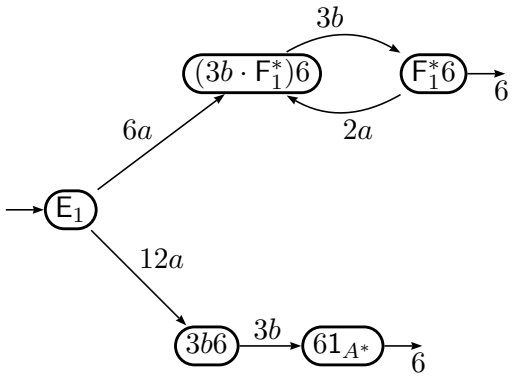


FIGURE 5.2 – L'automate $\mathcal{D}(\mathsf{E}_1)$

L'automate $\mathcal{D}(\mathsf{E})$ reconnaît la série dénotée par E et est un quotient de l'automate des positions [31]. C'est ce résultat qui a motivé deux choix dans nos définitions qui sont explicités dans les deux remarques suivantes :

Remarque 11 ([31])**.** Nous avons choisi comme définition pour les termes dérivés avec multiplicités la définition inductive. Ce n'est pas, comme la généralisation du cas booléen pourrait faire penser, l'ensemble des expressions qui apparaissent avec un coefficient non nul dans une dérivation par rapport à un mot non vide.

Ceci s'explique par la possibilité qu'un terme apparaissant lors d'une dérivation par rapport à une lettre d'un terme dérivé n'apparaisse pas dans la dérivation par rapport à un mot non-vide car il peut exister plusieurs chemins dont les coefficients s'annulent. Il est même possible que la dérivation par rapport à une lettre – et non un mot – ne fasse pas apparaître certains termes dérivés.

Par exemple, si l'on prend l'expression rationnelle à coefficients dans $\mathbb{Z}$: $a + (-1a)$ alors la dérivation par rapport à a donne :

$$\frac{\partial}{\partial a}\, a + (-1)a = 1 \oplus -1[1] = 0 \ .$$

Les dérivations par rapport à tout mot non vide donnent le même résultat. Si l'on prenait une définition des termes dérivés basés sur la dérivation, l'expression $a+(-1a)$ aurait pour automate de termes dérivés un automate avec un unique état initial. Ce n'est pas le quotient de l'automate des positions qui a un état initial et un état final (sans transition).

La définition des termes dérivés inductive donne $\mathrm{TD}\,(a + (-1a)) = \{1\}$, ce qui correspond donc a un automate des termes dérivés à deux états, un état initial, un état final et aucune transition. Cet automate est bien un quotient de l'automate des positions, qui a trois états, un initial, deux finaux, et des transitions étiquetées respectivement a et$-a$ de l'état initial aux deux états finaux (*cf.*Figure 5.4).

FIGURE 5.3 – L'automate $\mathcal{D}(a + (-1)a)$

Remarque 12. De la même manière, le choix de ne pas calculer les expressions modulo l'identité $(ak) \equiv (ka)$ se justifie par le fait que l'on veut que l'automate des termes dérivés soit un quotient de l'automate des positions.

En effet, l'expression rationnelle $((c \cdot b)k)$ est telle qu'avec cette identité, l'automate des termes dérivés construit ne serait pas le quotient de l'automate des positions. Dans ce cas c'est même le Théorème 46 qui ne s'applique

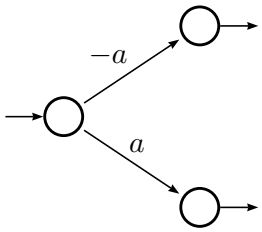


FIGURE 5.4 – L'automate des positions de $a + (-1)a$

plus, en effet :

$$\frac{\partial}{\partial c}(c \cdot b)k = (bk) \equiv (kb)$$
$$\frac{\partial}{\partial b}\, kb = k1_{A^*} \quad .$$

Pourtant la définition inductive donne :

$$\mathrm{TD}\,((c \cdot b)k) = \mathrm{TD}\,(c \cdot b)\,k = \{(bk), (1k)\} \equiv \{(kb), (k1)\}.$$

Le terme dérivé 1 n'apparaît pas dans la définition inductive, c'est $k1$ qui le remplace.

5.2.2 Calcul de l'automate

Chemins et termes dérivés

Afin de construire l'automate des termes dérivés d'une expression à multiplicités dans $\mathbb{K}$, nous allons d'abord montrer comment, comme dans le cas booléen, les termes dérivés peuvent être exprimés sous formes de mots sur les u-nœuds pondérés obtenus par un chemin dans l'arbre syntaxique.

Il est toujours possible de décorer l'arbre syntaxique à la manière de ce qui est fait dans le cas booléen. Seule une différence pour les liens : le lien qui part d'un fils gauche d'un nœud concaténation pointe sur son frère, comme dans le cas booléen, mais le lien qui part d'un fils d'un nœud étoilé pointe vers le u-nœud de son père. Cette modification se justifie par le fait que si deux u-nœuds sont fils d'une concaténation, alors l'expression qui est représentée est bien la concaténation des deux u-nœuds. Par contre, lorsqu'une sous-expression est étoilée, elle est unitaire et donc indépendante des poids sur le nœud père.

Les chemins qui permettent d'obtenir les termes dérivés sont les mêmes physiquement sur l'arbre que dans le cas booléen mais le mot qui leur est associé est différent car les poids droits sont récoltés également tout le long du trajet.

Formellement, *le chemin associé* au u-nœud pondéré $x = (k, u, k')$ est défini par le mot suivant :

$$\pi_r(x) = \begin{cases} \varepsilon & \text{si } x \text{ est la racine et si } k' = 1_{\mathbb{K}}, \\ k' & \text{si } x \text{ est la racine et } k' \neq 1_{\mathbb{K}}, \\ k' :: (1_{\mathbb{K}}, v, 1_{\mathbb{K}}) :: \pi_r(\phi(x)) & \text{si } v = x^*, \\ k' :: y :: \pi_r(\phi(x)) & \text{si } \phi(x) = x \cdot y, \\ k' :: \pi_r(\phi(x)) & \text{sinon.} \end{cases}$$

Lorsqu'il n'y a pas d'ambiguïté sur la racine, il est possible d'écrire $\pi(x)$. De même, si u est un u-nœud, il sera possible d'écrire $\pi(u)$ pour $\pi((1_{\mathbb{K}}, u, 1_{\mathbb{K}}))$.

Exemple 51 (*Ex. 38 cont.*)**.** On a par exemple :

$$\begin{aligned} \pi(u_4) &= 3 :: x_5 :: 2 :: 3 \ , \\ \pi(u_5) &= 2 :: 3 \ , \\ \pi(u_7) &= 3 :: x_8 :: (1_{\mathbb{K}}, u_3, 1_{\mathbb{K}}) :: 2 :: 3 \ , \\ \pi(u_8) &= (1_{\mathbb{K}}, u_3, 1_{\mathbb{K}}) :: 2 :: 3 \ . \end{aligned}$$

Comme dans le cas booléen, on se rend compte sur cet exemple, à travers des calculs déjà réalisés, que les expressions associées aux chemins des feuilles sont les termes dérivés.

Théorème 47. *Soit* E *une expression rationnelle,* T_{E} *son arbre syntaxique de racine* r*. Si* $\mathsf{First}(r, a) = \bigoplus_i k_i \ell_i$*, alors :*

$$\frac{\partial}{\partial a} \mathsf{E} = \bigoplus_i k_i [\![\pi(\ell_i)]\!] \ .$$

Démonstration. Dans cette preuve, nous allons calculer des chemins dans des sous arbres, nous allons donc préciser la racine de l'arbre dans lequel le chemin est calculé. Lorsque la racine sera omise, nous seront dans l'arbre complet, de racine r. La preuve de ce théorème est une induction sur la profondeur de E. Les cas de base où E est un atome sont triviaux. Supposons donc que $r = (k, u, k')$.

– Si $u = x + y$, c'est-à-dire que $\mathsf{E} = (k((\mathsf{F} + \mathsf{G})k'))$, alors :

$$\mathsf{First}(\mathsf{E}, a) = k\mathsf{First}(u, a) = k(\mathsf{First}(x, a) \oplus \mathsf{First}(y, a)) \ ,$$

Si $\mathsf{First}(x,a) = \bigoplus_j k_j \ell_j$, avec $k_j \neq 0_{\mathbb{K}}$, alors on a :

$$\pi_r(\ell_j) = \pi_x(\ell_j) :: k' \ ,$$

et donc $[\![\pi_r(\ell_j)]\!] = ([\![\pi_x(\ell_j)]\!]k')$.
De la même manière pour les feuilles de G : si $\mathsf{First}(y,a) = \bigoplus_h k_h \ell_h$ alors $[\![\pi_r(\ell_h)]\!] = ([\![\pi_y(\ell_h)]\!]k')$.
Finalement :

$$\begin{aligned}\bigoplus_i k_i [\![\pi(\ell_i)]\!] &= k((\bigoplus_j k_j [\![\pi(\ell_j)]\!]k') \oplus (\bigoplus_h k_h [\![\pi(\ell_h)]\!]k')) \\ &= k\left(\frac{\partial}{\partial a}(\mathsf{F}k' + \mathsf{G}k')\right) \\ &= \frac{\partial}{\partial a}\,\mathsf{E} \ .\end{aligned}$$

– Si $u = x \cdot y$, c'est-à-dire $\mathsf{E} = (k((\mathsf{F} \cdot \mathsf{G})k'))$, alors :

$$\mathsf{First}(\mathsf{E},a) = k\mathsf{First}(u,a) = k(\mathsf{First}(x,a) \oplus \mathsf{c}(x)\mathsf{First}(y,a)) \ .$$

Si $y = (h,v,h')$ et si $\mathsf{First}(x,a) = \bigoplus_j k_j \ell_j$, avec $k_j \neq 0_{\mathbb{K}}$, alors on a :

$$\pi_u(\ell_j) = \pi_x(\ell_j) :: (h,v,1_{\mathbb{K}}) :: h' :: k' \ ,$$

donc $[\![\pi_u(\ell_j)]\!] = (([\![\pi_x(\ell_j)]\!] \cdot \mathsf{G})k')$.
Si $\mathsf{First}(y,a) = \bigoplus_h k_h \ell_h$ alors $[\![\pi_r(l_h)]\!] = ([\![\pi_y(l_h)]\!]k')$ et donc :

$$\begin{aligned}\bigoplus_i k_i [\![\pi(\ell_i)]\!] &= k(\bigoplus_j k_j (([\![\pi(\ell_j)]\!] \cdot \mathsf{G})k') \bigoplus_h \mathsf{c}(s) k_h ([\![\pi(\ell_h)]\!]k')) \\ &= k\left(\frac{\partial}{\partial a}(\mathsf{F} \cdot \mathsf{G})k'\right) \\ &= \frac{\partial}{\partial a}\,\mathsf{E} \ .\end{aligned}$$

– Si $u = x^*$, c'est-à-dire $\mathsf{E} = (k((\mathsf{F}^*)k'))$, alors :

$$\mathsf{First}(\mathsf{E},a) = k\mathsf{c}(x)^*\mathsf{First}(x,a) \ .$$

Si $\mathsf{First}(x,a) = \bigoplus_j k_j \ell_j$, avec $k_j \neq 0_{\mathbb{K}}$, alors on a :

$$\pi_r(\ell_j) = \pi_u(\ell_j) :: (1_{\mathbb{K}}, x, 1_{\mathbb{K}}) :: k' \ .$$

donc $[\![\pi_r(\ell_j)]\!] = (([\![\pi_u(\ell_j)]\!] \cdot \mathsf{F}^*)k')$. et donc :

$$\bigoplus_i k_i [\![\pi(\ell_i)]\!] = k\mathsf{c}(x)^* \left((\frac{\partial}{\partial a}\,\mathsf{F} \cdot \mathsf{F}^*)k'\right) = \frac{\partial}{\partial a}\,\mathsf{E} \ .$$

□

Le théorème suivant, dont la preuve est similaire à celle du Théorème 36, permet de dériver des concaténations à droite de sous-expressions – unitaires pondérées – :

Théorème 48. *Soit* E *une expression rationnelle et* X *un mot sur les nœuds de* T_{E}. *Si* $\mathsf{First}(X, a) = \bigoplus_i k_i \ell_i$, *alors :*

$$\frac{\partial}{\partial a}[\![X]\!] = \bigoplus_i k_i [\![\pi(\ell_i)]\!] \ .$$

Pour une feuille propre h telle que $\mathsf{First}(\pi(h), a) = \bigoplus_i k_i \ell_i$, on a donc :

$$\frac{\partial}{\partial a}[\![\pi(h)]\!] = \bigoplus_i k_i [\![\pi(\ell_i)]\!] \ .$$

Et finalement, directement :

Théorème 49.

$$\mathrm{TD}\,(\mathsf{E}) = \{[\![\pi(\ell)]\!] \mid \ell \in \mathrm{fp}(\mathsf{E})\}$$

Construction de l'automate

Le théorème précédent, identique au cas booléen, permet, comme dans ce cas là, de considérer que les états $\mathcal{D}(\mathsf{E})$ ne sont plus les termes dérivés mais les mots réduits sur les marques qui les représentent. L'unique état initial est $\overline{r}$, la fonction finale est définie par le terme constant des mots de marques. Ainsi nous notons également $\theta(q)$ l'ensemble des feuilles propres ℓ de T_{E} dont la réduction du mot marqué associé au chemin est q. C'est-à-dire que si x et x' sont dans $\theta(q)$ alors : $\rho\left(\overline{\pi(x)}\right) = \rho\left(\overline{\pi(x')}\right) = q$. Un représentant x arbitrairement choisi dans $\theta(q)$ permet de définir $\mathsf{First}(q) = \mathsf{First}(\pi(x))$, justifié par :

Proposition 50. *Soit* q *un état, si* x *et* x' *sont dans* $\theta(q)$ *et si* $\mathsf{First}(\pi(x), a) = \bigoplus_i k_i \ell_i$ *et* $\mathsf{First}(\pi(x'), a) = \bigoplus_j k'_j \ell'_j$ *alors :*

$$\bigoplus_i k_i [\![\pi(\ell_i)]\!] = \bigoplus_j k'_j [\![\pi(\ell'_j)]\!] \ .$$

Enfin nous pouvons décrire de manière constructive l'ensemble des transitions de l'automate $\mathcal{D}(\mathsf{E})$: soient q_i et q_j deux états. Si $\mathsf{First}(\pi(q_i), a) = \bigoplus_m k_m \ell_m$, alors on a par le Théorème 48 :

$$a\mu_{i,j} = \sum_{\ell_n \in \theta(q_j)} k_n \ .$$

Cette équation permet de construire les transitions par le Théorème 46.

5.3 Termes dérivés cassés

Dans cette section, nous donnons finalement la définition des termes dérivés cassés pour le cas des expressions rationnelles à multiplicités. Une définition avait déjà été envisagé dans [31] mais ce n'est pas la définition que nous retenons et nous nous justifierons de ce choix. Nous achevons cette partie sur l'adaptation des constructions données précédemment afin de construire l'automate des termes dérivés cassés pour les expressions rationnelles à multiplicités.

5.3.1 Définition

Le cassage des expressions booléennes peut être interprété comme une dérivation par le mot vide. Il en résulte que le cassage d'une expression rationnelle booléenne donne un ensemble d'expressions rationnelles. Dans le cas des expressions à multiplicités, le cassage d'une expression sera donc, comme la dérivation, une combinaison linéaire à gauche d'expressions rationnelles.

Il nous faut alors quelques petites notations préliminaires. Soit X une combinaison linéaire à gauche d'expressions rationnelles. Le coefficient de l'expression $\mathsf{1}$ dans X est noté δ_X et la combinaison linéaire égale a X sauf pour l'expression $\mathsf{1}$ qui a un coefficient nul est notée $(X)_\mathsf{p}$. Il en découle entre autres que $(X \oplus Y)_\mathsf{p} = X_\mathsf{p} \oplus Y_\mathsf{p}$ et $\delta_{X_\mathsf{p}} = \mathsf{0}$ pour tout X et Y. Ces notations sont évidemment cohérentes avec celles posées dans le cas booléen.

Définition 25. *Le* cassage *de l'expression rationnelle* E *est la combinaison linéaire à gauche* $\mathrm{B}(\mathsf{E})$ *définie inductivement par :*

$$\begin{aligned}
\mathrm{B}(\mathsf{0}) = \mathsf{0}\,, \quad \mathrm{B}(\mathsf{1}) = \mathsf{1}\,, \quad \forall a \in A \quad \mathrm{B}(a) = a\,,\\
\mathrm{B}(\mathsf{F}+\mathsf{G}) = \mathrm{B}(\mathsf{F}) \oplus \mathrm{B}(\mathsf{G}) \ ,\\
\mathrm{B}(\mathsf{F}\cdot\mathsf{G}) = (\mathrm{B}(\mathsf{F}))_\mathsf{p} \cdot \mathsf{G} \oplus \delta_{\mathrm{B}(\mathsf{F})}\mathrm{B}(\mathsf{G}) \ ,\\
\mathrm{B}(\mathsf{F}^*) = \mathsf{F}^* \ ,\\
\mathrm{B}(k\mathsf{F}) = k\mathrm{B}(\mathsf{F}) \ , \quad \mathrm{B}(\mathsf{F}k) = (\mathrm{B}(\mathsf{F})\,k) \ .
\end{aligned}$$

Le cassage des expressions s'étend naturellement au cassage de combinaisons linéaires d'expressions par :

$$\mathrm{B}\left([kX]\right) = k\mathrm{B}\left(X\right) \qquad \text{et} \qquad \mathrm{B}\left(X \oplus Y\right) = \mathrm{B}\left(X\right) \oplus \mathrm{B}\left(Y\right) \ .$$

Exemple 52 (*Ex. 38 cont.*)**.** Le cassage de l'expression E_1 donne : $\mathrm{B}\left(\mathsf{E}_1\right) = [6(\mathsf{F}_1 6)] \oplus [3(\mathsf{F}_1^* 2)]$

Nous pouvons remarquer que nous n'avons pas appelé cette fonction termes cassés, comme c'est le cas dans le cas booléen. C'est parce que nous ne voulons pas confondre les 'termes', qui sont les états de l'automate des termes dérivés cassés et donc un ensemble avec la combinaison linéaire dont les coefficients permettront de calculer les poids des transitions de l'automate.

Pour cela, par analogie avec les séries rationnelles, nous noterons $\mathsf{supp}(X)$ le support de la combinaison linéaire X, c'est-à-dire l'ensemble des expressions dont le coefficient est non nul dans X. L'ensemble $\mathsf{supp}(\mathrm{B}\left(\mathsf{E}\right))$, qui correspond donc aux expressions qui apparaissent dans le cassage de E, est alors appelé, par analogie avec le cas booléen, l'ensemble des *termes cassés* de E. Sa définition inductive est :

$$\begin{aligned}
\mathsf{supp}(\mathrm{B}\left(0\right)) = \emptyset \ , \quad \mathsf{supp}(\mathrm{B}\left(1\right)) = \{1\} \ , \quad \forall a \in A \quad \mathsf{supp}(\mathrm{B}\left(a\right)) = \{a\} \ , \\
\mathsf{supp}(\mathrm{B}\left(\mathsf{F} + \mathsf{G}\right)) = \mathsf{supp}(\mathrm{B}\left(\mathsf{F}\right)) \cup \mathsf{supp}(\mathrm{B}\left(\mathsf{G}\right)) \ , \\
\mathsf{supp}(\mathrm{B}\left(\mathsf{F} \cdot \mathsf{G}\right)) = \left(\mathsf{supp}(\mathrm{B}\left(\mathsf{F}\right))\right)_{\mathsf{p}} \cdot \mathsf{G} \cup \delta_{\mathsf{supp}(\mathrm{B}(\mathsf{F}))}\mathsf{supp}(\mathrm{B}\left(\mathsf{G}\right)) \ , \\
\mathsf{supp}(\mathrm{B}\left(\mathsf{F}^*\right)) = \mathsf{F}^* \ , \\
\mathsf{supp}(\mathrm{B}\left(k\mathsf{F}\right)) = \mathsf{supp}(\mathrm{B}\left(\mathsf{F}\right)) \ , \quad \mathsf{supp}(\mathrm{B}\left(\mathsf{F}k\right)) = (\mathsf{supp}(\mathrm{B}\left(\mathsf{F}\right))k) \ .
\end{aligned}$$

De par la définition, les termes cassés de E sont nécessairement des sous-expressions unitaires à gauche de l'expression E, de plus, on peut montrer par une récurrence directe que :

$$|\mathrm{B}\left(\mathsf{E}\right)| = |\mathsf{E}| \ .$$

Définition 26. *La* $\mathbb{K}$-dérivation cassée *d'une expression rationnelle* E *sur* A *par rapport à une lettre* a *de* A *est définie par :*

$$\frac{\partial_{\mathsf{b}}}{\partial a}\mathsf{E} = \mathrm{B}\left(\frac{\partial}{\partial a}\mathsf{E}\right) \ .$$

La $\mathbb{K}$*-derivation cassée par rapport à un mot non vide* ua *est définie par induction sur la longueur des mots :*

$$\frac{\partial_{\mathsf{b}}}{\partial ua}\mathsf{E} = \frac{\partial_{\mathsf{b}}}{\partial a}\frac{\partial_{\mathsf{b}}}{\partial u}\mathsf{E} \ .$$

Une induction directe permet de montrer que pour toute expression E, on a :

$$\forall a \in A \quad \frac{\partial}{\partial a} \mathrm{B}\left(\mathsf{E}\right) = \frac{\partial}{\partial a} \mathsf{E}\,, \qquad \text{et donc} \qquad \forall f \in A^{+} \quad \frac{\partial_{\mathsf{b}}}{\partial f} \mathsf{E} = \mathrm{B}\left(\frac{\partial}{\partial f} \mathsf{E}\right) \quad .$$

Finalement nous prenons la définition des termes dérivés cassés suivant – comme pour les termes dérivés nous prenons la définition inductive – :

Définition 27. *L'ensemble des* vraies termes dérivés cassés *est l'ensemble* $\mathrm{TBD}\left(\mathsf{E}\right)$ *défini inductivement par :*

$$\begin{aligned}
\mathrm{TBD}\left(0\right) = \mathrm{TBD}\left(1\right) = \emptyset, \qquad \mathrm{TBD}\left(a\right) &= \{1\} \quad , \\
\mathrm{TBD}\left(\mathsf{F} + \mathsf{G}\right) &= \mathrm{TBD}\left(\mathsf{F}\right) \ \cup \ \mathrm{TBD}\left(\mathsf{G}\right) \quad , \\
\mathrm{TBD}\left(\mathsf{F} \cdot \mathsf{G}\right) &= \left(\mathrm{TBD}\left(\mathsf{F}\right)\right)_{\mathsf{p}} \cdot \mathsf{G} \ \cup \ \mathrm{TBD}\left(\mathsf{G}\right) \ \cup \ \delta_{\mathrm{TBD}(\mathsf{F})} \mathsf{supp}(\mathrm{B}\left(\mathsf{G}\right)) \quad , \\
\mathrm{TBD}\left(\mathsf{F}^{*}\right) &= \left[\mathrm{TBD}\left(\mathsf{F}\right)\right] \cdot \mathsf{F}^{*} \quad , \\
\mathrm{TBD}\left(k\mathsf{F}\right) = \mathrm{TBD}\left(\mathsf{F}\right) \ , \quad \mathrm{TBD}\left(\mathsf{F}k\right) &= \left(\mathrm{TBD}\left(\mathsf{F}\right) k\right) \quad .
\end{aligned}$$

Définition 28. *L'ensemble des* termes dérivés cassés *de* E *est :* $\mathrm{BD}\left(\mathsf{E}\right) = \mathrm{TBD}\left(\mathsf{E}\right) \cup \mathsf{supp}(\mathrm{B}\left(\mathsf{E}\right))$.

Ces définitions ont été choisies pour que l'on ait le résultat suivant :

Proposition 51. *Les termes dérivés cassés d'une expression rationnelle sont le support du cassage des termes dérivés de cette expression.*

Démonstration. Cette proposition se prouve par induction sur l'expression rationnelle. L'induction des cas de base, de la somme, de l'étoile et des multiplications par un scalaire est directe par la définition inductive de $\mathsf{supp}(\mathrm{B}\left(\mathsf{E}\right))$. Nous allons développer le cas $\mathsf{E} = \mathsf{F} \cdot \mathsf{G}$:

$$\begin{aligned}
\mathsf{supp}(\mathrm{B}\left(\mathrm{TD}\left(\mathsf{F} \cdot \mathsf{G}\right)\right)) &= \mathsf{supp}(\mathrm{B}\left(\mathrm{TD}\left(\mathsf{F}\right) \cdot \mathsf{G} \ \cup \ \mathrm{TD}\left(\mathsf{G}\right)\right)) \\
&= \mathsf{supp}(\mathrm{B}\left(\mathrm{TD}\left(\mathsf{F}\right) \cdot \mathsf{G}\right)) \ \cup \ \mathsf{supp}(\mathrm{B}\left(\mathrm{TD}\left(\mathsf{G}\right)\right)) \quad .
\end{aligned}$$

Par induction $\mathsf{supp}(\mathrm{B}\left(\mathrm{TD}\left(\mathsf{G}\right)\right)) = \mathrm{TBD}\left(\mathsf{G}\right)$ et on a :

$$\mathsf{supp}(\mathrm{B}\left(\mathrm{TD}\left(\mathsf{F}\right) \cdot \mathsf{G}\right)) = \left(\mathsf{supp}(\mathrm{B}\left(\mathrm{TD}\left(\mathsf{F}\right)\right))\right)_{\mathsf{p}} \cdot \mathsf{G} \ \cup \ \delta_{\mathsf{supp}(\mathrm{B}\left(\mathrm{TD}(\mathsf{F})\right))} \mathsf{supp}(\mathrm{B}\left(\mathsf{G}\right)) \quad .$$

Par induction $\left(\mathsf{supp}(\mathrm{B}\left(\mathrm{TD}\left(\mathsf{F}\right)\right))\right)_{\mathsf{p}} = \left(\mathrm{TBD}\left(\mathsf{F}\right)\right)_{\mathsf{p}}$ et donc la proposition est prouvée. □

Proposition 52. *Les termes dérivés cassés d'une expressions rationnelle* E *sont des concaténations à droite de sous-expressions unitaires de* E *pondérées.*

Théorème 53. *Soit* $\mathrm{BD}(\mathsf{E}) = \{\mathsf{K}'_1, ..., \mathsf{K}'_n\}$ *l'ensemble des termes dérivés cassés de* E*. Pour toute lettre* a *de* A^**, il existe une matrice carrée* $a\mu'$*, de dimension* n *et à coefficients dans* $\mathbb{K}$ *telle que :*

$$\forall i \qquad \frac{\partial_{\mathsf{b}}}{\partial a}\mathsf{K}'_i = \bigoplus_j a\mu'_{i,j}\mathsf{K}'_j \ .$$

Démonstration. Il suffit, pour démontrer ce théorème, de montrer que le support de la dérivation cassante d'un terme dérivé cassé est un ensemble de termes dérivés cassés. On sait déjà, par le Théorème 46, que le support de la dérivation des termes dérivés est un ensemble de termes dérivés. Or comme les termes dérivés cassés sont le support du cassage des termes dérivés, en passant au cassage de la dérivation, on obtient bien le résultat. □

Finalement nous pouvons donner la définition naturelle de l'automate des termes dérivés cassés :

Définition 29. L'automate des termes dérivés cassés *d'une expression* E *est l'automate* $\mathcal{D}_b(\mathsf{E})$ *dont les états sont les termes dérivés cassés.*
(i) *Les états initiaux sont les termes cassés de* E *et le poids associé est celui de l'expression dans le cassage* $\mathrm{B}(\mathsf{E})$.
(ii) *La fonction finale est le terme constant de chaque terme dérivé cassé.*
(iii) *Il y a une transition de* K_i *à* K_j *étiqueté par* ka *si* $k = a\mu'_{i,j} \neq 0_{\mathbb{K}}$.

Cet automate, reconnaît la série dénoté par E :

Théorème 54. *Le comportement de l'automate des termes dérivés cassés* $\mathcal{D}_b(\mathsf{E})$ *de l'expression* E *est la série dénotée par* E.

Démonstration. Nous montrons que le comportement de l'automate des termes dérivés cassés est le même que celui des termes dérivés. En effet, si un mot u a pour coefficient k dans la série reconnue par l'automate des termes dérivés d'une expression rationnelle E, cela veut dire que :

$$\mathsf{c}(\frac{\partial}{\partial u}\mathsf{E}) = k \ .$$

Comme le comportement du cassage d'une expression est le comportement de l'expression on a :

$$\mathsf{c}(\frac{\partial_{\mathsf{b}}}{\partial u}\mathsf{E}) = k \ .$$

Cela signifie que la somme de tous les coefficients des calculs réussis de u dans l'automate des termes dérivés cassés de E est k et donc le comportement de l'automate est le même que celui des termes dérivé : c'est la série dénotée par E. □

Exemple 53. Prenons l'exemple de l'expression $\mathsf{E}_3 = 2((3a+b)5\cdot 2(c+d))$. Le cassage de l'expression E_3 donne :

$$\begin{aligned} \mathrm{B}\left(\mathsf{E}_3\right) &= [6(a5\cdot 2(c+d))] \oplus [2(b5\cdot 2(c+d))] \ , \\ \mathsf{supp}(\mathrm{B}\left(\mathsf{E}_3\right)) &= \{a5\cdot 2(c+d) \ , \ b5\cdot 2(c+d)\} \ . \end{aligned}$$

Les vrais termes dérivés cassés de E_3 sont :

$$\mathrm{TBD}\left(\mathsf{E}_3\right) = \{c \ , \ d \ , \ 1\} \ .$$

La dérivation des termes cassés donne :

$$\begin{aligned} \frac{\partial_{\mathsf{b}}}{\partial a} a5\cdot 2(c+d) &= [10c] \oplus [10d] \\ \frac{\partial_{\mathsf{b}}}{\partial b} b5\cdot 2(c+d) &= [10c] \oplus [10d] \end{aligned}$$

L'automate des termes dérivés cassés de E_3 est donc l'automate de la Figure 5.5.

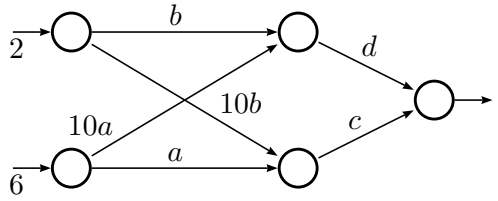


FIGURE 5.5 – L'automate des termes dérivés cassés de E_3

5.3.2 Calcul de l'automate

Une nouvelle fois, nous allons modifier la méthode de calcul des termes dérivés pour obtenir l'automate des termes dérivés cassés. Pour ce faire, comme la dérivation cassée est le cassage de la dérivation, il suffit de définir, comme dans le cas booléen, une opération de cassage des mots sur les u-nœuds pondérés.

Cassage de mots d'u-nœuds pondérés

Afin de définir le cassage de mots, nous introduisons d'abord la combinaison linéaire d'u-nœuds pondérés $\omega(x)$, où $x = (k, u, k')$, par :

$$\omega(x) = \begin{cases} k(1_{\mathbb{K}}, u, k') & \text{si } u \text{ est une feuille ou si } u = y^*, \\ k\omega(y) & \text{si } u = y\cdot z, \\ k\omega(y) \oplus k\omega(z) & \text{si } u = y + z. \end{cases}$$

Le *cassage* du mot X sur les u-nœuds pondérés est défini inductivement par la combinaison linéaire à gauche de mots suivante :
(i) $\mathrm{B}(X) = \varepsilon$ si $X = \varepsilon$
(ii) si $X = x_1 :: \ldots :: x_n$ où $x_i = (k_i, u_i, k'_i)$ et si $\omega(x_1) = \bigoplus_j t_j y_j$ où $y_j = (h_j, v_j, h'_j)$ alors :

$$\mathrm{B}(X) = \bigoplus_{v_j \neq 1} t_j(y_j :: \pi_{x_1}(y_j) :: x_2 :: \ldots :: x_n) \bigoplus_{v_j = 1} t_j \mathrm{B}(\pi_{x_1}(y) :: x_2 :: \ldots :: x_n) \ . \tag{5.5}$$

Cette définition de cassage des mots se justifie par le théorème suivant :

Théorème 55. *Soit X un mot sur les u-nœuds pondérés de l'arbre T_{E}. Si $\mathrm{B}(X) = \bigoplus_i k_i Y_i$, alors :*

$$\mathrm{B}(\llbracket X \rrbracket) = \bigoplus_i k_i \llbracket Y_i \rrbracket \ .$$

Démonstration. Le raisonnement de cette preuve est le même que celui dans le cas booléen. Pour simplifier l'écriture, nous allons uniquement traiter les cas qui ne sont pas traités dans le cas booléen, c'est-à-dire :

Si $\llbracket X \rrbracket = (k\mathsf{E})$, alors on a nécessairement X réduit à un seul u-nœud pondéré de la forme (k, u, k'). Par définition de $\omega(u)$, et comme le chemin d'un nœud ne récupère pas les coefficients à gauche, on a $\mathrm{B}((k, u, k')) = k\mathrm{B}((1_{\mathbb{K}}, u, k'))$. Comme $\llbracket (1_{\mathbb{K}}, u, k') \rrbracket = \mathsf{E}$ alors l'énoncé est vérifié.

Si $\llbracket X \rrbracket = (\mathsf{E}k)$, alors on a $X = x_1 :: \ldots :: x_n :: k$ ou bien $X = (1_{\mathbb{K}}, u, k)$. Dans le premier cas, la définition récursive du cassage de mot assure que $\mathrm{B}(X)) = \mathrm{B}(x_1 :: \ldots :: x_n) :: k$. Dans le deuxième cas, c'est la définition du chemin qui récupère les coefficients à droite qui assure que $\mathrm{B}(X)) = \mathrm{B}((1_{\mathbb{K}}, u, 1_{\mathbb{K}})) :: k$. Dans ces deux cas l'énoncé se vérifie. □

En particulier, si r est la racine de T_{E}, nous avons $\mathrm{B}(\mathsf{E}) = \mathrm{B}(r)$ et l'ensemble des termes cassés de E sont le support de $\mathrm{B}(r)$.

L'application du Théorème 55 et des résultats pour les termes dérivés cassés donne directement :

Théorème 56. *Soit E une expression rationnelle et X un mot sur les u-nœuds pondérés. Si $\mathsf{First}(X, a) = \bigoplus_i k_i \ell_i$ et $\mathrm{B}(\pi(\ell_i)) = \bigoplus_j h_{i,j} X_{i,j}$, alors :*

$$\frac{\partial_{\mathsf{b}}}{\partial a} \llbracket X \rrbracket = \bigoplus_{i,j} k_i h_{i,j} \llbracket X_{i,j} \rrbracket \ .$$

Théorème 57.

$$\mathrm{TD}\,(\mathsf{E}) = \bigcup_{\ell \in \mathrm{fp}(\mathsf{E})} \mathsf{supp}(\{[\![X]\!] \mid X \in \mathrm{B}(\pi(l))\})$$

Construction de l'automate

Il va donc être possible de construire l'automate des termes dérivés cassés de l'expression rationnelle par la représentation sous forme de mots sur les u-nœuds pondérés. Pour cela :

1. calcul pour tout nœud $x = (k, u, k')$ de l'arbre syntaxique – par un parcours de bas en haut – :
   – du terme constant $\mathsf{c}(x)$ et du terme constant $\mathsf{c}(u)$;
   – de l'ensemble des feuilles propres $\mathsf{First}(u)$;
   – de la marque associée $\overline{u}$;
2. calcul pour tout nœud x de l'arbre – par un parcours de haut en bas – de l'ensemble $\omega(x)$;
3. calcul du chemin $\pi(\ell)$ associé à toute feuille propre ℓ deT_{E} ; calcul du cassage $\mathrm{B}(\pi(\ell))$.
4. Pour tout mot de $\mathsf{supp}(\mathrm{B}(\pi(\ell)))$: calcul du mot réduit par ρ – par calculs successifs par ρ_1, ρ_2, ρ_3, ρ_4 et ρ_3 –. Pour chaque mot X il faut retenir l'ensemble des feuilles ℓ telles que $X \in \mathsf{supp}(\mathrm{B}(\pi(\ell)))$ ainsi que son coefficient dans le mot $\mathrm{B}(\pi(\ell))$;
5. identification des mots X et X' obtenus à l'étape précédente, tels que $\rho\left(\overline{X}\right) = \rho\left(\overline{X'}\right)$. Cette étape donne les états de l'automate, il faut garder la composition de chaque classe $\theta_b(q)$ de mots partageant la même marque ;
6. pour chaque état q, calcul de $\mathsf{First}(X)$ pour un représentant $X \in \theta_b(q)$ donné ;
7. identification des états finaux et de leur poids par le calcul de $\mathsf{c}(q)$ pour tous les états q ;
8. la fonction initiale est donnée par $\mathrm{B}(r)$;
9. il existe une transition (q_i, ka, q_j) si les trois conditions suivantes sont satisfaites :
   – il existe une feuille ℓ étiquetée par a avec un coefficient k' non nul dans $\mathsf{First}(q_i, a)$,
   – il existe $X \in \theta_b(q_j)$ avec un coefficient k'' non nul dans $\mathrm{B}(\pi(l))$ tel que $k'k'' = k$.

Deuxième partie

Enumération dans les langages rationnels

Chapitre 6

Systèmes de numération abstraits et fonctions rationnelles

Dans cette partie nous allons étudier des éléments de la théories des automates qui apparaissent naturellement dans les systèmes de numération abstraits. Ce chapitre joue un rôle de préliminaire à cette parie, en particulier nous donnons les définitions des systèmes de numération abstraits et certaines propriétés des fonctions rationnelles que nous utilisons par la suite.

Dans un cadre général, les systèmes de numération sont un moyen de représenter les nombres sous formes de mots afin de pouvoir les manipuler, d'effectuer des opérations. Par exemple dans le système binaire, les entiers naturels sont représentés par des mots sur l'alphabet $\{0, 1\}$. Plus précisément les représentations des entiers dans le système binaire sont, par convention, tous les mots sur $\{0, 1\}$ qui ne commencent pas par 0. Il apparaît donc que le langage des représentations forme un langage rationnel. Plus intéressant encore, les représentations sont rangées, pour un certain ordre – l'ordre radiciel, que nous définissons par la suite – dans le même ordre que leurs valeurs dans $\mathbb{N}$.

L'idée des systèmes de numération abstraits, définis dans [30], est de généraliser cette particularité, c'est-à-dire de prendre un langage L – le plus souvent rationnel – et d'associer à chaque mot la valeur correspondant à son rang pour l'ordre radiciel dans L. L'intérêt de prendre un langage rationnel est de pouvoir utiliser des automates pour faire des calculs sur L, donc sur les entiers. En particulier, nous allons nous intéresser dans cette partie au problème de l'énumération dans les langages rationnels. Ce problème peut

être vu sous deux aspects différents : le premier c'est de pouvoir 'compter' dans le système de numération donné, c'est-à-dire d'être capable de donner, à partir de la représentation de n'importe quel entier n, la représentation des entiers qui suivent. La deuxième opération importante est, à partir d'une représentation donnée, d'être capable d'en calculer la valeur.

Le premier problème sera traité par l'étude de la fonction successeur, qui associe à une représentation la représentation de l'entier suivant. Nous allons montrer que cette fonction appartient à une classe particulière de fonctions rationnelles qui sont réalisées par la lecture successive de mots dans des transducteurs 'déterministes'. Le deuxième problème sera traité par l'étude de la série énumératrice du système de numération. Cette série, par sa rationalité, permet le calcul direct de la valeur d'un mot dans le système.

6.1 Systèmes de numération abstraits

Dans cette section nous définissons plus formellement les systèmes de numération abstraits et la fonction successeur.

6.1.1 Définition

Soit A un alphabet muni d'un ordre total $<$. L'*ordre radiciel* $\prec$ sur A^* est défini par :[1]

$$u \prec v \quad \text{si} \quad \begin{cases} \text{soit} & |u| < |v| \ , \\ \text{ou} & |u| = |v| \ , \quad u = w\,a\,u' \ , \quad v = w\,b\,v' \quad \text{et} \quad a < b \ . \end{cases}$$

C'est-à-dire que l'ordre radiciel – également appelé ordre militaire – est l'ordre sur les mots de A^* qui prend tout d'abord en compte la longueur du mot puis l'ordre lexicographique.

Définition 30 ([30])**.** *Un* système de numération abstrait *(ou* SNA*) est un triplet $\mathcal{S} = (L, A, <)$ où A est un alphabet muni d'un ordre total $<$ et L est un langage infini de A^*.*

Le système $\mathcal{S}$ permet une bijection entre $\mathbb{N}$ et L en associant à tout entier n le $(n+1)$-ième mot de L dans l'ordre radiciel sur A^* donné par $<$. La représentation *de n est notée $\langle n \rangle_{\mathcal{S}}$ et, inversement, la* valeur *d'un mot w de L est notée $\pi_{\mathcal{S}}(w)$.*

1. Pour être plus précis, $\prec$ n'est pas réflexif et il n'est pas un ordre mais la partie stricte de l'ordre.

Nous avons donc :

$$\langle \pi_{\mathcal{S}}(w) \rangle_{\mathcal{S}} = w \qquad et \qquad \pi_{\mathcal{S}}(\langle n \rangle_{\mathcal{S}}) = n \ .$$

Dans la plupart des cas, l'alphabet A et l'ordre $<$ sur A sont fixés et il est possible de parler, par simplification, du SNA défini par le langage L et les notations peuvent être simplifiées en $\langle n \rangle_L$ et $\pi_L(w)$.

Si L est un langage rationnel de A^*, alors le système défini naturellement par L, $\mathcal{S} = (L, A, <)$, est dit rationnel.

Exemple 54. Soit $A = \{a, b\}$, avec $a < b$ et soit L_1 le langage des mots avec un nombre pair de b $L_1 = \{w \in \{a, b\}^* \mid |w|_b \equiv 0 \mod 2\}$. Le langage L_1 est reconnu par l'automate déterministe de la Figure 6.1 :

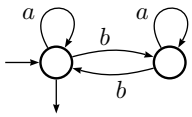


FIGURE 6.1 – Un automate déterministe reconnaissant les mots avec un nombre pair de b

La suite des représentations des entiers dans le système de numération abstrait défini par L_1 est : $\{\varepsilon, a, aa, bb, aaa, abb, bab, \ldots\}$ et, par exemple,

$$\langle 18 \rangle_{L_1} = a\,a\,b\,a\,b \qquad \text{and} \qquad \pi_{L_1}(b\,b\,a\,b\,b) = 29 \ .$$

Il est également possible de décrire le langage L_1 par l'arbre de la Figure 6.2 où les mots de L_1, jusqu'à la longueur 3, sont entourés.

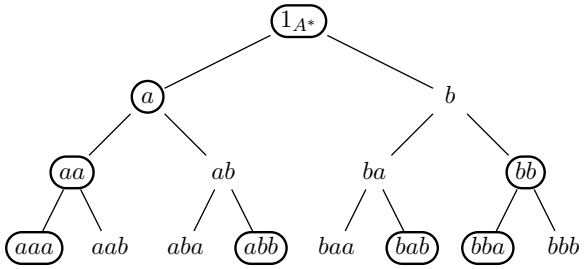


FIGURE 6.2 – L'arbre des mots de L_1

6.1.2 Enumération

Pour la suite, ce qui nous intéressera particulièrement, c'est l'énumération radicielle d'un langage rationnel, c'est-à-dire la suite des représentations des entiers dans le système de numération abstrait défini par ce langage. En particulier nous étudions la *fonction successeur* d'un langage L.

Définition 31. *La* fonction successeur *d'un langage L sur l'alphabet A, notée Succ_L, est la fonction de L dans L qui associe au mot $u \in L$ le mot v tel que v soit le plus petit mot de L plus grand que u pour l'ordre radiciel.*

Il résulte de cette définition que le successeur du mot u de L est le mot dont la valeur dans le système de numération abstrait défini par L est l'entier suivant :

$$\pi_L\left(\mathsf{Succ}_L(u)\right) = \pi_L\left(u\right) + 1 \ .$$

Exemple 55 (*Ex. 54 cont.*)**.** La fonction successeur apparaît naturellement sur l'arbre des représentations : le successeur d'un mot est le suivant en lisant l'arbre de gauche à droite puis de haut en bas. Par exemple : $\mathsf{Succ}_{L_1}(aa) = bb$, $\mathsf{Succ}_{L_1}(abb) = bab$ et $\mathsf{Succ}_{L_1}(bb) = aaa$.

6.2 Relations rationnelles

La fonction successeur d'un système de numération abstrait défini par un langage rationnel L est une fonction de L dans L. En fait c'est même une fonction rationnelle, c'est-à-dire que c'est une fonction réalisable par un transducteur fini. Nous donnons dans cette section toutes les définitions de ces notions, et certains résultats classiques qui nous seront utiles par la suite. Le lecteur pourra se référer à [43], par exemple, pour les énoncés et les preuves de cette section.

6.2.1 Relations et fonctions rationnelles

Définition 32. *Une relation α de A^* dans B^* est dite rationnelle si son graphe est une partie rationnelle de $A^* \times B^*$.*

L'image d'un élément u de A^* par une relations α de A^* dans B^* est l'ensemble des éléments v de B^* tels que (u, v) soit dans le graphe de α.

Définition 33. *Une relation rationnelle telle que l'image de tout mot a au plus un élément est appelée* fonction rationnelle.

Il apparaît de ces définitions que l'union et la concaténation de deux relations rationnelles sont des relations rationnelles. En revanche, l'union et la concaténation de fonctions rationnelles ne sont pas nécessairement des fonctions. Cependant, si les domaines de deux fonctions rationnelles sont disjoints alors l'union est également une fonction rationnelle. Si α est une fonction rationnelle dont le domaine est un ensemble préfixe, et si α' est une fonction rationnelle, alors la concaténation de α et de α', c'est-à-dire la concaténation des images en décomposant le mot d'entrée en une concaténation entre un mot du domaine de α et un mot du domaine de α', est également une fonction rationnelle.

Nous avons également :

Proposition 58. *La restriction d'une fonction rationnelle à un ensemble rationnel est une fonction rationnelle.*

Proposition 59. *Les fonctions (co-)séquentielles sont fermées par composition.*

6.2.2 Transducteurs finis

Les relations rationnelles sont les relations dont le graphe peut être reconnu par un automate fini sur $A^* \times B^*$. Nous appelons *transducteur* un tel automate – sur un produit de monoïdes libres –.

Comme l'ensemble $(A \times \{1_{B^*}\}) \cup (\{1_{A^*}\} \times B)$ est un ensemble générateur de $A^* \times B^*$, les relations rationnelles sont celles dont le graphe est le comportement d'un transducteur dont les transitions sont étiquetées par des éléments de cet ensemble générateur.

En fait, il est possible de voir ces objets, les transducteurs, d'un autre point de vue. En effet, plutôt que de voir une transition d'un transducteur comme un couple de $A^* \times B^*$, il est possible de le voir comme d'une fonction qui lit en entrée un élément de A^* pour sortir un élément de B^*. L'image d'un mot $u \in A^*$ est alors l'ensemble des sorties des chemins dont l'entrée est u. Avec cette vision, nous pouvons donner comme définition pour les transducteurs la définition suivante :

Définition 34. *Un* transducteur τ *de* A^* *dans* B^* *est un septuplet :*

$$\tau = \{Q, A^*, B^*, \delta, \eta, I, T\}, \qquad \textit{où :} \tag{6.1}$$

(i) *L'ensemble Q est un ensemble fini d'états.*
(ii) *La fonction de transition δ est une fonction partielle de $Q \times A^*$ dans Q dont le domaine est fini.*

(iii) *La fonction de sortie η est une fonction partielle de $Q \times A^*$ dans B^*, qui a le même domaine que δ.*
(iv) *La fonction initiale I est une fonction partielle de Q dans B^*.*
(v) *La fonction finale T est une fonction partielle de Q dans B^*.*

On note respectivement Q_I et Q_T les domaines des fonctions partielles I et T. On appelle automate d'entrée sous-jacent *de τ, l'automate $\{Q, A^*, \delta, Q_I, Q_T\}$.*

Un transducteur est dit *standard* si sont automate d'entrée sous-jacent est standard et la fonction initiale a pour valeur 1_{B^*} sur l'unique état initial i. Un transducteur est dit *lettre-à-lettre* si toutes les transitions sont étiquetées par des lettres en entrée et en sortie, c'est-à-dire que le domaine de δ et de η est dans $Q \times A$ et η est à valeurs dans B. Un transducteur est dit *fonctionnel* si il réalise une fonction rationnelle. Un transducteur fonctionnel co-accessible ne peut pas avoir deux chemins avec la même entrée et une sortie différente qui arrivent en un même état.

Finalement, et pour être complet, nous pouvons énoncer le résultat de décidabilité suivant :

Théorème 60 ([44])**.** *Il est décidable si un transducteur réalise une fonction.*

6.2.3 Fonction successeur

La fonction successeur d'un langage rationnel est une fonction rationnelle (cf. [24]), en fait on peut même énoncer le théorème plus précis suivant :

Théorème 61 ([25])**.** *La fonction successeur d'un langage rationnel est réalisée par un transducteur lettre-à-lettre.*

Afin d'affiner la classification des fonctions successeur pour les langages rationnels, nous allons considérer dans la suite des sous-classes de fonctions rationnelles : les fonctions séquentielles et séquentielles par morceaux. Nous allons ainsi montrer que la fonction successeur d'un langage rationnel est co-séquentielle par morceaux c'est-à-dire qu'elle peut être calculée par une union finie de transducteurs dont l'automate d'entrée sous-jacent est co-déterministe.

Chapitre 7

Fonctions séquentielles par morceaux

Dans la suite, nous nous intéressons à la fonction successeur d'un langage rationnel, en particulier à son calcul. Il apparaît donc utile de pouvoir classer cette fonction dans une sous-classe de fonctions rationnelles dans laquelle les fonctions peuvent avoir un calcul déterministe par des transducteurs.

Nous étudions dans ce chapitre la classe des fonctions séquentielles par morceaux. Cette classe de fonctions a été définie dans [21] sous le nom de fonctions plurisousséquentielles. Les fonctions séquentielles par morceaux étendent naturellement les fonctions séquentielles, qui sont les fonctions qui peuvent être réalisés par des transducteurs 'déterministes en entrée', c'est-à-dire tels que la lecture d'un mot se fait de manière déterministe et permet d'obtenir l'image de ce mot en un seul passage déterministe dans l'automate. Les fonctions séquentielles par morceaux peuvent être réalisées par une union finie de transducteurs réalisants des fonctions séquentielles ou, comme il a été montré dans [3], par des cascades de transducteurs séquentiels, c'est-à-dire une succession finie de transducteurs séquentiels telle que pour avoir l'image d'un mot, il faut lire celui-ci dans un transducteurs séquentiel puis l'état de sortie atteint détermine le transducteur séquentiel suivant qui lira l'image du mot obtenue.

Nous étudions ensuite les classes Seqpm et coSeqpm des fonctions séquentielles et co-séquentielles par morceaux et leur positionnement dans l'espace des fonctions rationnelles par rapport aux classes des fonctions séquentielles et co-séquentielles [3].

Les transducteurs réalisant les fonctions séquentielles par morceaux ont déjà été étudiés dans [21], et, en particulier, une preuve de la décidabilité de

ces fonctions, que nous retranscrirons, est donnée dans ce même papier.

7.1 Fonctions séquentielles

Les fonctions séquentielles sont particulièrement étudiées car elles peuvent être réalisées par des transducteurs dont le comportement ressemble à celui des automates déterministes. C'est-à-dire que l'on peut lire le mot d'entrée de manière déterministe dans le transducteur et obtenir ainsi l'image du mot avec une complexité linéaire en la taille du mot d'entrée. Nous rappelons dans cette section la définition et certaines caractérisations des fonctions séquentielles et des transducteurs qui leur sont associés. Pour les preuves manquantes de ce chapitre, le lecteur peut se référer à [43, IV].

7.1.1 Séquentialité

Par fonction séquentielle nous entendons ce qui est également communément appelé dans la littérature *fonction sous-séquentielle* (cf. [19] ou [7]). Nous définissons tout d'abord les transducteurs séquentiels :

Définition 35 ([45])**.** *Un* transducteur séquentiel τ *de* A^* *dans* B^* *est un transducteur dont l'automate d'entrée sous-jacent est déterministe. En particulier, la fonction initiale* I *est une fonction dont le domaine est l'unique état initial* i*.*

Définition 36 ([45])**.** *Une fonction rationnelle est* séquentielle *si elle est réalisée par un transducteur séquentiel.*

Nous notons Seq la famille des fonctions séquentielles. Il est possible de définir les notions duales :

Définition 37. *Un transducteur* co-séquentiel τ *de* A^* *dans* B^* *est un transducteur dont l'automate d'entrée sous-jacent est co-déterministe. En particulier, la fonction finale* T *est une fonction dont le domaine est l'unique état terminal* t*.*

Définition 38. *Une fonction rationnelle est* co-séquentielle *si elle est réalisée par un transducteur co-séquentiel.*

La famille des fonctions co-séquentielles sera notée coSeq. Et, comme pour les fonctions séquentielles, nous avons naturellement :

Remarque 13. Il est également possible de voir les transducteurs co-séquentiels, en prenant leur transposé, comme des transducteurs séquentiels qui lisent les mots de droite à gauche. Un tel transducteur sera appelé *transducteur séquentiel droit* – par opposition aux transducteurs séquentiels classiques qui pourront également être qualifiés de *gauche* –. Il est à noter que le terme gauche ou droit ne change pas la définition du transducteur séquentiel mais uniquement le sens de lecture du mot.

Nous choisissons de ne pas transcrire la notion de gauche/droit aux fonctions car elles associent des images à des mots sans tenir compte du sens de lecture du mot. Ainsi un fonction séquentielle sera réalisée par un transducteur séquentiel – gauche – ou par un transducteur co-séquentiel droit – par exemple le transposé du précédent –. Les fonctions co-séquentielles sont réalisées par des transducteurs séquentiels droits ou des transducteurs co-séquentiels gauches.

7.1.2 Caractérisation

Nous avons vu que les fonctions séquentielles sont les fonctions réalisées par des transducteurs séquentiels. Nous donnons maintenant une autre caractérisation, quasi-topologique, classique des fonctions séquentielles.

Pour cela, nous définissons tout d'abord la *distance préfixe* sur les mots d'un monoïde libre :

$$\mathsf{d_p}(u,v) = |u| + |v| - 2|u \wedge v| \ , \tag{7.1}$$

où $u \wedge v$ est le plus long commun préfixe de u et de v.

La donnée d'une distance permet d'énoncer la définition de fonctions lipschitziennes pour cette distance :

Définition 39. *Une fonction α de A^* dans B^* est dite k-Lipschitz (pour la distance préfixe) si :*

$$\forall u, v \in \mathsf{Dom}\,\alpha \ , \quad \mathsf{d_p}(\alpha(u), \alpha(v)) \leqslant k\ \mathsf{d_p}(u,v) \ .$$

Une fonction sera dite lipschitzienne si il existe k pour lequel elle est k-Lipschitz.

Cette définition donne une autre caractérisation des fonctions séquentielles :

Théorème 62 ([19])**.** *Soit α une fonction rationnelle de A^* dans B^*. La fonction α est séquentielle si, et seulement si, elle est lipschitzienne.*

Il existe naturellement une définition duale en prenant une distance suffixe $\mathsf{d_s}$ et pour laquelle les fonctions rationnelles lipschitziennes sont les fonctions co-séquentielles.

Il existe des fonctions rationnelles qui ne sont ni séquentielles ni co-séquentielles. En particulier ce résultat est connu pour les fonctions successeurs de langages rationnels (cf. par exemple [25]) : la fonction successeur d'un langage rationnel n'est pas nécessairement séquentielle ou co-séquentielle.

Exemple 56. Nous considérons la fonction $\alpha_0 = \mathsf{Succ}_{A^*}$, la fonction successeur sur A^* lui-même. Pour cet exemple nous prendrons l'alphabet A_2 à deux lettres : $A_2 = \{0, 1\}$. La fonction α_0 est co-séquentielle mais pas séquentielle.

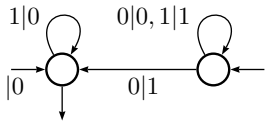


FIGURE 7.1 – Transducteur co-séquentiel réalisant α_0

La fonction α_0 est réalisée par le transducteur co-séquentiel de la Figure 7.1.

La caractérisation lipschitzienne des fonctions séquentielles nous permet de montrer la non séquentialité de α_0 : soit k un entier, posons $u_n = 1^n0$ et $v_n = 1^n$. La distance préfixe entre u_n et v_n vaut :

$$\mathsf{d_p}(u, v) = 1 \ .$$

Les images de u_n et de v_n par α_0 sont $\alpha_0(u_n) = 1^{n+1}$ et $\alpha_0(v_n) = 10^n$. Nous avons :

$$\mathsf{d_p}(\alpha_0(u_n), \alpha_0(v_n)) = 2n \ .$$

Il est donc possible de choisir n tel que $\mathsf{d_p}(\alpha_0(u_n), \alpha_0(v_n)) > k\ \mathsf{d_p}(u_n, v_n)$. La fonction α_0 n'est donc pas k-Lipschitz.

Comme séquentialité et co-séquentialité sont deux notions duales, il est facile d'imaginer à partir de cet exemple une fonction qui serait séquentielle mais non co-séquentielle. Ainsi, la fonction réalisée par le transducteur séquentiel de la Figure 7.2 n'est pas co-séquentielle.

Nous allons maintenant montrer par un nouvel exemple qu'il existe des fonctions qui ne sont ni séquentielles, ni co-séquentielles :

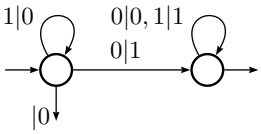


FIGURE 7.2 – Transducteur séquentiel réalisant la transposée de α_0

Exemple 57. [[25]] Considérons le système de numération de base le carré de la suite de Fibonacci. La base de ce système est définie par l'induction $u_{n+2} = 3u_{n+1} - u_n$. C'est-à-dire que la base est $U_\tau = \{1, 3, 8, 21, ...\}$. Les représentations dans ce système sont caractérisées par le langage rationnel $L_{sqf} = A^* \setminus \{A^*(21^*2)A^* \cup 0A^*\}$ où $A = \{0, 1, 2\}$.

La fonction $\alpha_{sqf} = \mathsf{Succ}_{L_{sqf}}$ n'est ni séquentielle, ni co-séquentielle.

Considérons $u_n = 2(1)^n$ et $v_n = 1^n$. Nous avons :

$$\mathsf{Succ}_{L_{sqf}}(u_n) = 1(0)^{n+1} \qquad \text{et} \qquad \mathsf{Succ}_{L_{sqf}}(v_n) = 1^{n-1}2 \ ,$$

Or, comme $\mathsf{d_s}(u_n, v_n) = 1$ et $\mathsf{d_s}(1(0)^{n+1}, 1^{n-1}2) = 2n+1$, la fonction α_{sqf} n'est pas lispchitzienne pour la distance suffixe, ce qui montre la non co-séquentialité.

Une étude similaire à l'exemple précédent avec α_0 montre également que la fonction α_{sqf} n'est pas séquentielle :

$$\mathsf{Succ}_{L_{sqf}}(21^n0) = 2(1)^{n+1} \qquad \text{et} \qquad \mathsf{Succ}_{L_{sqf}}(21^n) = 10^{n+1} \ .$$

On a donc 21^n0 et 21^n deux mots de distance préfixe 1 dont les successeurs ont pour distance préfixe $2n + 4$.

Rappelons qu'un ensemble de mots X est préfixe si et seulement si pour tout mot u de X il n'y a pas de préfixe propre de u dans X. Il est intéressant de donner le résultat suivant, qui correspond au Lemme 2 de [21] :

Proposition 63. *Soient α et β deux fonctions séquentielles. Si le domaine de α est un ensemble préfixe, alors le produit de α et β, défini par le produit des graphes de α et β :*

$$\alpha\beta(uv) = \alpha(u)\alpha(v)$$

est une fonction séquentielle.

7.1.3 Décidabilité

Afin d'étudier la décidabilité de la séquentialité de fonctions rationnelles – ou co-séquentialité par dualité –, nous allons donner une caractérisation des fonctions séquentielles. Plus précisément, le problème est de décider si un transducteur fonctionnel τ donné réalise une fonction séquentielle ou non. Ce problème a été résolu pour la première fois dans [19] (cf. également [7]) par une méthode qui utilise une caractérisation structurelle des transducteurs qui réalisent des fonctions séquentielles, à savoir que tous les états sont jumeaux deux à deux.

Afin de pouvoir exprimer plus simplement la propriété des transducteurs réalisants des fonctions séquentielles par morceaux, nous choisissons, dans la suite, de modifier la définition des états jumeaux qui est donnée dans [19] en séparant la notion de conjugaison et la notion de jumelage :

Définition 40. *Deux états q_1 et q_2 d'un automate sont dits jumeaux s'il existe deux mots u et v tels que $u \in L'_{q_1}$ et $u \in L'_{q_2}$ et s'il existe des cycles étiquetés par v sur q_1 et q_2.*

Par extension, deux états d'un transducteur fonctionnel sont dits jumeaux si les états correspondants dans l'automate d'entrée sous-jacent sont jumeaux (cf. Figure 7.3).

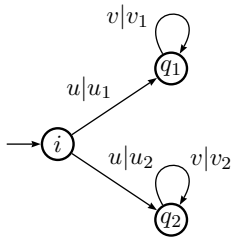


FIGURE 7.3 – Deux états jumeaux q_1 et q_2

Pour simplifier les définitions qui viennent, nous prendrons en hypothèse que le transducteur étudié est standard, avec pour unique état initial i. Cette hypothèse n'est pas trop forte puisqu'il est possible de transformer directement un transducteur en transducteur standard.

Définition 41. *Soit τ un transducteur fonctionnel standard de A^* dans B^*. Soient q_1 et q_2 deux états de τ. Les états q_1 et q_2 sont dits* q-conjugués *si pour tout couple (u, v) tel que u étiquette un chemin de q à q_1 et q_2 dont les images sont respectivement u_1 et u_2 et v étiquette l'entrée de circuits autour de q_1 et q_2 dont les images sont respectivement v_1 et v_2, on a :*

(i) *soit $v_1 = v_2 = 1_{B^*}$,*

(ii) *soit il existe* w *tel que* $u_2 = u_1 w$ *et* $w v_2 = v_1 w$,
(iii) *soit il existe* w *tel que* $u_2 w = u_1$ *et* $v_2 w = w v_1$.

Deux états états jumeaux d'un transducteur standard sont dits *conjugués* si ils sont i-conjugués. Il peut être intéressant d'insister sur le fait qu'il suffit d'un chemin d'entrant et d'un circuit ayant les mêmes entrées sur deux états pour être jumeaux. En revanche ce sont tous les couples des sorties qui doivent être vérifiés pour tester la conjugaison.

Dans la définition originale de [19], les états jumeaux sont définis comme deux états q_1 et q_2 tels que pour tout mots u et v qui satisfont la Figure 7.3 alors q_1 et q_2 sont conjugués. Nous avons changé ces définitions afin de décorreler la notion de jumelage de celle de conjugaison, ce qui est intéressant car on peut envisager de repérer dans un transducteur d'abord les états jumeaux – comme d'un motif sur un graphe – avant de tester leur conjugaison.

Le théorème donné dans [19] dit qu'un transducteur fonctionnel réalise une fonction séquentielle si tous les états sont deux-à-deux jumeaux – avec la définition d'états jumeaux de ce même papier –. Avec nos définitions, cela s'énonce :

Théorème 64 ([19])**.** *Soit* τ *un transducteur émondé réalisant une fonction rationnelle* α*. La fonction* α *est séquentielle si et seulement si tous les états jumeaux de* τ *sont conjugués.*

Ce théorème, qui donne donc une caractérisation des transducteurs fonctionnels qui réalisent une fonction séquentielle, permet de résoudre le problème de la décidabilité de la séquentialité d'une fonction rationnelle. La résolution du problème de jumelage des états est donnée en complexité exponentielle dans le papier original mais il a été montré qu'il est possible de l'obtenir en temps polynomial dans [48] et dans [6]. Dans ce dernier papier la séquentialité est prouvée en effectuant le produit par une action donnée sur le carré du transducteur – la preuve n'utilise pas directement le jumelage des états mais la Remarque 5 de [6] montre que les conditions énoncées pour la décidabilité sont les mêmes que la propriété de jumelage énoncée ci-dessus –.

7.2 Fonctions séquentielles par morceaux

Comme nous allons montrer par la suite que la fonction successeur d'un langage rationnel est une fonction qui peut-être réalisée par une union finie de transducteurs co-séquentiels par morceaux, nous allons étudier la classe de telles fonctions. Plus particulièrement, par dualité, nous allons étudier les

fonctions qui sont séquentielles sur un nombre fini de "morceaux" rationnels. Nous appelons ces fonctions les fonctions séquentielles par morceaux. Ces fonctions sont également réalisables par des passages successifs dans des transducteurs séquentiels qui permettent d'avoir l'image d'un mot de manière déterministe et en complexité linéaire en la taille du mot. La différence d'avec les fonctions séquentielles est qu'il faut lire le mot deux fois pour avoir l'image. Les fonctions séquentielles par morceaux sont même exactement les fonctions réalisées par de telles cascades de transducteurs séquentiels.

Dans cette section nous allons donc définir les fonctions séquentielles – et co-séquentielles – par morceaux, ainsi que donner des exemples de fonctions qui permettent de cerner la 'géographie' de ces classes de fonctions par rapport à l'ensemble des fonctions rationnelles, et aux classes des fonctions séquentielles et co-séquentielles. Pour finir, nous donnons une caractérisation structurelle des transducteurs qui réalisent des fonctions séquentielles par morceaux. Cette caractérisation reprend la notion d'états jumeaux appliquée à la caractérisation des fonctions séquentielles par morceaux donnée dans [21].

7.2.1 Définition

Définition 42. *Une fonction est dite séquentielle par morceaux si elle est une union finie de fonctions séquentielles dont les domaines sont deux à deux disjoints. Nous notons* Seqpm *la famille des fonctions séquentielles par morceaux.*

De cette définition il apparaît qu'une fonction séquentielle par morceaux peut être réalisée par une union finie de transducteurs séquentiels dont les domaines sont deux à deux disjoints. En fait, si l'union réalise une fonction, il est possible de se débarrasser de la condition de disjonction :

Proposition 65. *Une fonction est séquentielle par morceaux si, et seulement si, elle est réalisable par une union finie de transducteurs séquentiels.*

Démonstration. Si une fonction α est séquentielle par morceaux alors il existe $\alpha_1, \ldots, \alpha_n$ séquentielles et dont les domaines sont deux à deux disjoints tels que α soit l'union des α_i. Pour chaque i, il existe une transducteur séquentiel τ_i qui réalise α_i. L'union des transducteurs τ_i réalise α.

Réciproquement, soient $\tau_1, \ldots, \tau_n$ des transducteurs séquentiels dont l'union réalise une fonction α. Pour tout i, τ_i réalise une fonction séquentielle α_i. Les domaines des α_i ne sont pas nécessairement disjoints mais comme ils sont rationnels, l'intersection du domaine de α_i et du domaine

de α_j est un langage rationnel $X_{i,j}$. Comme l'union de α_i et α_j est une fonction, alors c'est également l'union de α_i et de la restriction de α_j a $\mathsf{Dom}\,\alpha_j - X$ et c'est donc une union de deux fonctions séquentielles à domaines disjoints. En réalisant la même opération pour tous les i et les j, on trouve que si l'union des transducteurs est une fonction, alors elle est séquentielle par morceaux. □

Les fonctions *co-séquentielles par morceaux* sont définie de manière duale par une union finie de fonctions co-séquentielles dont les domaines sont deux à deux disjoints. Leur famille est notée coSeqpm.

Comme les fonctions séquentielles et co-séquentielles sont fermées par composition, on a :

Proposition 66. *Les fonctions (co-)séquentielles par morceaux sont fermées par composition.*

7.2.2 Cascades de transducteurs séquentiels

Cette sous-section présente une nouvelle famille de machine déterministes qui permettent, comme les unions finies de transducteurs séquentiels, de réaliser les fonctions séquentielles par morceaux. Ces machines sont appelées cascades de transducteurs séquentiels et l'idée de ces cascades est de lire l'image d'un mot u par un transducteurs séquentiel dans un autre transducteur séquentiel qui dépend de l'état de sortie dans le premier transducteur. On peut très bien imaginer répéter ce processus h fois et obtenir ce que l'on appellera une cascade de hauteur h.

Définition 43 ([3])**.** *Une cascade de transducteurs séquentiels de hauteur h est un ensemble fini de transducteurs séquentiels*

$$\tau_0 \cup \bigcup_{1 \leqslant i \leqslant h-1} \tau_{i,q_{i,j}} \quad , \tag{7.2}$$

où les $q_{1,j}$ sont les états finaux de τ_0 et les $q_{i,j}$ sont tous les états finaux des transducteurs $\tau_{(i-1),q_{(i-1),j}}$.

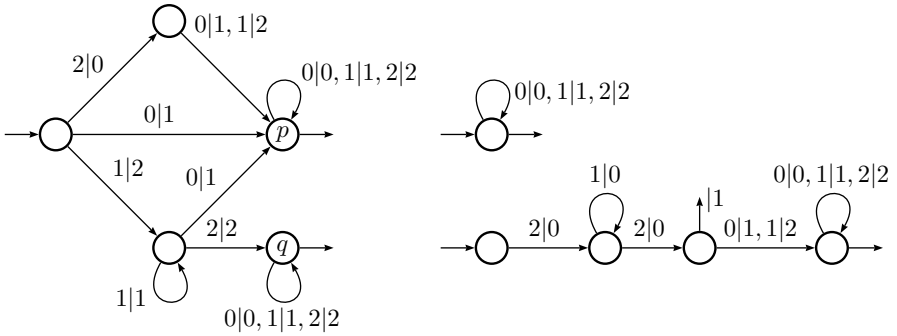
La fonction réalisée par une telle cascade est la fonction rationnelle α telle que :

$$\alpha(u) = \tau_{h,q_{i,j}}(\tau_{h-1,q_{i-1,j'}} \dots (\tau_0(u))) \quad , \tag{7.3}$$

où $q_{i,j}$ est l'état de sortie de $(\tau_{h-1,q_{i-1,j'}} \dots (\tau_0(u)))$.

Une cascade de hauteur 1 est donc un unique transducteur séquentiel et la fonction réalisée est une fonction séquentielle. Le transducteur de hauteur 1, τ_0, d'une cascade pourra être appelé *transducteur initial* de la cascade.

Exemple 58 (*Ex. 57 cont.*)**.** La fonction α_{sqf} est une fonction que nous avons montré être ni séquentielle ni co-séquentielle. La Figure 7.4 montre une cascade de transducteurs séquentiels droits réalisant la fonction α_{sqf}. Pour avoir l'image d'un mot u, il faut le lire dans le transducteur τ puis, suivant l'état de sortie p ou q, lire respectivement le mot de sortie alors obtenue dans le transducteur du haut σ_p ou du bas σ_q.



(a) τ (b) σ_p and σ_q

FIGURE 7.4 – Cascade de transducteurs séquentiels droits réalisant α_{fibo^2}

Nous montrons maintenant que les fonctions réalisées par les cascades de transducteurs séquentiels sont exactement les fonctions séquentielles par morceaux.

Théorème 67 ([3])**.** *Soit α une fonction rationnelle de A^* dans B^*. Les propositions suivantes sont équivalentes :*

(i) *α est une fonction séquentielle par morceaux ;*

(ii) *α est réalisable par une cascade de transducteurs séquentiels de hauteur 1 ou 2 ;*

(iii) *α est réalisable par une cascade de transducteurs séquentiels de hauteur $h \geqslant 1$.*

Démonstration. $(i) \Rightarrow (ii)$: Si α est séquentielle par morceaux, il est possible de construire une union finie de transducteurs séquentiels dont les domaines sont deux à deux disjoints et réalisant α. Notons ces transducteurs $\tau_1, \ldots, \tau_n$ et leurs domaines respectifs $L_1 \cdots L_n$. Il est possible de construire un automate déterministe $\mathcal{A}$ dont l'ensemble T des états finaux est partitionné en $T_1, \ldots, T_n$ de telle manière que le langage des mots reconnus dans $\mathcal{A}$ par les états de T_i est L_i.

Soit τ_0 le transducteur dont l'automate sous-jacent d'entrée est $\mathcal{A}$ et dont la fonction de sortie est l'identité sur chaque transition – ce transducteur réalise une restriction de l'identité –. La cascade de hauteur 2 dont le transducteur initial est τ_0 et dont les transducteurs de hauteurs 2 sont les $\tau_1, \ldots, \tau_n$ – associés aux états finaux $T_1, \ldots, T_n$ de τ_0 – réalise la fonction α.

$(ii) \Rightarrow (iii)$: Cette implication est triviale.

$(iii) \Rightarrow (i)$: Décomposons tout transducteur séquentiel τ^k de la hauteur k d'une cascade en une union de transducteurs identiques à τ^k mais avec chacun un unique état final. Comme τ^k est séquentiel, cette décomposition est une union finie de transducteurs, notés $\tau_1^k, \ldots, \tau_n^k$, séquentiels.

La composition de fonctions séquentielles est une fonction séquentielle donc la composition les fonctions réalisées par τ_i^k avec les fonctions réalisées τ_j^{k+1} est une fonction séquentielle. L'union de toutes ces fonctions séquentielles forme donc une fonction séquentielle par morceaux. □

Naturellement il y a également le théorème dual suivant :

Théorème 68. *Soit α une fonction rationnelle de A^* dans B^*. Les propositions suivantes sont équivalentes :*
(i) *α est une fonction co-séquentielle par morceaux ;*
(ii) *α est réalisable par une cascade de transducteurs co-séquentiels de hauteur 1 ou 2 ;*
(iii) *α est réalisable par une cascade de transducteurs co-séquentiels de hauteur $h \geqslant 1$.*

L'exemple précédent montre donc que la fonction α_{sqf} est co-séquentielle par morceaux.

7.2.3 coSeqpm et Seqpm

Dans un souci de complétude, nous étudions maintenant les familles des fonctions séquentielles et co-séquentielles par morceaux afin d'établir une "géographie" de cette classe relativement aux familles des fonctions séquentielles et co-séquentielles.

Par les exemples précédents, nous avons déjà montré que Seq $\subsetneq$ RatF et coSeq $\subsetneq$ RatF. Par définition, nous avons également Seqpm $\subseteq$ RatF, coSeqpm $\subseteq$ RatF ainsi que Seq $\subseteq$ Seqpm et coSeq $\subseteq$ coSeqpm.

L'exemple 57 montre également qu'il existe des fonctions qui sont co-séquentielles par morceaux sans être ni séquentielles, ni co-séquentielles :

Il existe des fonctions qui sont séquentielles sans être co-séquentielles et qui sont à la fois séquentielles et co-séquentielles par morceaux (cf. [3]).

Exemple 59. Considérons la fonction α_1 de A^* dans lui même, avec $A = \{0,1\}$ dont le domaine est $\mathsf{Dom}\,\alpha_1 = \{u_n = 00^n \mid n \in \mathbb{N}\} \cup \{v_n = 10^n \mid n \in \mathbb{N}\}$ et définie par :

$$\alpha_1(u_n) = 11^n \quad , \quad \alpha_1(v_n) = 1^n 0 \quad .$$

La fonction α_1 est séquentielle mais pas co-séquentielle mais elle trivialement l'union de fonctions séquentielles et co-séquentielles donc une fonction séquentielle et co-séquentielle par morceaux.

Nous montrons enfin qu'il existe des fonctions qui ne sont ni séquentielles par morceaux, ni co-séquentielles par morceaux.

Soit $A = \{a, b\}$ et $\varphi\colon A^* \to A^*$ la fonction, que l'on appelle *réduction de Fibonacci*, et qui associe à chaque mot w de A^* l'unique mot $\varphi(w)$ obtenu à partir de w en appliquant la règle de réécriture $abb \to baa$ et qui ne contient plus aucun facteur abb. La réduction de Fibonacci est une fonction rationnelle (cf. par exemple [7, Exer. III.5.5] ou [37]) qui n'est ni séquentielle, ni co-séquentielle (cf. par exemple [23]).

Proposition 69 ([3])**.** *La réduction de Fibonacci n'est ni une fonction séquentielle par morceaux, ni une fonction co-séquentielle par morceaux.*

Démonstration. Pour montrer que la fonction n'est pas séquentielle par morceaux, nous montrons que, pour tout K, on ne peut pas trouver k fonctions K-Lipschitz dont l'union réalise φ. Pour cela, étant donné K, il suffit de trouver, pour tout k, $k+1$ mots pour lesquels la fonction φ ne satisfait pas le critère K-Lipschitz deux à deux.

Notons tout d'abord que, même si $\varphi(uv) \neq \varphi(u)\varphi(v)$, le facteur aa permet de séparer un mot :

$$\varphi(u\,a\,a\,v) = \varphi(u)\,a\,\varphi(a\,v) \quad . \tag{7.4}$$

Considérons tout d'abord les mots $u_n = (ab)^n a$ et $v_n = (ab)^n b$. Nous avons $\mathsf{d_p}(u_n, v_n) = 2$ et :

$$\mathsf{d_p}(\varphi(u_n), \varphi(v_n)) = \mathsf{d_p}(u_n, (aa)^n b) = 4n + 1 \quad .$$

Considérons les $k+1$ mots suivants (pour $1 \leqslant i \leqslant k+1$) :

$$w_i = w_{i,k}\,a\,a\,w_{i,k-1}\,a\,a \ldots a\,a\,w_{i,1} \quad ,$$

où :

$$w_{i,j} = \begin{cases} v_{l_i} & \text{si i=j} \\ u_{l_i} & \text{sinon} \end{cases} \quad .$$

où les l_i forment une suite croissante qui sera définie explicitement par la suite. De (7.4) nous tirons que :

$$\varphi(w_i) = u_{l_k} a\, a\, u_{l_{k-1}} a\, a \ldots a\, a\, u_{l_{i+1}} a\, a\, b\, (a\, a)^{l_i} a\, a\, u_{l_{i-1}} a\, a \ldots a\, a\, u_{l_1}$$

pour $1 \leqslant i \leqslant k$, et :

$$\varphi(w_{k+1}) = X_{k+1} = u_{l_k} a\, a\, u_{l_{k-1}} a\, a \ldots a\, a\, u_{l_1} \quad .$$

Nous déduisons alors que pout tous i, j, $1 \leqslant i < j \leqslant k+1$, nous avons :

$$\mathsf{d_p}(w_i, w_j) = 2\left(\sum_{1 \leqslant m \leqslant i-1} (2l_m + 3)\right) \quad ,$$

$$\mathsf{d_p}(\varphi(w_i), \varphi(w_j)) = 2\left(\sum_{1 \leqslant m \leqslant i-1} (2l_m + 3) + 2l_i + 1\right) \quad .$$

Si nous choisissons les longueurs $l_1, \ldots, l_k$ inductivement avec :

$$l_1 > \frac{1}{2}(K-1)$$

$$l_i > \frac{1}{2}(K-1)\left(\sum_{1 \leqslant m \leqslant i-1} (2l_m + 3)\right)$$

alors pour tous i, j, $1 \leqslant i < j \leqslant n+1$, nous avons :

$$\mathsf{d_p}(\varphi(w_i), \varphi(w_j)) > K\,\mathsf{d_p}(w_i, w_j)$$

ainsi φ ne peut être l'union de k fonctions K-Lipschitz et n'est donc pas une fonction séquentielle par morceaux.

Une méthode similaire, en prenant $u_n = a\,(b\,b)^n$ et $v_n = a\,b\,(b\,b)^n$ montre également que φ n'est pas une fonction co-séquentielle par morceaux. □

Nous pouvons finalement établir la géographie des fonctions rationnelles comme le montre la Figure 7.5 où les inclusions sont toutes strictes.

7.2.4 Décidabilité

Finalement, dans cette sous-section, nous étudions la décidabilité de la séquentialité par morceaux des fonctions rationnelles. La décidabilité a déjà été prouvée dans [21] à l'aide d'une caractérisation des fonctions rationnelles par morceaux qui utilise la notion de branchement que nous allons rappeler.

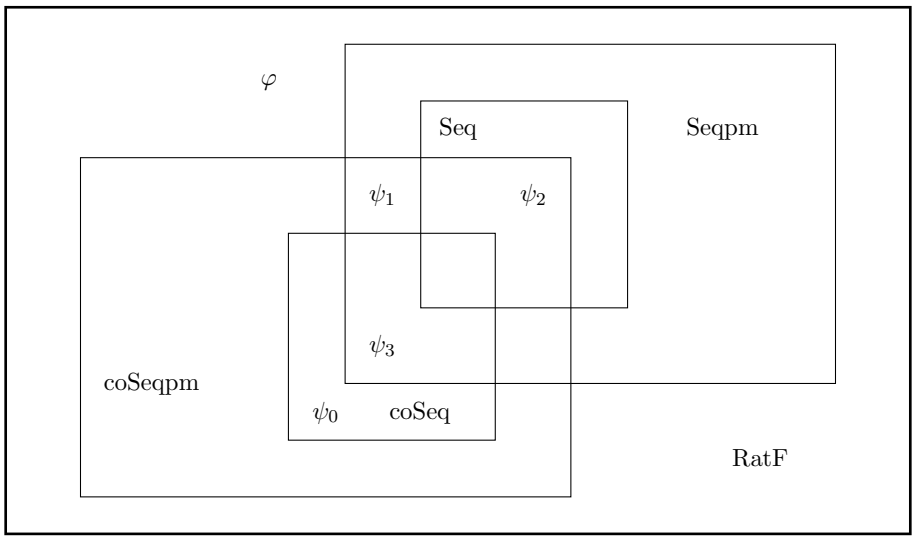


FIGURE 7.5 – Géographie des fonctions rationnelles

Nous allons ensuite transcrire les définitions de branchements et les caractérisations associées afin de reprendre une terminologie proche de celle des états jumeaux et de la conjugaison. Ce travail de transcription est la première étape pour envisager une décidabilité structurelle de la séquentialité par morceaux : c'est-à-dire la reconnaissance d'un motif dans le transducteur qui réalise la fonction.

Définition 44 ([21])**.** *Soit τ un transducteur fonctionnel. Un* branchement *est un triplet d'états (q, q_1, q_2) tel que l'intersection des langages des mots lus de q à q_1 et des mots lus de q à q_2 est non vide (cf. Figure 7.6).*

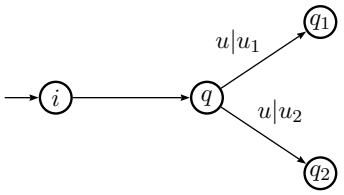


FIGURE 7.6 – Un branchement

Un branchement faible *est un branchement* (q, q_1, q_2) *tel que la distance préfixe des images de chemins étiquetés par le même mot de* q *à* q_1 *et* q_2 *est bornée : il existe un entier* $k > 0$ *tel que pour tout* u *qui est l'étiquette d'entrée d'un chemin de* q *à* q_1 *et à* q_2 *dont les images respectives sont* u_1 *et* u_2 *on a :* $\mathsf{d_p}(u_1, u_2) < k$.

Un branchement fort *est un branchement qui n'est pas faible. Un* branchement absolu *est un branchement fort* (q, q_1, q_2) *où* $q = q_1$ *ou* $q = q_2$.

L'existence de branchements forts dans un transducteur est en fait directement liée à l'existence d'états jumeaux non-conjugués. Nous retranscrivons ici la caractérisation de [21] avec la terminologie des états jumeaux. En particulier nous montrons par la suite que l'existence de branchements absolus est conjointe à l'existence d'états q_1 et q_2 *monozygotes* non q_1-conjugués :

Définition 45. *Deux états* q_1 *et* q_2 *d'un automate ou d'un transducteur standard émondé sont dits* monozygotes *si il existe un mot* u *tel que* q_1 *et* q_2 *sont accessibles à partir de* q_1 *par* u *et il existe un mot* v *qui est l'entrée d'un circuit sur* q_1 *et* q_2 *(cf. Figure 7.7)*

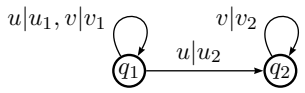


FIGURE 7.7 – Deux états monozygotes q_1 et q_2

Deux états monozygotes q_1 et q_2 sont jumeaux. Ce qui nous intéresse dans cette section, pour caractériser les transducteurs fonctionnels réalisant une fonction séquentielle par morceaux, c'est la q_1-conjugaison – ou q_2-conjugaison – des états monozygotes. C'est-à-dire que nous étudions la notion de conjugaison des états pour les chemins initiaux qui passent par l'un des deux états.

En particulier deux états monozygotes q_1 et q_2 peuvent être simultanément non-conjugués et q_1-conjugués (cf. Figure 7.8). En revanche, deux états monozygotes non q_1-conjugués sont nécessairement également des états jumeaux non-conjugués.

Comme pour les états jumeaux, la notion de q_1-conjugaison des états monozygotes, bien qu'intrinsèquement liée au fait qu'ils soient monozygotes, est séparée pour permettre de tester la séquentialité par morceaux, d'abord par une recherche de motif (celui de la Figure 7.7), puis un test de conjugaison sur les sorties des chemins impliqués.

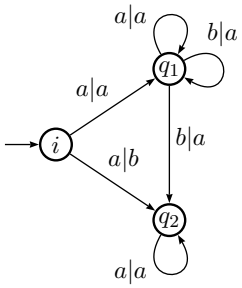


FIGURE 7.8 – Deux états monozygotes non-conjugués et q_1-conjugués

Afin de relier la notion d'états jumeaux et monozygotes au travail de [21] sur les branchements, nous montrons la propriété suivante :

Propriété 70. *Un transducteur standard possède un branchement absolu si, et seulement si, il existe un couple d'états monozygotes* (q_1, q_2) *non* q_1*-conjugués.*

Démonstration. Un branchement est fort s'il n'est pas faible, c'est-à-dire que les distances préfixes des images de chemins qui arrivent en q_1 et q_2 à partir de q par le même mot u ne sont pas bornées. Cette condition implique qu'il existe une infinité de tels mots u. Nous appelons *branchement infini* un tel branchement (q, q_1, q_2), avec une infinité de mots qui sont l'entrée de chemins à la fois de q à q_1 et de q à q_2. Un branchement fort ou absolu est nécessairement infini mais un branchement infini n'est pas nécessairement fort.

Pour qu'un branchement (q, q_1, q_2) soit infini il faut qu'il y ait une infinité de mots lus à la fois de q à q_1 et de q à q_2. Cela implique qu'il y a un état q'_1, respectivement q'_2, sur un chemin de q à q_1, resp. q_2, tels que q_1 et q_2 ont un circuit dont l'entrée est la même. Les états q'_1 et q'_2 sont alors des états jumeaux. La Figure 7.9 montre un branchement infini.

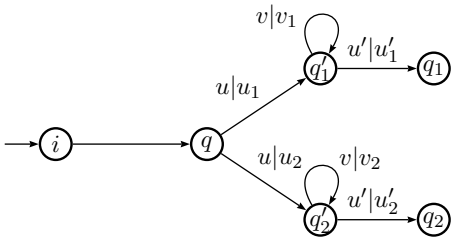


FIGURE 7.9 – Un branchement infini

Nous avons donc montré que l'existence d'un branchement infini im-

plique l'existence d'états jumeaux. La réciproque étant directe, nous montrons maintenant que l'existence d'état jumeaux non-conjugués est équivalente à l'existence de branchements forts.

Il est trivial de montrer que si deux états sont jumeaux non-conjugués alors la distance préfixe des images des mots obtenus en faisant des tours de boucles n'est pas bornée et donc ils forment, avec l'état initial, un branchement fort.

Montrons que si un automate a un branchement fort, alors il a des états jumeaux non-conjugués. Un branchement fort (q, q_1, q_2) induit une branchement infini comme sur la Figure 7.9 où les états q'_1 et q'_2 sont jumeaux et les distances préfixes des mots $u_1 v_1^n u'_1$ avec les mots $u_2 v_2^n u'_2$, pour tout entier n, ne sont pas bornées. Il résulte que, pour tout n, les distances préfixes de $u_1 v_1^n$ et $u_2 v_2^n$ ne sont pas bornées puisque u'_1 et u'_2 sont de longueurs fixes. Cette condition implique directement que les états q'_1 et q'_2 ne sont pas conjugués. Un branchement fort a donc la forme d'un branchement infini dont les états q'_1 et q'_2 ne sont pas conjugués.

Un branchement est absolu s'il est fort et que $q_1 = q$. C'est-à-dire qu'on a la situation de la Figure 7.10. Les états q'_1 et q'_2 sont jumeaux monozygotes avec un circuit commun v et comme boucle sur q'_1 qui amène également en q'_2 le mot $u'u$. Le même argument que pour la non-conjugaisons des états q'_1 et q'_2 lors d'un branchement fort montre que q'_1 et q'_2 sont monozygotes non q'_1-conjugués en cas de branchement absolu.

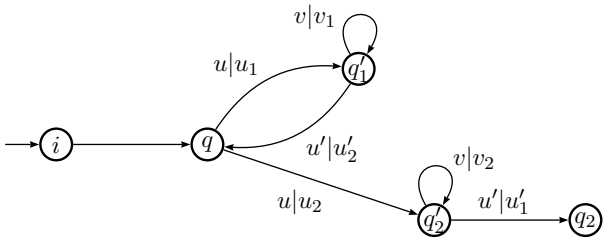


FIGURE 7.10 – Un branchement absolu

Finalement, pour la réciproque, supposons que deux états q_1 et q_2 soient monozygotes non q_1-conjugués, avec les notations de la Figure 7.7. Les distances préfixes des mots $u_1 v_1^n$ et $u_2 v_2^n$, pour tout n, ne sont pas bornées, puisque les états sont non q_1-conjugués, et le branchement (q_1, q_1, q_2) est absolu. □

Théorème 71 ([21]). *Soit τ un transducteur émondé standard réalisant une fonction rationnelle α. La fonction α est séquentielle par morceaux si et*

seulement si il n'y a pas d'états q_1 et q_2 de τ qui soient monozygotes non q_1-conjugués.

Démonstration. Cette preuve est la preuve de [21] adaptée à la terminologie des états jumeaux et monozygotes. Pour simplifier la preuve, nous reprenons l'hypothèse de [21] que τ n'a qu'un seul état initial i. Le sens direct est trivial.

Soit τ un transducteur réalisant la fonction α. Soit Q son ensemble d'états et Q_1 le sous-ensemble de Q des états qui ont un jumeau non conjugué. Montrons que si τ n'a pas d'états monozygotes q_1 et q_2 non q_1-conjugués, alors α est séquentielle par morceaux.

Pour tout état $q \in Q$, notons X_q l'ensemble des mots qui sont l'étiquette d'entrée d'un chemin de i à q dont aucun état intermédiaire n'est dans Q_1. L'union des X_q pour les états de Q_1 est $X = \bigcup_{q \in Q_1} X_q$. L'ensemble Y_q est l'ensemble des mots qui sont l'étiquette d'entrée d'un chemin de q à un état de sortie – c'est-à-dire que Y_q est égal à L_q dans l'automate d'entrée sous-jacent de τ –.

Posons $Z = \mathrm{Dom}(\alpha) \setminus XA^*$ et $W_q = X_q Y_q$. Il est possible de décomposer le domaine de α en :

$$\mathrm{Dom}(\alpha) = \bigcup_{q \in Q_1} W_q \cup Z$$

Étudions les restrictions de la fonction α à chaque ensemble rationnel W_q et à Z. Posons α' la restriction de α à l'ensemble Z. Cette restriction est réalisée par le transducteur τ', restriction de τ à l'ensemble d'états $Q - Q_1$. Le transducteur τ' n'a donc pas d'états jumeaux non-conjugués et donc, par le Théorème 64, α' est une fonction séquentielle.

Posons α_q la restriction de α à l'ensemble W_q. Comme X_q est un ensemble préfixe, il est possible de voir la fonction α_q comme le produit de deux fonctions β_q et γ_q de domaines respectifs X_q et Y_q. Plus précisément β_q est la fonction qui associe à chaque mot u de X_q l'image du chemin de i à q dont l'entrée est u. Cette fonction est réalisée par un transducteur qui est la restriction de τ à l'ensemble d'états $Q - (Q_1 \setminus \{q\})$ et dont l'ensemble des états finaux est $\{q\}$. Ce transducteur n'a pas d'états jumeaux non-conjugués et donc β_q est une fonction séquentielle.

La fonction γ_q est la fonction qui à tout mot v de Y_q associe le mot de l'image du chemin de q à un état final dont l'entrée est v dans τ. Cette fonction est réalisée par le transducteur τ_q construit à partir de τ mais dans lequel q est le seul état initial. Montrons que dans τ_q, i n'est pas un état accessible, c'est-à-dire qu'il n'y a aucun chemin de q à i dans τ.

Comme $q \in Q_1$, il existe $q' \in Q$ tel que q et q' soient des états jumeaux non-conjugués. Il existe donc un couple de mots (u, v) où u étiquette l'entrée de chemins de i à q et q' et v étiquette l'entrée de circuits sur q et q' et tels que q et q' sont non-conjugués pour le couple (u, v). Supposons qu'il existe un chemin de q à i étiqueté w. Alors q et q' sont tous les deux accessibles par le mot uwu et il y a un chemin de i à q' passant par q étiqueté uwu. Les états q et q' sont donc monozygotes. Comme q et q' sont non i-conjugués pour le couple (u, v), ils sont également non q-conjugués pour (uwu, v), ce qui est contradictoire avec l'hypothèse. L'état i n'est donc pas accessible dans τ_q.

Il est donc possible d'éliminer l'état i de τ_q sans changer la fonction réalisée. Ce transducteur à un état de moins que τ et pas d'états monozygotes q_1 et q_2 non q_1-conjugués. Par récurrence sur le nombre d'état, comme un transducteur avec un unique état est séquentiel, la fonction γ_q est donc séquentielle par morceaux.

Pour démontrer le théorème il reste juste à montrer que le produit de β_q et γ_q est une fonction séquentielle par morceaux. Notons $\gamma_{q,k}$ des fonctions séquentielles dont l'union donne γ_q. Le produit de β_q et de γ_q est l'union des produit de β_q et des $\gamma_{q,k}$. Comme X_q, le domaine de β_q est préfixe, ce produit est une fonction séquentielle. Il en résulte que α_q est séquentielle par morceaux et donc également α.

□

La preuve du théorème ci-dessus implique également la décidabilité de la séquentialité par morceaux d'une fonction réalisée par un transducteur fonctionnel. Cette décidabilité, qui est celle montrée dans [21], est en complexité exponentielle.

Chapitre 8

Fonction successeur de langages rationnels

Ce chapitre traite particulièrement de la fonction successeur pour des langages rationnels. Nous avons déjà vu dans les chapitres précédents que la fonction successeur dans un langage rationnel est une fonction rationnelle mais qu'elle n'est pas nécessairement séquentielle ou co-séquentielle, à travers les exemples 56 et 57. Nous allons maintenant montrer que le résultat trouvé pour ces deux exemples est un résultat général :

Théorème 72 ([3])**.** *La fonction successeur d'un langage rationnel est co-séquentielle par morceaux.*

La preuve de ce théorème, que nous donnons dans ce chapitre, construit une union finie de transducteurs séquentiels droits réalisant la fonction successeur d'un langage L à partir d'un automate fini déterministe qui reconnaît L. En fait nous réduisons le problème de la fonction successeur à une fonction successeur restreinte aux mots dont le successeur est de même longueur. Pour retrouver la fonction successeur à partir de cette restriction, il faut imaginer introduire un caractère spécial que l'on insère autant de fois que nécessaire en début de mot.

Afin de montrer la co-séquentialité par morceaux des fonctions successeur à longueur constante, nous présentons tout d'abord des constructions particulières sur des automates et sur des transducteurs pour avoir des transducteurs séquentiels – nous travaillons sur les transducteurs séquentiels sans préciser s'ils sont droits ou gauches –.

8.1 Produit synchronisé d'automates monocyles

Dans cette section nous voulons établir le résultat suivant qui sera utilisé par la suite dans la preuve du Théorème 72 :

Proposition 73. *Si L et K sont deux langages rationnels avec au plus un mot par longueur, alors la fonction α qui associe à tout mot de L le mot de K, de même longueur, si il existe, est une fonction séquentielle et co-séquentielle par morceaux.*

Pour ce faire nous définissons les *langages rayon*, qui sont les langages reconnus par des automates avec un seul cycle, appelés *automates monocycles*. Nous montrons ensuite que le produit synchronisé de deux automates monocycles réalise une fonction séquentielle par morceaux.

8.1.1 Définitions

Langages, automates et transducteurs monocycles

Nous utilisons une terminologie utilisée par Reutenauer dans [38] et nous appelons *langage rayon* un langage L de la forme $L = uv^*w$. L'intérêt de tels langages est que tout langage à croissance bornée est une union finie de langages rayons (folklore, cf. par exemple [43, I.8.4]). En particulier, ce qui va nous intéresser, les langages qui n'ont qu'au plus un mot par longueur sont une union fini de langages rayons.

Un *automate monocycle* est un automate émondé avec un seul état initial, un seul état final et qui est composé d'un unique circuit et de deux chemins qui relient le circuit à l'état initial et l'état final. Un *transducteur monocycle* est un transducteur dont l'automate d'entrée sous-jacent est monocycle.

Un automate monocycle est dit à *boucle préfixe* soit s'il n'a pas de circuit, soit si celui-ci passe par l'état initial. Il est dit à *boucle suffixe* s'il n'a pas de circuit ou bien si celui-ci passe par l'état final.

La proposition suivante traduit les définitions précédentes par la caractérisation des degrés entrants et sortants des états d'un automate monocycle :

Proposition 74. *Les automates monocycles sont les automates émondés avec :*
(i) *un unique état initial,*
(ii) *un unique état final,*
(iii) *le degré entrant de tous les états est au plus* 1 *sauf pour au plus un état pour lequel il vaut* 2*,*

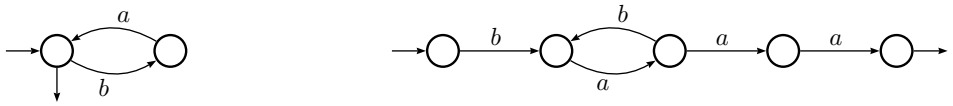
(iv) *le degré sortant de tous les états est au plus* 1 *sauf pour au plus un état pour lequel il vaut* 2.

Les automates monocycles à boucle suffixe (resp. à boucle préfixe) sont les automates émondés avec un unique état initial (resp. état final) dont tous les états ont un degré sortant (resp. entrant) d'au plus 1.

Proposition 75. *Les langages rayons sont exactement les langages reconnus par les automates monocycles.*

Exemple 60. La Figure 8.1 montre deux exemples d'automates monocycles $\mathcal{A}_r$ et $\mathcal{B}_r$. L'automate $\mathcal{A}_r$ reconnaît le langage rayon $(ba)^*$ et $\mathcal{B}_r$ reconnaît $ba(ba)^*aa$.

L'automate $\mathcal{A}_r$ est à la fois à boucle préfixe et à boucle suffixe alors que $\mathcal{B}_r$ n'est ni l'un ni l'autre.



(a) L'automate $\mathcal{A}_r$ (b) L'automate $\mathcal{B}_r$

FIGURE 8.1 – Deux automates monocycles

Proposition 76. *Un automate monocycle $\mathcal{A}$ non déterministe est transformable en un automate monocycle déterministe équivalent.*

Démonstration. La seule source possible de non déterminisme dans un automate $\mathcal{A}$ monocycle est à partir de l'état q dont on sort de la boucle puisque c'est le seul à avoir un degré sortant de 2. Il est possible d'éliminer ce non déterminisme sur q en faisant rouler le circuit vers l'état final, c'est-à-dire :
(i) si q est à la fois l'entrée et la sortie du circuit et qu'il y a une transition de q étiquetée par a à la fois vers q' et q'' alors il faut fusionner q' et q'',
(ii) sinon il faut tout d'abord décaler le cycle d'un état : si q_1 est l'état d'entrée dans le cycle et q_2 sont successeur par la lettre a, alors il faut séparer q_1 en deux états : le premier appartient au cycle mais n'est plus l'état d'entrée du cycle, le deuxième est un état intermédiaire qui n'appartient pas au cycle mais dont la transition sortante va en q_2 (qui devient l'état d'entrée du cycle). Ensuite il faut 'rembobiner' le chemin de sortie : si q a une transition sortante étiqueté par a dans la boucle vers l'état q' et une

transition étiquetée par a vers un état q'' sur le chemin vers l'état final, il est possible de fusionner q' et q''. L'état q' devient alors l'état de sortie de la boucle à la place de q.

Comme le chemin de sortie est fini, si le non déterminisme subsiste, il suffit de refaire rouler le circuit un nombre fini de fois (dans le pire des cas, l'automate deviendra alors a boucle suffixe). La transformation est montrée sur la Figure 8.2.

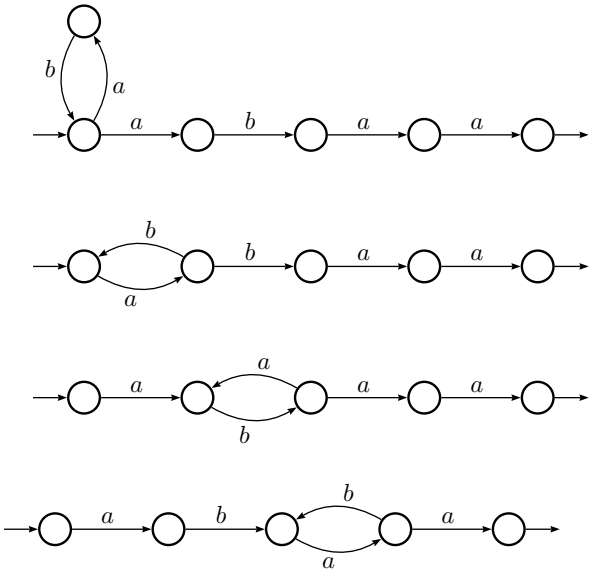


FIGURE 8.2 – Transformation d'un automate monocycle en automate monocycle déterministe

□

La transformation ci-dessus est adaptable de manière duale pour obtenir un automate monocycle co-déterministe en roulant le cycle vers l'état initial.

Produit synchronisé de deux automates

Définition 46. *Soient* $\mathcal{A} = \langle Q, A, E, I, T\rangle$ *et* $\mathcal{B} = \langle R, A, F, J, U\rangle$ *deux automates finis. Le produit synchronisé de* $\mathcal{A}$ *par* $\mathcal{B}$ *est la partie émondée du transducteur :*

$$\mathcal{A} \bowtie \mathcal{B} = \langle Q \times R, A, A, G, I \times J, T \times U\rangle \ ,$$

où l'ensemble des transitions G est défini par :

$$G = \{((p,r),(a,b),(q,s)) \mid (p,a,q) \in E, (r,b,s) \in F\} \ .$$

Si $\mathcal{A}$ et $\mathcal{B}$ sont deux automates finis sur A^* qui reconnaissent respectivement les langages L et K alors le produit synchronisé [1] $\mathcal{A} \bowtie \mathcal{B}$ réalise la relation de A^* dans A^* dont le graphe est :

$$\{(u,v) \mid u \in L, v \in K, |u| = |v|\} = L \times K \cap (A \times A)^* \ .$$

Si $\mathcal{B}$ reconnaît un langage avec au plus un seul mot par longueur alors le transducteur $\mathcal{A} \bowtie \mathcal{B}$ réalise une fonction.

Le produit synchronisé est distributif par rapport à l'union :

$$\left[\bigcup_{i\in I} \mathcal{A}_i\right] \bowtie \left[\bigcup_{j\in J} \mathcal{B}_j\right] = \bigcup_{(i,j)\in I\times J} (\mathcal{A}_i \bowtie \mathcal{B}_j) \ .$$

Le degré sortant d'un état (p,q) de $\mathcal{A} \bowtie \mathcal{B}$ est le produit des degrés sortants des états p de $\mathcal{A}$ et q de $\mathcal{B}$. De la définition du produit synchronisé on tire que, même si $\mathcal{A}$ est déterministe, le transducteur $\mathcal{A} \bowtie \mathcal{B}$ n'est pas séquentiel dès $\mathcal{B}$ a un état de degré sortant au moins 2.

Proposition 77. *Si $\mathcal{A}$ est un automate monocycle déterministe et si $\mathcal{B}$ est un automate monocycle à boucle suffixe (resp. préfixe) alors le produit synchronisé $\mathcal{A} \bowtie \mathcal{B}$ est un transducteur monocycle séquentiel (resp. co-séquentiel).*

Démonstration. Cette proposition est un corollaire direct de la Proposition 74 car $\mathcal{B}$ à boucle suffixe implique que le degré sortant de chaque état est au plus 1 et donc il n'y a qu'une transition créée dans $\mathcal{A} \bowtie \mathcal{B}$ pour chaque transition sortante de $\mathcal{A}$ par état de $\mathcal{A} \bowtie \mathcal{B}$ correspondant.

Comme $\mathcal{A}$ est déterministe, l'automate d'entrée sous-jacent à $\mathcal{A} \bowtie \mathcal{B}$ est déterministe. □

Exemple 61 (*Ex. 60 cont.*)**.** La Figure 61 montre le produit synchronisé de $\mathcal{A}_r$ par $\mathcal{B}_r$ qui n'est ni séquentiel ni co-séquentiel.

1. Le produit synchronisé défini ici diffère un peu de celui défini en [43, Exerc. IV.6.17] qui réalise la relation dont le graphe est $L \times K$.

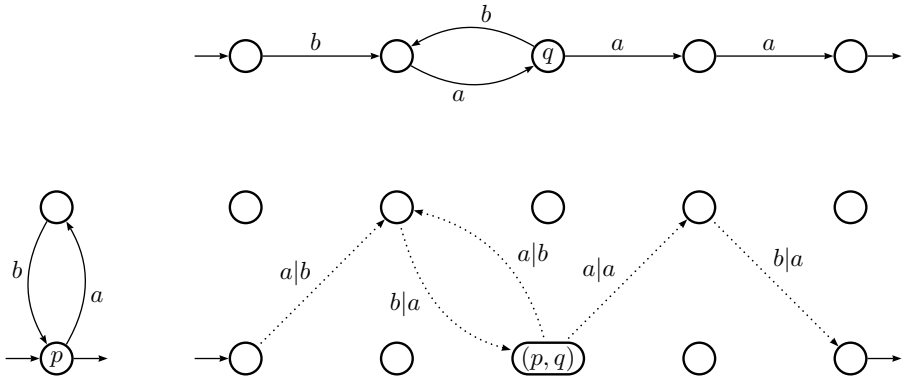


FIGURE 8.3 – Le produit synchronisé $\mathcal{A}_r \bowtie \mathcal{B}_r$

8.1.2 Séquentialité du produit de deux automates monocycles

Dans cette sous-section, nous allons montrer que la fonction réalisée par le produit synchronisé de deux automates monocycles est une fonction séquentielle et co-séquentielle.

Pour cela, nous allons modifier légèrement la définition des automates pour y ajouter des transitions spontanées et des fonctions initiale et finale :

- les transitions d'un automate sont soit étiquetées par une lettre a de A, soit par le mot vide 1_{A^*} ;
- les ensembles initiaux et finaux I et T sont remplacés par des fonctions de Q dans A^* que nous notons également I et T et dont les valeurs pour un état q seront notées I_q et T_q.

Quand ce sera nécessaire, dans cette sous-section, nous appellons *automates classiques* les automates sans transitions spontanées et sans fonctions initiales et finales. Il est connu que les automates – avec la définition étendue – reconnaissent exactement les mêmes langages que les automates classiques et que les automates classiques sont vus comme des automates en définissant une fonction initiale (resp. finale) ayant pour valeur 1_{A^*} sur les états initiaux (resp. finaux) – et $\emptyset$ sur les autres –.

Si c est un calcul réussi d'un automate $\mathcal{A}$, alors nous notons $\ell(c)$ la *longueur* de c, c'est-à-dire le nombre de transitions de c.

Cette définition élargie de la notion d'automate nous permet de montrer la proposition suivante :

Proposition 78. *Tout automate monocycle classique $\mathcal{B}$ peut être transformé en un automate monocycle $\breve{\mathcal{B}}$ tel que :*
(i) *$\breve{\mathcal{B}}$ est équivalent à $\mathcal{B}$;*
(ii) *$\breve{\mathcal{B}}$ est à boucle suffixe ;*
(iii) *si les calculs c et c' respectivement de $\mathcal{B}$ et $\breve{\mathcal{B}}$ sont étiquetés par le même mot, alors $\ell(c) = \ell(c')$.*

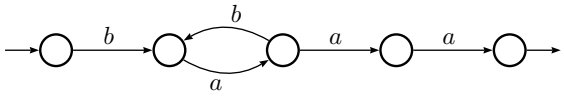
Démonstration. L'automate $\breve{\mathcal{B}}$ est obtenu à partir de l'automate monocycle $\mathcal{B}$ par les opérations suivantes :

(a) Si q est l'état de $\mathcal{B}$ par lequel on peut sortir du circuit et si w est l'étiquette du chemin qui va de q à l'état final de $\mathcal{B}$ alors on remplace ce chemin par une valeur pour la fonction finale en q de $\breve{\mathcal{B}}$ égale à w.

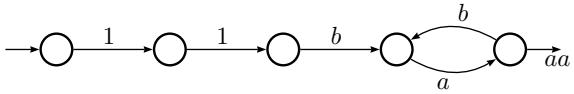
(b) Ajout avant l'état initial de $\mathcal{B}$ de $|w|$ transitions spontanées.

Par (a), l'automate $\breve{\mathcal{B}}$ est a boucle suffixe et est équivalent à $\mathcal{B}$; l'égalité des longueurs des calculs vient de l'opération (b). □

Exemple 62 (*Ex. 60 cont.*)**.** La construction appliquée à l'automate $\mathcal{B}_r$ est montrée sur la Figure 8.4.



(a) L'automate classique $\mathcal{B}_r$



(b) L'automate $\breve{\mathcal{B}}_r$

FIGURE 8.4 – Transformation d'un automate monocycle en un automate monocycle à boucle suffixe

Il existe une transformation duale qui transforme un automate monocycle en un automate monocycle à boucle préfixe.

La définition du produit synchronisé est également étendue à la nouvelle classe d'automates :

$$\mathcal{A} \bowtie \mathcal{B} = \langle Q \times R, A, A, G, I \times J, T \times U \rangle$$

où : $[I \times J]_{(p,q)} = (I_p, J_q)$, $[T \times U]_{(p,q)} = (T_p, U_q)$ et :

$$G = \{((p,r),(x,y),(q,s)) \mid (p,x,q) \in E, (r,y,s) \in F\} \ .$$

Exemple 63 (*Ex. 60 cont.*)**.** La Figure 8.5 montre le produit synchronisé de $\mathcal{A}_r$ par $\breve{\mathcal{B}}_r$.

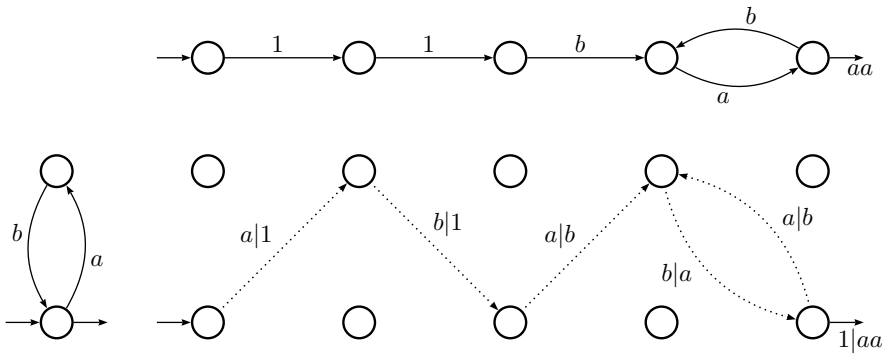


FIGURE 8.5 – Le produit synchronisé $\mathcal{A}_r \bowtie \breve{\mathcal{B}}_r$

Le graphe de la fonction réalisée par le produit synchronisé de deux automates $\mathcal{A}$ et $\mathcal{B}$ qui reconnaissent L et K n'est plus l'ensemble des couples (u, v) où u est dans L et v dans K tels que $|u| = |v|$ mais tels qu'il existe un calcul de u dans $\mathcal{A}$ de longueur égale à un calcul de v dans $\mathcal{B}$.

D'après la définition de $\breve{\mathcal{B}}$ on a donc :

Proposition 79. *Soit $\mathcal{A}$ un automate et $\mathcal{B}$ un automate monocycle. Le produit synchronisé $\mathcal{A} \bowtie \mathcal{B}$ est équivalent au produit synchronisé $\mathcal{A} \bowtie \breve{\mathcal{B}}$.*

Comme l'automate $\breve{\mathcal{B}}$ a un degré sortant d'au plus A, il apparaît que le produit synchronisé $\mathcal{A} \bowtie \breve{\mathcal{B}}$ est séquentiel lorsque $\mathcal{A}$ est déterministe.

Comme nous avons montré que la transformation $\breve{\mathcal{B}}$ est possible pour tout automate monocycle :

Théorème 80. *Le produit synchronisé $\mathcal{A} \bowtie \mathcal{B}$ d'un automate $\mathcal{A}$ déterministe avec un automate $\mathcal{B}$ monocycle réalise une fonction séquentielle.*

De la Proposition 77 nous déduisons également :

Proposition 81. *Soient $\mathcal{A}$ et $\mathcal{B}$ deux automates monocycles. Si $\mathcal{A}$ est déterministe, alors le produit synchronisé $\mathcal{A} \bowtie \breve{\mathcal{B}}$ est un transducteur monocycle séquentiel.*

Comme nous avons vu par la Proposition 76 qu'un automate monocycle $\mathcal{A}$ pouvait être transformé en un automate monocycle déterministe équivalent et comme des transformations duales existent, nous avons :

Théorème 82. *Le produit synchronisé $\mathcal{A} \bowtie \mathcal{B}$ de deux automates monocycles réalise une fonction séquentielle et co-séquentielle.*

Il faut noter que si le produit synchronisé de deux automates monocycles réalise une fonction à la fois séquentielle et co-séquentielle, c'est parce qu'il existe – et nous les construisons – un tranducteur séquentiel et un transducteur co-séquentiel qui la réalisent. Il n'y a cependant pas nécessairement de transducteur à la fois séquentiel et co-séquentiel pour réaliser cette fonction.

8.1.3 Les langages avec un mot par longueur au plus

Nous pouvons maintenant donner la preuve de la Proposition 73 :

Démonstration. Les langages L et K sont des unions finies de langages rayon et donc respectivement reconnus par des unions $\mathcal{A}_1, \ldots, \mathcal{A}_n$ et $\mathcal{B}_1, \ldots, \mathcal{B}_m$ d'automates monocycles. La fonction α est réalisée par l'union des produits synchronisés $\mathcal{A}_i \bowtie \mathcal{B}_j$.

Nous avons montré que $\breve{\mathcal{B}}_i$ est équivalent à $\mathcal{B}_i$ et donc K est également reconnu par l'union des automates monocycles $\breve{\mathcal{B}}_1, \ldots, \breve{\mathcal{B}}_m$ (Proposition 78).

Pour tout i et j, par la Proposition 81, le transducteur $\mathcal{A}_i \bowtie \breve{\mathcal{B}}_j$ est un transducteur séquentiel et donc α est une fonction séquentielle par morceaux.

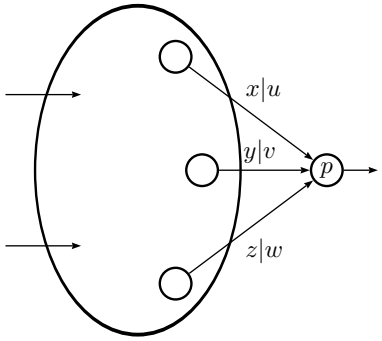
La co-séquentialité se prouve directement de manière duale. □

8.2 Concaténation de transducteurs co-séquentiels

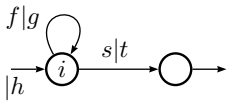
Dans cette section nous revenons à la définition classique des automates et nous allons donner des conditions sur des transducteurs co-séquentiels afin de réaliser la concaténation par un transducteur co-séquentiel. Plus précisément, et comme c'est le cas qui nous intéressera dans le développement de la preuve, nous étudions la concaténation d'un transducteur co-séquentiel avec un transducteur monocycle co-séquentiel.

Comme nous l'avons déjà montré, la concaténation de deux fonction séquentielles – ou co-séquentielles – n'est pas nécessairement séquentielle, ce n'est même pas nécessairement une fonction.

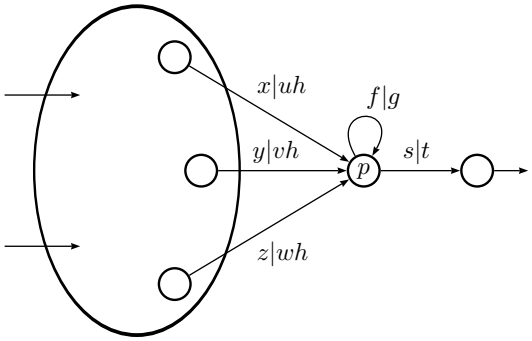
Sur la Figure 8.6 nous montrons la construction qui nous intéresse particulièrement : la concaténation d'un transducteur co-séquentiel et co-standard κ avec un transducteur monocycle co-séquentiel τ. Cette concaténation est notée $\kappa||\tau$.



(a) Le transducteur co-standard κ



(b) Le transducteur monocycle τ



(c) La concaténation $\kappa||\tau$

FIGURE 8.6 – La concaténation de deux transducteurs

Proposition 83. *Soit κ un transducteur co-séquentiel et co-standard. Si τ est un transducteur monocycle co-séquentiel, alors la concaténation $\kappa||\tau$ est un transducteur co-séquentiel si, et seulement si, on a l'une des conditions suivantes :*

(i) *τ n'a pas de circuit ;*

(ii) *τ n'est pas à boucle préfixe ;*

(iii) *τ est à boucle préfixe mais l'étiquette de l'unique transition entrante dans l'état initial de l'automate sous-jacent d'entrée de τ est différente de*

l'étiquette des transitions arrivant dans p, état final de l'automate sous-jacent de κ.

Pour comprendre cette propriété, il faut remarquer que lors de la concaténation de deux transducteurs co-séquentiels, la seule non co-séquentialité peut apparaître lors de la fusion de l'état initial i de τ et de l'état final p de κ. Il faut donc regarder les transitions entrantes dans l'état initial i de τ.

Si τ n'est pas à boucle préfixe, i a un degré entrant nul et l'affaire est entendue. Si i est sur le circuit de τ, la troisième condition assure que les étiquettes de la transition qui arrive en i et de celles qui arrivent en p sont différentes. La concaténation est alors assurée d'être co-séquentielle.

Il y a une et une seule transition entrante sur i si et seulement si i est sur le circuit de τ. Il convient alors de vérifier si le transducteur $\kappa||\tau$ est co-séquentielle en analysant les automates d'entrée sous-jacents.

8.3 La fonction successeur uniforme

Dans cette section nous définissons la fonction successeur uniforme et nous montrons que le problème de la co-séquentialité de la fonction successeur se réduit à la co-séquentialité de cette fonction uniforme.

Nous introduisons tout d'abord les langages des mots minimaux et des mots maximaux – pour chaque longueur – d'un langage L :

$$\begin{aligned}\mathsf{Min}(L) &= \{u \in L \mid \forall v \in L, |u| = |v| \Rightarrow u \preccurlyeq v\} \ , \\ \mathsf{Max}(L) &= \{u \in L \mid \forall v \in L, |u| = |v| \Rightarrow v \preccurlyeq u\} \ .\end{aligned}$$

On peut voir dans [42, 46], par exemple, que si L est rationnel alors les langages $\mathsf{Min}(L)$ et $\mathsf{Max}(L)$ sont également rationnels. Leur définition implique également que ce sont des langages avec au plus un mot par longueur.

Ces langages apparaissent utiles pour prouver la co-séquentialité de la fonction successeur car lorsque un mot u et son successeur v ont la même longueur, alors il existe $a < b$ deux lettres et u', v' deux mots tels que, si w est le plus long commun préfixe de u et v alors :

$$u = wau' \qquad \text{et} \qquad v = wbv' \ . \tag{8.1}$$

Et u est le mot maximal de longueur $|u|$ de L commençant par wa et v est le mot minimal de longueur $|u|$ de L commençant par wb. En fait v est le mot minimal de longueur $|u|$ dans L commençant par wc pour toute lettre c telle que $a < c$.

C'est pour cela que le bloc principal de la construction de notre preuve est le produit synchronisé de langages avec au plus un mot par longueur. Comme le produit synchronisé est une fonction qui préserve la longueur, et que la fonction successeur ne le fait pas, nous allons tout d' abord utiliser une restriction de la fonction successeur préservant la longueur :

Définition 47. *La fonction successeur uniforme d'un langage rationnel L, notée ULSucc_L, est la restriction de la fonction Succ_L à $(A \times A)^*$:*

$$\forall u \in L, \quad \mathsf{ULSucc}_L(u) = \mathsf{Succ}_L(u) \Leftrightarrow |u| = |\mathsf{Succ}_L(u)| \ ,$$

et si $|\mathsf{Succ}_L(u)| > |u|$ alors ULSucc_L n'est pas définie sur u.

Les mots de L pour lesquels la fonction ULSucc_L n'est pas définie sont exactement les mots dans $\mathsf{Max}(L)$ – c'est-à-dire dont le successeur est dans $\mathsf{Min}(L)$–.

De plus, si $\mathsf{Succ}_L(u) = v$, pour $u \in \mathsf{Max}(L)$, alors il n'y a aucun mot dans L de longueur l telle que $|u| < l < |v|$.

L'idée, pour réduire le problème de la fonction successeur à sa restriction uniforme, est d'introduire dans l'alphabet un nouveau caractère spécial – que nous notons \$ dans la suite –, qui est plus petit que toutes les autres lettres, et qui s'ajoute en tête de chaque mot du langage pour former le langage $K = \$^*L$. Ce langage est choisi car tout mot u de L peut être représenté par un mot de K de longueur $l \geqslant |u|$: $\$^{l-|u|}u$. En particulier le calcul de la fonction successeur uniforme sur K pour tout mot de K 'contient' le calcul de la fonction successeur sur L.

Exemple 64. La Figure 8.7 représente le langage $K_1 = \$^*L_1$ où L_1 est le langage des mots contenant un nombre pair d'occurrences de b.

Plus particulièrement, si u est dans $\mathsf{Max}(L)$, et donc son successeur v est de longueur plus grande, alors le mot $\$^{|v|-|u|}u$ de K a pour successeur v dans K et $\$^{|v|-|u|}u$ et v ont la même longueur. Cela permet de montrer que même si la fonction successeur ne préserve pas la longueur, la fonction successeur uniforme permet de retrouver tous les successeurs. En particulier, le résultat qui nous intéresse est :

Proposition 84. *Si $\mathsf{ULSucc}_{\$^*L}$ est co-séquentielle par morceaux, alors Succ_L l'est également.*

Démonstration. La fonction qui "efface" toutes les occurrences du caractère spécial \$ en début de mot est une fonction séquentielle et co-séquentielle réalisée par le transducteur de la Figure 8.8 pour un alphabet $\{a_1, \ldots, a_n\}$.

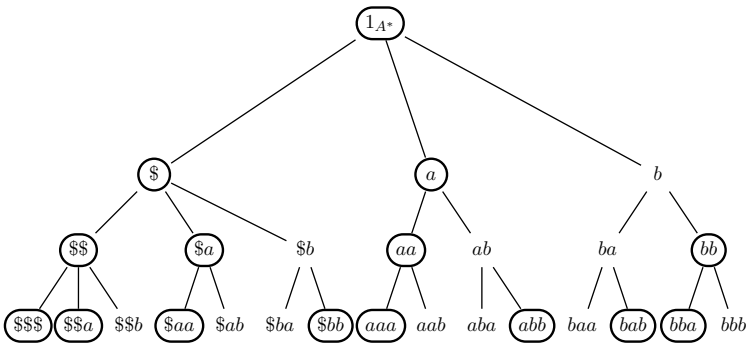


FIGURE 8.7 – L'arbre représentant K_1

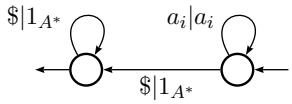


FIGURE 8.8 – Transducteur co-séquentiel qui efface les \$

Si la fonction ULSucc_L est co-séquentielle par morceaux, alors elle peut être réalisée par une cascade de transducteurs co-séquentiels de hauteur h. La cascade de transducteurs de hauteur $h+1$, formée par la cascade précédente et dont tous les transducteurs de hauteur $h+1$ sont le transducteur de la Figure 8.8, réalise la fonction successeur sur L.

Si on ajoute le transducteur de la Figure 8.8, nous obtenons une cascade de hauteur supérieure qui réalise la fonction Succ_L. La fonction Succ_L est donc co-séquentielle par morceaux. □

Remarque 14. Comme nous avons déjà montré qu'il est possible de transformer une union finie de transducteurs co-séquentiels en une cascade de transducteurs co-séquentiels de hauteur 2, la preuve ci-dessus donne une construction de la cascade réalisant Succ_L à partir de l'union finie réalisant ULSucc_L et que nous allons détailler ci-après.

Finalement, pour prouver le Théorème 72 il ne reste à montrer que le théorème suivant :

Théorème 85. *La fonction successeur uniforme* ULSucc_L *d'un langage rationnel* L *est une fonction co-séquentielle par morceaux.*

Démonstration. Nous donnons la construction d'une union finie de transducteurs co-séquentiels qui réalise ULSucc_L.

L'idée sous-jacente est que si $\mathsf{ULSucc}_L(u) = v$ alors il existe $a < b$ deux lettres de $A \cup \{\$\}$ et u', v' deux mots tels que $u = wau'$ et $v = wbv'$ où $w = u \wedge v$ et u est le mot maximal de longueur n de L commençant par wa ; v est le mot minimal de longueur n de L commençant par wc pour toute lettre c telle que $a < c$ dans L.

Soient $\mathcal{A} = \langle Q, A, \delta, i, T\rangle$, déterministe, qui reconnaît L et $q = i \cdot w$ dans $\mathcal{A}$, alors :

(i) au' est le mot maximal de longueur $|au'|$ que l'on peut lire depuis l'état q dans $\mathcal{A}$ qui commence par la lettre a,

(ii) bv' est le mot minimal de longueur $|bv'| = |au'|$ que l'on peut lire depuis l'état q dans $\mathcal{A}$ qui commence par un lettre plus grande que a.

Il s'agit donc, dans un premier temps, de réaliser la fonction qui associe bv' à au' pour chaque état q atteint. Cette fonction est réalisée par un transducteur monocycle co-séquentiel car est un produit synchronisé de langages avec au plus un mot par longueur.

Posons $L_q = \{w \in A^* \mid q \cdot w \in T\}$ le langage des mots qui sont lus à partir d'un état q dans $\mathcal{A}$ jusqu'à un état final. De même $L'_q = \{w \in A^* \mid i \cdot w = q\}$ sont les mots qui sont lus de l'état initial i jusqu'à q. Les langages L_q et L'_q sont rationnels.

Définissons maintenant $K_{q,a}$ (respectivement $H_{q,a}$) l'ensemble des mots maximaux (resp. minimaux) de L_q qui commencent par a (resp. une lettre plus grande que a) :

$$K_{q,a} = \mathsf{Max}(a(a^{-1}L_q)) \qquad \text{et} \qquad H_{q,a} = \mathsf{Min}(\bigcup_{c>a} c(c^{-1}L_q)) \ .$$

Les langages $K_{q,a}$ et $H_{q,a}$ sont rationnels et ont au plus un mot par longueur. La fonction $\alpha_{q,a}$ qui associe à chaque mot de $K_{q,a}$ le mot de $H_{q,a}$ de même longueur est donc, d'après la Proposition 73 réalisée par une union finie de transducteurs co-séquentiels monocycles.

Lemme 86. *Si* $\mathsf{ULSucc}_L(wau') = wbv'$*, avec* $a < b$ *et que* $w \in L'_q$*, alors la fonction qui associe* au' *à* bv' *est exactement la fonction* $\alpha_{q,a}$*.*

Démonstration. Nous avons déjà montré que au' est dans $K_{q,a}$ et bv' dans $H_{q,a}$ et ils ont la même longueur (par définition de la fonction successeur uniforme).

Il reste donc à montrer que pour tout $s \in K_{q,a}$ et $t \in H_{q,a}$ tels que $|s| = |t|$, tout mot $w \in L'_q$ est tel que $wt = \mathsf{ULSucc}_L(ws)$.

Comme $K_{q,a}$ et $H_{q,a}$ sont des sous-ensembles rationnels de L_q, tout mot $w \in L'_q$ est tel que ws et wt sont dans L. De plus comme s commence par a et t par une lettre plus grande que a, et comme $|ws| = |wt|$ alors $ws \prec wt$. Comme s et t ne commencent pas par la même lettre, et comme s est dans $K_{q,a}$, w est nécessairement le plus long commun préfixe de ws et de son successeur. Finalement son successeur est donc wt car t, dans $H_{q,a}$, est le mot le plus petit, de longueur $|t|$, qui soit plus grand que s et tel que wt soit dans L. □

Par la Proposition 73, la fonction $\alpha_{q,a}$ est une fonction réalisée par une union finie de transducteurs co-séquentiels monocycles que l'on peut construire. Notons $\tau_{q,1}, \ldots, \tau_{q,k}$ de tels transducteurs monocycles. Avec nos notations précédentes, cette union associe bv' à au'.

Pour réaliser la fonction qui associe wbv' à wau', if suffit de concaténer un transducteur réalisant la restriction de l'identité sur L'_q à chaque transducteur $\tau_{q,i}$.

Comme nous l'avons vu précédemment, la concaténation n'est pas nécessairement co-séquentielle. Nous allons donc essayer de réunir les conditions de la Proposition 83. Pour cela, prenons κ_q, un transducteur co-standard et co-séquentiel réalisant l'identité sur L'_q – c'est-à-dire un transducteur dont les entrées sont égales aux sorties et dont l'automate d'entrée sous-jacent est un automate co-déterministe reconnaissant L'_q –.

La Proposition 83 donne des conditions sur les $\tau_{q,i}$ pour que $\kappa_q || \tau_{q,i}$ soit co-séquentiel. Nous allons maintenant montrer que ces conditions sont soit réalisées, soit que l'on peut transformer κ_q et $\tau_{q,i}$ en des transducteurs qui réalisent les conditions :

Lemme 87. *Le transducteur $\theta_{q,i} = \kappa_q || \tau_{q,i}$ réalise une fonction co-séquentielle par morceaux.*

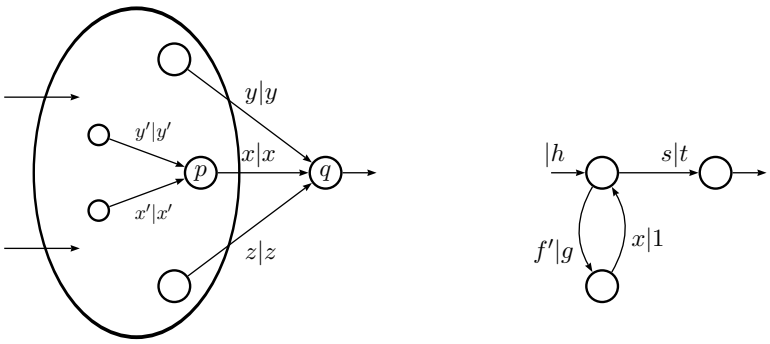
Démonstration. Le transducteur monocycle et co-séquentiel $\tau_{q,i}$ remplit l'une des conditions suivantes :

1. Soit $\tau_{q,i}$ n'a pas de circuit ; soit $\tau_{q,i}$ n'est pas à boucle préfixe ; soit $\tau_{q,i}$ est à boucle préfixe mais l'étiquette de l'unique transition entrante dans l'état initial de l'automate sous-jacent d'entrée de τ est différente de l'entrée des transitions arrivant dans p, état final de κ_q.
2. Ou bien $\tau_{q,i}$ est à boucle initial et l'étiquette d'entrée x de la transition entrante dans l'état initial est égale à l'entrée d'une transition qui va

d'un état p à l'état final q de κ_q. Nous notons $f|g$ l'étiquette du circuit de $\tau_{q,i}$ et donc il existe f tel que $f = f'x$.

Dans le premier cas, correspondant aux conditions de la Proposition 83, la concaténation $\theta_{q,i}$ est co-séquentielle.

Une représentation de $\tau_{q,i}$ et κ_q pour le deuxième cas est montrée sur la Figure 8.9. Pour que la figure soit complètement générale, il faudrait considérer le cas où f' est le mot vide. Cependant le raisonnement qui va suivre est toujours valide avec cette hypothèse.



(a) κ_q (b) $\tau_{q,i}$

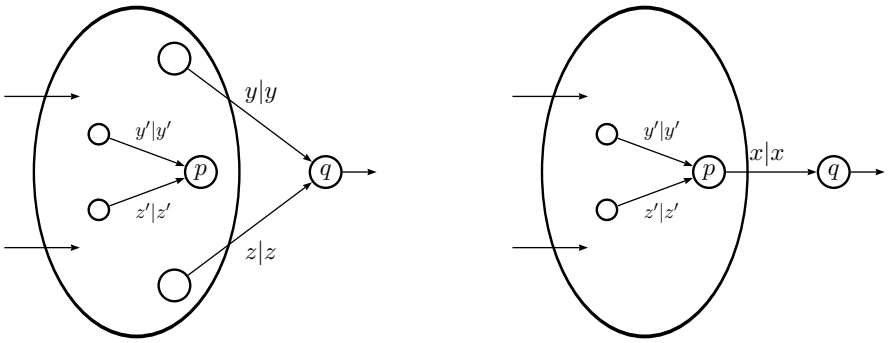
FIGURE 8.9 – κ_q and $\tau_{q,i}$

Dans ce deuxième cas, $\theta_{q,i}$ n'est pas co-séquentiel. Nous allons transformer les transducteurs κ_q et $\tau_{q,i}$ afin de construire une union finie de transducteurs co-séquentiels.

La première transformation consiste à dupliquer κ_q en deux copies $\kappa_{q,1}$ et $\kappa_{q,2}$. Dans $\kappa_{q,1}$ nous enlevons la transition entre p et q étiquetée par x. Dans le deuxièmes nous enlevons toutes les autres transitions entrantes en q – l'état final de κ_q. Cette "séparation" de κ_q en deux transducteurs co-séquentiels est montrée sur la Figure 8.10.

L'union de $\kappa_{q,1}$ et $\kappa_{q,2}$ réalise bien la même fonction identité que κ_q car ce dernier est co-déterministe et co-standard et donc chaque calcul réussi emprunte une unique transition arrivant en l'état final q. Le transducteur $\kappa_{q,1}||\tau_{q,i}$ est dans le cas 1) défini plus haut et donc co-séquentiel.

Maintenant il reste à traiter la concaténation de $\kappa_{q,2}$ avec $\tau_{q,i}$. Rappelons que la seule transition arrivant en l'état final q de $\kappa_{q,2}$ est la transition entre p et q étiquetée par x en entrée.



(a) $\kappa_{q,1}$ (b) $\kappa_{q,2}$

FIGURE 8.10 – Séparation de κ_q en deux composantes

Nous transformons $\kappa_{q,2}$ à nouveau en un transducteur κ'_q en effectuant les opération suivantes :
(i) toutes les transitions d'un état r de $\kappa_{q,2}$ à p, étiquetées par $y|y$ sont dupliquées entre r et q ;
(ii) on concatène x à la sortie de ces nouvelles transitions – l'état de cette transformation peut être vu sur la Figure 8.11 – ;
(iii) on supprime la transition de p à q.

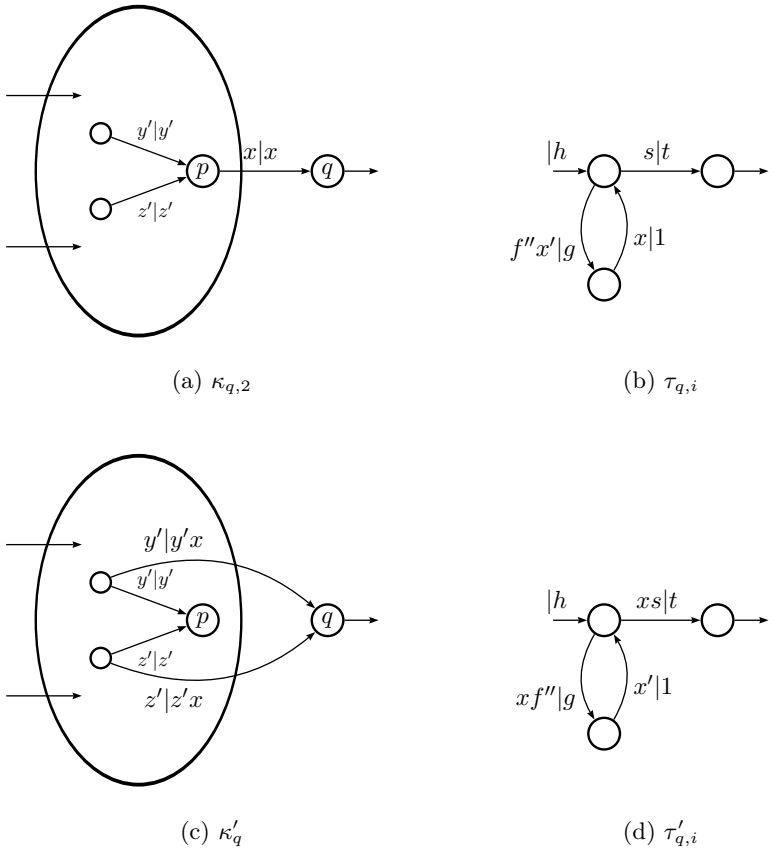
Nous transformons également $\tau_{q,i}$ en $\tau'_{q,i}$ en décalant la boucle de l'état initial. C'est à dire que l'étiquette de la boucle devient $xf'|g$ et nous ajoutons une transition $x|1_{A^*}$ entre la boucle et son chemin de sortie – voir Figure 8.11.

Montrons tout d'abord que la concaténation $\theta'_{q,i} = \kappa'_q||\tau'_{q,i}$ réalise la même fonction que $\kappa_{q,2}||\tau_{q,i}$: d'un coté, nous avons $\mathsf{Dom}\,\kappa'_q = [\mathsf{Dom}\,\kappa_{q,2}]\,x^{-1}$ et $\kappa'_q(u) = ux$, de l'autre coté : $\mathsf{Dom}\,\tau'_{q,i} = x\mathsf{Dom}\,\tau_{q,i}$ et $\tau'_{p,i}(xv) = \tau_{p,i}(v)$.

Nous avons donc finalement :

$$\begin{aligned}\left[\kappa'_q||\tau'_{q,i}\right](uxv) &= \kappa'_q(u)\,\tau'_{q,i}(xv)\\ &= \kappa_{q,2}(ux)\,\tau_{q,i}(v) = \left[\kappa_{q,2}||\tau_{q,i}\right](uxv)\quad .\end{aligned}$$

Nous pouvons remarquer que κ'_q n'a pas exactement les mêmes propriétés que κ_q puisque ses transitions qui arrivent en q ne sont plus étiquetées par une identité. Cependant cela ne joue aucun rôle dans notre construction et nous nous autorisons à faire une récurrence. Nous répétons donc les opérations et les transformations réalisées avec κ_q et $\tau_{q,i}$ avec les nouveaux transducteurs κ'_q et $\tau'_{q,i}$.



(a) $\kappa_{q,2}$ (b) $\tau_{q,i}$

(c) κ'_q (d) $\tau'_{q,i}$

FIGURE 8.11 – Transformation de $\kappa_{q,2}$ et $\tau_{q,i}$ en κ'_q and $\tau'_{q,i}$

Ce genre de transformation pourrait très bien ne pas s'arrêter (cf. Remarque 15 pour la construction d'automates κ et τ pour lesquels ces transformations seraient sans fin). Cependant dans notre cas précis, et de par la nature de $\tau_{q,i}$, cela est impossible et c'est ce que nous établissons maintenant.

Soit n le nombre d'états de $\mathcal{A}$ et $l = |f|$ la longueur de la boucle préfixe de $\tau_{q,i}$. Supposons que nous avons fait nl fois les transformations précédentes en retombant à chaque fois sur le cas 2). Nous aurions alors que f^n est un suffixe d'un mot de L'_q.

Il existe alors nécessairement une boucle dans $\mathcal{A}$ étiquetée par une puissance $k < n$ de f car $\mathcal{A}$ n'a que n états et on va de cette boucle en q en lisant une autre puissance de f. Or comme $\mathcal{A}$ est co-déterministe, cela implique que

$q \cdot f^k = q$.

Comme $\tau_{q,i}$ est une restriction de $K_{q,a} \bowtie H_{q,a}$, les mots $f^k s$ et $hg^k t$ sont des mots de L_q de même longueur. De plus la première lettre de f est a et la première lettre de h est b avec $b > a$. On déduit avec la boucle sur q que f^{2k} et $f^k hg^k t$ sont dans L_q également. Comme h commence par un b on a $f^{2k} \prec f^k hg^k t$. Or donc f^{2k} n'est pas un mot maximal commençant par a et ne devrait donc pas être dans le domaine de $\tau_{q,i}$. Nous en déduisons donc qu'il suffit de faire un nombre fini – inférieur à nl – de fois les transformations pour retomber uniquement sur le cas 1). □

Finalement, il reste à montrer que l'union des $\theta_{q,i}$ réalise exactement une restriction de ULSucc_L aux mots dont le plus grand commun préfixe avec son successeur est dans L'_q. Ceci est direct puisque κ_q réalise exactement l'identité sur L'_q et l'union des $\tau_{q,i}$ exactement la fonction dont l'image de ce que nous avions noté au' est bv'.

□

Remarque 15. Il est possible de construire des transducteurs κ et τ simples, tels que κ soit co-séquentiel et co-standard et τ soit un transducteur monocycle, pour lesquels la construction définie dans la preuve du Lemme 87 est sans fin.

En effet, si l'on prend les transducteurs de la Figure 8.12, la construction est sans fin puisque κ est invariant et que la boucle de τ reste étiquetée par $a|a$.

(a) tranducteur κ (b) transducteur τ

FIGURE 8.12 – Transducteurs pour lesquels la construction est sans fin

Remarque 16. Enfin il faut préciser que toutes les constructions pour la preuve sont symétriques sauf une. En particulier les constructions de produits synchronisés et de concaténations sont compatible avec le passage à la dualité. On pourrait donc s'attendre à pouvoir également construire de manière duale une fonction séquentielle par morceaux. Ce n'est pas le cas car, alors qu'il est possible de choisir un automate déterministe et standard – co-déterministe et co-standard dans notre preuve – reconnaissant un langage L,

il n'est pas possible, en général, de construire un automate déterministe et co-standard – ce qui serait nécessaire pour adapter cette preuve et montrer la séquentialité par morceaux.

En outre, nous avons déjà montré dans le chapitre précédent que la fonction successeur d'un langage rationnel n'est pas, en général, séquentielle par morceaux.

Chapitre 9

Séries énumératrices des langages rationnels

Dans ce chapitre, nous décrivons une autre manière de représenter l'énumération radicielle d'un langage : *la série énumératrice*.

Définition 48 ([4])**.** *Soit $\mathcal{S} = (L, A, <)$ un système de numération abstrait, la série énumératrice de $\mathcal{S}$ est la $\mathbb{N}$-série sur A^*, que l'on note $\mathbf{E}_{\mathcal{S}}$, définie par :*

$$\mathbf{E}_{\mathcal{S}} = \sum_{w \in L} \left(\pi_{\mathcal{S}}\left(w\right) + 1\right) w \quad .$$

La notation précédente peut être simplifiée en $\mathbf{E}_L = \sum_{w \in L} \left(\pi_L\left(w\right) + 1\right) w$.

La série énumératrice d'un langage L est donc la série formelle entière dont le support est L et qui associe à chaque mot u de L le coefficient qui correspond au rang de u lorsque L est ordonné dans l'ordre radiciel.

Remarque 17. Il pourrait sembler, au premier abord, qu'il est lourd de prendre $\pi_L\left(w\right) + 1$ comme définition du coefficient de w plutot que $\pi_L\left(w\right)$, plus naturel. Cependant cette définition est prise afin que la série énumératrice détermine le langage L. En effet, si w est le premier mot de L dans l'ordre radiciel, alors $\pi_L\left(w\right) = 0$ et il serait impossible de distinguer le mot w de tout mot qui n'est pas dans L. En particulier, la définition que nous avons donné de la série énumératrice donne :

$$L = \mathsf{supp}\left(\mathbf{E}_L\right) \quad .$$

Dans cette section nous étudions la rationalité de la série énumératrice d'un langage L rationnel qui a été établie pour la première fois dans [20]. Nous en donnons une preuve qui permet de donner une construction d'une représentation de la série énumératrice. Les calculs pour cette contruction sont ceux de [29] dans lequel la rationalité de la série énumératrice de L n'est pas remarquée.

De la rationalité découlent immédiatement des résultats déjà établis sur les systèmes de numération abstraits. Plus encore, la preuve de la rationalité par le calcul d'une représentation permet de faire les calculs très simplement. Nous montrons ainsi comment la représentation de la série énumératrice permet de calculer la valeur d'un mot dans un système de numération abstrait défini par un langage rationnel L. Ce problème avait été étudié auparavant dans [30].

Nous développons également comment la rationalité de la série énumératrice d'un langage rationnel permet de résoudre le problème de la reconnaissabilité des représentations, dans un système de numération abstrait rationnel, d'un ensemblre reconnaissable d'entier. Ce problème est résolu dans [30] et dans [29].

Finalement, nous essayons d'avoir une vue d'ensemble sur les séries rationnelles qui sont des séries énumératrices et nous démontrons qu'il est décidable de savoir si une série donnée est une série énumératrice d'un langage ou non ([4]).

9.1 Rationalité

Dans cette section nous montrons le théorème suivant, ainsi que des applications directes de son énoncé ou de la preuve que nous allons en donner :

Théorème 88 ([20])**.** *La série énumératrice d'un langage rationnel est une série* $\mathbb{N}$*-rationnelle.*

La preuve de [20] construit une représentation pour une relation rationnelle non-ambiguë associant à tout mot u les mots v qui sont plus grands que lui dans l'ordre radiciel. L'utilisation du théorème suivant, sur l'application de relations rationnelles aux langages rationnels et son extension aux séries rationnelles (Proposition 90), permet alors de construire une représentation pour la série énumératrice.

Théorème 89 ([35])**.** *L'image d'un langage rationnel par une relation rationnelle est un langage rationnel.*

L'application de ce dernier théorème aux séries rationnelles – voir [43] – :

Proposition 90. *Soit* $\varphi\colon A^* \to B^*$ *une relation rationnelle non-ambiguë et soit* s *une série* $\mathbb{K}$*-rationnelle sur* A^**. La série*

$$\underline{\varphi}(s) = \sum_{w \in A^*} \langle s, w \rangle \underline{\varphi}(w) = \sum_{u \in B^*} \langle s, \underline{\varphi^{-1}}(u) \rangle u \ , \tag{9.1}$$

si elle est définie, est une série $\mathbb{K}$*-rationnelle sur* B^**.*

Le résultat prouvé dans [20] est plus général que celui que nous exposons ici et s'applique à d'autres séries que la série énumératrice. En particulier il est possible d'avoir le même résultat pour d'autres ordres sur les mots d'un langage. Par exemple, la série qui donne le rang des mots pour l'ordre lexicographique, ou encore celle qui compte le nombre de préfixes d'un mot x dans le langage L sont également rationnelles.

Le résultat est également plus fort dans le sens où il permet également d'étendre le résultat aux langages algébriques : lorsque le langage est algébrique, alors la série énumératrice – et toutes les autres séries traitées dans [20] – de ce langage est également algébrique.

Une autre preuve du Théorème 88 peut être trouvée dans [39] à travers l'utilisation de la caractérisation de séries reconnaissables comme étant celles qui appartiennent à sous-module stable finiment engendré des séries formelles. Cependant, et contrairement à la preuve de [20], cette preuve ne donne aucun moyen de calculer une représentation de $\mathbf{E}_L$.

Finalement ce résultat est retrouvé, sans y être explicitement énoncé, dans [29] par la construction d'une $\mathbb{N}$-représentation de la série énumératrice d'un langage rationnel. Nous détaillons cette construction qui utilise, en particulier, une décomposition de l'ensemble des mots plus petits qu'un mot u donné. Nous donnons également quelques résultats, déjà connus, que la rationalité de la série énumératrice permet de rendre directs, par la définition ou par des résultats classiques des séries rationnelles. En particulier nous développons des méthodes constructives pour le calcul de la valeur d'un mot dans un système de numération abstrait, ou bien le calcul d'un automate permettant de reconnaître un ensemble reconnaissable d'entiers dans un système de numération abstrait.

9.1.1 Représentation de la série énumératrice

Afin de construire une représentation pour la série énumératrice d'un langage rationnel L, nous utilisons les notations suivantes : si a est une

lettre de l'alphabet A, alors A_a est l'ensemble des lettres de A plus petites que a :

$$A_a = \{b \in A \mid b < a\} \ .$$

Si u est un mot de A^*, alors nous notons $P(u)$ l'ensemble des mots de A^* strictement plus petits que u dans l'ordre radiciel :

$$P(u) = \{v \in A^* \mid v \prec u\} \ .$$

La série énumératrice est naturellement lié à l'ensemble $P(u)$ par :

$$\forall u \in L \qquad \pi_L(u) = \mathrm{card}\,(P(u) \cap L) \ .$$

L'ensemble $P(u\,a)$ peut être décomposé en plusieurs éléments : tout d'abord le mot vide est dans $P(u\,a)$, ensuite tous les mots vb, où v est un mot plus petit que u et b est n'importe quelle lettre, sont également dans $P(u\,a)$, enfin, tous les mots uc, où c est une lettre plus petite que a, sont dans $P(u\,a)$.

De cette décomposition, on obtient une définition inductive de $P(u)$:

Lemme 91 ([29])**.**

$$\forall u \in A^*\,,\ \forall a \in A \qquad P(u\,a) = 1_{A^*} \cup uA_a \cup P(u)A$$

Soit (λ, μ, ν) une $\mathbb{N}$-représentation d'un automate fini non-ambigu, de dimension k, reconnaissant L :

$$\forall w \in A^* \qquad \lambda \cdot \mu(w) \cdot \nu = 1 \quad \Longleftrightarrow \quad w \in L \ .$$

Nous utilisons la notation suivante : si K est un ensemble fini de A^* alors :

$$\mu(K) = \sum_{w \in K} \mu(w) \ .$$

Comme (λ, μ, ν) est une représentation non-ambiguë finie, nous avons :

$$\forall K \subseteq A^* \qquad \lambda \cdot \mu(K) \cdot \nu = \sum_{w \in K} \lambda \cdot \mu(w) \cdot \nu = \mathrm{card}\,(K \cap L) \ . \tag{9.2}$$

En particulier nous avons donc :

$$\forall u \in A^* \qquad \lambda \cdot \mu(P(u)) \cdot \nu = \mathrm{card}\,(\{v \in A^* \mid v \in L \quad \text{et} \quad v \prec u\}) \ .$$

et donc :

$$\forall w \in L \qquad \lambda \cdot \mu(P(w)) \cdot \nu = \pi_L(w) \ .$$

Du Lemme 91 nous avons – voir Lemme 2 de [29] – :

$$\forall u \in A^* \,,\ \forall a \in A$$
$$\lambda \cdot \mu\left(P(u\,a)\right) \cdot \nu = \lambda \cdot \mu\left(1_{A^*}\right) \cdot \nu + \lambda \cdot \mu\left(u\right) \cdot \mu\left(A_a\right) \cdot \nu + \lambda \cdot \mu\left(P(u)\right) \cdot \mu\left(A\right) \cdot \nu \quad . \tag{9.3}$$

Soit $\sigma = \mu\left(A\right)$ et, pour toute lettre a de A, $\sigma_a = \mu\left(A_a\right)$.

L'équation (9.3) peut se réécrire :

$$\forall u \in A^* \,,\ \forall a \in A$$
$$\lambda \cdot \mu\left(P(u\,a)\right) \cdot \nu = \lambda \cdot \nu + \lambda \cdot \mu\left(u\right) \cdot \sigma_a \cdot \nu + \lambda \cdot \mu\left(P(u)\right) \cdot \sigma \cdot \nu \quad . \tag{9.4}$$

Choisissons maintenant (η, κ, ζ) la représentation de dimension $2k+1$ décrite par la décomposition par blocs carrés de tailles $(1, k, k)$ ci-dessous :

$$\eta = \begin{pmatrix} 1 & \lambda & 0 \end{pmatrix} \,, \quad \forall a \in A \quad \kappa\left(a\right) = \begin{pmatrix} 1 & 0 & \lambda \\ 0 & \mu\left(a\right) & \sigma_a \\ 0 & 0 & \sigma \end{pmatrix} \,, \quad \zeta = \begin{pmatrix} 0 \\ 0 \\ \nu \end{pmatrix} \quad .$$

D'après le Lemme 91, par induction sur la longueur de u de A^*, nous avons :

$$\lambda \cdot \mu\left(P(u)\right) \cdot \nu = \eta \cdot \kappa\left(u\right) \cdot \zeta \quad .$$

Soit $\xi = \begin{pmatrix} 1 \\ 0 \\ \nu \end{pmatrix}$ et s la série réalisée par la représentation (η, κ, ξ) :

$$\forall u \in A^* \qquad <s, u> = 1 + \mathrm{card}\left(\{v \in A^* \mid v \in L \quad \text{and} \quad v \prec u\}\right) \quad .$$

C'est-à-dire que s est la série formelle dont le support est A^* et qui associe à tout mot u le nombre de mots de L plus petits que u.

Pour avoir la série énumératrice de L il faut donc prendre la restriction de la série s au langage L. Pour cela nous faisons le produit de Hadamard de s par la série caractéristique $\underline{L}$ de L :

$$\mathbf{E}_L = s \odot \underline{L} \quad ,$$

Comme la représentation (η, κ, ξ) de s est de dimension $2k+1$, et comme la dimension de la représentation la série caractéristique de L est de dimension k, il est possible de construire une représentation de taille $2k^2 + k$ de la série énumératrice $\mathbf{E}_L$. Nous verrons par la suite que, pour les calculs, il n'est pas nécessaire d'expliciter cette dernière représentation car il est possible d'utiliser la représentation (η, κ, ξ).

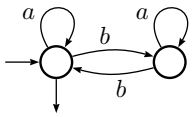


FIGURE 9.1 – Un automate déterministe reconnaissant les mots avec un nombre pair de b

Exemple 65 (*Ex. 54 cont.*)**.** Nous considérons toujours le langage L_1 des mots avec un nombre pair de b. Sont automate est remontré en Figure 9.1.

La représentation (λ, μ, ν) de cet automate est :

$$\lambda = \begin{pmatrix} 1 & 0 \end{pmatrix} , \quad \mu = \begin{pmatrix} a & b \\ b & a \end{pmatrix} , \quad \nu = \begin{pmatrix} 1 \\ 0 \end{pmatrix} .$$

Nous avons alors :

$$\sigma_a = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} , \quad \sigma_b = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} , \quad \sigma = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} .$$

La décomposition (η, κ, ξ) qui en découle est donc de dimension 5 :

$$\eta = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \end{pmatrix} , \quad \zeta = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} ,$$

et :

$$\kappa(a) = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} , \quad \kappa(b) = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} .$$

Et donc l'automate de la Figure 9.2 est un $\mathbb{N}$-automate qui renvoie, pour un mot donné de A^*, le nombre de mots, plus un, qui sont dans L_1 et qui sont plus petits que lui.

9.1.2 Calcul de la valeur d'un mot

La description par une $\mathbb{N}$-représentation d'une série $\mathbb{N}$-rationnelle donne directement un moyen de calculer les coefficients des mots dans s. La construction de la représentation précédente résout donc, de fait, le problème,

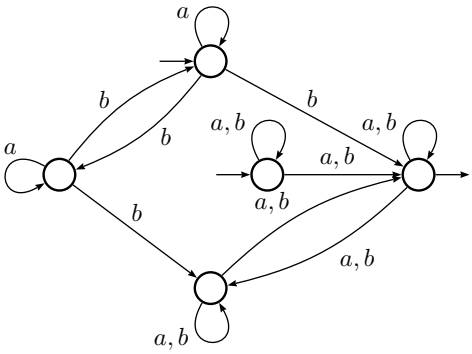


FIGURE 9.2 – $\mathbb{N}$-automate de représentation (η, κ, ξ)

déjà résolu dans [30] par une approche combinatoire similaire au Lemme 91, du calcul de la valeur $\pi_L(w)$ d'un mot w dans le système de numération abstrait rationnel défini par L.

Si une série s a une représentation (χ, ω, ϕ) de dimension n, et si w est un mot de longueur ℓ, alors la méthode générale consiste à calculer, pour i de 0 à $l-1$:

$$\chi \cdot \omega(w_{i+1}) = (\chi \cdot \omega(w_i)) \cdot \omega(a_{i+1}) \ ,$$

où a_i est la i-ième lettre de w et w_i son préfixe de longueur i. Chaque étape de cette méthode coûte $2n^2$ opérations, c'est-à-dire au total environ $2ln^2$ opérations.

Pour la série énumératrice, nous avons construit dans la sous-section précédente une représentation de dimension $2k^2 + k$. Afin de réduire l'ordre de grandeur du nombre d'opérations à effectuer, nous n'appliquons pas la méthode générale directement à cette représentation.

En effet, tout d'abord, nous avons vu que $\mathbf{E}_L = s \odot \underline{L}$, et nous avons en donnée une représentation (λ, μ, ν), de dimension k, de la série $\underline{L}$. Il est donc possible, pour calculer $\pi_L(w)$, de faire le produit de $\langle s, w \rangle$ et $\langle \underline{L}, w \rangle$ – c'est-à-dire successivement le calcul de $\langle s, w \rangle$ et de $\langle \underline{L}, w \rangle$ –. Avec la représentation (η, κ, ξ), de dimension $2k+1$, cela permet d'économiser un exposant carré.

Il est néanmoins possible d'aller encore plus loin concernant le calcul de $\langle s, w \rangle$. En effet la représentation (η, κ, ξ) par blocs de taille k (et 1) permet de réduire le problème à des calculs sur des vecteurs et des matrices de dimension k. Nous allons expliciter ces calculs.

Nous reprenons (λ, μ, ν) une représentation de dimension k d'un automate $\mathcal{A}$ non-ambigu reconnaissant L. Pour tout mot w de A^*, nous calculons deux vecteurs ligne $\alpha(w)$ et $\gamma(w)$ de dimension k. Le vecteur $\alpha(w)$ prend ses valeurs dans $\{0, 1\}$ et $\gamma(w)$ est dans $\mathbb{N}^k$. Ils sont définis par induction sur la

longueur ℓ de w par :

$$\alpha(1_{A^*}) = \lambda\,, \quad \beta(1_{A^*}) = \lambda\,, \quad \text{et} \quad \gamma(1_{A^*}) = 0 \ ,$$

et pour tout $0 \leqslant i < \ell$:

$$\alpha(w_{i+1}) = \alpha(w_i) \cdot \mu\,(a_{i+1}) \ , \quad \beta(w_{i+1}) = \alpha(w_i) \cdot \sigma_{a_{i+1}} \ , \\ \text{et} \quad \gamma(w_{i+1}) = \lambda + \beta(w_{i+1}) + \gamma(w_i) \cdot \sigma \ .$$

Le vecteur $\alpha(w)$ correspond aux chemins dans l'automate $\mathcal{A}$, depuis un état initial, étiquetés par w ; c'est-à-dire que l'entrée j de $\alpha(w)$ correspond au nombre de chemins étiquetés par w qui arrivent en un état q_j. Comme $\mathcal{A}$ est non-ambigu, les valeurs de $\alpha(w)$ sont bien dans $\{0, 1\}$.

Le vecteur $\beta(w_{i+1})$ correspond au nombre de chemins –donc de mots par non-ambiguïté – étiquetés par un mot $w_i b$, où $b < a_{i+1}$, et qui arrivent dans chaque état.

Enfin le vecteur $\gamma(w)$ correspond donc, d'après le Lemme 91, au nombre de chemins étiquetés par des mots plus petits que w arrivant en chaque état.

Finalement nous avons $\pi_{\underline{L}}(w) = \gamma(w) \cdot \nu$ car $\alpha(w) \cdot \nu = 1$ est exactement le calcul de la valeur $<\underline{L}, w>$. Cette méthode coûte environ $6\,\ell\,k^2$ opérations.

Exemple 66 (*Ex. 54 cont.*)**.** Nous considérons toujours le langage L_1 des mots avec un nombre pair de b.

Par exemple, le calcul de $\pi_{\underline{L_1}}(bbabb)$ s'effectue avec les calculs suivants :

i	a_i	α_i	β_i	γ_i
0		$(1,0)$	$(1,0)$	$(0,0)$
1	b	$(0,1)$	$(1,0)$	$(2,0)$
2	b	$(1,0)$	$(0,1)$	$(3,3)$

i	a_i	α_i	β_i	γ_i
3	a	$(1,0)$	$(0,0)$	$(7,6)$
4	b	$(0,1)$	$(1,0)$	$(15,13)$
5	b	$(1,0)$	$(0,1)$	$(29,29)$

Et finalement, $\pi_{\underline{L_1}}(bbabb) = (29,29) \cdot \nu_1 = 29$.

9.1.3 Sous-ensembles reconnaissables d'entiers

Une autre conséquence de la rationalité de la série énumératrice d'un langage rationnel est la rationalité de tout langages dont les coefficients forment un ensemble reconnaissable d'entiers (voir par exemple [8, Corol.III.2.4], [22, Th. VI.10.1] ou [43, Corol. III.4.21]) :

Théorème 92. *Soit s une série $\mathbb{N}$-rationnelle, pour tout ensemble reconnaissable X d'entiers, le langage des mots dont le coefficient dans la série s est dans X est rationnel.*

Et donc en particulier :

Corollaire 93 ([30]). *Un ensemble reconnaissable d'entiers est L-reconnaissable dans tout système de numération abstrait L.*

Si ce dernier corollaire est une conséquence directe du théorème précédent, il peut néanmoins être intéressant de donner une méthode pour calculer $\langle X \rangle_L$, où X est un ensemble reconnaissable d'entiers. En particulier en réutilisant la méthode de calcul utilisée dans la sous-section précédente pour calculer la valeur de chaque mot, nous obtenons la proposition suivante :

Soit L un langage rationnel sur A^* reconnu par un automate déterministe de dimension k. Pour tous entiers p et $r < p$, l'ensemble reconnaissable des entiers congruents à r modulo p est noté $X_{p,r} = p\mathbb{N} + r$. Le langage $\langle X_{p,r} \rangle_L$ est les langage des mots dont la valeur – pour le système de numération défini par L – est dans $X_{p,r}$.

Proposition 94 ([29]). *Soit L un langage rationnel et p et r des entiers tels que $r < p$, le langage $\langle X_{p,r} \rangle_L$ est reconnu par un automate déterministe de dimension au plus kp^k.*

Démonstration. Soit $\mathcal{A}$ un automate déterministe de dimension k reconnaissant L et soit (λ, μ, ν) sa $\mathbb{N}$-représentation associée. Notons Q l'ensemble des états de $\mathcal{A}$. Comme $\mathcal{A}$ est déterministe, λ and μ sont monomiaux en ligne et par conséquent, il en est de même de $\alpha(w)$. C'est-à-dire que $\alpha(w)$ a une seule – exactement si $\mathcal{A}$ est complet – coordonnée j non nulle. Il est donc possible de représenter $\alpha(w)$ par q_j, l'état de Q correspondant à la coordonnée j.

Notons $\delta(w) = \gamma(w) \mod p$ le vecteur ligne de dimension k dont les valeurs sont celles de $\gamma(w)$ modulo p. Ce vecteur prend un nombre fini de valeurs différentes – plus précisément p^k –.

L'idée de la preuve est de construire un automate dont chaque état est tel que les mots qui y parviennent sont les mots correspondant à une valeur fixée du couple $\big(\alpha(w), \delta(w)\big)$. L'ensemble des états de l'automate $\mathcal{B}$ que l'on construit est donc :

$$R = \{\big(\alpha(w), \delta(w)\big) \mid w \in A^*\} \ .$$

C'est-à-dire que les états de $\mathcal{B}$ sont les couples (q_j, δ_m) où q_j est un état de $\mathcal{A}$ et δ_m un vecteur ligne de dimension k à valeurs inférieures strictement à p.

Comme $\alpha(wa)$ et $\gamma(wa)$ dépendent de $\alpha(w)$ et $\gamma(w)$ mais pas de w précisément, on peut noter les transitions de $\mathcal{B}$ de la manière suivante :

$$\forall a \in A \qquad \big(q_j, \delta_m\big) \xrightarrow[\mathcal{B}]{a} \big(q_{j'}, \delta_{m'}\big) \ ,$$

si il existe w tel que $\alpha(w)$ représente q_j et $\delta(w) = \delta_m$ et tel que $\alpha(wa)$ représente $q_{j'}$ et $\delta(wa) = \delta_{m'}$. L'état initial de $\mathcal{B}$ est $(\lambda, 0)$ et les états finaux sont les états où q_j est final et $\delta_m \cdot \nu \mod p = r$.

L'automate $\mathcal{B}$ est déterministe, a kp^k états au plus, et reconnaît le langage $\langle X_{p,r} \rangle_L$, car en effet, chaque chemin de $\mathcal{A}$ se relève en un unique chemin de $\mathcal{B}$. □

Exemple 67 (*Ex. 54 cont.*)**.** La Figure 9.3 montre l'automate qui reconnaît l'ensemble $3\mathbb{N} + 1$ construit à partir de l'automate $\mathcal{A}_1$ (à droite). Les vecteurs γ correspondant à chaque état sont notés sur ceux-ci. La composante α se retrouve en projetant sur l'automate $\mathcal{A}_1$.

Cet automate n'est pas minimal – son quotient minimal a 8 états (voir Figure 9.4) – la méthode décrite ci-dessus ne donne donc pas un automate minimal.

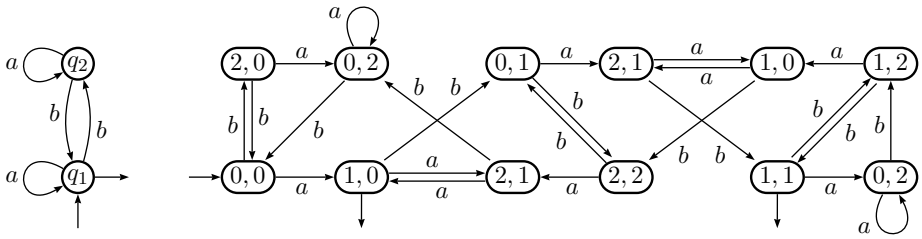


FIGURE 9.3 – Automate déterministe reconnaissant $3\mathbb{N} + 1$ dans L_1.

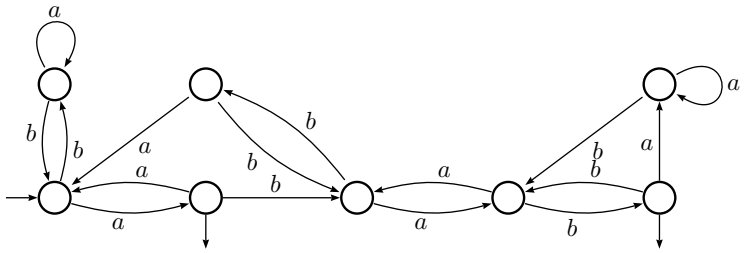


FIGURE 9.4 – Quotient minimal

Remarque 18. Comme toutes les constructions précédentes étaient valables à partir non seulement d'automates déterministes, mais plus généralement non-ambigus, cette construction est également possible à partir d'un automate non-ambigu et l'automate ainsi calculé a au plus $2^k p^k$ états.

Remarque 19. Une méthode différente pour calculer un automate reconnaissant $\langle X_{p,r} \rangle_L$ est donnée dans [9, Th. 3.3.1]. L'automate construit par cette méthode est co-déterministe mais pas déterministe et a, environ, $k\, p^{k+1}$

états. Comme l'automate obtenu est co-déterministe, sa déterminisation donne l'automate minimal de $\langle X_{p,r} \rangle_L$. La forme particulière de cet automate permet donc une déterminisation sans explosion exponentielle – puisque la Proposition 94 majore le nombre d'états de l'automate minimal –.

9.2 Décidabilité

De par la nature croissante des coefficients d'un série énumératrice, il est évident que toutes les séries rationnelles ne sont pas des séries énumératrices d'un certain système de numération abstrait rationnel.

Avant de montrer la décidabilité de l'appartenance d'une série à l'ensemble des séries énumératrices de langages rationnels, nous rappelons un résultat de décidabilité pour les séries rationnelles :

Théorème 95 ([22])**.** *Si $\mathbb{K}$ est un sous-semi-anneau d'un corps, alors il est décidable si une série $\mathbb{K}$-rationnelle s est la série nulle ou non.*

Théorème 96 ([4])**.** *Il est décidable si une série rationnelle s est la série énumératrice d'un système de numération abstrait rationnel L ou non.*

Démonstration. Une série formelle est une série énumératrice d'un système de numération rationnel L si et seulement si les trois conditions suivantes sont réalisées :
(i) son support $\mathsf{supp}\ s$ est rationnel ;
(ii) le mot w_0 le plus petit du langage pour l'ordre radiciel a pour coefficient 1 ;
(iii) la différence de coefficients entre deux mots successifs est exactement 1, c'est-à-dire que :

$$<s, \mathsf{Succ}_L(u)> - <s, u> = 1 \ .$$

Comme nous avons déjà vu, la fonction successeur est une fonction rationnelle, en particulier nous avons même montré au chapitre précédent que c'est une fonction séquentielle par morceaux. Succ_L est donc une relation rationnelle non-ambiguë et, d'après la Proposition 90, la série

$$\underline{\mathsf{Succ}_L}(s) = \sum_w <s, w> \mathsf{Succ}_L(w)$$

est une série $\mathbb{N}$-rationnelle et, donc, la série $t = s - \underline{\mathsf{Succ}_L}(s)$ est $\mathbb{Z}$-rationnelle.

La série s est série énumératrice si et seulement t est telle que $<t, w> = 1$ pour tous les mots de $L \setminus \{w_0\}$, c'est-à-dire si $t - \underline{L \setminus \{w_0\}} = 0$. D'après le

Théorème 95, et comme $\mathbb{Z}$ est évidemment un sous-anneau d'un corps, le problème est décidable. □

Bibliographie

[1] C. Allauzen and M. Mohri, A unified construction of the Glushkov, Follow, and Antimirov automata, in : *Proc. MFCS 2006*, R. Kralovic and P. Urzyczyn (ed.), *Lecture Notes in Computer Science* n° 4162, Springer, 2006, 110–121.

[2] P.-Y. Angrand, S. Lombardy and J. Sakarovitch, On the number of broken derived terms of a rational expression, *J. Automata, Languages and Combinatorics*, vol. 15,1/2 (2010), 27–51.

[3] P.-Y. Angrand and J. Sakarovitch, Radix enumeration of rational languages, *RAIRO Theor. Informatics and Appl.*, vol. 44 (2010), 19–36.

[4] P.-Y. Angrand and J. Sakarovitch, On the enumerating series of an abstract numeration system, *CoRR*, vol. abs/1108.5711 (2011).

[5] V. Antimirov, Partial derivatives of regular expressions and finite automaton constructions, *Theoret. Computer Sci.*, vol. 155 (1996), 291–319.

[6] M.-P. Béal, O. Carton, C. Prieur and J. Sakarovitch, Squaring transducers, *Theoret. Computer Sci.*, vol. 292 (2003), 45–63.

[7] J. Berstel, *Transductions and Context-Free Languages*, Teubner, 1979.

[8] J. Berstel and C. Reutenauer, *Les séries rationnelles et leurs langages*, Masson, 1984. Translation : *Rational Series and Their Languages*. Springer, 1986.

[9] V. Berthé and M. Rigo, *Combinatorics, Automata and Number Theory*, Cambridge University Press, 2010.

[10] A. Brüggemann-Klein, Regular expressions into finite automata, *Theoret. Computer Sci.*, vol. 120 (1993), 197–213.

[11] J. A. Brzozowski, Derivatives of regular expressions, *J. Assoc. Comput. Mach.*, vol. 11 (1964), 481–494.

[12] P. Caron and M. Flouret, Glushkov construction for multiplicities, in : *Proc. CIAA 2000*, A. Paun and S. Yu (ed.), *Lecture Notes in Comput. Sci.* n° 2088, 2001, 67–79.

[13] P. Caron and D. Ziadi, Characterization of Glushkov automata, *Theor. Comput. Sci.*, vol. 233,1-2 (2000), 75–90.

[14] J.-M. Champarnaud, É. Laugerotte, F. Ouardi and D. Ziadi, From regular weighted expressions to finite automata, *Int. J. Found. Comput. Sci.*, vol. 15,5 (2004), 687–700.

[15] J.-M. Champarnaud, F. Ouardi and D. Ziadi, An Efficient Computation of the Equation $\mathbb{K}$-automaton of a Regular $\mathbb{K}$-expression, *Fundam. Inform.*, vol. 90,1-2 (2009), 1–16.

[16] J.-M. Champarnaud and D. Ziadi, Computing the equation automaton of a regular expression in space and time, in : *Proc. CPM 2001*, A. Amir and G. M. Landau (ed.), *Lecture Notes in Comput. Sci.* n° 2089, Springer, 2001, 157–168.

[17] J.-M. Champarnaud and D. Ziadi, From C-Continuations to New Quadratic Algorithms for Automaton Synthesis, *Int. J. of Algebra and Computation*, vol. 11,6 (2001), 707–736.

[18] J.-M. Champarnaud and D. Ziadi, Canonical derivatives, partial derivatives and finite automaton constructions, *Theoret. Computer Sci.*, vol. 289 (2002), 137–163.

[19] C. Choffrut, Une caractérisation des fonctions séquentielles et des fonctions sous-séquentielles en tant que relations rationnelles, *Theoret. Computer Sci.*, vol. 5 (1977), 325–337.

[20] C. Choffrut and M. Goldwurm, Rational Transductions and Complexity of Counting Problems, *Math. Sci. Theory*, vol. 28 (1995), 437–450.

[21] C. Choffrut and M. P. Schützenberger, Décomposition de fonctions rationnelles, in : *Proc. STACS 1986*, B. Monien and G. Vidal-Naquet (ed.), *Lecture Notes in Comput. Sci.* n° 210, Springer, 1986, 213–226.

[22] S. Eilenberg, *Automata, Languages and Machines*, vol. A, Academic Press, 1974.

[23] C. Frougny, Fibonacci representations and finite automata, *I. E. E. E. Trans. Inform. Theory*, vol. 37 (1991), 393–399.

[24] C. Frougny, Representation of numbers and finite automata, *Math. Systems Theory*, vol. 25 (1992), 37–60.

[25] C. Frougny, On the sequentiality of the successor function, *Inform. and Computation*, vol. 139 (1997), 17–38.

[26] V. P. Glushkov, The abstract theory of automata, *Russian Math. Surveys*, vol. 16 (1961), 1–53.

[27] L. Ilie and S. Yu, Follow automata, *Inf. Comput.*, vol. 186,1 (2003), 140–162.

[28] A. Khorsi, F. Ouardi and D. Ziadi, Fast equation automaton computation, *J. Discrete Algorithms*, vol. 6,3 (2008), 433–448.

[29] D. Krieger, A. Miller, N. Rampersad, B. Ravikumar and J. Shallit, Decimations of languages and state complexity, *Theor. Comput. Sci.*, vol. 410,24-25 (2009), 2401–2409.

[30] P. B. A. Lecomte and M. Rigo, Numeration systems on a regular language, *Theory Comput. Syst.*, vol. 34,1 (2001), 27–44.

[31] S. Lombardy and J. Sakarovitch, Derivation of rational expressions with multiplicity, *Theoret. Computer Sci.*, vol. 332 (2005), 141–177.

[32] S. Lombardy and J. Sakarovitch, How expressions can code for automata, *RAIRO Theor. Informatics and Appl.*, vol. 39 (2005), 217–237.

[33] S. Lombardy and J. Sakarovitch, Corrigendum to our paper : How expressions can code for automata, *RAIRO - Theor. Inf. and Applic.*, vol. 44,3 (2010), 339–361.

[34] R. McNaughton and H. Yamada, Regular expressions and state graphs for automata, *IRE Trans. Electronic Computers*, vol. 9 (1960), 39–47.

[35] M. Nivat, Transductions des langages de Chomsky, *Ann. Inst. Fourier (Grenoble)*, vol. 18 (1968), 339–455.

[36] R. Paige and R. E. Tarjan, Three partition refinement algorithms, *SIAM J. Comput.*, vol. 16,6 (1987), 973–989.

[37] D. Perrin, Finite automata, in : *Handbook of Theoretical Computer Science*, J. van Leeuwen (ed.), vol. B, Elsevier, 1990, 1–53.

[38] C. Reutenauer, Une caractérisation de la finitude de l'ensemble des coefficients d'une série rationnelle en plusieurs variables non commutatives, *C. R. Acad. Sci. Paris 284*, 1877, 1159–1162.

[39] M. Rigo, Numeration systems on a regular language : arithmetic operations, recognizability and formal power series, *Theor. Comput. Sci.*, vol. 269,1-2 (2001), 469–498.

[40] M. Rigo, Numeration systems : a link between number theory and formal language theory, in : *Proc. DLT 2010*, Y. Gao, H. Lu, S. Seki and S. Yu (ed.), *Lecture Notes in Comput. Sci.* n° 6224, Springer, 2010, 33–53.

[41] J. M. Rutten, Behavioural differential equations : a coinductive calculus of streams, automata, and power series, *Theoret. Computer Sci.*, vol. 308 (2003), 1–53.

[42] J. Sakarovitch, Deux remarques sur un théorème de S. Eilenberg, *RAIRO Theor. Informatics and Appl.*, vol. 17 (1983), 23–48.

[43] J. Sakarovitch, *Eléments de théorie des automates*, Vuibert, 2003. English translation : *Elements of Automata Theory*, Cambridge University Press, 2009.

[44] M. P. Schützenberger, Sur les relations rationnelles, in : *Proc. Automata Theory and Formal Languages*, H. Brackhage (ed.), *Lecture Notes in Comput. Sci.* n° 33, 1975, 209–213.

[45] M. P. Schützenberger, Sur une variante des fonctions séquentielles, *Theoret. Computer Sci.*, vol. 4 (1977), 47–57.

[46] J. Shallit, Numeration systems ;linear recurrences, and regular sets, *Inform. and Comput.*, vol. 113 (1994), 331–347.

[47] K. Thompson, Regular expression search algorithm, *Comm. Assoc. Comput. Mach.*, vol. 11 (1968), 419–422.

[48] A. Weber and R. Klemm, Economy of description for single-valued transducers, *Inform. and Comput.*, vol. 118 (1995), 327–340.

[49] D. Wood, *Theory of Computation*, John Wiley, 1987.

[50] S. Yu, Regular languages, in : *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa (ed.), vol. 1, Elsevier, 1997, 41–111.

[51] D. Ziadi, J.-L. Ponty and J.-M. Champarnaud, Passage d'une expression rationnelle à un automate fini non-déterministe, *Bull. Belg. Soc. Math.*, 1997, 177–203.

Index des Notations

$\mathcal{A} \bowtie \mathcal{B}$ produit synchronisé de deux automates, 142

A^* monoïde libre d'un alphabet, 17

Seqpm ensemble des fonctions séquentielles par morceaux, 127

$|\mathsf{E}|$ Langage ou série dénotée par une expression, 20

coSeqpm ensemble des fonctions coséquentielles par morceaux, 128

$\mathrm{BD}(\mathsf{E})$ termes dérivés cassés d'une expression, 47

$\mathrm{D}(\mathsf{E})$ termes dérivés d'une expression, 36

$\pi_{\mathcal{S}}(u)$ valeur d'un mot dans un SNA, 115

$\frac{\partial_b}{\partial a}\mathsf{E}$ dérivation cassante d'une expression, 47

$\frac{\partial}{\partial a}\mathsf{E}$ dérivation d'une expression, 34

$\mathrm{B}(\mathsf{E})$ cassage d'une expression, 46

$\mathbb{K}\langle\langle A^*\rangle\rangle$ ensemble des séries engendrées par un monoïde libre sur un semi-anneau, 27

$\langle n\rangle_{\mathcal{S}}$ représentation d'un entier dans un SNA, 115

$\mathsf{RatE}(\mathsf{A})$ ensemble des expression rationnelles, 19

Succ_L fonction successeur d'un langage, 117

$\mathsf{c}(\mathsf{E})$ terme constant, 20

$\mathrm{TBD}(\mathsf{E})$ vrais termes dérivés cassés d'une expression, 47

$\mathrm{TD}(\mathsf{E})$ vrais termes dérivés d'une expression, 36

$<s, u>$ coefficient d'un mot dans une série, 27

$\mathsf{d_p}(u, v)$ distance préfixe de deux mots, 122

$\mathsf{d}(\mathsf{E})$ profondeur d'une expression, 19

$[a]^{-1}(\mathsf{E})$ expression dérivée d'une expression, 31

$\mathcal{D}_b(\mathsf{E})$ automate des termes dérivés cassés d'une expression, 50

$\mathcal{D}(\mathsf{E})$ automate des termes dérivés d'une expression, 40

$[\![x]\!]$ expression dénoté par un nœud, un mot de nœuds ou un mot de marques, 62

$\phi(x)$ père d'un nœud, 62

$\kappa||\tau$ concaténation de deux transducteurs, 148

Seq ensemble des fonctions séquentielles, 121

$\ell(\mathsf{E})$ Longueur littérale d'une expression, 19

$|\mathcal{A}|$ langage ou série reconnue par un automate, 18

$|u|$ longueur d'un mot, 17

$\mathrm{fp}(\mathsf{E})$ feuilles propres d'une expression, 22

$(X)_{\mathsf{p}}$ partie propre d'un ensemble, 46

coSeq ensemble des fonctions coséquentielles, 121

Index

alphabet, 17
automate
  d'entrée sous-jacent, 119
  des termes dérivés cassés d'une expression, 50
  des termes dérivés d'une expression, 40
  fini, 17
    accessible, 18
    ambigu, 18
    co-accessible, 18
    des positions d'une expression, 23
    déterministe, 18
    standard, 21
    à multiplicités, 28
    émondé, 18
  monocycle, 140

branchement, 133
  absolu, 134
  faible, 133
  fort, 134
  infini, 135

comportement d'un automate, 28
concaténation
  à droite de sous-expressions, 40

derivation, 34
distance préfixe, 122

expression
  constante étoilée, 53
  dérivée d'une expression, 31
  rationnelle, 19, 87
    en forme normale, 24
    équivalente à une autre, 20

feuilles propres d'une expression, 22
fonction
  co-séquentielle, 121
  lipschitzienne, 122
  rationnelle, 117
  successeur, 117
  successeur uniforme, 150
  séquentielle, 121
    par morceaux, 127

identités triviales, 20, 88

langage, 17
  dénoté par une expression, 20
  rationnel, 19
  rayon, 140
  reconnaissable, 18
  reconnu, 18
longueur
  d'un mot, 17
  littérale d'une expression, 19

monoïde libre, 17
mot, 17

ordre radiciel, 115

partie propre d'un ensemble, 46

produit synchronisé, 142
profondeur d'une expression, 19

représentation, 28
    d'un entier, 115

système de numération abstrait, 115
série
    $\mathbb{K}$-reconnaissables, 28
    caractéristique d'un langage, 27
    dénotée d'une expression, 89
    formelle, 27
    énumératrice d'un SNA, 159

terme constant, 20, 89
    d'un nœud, 62
termes cassés d'une expression, 46
termes dérivés
    cassés d'une expression, 47
    d'une expression, 36
transducteur, 118
    co-séquentiel, 121
    fonctionnel, 119
    lettre-à-lettre, 119
    monocycle, 140
    standard, 119
    séquentiel, 121

valeur d'un mot dans un SNA, 115
vrais termes dérivés
    cassés d'une expression, 47
    d'une expression, 36

états
    q-conjugués, 125
    jumeaux, 125
      conjugués, 126
    monozygotes, 134