Systèmes de fonctions holonomes : application à la théorie des automates

Florent Koechlin

Université Gustave Eiffel

Systèmes de fonctions holonomes : application à la théorie des automates

Thèse de doctorat de l'Université Gustave Eiffel

École doctorale n° 532, Mathématiques et Sciences et Technologies de l'Information et de la Communication, MSTIC
Spécialité de doctorat : Informatique
Unité de recherche : Laboratoire d'Informatique Gaspard-Monge, LIGM

Thèse présentée et soutenue à l'Université Gustave Eiffel, le 09/12/2021, par

Florent KOECHLIN

Composition du Jury

Mireille BOUSQUET-MÉLOU Directrice de Recherche, CNRS	Examinatrice
Patricia BOUYER-DECITRE Directrice de Recherche, CNRS	Présidente du jury et Rapportrice
Arnaud CARAYOL Professeur, Université Gustave Eiffel	Directeur de thèse
Julien CLÉMENT Chargé de Recherche, CNRS	Examinateur
Conrado MARTÍNEZ Professeur, Universitat Politècnica de Catalunya	Rapporteur
Marni MISHNA Professeure, Simon Fraser University	Examinatrice
Cyril NICAUD Professeur, Université Gustave Eiffel	Directeur de thèse
Sylvain SCHMITZ Professeur, Université de Paris	Examinateur

Thèse de Doctorat en Informatique Théorique

Systèmes de fonctions holonomes, application à la théorie des automates

Florent Koechlin

Thèse soutenue le 9 décembre 2021 devant le jury composé de :

Mireille Bousquet-Mélou	*examinatrice*	CNRS
Patricia Bouyer-Decitre	*rapportrice*	CNRS
Arnaud Carayol	*directeur de thèse*	Université Gustave Eiffel
Julien Clément	*examinateur*	CNRS
Conrado Martínez	*rapporteur*	Universitat Politècnica de Catalunya
Marni Mishna	*examinatrice*	Simon Fraser University
Cyril Nicaud	*directeur de thèse*	Université Gustave Eiffel
Sylvain Schmitz	*examinateur*	Université de Paris

École Doctorale 532
Mathématiques et Sciences et Technologies de l'Information et de la Communication (Mstic)

Université Gustave Eiffel

Thèse de Doctorat en Informatique Théorique

Systèmes de fonctions holonomes, application à la théorie des automates

Florent Koechlin

Thèse soutenue le 9 décembre 2021

Champs-sur-Marne, 2021

Dernière révision : 7 décembre 2021

Adresse de contact : florent.koechlin@free.fr

Table des matières

Résumé

Cette thèse s'articule autour de deux parties indépendantes, qui s'intéressent à des objets différents, mais qui se rassemblent dans la méthodologie appliquée pour les étudier : nous nous appuyons essentiellement dans les deux parties sur des outils mathématiques spécialisés dans l'étude des systèmes de séries génératrices.

Dans la première partie, nous étudions des arbres dont les nœuds sont étiquetés par des opérateurs, qui représentent des expressions avec une sémantique, comme les expressions régulières qui représentent des langages, ou les expressions logiques qui représentent des fonctions booléennes. Nous supposons que pour ces arbres il existe un opérateur $\circledast$ qui possède un élément absorbant $\mathcal{P}$, dans le sens où tout arbre de racine $\circledast$ et dont l'un des fils est $\mathcal{P}$ est équivalent sémantiquement à $\mathcal{P}$ (par exemple, 0 est absorbant pour $\times$, $(a+b)^*$ est absorbant pour $+$ sur les langages à deux lettres, etc). Nous définissons à partir de cet élément absorbant une fonction de réduction σ qui réduit une expression en simplifiant de bas en haut toutes les occurrences de l'élément $\mathcal{P}$ sous l'opérateur $\circledast$. Cette réduction préserve la sémantique de l'arbre, par définition d'un élément absorbant. La première partie de cette thèse étudie la taille moyenne de la réduction d'un arbre aléatoire de taille n, lorsque $n \to \infty$. Dans les deux premiers chapitres, nous nous sommes intéressés à des arbres d'expressions uniformes décrits respectivement par un système combinatoire et une équation récursive unidimensionnelle. Nous avons montré que la distribution uniforme était dégénérée pour ces arbres d'expression en présence d'un élément absorbant : la taille moyenne après réduction tend vers une constante lorsque $n \to \infty$. Ce résultat remet en cause l'utilité des expressions aléatoires uniformes pour l'étude de la complexité en moyenne des algorithmes, ainsi que leur pertinence pour les tests de performance automatisés (benchmarks). Nous avons ensuite affiné le résultat précédent dans le cas des expressions régulières, en utilisant plus de règles de simplifications sémantiques propres aux langages. Enfin, nous nous sommes penchés sur le comportement en moyenne de la réduction pour une autre distribution, la distribution ABR. Nous montrons que la taille moyenne après réduction dépend de la probabilité d'apparition de l'opérateur absorbant, avec des changements de phase lorsque ce paramètre varie de 0 à 1.

La deuxième partie étudie le lien entre les langages formels et les propriétés de leurs séries génératrice. Nous nous intéressons en particulier à notion de non-ambiguïté de certaines classes d'automates à compteurs. Intuitivement, un automate est non ambigu si tout mot qu'il accepte est accepté par un unique calcul acceptant dans l'automate. La non-ambiguïté permet de relier par une bijection les calculs

de l'automate, qui suivent un description combinatoire dictée par les transitions de l'automate, aux mots du langage accepté par l'automate. Cette bijection permet alors de traduire la structure particulière de l'automate en équation sur les séries génératrices associées à l'automate. Dans la deuxième partie, nous étudions le lien entre des classes d'automates à compteur non ambigus, dont font partie les automates de Parikh faiblement non ambigus, et la classe des séries génératrices holonomes. Nous commençons par définir de façon robuste la notion de non-ambiguïté pour ces classes d'automates, en les reliant à plusieurs classes de langages standard de la littérature. Nous utilisons ensuite le lien avec les séries génératrices holonomes pour développer des techniques de preuves d'intrinsèque ambiguïté. Nous proposons par ailleurs une preuve d'intrinsèque ambiguïté à la main d'un langage pour lequel les techniques précédentes ne s'appliquent pas. Enfin, en exploitant les récurrences satisfaites par les coefficients des séries holonomes, nous produisons des bornes de complexité pour le problème de l'inclusion des automates de Parikh faiblement non ambigus.

Abstract

This thesis is structured around two independent parts, which are dealing with different subjects, but can be unified by the methodology used to study them : in both parts we rely mainly on mathematical tools specialized in the study of generating series systems.

In the first part, we are interested in trees with nodes labelled by operators that represent expressions with semantics. For example, regular expressions represent languages, logical expressions represent boolean functions, etc. We assume that there is an operator $\circledast$ associated to an absorbing element $\mathcal{P}$, such that any tree having $\circledast$ as a root and $\mathcal{P}$ as one of its children is equivalent to $\mathcal{P}$ (for example, 0 is absorbing for $\times$, $(a+b)^*$ is absorbing for $+$ on two-letter languages, etc). From this absorbing pattern, we define a reduction function σ that reduces a random expression with n nodes, by recursively simplifying bottom-up all occurrences of the element $\mathcal{P}$ under the operator $\circledast$. This reduction preserves the semantics of the tree, by definition of an absorbing element. The first part of this thesis studies the average size of the reduction of a random tree of size n, when $n \to \infty$. In the first two chapters, we focus on uniform expression trees described by a combinatorial system, and a one-dimensional recursive equation respectively. We show that the uniform distribution is degenerated for these expression trees when there is an absorbing pattern : the average size after reduction tends to a constant when n tends to infinity. This result questions the usefulness of uniform random expressions in the average complexity analysis of algorithms, as well as their relevance for automated performance tests (such as benchmarks). We then refine the previous result in the case of regular expressions, using more language-specific simplification rules. Finally, we investigate the average behaviour of the reduction over another distribution, the BST-like distribution. We show that the average size after reduction depends on the probability of appearance of the absorbing operator, and we exhibit two different thresholds in the typical average reduction size, when this parameter varies from 0 to 1.

The second part studies the link between formal languages and the properties of their generating series. In particular, we are interested in the notion of unambiguity of some classes of counter automata. Intuitively, an automaton is unambiguous if any word is accepted by at most one single accepting run. Unambiguity makes it possible to link by a one-to-one correspondance the runs of the automaton, which follow a combinatorial specification described by the transitions of the automaton, to the words of the language accepted by the automaton. This correspondance can

be used then to translate the particular structure of the automaton into a system of equations over the generating series associated to the automaton. In this second part, we study the link between different equivalent classes of unambiguous counter automata, including weakly unambiguous Parikh automata, and the class of holonomic generating series. We start by introducing the notion of unambiguity for these classes of automata, by relating them with several classes of classical automate in the literature. We then use the link with holonomic generating series to develop techniques for proving inherent ambiguity. We also prove by hand the inherent ambiguity of a language for which the previous techniques do not apply. Finally, by analyzing the recurrences satisfied by the coefficients of holonomic series, we give complexity bounds for the inclusion problem for weakly unambiguous Parikh automata.

Remerciements

Je remercie chaleureusement Mireille Bousquet-Mélou, Patricia Bouyer-Decitre, Julien Clément, Marni Mishna, Conrado Martínez et Sylvain Schmitz qui ont eu la gentillesse d'accepter de faire partie de mon jury de thèse et de lire mon manuscrit.

Je remercie doublement Patricia Bouyer-Decitre et Conrado Martínez pour la bienveillance avec laquelle ils ont rapporté ma thèse, et pour leurs remarques et suggestions qui m'ont aidé à améliorer ce document, et qui me seront très utiles pour mes travaux futurs.

Je souhaite dire un énorme merci à mes directeurs Arnaud et Cyril. Il m'est difficile de décrire en quelques lignes tout ce que ces trois années de thèse à leurs côtés m'ont apporté, aussi bien scientifiquement qu'humainement. Je ne leur serai jamais assez reconnaissant pour tout leur soutien, leur aide, leurs conseils, leurs explications, leurs idées, leur recul, leur bienveillance, leur amitié. Mille mercis pour tous ces bons moments à travailler ensemble, à discuter, à déjeuner ou à faire la fête en juin.

Un grand merci à mon co-auteur et ami Pablo Rotondo, avec qui j'ai eu le plaisir de travailler. Je dois à cette collaboration plusieurs chapitres de cette thèse — sans compter tout ce que j'ai appris en faisant de la recherche avec lui.

Je tiens aussi à remercier Alin Bostan pour son aide précieuse sur les questions mathématiques que nous nous sommes posées sur les séries holonomes et les calculs de diagonales ; je le remercie pour avoir à maintes reprises réussi à rendre simples par ses explications des notions mathématiques qui m'étaient inaccessibles.

Je suis extrêmement reconnaissant à Fabian Reiter de m'avoir permis d'utiliser le magnifique style LaTeX de sa thèse ; j'ai gagné un temps extrêmement précieux dans la rédaction grâce à lui.

Je suis reconnaissant à tous les membres du laboratoire LIGM qui m'ont très bien accueilli pendant ces trois années de thèse (et quelques mois avec le stage de M2). Plus précisément mille mercis à Carine pour sa super bonne humeur, ses bonbons, et ses précieux conseils en Python ; Vincent pour sa gentillesse, ses blagues et ses jeux de mots ; Stéphane pour le fou rire face à mon combat avec le massicot (remporté par le massicot), et plus sérieusement pour toutes les pauses café organisées à distance pendant les confinements, qui ont beaucoup compté pour moi ; Claire pour tous ses conseils, ses connaissances sur le fonctionnement du labo et de la fac, et plus particulièrement pour avoir gentiment pris des nouvelles et proposé de discuter plusieurs fois pendant ma troisième année ; Nadime pour avoir pris longuement le temps de discuter et de m'expliquer des choses alors qu'il était débordé de travail, pour m'avoir proposé son aide au début de ma thèse, si jamais j'avais besoin d'un

regard extérieur ; Victor pour sa bonne humeur, et pour ses conversations toujours passionnantes avec Nadime aux pauses déjeuners ; Marc pour ce moment avant la surveillance d'un examen où il m'a expliqué de façon lumineuse les rotations gauches et droites sur les AVL. Je remercie aussi Marie-Pierre, Didier, Wenjie, Philippe G, Jamal, Dominique P et Giuseppina pour avoir toujours pris le temps de dire bonjour ou de prendre gentiment des nouvelles lorsque je les croisais dans les couloirs ; je remercie aussi tous ceux que j'ai pu croiser aux pauses déjeuners ou café pour ces moments de bonne humeur : Philippe B, Olivier B, Olivier C, Sylvain C (merci aussi pour la reprise d'Overkill pendant le confinement !), Éric C, Étienne, Rémi, Alfredo, Arnaud, Antoine, Dominique R, Chloé, Johan, Marie, Mathias, et j'oublie sûrement de nombreuses personnes que je connais de visage mais pas forcément de nom.

Merci à tous les doctorants que j'ai croisés au LIGM pour la bonne ambiance qui a régné dans les salles des doctorants, et au-delà des murs du labo : merci à Revekka, Yoshihiro, Christophe, Thomas, Weiqin, Adrien, Niloufar, Maxime, Doriann, Hélène, Zéphyr, Aaron, Raphaël, Nicolas, Corentin et Clément.

Merci à Séverine, Patrice et Nathalie pour leur aide précieuse pour que tout se passe au mieux dans le laboratoire. Merci à Corinne pour ses nombreux encouragements, et pour tous ces bons et longs moments passés à discuter. Merci enfin à toute l'équipe du restaurant administratif, et notamment à Klawdette et Hanan pour leur bonne humeur et leurs sourires tous les jours au déjeuner.

Merci à toute l'équipe du laboratoire de Caen pour son accueil chaleureux lors de son séminaire, plus précisément Ali Akhavi, Julien Clément, Julien Courtiel et Matthieu Dien. Un merci tout particulier à Brigitte Vallée pour ses précieux conseils et sa bienveillance pendant tous mes exposés.

Je remercie Anne Micheli, Dominique Poulalhon et Mo Foughali avec qui j'ai eu le plaisir de travailler et discuter dans le cadre de mon ATER à l'IRIF, et aussi Thomas Colcombet, Olivier Serre, Sylvain Schmitz de l'équipe automates, et Guillaume Chapuy, Enrica Duchi et Matthieu Josuat-Vergès de l'équipe combinatoire, pour leur accueil à l'IRIF. Merci à Elie De Panafieu pour ses encouragements à la fin d'un de mes exposés.

Je remercie toute la communauté ALEA et AofA ; il serait trop long de citer tous les noms, mais tout le monde participe activement à rendre ces journées très agréables et accueillantes aux nouveaux arrivants, et je leur en suis extrêmement reconnaissant. Je souhaiterais en particulier remercier Frédérique Bassino et Bruno Salvy pour cette agréable soirée passée dans un bon restaurant à Marseille, un soir d'AofA, en compagnie de Carine et Pablo.

Je remercie tous les professeurs de l'ENS Cachan qui m'ont formé en Informatique et appris tous les fondements de l'informatique théorique. Je remercie tous mes directeurs de stage qui m'ont introduit à la recherche au cours de ma scolarité à l'ENS Cachan : Florian de Vuyst, José Correa, et bien sûr Arnaud Carayol et Cyril Nicaud. Je remercie aussi mes professeurs de mathématiques de prépa, Madame Mahieux, Monsieur Espéret et Monsieur Francinou, à qui je dois tant ; merci aussi à M. Péchaud pour ses cours d'Informatique en prépa, et Monsieur Brunel et Monsieur Coup pour leurs cours de Physique, même si j'ai presque tout oublié. Enfin, je remercie mon professeur de Terminale, Monsieur Labelle, dont la passion extraordinaire pour les mathématiques et l'enseignement m'ont donné envie de continuer mes études dans cette direction.

Tout au long de ma thèse, j'ai bénéficié du soutien et des encouragements de tous mes amis, que je souhaiterais remercier ici. Mes amis du collège, Marimo,

Maud (à qui je souhaite bon courage pour sa propre thèse !), Sophia et Sylvain ; mes amis du lycée, Marc-Antoine, Wesley, Mylane, Robin, Clara, Florence, Ophir, Emma-Louise, Samy, Rémi, qui m'ont tous encouragé dans la dernière ligne droite de la thèse, et plus généralement tous les camarades de classe de la TS1 que j'ai pu revoir à l'occasion d'un concert de Marc-Antoine, ou chez Madame Stoliaroff, à qui je témoigne ici toute mon affection. Merci aussi à Madame Millot d'avoir gardé le contact et régulièrement continué à demander des nouvelles, pour ses mails humoristiques pendant les confinements, et pour avoir proposé gentiment de m'aider à réviser mon anglais pour préparer un exposé que je devais faire en anglais. Merci à Anthony pour ces découvertes à Marseille. Merci à Adeline et Hugo pour leur amitié et les invitations à leurs soirées pendant ces années de thèse, qui m'ont permis de garder le contact avec quelques camarades de prépa. Merci à mes amis de HX2 avec qui j'ai un peu gardé contact, Pierre, Alexis, Dédé, Clément, Théo, Juliette, Mercedes. Merci à Burhaan et Oscar pour avoir organisé une rencontre des HX2, 10 ans après. Merci à Guillaume, Sam, Clément, Thomas, Emmanuel et Mathieu pour leurs soirées fort sympathiques pendant ces trois années de thèse, mêlant jeux de sociétés, bon repas, et concerts de musique classique. Merci à Guilhem pour toutes ces conversations sur Skype qui durent depuis maintenant plusieurs années, et qui sont venues combler la solitude des confinements ; merci également pour avoir organisé toutes ces soirées au grand Rex pour aller voir en avant-première les derniers Avengers. Merci à Émilie, Julien, Alexandra, Paul pour leur amitié et pour avoir répondu présents aux soirées organisées par Guilhem. Merci à Emeline, Bastián, Pénélope, Pierre et Quentin.

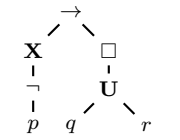
J'embrasse toute ma famille qui m'a encouragé dans toutes mes études ; mes parents Benoit et Nicole, ma soeur Aurore dont je vais peut-être enfin avoir le temps de lire le livre sur le féminisme, son mari Seb, mon frère Raphaël, et bien sûr Emmanuelle ma deuxième soeur. Merci à Pierre Simsolo pour son amitié de 28 ans. Merci à Papa pour avoir relu ma thèse et relevé de nombreuses fautes et coquilles. J'embrasse toute la famille Koechlin et Lancereau, et plus particulièrement Denis, Isabelle et Guillaume.

À Louis, À Anne.

Introduction générale à la thèse

Cette partie est une introduction globale pour présenter les idées et concepts généraux développés dans la thèse. Pour une introduction plus détaillée, avec notamment un état de l'art en bonne et due forme, je renvoie le lecteur aux introductions qui ouvrent chacune des deux parties.

Dans le domaine de la vérification logicielle, on cherche à vérifier automatiquement qu'un programme satisfait certaines contraintes. Celles-ci sont généralement exprimées par des formules logiques. Dans [DGV99], il est décrit un algorithme pour générer des expressions LTL aléatoires, qui est une des logiques utilisées en vérification. Ce procédé de génération est notamment utilisé par l'outil *LTL-to-Büchi translator testbench* (`lbtt`) de TCS [Tau00], ou encore est implémenté dans Spot [DLLF+16]. Lorsque l'on regarde la façon dont sont générées ces expressions aléatoires, il peut paraître surprenant de ne trouver dans l'algorithme aucune mention de la sémantique des expressions LTL : l'algorithme génère en fait des arbres purement syntaxiques, sans jamais utiliser le fait que ce sont des formules LTL et que les opérateurs et les feuilles ont un sens. Cela peut paraître paradoxal, car les arbres ainsi générés ont généralement vocation à être ensuite utilisés pour tester des algorithmes, qui eux les voient bien comme des expressions LTL, avec une signification précise en tant que formule de la logique temporelle.



$\mathbf{X}(\neg p) \rightarrow \Box(q \mathbf{U} r)$

Figure A. Exemple de formule LTL

D'un point de vue combinatoire, il est alors naturel de se demander si la distribution issue de cet algorithme de génération aléatoire est pertinente pour son utilisation principale, à savoir faire des *benchmarks*. Pour donner un autre exemple, certaines études de complexité en moyenne de l'algorithme de Glushkov [BMMR12, BMMR11], qui transforme une expression régulière en automate équivalent, se basent sur la distribution uniforme d'expressions régulières vues là encore comme des arbres purement syntaxiques, sans jamais prendre en compte le fait qu'ils décrivent des langages, et qu'un $+$ par exemple n'est pas juste un opérateur binaire, mais représente une union de langages. Or en faisant quelques tests, on observe très vite que la distribution uniforme sur ces expressions régulières a tendance à générer beaucoup de redondances, dans le sens où les arbres sont inutilement gros par rapport au langage qu'ils décrivent. Cela peut remettre en question la pertinence de l'analyse de complexité en moyenne de

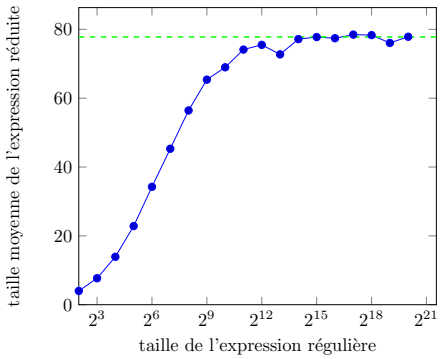


Figure B. En prenant en compte des règles élémentaires de simplifications du type $\Sigma^* \cup E \to \Sigma^*$, les expressions régulières aléatoires uniformes se réduisent considérablement : en moyenne, pour un alphabet de taille 2, une expression de taille plusieurs millions est équivalente à une expression de taille inférieure à 80 (courbe expérimentale, voir chapitre 4).

ces algorithmes, dont le but est en théorie d'estimer leur comportement lors d'une utilisation *réelle* : en effet, les utilisateurs de l'algorithme de Glushkov ne manipulent pas en pratique des expressions régulières avec autant de parties redondantes et inutiles.

Pour montrer qu'il faut faire attention en général avec ce type d'approches, et même pouvoir affirmer dans certains cas qu'il s'agit d'une mauvaise idée car les distributions en jeu ne sont pas pertinentes, il ne peut pas être satisfaisant de ne traiter que quelques exemples particuliers, même lorsqu'ils sont utilisés en pratique. En effet, aussi bien l'algorithme de génération des formules LTL, que la description des expressions régulières utilisées pour l'analyse en moyenne, sont hautement paramétrables : il suffit d'une variation du modèle étudié, en modifiant par exemple la probabilité d'un opérateur, l'arité de certains nœuds, en remplaçant un opérateur par un autre qui lui est équivalent, pour rapidement rendre caduque une preuve spécifique à l'exemple étudié. Non, pour pouvoir apporter un résultat négatif, il faut identifier un cadre général qui s'applique au plus de spécifications possibles, et qui soit par ailleurs robuste. Un outil de choix pour le faire est la *combinatoire analytique*, qui s'est justement beaucoup appliquée à rassembler dans un même cadre unifié l'étude de classes combinatoires pourtant en apparence très différentes, à l'aide notamment de la *méthode symbolique*.

Cette approche a permis d'identifier de nombreux comportements universels, comme par exemple la hauteur moyenne en $O(\sqrt{n})$ de la plupart des classes d'arbres uniformes à n nœuds. Avec ce même objectif en tête, nous donnerons un cadre suffisamment général dans lequel toute spécification décrivant des expressions régulières est dégénérée vis-à-vis de la distribution uniforme : elle génère des arbres beaucoup trop redondants comme sur l'exemple de la Figure B. Cela questionne l'intérêt en pratique des nombreuses études de complexité moyenne d'algorithmes dont l'entrée est une expression rationnelle suivant une distribution uniforme.

L'algorithme utilisé par lbtt ou Spot, quant à lui, ne génère pas des expressions aléatoires uniformes. La distribution correspondante est très classique en combinatoire. Elle produit des arbres qui ressemblent aux arbres binaires de recherche aléatoires, et est appelée distribution ABR. Statistiquement, de tels arbres sont très

différents de ceux suivant la distribution uniforme : par exemple, la hauteur d'un arbre typique de type ABR à n nœuds est en $O(\log n)$. Nous avons également étudié ce cadre et montré que les comportements sont plus diversifiés, avec des phénomènes de seuil selon le choix des probabilités des opérateurs.

L'idée de tester les limites d'un modèle est également abordée sous un angle différent dans la thèse. On la retrouve en effet naturellement en théorie des langages formels pour séparer les classes de langages : par exemple, trouver un langage qui n'est reconnu par aucun automate du modèle étudié permet de mieux saisir ses spécificités et ses limitations. Le problème historique de l'intrinsèque ambiguïté des langages algébriques est un exemple représentatif de ces questionnements, qui sera au centre de la seconde partie de la thèse. Un langage algébrique est dit intrinsèquement ambigu s'il n'est reconnu par aucune grammaire non ambiguë ; la non-ambiguïté des langages algébriques est une notion fondamentale pour les langages de programmation, pour la linguistique, mais elle a aussi des conséquences algorithmiques importantes en vérification, certains problèmes indécidables en général devenant décidables pour les versions non ambiguës. Il est naturel d'essayer d'identifier un grand nombre de langages algébriques intrinsèquement ambigus, pour comprendre ce qui est responsable de l'intrinsèque ambiguïté dans le modèle des langages algébriques.

Il s'agit d'une question difficile (indécidable en général), et historiquement les preuves fournies étaient très spécifiques au langage étudié. De toutes petites modifications du langage, qui intuitivement ne changent pas son intrinsèque ambiguïté, rendait la preuve complètement inutilisable : l'exemple le plus marquant étant peut-être celui du langage des mots $a^n b^m c^p$ avec $n = m$ ou $m = p$, dont la preuve échoue complètement sur le langage presque identique des mots $a^n b^m c^p$ avec $n \neq m$ ou $m \neq p$. En rupture avec ces précédentes approches essentiellement au cas par cas, Flajolet en 1987 [Fla87] a alors proposé un cadre général capable de capturer par des méthodes analytiques de nombreux marqueurs spécifiques à l'intrinsèque ambiguïté. Ce cadre s'est avéré suffisamment général et robuste pour démontrer ou redémontrer en un seul article l'intrinsèque ambiguïté de très nombreux langages algébriques, dont certains étaient encore à l'état de conjecture. Flajolet s'est appuyé sur une correspondance bien connue entre classes de langages et classes de séries formelles : les langages réguliers sont naturellement associés à des séries rationnelles, et les langages algébriques non ambigus à des séries algébriques (c'est le théorème de Chomsky-Schützenberger).

Toutes ces questions relatives à la non-ambiguïté ne sont pas réservées aux langages algébriques, mais concernent en réalité toutes les classes de langages associées à des modèles d'automates. La non-ambiguïté est à mi-chemin entre les versions déterministes des automates, dont le comportement est trop restreint, et les versions non déterministes, qui ont trop de degrés de liberté et sont difficiles à étudier : elle offre généralement un bon compromis entre expressivité et complexité du modèle. Ainsi, comme nous l'avons déjà évoqué pour les langages algébriques, certains problèmes indécidables sur les classes non déterministes deviennent décidables sur les versions non ambiguës : par exemple le problème de l'universalité, qui est indécidable pour les grammaires hors-contexte, ou les automates de Parikh que nous étudierons dans la seconde partie, est décidable sur les versions non ambiguës. Dans la même idée, la complexité des problèmes peut changer sur les versions non-ambiguës ; par exemple le problème de l'inclusion est polynomial pour les automates finis non

ambigus, mais est PSPACE-complet sur les versions non déterministes. Ces études de complexité, qui sont reliées à la taille des automates considérés, sont d'autant plus intéressantes qu'en général, les versions non ambiguës permettent de décrire les langages de façon plus succincte que les versions déterministes (par exemple le langage $\Sigma^* a \Sigma^k$, pour $k \in \mathbb{N}$ fixé, des mots qui ont un a en k-ième position en partant de la fin, est décrit par un automate fini non ambigu de taille linéaire en k, mais tout automate déterministe qui le reconnaît a une taille au moins exponentielle en k).

Malheureusement, la non-ambiguïté est une notion particulièrement difficile, en partie parce qu'elle est externe aux modèles d'automates, dans le sens où elle décrit une contrainte globale sur le comportement des automates, par opposition par exemple au déterminisme qui se définit localement sur les transitions. Même le cas des automates finis, où le modèle déterministe, non ambigu et non déterministe sont pourtant équivalents, n'est toujours pas entièrement compris et est toujours étudié de nos jours.

Pour toutes ces raisons (et bien d'autres, voir par exemple [Col15] pour une présentation plus détaillée son importance), la non-ambiguïté est une notion fondamentale de la théorie des automates, qui est systématiquement étudiée, au même titre que les versions déterministes. Vu la puissance des résultats obtenus par Flajolet sur les langages algébriques, il est raisonnable d'essayer de les étendre à d'autres classes d'automates. En vérification, il existe de nombreux autres modèles de langages que les langages algébriques, qui les généralisent simplement, en ajoutant par exemple des compteurs pour capturer des langages supplémentaires, comme $a^n b^n c^n$. Du côté des séries, les séries holonomes sont une extension naturelle des séries algébriques, et sont très utilisées en combinatoire. Si les séries rationnelles sont solutions d'équations linéaires et les séries algébriques sont solutions d'équations polynomiales, les séries holonomes sont quant à elles solutions d'équations différentielles linéaires à coefficients polynomiaux. Dans la deuxième partie de la thèse, nous avons donc cherché à généraliser l'approche de Flajolet en identifiant une classe de langages qui est à la fois pertinente dans le monde de la vérification, et dont la non-ambiguïté est capturée par l'holonomie de sa série associée.

Précisons un peu le contenu des deux parties. La première partie de cette thèse s'intéresse à des familles d'arbres décrivant des expressions, et à leur comportement en présence d'un élément absorbant. Afin de garantir un cadre d'étude le plus général possible, nous n'utilisons qu'une seule règle très simple de sémantique sur ces objets, induite par la présence d'un élément absorbant. Dans un ensemble d'arbres d'expressions, un élément $\mathcal{P}$ est appelé absorbant pour un opérateur $\circledast$ si toute expression de la forme $\overset{\circledast}{\underset{\mathcal{P}\ T}{/\!\backslash}}$ ou $\overset{\circledast}{\underset{T\ \mathcal{P}}{/\!\backslash}}$, avec T une expression quelconque, est équivalente sémantiquement à $\mathcal{P}$ (c'est-à-dire qu'elle représente le même objet que $\mathcal{P}$). Ce phénomène est suffisamment courant pour se retrouver chez un grand nombre de familles d'expressions très éloignées ($\bot$ est absorbant pour $\wedge$ pour les formules logiques, Σ^* est absorbant pour l'union $+$ des expressions régulières, 0 est absorbant pour $\times$ chez les formules arithmétiques, etc.). Nous nous sommes intéressés à la réduction naturelle impliquée par un tel élément absorbant : si nous prenons un arbre d'expression aléatoire, quelle est sa taille après avoir remplacé de bas en haut tous les sous-arbres de la forme $\overset{\circledast}{\underset{\mathcal{P}\ T}{/\!\backslash}}$ ou $\overset{\circledast}{\underset{T\ \mathcal{P}}{/\!\backslash}}$ par $\mathcal{P}$? Pour pouvoir étudier cette réduction, il est nécessaire de préciser le modèle aléatoire en répondant aux deux questions

suivantes :

a. comment sont décrits les arbres d'expressions étudiés ?

b. qu'entendons-nous par expression aléatoire ?

Prenons l'exemple des expressions régulières sur $\{a, b\}$. Leur syntaxe peut être décrite directement à partir de leur définition par la règle $E = a+b+\varepsilon+\overset{\star}{|}_{E}+\overset{+}{/\!\backslash}_{E\,E}+\overset{\cdot}{/\!\backslash}_{E\,E}$. Une étude expérimentale montre rapidement que les arbres générés uniformément à partir de cette description se simplifient considérablement, comme sur la Figure 1.6. On peut se demander si en limitant les redondances dans la description on peut échapper à cette faible expressivité des expressions aléatoires. Par exemple, le système suivant empêche l'apparition de deux étoiles de Kleene consécutives :

$$\begin{cases} \mathcal{L}_{\mathcal{R}} = \overset{\star}{|}_{\mathcal{S}} + \mathcal{S}, \\ \mathcal{S} = a + b + \varepsilon + \overset{+}{/\!\backslash}_{\mathcal{L}_{\mathcal{R}}\,\mathcal{L}_{\mathcal{R}}} + \overset{\bullet}{/\!\backslash}_{\mathcal{L}_{\mathcal{R}}\,\mathcal{L}_{\mathcal{R}}}. \end{cases} \tag{$\star\star$}$$

Pour obtenir un résultat le plus général possible, nous autorisons de tels systèmes en réponse à la question **a.** Nous montrons que cette dégénérescence est universelle pour la distribution uniforme, dès lors qu'il y a un élément absorbant.

Comme nous l'avons évoqué plus haut, il y a une autre réponse naturelle et utilisée en pratique à la question **b.**, qui est d'utiliser la distribution ABR à la place de la distribution uniforme. Dans ce cas, les comportements sont plus variés et dépend de la probabilité d'apparition de l'opérateur absorbant. Nous mettons en évidence l'existence de deux seuils et donc de cinq comportements, selon que l'on soit au-delà d'un seuil, sur un seuil, ou entre les deux.

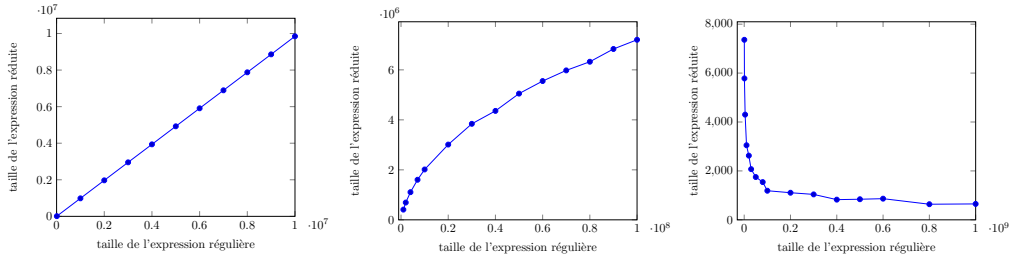


Figure C. Trois des cinq régimes observés expérimentalement sur des expressions régulières à deux lettres suivant la distribution ABR. Voir la chapitre 5 pour les détails.

Dans la seconde partie, nous proposons une extension de la correspondance entres les classes de langages formels et classes de séries génératrices aux séries holonomes. Ces travaux sont dans la lignée de ceux de Flajolet, en suivant une piste initiée par [Lip88] et [Mas93]. La classe d'automates que nous proposons est une extension des automates et grammaires hors-contexte avec des vecteurs d'entiers, bien connue dans le contexte de la vérification automatique de programmes. Comme pour les premiers niveaux, la correspondance capturera les versions non ambiguës de ces modèles. Dans un premier temps, nous montrons la robustesse de cette connexion en établissant l'équivalence des versions non ambiguës de plusieurs modèles d'automates (Reversal Bounded Counter Machines, Parikh automata, et leurs versions avec pile).

Ces équivalences étaient déjà connues pour les versions non déterministes.

Puis, en utilisant l'holonomie des séries associées à ces modèles d'automates, nous répondons à deux types de questions :

a. comment montrer qu'un langage n'est pas dans la classe de langages, c'est-à-dire qu'il n'est accepté par aucun automate non ambigu ?

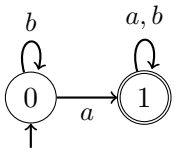
b. quel est le gain (en décidabilité ou complexité) apporté par l'utilisations de ces descriptions non ambiguës, par rapport à leurs versions non déterministes ?

Pour répondre à la première question, nous avons utilisé des critères de non holonomie pour établir des conditions suffisantes permettant de démontrer l'intrinsèque ambiguïté de certains langages. Pour la deuxième question, nous avons développé le problème de l'inclusion $L_{\mathcal{A}} \subseteq L_{\mathcal{B}}$. Ce problème est naturel dans le contexte de la vérification, où $\mathcal{A}$ est l'automate étudié, et l'automate $\mathcal{B}$ décrit les comportements que l'on autorise ; vérifier que le langage de $\mathcal{A}$ est inclus dans celui de $\mathcal{B}$, c'est vérifier que les comportements décrits par $\mathcal{A}$ restent autorisés. Le problème de l'inclusion est indécidable pour les langages algébriques, et les langages reconnus par des automates de Parikh, et bien qu'il soit toujours indécidable pour les langages algébriques non ambigus, il est décidable pour les automates de Parikh non ambigu. Si la décidabilité est une conséquence naturelle de l'holonomie des séries génératrices associées, nous sommes allés plus loin en analysant et adaptant les résultats de calcul formel sur les séries holonomes, pour en déduire un algorithme élémentaire, ainsi qu'une borne supérieure sur sa complexité.

Élémentaire dans le sens où il ne demande pas de connaître l'algorithmique des séries holonomes.

En calcul formel, les séries rationnelles, algébriques ou holonomes, ne sont pas toujours représentées sous une forme close, mais plutôt par des systèmes qui les définissent : une série algébrique est représentée par un système polynomial, une série holonome par un système d'équations différentielles…Les opérations effectuées sur ces séries correspondent en réalité à des manipulations sur les systèmes : lorsqu'on calcule la somme de deux séries holonomes $c(x) = a(x)+b(x)$, on calcule en fait une équation différentielle vérifiée par $c(x)$, à partir des équations différentielles de $a(x)$ et $b(x)$. Cela n'a rien de surprenant en informatique, et c'est en fait exactement ce que l'on fait en représentant un langage par un automate ou une grammaire. Lorsqu'on calcule l'union de deux langages, on calcule en fait un automate décrivant cette union. Cette analogie entre représentation de séries par des systèmes d'équations, et représentation finie des langages par des automates ou grammaires est au coeur de la correspondance entre séries et langages. Dans le cas des langages réguliers ou algébriques, la correspondance entre les deux est d'ailleurs presque syntaxique. En effet, un automate peut se traduire comme un système décrivant ses langages résiduels, les langages des mots acceptés depuis ses différents états. De même l'ensemble des règles de dérivation d'une grammaire hors-contexte décrivent naturellement les arbres de dérivations depuis chacun de ses non terminaux. Ces descriptions se traduisent automatiquement en un système sur les séries associées, comme on peut le voir dans la figure D. Suivant cette idée, nous avons analysé dans la seconde partie des systèmes d'équations différentielles de séries issues de la description d'automates à compteurs, comme les automates de Parikh ou leurs versions à pile. Nos objets d'étude sont donc décrits par des systèmes.

Dans les deux parties, notre objectif a été d'énoncer des résultats dans des cadres les plus généraux possibles. Pour ce faire, nous avons été amenés à définir puis étudier des systèmes combinatoires associés à des systèmes de séries génératrices. Les systèmes sont un outil formidable de la combinatoire, qui permettent de capturer de larges familles d'objets, au-delà des cas particuliers, tout en offrant une certaine régularité des propriétés liées aux objets décrits. Leur étude est permise par le fait que ces propriétés sont capturées par la description sous la forme d'un système : ainsi, il



$$\longrightarrow \begin{cases} L_0 = bL_0 + aL_1 \\ L_1 = \varepsilon + (a+b)L_1 \end{cases} \longrightarrow \begin{cases} L_0(x) = xL_0(x) + xL_1(x) \\ L_1(x) = 1 + Lxq_1(x) \end{cases}$$

$$\begin{cases} S \to S_1cS \,|\, S_2cS \,|\, \varepsilon \\ S_1 \to aS_1bS_1 \,|\, \varepsilon \\ S_2 \to dS_2d \,|\, eS_2e \,|\, \varepsilon \end{cases} \longrightarrow \begin{cases} S(x) = xS_1(x)S(x) + xS_2(x)S(x) + 1 \\ S_1(x) = x^2S_1(x)^2 + 1 \\ S_2(x) = 2x^2S_2(x) + 1 \end{cases}$$

Figure D. Exemple de traduction naturelle d'un automate ou d'une grammaire hors-contexte en un système sur leurs séries associées.

n'est pas nécessaire de savoir résoudre le système pour l'étudier, et la combinatoire analytique fournit tous les outils nécessaires pour à la fois décrire le comportement analytiques des séries associées aux objets combinatoire décrits par le système, ainsi que le comportement asymptotique de leurs coefficients. En ce sens, les systèmes constituent le thème fédérateur des deux parties.

1

Chapitre préliminaire

1.1 Préliminaires pour les langages formels

Dans cette section, nous introduisons les définitions de base des langages formels dont nous aurons besoin dans cette thèse. Pour une présentation plus détaillée, nous renvoyons le lecteur au très bon livre de [HMU01].

Un alphabet, généralement noté Σ, est un ensemble fini de lettres. Un mot w sur l'alphabet Σ est une suite finie $(w_1, \ldots, w_n)$ de lettres de Σ, notée en abrégé $w = w_1 \ldots w_n$, où n est la longueur du mot w, notée $|w|$, et $w_1, \ldots, w_n$ sont des lettres de Σ.

Par convention, la suite vide constitue un mot, noté ε et appelé mot vide, et sa longueur est 0.

L'ensemble des mots sur Σ est noté Σ^*. On peut munir Σ^* de l'opération de concaténation $\cdot$, définie par $w \cdot v = w_1 \ldots w_n v_1 \ldots v_m$, avec $w = w_1 \ldots w_n$ et $v = v_1 \ldots v_m$. Dans la suite, nous écrirons souvent wv, en omettant le symbole de concaténation.

Un *langage* est un ensemble (potentiellement infini) de mots sur un alphabet Σ.

1.1.1 Langages réguliers

Nous présentons très rapidement les définitions des langages réguliers et algébriques. Tous les résultats de cette section sont tirés de [Car08, HMU01].

Définition 1.1 (Langage rationnel).
▶ Soient L_1, L_2 deux langages sur un alphabet Σ. Les opérations $+$, $\cdot$ et $*$, appelées opérations rationnelles, sont définies comme suit :

- Le langage union, noté $L_1 + L_2$, désigne l'union des deux langages $L_1 \cup L_2$.
- Le langage produit, noté $L_1 \cdot L_2$ désigne l'ensemble des mots de la forme $w \cdot v$ avec $w \in L_1$ et $v \in L_2$.
- Nous définissons $L_1^0 = \{\varepsilon\}$, puis par récurrence, $L_1^{i+1} = L_1 \cdot L_1^i$ pour tout $i \in \mathbb{N}$.
- L'étoile de Kleene du langage L_1, noté L_1^*, désigne l'ensemble des mots obtenus en concaténant des mots de L_1 : plus précisément $L_1^* := \bigcup_{i \in \mathbb{N}} L_1^i$.

L'ensemble des *langages rationnels* est défini comme le plus petit ensemble de langages qui contient les langages finis et qui est stable par les opérations rationnelles. ◀

On peut décrire un langage rationnel par un automate fini, qui est une machine mathématique très simple. Il est bien connu que les langages rationnels sont exactement les langages acceptés par de tels automates finis.

Définition 1.2 (Automate fini).
▶ Un *automate fini non déterministe* est un tuple $\mathcal{A} = (\Sigma, Q, q_I, F, \Delta)$, où Σ désigne l'alphabet, Q l'ensemble d'états, $q_I \in Q$ est l'état initial, et $F \subseteq Q$ l'ensemble des états finaux. Enfin $\Delta \subseteq Q \times \Sigma \times Q$ est l'ensemble des transitions.

Un automate fini est déterministe si pour tout état q et toute lettre $a \in \Sigma$, il existe au plus une transition de la forme (q, a, q') dans Δ.

Un *calcul* de l'automate est une séquence $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots q_{n-1} \xrightarrow{a_n} q_n$ telle que pour tout $i \in [1, n]$, (q_{i-1}, a_i, q_i) est une transition de Δ. Un calcul de cette forme est étiqueté par le mot $a_1 \cdots a_n \in \Sigma^*$. Il est acceptant si $q_0 = q_I$, si l'état q_n est final. On dit alors que le mot w est accepté par $\mathcal{A}$.

Le *langage accepté* par $\mathcal{A}$ désigne l'ensemble des mots acceptés par l'automate, et est noté $\mathcal{L}(\mathcal{A})$.

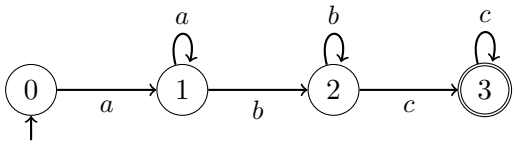
La notion de non-ambiguïté sera centrale dans ma thèse.

Un automate est *non ambigu* si tout mot accepté par l'automate étiquette un unique calcul acceptant de l'automate.

Les langages reconnus par un automate fini forment une famille de langages, appelés *langages réguliers* . ◀

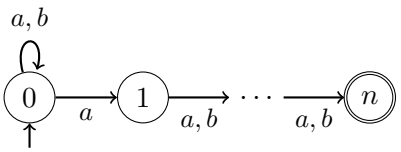
Exemple 1.3 (Le langage $a^+b^+c^+$).
▶ On représente généralement un automate fini sous la forme d'un multigraphe orienté étiqueté. Les états sont les nœuds du graphe, et les arêtes représentent les transitions. L'état initial est distingué par une flèche entrante, et les états finaux sont entourés deux fois.

Ainsi, le langage $L = \{a^n b^m c^p \mid n, m, p \in \mathbb{N}^*\}$ est reconnu par l'automate déterministe suivant :



◀

Exemple 1.4 (Le langage $\Sigma^* a \Sigma^{n-1}$).
▶ Soit $n \geqslant 1$. Le langage des mots dont la n-ième lettre en partant de la fin est un a est reconnu par l'automate fini non ambigu suivant :



Cet automate n'est pas déterministe. Il est classique que tout automate fini reconnaissant ce langage possède forcément au moins 2^n états. ◀

Proposition 1.5 (Voir par exemple [Car08, HMU01]).
▶ Chacune des trois classes d'automates finis, à savoir les automates finis non déterministes, les automates finis non ambigus, et les automate finis déterministes, reconnaît exactement les langages rationnels. ◀

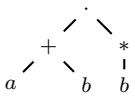
Nous définissons maintenant les expressions régulières, qui fournissent une autre façon de représenter par un objet fini un langage, qui est un objet potentiellement infini. Il se trouve que ce formalisme définit aussi exactement les langages rationnels.

Définition 1.6 (Expressions régulières).
▶ L'ensemble des expressions régulières (ainsi que les langages qu'elles décrivent) est défini par induction :

— Toute lettre de Σ est une expression régulière. Si $a \in \Sigma$, l'expression $E := a$ définit le langage $\{a\}$.
— L'expression $E := \emptyset$ est une expression régulière, désignant le langage vide. De même $E := \varepsilon$ est une expression régulière, représentant le langage $\{\varepsilon\}$.
— Si E_1 et E_2 sont deux expressions régulières, notons L_1 et L_2 les langages qu'elles décrivent. Alors :
  — $\overset{\cdot}{/\!\backslash}_{E_1\ E_2}$ est une expression régulière, représentant le langage $L_1 \cdot L_2$;
  — $\overset{+}{/\!\backslash}_{E_1\ E_2}$ est une expression régulière, représentant le langage $L_1 + L_2$;
  — $\overset{*}{|}_{E_1}$ est une expression régulière, représentant le langage L_1^*.

◀

Le langage décrit par une expression régulière est exactement ce à quoi nous pouvons nous attendre en évaluant l'arbre de façon inductive, en remplaçant les opérateurs par leurs opération ensemblistes associées. Une expression régulière est souvent décrite sous la forme d'une chaîne de caractère, pour plus de concision ; on utilise alors naturellement des parenthèses pour remplacer la structure d'arbre. Par exemple l'expression régulière suivante :



s'écrit $(a + b) \cdot b^*$, et désigne l'ensemble rationnel $(\{a\} \cup \{b\}) \cdot (\{b\}^*)$.

Proposition 1.7 (voir par exemple [Car08, HMU01]).
▶ Les langages rationnels sont exactement les langages reconnus par des automates finis. Ce sont aussi exactement les langages reconnus par des expressions régulières. ◀

1.1.2 Langages algébriques

Certains langages ne sont pas reconnaissables par des automates finis : c'est le cas par exemple du langage des mots de la forme $a^n b^n$ avec $n \in \mathbb{N}$. Les langages algébriques forment une famille de langages plus large qui inclut les langages réguliers.

Définition 1.8 (Grammaire hors-contexte).
▶ Une *grammaire hors-contexte* est un tuple $G = (N, \Sigma, S, D)$ où :

a. N est l'ensemble des non terminaux, Σ est l'ensembles des lettres terminales,

b. $S \in N$ est appelé l'axiome,

c. D est l'ensemble des règles de production de la grammaire, qui sont de la forme $A \to \alpha_1 \dots \alpha_r$, où $A \in N$, $r \geqslant 0$, et $\alpha_i \in N \cup \Sigma$ pour tout $1 \leqslant i \leqslant r$.

En voyant les règles de production de D comme des règles de réécriture, le langage de G, noté $\mathcal{L}(G)$ est l'ensemble des mots constitués de lettres terminales que l'on peut obtenir en partant de l'axiome S et en appliquant successivement des règles de réécriture de D.

Un langage reconnaissable par une grammaire hors-contexte est dit *algébrique*. ◀

Remarque 1.9 .

▶ Souvent, nous prendrons la convention de définir une grammaire particulière par son ensemble de règles de productions, en notant S l'axiome, les non terminaux par des lettres majuscules, et les lettres terminales par des lettres minuscules.

Il est courant aussi d'utiliser une notation compacte en rassemblant la description des règles de D selon le non terminal à gauche de la règle : ainsi $D = \{A \to a, A \to b\}$ est noté plus succinctement $D = \{A \to a \,|\, b\}$. ◀

Exemple 1.10 (Langage $a^n b^n$).

▶ Considérons la grammaire suivante :

$$S \to aSb \,|\, \varepsilon\,.$$

En partant de l'axiome S et en appliquant les règles de réécriture successivement, nous pouvons générer le mot vide, le mot ab et le mot $aabb$. Plus généralement le langage de la grammaire est l'ensemble des mots de la forme $a^n b^n$ avec $n \in \mathbb{N}$. ◀

Définition 1.11 (Dérivation d'une grammaire).

▶ Soit $G = (N, \Sigma, S, D)$ une grammaire hors-contexte. Une dérivation de G à partir d'un symbole $V \in N$ est une suite $w_0 = V \Rightarrow w_1 \Rightarrow \dots \Rightarrow w_r$, où chaque w_i est un mot de $(\Sigma \cup N)^*$, et pour tout $i \geqslant 1$, w_i est obtenu à partir de w_{i-1} en appliquant une règle de réécriture de la grammaire à un symbole non terminal.

Une dérivation est dite *terminale* si le dernier mot de la suite est un mot de Σ^* : il n'est plus possible d'appliquer de règles de dérivations. ◀

Exemple 1.12 .

Celle du milieu correspond à l'effacement du symbole S en appliquant $S \to \varepsilon$

▶ Les suites suivantes :

$$S \Rightarrow aSb \Rightarrow aaSbb \qquad S \Rightarrow \qquad S \Rightarrow aSb \Rightarrow ab$$

sont des dérivations de la grammaire définie par $S \to aSb \,|\, \varepsilon\,.$ ◀

Exemple 1.13 .

▶ Considérons la grammaire suivante des mots bien parenthésés :

$$S \to [S]S \,|\, \varepsilon\,.$$

Alors le mot [[][]] est bien parenthésé et est engendré par la dérivation terminale suivante :

$$S \Rightarrow [S]S \Rightarrow [[S]S]S \Rightarrow [[]S]S \Rightarrow [[][S]S]S \Rightarrow [[][]S]S \Rightarrow [[][]]S \Rightarrow [[][]]$$

Cette dérivation est appelée *dérivation gauche* car tout mot w_i de la suite est obtenu à partir du mot qui le précède en appliquant une règle de dérivation de la grammaire sur son non-terminal le plus à gauche.

Remarquons que ce mot possède d'autres dérivations, selon l'ordre des symboles non terminaux auxquels nous appliquons les règles de dérivation, par exemple :

$$S \Rightarrow [S]S \Rightarrow [S] \Rightarrow [[S]S] \Rightarrow [[]S] \Rightarrow [[][S]S] \Rightarrow [[][S]] \Rightarrow [[][]] \,.$$

Même si ces deux dérivations sont formellement distinctes, elle ne diffèrent en fait que dans l'ordre d'application des règles de dérivation. Nous utilisons plutôt la notion d'arbre de dérivation, pour éliminer les complications dues à l'ordre d'application des règles de dérivation. ◀

Définition 1.14 (Arbre de dérivation).
▶ Soit $G = (N, \Sigma, S, D)$ une grammaire hors-contexte. Pour tout $v \in N \cup \Sigma$, l'ensemble des arbres de dérivation de G enracinés en v, noté $Tree_G(v)$, est défini par induction :

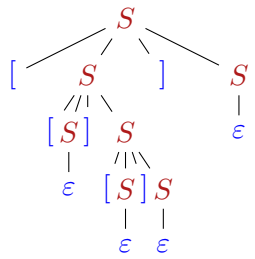
- pour $v \in N \cup \Sigma$, $v \in Tree_G(v)$;
- si $v \in N$ et $v \to v_1 \dots v_r \in D$ est une règle de dérivation, avec $r \geqslant 1$, et $v_i \in N \cup \Sigma$ et si $t_i \in Tree_G(v_i)$ pour tout $1 \leqslant i \leqslant r$, alors $(v, t_1, \dots, t_r) \in Tree_G(v)$.
- si $v \in N$ et $v \to \varepsilon \in D$ est une règle de dérivation, alors $(v, \varepsilon) \in Tree_G(v)$.

Nous notons $Tree_G = \bigcup_{v \in N \cup \Sigma} Tree_G(v)$ l'ensemble des arbres de dérivation de G.

Un arbre de dérivation est terminal si toutes ses feuilles sont des symboles de $\Sigma \cup \{\varepsilon\}$. ◀

Un arbre de dérivation permet de représenter une dérivation sans introduire d'ordre sur l'application des règles de dérivation. Il est facile de voir que les dérivations gauches terminales sont en bijection avec les arbres de dérivation terminaux, et que cette bijection préserve le mot généré.

Exemple 1.15 .
▶ L'arbre suivante est un arbre de dérivation de la grammaire $S \to [S]S \mid \varepsilon$ pour le mot $[[][]]$:



◀

Les langages algébriques sont reconnus par une extension des automates finis, les automates à pile : à chaque transition, l'automate peut lire le symbole stocké au sommet de sa pile, puis le dépiler ou empiler un nouveau symbole. L'automate bloque s'il essaie de dépiler ou de lire un symbole de sa pile alors que sa pile est vide : pour détecter cela dans l'automate, il est courant d'ajouter un symbole de fond de pile $\perp$ à la première transition de l'automate pour marquer le fond de la pile. Le langage

des mots de la forme $a^n b^n$ avec $n \in \mathbb{N}$ est par exemple reconnu en empilant dans la pile chaque lettre a lue, puis en dépilant la pile autant de fois qu'il y a de lettres b : l'automate va alors dans un état acceptant uniquement s'il atteint le symbole de fin de pile en lisant la dernière lettre b du mot.

Plus formellement :

Définition 1.16 (Automate à pile, voir par exemple [Car08]).
▶ Un *automate à pile* non déterministe est un tuple $\mathcal{A} = (\Sigma, \Gamma, Q, q_I, \bot, F, \Gamma, \Delta)$ où Σ désigne l'alphabet d'entrée, Γ désigne l'alphabet de pile, Q l'ensemble d'états, $q_I \in Q$ est l'état initial, $\bot$ est le symbole de fond de pile et $F \subseteq Q$ l'ensemble des états finaux. Enfin Δ est l'ensemble des transitions de la forme $q, z_1 \xrightarrow{a} q', z_2$ ou $q, z_1 \xrightarrow{\varepsilon} q', z_2$ avec $a \in \Sigma$, $q, q' \in Q$, $z_1 \in \Gamma \cup \{\varepsilon\}$ et $z_2 \in \Gamma^*$.

Une configuration de l'automate à pile est la donnée d'un tuple (q, w) avec $q \in Q$ et $w \in \Gamma^* \bot$ désigne le contenu de la pile. La configuration initiale de l'automate est $(q_0, \bot)$.

Soient (q, w) et (q', w') deux configurations. On dit que l'automate atteint la configuration (q', w') depuis la configuration (q, w) s'il existe une transition $t \in \Delta$, de la forme $t = q, z_1 \xrightarrow{x} q', z_2$ avec $x \in \Sigma \cup \{\varepsilon\}$ et telle que $w \in z_1\Gamma^*$ et $w' = z_2 w_1$, en notant $w = z_1 w_1$ avec $w_1 \in \Gamma^*$. Dans ce cas, on dit que $(q, w) \xrightarrow{x} (q', w')$ est une transition valide de l'automate, étiquetée par la lettre $x \in \Sigma \cup \{\varepsilon\}$.

On prend comme convention qu'un automate à pile ne dépile jamais le symbole de fin de pile $\bot$: pour tout transition $t = q, z_1 \xrightarrow{x} q', z_2$ dans Δ, si $z_1 = \bot$ alors $z_2 \in \Gamma^* \bot$ (la transition réempile le symbole de fond de pile).

Un calcul de $\mathcal{A}$ est une suite de transitions valides de la forme $(q_0, w_0) \xrightarrow{x_1} (q_1, w_1) \xrightarrow{x_2} \dots \xrightarrow{x_n} (q_n, w_n)$. Le mot étiquetant un tel calcul est la concaténation $x_1 \dots x_n \in \Sigma^*$ des lettres étiquetant les transitions. Un calcul est acceptant si $q_0 = q_I$ et $q_n \in F$.

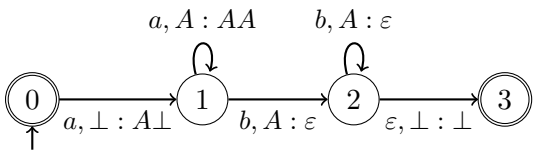
Le *langage accepté* par $\mathcal{A}$ désigne l'ensemble des mots acceptés par l'automate, et est noté $\mathcal{L}(\mathcal{A})$.

Un automate à pile est *non ambigu* si tout mot accepté par l'automate étiquette un unique calcul acceptant de l'automate. ◀

Proposition 1.17 (Voir par exemple [Car08]).
▶ Les automates à pile reconnaissent exactement les langages algébriques : un langage est reconnu par une grammaire hors-contexte si et seulement si il est reconnu par un automate à pile. ◀

Remarque 1.18 (Notation d'un automate à pile).
▶ Une transition de la forme $t = q, z_1 \xrightarrow{x} q', z_2$ est notée sur un automate par une flèche de l'état q à l'état q' d'étiquette $x, z_1 : z_2$. ◀

Exemple 1.19 (Automate à pile reconnaissant $a^n b^n$).
▶ L'automate à pile suivant reconnaît le langage $\{a^n b^n \mid n \in \mathbb{N}\}$:



◀

Remarque 1.20.
▶ Contrairement au cas des langages rationnels, les automates à pile non ambigus sont strictement moins expressifs que les automates à pile non déterministes. Il existe des langages algébriques qui ne sont reconnus par aucun automate à pile non ambigu. ◀

Nous parlerons plus en détail de ces langages, appelés intrinsèquement ambigus, dans le chapitre 7.

1.1.3 Ensembles semilinéaires

Un des objectifs de cette thèse est d'étendre certains résultats sur les langages algébriques, à d'autres familles de langages plus larges. Dans notre contexte, pour étendre les langages algébriques, nous aurons besoin de la notion d'ensembles semilinéaires.

Les ensembles semilinéaires sont à $\mathbb{N}^d$ ce que les langages rationnels sont à Σ^*. Ce sont des ensembles fondamentaux qui apparaissent naturellement en théorie des langages. Dans cette section nous présentons les définitions et propriétés de base des ensembles semilinéaires. Nous renvoyons le lecteur au *guide de survie de l'arithmétique de Presburger* de [Haa18] pour un excellent survey plus complet sur les ensembles semilinéaires. Dans la suite nous fixons d une dimension, et nous nous plaçons dans l'ensemble $\mathbb{N}^d$ des vecteurs à coordonnées entières.

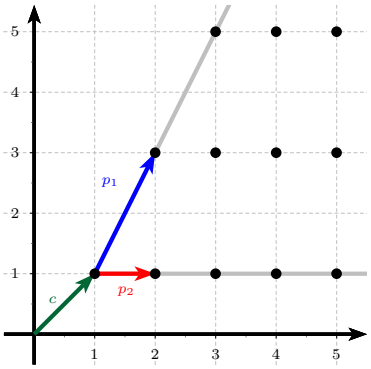


Figure 1.1. Représentation en dimension 2 d'un ensemble linéaire $\boldsymbol{c} + \{\boldsymbol{p_1}, \boldsymbol{p_2}\}^*$

Définition 1.21 (Ensemble semilinéaire, [Par66]).
▶ Un ensemble $L \subseteq \mathbb{N}^d$ est dit *linéaire* s'il existe un vecteur $\boldsymbol{c} \in \mathbb{N}^d$, appelé constante, et un ensemble fini de vecteurs $P = \{\boldsymbol{p_1}, \ldots, \boldsymbol{p_r}\}$ appelé ensemble des périodes, tel que $L = \{\boldsymbol{c} + \sum_{i=1}^{r} \lambda_i \boldsymbol{p_i} : (\lambda_i) \in \mathbb{N}^r\}$. Cet ensemble est alors noté $L = \boldsymbol{c} + P^*$.

Un ensemble $S \subseteq \mathbb{N}^d$ est *semilinéaire* s'il est une union finie d'ensembles linéaires, c'est-à-dire qu'il peut s'écrire sous la forme :

$$S = \bigcup_{j=1}^{k} (\boldsymbol{c_j} + P_j^*) \,.$$

◀

Exemple 1.22.
▶ En dimension 2, l'ensemble des vecteurs à coordonnées paires est linéaire : c'est $((2,0) + (0,2))^*$. ◀

Définition 1.23 (Image de Parikh, [Par66]).
▶ Nous considérons un alphabet fini $\Sigma = \{a_1, \ldots, a_d\}$, dont les lettres sont ordonnées. Pour $w \in \Sigma^*$, et $a \in \Sigma$, nous notons $|w|_a$ le nombre d'occurrences de la lettre a dans w. Le vecteur de Parikh d'un mot $w \in \Sigma^*$, notée $\boldsymbol{\pi_{kh}}(w)$ est le vecteur

Il s'agit donc simplement de compter le nombre d'occurrences de chaque lettre.

$$\boldsymbol{\pi_{kh}}(w) := (|w|_{a_1}, \ldots, |w|_{a_d}) \in \mathbb{N}^d .$$

Par exemple $\boldsymbol{\pi_{kh}}(abaacb) = (3, 2, 1)$ sur l'alphabet (ordonné) $\Sigma = \{a, b, c\}$.

Si L est un langage sur Σ, l'image de Parikh de L désigne l'ensemble des vecteurs de Parikh des mots du langage L ◀

Le théorème suivant explique pourquoi les ensembles semilinéaires ont une place prédominante en informatique, notamment dans la théorie des langages formels :

Théorème 1.24 (Théorème de Parikh, [Par66]).
▶ L'image de Parikh d'un langage algébrique est un ensemble semilinéaire. De plus pour tout langage algébrique L, il existe un langage rationnel qu'on peut calculer effectivement et qui a la même image de Parikh que L. ◀

Exemple 1.25 .
▶ Le langage algébrique $\{a^n b^n \mid n \in \mathbb{N}\}$ a la même image de Parikh que le langage régulier $(ab)^*$. ◀

Les ensemble semilinéaires jouissent de nombreuses propriétés de clôture :

Proposition 1.26 (Clôture, [GS64, GS66]).
▶ Les ensembles semilinéaires sont stables par union, intersection, produit cartésien, complémentaire, projection. ◀

Les ensembles semilinéaires sont par ailleurs exactement les ensembles descriptibles par l'arithmétique de Presburger.

Définition 1.27 (Langage des termes linéaires).
▶ Soit $\mathcal{X}$ un ensemble de variables. Le langage des termes linéaires sur $\mathcal{X}$ est défini par induction :

- si $x \in \mathcal{X}$, alors x est un terme linéaire sur $\mathcal{X}$;
- 0 et 1 sont des termes linéaires sur $\mathcal{X}$;
- si t, t' sont des termes linéaires sur $\mathcal{X}$, alors $t + t'$ est un terme linéaires sur $\mathcal{X}$.

Pour $n \in \mathbb{N}$ fixé, et t un terme arithmétique, nous notons $n \cdot t$ le terme $\underbrace{t + \ldots + t}_{n \text{ fois}}$. Par ailleurs, nous notons n le terme $n \cdot 1$. Il s'agit d'une notation purement syntaxique, nous n'autorisons par la multiplication entre deux termes. ◀

De façon informelle, si nous considérons que l'addition syntaxique $+$ est commutative, un terme linéaire peut être représenté sous la forme $b_0 \cdot 0 + b_1 \cdot 1 + c_1 \cdot x_1 + \ldots + c_n \cdot x_n$ où $b_0, b_1, c_1, \ldots, c_n$ sont des entiers de $\mathbb{N}$, 0 et 1 sont les termes constants de la syntaxe des termes linéaires, et $x_1, \ldots, x_d$ sont des variables.

Définition 1.28 (Formule de Presburger, [Pre29]).
▶ Soit $\mathcal{X}$ un ensemble de variables. Une *formule de Presburger* Φ est une formule du

premier ordre sur le langage des termes arithmétiques sur $\mathcal{X}$. Plus précisément, une formule de Presburger suit la syntaxe :

$$\Phi := t \leqslant t' \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \neg\Phi \mid \exists x\, \Phi$$

avec t, t' des termes arithmétiques sur $\mathcal{X}$, et x une variable de $\mathcal{X}$.

Nous autorisons des notations syntaxiques supplémentaires pour simplifier l'écriture de formules :

- $\forall x\, \Phi := \neg\exists x\, \neg\Phi$
- pour tout $k \in \mathbb{N}$ *fixé*, $t \equiv_k 0 := \exists y\, t = k \cdot y$, avec y une variable fraîche
- $t = t' := t \leqslant t' \wedge t' \leqslant t$
- $t \neq t' := \neg\, t = t'$
- $\Phi_1 \to \Phi_2 := \Phi_2 \wedge \neg\Phi_1$

Une formule de Presburger Φ avec d variables libres $x_1, \ldots, x_d$ sera notée sous la forme $\Phi(x_1, \ldots, x_d)$.

La sémantique des formules de Presburger est celle de l'arithmétique du premier ordre sur les entiers, sans la multiplication. Un terme linéaire sans variable est ainsi interprété comme un entier de $\mathbb{N}$. Si $\Phi(x_1, \ldots, x_d)$ est une formule de Presburger, et $(n_1, \ldots, n_d) \in \mathbb{N}^d$, nous notons $\Phi[n_1, \ldots, n_d]$ l'évaluation de la formule en substituant n_1 à la variable x_1, n_2 à la variable x_2, …, et n_d à la variable x_d.

Un ensemble $S \subseteq \mathbb{N}^d$ est défini par une formule de Presburger Φ si S est l'ensemble des vecteurs d'entiers $(n_1, \ldots, n_d) \in \mathbb{N}^d$ tel que $\Phi[n_1, \ldots, n_d]$ soit vrai ; autrement dit $(n_1, \ldots, n_d) \in S$ si et seulement si $\Phi[n_1, \ldots, n_d]$ est vrai.

Deux formules de Presburger à d variables libres sont dites *équivalentes* si elles définissent le même ensemble. ◀

Exemple 1.29.

▶ L'ensemble des vecteurs en dimension 2 à coordonnées paires est défini par la formule de Presburger $\Phi(x, y) := \exists x' \exists y'\, 2x' = x \wedge 2y' = y$.

En utilisant la macro $\equiv_2$, nous pouvons donner une formule plus concise : $\Phi'(x, y) := x \equiv_2 0 \wedge y \equiv_2 0$. ◀

Proposition 1.30 (Élimination des quantificateurs, [Coo72]).

▶ Toute formule de Presburger est équivalente à une formule de Presburger sans quantificateur, à condition d'autoriser les égalités modulo k pour tout $k \in \mathbb{N}$ dans la syntaxe des formules. Plus précisément, toute formule de Presburger est équivalente à une formule du premier ordre sur les entiers suivant la syntaxe :

$$\Phi := t \leqslant t' \mid (t \equiv_k 0)_{k\in\mathbb{N}} \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \neg\Phi$$

◀

Remarque 1.31.

▶ L'introduction de la famille de prédicats $(t \equiv_k 0)_{k\in\mathbb{N}}$ est nécessaire pour éliminer les quantificateurs. On vérifie facilement qu'il est impossible de décrire l'ensemble des nombres pairs sans utiliser de quantificateur ni de prédicats de la forme $t \equiv_k 0$. ◀

Proposition 1.32 (Équivalence des modèles, [GS66]).

▶ Un ensemble $S \subseteq \mathbb{N}^d$ est semilinéaire si et seulement si il est défini par une formule de Presburger. ◀

Les ensembles semilinéaires sont aussi reliés aux sous-ensembles rationnels de $\mathbb{N}^d$:

Proposition 1.33 ([ES69]).
▶ Les ensembles semilinéaires de $\mathbb{N}^d$ sont exactement les sous-ensembles rationnels du monoïde commutatif $(\mathbb{N}^d, +)$; ce sont exactement les ensembles reconnaissables par des automates finis étiquetés par des vecteurs de $\mathbb{N}^d$. ◀

La propriété suivante des semilinéaires est très importantes pour la suite de ma thèse :

Proposition 1.34 (Représentation non ambiguë d'un semilinéaire, [Ito69, ES69]).
▶ Tout ensemble semilinéaire $S \subseteq \mathbb{N}^d$ peut s'écrire sous la forme d'une union disjointe de semilinéaire dont les périodes sont linéairement indépendantes. Autrement dit tout semilinéaire admet une représentation :

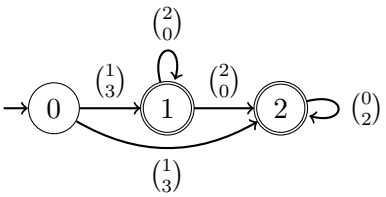
$$\biguplus_{j=1}^{r} (\boldsymbol{c_j} + P_j^*)$$

où pour tout j, k, $(\boldsymbol{c_j} + P_j^*) \cap (\boldsymbol{c_k} + P_k^*) = \emptyset$, et pour tout j, les vecteurs de P_j sont linéairement indépendants sur $\mathbb{Q}$. ◀

Ce qui est équivalent à être linéairement indépendants sur $\mathbb{Z}$

Cette dernière propriété permet notamment de reconnaître tout ensemble semilinéaire par un automate fini *non ambigu* sur le monoïde $(\mathbb{N}^d, +)$.

Exemple 1.35 (Automate fini reconnaissant un ensemble linéaire).
▶ L'ensemble linéaire $(1,3) + ((2,0) + (0,2))^*$ est reconnu par l'automate non ambigu suivant :



◀

1.2 Préliminaires de combinatoire analytique

Dans cette section, je présente rapidement quelques outils de combinatoire analytique que j'ai utilisés dans ma thèse. La combinatoire analytique étudie des ensembles combinatoires en s'appuyant sur les propriétés analytiques de leurs séries génératrices, vues comme des fonctions complexes. Ces préliminaires sont largement inspirés de [FS09].

1.2.1 Série formelle, série génératrice

Un objet mathématique central en combinatoire est la série formelle. Nous présentons ici les définitions et résultats fondamentaux sur ces objets qui nous seront utiles dans la suite, avant de les relier aux objects combinatoires dans la section suivante.

Nous notons $\mathbb{Q}$ l'ensemble des rationnels, $\mathbb{R}$ l'ensemble des réels, et $\mathbb{C}$ l'ensemble des nombres complexes. Pour $\mathbb{K} = \mathbb{Q}$, $\mathbb{R}$ ou $\mathbb{C}$, et $d \geqslant 1$, nous notons $\mathbb{K}[x_1, \ldots, x_d]$ l'ensemble des polynômes à coefficients dans $\mathbb{K}$, en les indéterminées $x_1, \ldots, x_d$. Nous notons $\mathbb{K}[[x_1, \ldots, x_d]]$ l'ensemble des séries formelles à coefficients dans $\mathbb{K}$ en les indéterminées $x_1, \ldots, x_d$, c'est-à-dire l'ensemble des polynômes formels infinis de la forme

$$\sum_{(i_1,\ldots,i_d)\in\mathbb{N}^d} a(i_1, \ldots, i_d) x_1^{i_1} \ldots x_d^{i_d}$$

avec $a(i_1, \ldots, i_d) \in \mathbb{K}$ pour tout $(i_1, \ldots, i_d) \in \mathbb{N}^d$. Une série formelle peut être vue comme l'encodage d'une suite $a(i_1, \ldots, i_d)$ indexée sur $\mathbb{N}^d$, sous la forme d'un polynôme infini.

Définition 1.36 (Série génératrice d'une suite).
▶ Soit $a(i_1, \ldots, i_d)$ une suite de $\mathbb{K}$ indexée sur $\mathbb{N}^d$. La série génératrice de $a(i_1, \ldots, i_d)$ est la série formelle définie par :

$$A(x_1, \ldots, x_d) = \sum_{(i_1,\ldots,i_d)\in\mathbb{N}^d} a(i_1, \ldots, i_d) x_1^{i_1} \ldots x_d^{i_d}\,.$$

◀

Nous présentons maintenant plusieurs opérations très utiles sur les séries, que nous relierons plus tard à des constructions ensemblistes sur ces classes combinatoires.

Définition 1.37 (Notation crochet, opérations sur les séries).
▶ Si $A(x_1, \ldots, x_d)$ est la série génératrice d'une suite $a(i_1, \ldots, i_d)$, la notation crochet $[x_1^{i_1} \ldots x_d^{i_d}]A(x_1, \ldots, x_d)$ désigne le coefficient $a(i_1, \ldots, i_d)$.

Pour simplifier les notations, nous notons $\boldsymbol{x}$ le tuple $(x_1, \ldots, x_d)$, $A(\boldsymbol{x})$ la série $A(x_1, \ldots, x_d)$.

L'ensemble des séries formelles $\mathbb{K}[[x_1, \ldots, x_d]]$ est stable par addition $+$, produit de Cauchy $\cdot$ et produit d'Hadamard $\odot$, où ces opérations sont définies comme suit : si $A(\boldsymbol{x})$ et $B(\boldsymbol{x})$ sont des séries de $\mathbb{K}[[x_1, \ldots, x_d]]$ alors :

— la série $A(\boldsymbol{x}) + B(\boldsymbol{x})$ est la série de $\mathbb{K}[[x_1, \ldots, x_d]]$ définie par

$$[x_1^{i_1} \ldots x_d^{i_d}](A(\boldsymbol{x}) + B(\boldsymbol{x})) = [x_1^{i_1} \ldots x_d^{i_d}]A(\boldsymbol{x}) + [x_1^{i_1} \ldots x_d^{i_d}]B(\boldsymbol{x})$$

— la série $A(\boldsymbol{x}) \odot B(\boldsymbol{x})$, appelée *produit d'Hadamard* des deux séries, est définie par

$$[x_1^{i_1} \ldots x_d^{i_d}](A(\boldsymbol{x}) \odot B(\boldsymbol{x})) = [x_1^{i_1} \ldots x_d^{i_d}]A(\boldsymbol{x}) \cdot [x_1^{i_1} \ldots x_d^{i_d}]B(\boldsymbol{x})$$

(c'est la multiplication terme à terme).

— la série $A(\boldsymbol{x}) \cdot B(\boldsymbol{x})$, appelée *produit de Cauchy* des deux séries, est définie par

$$[x_1^{i_1} \ldots x_d^{i_d}](A(\boldsymbol{x}) \cdot B(\boldsymbol{x})) := \sum_{\substack{j_1+k_1=i_1 \\ \cdots \\ j_d+k_d=i_d}} [x_1^{j_1} \ldots x_d^{j_d}]A(\boldsymbol{x}) \cdot [x_1^{k_1} \ldots x_d^{k_d}]B(\boldsymbol{x})$$

(c'est la généralisation aux séries de la multiplication des polynômes).

L'ensemble des séries formelles $(\mathbb{K}[[x_1, \ldots, x_d]], +, \cdot)$ a une structure d'anneau intègre. ◀

Remarque 1.38 (Notation condensée).
▶ En notant $\boldsymbol{i}$ le vecteur $(i_1, \dots, i_d)$, et $\boldsymbol{x}^{\boldsymbol{i}}$ le monôme $x_1^{i_1} \dots x_d^{i_d}$, nous pouvons simplifier la notation crochet $[x_1^{i_1} \dots x_d^{i_d}]A(x_1, \dots, x_d)$ en $[\boldsymbol{x}^{\boldsymbol{i}}]A(\boldsymbol{x})$. Par exemple, avec cette notation, le produit de Cauchy est alors défini par

$$[\boldsymbol{x}^{\boldsymbol{i}}](A(\boldsymbol{x}) \cdot B(\boldsymbol{x})) := \sum_{\boldsymbol{j}+\boldsymbol{k}=\boldsymbol{i}} [\boldsymbol{x}^{\boldsymbol{j}}]A(\boldsymbol{x}) \cdot [\boldsymbol{x}^{\boldsymbol{k}}]B(\boldsymbol{x})\,.$$

◀

Exemple 1.39.
▶ La série génératrice de la suite définie par $a_n = 1$ pour tout $n \in \mathbb{N}$ est la série formelle $A(x) = \sum_{n\in\mathbb{N}} x^n$. Cette série est notée $\frac{1}{1-x}$ car elle vérifie $A(x)(1-x) = 1$.
La série génératrice de la suite définie par $a_n = 2^n$ pour tout $n \in \mathbb{N}$ est la série formelle $A(x) = \sum_{n\in\mathbb{N}} 2^n x^n$. De même, cette série est notée $\frac{1}{1-2x}$. ◀

Nous notons $\mathbb{K}(x_1, \dots, x_d)$ l'ensemble des fractions rationnelles en les variables $x_1, \dots, x_d$, c'est-à-dire l'ensemble des fractions de la forme $\frac{P(x_1,\dots,x_d)}{Q(x_1,\dots,x_d)}$ avec P, Q des polynômes de $\mathbb{K}[x_1, \dots, x_d]$ et $Q \neq 0$.

1.2.2 Série rationnelle, série algébrique, série holonome

Dans cette section, nous fixons une dimension $d \geqslant 1$, et $\mathbb{K} = \mathbb{Q}$. Les séries étudiées sont donc à coefficients rationnels.

Définition 1.40 (Série rationnelle, série algébrique).
▶ Une série $F \in \mathbb{Q}[[x_1, \dots, x_d]]$ est appelée rationnelle s'il existe des polynômes $P, Q \in \mathbb{Q}[x_1, \dots, x_d]$ tels que $QF = P$ avec $Q \neq 0$.
Une série $F \in \mathbb{Q}[[x_1, \dots, x_d]]$ est appelée algébrique s'il existe un polynôme non nul $P \in \mathbb{Q}[Y, x_1, \dots, x_d]$ tels que $P(F, x_1, \dots, x_d) = 0$. ◀

Exemple 1.41.
▶ Toute série rationnelle est algébrique : si $F(\boldsymbol{x}) = P(\boldsymbol{x})/Q(\boldsymbol{x})$, alors le polynôme $P_2(Y, \boldsymbol{x}) = YQ(\boldsymbol{x}) - P(\boldsymbol{x})$ annule F.
Par exemple $\frac{1}{1-x_1x_2} = \sum_{n\geqslant 0} x_1^n x_2^n$ est rationnelle et vérifie $(1-x_1x_2)A(x_1, x_2) = 1$.
La série $A(x_1, x_2) = \sum_{n\in\mathbb{N}} \frac{1}{2^{2n}(1-2n)} \binom{2n}{n} x_1^n x_2^n$ est algébrique, car elle vérifie l'équation $A(x_1, x_2)^2 + (x_1x_2 - 1) = 0$; elle est généralement notée pour cette raison $\sqrt{1 - x_1x_2}$. Cette série n'est en revanche pas rationnelle. ◀

Remarque 1.42.
▶ En extrayant les coefficients de x^n pour $n > \deg(P)$ dans l'équation $Q(x)F(x) = P(x)$, nous remarquons que les coefficients d'une série rationnelle vérifient une équation linéaire à coefficients constants. Cela se généralise facilement à plusieurs variables. ◀

Remarque 1.43 (Racine carrée).
▶ Pour toute série formelle $A = \sum_n a_n x^n$, nous notons $\nu(A)$ la valuation de A définie par $\nu(A(x)) = \min\{n \mid a_n \neq 0\}$.

L'espace des séries formelles muni de la distance d, définie par $d(A,B) = 2^{-\nu(A-B)}$, est un espace métrique complet. Ainsi, si $a_0 = 0$, alors la suite de séries formelles $\sum_{k=0}^{N} A(x)^k$ converge lorsque $N \to +\infty$. Nous notons alors $\frac{1}{1-A(x)} := \sum_{k=0}^{+\infty} A(x)^k$.

De même, si $a_0 = 0$, alors la suite de séries $\sum_{n=0}^{N} \frac{1}{2^{2n}(1-2n)}\binom{2n}{n} A(x)^n$ converge, et nous notons alors :

$$\sqrt{1-A(x)} := \sum_{n=0}^{+\infty} \frac{1}{2^{2n}(1-2n)}\binom{2n}{n} A(x)^n \,.$$

◀

Nous introduisons à présent les séries holonomes. La classe des séries holonomes est une classe plus grande que celle des séries algébriques. Une très bonne introduction aux fonctions holonomes à une variable peut être trouvée dans [Sta80], et pour les fonctions multivariées, dans [Lip88] et [Lip89] ; nous citons aussi le début de [Chy94], qui détaille certaines preuves plus en détail.

L'opérateur différentiel ∂_{x_k} définit la dérivée partielle d'une série par rapport à la variable x_k. Il s'agit simplement de la généralisation de la dérivation des polynômes aux séries. Elle est définit formellement par

$$\partial_{x_k} A(\boldsymbol{x}) = \sum_{(i_1,\ldots,i_d)\in\mathbb{N}^d} i_k\, a(i_1,\ldots,i_d) x_1^{i_1}\ldots x_{k-1}^{i_{k-1}} x_k^{i_k-1} x_{k+1}^{i_{k+1}}\ldots x_d^{i_d}.$$

La dérivée partielle d'ordre supérieur $\partial_{x_k}^j$ est définie par récurrence : si $j \geqslant 1$ alors $\partial_{x_k}^1 := \partial_{x_k}$ et $\partial_{x_k}^{j+1} := \partial_{x_k} \circ \partial_{x_k}^j$.

Définition 1.44 (Série holonome, [Lip89]).
▶ Soit $A(x_1,\ldots,x_d) = \sum_{(i_1,\ldots,i_d)\in\mathbb{N}^d} a(i_1,\ldots,i_d) x_1^{i_1}\ldots x_d^{i_d} \in \mathbb{Q}[[x_d,\ldots,x_d]]$. La série $A(\boldsymbol{x})$ est *holonome* si elle vérifie un système d'équations aux dérivées partielles de la forme :

$$P_{i,n_i}(\boldsymbol{x})\partial_{x_i}^{n_i} A(\boldsymbol{x}) + \ldots + P_{i,1}(\boldsymbol{x})\partial_{x_i} A(\boldsymbol{x}) + P_{i,0}(\boldsymbol{x}) A(\boldsymbol{x}) = 0 \text{ pour chaque } i \in [d],$$

où chaque $n_i \in \mathbb{N}$ est un entier, et chaque $P_{i,j}(\boldsymbol{x})$ est un polynôme dans $\mathbb{Q}[\boldsymbol{x}]$.

Autrement dit pour chaque variable x_i, $A(\boldsymbol{x})$ satisfait une équation différentielle partielle en la variable x_i, qui est linéaire, et à coefficients polynomiaux en $\boldsymbol{x} = (x_1,\ldots,x_d)$.

De façon équivalente, une série $A(\boldsymbol{x})$ est holonome si le $\mathbb{Q}(x_1,\ldots,x_d)$-espace vectoriel engendré par la famille $\{\partial_{x_1}^{n_1}\ldots\partial_{x_d}^{n_d} A(x_1,\ldots,x_d) \;:\; (n_1,\ldots,n_d)\in\mathbb{N}^d\}$ est de dimension finie.

Nous noterons $\mathcal{H}$ l'ensemble des séries holonomes. ◀

Remarque 1.45 .
▶ Les séries holonomes sont aussi appelées D-finite. Les deux appellations sont utilisées indifféremment dans le cas des séries de $\mathbb{Q}[[x_1,\ldots,x_d]]$, cependant leur définition n'est historiquement pas la même. La définition que nous avons donnée est en réalité celle des séries D-finite, qui est plus simple à comprendre. La définition originale des séries holonomes implique la vitesse de croissance de la dimension

d'une famille d'espaces engendrés par des dérivées partielles, mais elle dépasse le cadre de cette thèse et nous n'en aurons pas besoin dans la suite. L'équivalence avec les séries D-finites provient de résultats profonds de Bernšteĭn [Ber72] et Kashiwara [Kas78, Tak92]. ◀

Exemple 1.46.
▶ Toute série rationnelle $A(\boldsymbol{x}) = \frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})}$ vérifie l'équation

$$P(\boldsymbol{x})Q(\boldsymbol{x})\partial_{x_i}A(\boldsymbol{x}) - (Q(\boldsymbol{x})\partial_{x_i}P(\boldsymbol{x}) - P(\boldsymbol{x})\partial_{x_i}Q(\boldsymbol{x}))A(\boldsymbol{x}) = 0$$

pour tout $i \in [d]$. Donc toute série rationnelle est holonome. ◀

Exemple 1.47.
▶ Nous verrons dans le chapitre 7 des exemples de séries qui ne sont pas holonomes, comme par exemple la série $\sum_{n\in\mathbb{N}} x^{n^2}$. ◀

Proposition 1.48 (Propriétés de clôture [Com64, Sta80, Lip88, Lip89]).
▶ Les séries holonomes sont closes sous de nombreuses opérations :

a. $\mathcal{H}$ est une sous-algèbre de $\mathbb{Q}[[\boldsymbol{x}]]$ (donc est close par addition, multiplication) ;

b. $\mathcal{H}$ contient toutes les séries algébriques ;

c. Si $A(\boldsymbol{x}) \in \mathcal{H}$ est une série holonome, et pour $\boldsymbol{y} = (y_1, \ldots, y_m)$, les séries $g_i(\boldsymbol{y}) \in \mathbb{Q}[[\boldsymbol{y}]]$ sont algébriques et telles que la composition $B(y_1, \ldots, y_m) := A(g_1(\boldsymbol{y}), \ldots, g_d(\boldsymbol{y}))$ ait un sens dans l'anneau des séries formelles, alors $B(\boldsymbol{y})$ est holonome ;

d. Si $A(\boldsymbol{x})$ et $B(\boldsymbol{x})$ sont deux séries holonomes ayant le même nombre de variables, alors $A(\boldsymbol{x}) \odot B(\boldsymbol{x})$ est holonome ;

◀

La stabilité par produit d'Hadamard est remarquable car ni la classe des séries rationnelles, ni celle des séries algébriques n'est close par cette opération.

Les séries holonomes possèdent aussi une caractérisation concernant leurs coefficients. À une seule variable, cette caractérisation est assez simple à comprendre :

Proposition 1.49 (Récurrence linéaire à coefficients polynomiaux, [Sta99]).
▶ Soit $A(x) = \sum_{n\in\mathbb{N}} a_n x^n \in \mathbb{Q}[[x]]$. Alors $A(x)$ est holonome si et seulement si la suite de ses coefficients (a_n) satisfait une équation linéaire de récurrence à coefficients polynomiaux de la forme :

$$p_r(n)a_{n+r} + \ldots + p_0(n)a_n = 0 \quad \text{pour tout } n \in \mathbb{N}. \qquad ◀$$

À plusieurs variables, la généralisation des récurrences linaires à coefficients polynomiaux est un peu plus compliquée.

Définition 1.50 (Section de suite, [Lip89]).
▶ Soit $a(i_1, \ldots, i_d)$ une suite de dimension d à valeurs dans $\mathbb{Q}$.

— On appelle *section* de $a(i_1, \ldots, i_d)$ toute suite de dimension strictement inférieure à d obtenue à partir de $a(i_1, \ldots, i_d)$ en fixant la valeur de certaines coordonnées.

— Une *k-section* de $a(i_1, \dots, i_d)$ est une section obtenue en fixant certaines coordonnées à des entiers strictement plus petit que k. Par exemple, $b(i_1, i_3) = a(i_1, 7, i_3, 5)$ est une 8-section de $a(i_1, i_2, i_3, i_4)$.

◀

Définition 1.51 (Suite P-récursive, [Lip89]).
▶ Une suite $a(i_1, \dots, i_d)$ est *P-récursive* s'il existe un entier $k \in \mathbb{N}$ tel que :

a. pour toute coordonnée $j \in [d]$, pour tout vecteur borné $\nu = (\nu_1, \nu_2, \dots, \nu_d) \in [k]^d$, il existe un polynôme $p_\nu^{(j)}$ non nul tel que

$$\forall i_1, i_2, \dots, i_d \geqslant k, \sum_{\nu \in [k]^d} p_\nu^{(j)}(i_j) a(i_1 - \nu_1, i_2 - \nu_2, \dots, i_d - \nu_d) = 0$$

b. si $d > 1$, toutes les k-sections de a sont P-récursives. ◀

Remarquons que si $d = 1$, cette définition coïncide avec la caractérisation que nous avons vue en dimension 1. Nous n'aurons pas vraiment besoin de cette caractérisation dans la suite, donc nous ne développons pas plus cette définition, si ce n'est pour mentionner qu'elle permet de généraliser en dimension supérieure la caractérisation des séries holonomes par leurs coefficients :

Théorème 1.52 (Équivalence des deux notions, [Lip89]).
▶ La série $A(\boldsymbol{x}) = \sum a(i_1, \dots, i_d) x_1^{i_1} \dots x_d^{i_d}$ est holonome si et seulement si la suite $a(i_1, \dots, i_d)$ est P-récursive. ◀

1.2.3 Classes combinatoires

Cette section définit les classes combinatoires et certaines techniques dont nous aurons besoin dans cette thèse. Elle est très largement inspirée des livres [FS09] et [Mis19].

Définition 1.53 (Classe combinatoire, [FS09]).
▶ Une classe combinatoire $(\mathcal{C}, |\cdot|)$ est un ensemble dénombrable muni d'une fonction de taille $|\cdot| : \mathcal{C} \to \mathbb{N}$, qui vérifie que pour tout $n \in \mathbb{N}$, il existe un nombre fini d'éléments de $\mathcal{C}$ de taille n. La taille d'un élément $t \in \mathcal{C}$ est notée $|t|$.

La série génératrice d'une classe combinatoire $\mathcal{C}$ est la série

$$C(x) = \sum_{t \in \mathcal{C}} x^{|t|} = \sum_{n \in \mathbb{N}} c_n x^n \,,$$

où c_n désigne le nombre fini d'éléments de $\mathcal{C}$ de taille n.

Deux classes combinatoires $\mathcal{C}_1$ et $\mathcal{C}_2$ sont dites *isomorphes* si elles ont la même série génératrice, ou de façon équivalente s'il existe une bijection de $\mathcal{C}_1$ dans $\mathcal{C}_2$ qui préserve la taille. ◀

Exemple 1.54 .
▶ L'ensemble $\{a, b\}^*$ muni de la fonction de taille qui à tout mot w associe sa longueur $|w|$, est une classe combinatoire. Comme il y a 2^n mots sur $\{a, b\}$ de longueur n, sa série génératrice est $\frac{1}{1-2x}$. ◀

Déterminer la série génératrice d'une classe combinatoire revient à savoir compter exactement pour toute taille $n \in \mathbb{N}$ le nombre c_n d'éléments de $\mathcal{C}$ de taille n. Il se trouve que de nombreuses classes combinatoires peuvent se construire à partir de classes combinatoires de base et d'opérations sur ces classes. Dans leur livre [FS09], Flajolet et Sedgewick expliquent comment la description d'une classe combinatoire sous la forme de cet assemblage de briques élémentaires se traduit automatiquement en une équation vérifiée par la série génératrice de la classe combinatoire.

Définition 1.55 (Spécification d'une classe combinatoire, [FS09]).
▶ Une classe neutre désigne une classe combinatoire constituée d'un seul élément de taille 0. Lorsque le choix du symbole ou de la classe intervient peu, nous prendrons comme convention que la classe $\mathcal{E} := \{\varepsilon\}$ désigne une classe neutre.

Une classe atomique désigne une classe combinatoire constituée d'un seul élément de taille 1. Lorsque le contenu de la classe importe peu, nous utiliserons la notation $\mathcal{Z}$ pour désigner une classe combinatoire atomique quelconque.

Si $\mathcal{C}_1$ et $\mathcal{C}_2$ sont deux classes combinatoires, munies des fonctions de tailles $|\cdot|_1$ et $|\cdot|_2$, alors :

— la classe combinatoire *produit* $\mathcal{C}_1 \times \mathcal{C}_2$ désigne l'ensemble des couples de la forme (t_1, t_2), avec $t_1 \in \mathcal{C}_1$ et $t_2 \in \mathcal{C}_2$, muni de la fonction de taille définie par $|(t_1, t_2)| = |t_1|_1 + |t_2|_2$.

— nous généralisons le produit cartésien à plus de deux classes combinatoires, en notant $\mathcal{C}_1^0 = \mathcal{E}$, $\mathcal{C}_1^1 = \mathcal{C}$ et pour $k \geqslant 2$, $\mathcal{C}_1^{k+1} := \mathcal{C}_1 \times \mathcal{C}_1^k$.

— si $\mathcal{C}_1$ et $\mathcal{C}_2$ n'ont aucun élément en commun, alors la classe combinatoire *somme* $\mathcal{C}_1 + \mathcal{C}_2$ désigne l'union des deux classes. Si les deux classes s'intersectent, nous colorions les éléments de $\mathcal{C}_1$ d'une couleur, et ceux de $\mathcal{C}_2$ d'une autre, pour les considérer comme disjoints. Plus formellement, la classe $\mathcal{C}_1 + \mathcal{C}_2$ est définie par

$$\mathcal{C}_1 + \mathcal{C}_2 := (\mathcal{C}_1 \times \{\square\}) \cup (\mathcal{C}_2 \times \{\lozenge\})$$

avec $|(t, \square)| = |t|_1$ si $t \in \mathcal{C}_1$ et $|(t, \lozenge)| = |t|_2$ si $t \in \mathcal{C}_2$ (et $\{\square\}$ et $\{\lozenge\}$ désignent deux classes neutres représentant les couleurs).

— la somme précédente se généralise pour définir la classe combinatoire associée à une somme infinie de classes combinatoires, $\sum_{n\in\mathbb{N}} \mathcal{C}_n$, pourvu que les classes sommées soient disjointes et que pour toute taille $i \in \mathbb{N}$, il existe un rang n_i à partir duquel aucun $\mathcal{C}_n$ pour $n \geqslant n_i$ ne possède d'élément de taille plus petite que i.

— si $\mathcal{C}_1$ ne contient aucun élément de taille nulle, la classe *séquence*, notée $\textsc{Seq}(\mathcal{C}_1)$ ou $\mathcal{C}_1^*$ avec l'étoile de Kleene, est la classe :

$$\textsc{Seq}(\mathcal{C}_1) := \sum_{k\in\mathbb{N}} \mathcal{C}_1^k .$$

Une *spécification* d'une classe combinatoire $\mathcal{C}$ est une classe combinatoire $\mathcal{C}_1$ isomorphe à $\mathcal{C}$ obtenue à partir de classes neutres, atomiques, et d'opérations d'unions, de produit et de séquences. Si nous autorisons aussi la classe $\mathcal{C}_1$ parmi les briques de base, la spécification est dite *récursive*. ◀

Remarque 1.56 .
▶ Si $\mathcal{C}_1$ contient un élément de taille nulle, alors $\textsc{Seq}(\mathcal{C}_1)$ n'est pas une classe combinatoire, car il existe une infinité d'éléments de taille nulle dans $\sum_{k\in\mathbb{N}} \mathcal{C}_1^k$. ◀

Remarque 1.57 .
▶ L'intérêt de rendre les unions disjointes dans la définition de la somme est double : premièrement, s'il existait un élément en commun t dans $\mathcal{C}_1$ et $\mathcal{C}_2$, mais $|t|_1 \neq |t|_2$, nous aurions un problème pour définir la taille de t dans la classe somme. Ensuite, "dédoubler" les éléments en commun permet d'exprimer facilement la série génératrice de la somme à partir de celles des classes $\mathcal{C}_1$ et $\mathcal{C}_2$. ◀

Les spécifications obtenues à partir des briques élémentaires neutres et atomiques, et les opérations somme et produit, ont le grand avantage de se traduire automatiquement en équation vérifiée par les séries génératrices des classes combinatoires associées :

Proposition 1.58 (Dictionnaire vers les séries génératrices, [FS09]).
▶ Soient $\mathcal{C}_1$ et $\mathcal{C}_2$ deux classes combinatoires, de séries génératrices $C_1(x)$ et $C_2(x)$, alors :

- la série génératrice de la classe neutre $\mathcal{E}$ est $E(x) = 1$; celle de la classe atomique $\mathcal{Z}$ est $Z(x) = x$;
- la série génératrice de la classe somme $\mathcal{C}_1 + \mathcal{C}_2$ est $C_1(x) + C_2(x)$;
- la série génératrice de la classe produit $\mathcal{C}_1 \times \mathcal{C}_2$ est $C_1(x)C_2(x)$;
- la série génératrice de la classe séquence $\textsc{Seq}(\mathcal{C}_1)$ est $\frac{1}{1-C_1(x)}$ (en supposant que $\mathcal{C}_1$ ne contient aucun élément de taille nulle).

◀

Remarque 1.59 .
▶ Le dictionnaire est en réalité beaucoup plus vaste, et permet d'autres constructions, comme des ensembles d'objets, des cycles d'objets, mais nous n'en aurons pas besoin dans cette thèse. ◀

Exemple 1.60 (Série associée à une expression régulière).
▶ Une spécification d'une classe combinatoire peut être interprétée comme un choix de décomposition de cette classe en briques élémentaires. Il faut faire attention au fait qu'on puisse mal décomposer une classe combinatoire et obtenir alors une "mauvaise" spécification, qui n'est pas isomorphe à celle de départ, car certains objets admettent plusieurs présentations.

Parfois cependant c'est justement les différentes décompositions possibles que nous voulons énumérer, donc une telle spécification n'est pas en soi "mauvaise".

Prenons la classe combinatoire $\mathcal{C}$ des mots sur $\Sigma = \{a, b\}$ qui contiennent au moins une occurrence de la lettre a, avec comme fonction de taille la longueur des mots.

Une expression régulière (ambiguë) décrivant cet ensemble est $\Sigma^* a \Sigma^*$. Nous pouvons nous inspirer de cette expression régulière pour considérer la classe combinatoire :

$$\mathcal{C}_1 = \textsc{Seq}(\mathcal{Z}_a + \mathcal{Z}_b) \times \mathcal{Z}_a \times \textsc{Seq}(\mathcal{Z}_a + \mathcal{Z}_b)\,,$$

avec $\mathcal{Z}_a = \{a\}$ et $\mathcal{Z}_a = \{b\}$ deux classes atomiques. Cette classe combinatoire n'est pas isomorphe à $\mathcal{C}$, car un même mot de $\mathcal{C}$ est en fait associé à plusieurs éléments de $\mathcal{C}_1$: ainsi les éléments $(a, (a, a))$, $(\varepsilon, (a, (a, a)))$ et $((a, a), (a, \varepsilon))$ de $\mathcal{C}_1$ représentent le même mot aaa de $\mathcal{C}$.

Il est courant, pour simplifier l'écriture, de considérer que $\mathcal{C}_1$ est un multi-ensemble de *mots* sur Σ : nous identifions par abus de notation les différents éléments $(a, (a, a))$,

$(\varepsilon, (a, (a, a)))$ et $((a, a), (a, \varepsilon))$, en disant que le mot aaa appartient à la classe $\mathcal{C}_1$, avec multiplicité 3. Avec cet abus de langage, tout mot de $\mathcal{C}_1$ a comme multiplicité son nombre d'occurrences de a.

En utilisant le dictionnaire, nous obtenons l'expression suivante pour $C_1(x)$:

$$C_1(x) = \frac{1}{1-(x+x)} \cdot x \cdot \frac{1}{1-(x+x)} = \frac{x}{(1-2x)^2}\,.$$

En décrivant les mots de $\mathcal{C}$ à l'aide de l'expression régulière non ambiguë $b^*a\Sigma^*$ (nous décomposons les mots selon leur première occurrence de la lettre a), nous obtenons la spécification suivante pour $\mathcal{C}$:

$$\mathcal{C}_2 = \textsc{Seq}(\mathcal{Z}_b) \times \mathcal{Z}_a \times \textsc{Seq}(\mathcal{Z}_a + \mathcal{Z}_b)\,.$$

Nous vérifions facilement que cette fois la classe combinatoire est bien isomorphe à $\mathcal{C}$. Le mot aaa de $\mathcal{C}$ est ainsi représenté par l'élément $(\varepsilon, (a, (a, a)))$ de $\mathcal{C}_2$, que nous noterons abusivement aussi aaa. Nous pouvons alors calculer facilement la série génératrice de $\mathcal{C}$ en utilisant le dictionnaire :

$$C(x) = \frac{1}{1-x} \cdot x \cdot \frac{1}{1-2x} = \frac{x}{(1-x)(1-2x)} \neq C_1(x)\,.$$

◀

Exemple 1.61 (Exemple de spécification récursive : les arbres binaires).
▶ Un arbre binaire est un arbre plan dont les nœuds, étiquetés par un symbole •, ont soit zéro soit deux fils. Nous définissons la taille $|t|$ d'un arbre binaire t comme son nombre de nœuds. La classe combinatoire $\mathcal{C}$ des arbres binaires se définit alors par induction :

— un nœud • est un arbre binaire, de taille 1 par définition ;
— si t_1 et t_2 sont deux arbres binaires, alors $\overset{\bullet}{\underset{t_1\ t_2}{\wedge}}$ est un arbre binaire. Sa taille est par définition $1 + |t_1| + |t_2|$.

En définissant $\mathcal{Z} = \{\bullet\}$ la classe atomique contenant un nœud • de taille 1, et en représentant un arbre de la forme $\overset{\bullet}{\underset{t_1\ t_2}{\wedge}}$ par un tuple $(\bullet, t_1, t_2)$, nous remarquons que la classe des arbres binaires admet la spécification suivante :

$$\mathcal{C} = \mathcal{Z} + \mathcal{Z} \times \mathcal{C} \times \mathcal{C}\,,$$

en vérifiant bien que la taille canonique associée à cette spécification correspond bien au nombre de nœuds d'un arbre. Par la Proposition 1.58, nous en déduisons que la série génératrice de la classe $\mathcal{C}$ vérifie :

Il y a deux solutions à cette équation du second degré, mais une seule peut s'écrire comme une série formelle à coefficients positifs.

$$C(x) = x + x \cdot C(x) \cdot C(x)$$

et ainsi, en résolvant cette équation du second degré, $C(x) = \frac{1-\sqrt{1-4x^2}}{2x}$. ◀

Remarque 1.62 .
▶ Si nous voulons compter uniquement les nœuds internes dans la taille d'un arbre binaire, il suffit de représenter les feuilles avec une classe neutre, et les nœuds internes par une classe atomique, ce qui donne la spécification suivante :

$$\mathcal{C}_{int} = \mathcal{E} + \mathcal{Z} \times \mathcal{C}_{int} \times \mathcal{C}_{int}\,,$$

dont on calcule facilement la série génératrice $C_{int}(x) = \frac{1-\sqrt{1-4x}}{2}$. ◀

Exemple 1.63 (Généralisation : variété simple d'arbres [FS09, VII. 3]).
▶ Tous les arbres qui suivent dans cet exemple ont pour taille leur nombre de nœuds.

Nous avons vu que la série génératrice des arbres binaires vérifiait l'équation $C_2(x) = x(1 + C_2(x)^2)$. Si nous regardons les arbres dont les nœuds sont d'arité 1, 2 ou 3, ils sont spécifiés par :

$$\mathcal{C}_3 = \mathcal{Z} + \mathcal{Z} \times \mathcal{C}_3 + \mathcal{Z} \times \mathcal{C}_3^2 + \mathcal{Z} \times \mathcal{C}_3^3 \,,$$

et ainsi leur série génératrice vérifie $C_3(x) = x(1 + C_3(x) + C_3(x)^2 + C_3(x)^3)$.

Plus généralement, soit $\Omega \subseteq \mathbb{N}$ un ensemble d'arités (potentiellement infini) qui contient 0. Nous considérons la classe $\mathcal{T}_\Omega$ des arbres plan enracinés dont les nœuds ont une arité dans l'ensemble Ω. Alors la série génératrice de la classe $\mathcal{T}_\Omega$ vérifie :

$$T(x) = x\phi(T(x)) \quad \text{avec } \phi(u) := \sum_{n \in \Omega} u^n \,.$$

Nous pouvons encore généraliser, en introduisant plusieurs types de nœuds pour chaque arités : par exemple s'il existe exactement deux types de nœuds unaires $\circ$ et $\bullet$ dans la spécification des arbres, alors $[u^1]\phi(u) = 2$.

Une classe d'arbres plans enracinés $\mathcal{T}$ dont la série génératrice vérifie une équation de la forme $T(x) = x\phi(T(x))$, avec $\phi(u)$ une série à coefficients positifs telle que $\phi(0) \neq 0$, est appelée une *variété simple d'arbres*. ◀

Remarque 1.64 .
▶ Le but des exemples précédents est de montrer que le calcul d'une équation vérifiée par la série génératrice d'une classe combinatoire est automatique à partir d'une bonne spécification de celle-ci : nous n'avons pas eu besoin de compter à la main le nombre de mots de chaque taille n, pour ensuite multiplier chaque élément par x^n, puis sommer, puis ensuite tenter de reconnaître des séries usuelles. ◀

Nous pouvons aussi définir des systèmes récursifs de spécification, qui s'avèrent très utiles pour mieux décrire certaines classes combinatoires :

Définition 1.65 (Spécification d'un système de classes combinatoires, [FS09, Définition I.7]).
▶ Une spécification récursive d'un tuple de classes combinatoires $(\mathcal{C}_1, \ldots, \mathcal{C}_r)$ est un système d'équations de la forme :

$$\begin{cases} \mathcal{C}_1 = \Phi_1(\mathcal{C}_1, \ldots, \mathcal{C}_r) \\ \quad \vdots \\ \mathcal{C}_r = \Phi_r(\mathcal{C}_1, \ldots, \mathcal{C}_r) \end{cases}$$

où $\Phi_1, \ldots, \Phi_r$ sont des constructions combinatoires obtenues en utilisant les opérations sommes $+$, produit cartésien $\times$ et séquence SEQ à partir des classes $\mathcal{C}_1, \ldots, \mathcal{C}_r$, de classes neutres et de classes atomiques. ◀

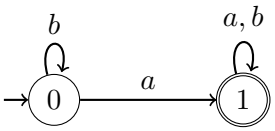
Remarque 1.66 .
▶ Toute spécification de cette forme ne définit pas forcément des classes combinatoires. Par exemple, à une dimension, la spécification $\mathcal{C} = \mathcal{Z} + \mathcal{C}$ ne définit pas une classe combinatoire valide, ni même la spécification $\mathcal{C} = \mathcal{C}$!

Lors des calculs pratiques de combinatoire analytique, la question ne se pose pas car nous partons généralement d'une classe combinatoire qui existe, et nous construisons rétrospectivement une spécification de cette classe, qui par construction engendre exactement la classe combinatoire que nous étudions.

D'un point de vue théorique, il est cependant nécessaire pour construire une théorie des classes combinatoires et de leur spécification de définir proprement celles qui définissent des classes combinatoires valides. Il ne s'agit d'ailleurs pas que d'une question théorique, puisque les spécifications servent aussi à rendre automatique la génération aléatoire : un algorithme automatique de génération aléatoire doit être capable de détecter que la spécification qu'on lui donne en entrée décrit une classe combinatoire invalide.

Ce sont des questions profondes, et qui sortent du cadre de cette thèse. Le but de ces préliminaires n'est pas de définir les propriétés que doit vérifier une spécification pour définir une classe combinatoire. Nous renvoyons pour cela le lecteur à l'article de [PSS12] qui définit précisément et rigoureusement des conditions sur de tels systèmes qui permettent de garantir l'existence et l'unicité de la solution, en utilisant le formalisme des espèces [BBLL98]. En pratique nous utiliserons dans les prochains chapitres des spécifications suffisamment simples pour que la question ne se pose pas, qui vérifieront presque toutes les conditions de [PSS12]. Les seules qui pourraient poser des problèmes seront celles issues des automates, car les spécifications produisent naturellement des classes neutres isolées, ce qui est interdit dans les conditions de [PSS12]. Cependant ces systèmes ont naturellement une solution, puisqu'ils décrivent exactement les calculs des automates, et ils sont d'autre part très simples (ils sont linéaires). ◀

Exemple 1.67 (Série des calculs d'un automate fini).
▶ Considérons l'automate suivant :



Notons $\mathcal{C}_0$ (resp. $\mathcal{C}_1$) la classe combinatoire des calculs de l'automate joignant l'état q_0 (resp. q_1) à l'état final q_1, avec comme fonction de taille le nombre de transitions empruntées.

Un calcul de $\mathcal{C}_1$ peut se décomposer de la façon suivante : ou bien c'est un calcul vide, ou bien c'est une transition de la forme (q_1, a, q_1) ou (q_1, b, q_1) suivie d'un calcul de $\mathcal{C}_1$. Nous pouvons donc spécifier $\mathcal{C}_1$:

$$\mathcal{C}_1 = \mathcal{E} + (\mathcal{Z}_{1,a,1} + \mathcal{Z}_{1,b,1}) \times \mathcal{C}_1$$

avec $\mathcal{Z}_{1,a,1} := \{(q_1, a, q_1)\}$ qui est une classe atomique (nous définissons de même $\mathcal{Z}_{1,b,1}, \mathcal{Z}_{0,a,1}, \mathcal{Z}_{0,b,0}$), et $\mathcal{E}$ une classe neutre représentant un calcul vide.

Un calcul de $\mathcal{C}_0$ commence soit par la transition (q_0, b, q_0) puis est suivi d'un calcul de $\mathcal{C}_0$, soit emprunte la transition (q_0, a, q_1) et est alors suivi d'un calcul de $\mathcal{C}_1$. Nous avons donc finalement le système suivant pour les calculs de l'automate :

$$\begin{cases} \mathcal{C}_0 = \mathcal{Z}_{0,b,0} \times \mathcal{C}_0 + \mathcal{Z}_{0,a,1} \times \mathcal{C}_1 \\ \mathcal{C}_1 = \mathcal{E} + (\mathcal{Z}_{1,a,1} + \mathcal{Z}_{1,b,1}) \times \mathcal{C}_1 \end{cases}$$

D'un point de vue purement combinatoire, nous n'avons pas besoin de distinguer les différentes transitions, qui comptent toutes pour un dans la taille d'un calcul, si bien que nous utilisons plus simplement la spécification suivante :

$$\begin{cases} \mathcal{C}_0 = \mathcal{Z} \times \mathcal{C}_0 + \mathcal{Z} \times \mathcal{C}_1 \\ \mathcal{C}_1 = \mathcal{E} + (\mathcal{Z} + \mathcal{Z}) \times \mathcal{C}_1 \end{cases}$$

qui génère bien les calculs de l'automate, en "oubliant" les étiquettes des transitions. Leur série génératrice vérifie donc le système :

$$\begin{cases} C_0(x) = x \times C_0(x) + x \times C_1(x) \\ C_1(x) = 1 + (x + x) \times C_1(x) \end{cases}$$

Dans ce cas particulier, le système est triangulaire, il se résout donc très facilement : $C_1(x) = \frac{1}{1-2x}$ et $C_0(x) = \frac{x}{(1-x)(1-2x)}$. Comme l'automate est non ambigu, il y a autant de calculs acceptants de longueur n que de mots de longueur n acceptés par l'automate. Il n'est donc pas étonnant que $C_0(x)$ soit égal à la série génératrice du langage accepté par l'automate, à savoir le langage des mots qui ont au moins une occurrence de la lettre a. ◀

1.2.4 Paramètre combinatoire, moments

Définition 1.68 (paramètre combinatoire).
▶ Soit $\mathcal{C}$ une classe combinatoire. Un paramètre est une fonction $\chi : \mathcal{C} \to \mathbb{N}$ qui associe à tout élément t de la classe $\mathcal{C}$ une valeur entière $\chi(t)$. La série génératrice bivariée de $\mathcal{C}$ associée au paramètre χ est la série :

$$C(x,u) := \sum_{t \in \mathcal{C}} x^{|t|} u^{\chi(t)} = \sum_{n,k \in \mathbb{N}} c_{n,k} x^n u^k$$

où $c_{n,k}$ désigne le nombre d'éléments t de $\mathcal{C}$ de taille n qui vérifient $\chi(t) = k$.

En posant $u = 1$, nous remarquons que $C(x,1) = C(x)$ est la série génératrice de la classe combinatoire $\mathcal{C}$.

La série $QC(x) := (\partial_u C(x,u))\big|_{u=1} = \sum_{n \in \mathbb{N}} (\sum_{k \in \mathbb{N}} k c_{n,k}) x^n$ est appelée la série génératrice cumulée de $(\mathcal{C}, \chi)$. ◀

Soit $\mathcal{C}$ une classe combinatoire avec χ un paramètre combinatoire. Pour tout $n \in \mathbb{N}$, notons $\mathcal{C}_n$ l'ensemble des éléments de $\mathcal{C}$ de taille n, c_n son cardinal, et $c_{n,k}$ le nombre d'éléments t de $\mathcal{C}_n$ vérifiant $\chi(t) = k$.

La distribution uniforme sur $\mathcal{C}_n$ associe à tout élément t de $\mathcal{C}_n$ une probabilité $\mathbb{P}_n(t) = 1/c_n$. Nous pouvons considérer χ comme une variable aléatoire sur l'espace probabilisé discret $\mathcal{C}_n$ muni de la probabilité $\mathbb{P}_n$. Alors $\mathbb{P}_n(\chi = k) = \frac{c_{n,k}}{c_n}$, et l'espérance de χ sur $\mathcal{C}_n$ est définie par

$$\mathbb{E}_n(\chi) := \sum_{k \in \mathbb{N}} k \cdot \mathbb{P}_n(\chi = k) = \sum_{k \in \mathbb{N}} \frac{k c_{n,k}}{c_n} .$$

Les moments factoriels d'ordre r de χ sur $\mathcal{C}_n$ sont définis par $\mathbb{E}_n(\chi(\chi-1)\dots(\chi-r+1))$. La proposition suivante se vérifie simplement :

Proposition 1.69 (Moments, [FS09, Proposition III.2]).
▶ En gardant les notations de cette section, si χ est un paramètre associé à une classe combinatoire $\mathcal{C}$, alors :

$$\mathbb{E}_n(\chi) = \frac{[x^n]\partial_u C(x,u)\big|_{u=1}}{[x^n]C(x,1)} \tag{1.1}$$

et plus généralement :

$$\mathbb{E}_n(\chi(\chi-1)\dots(\chi-r+1)) = \frac{[x^n]\partial_u^r C(x,u)\big|_{u=1}}{[x^n]C(x,1)} \tag{1.2}$$

où $\partial_u^r C(x,u)$ désigne la dérivée partielle de $C(x,u)$ par rapport à u effectuée r fois. Enfin, les moments d'ordre supérieurs s'expriment aussi à l'aide de $C(x,u)$:

$$\mathbb{E}_n(\chi^r) = \frac{[x^n](u\partial_u)^r C(x,u)\big|_{u=1}}{[x^n]C(x,1)} \tag{1.3}$$

où $(u\partial_u)^r C(x,u)$ signifie qu'on a effectué r fois l'opération de dériver par rapport à u puis multiplier le résultat par u. ◀

1.2.5 Paramètre hérité

Il sera utile dans la thèse de considérer des paramètres compatibles avec les spécifications des classes combinatoires. Ces paramètres sont appelés *hérités*. Nous les définissons plus généralement pour des tuples de d paramètres $\chi = (\chi_1, \dots, \chi_d)$.

Définition 1.70 (Paramètre hérité, [FS09, Definition III.5]).
▶ Soient $(\mathcal{C}_1, \chi)$, $(\mathcal{C}_2, \xi)$, $(\mathcal{C}_3, \zeta)$ trois classes combinatoires associées à des paramètres multidimensionnels de dimension d.

- si $\mathcal{C}_1 = \mathcal{C}_2 + \mathcal{C}_3$, alors χ est dit hérité de ξ et ζ si pour tout $t \in \mathcal{C}_1$, $\chi(t) = \xi(t)$ si $t \in \mathcal{C}_2$, et $\chi(t) = \zeta(t)$ si $t \in \mathcal{C}_3$. Nous notons alors cet héritage en écrivant $(\mathcal{C}_1, \chi) = (\mathcal{C}_2, \xi) + (\mathcal{C}_3, \zeta)$;
- si $\mathcal{C}_1 = \mathcal{C}_2 \times \mathcal{C}_3$, alors χ est dit hérité de ξ et ζ si pour tout $t_1 = (t_2, t_3) \in \mathcal{C}_1$, $\chi(t_1) = \xi(t_2) + \zeta(t_3)$. Nous notons alors cet héritage en écrivant $(\mathcal{C}_1, \chi) = (\mathcal{C}_2, \xi) \times (\mathcal{C}_3, \zeta)$;
- si $\mathcal{C}_1 = \textsc{Seq}(\mathcal{C}_2)$, alors χ est dit hérité de ξ si pour tout $t \in \mathcal{C}_1$ de la forme $(t_1, \dots, t_k) \in \mathcal{C}_2^k$ pour un certain $k \in \mathbb{N}$, $\chi(t) = \xi(t_1) + \dots + \xi(t_k)$. Nous notons alors cet héritage en écrivant $(\mathcal{C}_1, \chi) = \textsc{Seq}(\mathcal{C}_2, \xi)$.

◀

La sérié multivariée associée à une classe combinatoire $(\mathcal{C}, \chi)$ est la série

$$C(x, y_1, \dots, y_d) = \sum_{n, i_1, \dots, i_d \in \mathbb{N}} c_{n,i_1,\dots,i_d} x^n y_1^{i_1} \dots y_d^{i_d}$$

où $c_{n,i_1,\dots,i_d}$ désigne le nombre d'éléments t de $\mathcal{C}$ de taille n vérifiant $\chi_1(t) = i_1, \dots, \chi_d(t) = i_d$.

Nous noterons cette série $C(x, \boldsymbol{y})$ pour condenser la notation.

Proposition 1.71 (Paramètre hérité, [FS09, Theorem III.1]).
▶ En gardant les mêmes notations que précédemment :

– si $(\mathcal{C}_1, \chi) = (\mathcal{C}_2, \xi) + (\mathcal{C}_3, \zeta)$, alors $C_1(x, \boldsymbol{y}) = C_2(x, \boldsymbol{y}) + C_3(x, \boldsymbol{y})$;
– si $(\mathcal{C}_1, \chi) = (\mathcal{C}_2, \xi) \times (\mathcal{C}_3, \zeta)$, alors $C_1(x, \boldsymbol{y}) = C_2(x, \boldsymbol{y}) \cdot C_3(x, \boldsymbol{y})$;
– si $(\mathcal{C}_1, \chi) = \textsc{Seq}(\mathcal{C}_2, \xi)$, alors $C_1(x, \boldsymbol{y}) = \frac{1}{1-C_2(x,\boldsymbol{y})}$.

◀

Exemple 1.72 (Marquage de sous-structures).
▶ Un cas particulier de paramètre hérité très utile intervient en marquant dans une spécification un atome ou une structure par une classe neutre $\mathcal{U}$ (de taille nulle). Le paramètre considéré est le nombre de marquages dans un élément de la classe combinatoire ; on vérifie facilement que ce paramètre est hérité. Cette technique est très utile pour compter le nombre d'occurrences d'une sous-structure identifiable dans la spécification de $\mathcal{C}$.

Par exemple, rappelons que les arbres unaire-binaires sont spécifiés récursivement par :

$$\mathcal{C} = \mathcal{Z} + \mathcal{Z} \times \mathcal{C} + \mathcal{Z} \times \mathcal{C} \times \mathcal{C}\,.$$

Si nous voulons compter avec χ le nombre de feuilles des arbres unaire-binaires, nous ajoutons une marque $\mathcal{U}$ au niveau des feuilles :

$$\mathcal{C} = \mathcal{Z} \times \mathcal{U} + \mathcal{Z} \times \mathcal{C} + \mathcal{Z} \times \mathcal{C} \times \mathcal{C}\,,$$

qui se traduit au niveau des séries génératrices par : $C_f(x, u) = zu + zC_f(x, u) + zC_f(x, u)^2$.

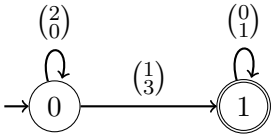
Si nous voulions compter les nœuds unaires à la place des feuilles, il suffisait de marquer les nœuds unaires :

$$\mathcal{C} = \mathcal{Z} + (\mathcal{Z} \times \mathcal{U}) \times \mathcal{C} + \mathcal{Z} \times \mathcal{C} \times \mathcal{C}\,,$$

qui donne l'équation $C_{un}(x, u) = z + zuC_{un}(x, u) + zC_{un}(x, u)^2$.

Si nous voulons compter les deux d'un coup, nous obtiendrions de même $C_{f,un}(x, u) = zu + zuC_{f,un}(x, u) + zC_{f,un}(x, u)^2$. ◀

Exemple 1.73 (Série des calculs d'un automate de vecteurs).
▶ Considérons l'automate fini suivant sur $(\mathbb{N}^2, +)$ reconnaissant un ensemble semi-linéaire :



Notons $\mathcal{C}_0$ (resp. $\mathcal{C}_1$) la classe combinatoire des calculs de l'automate joignant l'état q_0 (resp. q_1) à l'état final q_1, avec comme fonction de taille le nombre de transitions empruntées. Nous avons déjà vu qu'en considérant uniquement la taille, les calculs pouvaient être spécifiés par la classe combinatoire suivante :

$$\begin{cases} \mathcal{C}_0 = \mathcal{Z} \times \mathcal{C}_0 + \mathcal{Z} \times \mathcal{C}_1 \\ \mathcal{C}_1 = \mathcal{E} + (\mathcal{Z} + \mathcal{Z}) \times \mathcal{C}_1 \end{cases}$$

Nous notons χ_1 (resp. χ_2) le paramètre qui à tout calcul de l'automate associe la première coordonnée (resp. seconde) de la somme des vecteurs présents dans ses

transitions. Ce paramètre est naturellement hérité, et nous remarquons qu'un marquage avec une classe neutre $\mathcal{Y}_1$ (resp. $\mathcal{Y}_2$) permet de l'insérer dans la spécification : nous marquons deux fois avec $\mathcal{Y}_1$ la transition de q_0 à q_0, une fois avec $\mathcal{Y}_1$ et trois fois avec $\mathcal{Y}_2$ celle de q_0 à q_1, et une fois avec $\mathcal{Y}_2$ celle de q_1 à q_1 :

$$\begin{cases} \mathcal{C}_0 = \mathcal{Y}_1^2 \times \mathcal{Z} \times \mathcal{C}_0 + \mathcal{Y}_1 \times \mathcal{Y}_2^3 \times \mathcal{Z} \times \mathcal{C}_1 \\ \mathcal{C}_1 = \mathcal{E} + \mathcal{Y}_2 \times (\mathcal{Z} + \mathcal{Z}) \times \mathcal{C}_1 \end{cases}$$

de telle sorte que le nombre de marques $\mathcal{Y}_1$ (resp. $\mathcal{Y}_2$) dans un élément t construit à partir de la spécification correspond bien à $\chi_1(t)$ (resp. $\chi_2(t)$).

En appliquant la Proposition 1.71, nous obtenons le système suivant pour les séries génératrices des calculs de l'automate :

$$\begin{cases} C_0(x, y_1, y_2) = xy_1^2 C_0(x, y_1, y_2) + xy_1y_2^3 C_1(x, y_1, y_2) \\ C_1(x, y_1, y_2) = 1 + 2xy_2 C_1(x, y_1, y_2) \end{cases}$$

et ainsi $C_0(x, y_1, y_2) = \dfrac{xy_1y_2^3}{(1 - xy_1^2)(1 - 2xy_2)}$. ◀

Remarque 1.74.
▶ La taille peut être vue comme un paramètre hérité quelconque. Si nous pouvons garantir que pour tout $i_1, \ldots, i_d$, le nombre d'éléments de $\mathcal{C}$ vérifiant $\chi_1(t) = i_1$, …, $\chi_d(t) = i_d$ est fini, il est possible d'omettre la taille et la variable x si cette information ne nous intéresse pas ; cela revient à poser $x = 1$ dans $C(x, \boldsymbol{y})$. ◀

Exemple 1.75 (Série d'un ensemble linéaire décrit de façon non ambiguë).
▶ Regardons l'ensemble linéaire $L := \binom{1}{3} + \{\binom{1}{1}, \binom{0}{2}\}^* = \{(1 + n, 3 + n + 2m) \mid n, m \in \mathbb{N}\} \subseteq \mathbb{N}^2$.

Nous voulons calculer la série du support de L, définie par

$$L(y_1, y_2) := \sum_{(i_1,i_2)\in L} y_1^{i_1} y_2^{i_2} .$$

Normalement, dans le contexte des classes combinatoires, il faut définir une fonction de taille à valeur dans $\mathbb{N}$. Nous considérons donc la classe $\mathcal{C}$ des vecteurs de L, en prenant pour taille d'un vecteur la somme de ses coordonnées (sa norme 1). Nous marquons avec $\mathcal{Y}_1$ et $\mathcal{Y}_2$ chaque coordonnée. Comme tout vecteur de $\binom{1}{3} + \{\binom{1}{1}, \binom{0}{2}\}^*$ se décompose d'une unique façon sous la forme $\binom{1}{3} + n\binom{1}{1} + m\binom{0}{2}$ avec $n, m \in \mathbb{N}$, nous pouvons écrire la spécification suivante pour les vecteurs de $\mathcal{C}$:

$$\mathcal{C} = \mathcal{Z}^4 \times \mathcal{Y}_1 \times \mathcal{Y}_2^3 \times \textsc{Seq}(\mathcal{Z}^2 \times \mathcal{Y}_1 \times \mathcal{Y}_2) \times \textsc{Seq}(\mathcal{Z}^2 \times \mathcal{Y}_2^2)$$

ce qui donne la série génératrice multivariée :

$$C(x, y_1, y_2) := \frac{x^4 y_1 y_2^3}{(1 - x^2 y_1 y_2)(1 - x^2 y_2^2)} .$$

En posant $x = 1$, nous obtenons $L(y_1, y_2) = C(1, y_1, y_2) = \frac{y_1 y_2^3}{(1-y_1y_2)(1-y_2^2)}$.

En considérant par la remarque précédent que la taille est un marquage comme un autre, nous pouvons nous passer de la variable x, et de la classe $\mathcal{Z}$ dans la spécification. En effet, la spécification est non ambiguë, de telle sorte que pout toute valeur (i_1, i_2)

du nombre d'occurrences des marqueurs $\mathcal{Y}_1$ et $\mathcal{Y}_2$, il y a au plus une seule structure engendrée par la spécification ayant ce nombre de marqueurs. Nous utilisons donc directement la spécification :

Cela revient à généraliser la notion de taille d'un objet comme une fonction de $\mathcal{C}$ dans $\mathbb{N}^2$ plutôt que dans $\mathbb{N}$.

$$\mathcal{C} = \mathcal{Y}_1 \times \mathcal{Y}_2^3 \times \textsc{Seq}(\mathcal{Y}_1 \times \mathcal{Y}_2) \times \textsc{Seq}(\mathcal{Y}_2^2)$$

qui donne directement $L(y_1, y_2) = \frac{y_1 y_2^3}{(1-y_1y_2)(1-y_2^2)}$. ◀

1.2.6 Singularités et Théorème de Transfert

Je reprends dans cette section la présentation de [FS09] et [Mis19]. L'idée de la combinatoire analytique est de regarder les séries formelles introduites précédemment comme des fonctions analytiques complexes. Nous nous en servirons notamment pour relier le comportement de ces fonctions analytiques au voisinage de leurs singularités au comportement asymptotique de leurs coefficients de Taylor.

Définition 1.76 (Rappels d'analyse complexe).
▶ Soit $f(z) : \mathbb{C} \to \mathbb{C}$ une fonction, et $z_0 \in \mathbb{C}$. Alors

- la fonction f est dite dérivable en z_0 si le quotient $\frac{f(z_0+h)-f(z_0)}{h}$ admet une limite lorsque $h \in \mathbb{C}$ tend vers 0.
- f est dite holomorphe en z_0 si elle est dérivable en tout point d'un voisinage de z_0
- f est dite analytique en z_0 s'il existe une série entière de la forme $\sum_{n\in\mathbb{N}} a_n z^n$, de rayon de convergence strictement positif R, telle qu'à l'intérieur du disque ouvert $D(z_0, R)$ de centre z_0 et de rayon R, on ait $f(z) = \sum_{n\in\mathbb{N}} a_n(z - z_0)^n$.

Ces deux dernières notions coïncident : une fonction est holomorphe en un point z_0 si et seulement si elle est analytique en z_0. ◀

Définition 1.77 (Prolongement analytique, singularité, [FS09, Definition IV.4]).
▶ Soit f une fonction analytique définie sur un ouvert Ω non vide défini par l'intérieur d'une courbe $\gamma : [0, 1] \to \mathbb{C}$ simple (γ est injective sur $[0, 1[$) et fermée ($\gamma(0) = \gamma(1)$). Soit z_0 un point de $\mathbb{C}$ sur la courbe γ.

On dit que f admet un prolongement analytique en z_0 s'il existe une fonction analytique f^* définie sur un ouvert Ω^* contenant z_0, telle que $f^*(z) = f(z)$ pour tout $z \in \Omega \cap \Omega^*$.

Il y a unicité du prolongement analytique : si deux fonctions analytiques définies sur un ouvert connexe par arc Ω de $\mathbb{C}$ coïncident sur un voisinage d'un point de U, alors ces fonctions sont égales sur Ω tout entier.

La vraie formulation du prolongement analytique utilise des ouverts connexes, mais nous aurons uniquement besoin de la version sur des connexes par arc.

Un point z_0 est appelé singularité de f si f n'admet pas de prolongement analytique en z_0. ◀

Exemple 1.78 (Exemples de singularités, [FS09, Definition IV.4]).
▶ La fonction $\frac{1}{1-z}$ admet une unique singularité en $z = 1$.

La fonction $\sqrt{1-z} := \sum_{n\in\mathbb{N}} \frac{1}{2^{2n}(1-2n)}\binom{2n}{n} z^n$ définie sur le disque ouvert de centre 0 et de rayon 1, est prolongeable analytiquement sur $\mathbb{C}$ privé de la droite $[1, +\infty[$, mais n'admet aucun prolongement analytique en 1, qui est une singularité de $\sqrt{1-z}$. ◀

Proposition 1.79 (Singularités sur le cercle de convergence, [FS09, Theorem IV.5 et IV.6]).

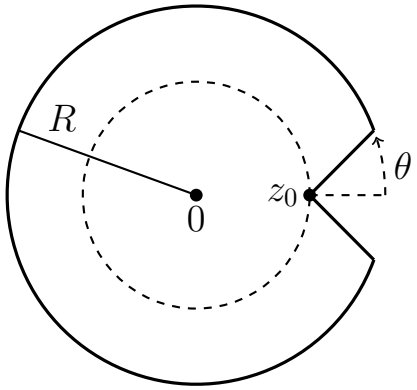


Figure 1.2. Représentation d'un Δ-domaine.

▶ Soit $f(z)$ une fonction analytique au voisinage de 0, qui admet le développement en série entière $f(z) = \sum_n a_n z^n$. Soit R le rayon de convergence de cette série entière. Alors :

— f admet une singularité sur le cercle de convergence $|z| = R$;

— de plus, si tous les a_n sont des réels positifs, alors $z = R$ est une singularité de f. C'est le *théorème de Pringsheim.*

Les singularités qui sont présentes sur le cercle de convergence sont appelées *singularités dominantes* de f. ◀

L'étude des singularités dominantes d'une fonction est au coeur de la combinatoire analytique, car sous certaines conditions, le comportement d'une série analytique au voisinage de ses singularités dominantes se transfère sur le comportement asymptotique de ses coefficients.

Définition 1.80 (Δ-domaine).
▶ Soit $z_0 \in \mathbb{C}$ un point du plan complexe non nul. Un Δ-domaine de z_0 est un ouvert de la forme :

$$\Delta_{z_0}(\theta, R) = \{z \,|\, |z| < R,\ z \neq z_0,\ |\arg(z - z_0)| > \theta\}$$

pour un certain angle θ, et un rayon $R > |z_0|$ (voir Figure 1.2). ◀

Théorème 1.81 (Théorème de Transfert, [FO90], [FS09, Theorem VI.3]).
Nous le citons dans le cas d'une unique singularité dominante réelle positive, mais il existe des versions qui gèrent le cas de plusieurs singularités dominantes.

▶ Soit $\alpha, \beta \in \mathbb{R}$ avec $\alpha \notin \{0, -1, -2, \ldots\}$, et f une fonction analytique en $z = 0$, ayant comme unique singularité dominante $z = \rho > 0$ réelle positive. Nous supposons de plus que f est analytique sur un Δ-domaine de $z_0 = \rho$. Alors :

— Si $f(z)$ satisfait lorsque z tend vers ρ, en restant dans le Δ-domaine,

$$f(z) = O\left((1 - z/\rho)^{-\alpha}(-\log(1 - z/\rho))^{\beta}\right),$$

alors $[z^n]f(z) =_{n\to\infty} O\left(\rho^{-n} n^{\alpha-1}(\log(n))^{\beta}\right)$.

— Si $f(z)$ satisfait lorsque z tend vers ρ, en restant dans le Δ-domaine, la condition

$$f(z) = o\left((1-z/\rho)^{-\alpha}(-\log(1-z/\rho))^{\beta}\right),$$

alors $[z^n]f(z) =_{n\to\infty} o\left(\rho^{-n}n^{\alpha-1}(\log(n))^{\beta}\right)$.

— Si $f(z)$ satisfait lorsque z tend vers ρ en restant dans le Δ-domaine, la condition

$$f(z) \sim \lambda(1-z/\rho)^{-\alpha}(-\log(1-z/\rho))^{\beta},$$

avec λ une constante, alors

$$[z^n]f(z) \sim_{n\to\infty} \frac{\lambda}{\Gamma(\alpha)}\rho^{-n}n^{\alpha-1}(\log(n))^{\beta}.$$

$\Gamma(z) := \int_0^{+\infty} e^{-t}t^{z-1}dt$ *est la fonction d'Euler, la généralisation de la factorielle : en particulier pour* $n \in \mathbb{N}^*, \Gamma(n) = (n-1)!$, $\Gamma(1/2) = \sqrt{\pi}$, $\Gamma(-1/2) = -2\sqrt{\pi}$.

Autrement dit, le comportement de f autour de sa singularité dominante se transfère dans le comportement asymptotique de ses coefficients. ◀

Remarque 1.82.
▶ Une des forces de la combinatoire analytique est qu'elle fournit de nombreux outils pour décrire le comportement asymptotique de séries décrites par des équations, sans avoir besoin de résoudre ces équations. Couplée avec le théorème de Transfert, elle permet ainsi d'obtenir des équivalents asymptotiques de suites sans avoir besoin d'une forme close. ◀

L'exemple suivant vise à résumer tous les outils de combinatoire analytique que nous avons présenté dans ces préliminaires.

Exemple 1.83 (Exemple récapitulatif).
▶ Nous définissons l'ensemble $\mathcal{L}_{\mathcal{R}}$ des arbres représentant des expressions régulières sur l'alphabet $\{a, b\}$ de façon inductive :

— ε, a, b sont des arbres d'expressions régulières ;

— si $T \in \mathcal{L}_{\mathcal{R}}$, alors $\overset{\star}{|}_{T} \in \mathcal{L}_{\mathcal{R}}$;

— si T_1 et T_2 sont dans $\mathcal{L}_{\mathcal{R}}$, alors $\overset{+}{\wedge}_{T_1\ T_2}$ et $\overset{\bullet}{\wedge}_{T_1\ T_2}$ sont dans $\mathcal{L}_{\mathcal{R}}$.

Nous prendrons l'habitude de donner à des définitions d'arbres de cette forme une description récursive de la forme :

$$\mathcal{L}_{\mathcal{R}} = \varepsilon + a + b + \overset{\star}{|}_{\mathcal{L}_{\mathcal{R}}} + \overset{+}{\wedge}_{\mathcal{L}_{\mathcal{R}}\ \mathcal{L}_{\mathcal{R}}} + \overset{\bullet}{\wedge}_{\mathcal{L}_{\mathcal{R}}\ \mathcal{L}_{\mathcal{R}}},$$

qui est en quelque sorte un mélange entre une définition récursive, et une spécification combinatoire. Cela permet de définir succinctement les arbres que l'on étudie, et est plus lisible qu'une spécification combinatoire de $\mathcal{L}_{\mathcal{R}}$, qui utilise des produits cartésiens :

$$\mathcal{L}_{\mathcal{R}} = \mathcal{Z} + \mathcal{Z} + \mathcal{Z} + \mathcal{Z} \times \mathcal{L}_{\mathcal{R}} + \mathcal{Z} \times \mathcal{L}_{\mathcal{R}} \times \mathcal{L}_{\mathcal{R}} + \mathcal{Z} \times \mathcal{L}_{\mathcal{R}} \times \mathcal{L}_{\mathcal{R}},$$

en marquant chaque nœud des arbres par une classe atomique $\mathcal{Z}$. Ainsi, en traduisant cette spécification en équation sur les séries génératrices :

$$L_R(z) = 3z + zL_R(z) + 2zL_R(z)^2$$

nous pouvons résoudre cette équation :

$$L_R(z) = \frac{1 - z - \sqrt{\Delta(z)}}{4z}, \text{ avec } \Delta(z) = (1 - z/\rho_1)(1 - z/\rho_2),$$

avec $\rho_1 = \frac{1}{1+2\sqrt{6}}$ et $\rho_2 = \frac{1}{1-2\sqrt{6}}$. $L_R(z)$ est analytique en $z = 0$, donc il y a une unique singularité dominante : $\rho_1 > 0$. En utilisant le théorème de Transfert, nous déduisons :

$$[z^n]L_R(z) \sim \frac{\sqrt{1 - \rho_1/\rho_2}}{8\rho_1\sqrt{\pi}\, n^{3/2}}(1 + 2\sqrt{6})^n.$$

Nous nous intéressons désormais à la transformation suivante : étant donné un arbre d'expression régulière $T \in \mathcal{L}_\mathcal{R}$, nous notons $\sigma(T)$ l'arbre obtenu à partir de T en supprimant toutes les étoiles $\star$ qui se trouvent directement au-dessus d'une étoile ; autrement dit $\sigma(T)$ est obtenu en remplaçant tous les sous-arbres de T de la forme $(T_1^\star)^\star$ par $T_1^\star$ et en recommençant jusqu'à ce qu'il ne soit plus possible de trouver deux étoiles consécutives.

Nous nous intéressons à la taille moyenne d'un arbre de taille n après simplification. Nous cherchons donc à marquer dans la spécification les étoiles qui disparaissent après simplification. Actuellement, la spécification de $\mathcal{L}_\mathcal{R}$ ne nous permet pas de distinguer les étoiles qui vont rester de celles qui vont disparaître, nous devons introduire des classes supplémentaires pour le faire : nous notons $\mathcal{A}$ l'ensemble des arbres dont la racine n'est pas une étoile, et $\mathcal{S}$ l'ensemble des arbres qui commencent par une étoile. Nous avons la définition récursive suivante :

$$\begin{cases} \mathcal{L}_\mathcal{R} = \mathcal{A} + \mathcal{S} \\ \mathcal{S} = \overset{\star}{|}_{\mathcal{A}} + \overset{\star}{|}_{\mathcal{S}} \\ \mathcal{A} = a + b + \varepsilon + \overset{\bullet}{\wedge}_{\mathcal{L}_\mathcal{R}\ \mathcal{L}_\mathcal{R}} + \overset{+}{\wedge}_{\mathcal{L}_\mathcal{R}\ \mathcal{L}_\mathcal{R}} \end{cases} \tag{1.4}$$

Ce système se traduit directement en système combinatoire, en marquant chaque nœud par $\mathcal{Z}$, et en prenant soin de marquer par une classe neutre $\mathcal{U}$ l'étoile de $\overset{\star}{|}_{\mathcal{S}}$, qui disparaît après simplification, car elle se trouve forcément au-dessus une étoile :

$$\begin{cases} \mathcal{L}_\mathcal{R} = \mathcal{A} + \mathcal{S} \\ \mathcal{S} = \mathcal{Z} \times \mathcal{A} + \mathcal{Z} \times \mathcal{U} \times \mathcal{S} \\ \mathcal{A} = \mathcal{Z} + \mathcal{Z} + \mathcal{Z} + \mathcal{Z} \times \mathcal{L}_\mathcal{R} \times \mathcal{L}_\mathcal{R} + \mathcal{Z} \times \mathcal{L}_\mathcal{R} \times \mathcal{L}_\mathcal{R} \end{cases} \tag{1.5}$$

et donc en utilisant le dictionnaire nous obtenons le système suivant satisfait par les séries génératrices :

$$\begin{cases} L_R(z,u) = A(z,u) + S(z,u) \\ S(z,u) = zA(z,u) + zuS(z,u) \\ A(z,u) = 3z + 2zL_R(z,u)^2. \end{cases}$$

Remarque 1.84 .

▶ Dans la suite des chapitres, nous serons moins rigoureux, nous dirons abusivement que le système récursif (1.4) est une spécification, et nous passerons directement à l'équation sur les séries génératrices en expliquant à l'écrit quelle variable marque quel type de nœud. ◀

Nous rappelons qu'en notant $QL_R(z) = \partial_u L_R(z,u)\big|_{u=1}$, le nombre moyen e_n d'étoiles supprimées lors de la simplification d'un arbre de taille n est donné par la formule de la Proposition 1.69 :

$$e_n = \frac{[z^n]QL_R(z)}{[z^n]L_R(z)}\,.$$

En dérivant le système d'équation par rapport à u et en posant $u = 1$, nous obtenons le système suivant :

$$\begin{cases} QL_R(z) = QA(z) + QS(z) \\ QS(z) = zQA(z) + zQS(z) + zS(z) \\ QA(z) = 4zL_R(z)QL_R(z). \end{cases}$$

Ce système se résout facilement, en utilisant le fait que $S(z) = zL_R(z)$. Nous obtenons alors :

$$QL_R(z) = \frac{zS(z)}{1-z-4zL_R(z)} = \frac{z^2L_R(z)}{\sqrt{\Delta(z)}} = \frac{z-z^2-z\sqrt{\Delta(z)}}{4\sqrt{\Delta(z)}}.$$

Ainsi $QL_R(z)$ possède une unique singularité dominante en $z = \rho_1$. Comme au voisinage de $z = \rho_1$, $QL_R(z) \sim \frac{\rho_1-\rho_1^2}{4\sqrt{1-\rho_1/\rho_2}\sqrt{1-z/\rho_1}}$, nous pouvons appliquer le théorème de Transfert (car $QL_R(z)$ est bien anaytique sur un Δ-domaine de ρ_1) :

$$[z^n]QL_R(z) \sim \frac{\rho_1-\rho_1^2}{4\sqrt{\pi n(1-\rho_1/\rho_2)}}(1+2\sqrt{6})^n.$$

Ainsi le nombre moyen d'étoiles supprimées dans un arbre de taille n est asymptotiquement équivalent à

$$e_n = \frac{[z^n]QL_R(z)}{[z^n]L_R(z)} \sim \frac{\rho_1-\rho_1^2}{4\sqrt{1-\rho_1/\rho_2}}\frac{8\rho_1}{\sqrt{1-\rho_1/\rho_2}}\,n = \frac{25-4\sqrt{6}}{529}\,n \approx 0.0287\,n.$$

En conclusion, la taille moyenne d'un arbre après suppression des étoiles consécutives est asymptotiquement équivalent à κn avec $\kappa = 1 - \frac{25-4\sqrt{6}}{529} = \frac{4}{529}(126+\sqrt{6})$. ◀

1.3 Préliminaires sur les systèmes de séries et le théorème de Drmota

1.3.1 Systèmes polynomiaux

Nous expliquerons dans la suite la différence que nous faisons entre le théorème de Drmota et le théorème de Drmota-Lalley-Woods.

Un système polynomial d'équations de dimension d est un système d'équations, dont les inconnues sont des fonctions $(y_1(z),\dots,y_d(z))$, de la forme :

$$\begin{cases} y_1(z) &= F_1(z; y_1(z),\dots,y_d(z)) \\ &\vdots \\ y_d(z) &= F_d(z; y_1(z),\dots,y_d(z)) \end{cases}$$

où les $F_i(Z; Y_1,\dots Y_d)$ sont des polynômes de $\mathbb{Q}[Z, Y_1,\dots,Y_d]$ pour $i \in [d]$. En notant $\boldsymbol{y}(z) = (y_1(z),\dots,y_d(z))$, et $\boldsymbol{F} = (F_1,\dots,F_d)$, on peut réécrire le système sous la forme vectorielle :

$$\boldsymbol{y}(z) = \boldsymbol{F}(z; \boldsymbol{y}(z))\,. \tag{1.6}$$

Toute solution d'un système polynômial est algébrique. Il s'agit d'une conséquence de la théorie de l'élimination, que je ne présente pas ici. J'invite le lecteur à consulter [FS09, annexe B.1] pour une introduction à l'élimination par les résultants, et le Théorème 2 de [CLO07, p.122] pour une approche par les bases de Groebner.)

Proposition 1.85 (Voir par exemple [FS09, annexe B.1], [CLO07, The Elimination Theorem p.122]).
▶ Soit $(y_1(z), \dots, y_d(z))$ une solution du système 1.6. Alors pour tout $i \in [d]$, $y_i(z)$ est algébrique, c'est-à-dire qu'il existe un polynôme bivarié $P_i(z, Y_i)$ tel que $P_i(z, y_i(z)) = 0$. ◀

Proposition 1.86 (Unicité de la solution, [Drm97, FS09]).
▶ Si les coefficients des polynômes F_i sont positifs et si $\boldsymbol{F}(0; \boldsymbol{y}) = \mathbf{0}$, alors il existe une unique solution au système 1.6. ◀

Idée de la preuve. On montre que $\boldsymbol{F}$ est une contraction dans l'espace métrique complet des séries formelles, puis on applique le théorème du point fixe. ■

Définition 1.87 (Jacobien).
▶ Si $F_i(Z; Y_1, \dots Y_d)$ sont des polynômes de $\mathbb{Q}[Z, Y_1, \dots, Y_d]$ pour $i \in [d]$, nous notons $\boldsymbol{F}$ le vecteur $F_1, \dots, F_d)$. Alors la matrice jacobienne de $\boldsymbol{F}$ par rapport à y, notée $\mathtt{Jac}_{\boldsymbol{y}} \boldsymbol{F}(Z, Y_1, \dots, Y_d)$, est la matrice carrée, de dimension $d \times d$, dont les coefficients sont des polynômes définis par :

$$(\mathtt{Jac}_{\boldsymbol{y}} \boldsymbol{F}(Z, Y_1, \dots, Y_d))_{i,j} := \partial_{Y_j} F_i(Z; Y_1, \dots, Y_d)\,.$$

La matrice Jacobienne est notamment utile pour calculer des dérivées : si $\boldsymbol{y}(z) = (y_1(z), \dots, y_d(z))$ est un vecteur de séries vérifiant le système 1.6, alors en notant $\boldsymbol{y}'(z) = (y_1'(z), \dots, y_d'(z))$, nous avons :

$$\boldsymbol{y}'(z) = \mathtt{Jac}_{\boldsymbol{y}} \boldsymbol{F}(z, y_1(z), \dots, y_d(z)) \cdot \boldsymbol{y}'(z)\,.$$

Il s'agit simplement d'une notation condensée, sous forme de produit matriciel, de la règle de dérivation en chaîne pour des fonctions composées. ◀

1.3.2 Systèmes analytiques bien conditionnés, théorème de Drmota

Les systèmes qui nous ont intéressés dans cette thèse ont des propriétés supplémentaires qui assurent entre autres l'unicité de la solution, de la singularité dominante, et un comportement en racine carrée. Le cas analytique est étudié par Drmota dans [Drm97, Drm09] avec des paramètres u que nous n'utiliserons pas, et reformulé (sans paramètre) dans [BBY10] qui précise les conditions du théorème sur les points caractéristiques. Flajolet et Sedgewick donnent une preuve de ce résultat, qu'ils appellent théorème de Drmota-Lalley-Woods, dans le cas polynomial [FS09, p. 489].

Les systèmes étudiés sont appelés bien conditionnés par [BBY10]. Ils vérifient un ensemble de propriétés que l'on retrouve plus ou moins, sous une forme ou un autre, dans les conditions de [FS09], de [Drm97] ou de [Drm09].

Définition 1.88 ([BBY10]).
▶ Un système d'équations $\boldsymbol{y} = \boldsymbol{F}(z; \boldsymbol{y})$ est *bien conditionné* s'il vérifie les conditions suivantes :

a. chaque F_i est une série entière à coefficients positifs ;

b. $\boldsymbol{F}(z;\boldsymbol{y})$ est holomorphe dans un voisinage de l'origine ;

c. $\boldsymbol{F}(0;\boldsymbol{y}) = \mathbf{0}$;

d. le système n'est pas linéaire ;

e. le graphe de dépendance associé est fortement connexe ;

f. pour tout i, $F_i(z,\mathbf{0}) \neq 0$.

◀

La dernière condition est énoncée différemment chez [Drm97, Drm09], qui demandent qu'il existe un i tel que $F_i(z,\mathbf{0}) \neq 0$. Cette condition est équivalente par forte connexité, dès qu'on peut assurer qu'aucune composante solution ne peut être nulle. Dans plusieurs chapitres nous utiliserons cette version, en remplaçant la condition **f.** par la condition :

f'. il existe i tel que $F_i(z,\mathbf{0}) \neq 0$, et il existe un vecteur solution $\boldsymbol{y}(z)$ de séries non nulles, dont les coefficients de Taylor sont positifs.

De même, dans [Drm97], la condition de non linéarité semble plus forte, car elle demande l'existence d'un terme au moins carré $\partial^2 F_i/\partial y_j^2 \neq 0$ pour certains indices i et j ; ce n'est pas un problème, car par forte connexité, on peut obtenir un tel terme à partir d'un terme non linéaire, en utilisant les transformations présentées dans [BBY10] (*minimal self-substitution*). Dans [BBY10], les auteurs ont ajouté une condition ($\det(I - \mathtt{Jac}_{\boldsymbol{y}}\boldsymbol{F}(0;\mathbf{0})) \neq 0$), qui a été par la suite retirée, dans une version corrigée de l'article sur Arxiv. Enfin, dans [Drm97], l'auteur a ajouté la condition $\boldsymbol{F}_z(z,\boldsymbol{y}) \neq \mathbf{0}$. Ces deux dernières conditions sont en fait impliquées, dans le cas où $\boldsymbol{F}$ n'est pas nul, par la condition $\boldsymbol{F}(0;\boldsymbol{y}) = \mathbf{0}$ et l'analycité de $\boldsymbol{F}$ en $\mathbf{0}$ (car $\boldsymbol{F}$ s'écrit alors sous la forme $\boldsymbol{F}(z,\boldsymbol{y}) = z\boldsymbol{K}(z,\boldsymbol{y})$). Un travail de nettoyage des conditions des différentes versions du théorème est en cours de réalisation par Carine Pivoteau et Bruno Salvy, que je remercie pour m'avoir éclairé sur le sujet.

Le théorème de Drmota permet de démontrer que les solutions de systèmes bien conditionnés ont un comportement asymptotique en racine carrée. Plus précisément :

Théorème 1.89 (Théorème de Drmota).
▶ Soit $\boldsymbol{y} = \boldsymbol{F}(z;\boldsymbol{y})$ un système analytique bien conditionné. Alors :

a. Il existe un unique vecteur de séries entières $\boldsymbol{y}(z)$ solution du système $\boldsymbol{y} = \boldsymbol{F}(z;\boldsymbol{y})$.

b. Les coefficients des séries solutions sont positifs, et s'obtiennent par itération du système.

c. Tous les $y_i(z)$ ont le même rayon de convergence $0 < \rho < +\infty$, et présentent une singularité en $z = \rho$. De plus $\tau_i := y_i(\rho) < +\infty$.

Si de plus $(\rho,\boldsymbol{\tau})$ est dans le domaine de convergence de $\boldsymbol{F}(z;\boldsymbol{y})$ (c'est le cas automatiquement si le système est polynomial), alors :

d. $(\rho,\boldsymbol{\tau})$ vérifie le système caractéristique :

$$\begin{cases} \boldsymbol{\tau} &= \boldsymbol{F}(\rho;\boldsymbol{\tau}) \\ 0 &= \det(\mathtt{Id} - \mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{F}](\rho;\boldsymbol{\tau})) \end{cases}$$

e. Chaque $y_i(z)$ peut s'écrire sous la forme $y_i(z) = g_i(z) - h_i(z)\sqrt{1 - z/\rho}$, dans un voisinage de $z = \rho$ privé de $(\rho, +\infty)$, avec $g_i(z), h_i(z)$ des fonctions analytiques en $z = \rho$, et $h_i(\rho) \neq 0$.

f. Enfin, si à partir d'un certain rang tous les coefficients de Taylor des $y_i(z)$ sont non nuls, alors ρ est l'unique singularité dominante des $y_i(z)$, et il existe des constantes C_i telles que pour tout $i \in [d]$:

$$[z^n]y_i(z) \underset{n\to\infty}{\sim} C_i \rho^{-n} n^{-3/2}$$

◀

Le but de l'article [BBY10] est de préciser dans quels cas on peut être sûr que $(\rho, \boldsymbol{\tau})$ est bien un point caractéristique, et comment le caractériser parmi tous les points caractéristiques. Nous n'avons pas besoin de ces subtilités dans cette thèse, car nous utilisons généralement des systèmes polynomiaux, et dans le seul cas analytique, nous supposons que la singularité vérifie bien l'équation caractéristique. Par ailleurs le système provient souvent de la combinatoire, ce qui donne des renseignements supplémentaires sur ρ.

Première partie

Réductions d'arbres d'expressions en présence d'un élément absorbant

Introduction à la partie

Arbre d'expression

Les expressions sont une description simple d'objets informatiques, très souvent utilisées par les programmes qui s'adressent à un public non-expert. Nous pouvons citer le cas des expressions régulières qui sont présentes dans tous les langages de programmation ; mais aussi les formules logiques qui permettent aux utilisateurs de spécifier facilement les propriétés qu'ils veulent faire vérifier au programme. Bien que ces expressions soient souvent ensuite transformées dans la mécanique interne des programmes sous une forme plus adaptée au traitement algorithmique, elles sont largement choisies comme format d'entrée.

Plus généralement, les expressions consistent à décrire par une suite finie de symboles facile des objets ou des ensembles informatiques. Par exemple :

— les expressions régulières, que nous avons déjà évoquées, représentent des langages, des ensembles potentiellement infinis de mots. Par exemple, l'expression régulière $(a + b)^*$ représente l'ensemble de tous les mots possibles formés avec les lettres a et b.

— les formules logiques représentent des fonctions booléennes. Par exemple, la formule $x_1 \wedge x_2$ définit la fonction qui à (x_1, x_2) associe vrai si et seulement si x_1 et x_2 sont vrais.

— les formules de Presburger permettent de représenter des sous-ensembles de $\mathbb{N}^d$. Ainsi la formule $\phi(x) = \exists y,\ 2 \cdot y = x$ représente l'ensemble des nombres pairs.

— les expressions arithmétiques représentent des fonctions mathématiques. Ainsi l'expression $x/2$ représente la fonction qui à $x \in \mathbb{R}$ associe sa moitié.

— les formules LTL, qui suivent la syntaxe $\Phi := p|\Phi \wedge \Phi|\neg\Phi|\mathbf{X}\Phi|\Phi\mathbf{U}\Phi$, représentent des fonctions booléennes sur des mots infinis décrivant par exemple les états d'un système informatique.

Si ces exemples ne sont bien sûr pas exhaustifs, nous pouvons remarquer que leur description obéit à chaque fois à une syntaxe précise ; nous pouvons ainsi identifier des constantes (a et b dans les expressions régulières, les variables x_1 et x_2 dans les expressions logiques, …), et des opérateurs ($+, *, \wedge, \ldots$), qui ont une arité fixée (l'étoile de Kleene $*$ est d'arité 1 par exemple, c'est-à-dire qu'elle s'applique à une expression régulière, tandis que l'opérateur $\wedge$ est d'arité 2 et relie ainsi deux formules logiques).

Les différentes expressions vues ci-dessus partagent un autre point commun : elles se représentent facilement sous la forme d'un arbre ou d'un terme (voir Figure 1.3). Cette structure d'arbre permet notamment de se passer des parenthèses, qui sont nécessaires pour les expressions écrites en notation infixée. Ainsi dans la figure 1.3, nous avons pu retirer dans l'arbre les parenthèses de l'expression régulière $(a + b)^*$. Par ailleurs la représentation sous la forme d'un arbre offre des outils théoriques et pratiques pour manipuler ces expressions.

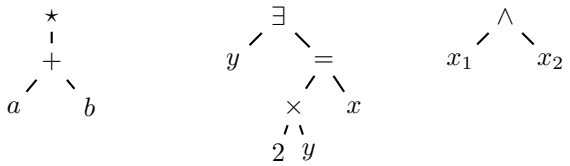


Figure 1.3. Exemple d'arbres représentant les expressions $(a + b)^*$, $\exists y,\ 2 \times y = x$, et $x_1 \wedge x_2$

Analyse en moyenne et benchmarks

En résumé, de nombreux programmes sont amenés à traiter des expressions qu'ils reçoivent en entrée, qu'ils les utilisent directement sous leur forme arborescente pour répondre à certaines questions (par exemple tester si une expression régulière reconnaît le mot vide, ou encore calculer la dérivation formelle d'une fonction), ou qu'ils les transforment en une autre structure plus adaptée au problème à résoudre (par exemple, une expression régulière encodée par un arbre est typiquement convertie en automate fini pour faire des recherches dans un texte). Nous souhaitons généralement estimer l'efficacité de tels programmes. La mesure classique pour étudier l'efficacité d'un outil est la complexité au pire cas, mais il y a souvent une grande différence entre ce qui est prédit dans le pire cas, et ce qui est observé à l'exécution. Une approche plus pragmatique consiste à tester les algorithmes sur de nombreuses entrées, appelées *benchmarks*, et comparer leurs performances expérimentalement. La difficulté est alors reportée sur la constitution d'une bibliothèque d'un grand nombre de tests pour comparer les algorithmes entre eux. Pour que ces comparaisons soient pertinentes, l'idéal est de tester les programmes sur des entrées issues de cas

Figure 1.4. La hauteur moyenne d'un arbre aléatoire uniforme de taille n est $\Theta(\sqrt{n})$.

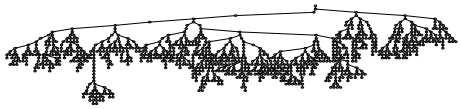


Figure 1.5. La hauteur moyenne d'un arbre aléatoire ABR de taille n est $\Theta(\log n)$.

d'utilisation réels, mais avoir accès à de tels exemples, en nombre et de toutes les tailles, peut s'avérer difficile. En pratique, on teste les algorithmes sur des expressions générées aléatoirement, car c'est un moyen simple d'avoir accès à un grand nombre d'expressions diverses. D'un point de vue plus théorique, il est par ailleurs possible d'analyser le comportement de l'algorithme sur ces entrées aléatoires ; il s'agit de la complexité en moyenne. Pour que les expressions aléatoires reflètent bien le comportement de l'algorithme, il faut cependant choisir une distribution sur les entrées de taille n, qui soit à même de favoriser des exemples *réels* ; lorsqu'aucune information supplémentaire n'est connue sur la distribution des entrées, il est naturel de se tourner vers deux distributions standard :

a. la distribution uniforme, pour laquelle chaque expression de taille n est équiprobable. Elle a l'avantage de donner la même probabilité à toute expression d'une taille fixée, et ainsi couvre bien toutes les possibilités.

b. la distribution ABR, qui favorise les arbres plus équilibrés. Le choix de cette distribution s'explique par la facilité avec laquelle on peut générer un arbre aléatoire de taille n : l'algorithme a une complexité linéaire, et est surtout élémentaire à implémenter [1]. Il est par ailleurs facile de paramétrer les probabilités de chaque opérateur pour ajuster la distribution des expressions tirées. L'Algorithme 1 représente une instance particulière de l'algorithme de génération d'un arbre d'expression suivant la distribution ABR, dans le cadre des expressions LTL : pour tirer un arbre de taille $n \geqslant 3$, on tire au hasard sa racine parmi tous les opérateurs possibles ; si l'opérateur tiré est unaire, on génère récursivement le fils de la racine ; si l'opérateur tiré est binaire, on tire uniformément au hasard les tailles des deux fils (qui doivent sommer à $n-1$), puis on les génère récursivement selon les tailles tirées.

Nous nous intéresserons à ces deux distributions dans cette partie de la thèse.

La distribution uniforme sur les expressions aléatoires uniformes a déjà été bien étudiée à l'aide des outils de la combinatoire analytique. Par exemple, la hauteur moyenne d'une expression aléatoire uniforme croît en $\Theta(\sqrt{n})$ [FO82]. Les auteurs de [FSS90] on montré aussi que si l'on compresse un arbre aléatoire uniforme, en identifiant des sous-expressions communes, le graphe acyclique correspondant a en moyenne une taille en $\Theta(\frac{n}{\sqrt{\log n}})$. Il existe aussi des résultats associés à des classes d'expressions plus spécifiques. Par exemple, la taille en moyenne de l'automate construit à partir d'une expression régulière uniforme est linéaire [Nic09, BMMR15]. Dans le contexte des expressions formelles, le coût moyen en temps et en espace du calcul de la dérivée d'une fonction aléatoire uniforme est en $\Theta(n^{3/2})$ [FS87]. Dans sa thèse [NT04], Michel Nguyên-Thê s'est intéressé à plusieurs arbres d'expressions uniformes (avec des poids sur les étiquettes des nœuds des arbres), notamment des expressions arithmétiques "min/+", "+/−", "+/×", dont il décrit précisément

1. Il est en fait aussi possible de générer des arbres uniformes en temps linéaire [Dev12], mais l'algorithme est beaucoup moins simple à comprendre et à implémenter.

la distribution asymptotique des valuations, lorsque la taille des arbres tend vers l'infini. De façon complémentaire à cette approche centrée sur la valuation exacte des arbres, [CFCS90] (resp. [NT04, chapitre 4]) étudient des phénomènes de simplification d'arbres binaires (resp. p-aires) plus généraux, lorsque tous les noeuds internes, étiquetés $\circ$, sont idempotents $\overset{\circ}{\underset{T\ T}{\wedge}} \equiv T$ pour tout T), ou encore nilpotents ($\overset{\circ}{\underset{T\ T}{\wedge}} \equiv e$ pour tout T, avec e appelé élément neutre). Les auteurs montrent que les simplifications induites par ces opérateurs réduisent assez peu les arbres considérés, d'un facteur constant en moyenne. Finalement, [NT04] exhibe un phénomène plus surprenant, dans le cas particulier des expressions booléennes : en utilisant 8 règles de simplification, dont la moitié provient des règles d'absorbance $x \vee \top \equiv \top \vee x \equiv \top$ et $x \wedge \bot \equiv \bot \wedge x \equiv \bot$, alors la réduction induite par ces règles simplifie en moyenne considérablement la taille des arbres booléens : la taille moyenne des arbres booléens après réduction tend vers une constante lorsque la taille des arbres tend vers l'infini. C'est ce phénomène que nous avons cherché à étudier, au-delà des expressions booléennes.

Le lecteur peut se référer à [CDM91] pour un survey sur différentes techniques symboliques utilisées pour étudier des arbres aléatoires suivant la distribution uniforme ou ABR.

La distribution ABR est une distribution tout aussi étudiée, bien que très différente de la distribution uniforme : par exemple, la hauteur typique d'un arbre à n nœuds suivant la distribution ABR est en $\Theta(\log(n))$ [Dev86], alors qu'elle est en $\Theta(\sqrt{n})$ pour la distribution uniforme. Les Figures 1.5 et 1.4 permettent de mieux apprécier la forme typique des arbres pour ces deux distributions. Le comportement en moyenne d'un algorithme sera donc sensiblement différent selon que la distribution des entrées suit la distribution uniforme ou la distribution ABR. Par exemple, l'automate de Glushkov associé à une expression régulière suivant la distribution ABR a en moyenne $\Theta(n^2)$ transitions [NPR10] ; tester l'égalité de deux arbres binaires non étiquetés de taille totale n se fait en $O(1)$ en moyenne pour la distribution uniforme, mais est en $\Theta(\log(n))$ en moyenne pour la distribution ABR [Mar91]. Dans le cas des expressions arithmétiques, les travaux de [NT04] sur des arbres binaires "+/-" montrent un comportement différent de la distribution uniforme en ce qui concerne la valuation moyenne de l'expression représentée ; par ailleurs, l'auteur exhibe un phénomène de seuil lorsque la probabilité de l'opérateur $+$ dépasse $3/4$: les distributions limites avant et après ce seuil sont différentes. Pour les fonctions booléennes, nous pouvons citer les travaux de [CGM11] sur la distribution ABR des arbres "et/ou". Les auteures montrent que pour tout choix de probabilité de l'opérateur $\vee$, la distribution ABR charge uniquement les fonctions $True$ et $False$, quand la taille des arbres générés tend vers l'infini. Dans la continuité des travaux de [CFCS90, NT04], [SCFC06] ont étudié, pour des arbres binaires suivant la distribution ABR, les simplifications induites lorsque tous les noeuds internes sont idempotents ou nilpotents : les auteurs démontrent, entre autres, que comme pour les arbres uniformes, ces réductions réduisent assez peu la taille des arbres, d'un facteur multiplicatif constant en moyenne.

Enfin, il est important de préciser que cette distribution est utilisée en pratique pour générer des expressions aléatoires afin de tester des algorithmes de vérification : par exemple, le générateur aléatoire de formules LTL de Spot [DLLF+16], ou encore l'utilitaire `ltlbtt` de TCS [Tau00, p.46], utilisent cette distribution pour générer des formules LTL aléatoires (voir aussi [DGV99] et l'Algorithme 1).

```
function RandomFormula(n) :
if n = 1 then
    p := random symbol in AP ∪ {⊤, ⊥};
    return p;
else if n = 2 then
    op := random operator in {¬, X, □, ◊};
    f := RandomFormula(1);
    return op f;
else
    op := random operator in {¬, X, □, ◊, ∧, ∨, →, ↔, U, R};
    if op in {¬, X, □, ◊} then
        f := RandomFormula(n − 1);
        return op f;
    else
        x := random integer in the interval [1, n − 2];
        f1 := RandomFormula(x);
        f2 := RandomFormula(n − x − 1);
        return (f1 op f2);
```

Algorithme 1 : Le pseudo-code utilisé par TCS [Tau00, p.46] pour générer une formule LTL aléatoire. C'est une instance particulière de l'algorithme de génération d'un arbre d'expression suivant la distribution ABR.

Redondance des expressions, élément absorbant

On fait généralement la distinction entre la syntaxe d'une expression, c'est-à-dire l'arbre étiqueté, et sa sémantique, c'est-à-dire l'objet qu'il représente. Pour de nombreuses classes d'expressions syntaxiques, on peut voir apparaître un phénomène de *redondance* : plusieurs expressions différentes représentent les mêmes objets. Par exemple :

— les expressions régulières $(a+b)^*$, $(a+b)^* + (bab)^*$ et $(a+b)(a+b)^* + \varepsilon$ représentent le même langage, celui de tous les mots sur l'alphabet $\{a, b\}$

— les formules logique $\top$, $\top \vee x$, et $\neg\bot$ représentent la même fonction booléenne, la tautologie vraie.

— les expressions arithmétiques $x/2$ et $4 \times (x/8)$ sont équivalentes.

Il apparaît ainsi immédiat que tirer au hasard une expression n'est pas du tout équivalent à tirer au hasard l'objet qu'elle représente ; certains objets sont sur-représentés par rapport à d'autres, selon le nombre d'expressions d'une taille donnée qui les décrivent. Dans ce cas, les analyses en moyennes et les benchmarks utilisant des expressions aléatoires sont-ils pertinents ? Nous allons voir dans cette partie de la thèse que la réponse à cette question est souvent négative.

Dans cette partie de la thèse, nous nous sommes restreints à la prise en compte d'une redondance très simple sur les arbres d'expressions, due à la présence d'un *élément absorbant*. Un arbre $\mathcal{P}$ est dit absorbant pour un opérateur $\circledast$ si toute expression de la forme $\overset{\circledast}{/\backslash}{}_{\mathcal{P}\ T}$ ou $\overset{\circledast}{/\backslash}{}_{T\ \mathcal{P}}$, avec T une expression quelconque, est équivalente sémantiquement à $\mathcal{P}$ (c'est-à-dire qu'elle représente le même objet que $\mathcal{P}$).

Dans tous les exemples mentionnés précédemment on trouve des éléments absorbants :

— l'expression $(a+b)^*$ est absorbante pour l'union $+$ pour les expressions régulières sur les lettres a et b. En effet, l'union d'un langage quelconque avec tous les mots possibles donne encore le langage de tous les mots. Ce n'est d'ailleurs pas le

seul élément absorbant pour l'union, il y a aussi $(a^* + b)^*$, $(a^*b^*)^*$, $((a + \varepsilon)b)^*$, $\varepsilon + (a + b)(a + b)^*$, …

— $\top$ est absorbant pour l'opérateur $\vee$. En effet, vrai ou n'importe quelle expression est toujours vrai. De même $\bot$ est absorbant pour $\wedge$. Et toute expression équivalente à $\bot$ (comme $x \wedge \neg x$) est absorbante pour $\wedge$.

— Pour LTL, $\top$ est aussi absorbant pour l'opérateur $\vee$, et $\bot$ est absorbant pour $\wedge$. Si nous voulons utiliser la sémantique propre de LTL, nous pouvons dire que $\bot$ est absorbant *à droite* pour $\mathbf{U}$: $\phi \,\mathbf{U}\, \bot$ est équivalent à $\bot$, mais nous sortons légèrement du cadre que nous avons étudié dans cette thèse, car l'élément n'est absorbant que d'un côté.

On se convainc facilement que le cadre de travail du chapitre 2 permet aussi d'étudier les éléments absorbants à droite. Pour les autres chapitres, il faut vérifier plus en détail les calculs, même si les résultats ne devraient a priori pas changer.

— 0 (et $x - x$, $\ln(1)$, …) est absorbant pour la multiplication $\times$.

Le but principal de cette partie sur les arbres d'expressions est de montrer que la seule présence d'un élément absorbant pour un opérateur fixé, dans la sémantique des arbres d'expressions, suffit en général à montrer que la distribution uniforme est dégénérée par rapport aux objets représentés. Par conséquent cette distribution ne devrait pas être utilisée pour l'étude en moyenne des algorithmes – qui étudient finalement plus la dégénérescence de la distribution uniforme que le comportement des algorithmes – ni pour la génération aléatoire pour les benchmarks. Ce phénomène est suffisamment général pour nous permettre d'exhiber un grand nombre de classes d'expressions qui rentrent dans ce cadre, tout en évitant d'avoir à adapter les preuves au cas par cas, selon la sémantique particulière de chacune des expressions étudiées.

Nous clôturons cette partie en étudiant la distribution ABR, et en montrant notamment que l'influence d'un élément absorbant sur cette distribution est a priori plus complexe que pour la distribution uniforme.

Plan de la partie

Dans le chapitre 2, nous étudions les conséquences de la présence d'un élément absorbant pour des arbres d'expressions, spécifiés par des grammaires algébriques, que l'on tire uniformément au hasard. Ces systèmes permettent notamment d'éviter certaines redondances ou certains motifs dans les arbres tirés au sort. Par exemple, si on souhaite considérer des expressions régulières sans deux étoiles de Kleene consécutives, on peut considérer la spécification suivante :

$$\begin{cases} \mathcal{L}_\mathcal{R} = \overset{\star}{\underset{\mathcal{S}}{|}} + \mathcal{S}, \\ \mathcal{S} = a + b + \varepsilon + \underset{\mathcal{L}_\mathcal{R}\ \mathcal{L}_\mathcal{R}}{\overset{+}{\wedge}} + \underset{\mathcal{L}_\mathcal{R}\ \mathcal{L}_\mathcal{R}}{\overset{\bullet}{\wedge}}. \end{cases}$$

Le résultat principal de ce chapitre est qu'en présence d'un élément absorbant, un arbre uniforme de taille n spécifié par un tel système est, sous certaines conditions sur le système, équivalent à un arbre plus petit, de taille en moyenne bornée par une constante lorsque n tend vers l'infini. Par ailleurs cet arbre équivalent s'obtient par réduction simple de l'arbre, en simplifiant par l'élément absorbant lorsqu'il se trouve sous l'opérateur $\circledast$. Nous montrons un résultat similaire pour les moments d'ordre supérieur associés à la taille après réduction. Les conditions sur le système sont suffisamment générales pour s'appliquer à de nombreux systèmes, notamment dans le domaine des expressions régulières : par exemple, notre résultat s'applique aux spécifications de [LS05], utilisées pour dénombrer les expressions régulières, et qui tentent d'éviter certaines redondances.

Dans le chapitre 3, nous étudions plus en détail le cas des expressions spécifiées par une équation unidimensionnelle, mais en autorisant, avec quelques contraintes, des arités non bornées pour les opérateurs, afin de pouvoir modéliser en partie l'associativité de certains opérateurs. Cela permet notamment d'utiliser la construction de type Seq dans les spécifications d'arbres, pour capturer aussi bien les expressions régulières spécifiées par la grammaire $\mathcal{L}_\mathcal{R} = a + b + \varepsilon + \overset{\star}{|}_{\mathcal{L}_\mathcal{R}} + \overset{\bullet}{/\!\backslash}_{\mathcal{L}_\mathcal{R}\ \mathcal{L}_\mathcal{R}} + \overset{+}{/\!\backslash}_{\mathcal{L}_\mathcal{R}\ \mathcal{L}_\mathcal{R}}$, que les expressions régulières spécifiées par

$$\mathcal{L}_\mathcal{R} = a + b + \varepsilon + \overset{\star}{|}_{\mathcal{L}_\mathcal{R}} + \overset{\bullet}{/\!\backslash}_{\mathcal{L}_\mathcal{R}\ \mathcal{L}_\mathcal{R}} + \overset{+}{/\!\backslash}_{\mathcal{L}_\mathcal{R}\ \mathcal{L}_\mathcal{R}} + \overset{+}{/\,|\,\backslash}_{\mathcal{L}_\mathcal{R}\ \mathcal{L}_\mathcal{R}\ \mathcal{L}_\mathcal{R}} + \sum_{n=4}^{\infty} \overset{+}{/\,|\,\backslash}_{\underbrace{\mathcal{L}_\mathcal{R}\ \dots\ \mathcal{L}_\mathcal{R}}_{n \text{ fois}}},$$

qui traduit le fait que l'opérateur $+$ peut avoir un nombre arbitraire de fils. Plus précisément, nous nous sommes intéressés à des classes combinatoire $\mathcal{E}$ définissant des variétés simples d'arbres, dont la série génératrice $T(z) := \sum_n t_n z^n$, où t_n compte le nombre d'arbres de $\mathcal{E}$ de taille n, satisfait donc une équation unidimensionnelle de la forme $T(z) = z\phi(T(z))$. Nous nous sommes toutefois restreints au cas simple où ϕ est une fonction analytique vérifiant des hypothèses analytiques classiques précises (appelées schéma d'inversion lisse). Par exemple, pour $\mathcal{L}_\mathcal{R}$ définie ci-dessus, la série génératrice vérifie

$$L_\mathcal{R}(z) = z(3 + L_\mathcal{R}(z) + L_\mathcal{R}(z)^2 + \frac{L_\mathcal{R}(z)^2}{1 - L_\mathcal{R}(z)})$$

Nous avons montré ainsi que sous ces hypothèses, qui sont naturelles et vérifiées dans de nombreux cas, on observe le même phénomène de dégénérescence de la distribution uniforme : un arbre d'expressions uniforme de taille n est équivalent à un arbre de taille plus petit, dont la taille moyenne tend vers une constante lorsque n tend vers l'infini (et de même pour les moments d'ordre supérieur).

Si les réductions induites par un élément absorbant sont très générales et s'appliquent à de nombreuses classes d'expressions, elles sont trop grossières pour capturer précisément la sémantique des arbres étudiés. Ainsi, les constantes annoncées dans les deux précédents articles sont énormes, même pour des équations unidimensionnelles très simples (plusieurs millions pour les expressions régulières sur un alphabet à deux lettres). Ceci pourrait remettre en question notre mise en garde de ne pas utiliser en pratique la distribution uniforme sur des arbres d'expressions : il pourrait finalement s'agir d'une objection purement théorique, que l'on n'observera jamais en pratique sur des arbres de taille raisonnable. Dans le chapitre 4, nous étudions plus en détail la sémantique des expressions régulières uniformes, en introduisant des règles supplémentaires spécifiques aux expressions régulières. Le but est de raffiner les résultats annoncés au chapitre précédent, et faire baisser les constantes en jeu. Nous montrons notamment qu'une expression régulière uniforme de taille n est équivalente lorsque n tend vers l'infini à une expression de taille environ 77 en moyenne. Cette constante est confirmée par des courbes expérimentales (voir Figure 1.6).

Enfin, dans le chapitre 5, nous étudions ce qui se passe lorsque la distribution des arbres n'est plus uniforme, mais la distribution ABR, pour des arbres d'expressions unaire-binaires. Cette fois, le comportement asymptotique des tailles après réduction est plus complexe, et dépend de la probabilité $p_\circledast$ d'apparition de l'opérateur absorbant, et la probabilité p_I d'apparition d'un opérateur unaire. Intuitivement, il y

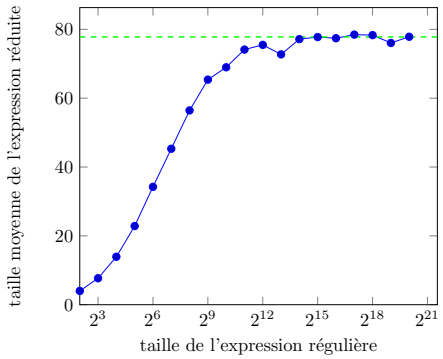


Figure 1.6. Taille moyenne après réduction, observée expérimentalement sur des expressions régulières uniformes, à deux lettres (échelle logarithmique)

a peu de réductions lorsque $p_\circledast$ est trop petit, et il y a beaucoup de simplifications lorsque $p_\circledast$ est proche de 1. Nous avons détaillé précisément la dynamique de la taille asymptotique moyenne après réduction, en fonction de la probabilité $p_\circledast$ d'apparition de l'opérateur absorbant, et identifié deux seuils de changement de phase dans le comportement de la réduction, confirmés par les expériences. Ainsi la taille réduite moyenne d'un arbre aléatoire de taille n suivant la distribution ABR, a un comportement asymptotique décrit par le schéma suivant, selon la probabilité d'apparition de l'opérateur absorbant :

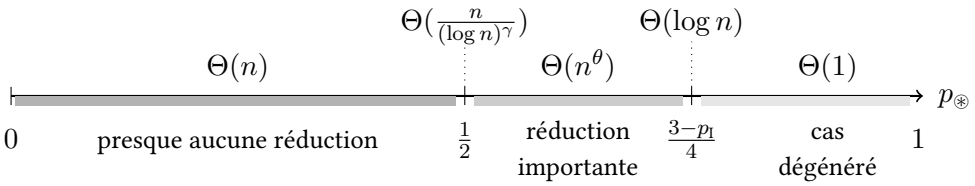


Figure 1.7. Comportement des tailles réduites asymptotiques dans le cas des arbres d'expressions suivant la distribution ABR

Cette partie est un travail en commun avec Cyril Nicaud et Pablo Rotondo. Le chapitre 2 a fait l'objet d'une publication à DLT en 2020, et d'une publication en version longue (en tant que special issue de DLT) à IJFCS en 2021. Le chapitre 3 a été publié à MFCS en 2020, le chapitre 4 à CSR en 2021, et le chapitre 5 à STACS en 2021. Je conclus cette introduction avec le tableau suivant qui résume les contributions de chaque chapitre :

Chap.	Type d'équation(s)	Réduction considérée	Distribution	Taille moyenne
2	système polynomial	élément absorbant	uniforme	constante
3	unidimensionnelle analytique	élément absorbant	uniforme	constante
4	$\mathcal{L}_\mathcal{R} = a + b + \varepsilon + \overset{\star}{\mid}_{\mathcal{L}_\mathcal{R}} + \overset{\bullet}{/\!\backslash}_{\mathcal{L}_\mathcal{R}\,\mathcal{L}_\mathcal{R}} + \overset{+}{/\!\backslash}_{\mathcal{L}_\mathcal{R}\,\mathcal{L}_\mathcal{R}}$	détection heuristique d'arbres universels	uniforme	constante ($\simeq 77$)
5	unidimensionnelle de degré 2	élément absorbant	ABR	de constante à linéaire

Table 1.1. Résumé des contributions des différents chapitres

Ce chapitre est en grande partie issu de l'article [KNR20] et de sa version longue [KNR21].

2

Réduction d'expressions spécifiées par un système

2.1 Introduction

Dans ce chapitre nous nous intéressons aux phénomènes de réduction dus à la présence d'un élément absorbant pour des expressions spécifiées par des systèmes combinatoires. Un exemple classique d'un tel système est la description de la classe $\mathcal{L}_{\mathcal{R}}$ des arbres décrivant des expressions régulières :

$$\mathcal{L}_{\mathcal{R}} = a + b + \varepsilon + \overset{\star}{\underset{\mathcal{L}_{\mathcal{R}}}{|}} + \overset{\bullet}{\underset{\mathcal{L}_{\mathcal{R}}}{/\!\backslash}}\!\!\underset{\mathcal{L}_{\mathcal{R}}}{} + \overset{+}{\underset{\mathcal{L}_{\mathcal{R}}}{/\!\backslash}}\!\!\underset{\mathcal{L}_{\mathcal{R}}}{}. \tag{$\star$}$$

Il s'agit d'une spécification récursive à une dimension, dans le sens où elle ne dépend que d'une classe combinatoire inconnue $\mathcal{L}_{\mathcal{R}}$, qui définit toutes les expressions régulières syntaxiquement correctes. Plus généralement, nous nous intéressons aux classes définies par des systèmes combinatoires, afin d'étudier les conséquences de la présence d'un élément absorbant sur des sous-ensembles d'expressions ; nous pouvons par exemple grâce aux systèmes étudier les expressions régulières qui n'ont pas deux étoiles de Kleene successives, en utilisant le système :

$$\begin{cases} \mathcal{L}_{\mathcal{R}} = \overset{\star}{\underset{\mathcal{S}}{|}} + \mathcal{S}, \\ \mathcal{S} = a + b + \varepsilon + \overset{+}{\underset{\mathcal{L}_{\mathcal{R}}}{/\!\backslash}}\!\!\underset{\mathcal{L}_{\mathcal{R}}}{} + \overset{\bullet}{\underset{\mathcal{L}_{\mathcal{R}}}{/\!\backslash}}\!\!\underset{\mathcal{L}_{\mathcal{R}}}{}. \end{cases} \tag{$\star\star$}$$

Le résultat principal de ce chapitre est le suivant :

Théorème 2.1 (Énoncé informel)**.**
▶ Pour toute classe d'expressions définies par un système d'équations combinatoires, en présence d'un élément absorbant pour un opérateur d'arité au moins 2, si certaines conditions sur le système sont respectées, alors une expression uniforme de taille n est équivalente à une expression de taille bornée par une constante en moyenne quand n tend vers l'infini. De plus tous les moments d'ordre supérieur associés à cette taille sont aussi bornés. Par ailleurs le calcul de cette expression équivalente se fait par un simple parcours de l'arbre, en temps linéaire. ◀

Autrement dit, la distribution uniforme est dégénérée pour ces expressions : les objets qui ne peuvent être décrits par des arbres de petite taille sont sous-représentés. Ce résultat remet en question la pertinence de l'analyse en moyenne des algorithmes prenant en entrée de telles expressions, avec la distribution uniforme : si on réduit d'abord une expression de taille n en prenant en compte l'élément absorbant, alors sa taille est bornée par une constante en moyenne. L'algorithme a donc une complexité en moyenne linéaire.

Bien sûr des conditions sur le système sont nécessaires pour imposer ce résultat. Certaines sont naturelles et seront naturellement vérifiées par la plupart des systèmes décrivant des arbres d'expressions. Une des conditions s'assure notamment qu'on ait identifié un élément absorbant pertinent qui ne soit pas évité par le système. Une autre condition sur le système (la forte connexité) s'avèrera moins naturelle, mais nécessaire pour résoudre le problème en toute généralité, sans avoir à s'intéresser au cas particulier de la sémantique de chaque classe d'expressions étudiée.

2.2 Système combinatoire d'arbres

Dans cette partie, nous définissons proprement ce que nous entendons par systèmes combinatoires d'expressions.

2.2.1 Définition des arbres d'expressions sous forme de classe combinatoire et de systèmes

Dans toute cette partie, les arbres considérés sont des *arbres planaires étiquetés enracinés*, c'est-à-dire qui vérifient les conditions suivantes :

a. (enracinés) tout nœud possède un unique père, à l'exception d'un seul nœud, appelé racine

b. (planaires) les fils d'un nœud sont ordonnés, si bien que les deux arbres  et  sont distincts

c. (étiquetés) chaque nœud est étiqueté par un symbole.

La *taille* d'un arbre est son nombre de nœuds. Pour un nœud N d'un arbre planaire enraciné, nous appelons *arité* de N le nombre de fils du nœud N. Les nœuds d'arité 0 sont classiquement appelés feuilles, les nœuds d'arité 1 sont appelés nœuds unaires, ceux d'arité 2 nœuds binaires. Nous employons le terme arité et non degré comme en théorie des graphes, car pour un arbre, vu comme un graphe, le degré d'un nœud différent de la racine est son arité plus un.

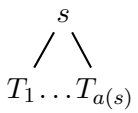
Nous nous intéressons à des arbres d'expressions qui suivent une syntaxe précise. Les étiquettes des nœuds sont associées à une arité fixe : par exemple, pour la syntaxe des expressions logiques, l'opérateur $\neg$ est unaire, tandis que l'opérateur $\wedge$ est binaire. Dans la suite de cette section, chaque opérateur aura une seule arité fixée.

Les systèmes de séries génératrices associées seront donc polynomiaux. Comme nous l'avons expliqué dans l'introduction, nous pourrions vouloir considérer que $\wedge$ a une arité non fixée pour modéliser son associativité. Nous sortons du cadre des systèmes polynomiaux dans ce cas, cf le chapitre 3.

De façon plus formelle, soit S un ensemble fini d'éléments appelés *opérateurs*, et soit a une fonction de S dans $\mathbb{N}$. Pour un symbole d'opérateur s, la valeur $a(s)$ est appelée l'*arité* de l'opérateur s, et désigne le nombre de fils qu'un opérateur peut avoir.

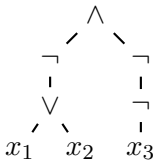
Définition 2.2 .
▶ L'ensemble des expressions $\mathcal{T}(S, a)$ associées à la paire (S, a) est défini par induction :

— pour tout opérateur $s_0 \in S$ d'arité 0, $s_0 \in \mathcal{T}(S, a)$;
— pour tout opérateur $s \in S$ et $T_1, \cdots, T_{a(s)} \in \mathcal{T}(S, a)$, alors $(s, T_1, \ldots, T_{a(s)}) \in \mathcal{T}(S, a)$. On représente ce tuple formel sous la forme d'un arbre :



◀

Exemple 2.3 .
▶ L'arbre suivant :



représente l'expression $(\neg(x_1 \vee x_2)) \wedge \neg\neg x_3$ associée à $S = \{\wedge, \vee, \neg, x_1, x_2, x_3\}$ avec $a(\wedge) = a(\vee) = 2$, $a(\neg) = 1$ et $a(x_1) = a(x_2) = a(x_3) = 0$: ◀

Soit $\square$ un nouveau symbole de feuille qui n'est pas dans S. On étend la fonction d'arité à ce nouveau symbole en posant $a(\square) = 0$.

Définition 2.4 (Expression incomplète).
▶ Une *expression incomplète* sur S est une expression de $\mathcal{T}(S \cup \{\square\}, a)$. La taille d'une expression incomplète est son nombre de nœuds dans S (les feuilles $\square$ ne sont pas comptées). ◀

Ne pas compter les $\square$ dans la taille d'un arbre simplifiera les notations lors de l'expression de la taille d'un arbre provenant d'une substitution.

Autrement dit, une expression incomplète est simplement un arbre de $\mathcal{T}(S, a)$ dont certaines feuilles sont étiquetées par un nouveau symbole $\square$ (par exemple l'arbre T_1 de la figure 2.1). De façon informelle, ces arbres incomplets représentent des expressions partielles, chaque feuille $\square$ étant destinée à être complétée par une expression. Une expression est simplement une expression incomplète qui ne contient aucune feuille $\square$. Nous appelerons parfois *expression complète* une telle expression par opposition aux expressions incomplètes.

Les expressions incomplètes sont aussi appelées des contextes.

Définition 2.5 (Arité d'une expression incomplète).
▶ Si T est une expression incomplète sur S, nous appelons arité de T, noté $a(T)$ son nombre de feuilles $\square$. ◀

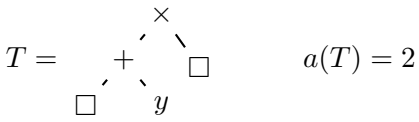


Figure 2.1. Exemple d'expression incomplète, d'arité 2

Cette définition est consistante avec l'arité d'un symbole, en considérant un symbole $s \in S$ d'arité $a(s)$ comme une expression incomplète constituée d'une racine

étiquetée par s, et ayant $a(s)$ enfants étiquetés par $\square$ (par exemple l'opérateur $\wedge$ peut être vu comme $\overset{\wedge}{\square\ \square}$).

Dans toute la suite, l'arité des symboles de S est sous-entendue fixée. On désigne par $\mathcal{T}_\square(S) := \mathcal{T}(S \cup \{\square\}, a)$ (resp. $\mathcal{T}(S) := \mathcal{T}(S, a)$) l'ensemble des expressions incomplètes (resp. complètes) sur S. Naturellement, $\mathcal{T}(S) \subseteq \mathcal{T}_\square(S)$.

Remarque 2.6 .
▶ Les expressions incomplètes sont très proches des motifs définis par [Mar92], dans le cadre de la recherche de motifs dans des arbres. L'auteur utilise le joker (*wildcard*) $*$ comme symbole spécial de feuille, à la place de $\square$. ◀

Définition 2.7 (Substitution d'arbres).
▶ Si T est une expression incomplète sur S, d'arité $t = a(T)$, et $T_1, \ldots, T_t$ sont des expressions (possiblement incomplètes) sur S, alors $T[T_1, \ldots, T_t]$ désigne l'expression obtenue en remplaçant la i-ième feuille $\square$ par T_i pour chaque $i \in [t]$ (en ordonnant les t feuilles $\square$ par ordre d'apparition dans le parcours en profondeur de l'arbre, en traitant les fils d'un nœud de gauche à droite).

Nous généralisons cette notation à des ensembles d'expressions : si $\mathcal{T}_1, \ldots, \mathcal{T}_t$ sont des ensembles d'expressions (incomplètes), alors

$$T[\mathcal{T}_1, \ldots, \mathcal{T}_t] := \{T[T_1, \ldots, T_t] : T_1 \in \mathcal{T}_1, \ldots, T_t \in \mathcal{T}_t\}$$

désigne l'ensemble des expressions obtenues en remplaçant la i-ième feuille $\square$ par un élément de $\mathcal{T}_i$. ◀

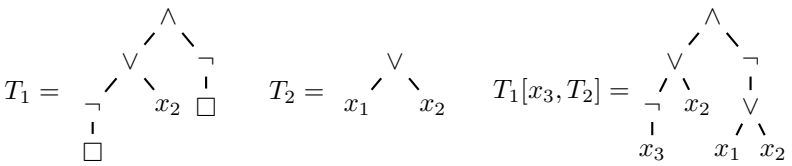


Figure 2.2. Exemple de substitution de deux expressions complètes x_3 et T_2 dans une expression incomplète T_1

Définition 2.8 (Règle).
▶ Une *règle* de dimension $m \geqslant 1$ sur S est une expression incomplète $T \in \mathcal{T}_\square(S)$ dans laquelle chaque nœud $\square$ est de surcroît étiqueté par un entier dans $[m]$. ◀

Intuitivement, une règle précise pour chaque nœud $\square$ les ensembles d'arbres (numérotés de 1 à m) autorisés pour effectuer une substitution. Une autre façon de présenter une règle est de la voir comme un tuple $\mathcal{M} = (T, i_1, \ldots, i_t)$, où T est une expression incomplète d'arité t, et $i_1, \ldots, i_t$ sont les étiquettes associées aux t nœuds $\square$ de T, ordonnés par parcours en profondeur. L'arité $a(\mathcal{M})$ d'une règle $\mathcal{M}$ est l'arité de son expression incomplète, et $\textsc{ind}(\mathcal{M}) = (i_1, \ldots, i_t)$ est le tuple des étiquettes de ses nœuds $\square$. Les règles sont les briques de base pour décrire un système combinatoire sur des classes d'arbres :

Définition 2.9 (Système combinatoire d'arbres).
▶ Un *système combinatoire d'arbres* $\mathcal{E} = [E_1, \ldots, E_m]$ de dimension $m \geqslant 1$ sur S est une liste ordonnée de m ensembles finis de règles de dimension m.

Nous écrivons en pratique un tel système combinatoire d'arbres sous la forme d'une spécification récursive de classes combinatoires d'arbres $\mathcal{L}_1, \ldots, \mathcal{L}_m$:

$$\begin{cases} \mathcal{L}_1 &= \displaystyle\bigcup_{(T,i_1,\ldots,i_t)\in E_1} T[\mathcal{L}_{i_1},\ldots,\mathcal{L}_{i_t}] \\ &\vdots \\ \mathcal{L}_m &= \displaystyle\bigcup_{(T,i_1,\ldots,i_t)\in E_m} T[\mathcal{L}_{i_1},\ldots,\mathcal{L}_{i_t}]. \end{cases} \tag{2.1}$$

L'idée derrière cette écriture est qu'un système combinatoire définit des familles d'arbres décrites de façon récursive. ◀

Pour définir formellement les familles d'arbres décrites par la spécification de l'équation (2.1), nous avons besoin d'introduire les itérations de systèmes. Il s'agit de l'opération qui remplace simultanément, dans toutes les règles de $\mathcal{E}$, et pour tout $i \in [m]$, chaque feuille $\square$ étiquetée i par toutes les règles de E_i. Commençons d'abord par un exemple pour illustrer cette notion : nous voulons définir une opération, appelée itération, qui une fois appliquée au système "fil rouge" des expressions régulières sans étoile de Kleene successives,

$$\begin{cases} \mathcal{L}_1 = \overset{\star}{|}_{\mathcal{L}_2} + \mathcal{L}_2, \\ \mathcal{L}_2 = a + b + \varepsilon + \overset{+}{/\!\backslash}_{\mathcal{L}_1\ \mathcal{L}_1} + \overset{\bullet}{/\!\backslash}_{\mathcal{L}_1\ \mathcal{L}_1}. \end{cases} \tag{$\star\star$}$$

fournit le système suivant :

$$\begin{cases} \mathcal{L}_1 = \overset{\star}{|}_{a} + \overset{\star}{|}_{b} + \overset{\star}{|}_{\varepsilon} + \overset{\star}{|}\overset{+}{/\!\backslash}_{\mathcal{L}_1\ \mathcal{L}_1} + \overset{\star}{|}\overset{\bullet}{/\!\backslash}_{\mathcal{L}_1\ \mathcal{L}_1} + a + b + \varepsilon + \overset{+}{/\!\backslash}_{\mathcal{L}_1\ \mathcal{L}_1} + \overset{\bullet}{/\!\backslash}_{\mathcal{L}_1\ \mathcal{L}_1} \\ \mathcal{L}_2 = a + b + \varepsilon + \overset{+}{/\!\backslash}_{\mathcal{L}_2\ \mathcal{L}_2} + \overset{+}{/\!\backslash}\overset{\star}{|}_{\mathcal{L}_2} + \overset{+}{/\!\backslash}\overset{\star}{|}_{\mathcal{L}_2} + \overset{+}{/\!\backslash}\overset{\star}{|}\overset{\star}{|}_{\mathcal{L}_2\ \mathcal{L}_2} + \overset{\bullet}{/\!\backslash}_{\mathcal{L}_2\ \mathcal{L}_2} + \overset{\bullet}{/\!\backslash}\overset{\star}{|}_{\mathcal{L}_2} + \overset{\bullet}{/\!\backslash}\overset{\star}{|}_{\mathcal{L}_2} + \overset{\bullet}{/\!\backslash}\overset{\star}{|}\overset{\star}{|}_{\mathcal{L}_2\ \mathcal{L}_2}. \end{cases} \tag{$\star\star\star$}$$

Plus rigoureusement, nous définissons l'itération d'un système comme suit :

Définition 2.10 (Itération d'un système).
▶ Les itérés $\mathcal{E}^t$ du système $\mathcal{E} = \{E_1, \ldots, E_m\}$ sont définis par récurrence :

a. si $t = 1$, $\mathcal{E}^1 := \mathcal{E} = \{E_1, \ldots, E_m\}$.

b. si $t \geqslant 1$, on pose $\mathcal{E}^t = \{E_{1,t}, \ldots, E_{m,t}\}$.
Alors le système $\mathcal{E}^{t+1} := \{E_{1,t+1}, \ldots, E_{m,t+1}\}$ est défini par les égalités :

$$E_{i,t+1} := \bigcup_{(T,i_1,\ldots,i_s)\in E_{i,t}} T\left[E_{i_1},\ldots,E_{i_s}\right], \quad \text{pour tout } i \in [m];$$

autrement dit, les règles de $\mathcal{E}^{t+1}$ sont obtenues à partir des règles de $\mathcal{E}^t$ en remplaçant leurs feuilles $\square$ par des règles de $\mathcal{E}$ compatibles. ◀

On pourrait définir $\mathcal{E}^0 := \{\{\boxed{1}\}, \ldots, \{\boxed{m}\}\}$. Dans ce cas, la définition par récurrence donne $E_{i,1} = E_i$ pour tout $i \in [m]$, et on retrouve bien $\mathcal{E}^1 = \mathcal{E}$.

Nous pouvons dès lors définir proprement les familles d'arbres définies par la spécification de l'Eq. (2.1) :

Définition 2.11 (Solution d'un système combinatoire).
▶ Soit $\mathcal{E} = [E_1, \dots, E_m]$ un système combinatoire de dimension $m \geqslant 1$ sur S. Nous définissons comme dans la Définition 2.10 les itérations $\mathcal{E}^t = \{E_{1,t}, \dots, E_{m,t}\}$ du système pour $t \geqslant 1$. Nous posons alors pour tout $t \geqslant 1$ et $i \in [m]$,

$$\mathcal{L}_{i,t} = E_{i,t} \cap \mathcal{T}(S)$$

l'ensemble des règles de $E_{i,t}$ qui sont complètes. Remarquons que par la Définition 2.10, nous avons $\mathcal{L}_{i,t} \subseteq \mathcal{L}_{i,t+1}$ pour tout $t \geqslant 1$ et $i \in [m]$. Nous posons alors pour tout $i \in [m]$

$$\mathcal{L}_i := \bigcup_{t=1}^{\infty} \mathcal{L}_{i,t}\,.$$

Nous disons alors que le système combinatoire $\mathcal{E}$ décrit les classes combinatoires $(\mathcal{L}_1, \dots, \mathcal{L}_m)$. ◀

Remarque 2.12.
▶ La construction par itération présentée ci-dessus est très classique en informatique pour définir des objets comme des plus petits points fixes de fonctions Scott-continues sur des ordres partiels complets dirigés. Nous avons choisi de ne pas utiliser ce formalisme pour définir les classes combinatoires définies par des systèmes combinatoires, car il n'était pas nécessaire dans notre contexte. ◀

2.2.2 Exemples de systèmes

Exemple 2.13 (Expressions régulières).
▶ La spécification définissant les expressions régulières

$$\mathcal{L}_{\mathcal{R}} = a + b + \varepsilon + \overset{\star}{|}_{\mathcal{L}_{\mathcal{R}}} + \overset{\bullet}{\wedge}_{\mathcal{L}_{\mathcal{R}}\,\mathcal{L}_{\mathcal{R}}} + \overset{+}{\wedge}_{\mathcal{L}_{\mathcal{R}}\,\mathcal{L}_{\mathcal{R}}}$$

s'écrit dans la représentation par tuple sous la forme :

$$E_1 = \{a, b, \varepsilon, \left(\overset{\star}{|}_{\square}, 1\right), \left(\overset{\bullet}{\wedge}_{\square\,\square}, 1, 1\right), \left(\overset{+}{\wedge}_{\square\,\square}, 1, 1\right)\},$$

et s'écrit dans la représentation par feuilles numérotées :

$$E_1 = \{a, b, \varepsilon, \overset{\star}{|}_{\boxed{1}}, \overset{\bullet}{\wedge}_{\boxed{1}\,\boxed{1}}, \overset{+}{\wedge}_{\boxed{1}\,\boxed{1}}\}.$$

◀

Exemple 2.14.
▶ Représentons dans notre formalisme le système de Eq. (⋆⋆) de la page 51. Pour ce système $m = 2$, et nous pouvons renommer sans perte de généralité $\mathcal{L}_1 := \mathcal{L}_{\mathcal{R}}$ et $\mathcal{L}_2 := \mathcal{S}$:

$$\begin{cases} \mathcal{L}_1 = \overset{\star}{|}_{\mathcal{L}_2} + \mathcal{L}_2, \\ \mathcal{L}_2 = a + b + \varepsilon + \overset{+}{\wedge}_{\mathcal{L}_1\,\mathcal{L}_1} + \overset{\bullet}{\wedge}_{\mathcal{L}_1\,\mathcal{L}_1}. \end{cases} \tag{⋆⋆}$$

La représentation du système par tuple est alors la suivante :

$$E_1 = \left\{ \left(\overset{\star}{|}_{\square}, 2 \right), (\square, 2) \right\}, \quad E_2 = \left\{ (a), (b), (\varepsilon), \left(\overset{\bullet}{\wedge}_{\square\,\square}, 1, 1 \right), \left(\overset{+}{\wedge}_{\square\,\square}, 1, 1 \right) \right\} ;$$

tandis que sa représentation par feuilles $\square$ numérotées est la suivante :

$$E_1 = \left\{ \overset{\star}{|}_{\boxed{2}}, \boxed{2} \right\}, \quad E_2 = \left\{ a, b, \varepsilon, \overset{\bullet}{\wedge}_{\boxed{1}\,\boxed{1}}, \overset{+}{\wedge}_{\boxed{1}\,\boxed{1}} \right\}.$$

Les représentations sous la forme d'un système comme dans Eq. (⋆⋆) sont en pratique plus lisibles, mais le formalisme par tuple ou feuilles $\square$ numérotées est plus approprié aux preuves. ◀

Exemple 2.15 (Expressions arithmétiques).
▶ La spécification suivante évite la redondance liée à l'associativité du $+$ et du $\times$ dans les expressions arithmétiques, en les forçant à s'empiler sur la gauche :

$$\begin{cases} \mathcal{L} & = \mathcal{L}_+ + \mathcal{L}_\times + \mathcal{L}_2, \\ \mathcal{L}_\times & = \overset{+}{\wedge}_{\mathcal{L}\,\mathcal{L}_2} + \overset{+}{\wedge}_{\mathcal{L}\,\mathcal{L}_\times} \\ \mathcal{L}_+ & = \overset{\times}{\wedge}_{\mathcal{L}\,\mathcal{L}_2} + \overset{\times}{\wedge}_{\mathcal{L}\,\mathcal{L}_+} \\ \mathcal{L}_2 & = 0 + 1 + x + \overset{-}{|}_{\mathcal{L}} \end{cases}$$

La représentation du système par feuilles $\square$ numérotées est alors la suivante, en renommant $\mathcal{L}_1 := \mathcal{L}$, $\mathcal{L}_3 := \mathcal{L}_+$ et $\mathcal{L}_4 := \mathcal{L}_\times$:

$$E_1 = \{\boxed{2}, \boxed{3}, \boxed{4}\}, E_2 = \left\{ 0, 1, x, \overset{-}{|}_{\boxed{1}} \right\}, E_3 = \left\{ \overset{+}{\wedge}_{\boxed{1}\,\boxed{2}}, \overset{+}{\wedge}_{\boxed{1}\,\boxed{4}} \right\}, E_4 = \left\{ \overset{\times}{\wedge}_{\boxed{1}\,\boxed{2}}, \overset{\times}{\wedge}_{\boxed{1}\,\boxed{3}} \right\}$$

◀

2.2.3 Traduction en système de séries génératrices

Le système combinatoire (2.1) se traduit automatiquement en un système sur des séries génératrices :

$$\begin{cases} L_1(z) & = \displaystyle\sum_{(T, i_1, \dots, i_{a(T)}) \in E_1} z^{|T|} L_{i_1}(z) \cdots L_{i_{a(T)}}(z) \\ & \vdots \\ L_m(z) & = \displaystyle\sum_{(T, i_1, \dots, i_{a(T)}) \in E_m} z^{|T|} L_{i_1}(z) \cdots L_{i_{a(T)}}(z). \end{cases} \tag{2.2}$$

où pour $i \in [m]$, $L_i(z)$ compte les arbres dans la classe $\mathcal{L}_i$, si le système décrit chaque arbre de la classe $\mathcal{L}_i$ de façon non ambiguë.

2.2.4 Systèmes mal fondés et cas pathologiques

Les systèmes décrivant des arbres comme Eq. (2.1) ne sont pas toujours bien fondés. Parfois, certaines équations sont redondantes ou inutiles. Pire, il peut arriver qu'aucune solution non vide n'existe. Nous n'aborderons pas dans cette thèse toutes

les conditions qui assurent le caractère bien fondé d'un système combinatoire (le lecteur peut se référer pour cela à [AU72, FS87, PSS12]). Cependant, nous allons nous fixer quelques conditions suffisantes sur les systèmes pour qu'ils ne soient pas pathologiques. Nous présentons dans cette partie des exemples de mauvaises propriétés que nous souhaitons éviter.

Ambiguité. Si le système est ambigu, il peut générer un arbre de plusieurs façons différentes. Dans ce cas, le système de séries génératrices qui en découle compte les arbres avec comme multiplicité leur nombre de dérivations possibles (et les séries ne sont pas bien définies si un arbre possède une infinité de façons d'être dérivé). Par exemple, le système $\left\{\mathcal{L}_1 = a + \overset{\star}{\underset{\mathcal{L}_1}{|}} + \overset{\star}{\underset{\mathcal{L}_2}{|}}; \mathcal{L}_2 = \overset{\star}{\underset{\mathcal{L}_1}{|}} + a + b + \varepsilon\right\}$ est ambigu car l'expression $\overset{\star}{\underset{a}{|}}$ peut être produite de deux façons différentes à partir de $\mathcal{L}_1$. Ainsi $\overset{\star}{\underset{a}{|}}$ est compté deux fois dans la série génératrice $L_1(z)$.

Composantes vides. Certaines spécifications décrivent des classes d'expressions vides. Par exemple, dans le système $\left\{\mathcal{L}_1 = \overset{\bullet}{\underset{\mathcal{L}_1}{\wedge}}{}_{\mathcal{L}_2};\ \mathcal{L}_2 = a + b + \varepsilon + \mathcal{L}_1\right\}$, la seule solution possible est $\mathcal{L}_1 = \emptyset$ et $\mathcal{L}_2 = \{a, b, \varepsilon\}$.

Nous pourrions utiliser les algorithmes classiques de théorie des langages pour éliminer les règles unité pour palier ce problème ; cependant leur élimination peut modifier la forte connexité du système.

Cycles de règles unité. Le *graphe de dépendance unité* $\mathcal{G}_\square(\mathcal{E})$ d'un système $\mathcal{E}$ de taille m est un graphe orienté, d'ensemble de sommets $[m]$, et tel qu'il y a une arête $i \to j$ si et seulement si $(\square, j) \in E_i$ (autrement dit nous avons une équation de la forme $\mathcal{L}_i = \ldots + \mathcal{L}_j + \ldots$ dans le système). Une telle règle est appelée *règle unité*, en suivant le formalisme des grammaires hors-contextes. On dit alors que $\mathcal{L}_i$ dépend directement de $\mathcal{L}_j$. Par exemple, dans l'Équation ($\star\star$), la classe $\mathcal{L}_\mathcal{R}$ dépend directement de $\mathcal{S}$. Si le graphe $\mathcal{G}_\square(\mathcal{E})$ présente un cycle, les équations appartenant à ce cycle sont inutiles ou mal définies. Par exemple, considérons le système suivant et son graphe de dépendance des règles unité :

$$\begin{cases} \mathcal{L}_1 &= \mathcal{L}_2 + \overset{\star}{\underset{\mathcal{L}_1}{|}} \\ \mathcal{L}_2 &= a + b + \varepsilon + \mathcal{L}_1 \end{cases}$$

Le graphe de dépendance unité n'est pas acyclique, et il y a ainsi une infinité de façons de dériver a à partir de $\mathcal{L}_2$: $\mathcal{L}_2 \to a$, $\mathcal{L}_2 \to \mathcal{L}_1 \to \mathcal{L}_2 \to a \ldots$ Les séries solutions du système traduit ne peuvent pas avoir tous leurs coefficients positifs. Nous ne considérerons par la suite que des systèmes avec un graphe de dépendance unité acyclique.

Graphe non fortement connexe. Le *graphe de dépendance* $\mathcal{G}(\mathcal{E})$ du système $\mathcal{E}$ est un graphe orienté, d'ensemble de sommets $[m]$, avec un arête $i \to j$ dès qu'il existe une règle $\mathcal{M} \in E_i$ telle que $j \in \textsc{ind}(\mathcal{M})$: autrement dit $\mathcal{L}_j$ apparaît quelque part dans l'équation définissant $\mathcal{L}_i$. En toute généralité, certaines parties du système peuvent être indépendantes, ce qui rend le système plus complexe à étudier. Une condition suffisante pour empêcher cela est d'imposer au système d'avoir un graphe de dépendance fortement connexe. Contrairement aux précédentes conditions, cette restriction n'est pas nécessaire dans le sens où des systèmes bien fondés et très naturels ne sont pas fortement connexes. Cependant elle se révèlera utile pour la

Nous rencontrerons d'ailleurs un système non fortement connexe dans la conclusion de ce chapitre, ou dans le chapitre 4.

preuve du Théorème 2.32. Un système et son graphe de dépendance sont représentés dans la Figure 2.3.

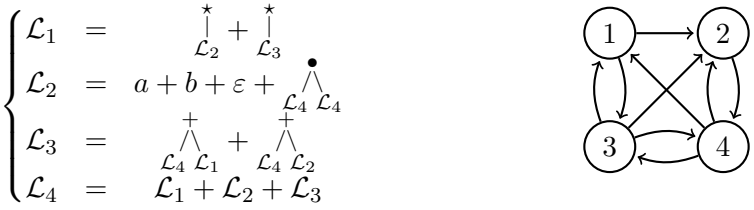


Figure 2.3. Un système et son graphe de dépendance fortement connexe.

2.3 Formalisme, hypothèses de travail et simplifications

2.3.1 Élément absorbant

Le but de cette section est de décrire le cadre dans lequel nous nous plaçons : le type de systèmes que nous allons regarder, et la simplification syntaxique associée que nous allons étudier.

Soit $\mathcal{E}$ un système combinatoire d'arbres sur S, de dimension m, décrivant les classes combinatoires $(\mathcal{L}_1, \ldots, \mathcal{L}_m)$.

Définition 2.16 (Ensemble défini par un système).
▶ Un ensemble d'expressions $\mathcal{L}$ sur S est *défini par* $\mathcal{E}$ s'il existe un sous-ensemble non vide I de $[m]$ tel que $\mathcal{L} = \cup_{i\in I}\mathcal{L}_i$. ◀

À partir de maintenant, nous supposons que S contient un opérateur $\circledast$, d'arité au moins 2. Nous supposons de plus qu'il existe une expression complète $\mathcal{P} \in \mathcal{T}(S)$, telle que toute expression ayant pour racine $\circledast$, et $\mathcal{P}$ parmi ses fils directs, soit équivalente à $\mathcal{P}$, une fois interprétée avec la sémantique des expressions considérées. Ainsi, l'interprétation de $\mathcal{P}$ est absorbante pour l'opérateur $\circledast$. On dit que $\mathcal{P}$ est un *élément absorbant* pour l'opérateur $\circledast$, appelé *opérateur absorbant.*

Exemple 2.17 .
▶ Reprenons notre exemple directeur des expressions régulières sans deux étoiles consécutives, défini par le système Eq. ($\star\star$), avec $\mathcal{L} = \mathcal{L}_{\mathcal{R}}$. Les expressions régulières peuvent être interprétées comme des langages réguliers sur l'alphabet $\{a, b\}$. Comme le langage $(a + b)^\star$ est absorbant pour l'union, nous pouvons prendre pour $\mathcal{P}$ l'expression associée à $(a + b)^\star$, et alors $+$ est l'opérateur absorbant. ◀

La *simplification* d'une expression complète T est l'expression complète $\sigma(T)$ obtenue à partir de T en appliquant de bas en haut la règle de réécriture suivante, en notant a l'arité de $\circledast$:

$$\overset{\circledast}{T_1 \cdots T_a} \rightsquigarrow \mathcal{P}\text{ , si } T_i = \mathcal{P} \text{ pour un } i \in \{1, \ldots, a\}.$$

Définition 2.18 (Définition formelle de σ).
▶ La simplification $\sigma(T, \mathcal{P}, \circledast)$ de T, ou juste $\sigma(T)$ quand le contexte est clair, est définie de façon inductive par :

— si T est une feuille, $\sigma(T) = T$

— sinon :

$$\sigma((\oplus, T_1, \dots, T_d)) = \begin{cases} \mathcal{P} & \text{si } \oplus = \circledast \text{ et } \exists i, \sigma(T_i) = \mathcal{P}, \\ (\oplus, \sigma(T_1), \dots, \sigma(T_d)) & \text{sinon.} \end{cases}$$

◀

Définition 2.19 (Expression entièrement réductible).

▶ Une expression T est dite *entièrement réductible* si $\sigma(T) = \mathcal{P}$. ◀

Nous avons besoin pour la suite d'imposer certaines conditions sur le système $\mathcal{E}$. Certaines viennent de la discussion de la Section 2.2.4, tandis que d'autres sont indispensables aux techniques de combinatoire analytique utilisées dans les preuves.

Définition 2.20 (Condition (**H**) sur les systèmes).

▶ Un système $\mathcal{E}$ satisfait *l'hypothèse* (**H**) si :

(**H**$_1$) Le graphe $\mathcal{G}(\mathcal{E})$ est fortement connexe et $\mathcal{G}_{\square}(\mathcal{E})$ est acyclique.

(**H**$_2$) Le système est *apériodique* : il existe un indice N tel que pour tout $n \geqslant N$, chaque classe $\mathcal{L}_i$ solution du système contient une expression de taille n.

(**H**$_3$) Il existe un sous-arbre T d'une règle de E, de racine $\circledast$, ayant au moins deux enfants distincts T' and T'', tels qu'il soit possible de produire une expression entièrement réductible à partir de T', et $a(T'') \geqslant 1$ (autrement dit T'' contient une feuille $\square$).

Le fait que $\circledast$ soit d'arité 2 est indépendant du fait qu'une règle du système (i.e. un arbre incomplet) ait au moins deux feuilles $\square$.

(**H**$_4$) Le système n'est *pas linéaire* : il existe une règle d'arité au moins 2.

(**H**$_5$) Le système est *non ambigu* : toute expression complète peut se dériver du système d'au plus une seule façon.

◀

Les conditions (**H**$_1$) et (**H**$_5$) ont déjà été abordées dans la Section 2.2.4. La condition (**H**$_4$) exclut les systèmes qui ne génèrent que des listes, c'est-à-dire des arbres dont les nœuds internes sont d'arité 1, ou plus généralement des arbres dégénérés en peigne dont la taille est linéaire en le nombre de nœuds, comme pour le système $\mathcal{L} = \overset{+}{\underset{\mathcal{L}}{\wedge}}\!\!\underset{a}{\backslash} + b$. Sans la condition (**H**$_3$), notre résultat ne peut s'appliquer car le système pourrait empêcher toute simplification en interdisant à l'élément absorbant d'apparaître sous $\circledast$, ou encore empêcher les simplifications de couper des arbres de taille arbitraire. Enfin, les conditions (**H**$_1$) et (**H**$_2$) sont nécessaires pour que l'analyse reste générale tout en restant abordable.

Remarque 2.21 (Condition (**H**$_3$)).

▶ Si la condition (**H**$_3$) n'est pas vérifiée parce que le système empêche $\mathcal{P}$ d'apparaître sous $\circledast$, il est toujours possible d'essayer d'identifier un autre élément absorbant $\mathcal{P}$ qui convienne. Par exemple, il est aisé de décrire un système pour les expressions régulières qui évite les motifs $(a+b)^*$ et $(b+a)^*$ sous l'opérateur $+$; cependant pour invalider l'hypothèse (**H**$_3$), il faut qu'il soit aussi capable d'éviter tous les absorbants $\mathcal{P}$ possibles : $(a^* + b^*)^*$, $((a+\varepsilon)+(b+\varepsilon))^*$, etc. Comme le problème de savoir si une expression régulière quelconque reconnaît tous les mots est PSPACE-complet [MS72], il semble difficile de réussir à éviter tous les éléments absorbants sans contraindre fortement la forme des expressions régulières décrites. ◀

Cf. la discussion en fin de chapitre, à la section 2.7.

Remarque 2.22 (Condition ($\mathbf{H}_5$)).
▶ La condition ($\mathbf{H}_5$) peut être remplacée par une condition plus faible d'ambiguïté finie : s'il existe une constante k telle que tout arbre du système puisse être produit d'au plus k façons différentes par le système, le résultat principal reste vrai par une très légère adaptation de la preuve finale. Nous somme restés sur la condition de non ambiguïté pour simplifier les hypothèses. ◀

Exemple 2.23.
▶ Tous les exemples de la section 2.2.2 vérifient les hypothèses ($\mathbf{H}$). ◀

2.3.2 Systèmes propres

Le but de cette section est de préparer $\mathcal{E}$ à l'analyse analytique, en éliminant les règles unité et en remontant les feuilles $\Box$ à profondeur 1. En effet, l'absence de règles unité est cruciale pour appliquer le théorème de Drmota, qui assure que tous les arbres du système ont un comportement asymptotique commun, et la profondeur 1 des feuilles $\Box$ permet de spécifier plus facilement la réduction, en se concentrant sur les racines des règles.

Définition 2.24 (Système combinatoire propre).
▶ Un système combinatoire d'arbres $\mathcal{E}$ est *propre* si toutes les feuilles $\Box$ de ses règles sont à profondeur exactement 1 (donc les feuilles $\Box$ sont situées directement sous les racines des règles). En particulier, $\mathcal{E}$ ne contient aucune règle unité. ◀

Une transformation très utile pour rendre un système propre est l'itération de système, définie à la Définition 2.10. Le lemme suivant étudie les propriétés qui sont conservées en itérant un système :

Lemme 2.25.
▶ Si un ensemble d'expressions $\mathcal{L}$ est défini par un système $\mathcal{E}$, il est aussi défini par les itérés $\mathcal{E}^t$, pour tout $t \geqslant 1$. De plus, si $\mathcal{E}$ satisfait l'hypothèse ($\mathbf{H}$), alors chaque itération $\mathcal{E}^t$ satisfait aussi ($\mathbf{H}$), à l'exception de la condition ($\mathbf{H}_1$), car il se peut que $\mathcal{G}(\mathcal{E}^t)$ ne soit pas fortement connexe. ◀

Idée de la preuve. On prouve facilement par récurrence que $\mathcal{E}^t$ et $\mathcal{E}$ définissent les mêmes familles d'arbres. Le point clef pour vérifier l'hypothèse ($\mathbf{H}$) de $\mathcal{E}^t$ est de remarquer que les arêtes du graphe de dépendance $\mathcal{G}(\mathcal{E}^t)$ (resp. $\mathcal{G}_\Box(\mathcal{E}^t)$) correspondent aux chemins de longueur t du graphe de dépendance $\mathcal{G}(\mathcal{E})$ (resp. $\mathcal{G}_\Box(\mathcal{E})$). ■

Exemple 2.26.
▶ Le système ($\star\star\star$) que nous avons obtenu après itération du système $\star\star$ n'est plus fortement connexe. En effet, le graphe de dépendance du système de départ ($\star\star$) était :

tandis que le graphe de dépendance du système itéré est :

◀

Pour rendre $\mathcal{E}$ propre, nous procédons en deux étapes : nous éliminons dans un premier temps les règles unité (Lemme 2.27), puis nous expliquons comment remonter les feuilles $\Box$ jusqu'aux racines (Lemme 2.28).

Lemme 2.27 (Élimination des règles unité)**.**
▶ Si le système $\mathcal{E}$ satisfait $(\mathbf{H})$, alors nous pouvons trouver une itération $\mathcal{E}'$ de $\mathcal{E}$ sans règle unité qui définit les mêmes arbres d'expression et satisfait toujours l'hypothèse $(\mathbf{H})$. ◀

Démonstration. Par le Lemme 2.25, toutes les conditions de $(\mathbf{H})$ sont remplies en itérant le système à l'exception de la condition de forte connexité. Nous voulons prouver qu'il est possible de trouver une itération de $\mathcal{E}$ qui soit aussi fortement connexe. Soit $k \geqslant 1$. Les arêtes du graphe de dépendance du système $\mathcal{E}^k$ correspondent aux chemins de longueur k dans le graphe de dépendance initial de $\mathcal{G}(\mathcal{E})$. Comme $\mathcal{G}_{\Box}(\mathcal{E})$ est acyclique par $(\mathbf{H}_1)$, si k est plus grand que le nombre d'équations m, alors l'itéré $\mathcal{E}^k$ ne peut plus contenir de règle unité.

Soit $N > m$ un entier premier avec les longueurs des cycles élémentaires de $\mathcal{G}(\mathcal{E})$. Puisque $N > m$, $\mathcal{E}^N$ ne contient pas de règle unité. Montrons que le graphe $\mathcal{G}(\mathcal{E}^N)$ associé au système itéré N fois est fortement connexe. Soient deux indices $i, j \in [m]$. Par forte connexité de $\mathcal{G}(\mathcal{E})$, les sommets i et j appartiennent chacun à un cycle élémentaire.

Si i et j sont dans le même cycle élémentaire $\mathcal{C}$ de longueur ℓ, alors ils appartiennent toujours au même cycle après N itérations, car N est premier avec ℓ. Donc i et j sont connectés entre eux dans $\mathcal{G}(\mathcal{E}^N)$. Ainsi tous les sommets d'un cycle élémentaire de $\mathcal{G}(\mathcal{E})$ restent connectés dans $\mathcal{G}(\mathcal{E}^N)$.

Supposons dans le cas contraire que i et j sont dans deux cycles élémentaires distincts $\mathcal{C}_i$ and $\mathcal{C}_j$. Il suffit de montrer par le cas précédent qu'il existe un chemin de i jusqu'à $\mathcal{C}_j$ dans $\mathcal{G}(\mathcal{E}^N)$. Comme le graphe de départ $\mathcal{G}(\mathcal{E})$ est fortement connexe, il existe un chemin simple de i à j de longueur $t < m$. Puisque $N > m > t$, on peut étendre ce chemin en un chemin de longueur N en ajoutant $N - t$ étapes dans le cycle élémentaire $\mathcal{C}_j$. Cela signifie que i est relié par une arête à un sommet de $\mathcal{C}_j$ dans le graphe $\mathcal{G}(\mathcal{E}^N)$. Ceci conclut la preuve. ■

Le lemme suivant explique comment faire remonter les feuilles $\Box$ à profondeur 1. L'idée principale consiste à diminuer la profondeur des feuilles $\Box$ en découpant les règles en petits morceaux, comme dans l'exemple suivant :

$$\mathcal{L}_1 = \overset{\star}{\underset{\mathcal{L}_3}{|}} + \overset{\bullet}{\underset{\mathcal{L}_1}{\overset{\star}{|}}}\!\!\mathcal{L}_2 \quad \to \quad \begin{cases} \mathcal{L}_1 = \overset{\star}{\underset{\mathcal{L}_3}{|}} + \overset{\bullet}{\mathcal{K}}\!\!\mathcal{L}_2 \\ \mathcal{K} = \overset{\star}{\underset{\mathcal{L}_1}{|}}. \end{cases}$$

Depth First Search

Pour faciliter la rédaction de la preuve, nous utilisons le formalisme des feuilles $\Box$ étiquetées par un numéro. Nous rappelons que dans cette représentation, chaque règle $(T, i_1, \ldots, i_t) \in E_j$ est représentée par un seul arbre T, dans lequel la k-ième feuille $\Box$ est étiquetée par le nombre i_k (en ordonnant les feuilles selon l'ordre DFS du parcours en profondeur, en traitant les fils de gauche à droite). Cette feuille étiquetée est représentée dans l'arbre par le symbole $\boxed{i_k}$. Cela nous permet de découper les règles à partir de certains sous-arbres, sans avoir à expliciter les indices. Nous rappelons que $\textsc{ind}(T) = (i_1, \ldots, i_t)$ désigne toujours le tuple des étiquettes des feuilles $\Box$ de T, lues dans l'ordre DFS.

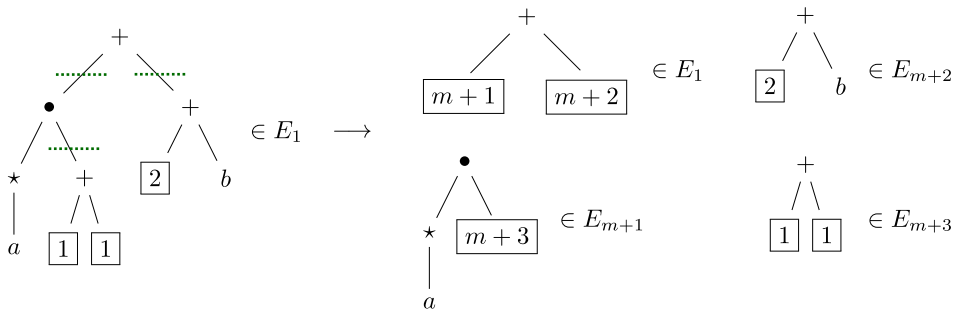


Figure 2.4. Principe du découpage effectué pour remonter les feuilles □ à profondeur 1

Lemme 2.28 (Remontée des □).
▶ Tout système satisfaisant (**H**) est équivalent à un système vérifiant aussi (**H**), dans lequel toutes les feuilles □ sont à profondeur 1. ◀

Démonstration. Par le lemme 2.27, nous pouvons supposer que le système ne contient aucune règle unité. Soit $T \in E_j(\mathcal{E})$ une règle de E_j pour un certain $j \in [m]$, qui possède une feuille □ à une profondeur au moins 2. Soient $T_1, \dots, T_s$ les fils de la racine de T qui ont une arité plus grande que 1 et qui ne sont pas réduits à une feuille □. Soit $\tilde{T}$ l'arbre obtenu à partir de T en remplaçant simultanément T_i par la feuille $\boxed{m+i}$ pour chaque $i \in [s]$. Le nouveau système $\mathcal{E}'$ que nous créons s'obtient à partir de $\mathcal{E}$ en remplaçant T par $\tilde{T}$ dans E_j, et en ajoutant les nouveaux ensembles de règles $E_{m+i}(\mathcal{E}') = \{T_i\}$ pour chaque i. Ainsi $\mathcal{E}'$ est de dimension $m+s$; par ailleurs il ne contient pas de règle unité, car les arbres $T_1, \dots, T_s$ ne sont pas des feuilles □.

Ce carré n'est pas le qed de fin de preuve.

Nous répétons cette construction tant qu'il reste des règles avec une feuille □ de profondeur au moins 2. Ce processus termine puisque la somme des profondeurs de toutes les feuilles □ de profondeur au moins 2 diminue à chaque itération. Bien que le nombre d'équations augmente à chaque étape, on vérifie facilement par induction que les m premières équations définissent toujours les classes $\mathcal{L}_1, \dots, \mathcal{L}_m$.

Nous observons aussi facilement qu'à chaque étape, le nouveau système vérifie la condition (**H**). ■

Comme la construction du Lemme 2.28 n'introduit aucune règle unité, nous pouvons appliquer à un système le Lemme 2.27 puis le Lemme 2.28 pour le rendre propre :

Proposition 2.29 .
▶ Si $\mathcal{L}$ est défini par un système $\mathcal{E}$ vérifiant (**H**), alors $\mathcal{L}$ est aussi défini par un système propre $\mathcal{E}'$ vérifiant (**H**). ◀

Remarque 2.30 .
▶ Si $\mathcal{E}$ est propre et vérifie la condition (**H**), alors l'hypothèse ($\mathbf{H}_3$) implique qu'il existe une règle de racine $\circledast$, ayant deux enfants T' and T'', tels qu'il soit possible de produire une expression entièrement réductible à partir de T', et T'' est une feuille □. ◀

Nous aurons besoin de la variante suivante d'un système propre vérifiant les hypothèses (**H**) :

Proposition 2.31 .
▶ Si $\mathcal{L}$ est défini par un système $\mathcal{E}$ vérifiant $(\mathbf{H})$, alors $\mathcal{L}$ est aussi défini par un système propre $\mathcal{E}'$ vérifiant $(\mathbf{H})$, ainsi que l'hypothèse supplémentaire suivante :

$(\mathbf{H}'_3)$ Il existe une règle d'arité non nulle, de racine $\circledast$, et dont un fils est un arbre complet entièrement réductible.

◀

Démonstration. Par la Proposition 2.29, $\mathcal{L}$ est défini par un système propre $\mathcal{E}'' = \{E_1, \dots, E_m\}$. Par l'hypothèse $(\mathbf{H}_3)$ et la Remarque 2.30, il existe une règle T dans un certain E_j, de racine $\circledast$, tel que T a deux fils distincts T' et T'', tels que le système peut engendrer à partir de T' un arbre entièrement réductible T_R, et T'' est une feuille $\square$. Par le Lemme 2.27, en itérant le système suffisamment de fois, on obtient un système équivalent $\tilde{\mathcal{E}}$, avec une règle de la forme $\tilde{T} \in \tilde{E}_j$, avec $\tilde{T}$ qui a parmi ses fils deux arbres T_1, T_2, avec $T_1 = T_R$ entièrement réductible, et $a(T_2) \geqslant 1$ (par forte connexité).

Comme après itération, les feuilles $\square$ ne sont plus à profondeur 1, il faut rendre le système propre à nouveau par le Lemme 2.28. La construction dans la preuve de ce dernier lemme laisse inchangés les arbres complets qui sont fils des racines des règles : le nouveau système $\mathcal{E}'$ obtenu vérifie donc les hypothèses $(\mathbf{H})$ et l'hypothèse supplémentaire $(\mathbf{H}'_3)$. ■

2.4 Résultat principal

Notre résultat principal met en évidence le caractère dégénéré des expressions aléatoires uniformes en présence d'un élément absorbant :

Théorème 2.32 .
▶ Soit $\mathcal{E}$ un système combinatoire d'expressions sur S, vérifiant $(\mathbf{H})$, et possédant un élément absorbant $\mathcal{P}$ associé à l'opérateur absorbant $\circledast$. Soit $\mathcal{L}$ une classe définie par $\mathcal{E}$. Alors il existe une constante $C > 0$ telle que pour toute taille $n \in \mathbb{N}$, une expression tirée uniformément parmi les expressions de $\mathcal{L}$ de taille n a en moyenne une taille bornée par C après simplification par l'élément absorbant. ◀

Nous consacrons le reste de cette section à la preuve du Théorème 2.32. Nous pouvons supposer que $\mathcal{E}$ est propre, par la Proposition 2.29. La Condition $(\mathbf{H}_5)$ impose au système d'être non ambigu, donc nous pouvons directement obtenir à partir de $\mathcal{E}$ un système d'équations sur les séries de comptage des différentes classes, comme expliqué dans la Section 2.2.3. À partir de maintenant, pour améliorer la lisibilité et la concision des formules, nous utiliserons une notation vectorielle : $\boldsymbol{L}(z)$ désigne ainsi le vecteur de séries formelles $(L_1(z), \dots, L_m(z))$, et nous pouvons ainsi réécrire le système de l'Eq. (2.2) sous la forme plus compacte

$$\boldsymbol{L}(z) = z\,\boldsymbol{\phi}(z; \boldsymbol{L}(z)), \tag{2.3}$$

où $\boldsymbol{\phi} = (\phi_1, \dots, \phi_m)$ et $\phi_i(z; \boldsymbol{y}) = \sum_{(T, i_1, \dots, i_{a(T)}) \in E_i} z^{|T|-1} \prod_{j=1}^{a(T)} y_{i_j}$. Remarquons que le terme z en facteur dans l'Eq. (2.3) compte la racine de chaque règle (il y a toujours une racine pour chaque règle, puisque les règles unité ont été éliminées). Ce z en facteur implique notamment l'unicité de la solution $\boldsymbol{L}(z)$ au système (2.3) (voir la Proposition 1.86 des préliminaires).

Nous introduisons maintenant le paramètre combinatoire défini par $\chi(t) := |\sigma(t)|$ qui représente la taille d'un arbre après réduction. Nous associons à $\mathcal{L}$ la série bivariée $L(z,u) = \sum_{T\in\mathcal{L}} u^{|\sigma(T)|} z^{|T|}$, qui compte en u la taille des arbres de $\mathcal{L}$ après réduction. De même, pour chaque classe d'arbre $\mathcal{L}_i$ du système, nous associons la série bivariée $L_i(z,u) := \sum_{T\in\mathcal{L}_i} u^{|\sigma(T)|} z^{|T|}$, et nous introduisons le vecteur des séries bivariées :

Voir le chapitre 1 des préliminaires sur la méthode symbolique.

$$\boldsymbol{L}(z,u) = (L_1(z,u), \dots, L_m(z,u))\,,$$

Ainsi, si l'union $\mathcal{L} = \cup_{i\in I}\mathcal{L}_i$ est disjointe, la série génératrice $L(z,u)$ s'exprime comme une somme de composantes de ce vecteur : . $L(z,u) = \sum_{i\in I} L_i(z,u)$.

Nous rappelons que par la formule de l'Eq. (1.1) des préliminaires, la taille moyenne des arbres de la classe $\mathcal{L}$ après réduction est donnée par

Comme nous cherchons juste à borner asymptotiquement la taille moyenne, il ne sera pas nécessaire que l'union soit disjointe.

$$\mathbb{E}_n(|\sigma|) = \frac{[z^n]\partial_u L(z,u)\big|_{u=1}}{[z^n]L(z)}\,, \tag{2.4}$$

où pour alléger les notations, nous avons décidé de noter $\mathbb{E}_n(|\sigma|)$ à la place de $\mathbb{E}_n(\chi)$. Pour exploiter cette formule, nous procédons en deux temps :

a. Nous étudions d'abord le dénominateur de l'Équation 2.4, qui correspond au nombre total d'arbres dans la classe $\mathcal{L}$. Cette partie est simple, car les hypothèses (**H**) nous placent automatiquement dans le cadre du théorème de Drmota (cf le Théorème 1.89 des préliminaires), qui nous donne des informations précises sur la forme du vecteur de solutions $\boldsymbol{L}(z)$. Les détails se trouvent en Section 2.4.1.

b. L'étude du numérateur est donc le point central de la preuve. Une première difficulté est d'arriver à rendre le paramètre χ hérité en marquant avec la variable u les nœuds qui restent après réduction ; pour cela nous avons besoin d'introduire des classes supplémentaires dans le système décrivant $\mathcal{L}$. S'il existe bien une version à plusieurs variables du théorème de Drmota, qui gère les systèmes polynomiaux avec des paramètres, celle-ci ne s'applique pas à ce nouveau système, car il n'est plus nécessairement fortement connexe. Pour étudier le comportement des solutions, nous allons plutôt borner celles-ci par des fonctions dont nous connaissons le comportement, en l'occurence les composantes de $\boldsymbol{L}(z)$ qui ont été étudiées à la première étape. Les détails se trouvent en Section 2.4.2.

2.4.1 Analyse du dénominateur

La proposition 2.33 ci-dessous caractérise le comptage des arbres pour chaque classe $\mathcal{L}_j$, qui intervient au dénominateur de l'Eq. (2.4). Ici $\mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{\phi}](z;\boldsymbol{y})$ désigne la matrice Jacobienne du système, de dimensions $m\times m$, définie par $\mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{\phi}](z;\boldsymbol{y})_{i,j} = \partial_{y_j}\phi_i(z;\boldsymbol{y})$. Cette matrice intervient naturellement dans l'analyse des systèmes d'équations.

Proposition 2.33 (Comportement asymptotique commun des arbres de la spécification).
▶ Les composantes du vecteur solution $\boldsymbol{L}(z)$ du système d'équations (2.3) partagent la même singularité $\rho \in (0,1]$, et de plus $\tau_j := L_j(\rho) < \infty$ pour $1 \leqslant j \leqslant m$. La singularité ρ et les valeurs $\boldsymbol{\tau} = (\tau_j)_j$ satisfont le système caractéristique

$$\begin{cases} \boldsymbol{\tau} &= \rho\,\boldsymbol{\phi}(\rho;\boldsymbol{\tau}) \\ 0 &= \det(\mathtt{Id}_{m\times m} - \rho\,\mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{\phi}](\rho;\boldsymbol{\tau})) \end{cases}.$$

De plus, pour tout j, il existe deux fonctions $g_j(z)$ et $h_j(z)$, analytiques en $z = \rho$, telles que dans un voisinage de $z = \rho$, avec $z \notin [\rho, +\infty)$,

$$L_j(z) = g_j(z) - h_j(z)\sqrt{1 - z/\rho}\,, \quad \text{avec } h_j(\rho) \neq 0\,.$$

Enfin, nous avons les équivalents asymptotiques $[z^n]L_j(z) \sim C_j \rho^{-n}/n^{3/2}$ pour une certaine constante positive $C_j > 0$. ◀

Démonstration. Il suffit de vérifier que le théorème de Drmota s'applique (voir Théorème 1.89 des préliminaires).

Pour suivre les notations de la Proposition 1.89, nous écrivons le système sous la forme $[L_1, \dots, L_m] = \boldsymbol{F}(z; L_1, \dots, L_m)$, où $\boldsymbol{F}(z; y_1, \dots, y_m) := z\,\boldsymbol{\phi}(z; y_1, \dots, y_m)$.

La fonction $\boldsymbol{F}(z, \boldsymbol{y})$ est analytique en $(z, \boldsymbol{y}) = (0, \mathbf{0})$, puisque chaque composante est un polynôme, et $\boldsymbol{F}(0, \mathbf{0}) \equiv \mathbf{0}$. Le graphe de dépendance de $\boldsymbol{F}$ en $\boldsymbol{y}$ est fortement connexe par $(\mathbf{H}_1)$. Les coefficients de Taylor de $\boldsymbol{F}$ sont positifs, $\boldsymbol{F}(0, \boldsymbol{y}) \equiv 0$ puisqu'il y a un terme z en facteur (car le système est propre), $\boldsymbol{F}(z, \mathbf{0}) \not\equiv 0$ puisqu'au moins une règle possède une feuille (autrement les classes solutions seraient vide), et le système n'est pas linéaire à cause de $(\mathbf{H}_4)$. Enfin, les coefficients de Taylor $L_{i,n} = [z^n]L_i(z)$ des solutions du système sont non nuls à partir d'un certain rang par $(\mathbf{H}_2)$, donc les $L_i(z)$ sont apériodiques.

Nous pouvons alors appliquer le théorème de Drmota (Théorème 1.89) : toutes les séries solutions $L_j(z)$ ont une même unique singularité dominante ρ, qui est leur rayon de convergence commun, avec $0 < \rho \leqslant 1$. Par ailleurs $L_j(\rho) = \tau_j < \infty$. De plus, comme $\boldsymbol{F}$ est un polynôme, $(\rho, \boldsymbol{\tau})$ est un point caractéristique qui se situe dans le rayon de convergence de $\boldsymbol{F}$, et ainsi $\boldsymbol{\tau} = \rho\,\boldsymbol{\phi}(\rho; \boldsymbol{\tau})$ et $0 = \det(\mathtt{Id}_{m\times m} - \rho\,\mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{\phi}](\rho; \boldsymbol{\tau}))$.

De plus pour tout j, il existe deux fonctions $g_j(z)$ et $h_j(z)$ analytiques en $z = \rho$ et qui satisfont $L_j(z) = g_j(z) - h_j(z)\sqrt{1 - z/\rho}$ dans un voisinage de $z = \rho$, avec $z \notin [\rho, +\infty)$ et $h_j(\rho) \neq 0$. Enfin, la fin du théorème de Drmota (qui est juste une application du théorème de Transfert (cf. Théorème 1.81 des préliminaires)) donne la forme des équivalents asymptotiques. ■

Corollaire 2.34 .
▶ Il existe une constante $D > 0$ telle que $[z^n]L(z) \geqslant Dn^{-3/2}\rho^{-n}$ pour n suffisamment grand. ◀

Démonstration. Nous rappelons que $\mathcal{L} = \cup_{i\in I}\mathcal{L}_i$. Si le système décrit de façon non ambiguë les arbres des classes $\mathcal{L}_i$, nous n'avons pas imposé que l'union soit disjointe, donc cette égalité nous donne seulement l'inégalité $[z^n]L(z) \leqslant [z^n]\sum_{i\in I} L_i(z)$ (donc le mauvais sens). Comme chaque classe est décrite de façon non ambiguë, un arbre de $T \in \mathcal{L}$ apparaît dans l'union en maximum $|I| \leqslant m$ exemplaires. Ainsi $[z^n]\sum_{i\in I} L_i(z) \leqslant m[z^n]L(z)$.

Nous avons donc $[z^n]L(z) \geqslant m^{-1}[z^n]\sum_{i\in I} L_i(z) \sim D'n^{3/2}\rho^{-n}$ par la Proposition 2.33, avec $D' = \frac{1}{m}\sum_{i\in I} C_i > 0$. En choisissant $\varepsilon > 0$ de façon à ce que $D := D' - \varepsilon$ soit strictement positif et légèrement plus petit que D', nous pouvons affirmer qu'à partir d'un certain rang $[z^n]L(z) \geqslant Dn^{-3/2}\rho^{-n}$. ■

2.4.2 Analyse du numérateur

Dans cette section, nous nous contentons d'établir une borne supérieure pour le numérateur, plutôt qu'un équivalent asymptotique plus précis : nous allons montrer que pour tout $j \in [m]$, $[z^n]\partial_u L_j(z, u)|_{u=1} \leqslant \alpha\rho^{-n}n^{-3/2}$, pour une certaine

constante $\alpha > 0$. Cela suffit pour prouver le théorème principal. Nous attaquons le problème en deux étapes, incarnées par les Propositions 2.35 et 2.36 ci-dessous. Les preuves techniques viendront plus tard, en Section 2.5 ; nous donnons néanmoins ici un aperçu des idées principales du cheminement de preuve.

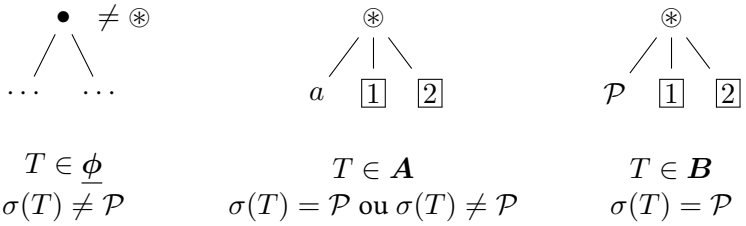


Figure 2.5. Les trois situations typiques observées pour spécifier $\boldsymbol{L}(z, u)$

Dans un premier temps, nous divisons le système $\boldsymbol{\phi}$ en trois morceaux distincts $\boldsymbol{\phi} = \underline{\boldsymbol{\phi}} + \boldsymbol{A} + \boldsymbol{B}$ où : $\underline{\boldsymbol{\phi}}$ correspond aux règles de $\boldsymbol{\phi}$ dont la racine n'est pas $\circledast$ et $\boldsymbol{B}$ regroupe les règles de racine $\circledast$ qui possèdent un fils constant complètement réductible. Par la Proposition 2.31, nous pouvons assurer que $\boldsymbol{B}$ n'est pas une fonction constante de $\boldsymbol{y}$. Cette division en trois groupes nous guide alors vers une définition récursive de $\boldsymbol{L}(z, u)$ qui dépend de deux classes clefs : celle des expressions entièrement réductibles, et son complémentaire. En dérivant cette équation, nous faisons apparaître les matrices jacobiennes de $\underline{\boldsymbol{\phi}}$ et $\boldsymbol{A}$. Nous bornons alors les coefficients des séries génératrices des différentes classes par ceux de la série $\boldsymbol{L}(z)$, dont le comportement est bien connu grâce à la Proposition 2.33. Nous obtenons alors la borne suivante sur les coefficients de Taylor des coordonnées du vecteur $\partial_u \boldsymbol{L}(z, u)$:

Proposition 2.35.
▶ Pour une certaine constante $C > 0$, les bornes suivantes sont vérifiées coordonnée par coordonnée :

$$[z^n]\Big\{\partial_u \boldsymbol{L}(z,u)\big|_{u=1}\Big\} \leqslant C \cdot [z^n]\Big\{\big(\mathtt{Id}_{m\times m} - z \cdot \mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}} + \boldsymbol{A}](z; \boldsymbol{L}(z))\big)^{-1} \cdot \boldsymbol{L}(z)\Big\}.$$

où nous utilisons la notation $(\mathtt{Id}_{m\times m} - zJ(z))^{-1}$, avec $J(z)$ une matrice $m \times m$ de séries formelles, pour désigner la matrice de séries formelles définie par $\sum_n z^n J(z)^n$. ◀

Une fois cette proposition démontrée, nous passons à l'analyse du second terme de l'inégalité de la Proposition 2.35, dont les singularités dominantes sont, malgré les apparences, plus simples à étudier que $\partial_u \boldsymbol{L}|_{u=1}$. Nous étudions pour cela le spectre de la matrice $J(z) = \mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}} + \boldsymbol{A}](z; \boldsymbol{L}(z))$, pour montrer que la matrice $\big(\mathtt{Id}_{m\times m} - z \cdot J(z)\big)$ est inversible en $z = \rho$, et que ses coefficients ont au plus une singularité dominante en ρ sur le cercle $|z| = \rho$, ainsi qu'un comportement en racine carrée. Ceci est résumé dans la proposition suivante :

Proposition 2.36.
▶ Les coordonnées de $\boldsymbol{K} : z \mapsto (\mathtt{Id} - z \cdot J(z))^{-1} \cdot \boldsymbol{L}(z)$ ont au plus une singularité dominante en $z = \rho = \rho_{\boldsymbol{L}}$ sur le cercle $|z| = \rho$. De plus, pour tout indice j, il existe des fonctions $\tilde{g}_j, \tilde{h}_j$ analytiques en $z = \rho$ telles que $K_j(z) = \tilde{g}_j(z) - \tilde{h}_j(z)\sqrt{1 - z/\rho}$ avec $\tilde{h}_j(\rho) \neq 0$. Enfin, nous avons les équivalents asymptotiques $[z^n]K_j(z) \sim D_j \rho^{-n} n^{-3/2}$, pour des constantes $D_j > 0$. ◀

2.4.3 Preuve du théorème principal

En supposant les deux propositions 2.35 et 2.36 démontrées, nous pouvons les appliquer à l'Eq. (2.4). Nous obtenons alors la borne

$$[z^n]\partial_u L_i(z,u)|_{u=1} = O\left(\frac{\rho^{-n}}{n^{3/2}}\right),$$

pour tout $i \in [m]$. Or, nous avons les inégalités suivantes :

$$[z^n]\partial_u L(z,u)|_{u=1} \leqslant \sum_{i\in I}[z^n]\partial_u L_i(z,u)|_{u=1} = O\left(\frac{\rho^{-n}}{n^{3/2}}\right)$$

car $\mathcal{L} = \cup_{i\in I}\mathcal{L}_i$. Ainsi, par le Corollaire 2.34, il existe une constante $C > 0$ telle que pour n assez grand on ait :

$$\mathbb{E}_n[|\sigma|] = \frac{[z^n]\partial_u L(z,u)\big|_{u=1}}{[z^n]L(z)} \leqslant C\,. \tag{2.5}$$

Ceci conclut la preuve du Théorème 2.32.

2.5 Preuve des propositions techniques

2.5.1 Preuve de la Proposition 2.35

Dans cette section, nous démontrons la proposition suivante :

Proposition 2.35 .
▶ Pour une certaine constante $C > 0$, les bornes suivantes sont vérifiées coordonnée par coordonnée :

$$[z^n]\Big\{\partial_u \boldsymbol{L}(z,u)\big|_{u=1}\Big\} \leqslant C\cdot[z^n]\Big\{\big(\mathtt{Id}_{m\times m} - z\cdot \mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}} + \boldsymbol{A}](z;\boldsymbol{L}(z))\big)^{-1}\cdot \boldsymbol{L}(z)\Big\}\,.$$

où nous utilisons la notation $(\mathtt{Id}_{m\times m} - zJ(z))^{-1}$, avec $J(z)$ une matrice $m\times m$ de séries formelles, pour désigner la matrice de séries formelles définie par $\sum_n z^n J(z)^n$. ◀

Partie symbolique : séries génératrices bivariées

Pour mieux comprendre le processus de réduction, et arriver à en donner une spécification combinatoire, nous avons besoin d'identifier plusieurs sous-classes d'arbres pertinentes. En effet, la description $\mathcal{E}$ des arbres qui nous est donnée n'est pas adaptée pour suivre le processus de simplification, nous devons décomposer les différentes règles en plusieurs groupes pour pouvoir marquer les nœuds qui restent après réduction.

Rappelons notre cadre de travail : par la Proposition 2.31, nous étudions un ensemble d'arbres $\mathcal{L}_1,\dots,\mathcal{L}_m$ décrit par un système $\mathcal{E} = \{E_1,\dots,E_m\}$ *propre*, dans le sens où toutes les feuilles $\square$ des règles de $\mathcal{E}$ sont situées directement sous la racine, à profondeur exactement 1. De plus ce système vérifie les hypothèse $(\mathbf{H})$, ainsi que l'hypothèse supplémentaire $(\mathbf{H}'_3)$ suivante : il existe une règle d'arité non nulle, de racine $\circledast$, et dont un fils est un arbre complet entièrement réductible.

Pour marquer avec une variable u la taille des réductions des arbres, nous voulons identifier les nœuds des règles qui subsistent après la réduction. Pour ce faire, nous divisons chaque classe $\mathcal{L}_j$ en une somme disjointe de deux classes $\mathcal{R}_j := \{T \in \mathcal{L}_j : \sigma(T) = \mathcal{P}\}$ et $\mathcal{G}_j := \mathcal{L}_j \setminus \mathcal{R}_j$, où $\mathcal{R}_j$ rassemble les arbres de $T \in \mathcal{L}_j$ qui se réduisent entièrement à $\mathcal{P}$.

Nous introduisons alors les séries bivariées associées à ces classes, qui comptent la taille de la réduction en u. Par définition, pour tout $i \in [m]$:

$$\begin{aligned} L_i(z,u) &:= \sum_{L\in\mathcal{L}_i} z^{|L|}u^{|\sigma(L)|} = G_i(z,u) + R_i(z,u), \\ G_i(z,u) &:= \sum_{G\in\mathcal{G}_i} z^{|G|}u^{|\sigma(G)|}, \\ R_i(z,u) &:= \sum_{R\in\mathcal{R}_i} z^{|R|}u^{p} := u^p R_i(z)\,, \end{aligned}$$

où nous rappelons que p désigne la taille de $\mathcal{P}$. Nous écrirons $\boldsymbol{L}(z,u)$, $\boldsymbol{R}(z,u)$, et $\boldsymbol{G}(z,u)$ pour désigner les vecteurs correspondants de séries génératrices.

L'égalité $R_i(z,u) = u^p R_i(z)$ permet de séparer les variables z et u, si bien que la dérivation de $R_i(z,u)$ par rapport à u est aisée, même si nous ne connaissons pas $R_i(z)$.

Intéressons-nous donc plutôt à $G_i(z,u)$, pour $i \in [m]$. Nous cherchons à transformer la spécification de $\mathcal{L}_i$ en une spécification pour $\mathcal{G}_i$, qui permette de marquer avec la variable u les nœuds qui restent après la réduction : le but étant d'obtenir une équation sur les séries génératrices bivariées qui soit exploitable pour calculer les dérivées $\partial_u G_i(z,u)$. La forme propre du système $\mathcal{E}$ permet justement de spécifier exactement les arbres qui ne sont pas entièrement réductibles.

Exemple 2.37 .

▶ Supposons que E_1 est donné par :

$$E_1 = \left\{ \begin{array}{c} \star \\ | \\ \boxed{2} \end{array} , \quad \begin{array}{ccc} & \circledast & \\ / & | & \backslash \\ a & \boxed{1} & \star \\ & & | \\ & & a \end{array} , \quad \begin{array}{ccc} & \circledast & \\ / & | & \backslash \\ \mathcal{P} & \boxed{1} & \star \\ & & | \\ & & a \end{array} \right\}$$

avec $|\mathcal{P}| = p \geqslant 3$ (donc $\mathcal{P} \notin \{a, \overset{\star}{|}\,\}$), et que de plus l'arbre constant $\mathcal{P}$ n'est pas constructible à partir de E_1 ($\mathcal{P} \notin \mathcal{L}_1$). Alors :

— Aucun arbre de $\mathcal{L}_1$ construit à partir de la règle $\overset{\star}{|}\,\boxed{2}$ n'est entièrement réductible, peu importe l'arbre de $\mathcal{L}_2$ substitué à la feuille $\boxed{2}$. La racine $\star$ reste après la réduction et peut donc être marquée par u.

— Aucun arbre construit à l'aide de la règle $\begin{array}{ccc} & \circledast & \\ / & | & \backslash \\ a & \boxed{1} & \star \\ & & | \\ & & a \end{array}$ n'est entièrement réductible si on remplace la feuille $\boxed{1}$ par un arbre de $\mathcal{G}_1$. Dans ce cas, la racine peut être marquée par u car elle reste après réduction, ainsi que les nœuds des sous-arbres complets a et a^*.

— Enfin, tous les arbres engendrés par la dernière règle sont forcément entièrement réductibles à cause du fils complet entièrement réductible sous la racine $\circledast$. Il ne faut pas prendre en compte cette règle dans la spécification de $\mathcal{G}_1$.

Nous avons donc $G_1(z,u) = zuL_2(z,u) + zu(zu)^3 G_1(z,u)$. ◀

Remarque 2.38.
▶ Dans l'exemple précédent, si $\mathcal{P}$ était constructible à partir de E_1, alors nous aurions plutôt

$$G_1(z,u) = -(zu)^p + zuL_2(z,u) + zu(zu)^3 G_1(z,u),$$

car $\mathcal{P} \notin \mathcal{G}_1$ mais est comptabilisé dans $zuL_2(z,u) + zu(zu)^3 G_1(z,u)$. Il faut le retirer de l'équation pour avoir exactement $G_1(z,u)$. ◀

Pour $i \in [m]$, un arbre $T \in \mathcal{L}_i$ n'est ainsi pas entièrement réductible si et seulement si il n'est pas égal à $\mathcal{P}$ et vérifie l'une des conditions suivantes :

— ou bien il est construit à partir d'une règle de E_i dont la racine n'est pas $\circledast$

Rappelons que comme le système est propre, les feuilles $\square$ sont directement sous la racine.

— ou bien il est de la forme $T = T_i[t_1, \dots, t_a]$, avec T_i une règle de E_i d'arité $a \geqslant 0$, de racine $\circledast$, dont aucun sous-arbre complet sous la racine n'est entièrement réductible, et aucun des arbres $t_1, \dots, t_a$ placés au niveau des feuilles $\square$ de T_i ne sont entièrement réductibles.

Nous notons ainsi pour tout $i \in [m]$:

— $\underline{E_i}$ l'ensemble des règles de E_i dont la racine n'est pas $\circledast$;

— $\mathcal{A}_i$ l'ensemble des règles de $T \in E_i$ de racine $\circledast$, dont aucun fils complet n'est entièrement réductible.

Remarquons que par l'hypothèse $(\mathbf{H}'_3)$, il existe au moins un indice j tel que $\underline{E_j} \cup \mathcal{A}_j \subsetneq E_j$. Nous définissons aussi le polynôme associé suivant :

$$A_i(z,u;y_1,\dots,y_m) = \sum_{T\in\mathcal{A}_i} z^{|T|-1} u^{|\sigma(T)|-1} \prod_{j\in \mathtt{ind}(T)} y_j\,,$$

Nous rappelons que nous avons décidé de ne pas compter les feuilles $\square$ dans la taille d'un arbre incomplet.

où nous avons étendu la réduction σ aux arbres incomplets en posant $\sigma(\square) = \square$. Il s'agit bien d'un polynôme en $z, u, y_1, \dots, y_m$ car il y a un nombre fini de règles. De façon similaire nous définissons le polynôme

$$\underline{\phi_i}(z,u;y_1,\dots,y_m) = \sum_{T\in\underline{E_i}} z^{|T|-1} u^{|\sigma(T)|-1} \prod_{j\in \mathtt{ind}(T)} y_j\,.$$

Nous définissons enfin les vecteurs de polynômes correspondants $\boldsymbol{A}$ et $\underline{\boldsymbol{\phi}}$. La discussion de ce paragraphe se traduit dans le monde des séries par l'égalité suivante, vérifiée pour tout $i \in [m]$:

$$G_i(z,u) = -a_i(zu)^p + zu\underline{\phi_i}(z,u;\boldsymbol{L}(z,u)) + zuA_i(z,u;\boldsymbol{G}(z,u))$$

où $a_i = 1$ si $\mathcal{P} \in \mathcal{L}_i$, et $a_i = 0$ sinon. Ainsi, nous avons démontré la récurrence suivante satisfaite par $\boldsymbol{L}(z,u)$:

Lemme 2.39.
▶ Le vecteur $\boldsymbol{L}(z,u)$ satisfait l'équation :

$$\boldsymbol{L}(z,u) = u^p\left(\boldsymbol{R}(z) - \boldsymbol{P}(z)\right) + zu\left(\underline{\boldsymbol{\phi}}(z,u;\boldsymbol{L}(z,u)) + \boldsymbol{A}(z,u;\ \boldsymbol{G}(z,u))\right)\,,$$

où $\boldsymbol{P}(z)$ est un vecteur de la forme $(a_1 z^p, \dots, a_m z^p)$, avec $a_i \in \{0,1\}$. ◀

Inégalités coefficient par coefficient des séries formelles

Dans les preuves qui suivent, nous allons introduire des inégalités sur les coefficients de la série $\partial_u \boldsymbol{L}(z,u)\big|_{u=1}$. Nous introduisons dans un premier temps la notation $\preceq$ pour manier plus facilement de telles inégalités.

Définition 2.40.
▶ Soient deux séries $F(z) = \sum_{n\in\mathbb{N}} f_n z^n$ et $G(z) = \sum_{n\in\mathbb{N}} g_n z^n$ à coefficients réels. On écrit $F(z) \preceq G(z)$ si pour tout $n \geqslant 0$, nous avons l'inégalité $f_n \leqslant g_n$.
Nous étendons la notation aux matrices de séries, composante par composante. Soient $\boldsymbol{M}(z) = (M_{i,j}(z))_{i\in[m],j\in[n]}$, $\boldsymbol{N}(z) = (N_{i,j}(z))_{i\in[m],j\in[n]}$ deux matrices de séries de dimension $m \times n$, nous notons $\boldsymbol{M} \preceq \boldsymbol{N}$ si pour chaque $i, j \geqslant 0$, $M_{i,j}(z) \preceq N_{i,j}(z)$. ◀

Nous résumons les propriétés de la relation $\preceq$ dans le lemme suivant.

Lemme 2.41 (Boîte à outils pour $\preceq$).
▶ La relation $\preceq$ vérifie les propriétés suivantes :
— Soient $0 \preceq F(z), G(z), H(z), J(z)$ quatre séries à coefficients positifs, telles que $F(z) \preceq G(z)$ et $H(z) \preceq J(z)$. Alors $F(z) + H(z) \preceq G(z) + J(z)$ et $F(z)H(z) \preceq G(z)J(z)$ (autrement dit $\preceq$ est compatible avec l'addition et la multiplication de séries à termes positifs).
— Soient $p(z, y_1, \dots, y_n)$ un polynôme en $z, y_1, \dots, y_n$ à coefficients positifs, et des séries $0 \preceq F_i(z) \preceq G_i(z)$ pour $i \in [n]$. Alors $p(z, F_1(z), \dots, F_n(z)) \preceq p(z, G_1(z), \dots, G_n(z))$.
— Soient $\mathbf{0}_{m\times n} \preceq \boldsymbol{M}(z) \preceq \boldsymbol{N}(z)$, et $\mathbf{0}_{n\times 1} \preceq \boldsymbol{F}(z) \preceq \boldsymbol{G}(z)$, alors $\mathbf{0}_{m\times 1} \preceq \boldsymbol{M}(z)\boldsymbol{F}(z) \preceq \boldsymbol{N}(z)\boldsymbol{G}(z)$.

◀

Lemme 2.42.
▶ Il existe une constante $C > 0$ telle que

$$\partial_u \boldsymbol{L}\big|_{u=1} \preceq C \cdot \boldsymbol{L}(z) + z\, \mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}} + \boldsymbol{A}](z; \boldsymbol{L}(z)) \cdot \partial_u \boldsymbol{L}\big|_{u=1}\,.$$

◀

Démonstration. Dérivons par rapport à u l'équation du Lemme 2.39 sur $\boldsymbol{L}(z,u)$:

$$\begin{aligned}\partial_u \boldsymbol{L}\big|_{u=1} = {} & p\left(\boldsymbol{R}(z) - \boldsymbol{P}(z)\right) + z\left(\underline{\boldsymbol{\phi}}(z,1;\boldsymbol{L}(z)) + \boldsymbol{A}(z,1;\ \boldsymbol{G}(z))\right)\\ & + z\left(\partial_u \underline{\boldsymbol{\phi}}(z,1;\boldsymbol{L}(z)) + \partial_u \boldsymbol{A}(z,1;\ \boldsymbol{G}(z))\right)\\ & + z\left(\mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}}](z,1;\boldsymbol{L}(z)) \cdot \partial_u \boldsymbol{L}\big|_{u=1} + \mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{A}](z,1;\boldsymbol{G}(z)) \cdot \partial_u \boldsymbol{G}\big|_{u=1}\right)\,.\end{aligned}$$

Nous observons alors que :
— Par un argument combinatoire direct, $\boldsymbol{R}(z) - \boldsymbol{P}(z) \preceq \boldsymbol{L}(z)$
— Comme les composantes de $\boldsymbol{A}$ sont des polynômes à coefficients positifs, et $\boldsymbol{G}(z) \preceq \boldsymbol{L}(z)$, alors $\boldsymbol{A}(z,1;\boldsymbol{G}(z)) \preceq \boldsymbol{A}(z,1;\boldsymbol{L}(z))$ et ainsi :

$$\begin{aligned}& z\big(\underline{\boldsymbol{\phi}}(z,1;\boldsymbol{L}(z)) + \boldsymbol{A}(z,1;\ \boldsymbol{G}(z))\big)\\ \preceq\ & z\big(\underline{\boldsymbol{\phi}}(z,1;\boldsymbol{L}(z)) + \boldsymbol{A}(z,1;\ \boldsymbol{L}(z)) + \boldsymbol{B}(z,1;\ \boldsymbol{L}(z))\big) = \boldsymbol{L}(z)\,.\end{aligned}$$

De même, $\partial_u \boldsymbol{A}(z,1;\boldsymbol{G}(z)) \preceq \partial_u \boldsymbol{A}(z,1;\boldsymbol{L}(z))\,.$

— Pour la même raison, $\mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{A}](z,1;\boldsymbol{G}(z)) \preceq \mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{A}](z,1;\boldsymbol{L}(z))$

— Finalement, nous vérifions facilement à la main que $\partial_u \boldsymbol{G}|_{u=1} \preceq \partial_u \boldsymbol{L}|_{u=1}$.

En reportant ces trois inégalités dans l'équation sur $\partial_u \boldsymbol{L}|_{u=1}$, nous obtenons

$$\partial_u \boldsymbol{L}\big|_{u=1} \preceq (p+1)\boldsymbol{L}(z) + z\partial_u[\underline{\boldsymbol{\phi}} + \boldsymbol{A}](z,1;\boldsymbol{L}(z)) + z\,\mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}} + \boldsymbol{A}](z,1;\boldsymbol{L}(z)) \cdot \partial_u \boldsymbol{L}\big|_{u=1}\,.$$

Rappelons que $\underline{\boldsymbol{\phi}} + \boldsymbol{A}$ est un vecteur de polynômes en u (et aussi en z et en $\boldsymbol{y}$). Soit d le plus grand degré en u de ces polynômes. Par définition, $d+1$ (car il manque la racine) correspond à la plus grande taille possible obtenue en réduisant par σ les règles incomplètes. Alors $z\partial_u[\underline{\boldsymbol{\phi}} + \boldsymbol{A}](z,1;\boldsymbol{L}(z)) \preceq d \cdot z[\underline{\boldsymbol{\phi}} + \boldsymbol{A}](z,1;\boldsymbol{L}(z))$, et par ailleurs ce second membre est inférieur à $d \cdot \boldsymbol{L}(z)$, comme conséquence de l'Eq. (2.3). Ainsi

$$\partial_u \boldsymbol{L}\big|_{u=1} \preceq (p+1+d) \cdot \boldsymbol{L}(z) + z\,\mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}} + \boldsymbol{A}](z,1;\boldsymbol{L}(z)) \cdot \partial_u \boldsymbol{L}\big|_{u=1}\,,$$

ce qui prouve le résultat en posant $C = p+1+d$. ■

Nous souhaitons ensuite isoler le terme $\partial_u \boldsymbol{L}|_{u=1}$ qui apparaît à droite de l'inégalité. Nous ne pouvons pas soustraire simplement, car alors nous nous retrouvons avec des séries à coefficients quelconques (possiblement négatifs), et notre boîte à outils ne s'applique plus. Nous nous appuyons à la place sur le lemme suivant :

Lemme 2.43 .
▶ Soient $\mathbf{0}_{m\times 1} \preceq \boldsymbol{v}(z), \boldsymbol{b}(z), \mathbf{0}_{m\times m} \preceq \boldsymbol{M}(z)$, et $\boldsymbol{v}(z) \preceq \boldsymbol{b}(z) + z\boldsymbol{M}(z)\boldsymbol{v}(z)$ alors nous avons l'inégalité suivante

$$\boldsymbol{v}(z) \preceq (\mathtt{Id} - z\boldsymbol{M}(z))^{-1} \cdot \boldsymbol{b}(z)\,,$$

où $(\mathtt{Id} - z\boldsymbol{M}(z))^{-1} = \sum_{k\geqslant 0} z^k (\boldsymbol{M}(z))^k$. ◀

Remarque 2.44 .
▶ La série $\sum z^k(\boldsymbol{M}(z))^k$ converge dans l'anneau des séries formelles $\mathbb{Q}[[z]]$, car les coefficients d'ordre n restent fixés après avoir sommé les $n+1$ premiers termes de la série. ◀

Démonstration. En appliquant l'inégalité k fois sur le membre droit on obtient

$$\boldsymbol{v}(z) \preceq \boldsymbol{b}(z) + z\boldsymbol{M}(z)\boldsymbol{b}(z) + \ldots + (z\boldsymbol{M}(z))^k \boldsymbol{b}(z) + (z\boldsymbol{M}(z))^{k+1}\boldsymbol{v}(z)\,,$$

Comme $[z^k]\left(z\boldsymbol{M}(z))^{k+1}\boldsymbol{v}(z)\right) = \mathbf{0}_{m\times 1}$, on déduit que

$$\begin{aligned}[z^k]\boldsymbol{v}(z) &\leqslant [z^k]\left(\boldsymbol{b}(z) + z\boldsymbol{M}(z)\boldsymbol{b}(z) + \ldots + (z\boldsymbol{M}(z))^k\boldsymbol{b}(z)\right)\\ &= [z^k]\,(\mathtt{Id} - z\boldsymbol{M}(z))^{-1}\,\boldsymbol{b}(z),\end{aligned}$$

pour tout k, et donc par définition $\boldsymbol{v}(z) \preceq (\mathtt{Id} - z\boldsymbol{M}(z))^{-1} \cdot \boldsymbol{b}(z)$. ■

Les Lemmes 2.42 et 2.43 prouvent alors la Proposition 2.35 en établissant la borne :

$$[z^n]\left\{\partial_u \boldsymbol{L}(z,u)\big|_{u=1}\right\} \leqslant C \cdot [z^n]\left\{\left(\mathtt{Id}_{m\times m} - z\cdot \mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}} + \boldsymbol{A}](z;\boldsymbol{L}(z))\right)^{-1} \cdot \boldsymbol{L}(z)\right\}.$$

2.5.2 Compléments techniques sur le théorème de Drmota

Les preuves des propositions de cette partie technique peuvent être passées en première lecture.

Avant de pouvoir prouver la Proposition 2.36, nous avons besoin de résultats intermédiaires pour manipuler les solutions des systèmes qui entrent dans le cadre du théorème de Drmota. Les propositions qui suivent étaient un peu trop arides pour être exposées dans les Préliminaires, donc nous avons choisi de les reporter ici.

Les résultats de cette sous-section ne sont pas spécifiques à ce chapitre, et seront aussi très utiles pour les chapitres 3 et 4. Nous avons donc essayé de rester le plus général possible.

Le théorème de Drmota (voir Théorème 1.89) nous dit que sous certaines conditions, les solutions d'un système bien conditionné ont un comportement en racine carrée autour de leur singularité dominante commune. Ce comportement universel s'explique par la grande stabilité des fonctions de cette forme sous de nombreuses opérations, que nous exposons dans cette section.

Stabilité des fonctions en racine carrée pour les opérations usuelles

Le lemme suivant énonce les premières propriétés remarquables de stabilité des séries qui ont un comportement en racine carrée autour de leur singularité dominante $z = \rho$.

Lemme 2.45 .
▶ Soient $A(z)$ et $B(z)$ deux séries qui ont un développement asymptotique autour de $z = \rho$ de la forme :

$$A(z) = g_A(z) - h_A(z)\sqrt{1 - \tfrac{z}{\rho}}, \quad B(z) = g_B(z) - h_B(z)\sqrt{1 - \tfrac{z}{\rho}},$$

où $g_A(z), g_B(z), h_A(z), h_B(z)$ sont analytiques autour de $z = \rho$. Alors :

- **a.** La somme $A(z)+B(z)$ et la différence $A(z)-B(z)$ ont aussi un développement asymptotique $A(z)+B(z) = g_{A+B}(z)-h_{A+B}(z)\sqrt{1-z/\rho}$ et $A(z)-B(z) = g_{A-B}(z) - h_{A-B}(z)\sqrt{1-z/\rho}$, autour de $z = \rho$, avec $g_{A+B}(z)$, $h_{A+B}(z)$, $g_{A-B}(z)$, $h_{A-B}(z)$ analytiques en $z = \rho$.
- **b.** Le produit $A(z) \cdot B(z)$ a aussi un développement asymptotique $g_{A\cdot B}(z) - h_{A\cdot B}(z)\sqrt{1-z/\rho}$, autour de $z = \rho$, avec $g_{A\cdot B}(z)$ et $h_{A\cdot B}(z)$ analytiques at $z = \rho$.
- **c.** Soit $H(y)$ analytique en $y = A(\rho)$ et tel que $H'(A(\rho)) \neq 0$. Alors la composée de H et A a aussi un développement asymptotique $H(A(z)) = g_{H\circ A}(z) - h_{H\circ A}(z)\sqrt{1-z/\rho}$ autour de $z = \rho$, avec $g_{H\circ A}$ et $h_{H\circ A}$ analytiques en $z = \rho$.
- **d.** Si $B(\rho) \neq 0$, le quotient $A(z)/B(z)$ a aussi un développement asymptotique $g_{A/B}(z) - h_{A/B}(z)\sqrt{1-z/\rho}$ autour de $z = \rho$, avec $g_{A/B}(z)$ et $h_{A/B}(z)$ analytiques en $z = \rho$.

◀

Démonstration. Nous ne développons que le dernier cas (la division), car la somme, la différence et le produit sont immédiats, et (3) est un cas particulier d'un lemme de Drmota [Drm09, Lemma 2.26].

Comme $g_B(\rho) \neq 0$, nous pouvons écrire

$$\frac{A(z)}{B(z)} = \frac{g_A(z) - h_A(z)\sqrt{1-\frac{z}{\rho}}}{g_B(z) - h_B(z)\sqrt{1-\frac{z}{\rho}}} = \frac{g_A(z)/g_B(\rho) - h_A(z)/g_B(\rho)\sqrt{1-\frac{z}{\rho}}}{1 + (g_B(z)/g_B(\rho) - 1) - h_B(z)/g_B(\rho)\sqrt{1-\frac{z}{\rho}}}.$$

La fonction $H(y) = \frac{1}{1+y} = 1 - y + y^2 - y^3 \pm \dots$ est analytique sur $|y| < 1$. Puisque $\delta\colon z \mapsto (g_B(z)/g_B(\rho) - 1) - h_B(z)/g_B(\rho)\sqrt{1-\frac{z}{\rho}}$ vaut 0 en $z = \rho$, nous pouvons appliquer (3) à $H(\delta(z))$ pour développer le dénominateur. Nous concluons en utilisant la stabilité par produit. ■

Inversion d'une matrice dont les coefficients ont un comportement en racine carrée

La Proposition 2.35 borne $\partial_u \boldsymbol{L}(z,u)\big|_{u=1}$ à l'aide de l'inverse formelle de la matrice $J(z) = \mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}} + \boldsymbol{A}](z; \boldsymbol{L}(z))$. Dans le chapitre 4, nous rencontrerons aussi l'inversion d'une matrice obtenue à partir d'un système suivant les conditions du théorème de Drmota. Nous avons ainsi besoin de comprendre à quoi ressemblent les coefficients de telles matrices, et pour les inverser, nous devons vérifier que cette opération est d'abord possible, et qu'elle n'introduit pas de nouvelles singularités. Pour cela, il nous faut entrer un peu plus dans les détails des outils classiques utilisés dans l'étude des systèmes reliés au théorème de Drmota.

Nous commençons par rappeler des propriétés classiques concernant le rayon spectral d'une matrice.

Définition 2.46 .
▶ Soit $A \in \mathcal{M}_n(\mathbb{C})$ une matrice carrée de taille n. Le *rayon spectral* de A, noté $\mathtt{Sp}(A)$, désigne le plus grand module de ses valeurs propres. ◀

Définition 2.47 .
Nous gardons la notation $\|A\|_\infty$ pour $\max_{1\leqslant i,j\leqslant n} |a_{i,j}|$.
▶ Pour $A = (a_{i,j}) \in \mathcal{M}_n(\mathbb{C})$, nous définissons

$$|A|_\infty := \max_{1\leqslant i\leqslant n} \sum_{j=1}^{n} |a_{i,j}|\,.$$

Il s'agit d'une norme sur $\mathcal{M}_n(\mathbb{C})$. ◀

Proposition 2.48 (Formule de Gelfand, voir par exemple [Rom21]).
▶ Pour toute matrice $A \in \mathcal{M}_n(\mathbb{C})$:

$$\mathtt{Sp}(A) = \lim_{k\to\infty} |A^k|_\infty^{1/k}$$

◀

Définition 2.49 .
▶ Soit $A, B \in \mathcal{M}_n(\mathbb{R})$. La matrice A est dite positive si tous les coefficients de A sont positifs. Dans ce cas, nous notons $0 \leqslant A$. Enfin, nous notons $A \leqslant B$ si $0 \leqslant B - A$. ◀

Proposition 2.50 (Croissance du rayon spectral, voir par exemple [Rom21]).
▶ Soit $A \in \mathcal{M}_n(\mathbb{C})$, et $B, C \in \mathcal{M}_n(\mathbb{R})$ telles que $B \leqslant C$. Alors :

a. $\mathtt{Sp}(A) \leqslant \mathtt{Sp}(|A|)$ en notant $|A| = (|a_{i,j}|)$;

b. $\mathtt{Sp}(B) \leqslant \mathtt{Sp}(C)$.

◀

Proposition 2.51 (Continuité, voir par exemple [Rom21, Théorème 3.16]).
▶ L'application $\mathtt{Sp} : \mathcal{M}_n(\mathbb{C}) \to \mathbb{R}$ est continue. ◀

Proposition 2.52 (Série géométrique, voir par exemple [Rom21, Théorème 3.17]).
▶ Soit $A \in \mathcal{M}_n(\mathbb{C})$. Alors :

$$(\mathtt{Id} - A) \text{ est inversible } \Leftrightarrow \mathtt{Sp}(A) < 1\,.$$

De plus si $\mathtt{Sp}(A) < 1$, alors $\sum_k A^k$ converge, et dans ce cas $\sum_{k=0}^{\infty} A^k = (\mathtt{Id} - A)^{-1}$. ◀

La proposition précédente donne ainsi une condition nécessaire et suffisante pour qu'une matrice à coefficients complexes soit inversible, et exprime dans ce cas l'inverse comme une série de matrice. Nous voulons faire de même, mais cette fois avec une matrice de séries formelles, qui ont un comportement en racine carrée autour d'une unique singularité dominante ρ. La proposition suivante est un peu technique, car elle vise à unifier les deux approches légèrement différentes utilisées dans ce chapitre et le chapitre 4. En effet, dans ce chapitre, les inversions seront des inversions formelles définies comme des séries de matrices, tandis que dans le chapitre 4 nous inverserons des matrices à coefficients dans $\mathbb{C}$. Nous vérifions que ces deux approches sont cohérentes :

Proposition 2.53 .
▶ Soit $M(z; \boldsymbol{y})$ une matrice carrée de taille n, dont les coefficients sont des polynômes formels en z et $\boldsymbol{y}$, à coefficients positifs.

Soit $\boldsymbol{y}(z)$ un vecteur de fonctions analytiques à coefficients de Taylor positifs, qui ont une unique singularité dominante en $z = \rho > 0$, et qui autour de $z = \rho$ ont une écriture de la forme :

$$\boldsymbol{y}(z) = \boldsymbol{g}(z) - \boldsymbol{h}(z)\sqrt{1 - z/\rho}$$

avec $\boldsymbol{g}(z)$ et $\boldsymbol{h}(z)$ des vecteurs de fonctions analytiques en $z = \rho$. Nous supposons que $\boldsymbol{y}(z)$ est continue sur $[0, \rho]$, et nous notons $\boldsymbol{\tau} := \boldsymbol{y}(\rho)$.

Supposons de plus que $\mathtt{Sp}(M(\rho, \boldsymbol{\tau})) < 1$. Alors :

a. $\mathtt{Id} - M(z, \boldsymbol{y}(z))$ est inversible pour tout $|z| \leqslant \rho$, et $(\mathtt{Id} - M(z, \boldsymbol{y}(z)))^{-1} = \sum_{k=0}^{\infty}(M(z, \boldsymbol{y}(z))^k$ pour tout $|z| \leqslant \rho$.

b. L'application $J := z \mapsto (\mathtt{Id} - M(z, \boldsymbol{y}(z))^{-1}$ est une matrice de fonctions analytiques en 0, continues sur $[0, \rho]$, qui ont au plus une unique singularité dominante sur le cercle $|z| = \rho$, en $z = \rho$, et sont localement en $z = \rho$ de la forme $g(z) - h(z)\sqrt{1 - z/\rho}$, avec $g(z)$ et $h(z)$ analytiques en $z = \rho$.

c. Les coefficients de la matrice de séries de fonctions $J_2 := z \mapsto \sum_{k=0}^{\infty}(M(z, \boldsymbol{y}(z))^k$ convergent normalement sur $|z| \leqslant \rho$, et coincident pour $|z| < \rho$ avec ceux de J.

◀

Démonstration. **a.** La matrice formelle $M(z;\boldsymbol{y})$ est inversible en se plaçant sur le corps des fractions rationnelles. En effet, comme $\mathtt{Sp}(M(\rho,\boldsymbol{\tau}))<1$, on sait que $\det(\mathtt{Id}-M(\rho,\boldsymbol{\tau}))\neq 0$, et donc $\det(\mathtt{Id}-M(z;\boldsymbol{y}))$ n'est pas le polynôme nul. Ainsi on note $J(z;\boldsymbol{y})=M(z;\boldsymbol{y})^{-1}$. On sait par la formule de la comatrice que $J(z;\boldsymbol{y})=\frac{P(z;\boldsymbol{y})}{\det(\mathtt{Id}-M(z;\boldsymbol{y}))}$ où $P(z;\boldsymbol{y})$ est une matrice de polynômes en z et $\boldsymbol{y}$. Ainsi, si $\det(\mathtt{Id}-M(z;\boldsymbol{y}(z)))\neq 0$, on a $J(z;\boldsymbol{y}(z))=M(z;\boldsymbol{y}(z))^{-1}$.
Soit $|z|\leqslant\rho$. Alors $|M(z;\boldsymbol{y}(z)|\leqslant M(\rho;\boldsymbol{\tau})$, d'où $\mathtt{Sp}(M(z,\boldsymbol{y}(z))<1$. Donc $\det(\mathtt{Id}-M(z;\boldsymbol{y}(z)))\neq 0$, et ainsi $(\mathtt{Id}-M(z,\boldsymbol{y}(z))$ est inversible, d'inverse $J(z;\boldsymbol{y}(z))$, par unicité de l'inverse sur $\mathcal{M}_n(\mathbb{C})$.
De plus, comme $\mathtt{Sp}(M(z,\boldsymbol{y}(z))<1, \sum_{k=0}^{\infty}(M(z,\boldsymbol{y}(z))^k=(\mathtt{Id}-M(z,\boldsymbol{y}(z)))^{-1}$.

b. On pose $J: z\mapsto J(z;\boldsymbol{y}(z))=\frac{P(z;\boldsymbol{y}(z))}{\det(\mathtt{Id}-M(z;\boldsymbol{y}(z)))}$.
Soit $1\leqslant i,j\leqslant n$, et on regarde la fonction $a(z)=(J(z))_{i,j}$. On sait que $a(z)$ est de la forme $\frac{p(z;\boldsymbol{y}(z))}{\det(\mathtt{Id}-M(z;\boldsymbol{y}(z)))}$, avec :
- $p(z;\boldsymbol{y}(z))$ qui est un polynôme en ses entrées, donc par stabilité, est une fonction analytique en 0, continue sur $[0,\rho]$, qui a au plus une unique singularité dominante sur le cercle $|z|=\rho$, en $z=\rho$, et est localement en $z=\rho$ de la forme $g(z)-h(z)\sqrt{1-z/\rho}$, avec $g(z)$ et $h(z)$ analytiques en $z=\rho$;
- $\det(\mathtt{Id}-M(z;\boldsymbol{y}(z))$ qui est aussi un polynôme en ses entrées et donc a la même propriété.

Comme $\det(\mathtt{Id}-M(\rho;\boldsymbol{y}(\rho))\neq 0$, par stabilité par quotient, $a(z)$ est une fonction analytique en 0, continue sur $[0,\rho]$, qui a au plus une unique singularité dominante sur le cercle $|z|=\rho$, en $z=\rho$, et est localement en $z=\rho$ de la forme $g(z)-h(z)\sqrt{1-z/\rho}$, avec $g(z)$ et $h(z)$ analytiques en $z=\rho$.

c. On fixe $1\leqslant i,j\leqslant n$. On note pour tout k :
- $a_k(z)$ le coefficient en position (i,j) dans la matrice $(M(z,\boldsymbol{y}(z))^k$
- v_k le coefficient en position (i,j) dans la matrice $(M(\rho,\tau))^k$
- $b(z)$ le coefficient en position (i,j) dans la matrice $J_2(z)$.

Par définition, $b(z)=\sum_{k=0}^{\infty}a_k(z)$, qui est une série de fonctions analytiques sur $|z|<\rho$ et continues sur $|z|\leqslant\rho$.
Par ailleurs, pour tout $|z|\leqslant\rho$, on a $|a_k(z)|\leqslant v_k$ car B est à coefficients positifs. Comme $\mathtt{Sp}(M(\rho,\boldsymbol{\tau}))<1$, la série de matrices $\sum_{k=0}^{\infty}(M(\rho,\boldsymbol{\tau}))^k$ converge, et donc la série $\sum_{k=0}^{\infty}v_k$ converge.
Il y a donc convergence normale de la série de fonctions $\sum a_k(z)$ sur $|z|\leqslant\rho$. Donc $b(z)$ est analytique sur $|z|<\rho$ et continue sur $|z|\leqslant\rho$.
Ainsi, $J_2(z)$ et $J(z)$ sont des matrices de fonctions analytiques sur $|z|<\rho$. On remarque qu'elles coincident sur $[0,\rho]$, elles sont donc égales.

■

La condition $\mathtt{Sp}(M(\rho,\boldsymbol{\tau}))<1$ dans la proposition précédente est fondamentale pour pouvoir effectuer l'inversion. Afin de l'appliquer sur des systèmes issus de l'application du théorème de Drmota, nous avons besoin d'outils pour calculer leur rayon spectral. Il se trouve que les systèmes auxquels nous appliquons le théorème de Drmota sont déjà directement reliés à une matrice de rayon spectral 1 :

Proposition 2.54 ([BBY10, Lemme 12]).
▶ Soit $\boldsymbol{y}=\boldsymbol{F}(z;\boldsymbol{y})$ un système polynomial bien conditionné. Soit $(\rho,\boldsymbol{\tau})$ défini

comme dans le Théorème 1.89, tel que $(\rho, \boldsymbol{\tau})$ est dans le domaine de convergence de $\boldsymbol{F}(z; \boldsymbol{y})$. Alors

$$\mathtt{Sp}(\mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{F}](\rho; \boldsymbol{\tau})) = 1\,.$$

En particulier la matrice $\mathtt{Id} - \mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{F}](\rho; \boldsymbol{\tau})$ n'est pas inversible. ◀

Nous introduisons maintenant le théorème de Perron-Frobenius, qui décrit entre autres quelques opérations qui font diminuer strictement le rayon spectral d'une matrice.

Définition 2.55 (Graphe de dépendance d'une matrice, matrice irréductible, voir par exemple [Rom21, GR01])**.**
▶ Soit $A = (a_{i,j}) \in \mathcal{M}_n(\mathbb{R})$. Le graphe de dépendance associé à A est le graphe orienté à n sommets tel qu'il existe une arête du nœud i vers le nœud j si et seulement si $a_{i,j} > 0$.

La matrice A est dite irréductible si son graphe de dépendance est fortement connexe. ◀

Théorème 2.56 (Théorème de Perron-Frobenius, voir par exemple [GR01, Theorem 8.8.1])**.**
▶ Soit A une matrice réelle positive et irréductible. Alors :

a. $\mathtt{Sp}(A)$ est une valeur propre de A et le sous-espace propre associé est de dimension 1, engendré par un vecteur dont les coordonnées sont strictement positives ;

b. soit B une matrice réelle positive, telle que $0 \leqslant B \leqslant A$. Alors $\mathtt{Sp}(B) \leqslant \mathtt{Sp}(A)$ avec égalité si et seulement si $A = B$.

◀

Nous utiliserons notamment le corollaire suivant, qui est une conséquence immédiate du point **b.** du théorème de Perron-Frobenius :

Proposition 2.57 .
▶ Soit A une matrice réelle positive et irréductible. Alors :

— Si $0 \leqslant B$ est obtenue à partir de A en diminuant strictement des coefficients, alors $\mathtt{Sp}(B) < \mathtt{Sp}(A)$.

— Si $0 \leqslant B$ est la matrice obtenue à partir de A en supprimant sa première ligne et sa première colonne, alors $\mathtt{Sp}(B) < \mathtt{Sp}(A)$.

◀

Pour conclure, le corollaire suivant illustre deux configurations courantes qui nous permettront d'appliquer la Proposition 2.53 :

Corollaire 2.58 .
▶ Soit $\boldsymbol{y} = \boldsymbol{F}(z; \boldsymbol{y})$ un système polynomial bien conditionné. Soit $(\rho, \boldsymbol{\tau})$ défini comme dans le Théorème 1.89, tel que $(\rho, \boldsymbol{\tau})$ est dans le domaine de convergence de $\boldsymbol{F}(z; \boldsymbol{y})$.

— Si $0 \leqslant M$ est obtenue à partir de $\mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{F}](\rho; \boldsymbol{\tau})$ en diminuant strictement des coefficients, alors $\mathtt{Sp}(B) < 1$.

— Si $0 \leqslant M$ est la matrice obtenue à partir de $\mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{F}](\rho; \boldsymbol{\tau})$ en supprimant la première ligne et la première colonne, alors $\mathtt{Sp}(B) < 1$.

◀

Démonstration. C'est immédiat en appliquant la Proposition 2.57 et le Corollaire 2.58 au résultat de la Proposition 2.54. ■

2.5.3 Preuve de la Proposition 2.36

Dans cette section, nous souhaitons démontrer la proposition suivante, où nous rappelons que $J(z)$ est définie par $J(z) = \mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}} + \boldsymbol{A}](z; \boldsymbol{L}(z))$:

Proposition 2.36 .
▶ Les coordonnées de $\boldsymbol{K} : z \mapsto (\mathtt{Id} - z \cdot J(z))^{-1} \cdot \boldsymbol{L}(z)$ ont au plus une singularité dominante en $z = \rho = \rho_{\boldsymbol{L}}$ sur le cercle $|z| = \rho$. De plus, pour tout indice j, il existe des fonctions $\tilde{g}_j, \tilde{h}_j$ analytiques en $z = \rho$ telles que $K_j(z) = \tilde{g}_j(z) - \tilde{h}_j(z)\sqrt{1 - z/\rho}$ avec $\tilde{h}_j(\rho) \neq 0$. Enfin, nous avons les équivalents asymptotiques $[z^n]K_j(z) \sim D_j \rho^{-n} n^{-3/2}$, pour des constantes $D_j > 0$. ◀

Nous quittons le monde des séries formelles, et interprétons le terme droit de l'inégalité de la Proposition 2.35 dans le monde des fonctions analytiques de la variable complexe. Nous montrons que ses coefficients ont la même croissance asymptotique que ceux de $\boldsymbol{L}(z)$ à une constante près.

Le comportement de $\boldsymbol{L}(z)$ est déjà connu par la Proposition 2.33. Nous décrivons maintenant les propriétés de la fonction $\tilde{\boldsymbol{J}}(z) := (\mathtt{Id} - z \cdot J(z))^{-1}$. Nous montrons que la singularité dominante de $\tilde{\boldsymbol{J}}(z)$ vient uniquement de celle de $\boldsymbol{L}(z)$ et non de l'inversion de la matrice.

Proposition 2.59 .
▶ Les entrées de la matrice de fonctions $\tilde{\boldsymbol{J}}(z)$ définie par

$$\tilde{\boldsymbol{J}} : z \mapsto \left(\mathtt{Id} - z \cdot \mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}} + \boldsymbol{A}](z; \boldsymbol{L}(z))\right)^{-1}$$

sont analytiques pour $|z| < \rho$, ont au plus une singularité dominante sur le cercle $|z| = \rho$, en $z = \rho$, sont continues sur $[0, \rho]$, et sont de la forme $\tilde{g}(z) - \tilde{h}(z)\sqrt{1 - z/\rho}$ autour de $z = \rho$, avec $\tilde{g}$ et $\tilde{h}$ analytiques en $z = \rho$. ◀

Démonstration. Rappelons que pour tout $i \in [i]$, $\underline{E_i}$ désigne l'ensemble des règles de E_i dont la racine n'est pas $\circledast$ et $\mathcal{A}_i$ l'ensemble des règles de $T \in E_i$ de racine $\circledast$, dont aucun fils complet n'est entièrement réductible. Notons $\mathcal{B}_i$ l'ensemble des règles $T \in E_i$ qui ont pour racine $\circledast$ et dont un des fils T' est entièrement réductible, de façon à avoir $E_i = \underline{E_i} \uplus \mathcal{A}_i \uplus \mathcal{B}_i$. Nous définissons alors

$$B_i(z, u; y_1, \ldots, y_m) = \sum_{T \in \mathcal{B}_i} z^{|T|-1} u^{|\sigma(T)|-1} \prod_{j \in \mathtt{ind}(T)} y_j \,,$$

et $\boldsymbol{B}(z, u; y_1, \ldots, y_m)$ le vecteur des $B_i(z, u; y_1, \ldots, y_m)$

Par l'hypothèse $(\mathbf{H}'_3)$, il existe au moins un indice j tel que $\mathcal{B}_j$ n'est pas vide et contient une règle d'arité non nulle. En particulier, $\mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{B}] \neq \mathbf{0}_{m \times m}$.

Posons $M(z; \boldsymbol{y}) = z \cdot \mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}} + \boldsymbol{A}](z; \boldsymbol{y})$, qui est une matrice carrée de polynômes en z et $\boldsymbol{y}$, à coefficients positifs.

Les composantes du vecteur $\boldsymbol{L}(\rho)$ étant strictement positives, une entrée de la matrice $\mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{B}](\rho; \boldsymbol{L}(\rho))$ est strictement positive. En remarquant que $\mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{\phi}] = \mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}}] + \mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{A}] + \mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{B}]$, nous en déduisons que $M(\rho; \boldsymbol{L}(\rho))$ est une matrice positive obtenue à partir de $\rho \mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{\phi}](\rho; \boldsymbol{L}(\rho))$ en diminuant strictement certaines entrées. Par le corollaire 2.58 de la section technique précédente, nous en déduisons que $\mathtt{Sp}\,(M(\rho; \boldsymbol{L}(\rho))) < 1$. En appliquant la Proposition 2.53, nous obtenons alors directement le résultat annoncé. ■

Preuve de la Proposition 2.36. Considérons la fonction

$$\boldsymbol{K}(z) := \Big(\mathtt{Id} - z \cdot \mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}} + \boldsymbol{A}](z; \boldsymbol{L}(z))\Big)^{-1} \cdot \boldsymbol{L}(z) = \tilde{\boldsymbol{J}}(z) \cdot \boldsymbol{L}(z)\,,$$

Alors le rayon de convergence des composantes de $\boldsymbol{K}(z)$ est ρ. En effet, d'après la Proposition 2.35, nous pouvons affirmer que :

$$\boldsymbol{L}(z) \preceq \partial_u \boldsymbol{L}|_{u=1} \preceq C \cdot \boldsymbol{K}(z)\,.$$

où $C > 0$ est la constante de la Proposition 2.35. Ainsi $[z^n]L_i(z) \leqslant [z^n]C \cdot K_i(z)$ et le rayon de convergence de K_i est inférieur ou égal à celui de L_i, c'est-à-dire ρ. Par la Proposition 2.59 :

- les entrées de $\boldsymbol{K}(z) = \boldsymbol{J}(z) \cdot \boldsymbol{L}(z)$ sont analytiques pour $|z| < \rho$. Donc par le théorème de Pringsheim (cf. la Proposition 1.79 des préliminaires), ρ est une singularité pour chaque entrée du vecteur $\boldsymbol{K}$.
- $z = \rho$ est la seule singularité possible sur le cercle $|z| = \rho$.
- les composantes du vecteur $\boldsymbol{K}(z)$ sont de la forme $K_i(z) = \tilde{g}_i(z) - \tilde{h}_i(z)\sqrt{1 - z/\rho}$ autour de $z = \rho$, avec $\tilde{g}_i(z)$ and $\tilde{h}_i(z)$ analytiques en $z = \rho$, par stabilité de ces fonctions par addition et multiplication (cf Lemme 2.45 des préliminaires). De plus, $\tilde{h}_i(\rho) \neq 0$ pour tout i sinon par le Théorème 1.81 de Transfert, $C \cdot [z^n]K_i(z)$ serait négligeable asymptotiquement devant $[z^n]L_i(z)$, ce qui contredit $\boldsymbol{L}(z) \preceq C \cdot \boldsymbol{K}(z)$.

Enfin le Théorème 1.81 de Transfert donne l'équivalent $[z^n]K_i(z) \sim D_i \frac{\rho^{-n}}{n^{3/2}}$, pour une certaine constante $D_i > 0$. ■

2.6 Moments d'ordres supérieurs et temps d'exécution moyen des algorithmes polynomiaux

Notre résultat principal s'étend facilement à tous les moments de la variable aléatoire associée à la taille après réduction.

Théorème 2.60 .
▶ Soit $\mathcal{E}$ un système combinatoire sur S, d'opérateur absorbant $\circledast$ et d'élément absorbant $\mathcal{P}$, qui satisfait $(\mathbf{H})$. Soit $\mathcal{L}$ une classe définie par $\mathcal{E}$. Alors tous les moments d'ordre $t > 0$ associés à la taille après réduction d'une expression aléatoire uniforme de $\mathcal{L}$ de taille n sont bornés par une constante $C_t > 0$ indépendant de n : autrement dit pour tout t et $n \in \mathbb{N}$, $\mathbb{E}_n(|\sigma|^t) \leqslant C_t$. ◀

Démonstration. La preuve suit le même raisonnement que pour le Théorème 2.32. Nous rappelons que les moments d'ordre supérieur sont reliés aux séries génératrices par la formule (voir l'Équation 1.3 des préliminaires) :

$$\mathbb{E}_n\big[|\sigma|^k\big] = \frac{[z^n](u\partial_u)^k L(z,u)\big|_{u=1}}{[z^n]L(z)}\,, \tag{2.6}$$

où $(u\partial_u)^k$ signifie que l'on a répété k fois l'opération qui consiste à dériver par rapport à u puis multiplier par u.

L'analyse du numérateur se fait par récurrence sur k et est décrite ci-dessous. Le cas $k = 1$ correspond à la valeur moyenne traitée à la section précédente. En dérivant l'équation de $L(z, u)$ on trouve

$$\begin{aligned}\partial_u \boldsymbol{L}(z,u) = pu^{p-1}\,(\boldsymbol{R}(z) - \boldsymbol{P}(z)) &+ z\left(\underline{\boldsymbol{\phi}}(z,u;\boldsymbol{L}(z,u)) + \boldsymbol{A}(z,u;\ \boldsymbol{G}(z,u))\right)\\ &+ zu\left(\partial_u\underline{\boldsymbol{\phi}}(z,u;\boldsymbol{L}(z,u)) + \partial_u\boldsymbol{A}(z,u;\ \boldsymbol{G}(z,u))\right)\\ &+ zu\mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}}](z,u;\boldsymbol{L}(z,u))\cdot\partial_u\boldsymbol{L}(z,u)\\ &+ zu\mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{A}](z,u;\boldsymbol{G}(z,u))\cdot\partial_u\boldsymbol{G}(z,u)\,.\end{aligned}$$

En étendant la notation $\preceq$ pour des séries bivariées, nous obtenons alors, comme à la Proposition 2.35, la majoration :

$$\begin{aligned}u\,\partial_u\boldsymbol{L}(z,u) \preceq\ \ & pu^p\boldsymbol{L}(z) + \boldsymbol{L}(z,u)\\ &+ zu^2\left(\partial_u\underline{\boldsymbol{\phi}}(z,u;\boldsymbol{L}(z,u)) + \partial_u\boldsymbol{A}(z,u;\ \boldsymbol{L}(z,u))\right)\\ &+ zu\cdot\mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}}+\boldsymbol{A}](z,u;\boldsymbol{L}(z,u))\cdot u\partial_u\boldsymbol{L}(z,u)\,.\end{aligned}$$

Comme les coefficients de la série sont positifs, nous pouvons dériver en u tout en conservant l'inégalité. On prouve alors par récurrence directe que :

$$\begin{aligned}(u\,\partial_u)^k\boldsymbol{L}(z,u) \preceq\ & \boldsymbol{p}_k\big(z,u,\boldsymbol{L}(z),\ \boldsymbol{L}(z,u),(u\partial_u)\boldsymbol{L}(z,u),\dots,(u\partial_u)^{k-1}\boldsymbol{L}(z,u)\big)\\ &+ zu\cdot\mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}}+\boldsymbol{A}](z,u;\boldsymbol{L}(z,u))\cdot(u\partial_u)^k\boldsymbol{L}(z,u)\,,\end{aligned}$$

où les $\boldsymbol{p}_k$ sont des polynômes à coefficients positifs.
En spécialisant $u = 1$, on préserve l'inégalité :

$$\begin{aligned}(u\,\partial_u)^k\boldsymbol{L}(z,u)\big|_{u=1} \preceq\ & \boldsymbol{p}_k\big(z,1,\boldsymbol{L}(z),\ \boldsymbol{L}(z),\dots,(u\partial_u)^{k-1}\boldsymbol{L}(z,u)\big|_{u=1}\big)\\ &+ z\cdot\mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}}+\boldsymbol{A}](z;\boldsymbol{L}(z))\cdot(u\partial_u)^k\boldsymbol{L}(z,u)\big|_{u=1}\,,\end{aligned}$$

et ainsi le Lemme 2.43 implique l'inégalité

$$\begin{aligned}&(u\,\partial_u)^k\boldsymbol{L}(z,u)\big|_{u=1} \preceq\\ &\big(\mathtt{Id} - z\cdot\mathtt{Jac}_{\boldsymbol{y}}[\underline{\boldsymbol{\phi}}+\boldsymbol{A}](z;\boldsymbol{L}(z))\big)^{-1}\,\boldsymbol{p}_k\big(z,1,\boldsymbol{L}(z),\ \boldsymbol{L}(z),\dots,(u\partial_u)^{k-1}\boldsymbol{L}(z,u)\big|_{u=1}\big)\end{aligned}$$

La preuve suit par récurrence. En effet, si nous supposons avoir prouvé que les fonctions $(u\partial_u)^j\boldsymbol{L}(z,u)\big|_{u=1}$, pour $j < k$, sont toutes bornées par des fonctions qui ont ρ comme unique singularité sur le cercle $|z| = \rho$, et qui ont une expansion locale en racine carrée autour de $z = \rho$, nous pouvons appliquer la Proposition 2.36 (qui étudie la fonction $(\mathtt{Id} - z\cdot J(z))^{-1}$) pour en déduire que $(u\,\partial_u)^k\boldsymbol{L}(z,u)$ est bornée par une combinaison linéaire de produits de fonctions qui ont ρ comme unique singularité sur le cercle $|z| = \rho$, et qui ont une expansion locale en racine carrée autour de $z = \rho$. En utilisant les propriétés de clôture des fonctions de cette forme, exposées au Lemme 2.45, nous en déduisons l'hérédité de la récurrence. ∎

Au-delà de l'intérêt purement mathématique des moments d'ordres supérieurs, le Théorème 2.60 a une autre conséquence sur l'analyse des algorithmes en temps polynomial qui traitent des expressions aléatoires :

Proposition 2.61 .
▶ Soit $\mathcal{A}$ un algorithme qui s'exécute en temps polynomial dans le pire cas, dont les entrées sont des expressions spécifiées par un système satisfaisant les hypothèses du Théorème 2.60. Alors l'algorithme qui consiste à réduire d'abord l'entrée avant d'appliquer l'algorithme $\mathcal{A}$ dessus, a une complexité en moyenne linéaire en la taille de l'entrée. ◀

Démonstration. Le résultat vient du fait que la réduction σ s'exécute en temps linéaire sur l'entrée. Comme en moyenne, la taille de la réduction est bornée par une constante, l'algorithme $\mathcal{A}$ s'exécute ensuite en temps moyen constant.

Plus rigoureusement, soit c la fonction de complexité qui à une expression T associe le nombre $c(T)$ d'opérations élémentaires utilisées par l'algorithme $\mathcal{A}$ pour traiter T. Soit $p(x) = \sum_{k=0}^{r} a_k x^k$ un polynôme tel que pour tout T de taille $n \in \mathbb{N}$, $c(T) \leqslant p(n)$.

Soit $pre(T) \leqslant \alpha|T|$ la fonction de complexité linéaire du programme qui calcule la réduction $\sigma(T)$ d'une expression T.

Notons $c'(T) = pre(T) + c(\sigma(T))$ la fonction de complexité du programme $\mathcal{A}'$ qui consiste à réduire d'abord T et à appliquer ensuite l'algorithme $\mathcal{A}$ sur $\sigma(T)$.

Ainsi si $|T| = n$, $c'(T) \leqslant \alpha n + \sum_{k=0}^{r} a_k(\sigma(T))^k$.

Alors la complexité en moyenne de l'algorithme $\mathcal{A}'$ sur les entrées de taille n est linéaire, par le Théorème 2.60 :

$$\mathbb{E}_n(c') \leqslant \alpha n + \sum_{k=0}^{r} a_k \mathbb{E}_n(|\sigma|^k) \leqslant \alpha n + \sum_{k=0}^{r} a_k C_k \,.$$

■

2.7 Conclusion de la partie et discussion sur le cas des expressions régulières

La contribution principale de ce chapitre est de montrer que les arbres d'expressions uniformes décrits par des systèmes manquent d'expressivité, car ils se réduisent considérablement dès qu'un opérateur admet un élément absorbant. Cela remet en question la pertinence de cette distribution pour l'analyse théorique en moyenne et la génération d'arbres d'expressions. Comme nous l'avons déjà mentionné dans le coeur du chapitre, les hypothèses que nous avons données sur les systèmes n'ont pas pour but de caractériser tous les systèmes bien fondés où un tel phénomène de dégénérescence se produit. Nous voulions juste identifier un cadre suffisamment général pour capturer de nombreux exemples, tout en gardant des conditions faciles à vérifier.

Discussion sur les hypothèses du chapitre. Nous pouvons ainsi discuter un peu de ce qui se passe si les hypothèses (**H**) ne sont pas vérifiées.

Pour ($\mathbf{H}_1$), si le graphe de dépendance n'est pas fortement connexe, nous pouvons considérer le DAG de ses composantes fortement connexes ; s'il n'y a qu'une seule composante fortement connexe finale, ou s'il n'y a qu'une seule composante dominante, qui produit exponentiellement plus d'expressions que les autres composantes, on devrait pouvoir se réduire à notre cadre en analysant ce qui se passe dans cette composante principale. Si plusieurs composantes connexes sont en concurrence, la

situation est moins claire, et plus difficile à analyser d'un point de vue combinatoire, les asymptotiques ne sont plus forcément de la forme $\alpha\beta^n n^{-3/2}$ (voir [BD15] pour les formes des asymptotiques possibles).
Si $\mathcal{G}_\square$ est cyclique, alors le système soit n'est pas bien fondé, soit il est possible de retirer certains cycles.

Pour ($\mathbf{H}_2$), nous pouvons essayer de traiter séparément les tailles modulo la période. Cela se produit notamment lorsque tous les opérateurs sont binaires (dans ce cas, la taille des arbres est nécessairement impaire).

Pour ($\mathbf{H}_3$), on pourrait affaiblir cette condition, mais elle est là essentiellement pour garantir que le système n'est pas capable d'empêcher l'élément absorbant d'apparaître un grand nombre de fois sous son opérateur, ni de limiter la taille des arbres réduits par la règle d'absorbance.

Pour ($\mathbf{H}_4$), si le système est linéaire, les arbres sont filiformes, et l'analyse est complètement différente, bien que sans doute plus facile. Nous ne nous sommes pas intéressés au cas linéaire car nous n'avions pas d'exemples concrets suffisamment motivés.

Si ($\mathbf{H}_5$) n'est pas vérifiée, nous avons déjà expliqué en note de marge que le résultat restait valide si la spécification était finiment ambiguë. Si la spécification est infiniment ambiguë, les séries génératrices déduites du système ne représentent pas bien la combinatoire de l'ensemble d'arbres d'expressions étudié, si bien qu'il est nécessaire de chercher une spécification non ambiguë du problème.

Comparaison avec des systèmes utilisés dans la littérature. Si les restrictions des hypothèses ($\mathbf{H}$) semblent contraignantes, il est important de rappeler qu'elles sont suffisamment larges pour s'appliquer à des systèmes naturels de la littérature. Ainsi, dans [LS05], les auteurs étudient plusieurs grammaires pour décrire les expressions régulières, dans le but de les énumérer, en tentant d'éviter certaines redondances. Par exemple, les auteurs considèrent la grammaire suivante pour compter les expressions régulières :

$$\begin{aligned}
S &\to E_+ \,|\, E_\bullet \,|\, E_* \,|\, C \\
E_+ &\to E_+ + F \,|\, F + F \\
F &\to E_\bullet \,|\, E_* \,|\, C \\
E_\bullet &\to E_\bullet G \,|\, GG \\
G &\to (E_+) \,|\, E_* \,|\, C \\
C &\to \emptyset \,|\, \varepsilon \,|\, a \quad (a \in \Sigma) \\
E_* &\to (E_+)* \,|\, (E_\bullet)* \,|\, E_** \,|\, C*
\end{aligned}$$

Techniquement, nous ne pouvons pas utiliser tels quels les théorèmes de cette partie, car la grammaire ci-dessus définit non pas des arbres d'expressions régulières, mais des chaînes de caractères définissant des expressions régulières, ce qui est différent. La grammaire doit ainsi s'occuper de parenthéser correctement les expressions régulières, tandis que les points de concaténation sont implicites. Si les tailles ne seront donc pas exactement les mêmes, nous pouvons facilement transposer ce système dans le monde des arbres d'expressions, et le nouveau système vérifie facilement les hypothèses ($\mathbf{H}$) : la seule condition non remplie est celle de la forte connexité, mais nous pouvons la régler en modifiant légèrement le système. En effet nous pouvons supprimer la règle $C \to \emptyset \,|\, \varepsilon \,|\, a \quad (a \in \Sigma)$ en remplaçant C par ses

productions. Dans notre formalisme arborescent, nous obtenons ainsi le système combinatoire suivant :

J'ai supprimé les parenthèses de la grammaire, inutiles dans le formalisme arborescent.

$$
\begin{aligned}
E_+ &= \overset{+}{\underset{E_+\ F}{\wedge}} + \overset{+}{\underset{F\ F}{\wedge}} \\
F &= E_\bullet + E_* + \emptyset + \varepsilon + a \quad (a \in \Sigma) \\
E_\bullet &= \overset{\bullet}{\underset{E_\bullet\ G}{\wedge}} + \overset{\bullet}{\underset{G\ G}{\wedge}} \\
G &= E_+ + E_* + \emptyset + \varepsilon + a \quad (a \in \Sigma) \\
E_* &= \overset{*}{\underset{E_+}{|}} + \overset{*}{\underset{E_\bullet}{|}} + \overset{*}{\underset{E_*}{|}} + \overset{*}{\underset{\emptyset}{|}} + \overset{*}{\underset{\varepsilon}{|}} + \overset{*}{\underset{a}{|}} \quad (a \in \Sigma)
\end{aligned}
$$

et la classe d'arbres étudiés est la classe $\mathcal{L} = E_+ \cup F \cup E_\bullet \cup G \cup E_*$. On vérifie bien que ce système satisfait toutes les hypothèses de (**H**).

Pour les mêmes raisons, la grammaire suivante considérée aussi dans [LS05] vérifie à quelques légères modifications près les hypothèses (**H**) :

$$
\begin{aligned}
S &\to A \mid B \mid C \mid D \mid \varepsilon \mid \emptyset \\
A &\to A_0 + A_0 \mid A_0 + A \\
A_0 &\to B \mid C \mid D \mid \varepsilon \\
B &\to B_0 B_0 \mid B_0 B \\
B_0 &\to (A) \mid C \mid D \\
C &\to (A)^* \mid (B)^* \mid D^* \\
D &\to \emptyset \mid \varepsilon \mid a \quad (a \in \Sigma)
\end{aligned}
$$

Afin de prendre en compte la commutativité du $+$, et certaines redondances, comme le fait que $(L_1^* + \ldots + L_n^*)^* = (L_1 + \ldots + L_n)^*$, les auteurs remplacent dans la grammaire précédente les règles de A, A_0 et C par :

$$
\begin{aligned}
A &\to \varepsilon + A_B \mid C + A_C \mid B + A_B \mid D + A_D \mid \varepsilon \mid \emptyset \\
A' &\to B + A_B \mid D + A_D \\
A_C &\to C \mid C + A_C \mid A_B \\
A_B &\to B \mid B + A_D \mid A_D \\
A_D &\to D \mid D + A_D \\
C &\to (A')^* \mid (B)^* \mid D^*
\end{aligned}
$$

Avec ces modifications, nous avons à nouveau un problème apparent de forte connexité pour vérifier les hypothèses (**H**), à cause des règles $D \to \emptyset \mid \varepsilon \mid a \quad (a \in \Sigma)$ et $A_D \to D \mid D + A_D$. Cette fois il n'est pas possible de remplacer simplement D et A_D par leurs productions, car le langage de A_D est infini. Néanmoins, le but de la grammaire étant d'éviter des redondances, nous pouvons questionner la pertinence de la règle $A_D \to D \mid D + A_D$, qui autorise des expressions complètement redondantes, comme $a + a$, $a + a$, $a + a + a + a$, ... En remplaçant les occurrences de D par ses productions finies, et les occurrences de A_D par $\bigcup_{X \subseteq \Sigma \cup \{\varepsilon\}} \sum_{x \in X} x$, nous observons que la grammaire vérifie bien les hypothèses (**H**).

Abondance d'éléments absorbants. Nous finissons cette conclusion en soulignant le fait que dans tout le chapitre, le pattern $\mathcal{P}$ désigne un élément absorbant fixé

après la spécification combinatoire des arbres. Si la spécification donnée empêche l'apparition d'un élément absorbant particulier, il peut être tout à fait possible d'en trouver d'autres. Par exemple, dans le cas des expressions régulières, il est très facile de trouver des spécifications empêchant les motifs $(a+b)^*$ et $(b+a)^*$ sous un $+$. Nous pouvons alors trouver à la place, comme candidats pour $\mathcal{P}$, un grand nombre d'expressions équivalentes, comme $(a+\varepsilon+b)^*$, $((a+\varepsilon)(b+\varepsilon))^*$, etc. En fait, il semble impossible par une simple spécification d'empêcher l'apparition d'un sous-arbre universel sous un opérateur $+$, à moins de considérablement restreindre les expressions régulières décrites par la spécification. Cette intuition est motivée par le fait que détecter l'universalité d'une expression régulière est PSPACE-complet [MS72]. Dans un article en prépublication [BMMR21], les auteurs étudient un système qui évite les motifs $(a+b)^*$ et $(b+a)^*$:

L'abondance d'expressions universelles de cette forme sera exploitée dans le chapitre 4.

$$\begin{aligned}
\alpha &\to \varepsilon \mid a \mid b \mid (\alpha \cdot \alpha) \mid (\alpha^*) \mid (\alpha_P + \alpha_P)\\
\alpha_P &\to \varepsilon \mid a \mid b \mid (\alpha \cdot \alpha) \mid (\alpha_\Sigma^*) \mid (\alpha_P + \alpha_P)\\
\alpha_\Sigma &\to \varepsilon \mid a \mid b \mid (\alpha \cdot \alpha) \mid (\alpha^*) \mid \gamma\\
\gamma &\to (\alpha_{ab} + \alpha_{ab}) \mid (\alpha_{ab} + a) \mid (\alpha_{ab} + b) \mid (a + \alpha_{ab}) \mid (b + \alpha_{ab}) \mid (a + a) \mid (b + b)\\
\alpha_{ab} &\to \varepsilon \mid (\alpha \cdot \alpha) \mid (\alpha_\Sigma^*) \mid (\alpha_P + \alpha_P)
\end{aligned}$$

Les auteurs affirment à tort qu'en évitant ces deux motifs, notre résultat ne s'applique pas à leur système. Même si on ne peut choisir $\mathcal{P} = (a+b)^*$, on remarque que α_P engendre l'expression universelle $(((a+b)^*)^*)$:

$$\alpha_P \to (\alpha_\Sigma^*) \to ((\alpha^*)^*) \to (((\alpha_P + \alpha_P)^*)^*) \to (((a+b)^*)^*)\,.$$

En choisissant $\mathcal{P} = (((a+b)^*)^*)$, on vérifie aisément que ce système vérifie les hypothèses (**H**). Par conséquent, ce système n'est pas pertinent pour étudier en moyenne des algorithmes portant sur des expressions régulières.

Expressions régulières sans mot vide. On pourrait objecter que pour les expressions régulières, il est en fait facile d'éviter l'arbre universel $(a+b)^*$ en interdisant à la grammaire décrivant les arbres de faire l'union de deux expressions qui reconnaissent le mot vide ε. C'est tout à fait envisageable car cela ne réduit presque pas la puissance des expressions régulières : en effet toute expression régulière E se transforme facilement par induction en une expression régulière $\psi(E)$ reconnaissant le même langage que E privé de ε :

- $\psi(a) = a$, $\psi(b) = b$, $\psi(\varepsilon) = \emptyset$
- $\psi(E_1 + E_2) = \psi(E_1) + \psi(E_2)$
- $\psi(E_1 \cdot E_2) = \psi(E_1) \cdot \psi(E_2) + \psi(E_1)\mathbf{1}_{\varepsilon\in\mathcal{L}(E_1)} + \psi(E_2)\mathbf{1}_{\varepsilon\in\mathcal{L}(E_2)}$
- $\psi(E_1^*) = \psi(E_1) \cdot \psi(E_1)^*$.

On vérifie aisément que dans l'expression $\psi(E)$, l'opérateur $+$ n'a aucun fils reconnaissant le mot vide.

Si des expressions régulières sont spécifiées par une grammaire interdisant l'union de langages reconnaissant ε, il n'est alors plus possible de trouver d'expressions équivalente à $(a+b)^*$ sous un $+$: l'élément absorbant semble avoir été neutralisé. Néanmoins, pour des expressions de cette forme particulière, l'union $+$ admet alors un nouvel élément absorbant, le langage représenté par l'expression $(a+b)^+$, et donc nos résultats s'appliquent encore.

Ce chapitre est en grande partie issu de l'article de conférence [KNR19].

3

Réduction d'expressions spécifiées par une seule équation

3.1 Introduction

Dans ce chapitre, nous nous intéressons à l'influence de la présence d'un élément absorbant sur l'ensemble particulier de tous les arbres compatibles avec un ensemble d'opérateurs S et une fonction d'arité a donnés. L'ensemble d'arbres étudié est donc $\mathcal{T}(S, a)$ tout entier, en reprenant les notations du chapitre précédent, et non plus un sous-ensemble spécifié par un système d'équations sur les arbres. Nous généralisons dans ce chapitre légèrement la fonction d'arité par rapport au chapitre précédent, en autorisant des opérateurs d'arité quelconque, ce qui permet de modéliser l'associativité de certains opérateurs. L'ensemble $\mathcal{T}(S, a)$ est décrit par une seule équation unidimensionnelle très simple, ce qui permet d'obtenir des résultats plus précis sur son comportement vis-à-vis de la simplification, notamment la convergence de l'espérance de la taille après réduction.

Le but de cette section est de démontrer le théorème suivant, dont nous préciserons en détail les conditions dans la prochaine sous-section 3.2 :

Théorème 3.1.
▶ Soit $\mathcal{L}$ un ensemble d'expressions dont la série génératrice $L(z)$ vérifie le *schéma d'inversion lisse*, avec $L(z) = z\phi(L(z))$ et ϕ apériodique.
Soit $\mathcal{P} \in \mathcal{L}$ un arbre d'expression, et $\circledast$ un opérateur d'arité au moins 2, tel que $\mathcal{P}$ est absorbant pour l'opérateur $\circledast$. Nous considérons la simplification σ induite par cette relation d'absorption.

Alors l'espérance de la taille, après simplification par la réduction σ, d'une expression uniforme de taille n tend vers une constante quand n tend vers l'infini : $\lim_{n\to\infty} \mathbb{E}_n[|\sigma|] = \delta$, avec $\delta > 0$. De plus, tous les moments de σ convergent aussi : pour tout $i \in \mathbb{N}^*$, $\lim_{n\to\infty} \mathbb{E}_n[|\sigma|^i] = \delta_i$, avec $\delta_i > 0$. ◀

Les techniques utilisées dans ce chapitre sont plus simples qu'au chapitre précédent, puisqu'il n'y a pas de système à gérer : l'équation $L(z) = z\phi(L(z))$ est à une seule dimension. Au cours de la thèse, ce travail est venu historiquement en premier par

rapport au chapitre précédent ; il constitue notre premier travail sur les simplifications d'arbres par un élément absorbant.

3.2 Contexte et restrictions

Comme l'ensemble d'arbres étudiés est moins contraint que dans le chapitre précédent (une seule équation unidimensionnelle), nous pouvons généraliser un petit peu l'ensemble d'opérateurs et la fonction d'arité associée sans trop compliquer les techniques de preuve. Le cadre d'étude de ce chapitre est donc un peu plus général, et ne se résume pas à une simple restriction du chapitre précédent aux systèmes à une dimension.

3.2.1 Arbres d'expressions

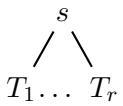
Soit S un ensemble *dénombrable* d'opérateurs, et a une fonction d'arité associée, qui a tout opérateur lui associe un ensemble (non forcément fini) d'arités possibles. Nous notons $\mathcal{A}_i$ le sous-ensemble d'opérateurs d'arité $i \in \mathbb{N}$. Nous supposons que tous les $\mathcal{A}_i$ sont finis (autrement dit a n'associe la même arité qu'à un nombre fini d'opérateurs ; cette condition est nécessaire pour avoir un nombre fini d'arbres d'une taille donnée).

Exemple 3.2 .
▶ Dans le cas des expressions régulières, nous pouvons considérer que l'union $+$ est associative. Ainsi, un nœud étiqueté par $+$ peut avoir plusieurs fils. Dans ce cas $a(+) = \mathbb{N}_{\geqslant 2}$, et alors $\mathcal{A}_0 = \{a, b, \varepsilon\}$, $\mathcal{A}_1 = \{\star\}$, $\mathcal{A}_2 = \{\bullet, +\}$ et enfin $\mathcal{A}_i = \{+\}$ pour $i \geqslant 3$. Nous définissons $\mathcal{L}'_{\mathcal{R}}$ l'ensemble des expressions régulières associées à ces arités. ◀

Nous généralisons alors facilement la définition d'arbres d'expressions au cas dénombrable :

Définition 3.3 .
▶ L'ensemble des expressions $\mathcal{T}(S, a)$ associées à la paire (S, a) est défini par induction :

- pour tout opérateur $s_0 \in S$ d'arité 0, $s_0 \in \mathcal{T}(S, a)$;
- pour tout opérateur $s \in S$ et $(T_1, \ldots, T_r) \in \mathcal{T}(S, a)$ avec $r \in a(S)$, alors $(s, T_1, \ldots, T_r) \in \mathcal{T}(S, a)$. Ce tuple formel est représenté sous la forme d'un arbre :



◀

L'ensemble $\mathcal{T}(S, a)$ est donc entièrement caractérisé par la suite $\mathcal{A} = (\mathcal{A}_i)_{i \in \mathbb{N}}$. Nous nous plaçons dans le cas particulier où :

a. $\mathcal{A}_0 \neq \emptyset$ (il y a des feuilles) ;

b. il existe un opérateur $\circledast$ et une arité $\mathfrak{a} \geqslant 2$ telle que $\circledast \in \mathcal{A}_{\mathfrak{a}}$.

3.2.2 Élément absorbant et réduction

Dans la suite, nous fixons une arité $\mathfrak{a} \geqslant 2$ de l'opérateur $\circledast$. Nous considérons la réduction induite par l'existence d'un arbre constant fixé, $\mathcal{P} \in \mathcal{T}(S, a)$, qui est absorbant pour l'opérateur $\circledast$ d'arité $\mathfrak{a}$. On s'intéresse donc à la règle de simplification suivante :

$$\overset{\circledast}{/\ \ \backslash}_{T_1 \cdots T_{\mathfrak{a}}} \rightsquigarrow \mathcal{P}\,, \text{ si } T_i = \mathcal{P} \text{ pour un } i \in \{1, \ldots, \mathfrak{a}\}.$$

Remarquez que la réduction considérée ne s'applique que si le père de $\mathcal{P}$ est un nœud $\circledast$ d'arité $\mathfrak{a}$, alors que l'opérateur $\circledast$ peut avoir d'autres arités. Dans l'exemple 3.2 des expressions régulières $\mathcal{L}'_{\mathcal{R}}$ avec une union d'arité quelconque, nous ne considèrerons, par exemple, que la réduction sur les opérateurs $+$ d'arité 2. Cela suffit pour montrer la dégénérescence des expressions.

3.2.3 Le schéma d'inversion lisse

Dans cette section nous définissons le cadre dans lequel nous nous plaçons.

Définition 3.4 .
▶ La *série caractéristique* associée à la suite $\mathcal{A} = (\mathcal{A}_i)_{i\in\mathbb{N}}$ est la série formelle $\phi_{\mathcal{A}}(y) = \sum_{i\geqslant 0} \phi_i z^i$, où $\phi_i = |\mathcal{A}_i|$. ◀

Pour simplifier les notations, nous notons $\mathcal{L} := \mathcal{T}(S, a)$ l'ensemble d'arbres étudié, $\mathcal{A}$ sa suite d'opérateurs triés par arité, et ϕ la série caractéristique associée à $\mathcal{A}$. Avec ces notations, la série génératrice de $\mathcal{L}$ satisfait l'équation :

$$L(z) = z\,\phi(L(z))\,. \tag{3.1}$$

Si on prend l'exemple des expressions régulières $\mathcal{L}_{\mathcal{R}}$, $\phi(X) = 3 + X + 2X^2$ est un polynôme de degré 2 et l'Eq. (3.1) peut se résoudre. Ce n'est pas possible en général lorsque ϕ est une série quelconque. Cependant dans certains cas on peut dériver le comportement asymptotique de l'équation fonctionnelle, comme on l'a déjà fait dans le cas des systèmes. Pour cela on demande un certain nombre de conditions, regroupées sous le nom de *schéma d'inversion lisse pour les arbres* [FS09, Definition VII.3] :

Définition 3.5 (Schéma d'inversion lisse [FS09])**.**
▶ Une série $L(z)$ satisfaisant l'équation fonctionnelle $L(z) = z\phi(L(z))$ vérifie le *schéma d'inversion lisse* si

Smooth inverse-function schema en anglais.

a. $L(z)$ est analytique en $z = 0$

b. $\phi(y)$ est analytique en $y = 0$

c. $\phi(0) \neq 0$

d. les coefficients de Taylor de ϕ sont positifs

e. ϕ n'est pas linéaire

f. il existe une solution τ à l'équation $\phi(\tau) - \tau\phi'(\tau) = 0$, avec $0 < \tau < \rho_\phi$ où ρ_ϕ est le rayon de convergence de ϕ.

◀

Il s'agit du pendant unidimensionnel du théorème de Drmota analytique présenté dans les préliminaires (Théorème 1.89).

Analysons cette définition dans le cas où $\mathcal{L}$ est une ensemble d'expressions de série caractéristique $\phi(y)$. L'analycité de $\phi(y)$ en 0 implique qu'il n'y a pas trop d'opérateurs d'arité i (leur nombre ne peut pas être super-exponentiel en i), ce qui est une contrainte naturelle, vérifiée dans la totalité des exemples proposés dans cette section. Toutes les autres conditions sont satisfaites par les hypothèses de travail sur la suite $\mathcal{A}$ énoncées en sous-section 3.2.1, à l'exception de la dernière. En fait, la solution τ à l'équation $\phi(\tau) - \tau\phi'(\tau) = 0$ existe toujours et est unique, mais nous devons nous assurer qu'elle est bien plus petite que le rayon de convergence de ϕ pour garantir un comportement asymptotique en racine carrée. La remarque suivante est élémentaire mais très utile en pratique :

Remarque 3.6 .
▶ Si seulement un nombre fini de $\mathcal{A}_i$ est non vide, i.e. si ϕ est un polynôme, alors la série des arbres construits sur les opérateurs $\mathcal{A}$ vérifie le *schéma d'inversion lisse.* ◀

En clair, le schéma d'inversion lisse n'est là que pour s'assurer, dans le cas où ϕ n'est pas un polynôme, que les arbres décrits ont un comportement standard d'arbres.

Dans le cas des expressions régulières $\mathcal{L}'_{\mathcal{R}}$ avec l'opérateur $+$ d'arité non bornée, la série caractéristique associée est $\phi(y) = 3 + y + 2y^2 + \frac{y^3}{1-y} = \frac{1}{1-y} + y^2 + 2$. Son rayon de convergence est $\rho_\phi = 1$, et la solution positive de $\phi(\tau) - \tau\phi'(\tau) = 0$ est $\frac{\sqrt{5}-1}{2} \approx 0.618$. Donc la série génératrice de $\mathcal{L}'_{\mathcal{R}}$ satisfait le *schéma d'inversion lisse.*

3.2.4 Comportement asymptotique du nombre d'arbres dans $\mathcal{L}$

Le théorème suivant fournit le développement asymptoptique des fonctions satisfaisant le *schéma d'inversion lisse.* Il est cité dans le cas où $\phi(y)$ est *apériodique*, c'est-à-dire quand $\phi(y) = y^r\psi(y^d)$ implique $d = 1$. Cette restriction, qui assure que $L(z)$ est aussi apériodique et a une unique singularité dominante, peut être évitée (cf [FS09, Ex. VI.17]), mais elle suffira en pratique.

Théorème 3.7 ([Drm97, FS09, MM89]).
▶ Soit $L(z)$ une série vérifiant le *schéma d'inversion lisse*, avec $L(z) = z\phi(L(z))$. Si $\phi(y)$ est apériodique, alors $L(z)$ a une unique singularité dominante $\rho_L = \tau/\phi(\tau)$. De plus, il existe deux fonctions $g(z)$ et $h(z)$ analytiques en $z = \rho_L$, telles que localement autour de $z = \rho_L$ on ait :

$$L(z) = g(z) - h(z)\sqrt{1 - z/\rho_L}, \text{ avec } h(\rho_L) \neq 0. \tag{3.2}$$

De plus, nous avons l'équivalent asymptotique $[z^n]L(z) \sim C_L \frac{\rho_L^{-n}}{n^{3/2}}$, avec $C_L = \sqrt{\frac{\phi(\tau)}{2\phi''(\tau)}}$. ◀

Maintenant que la série totale des arbres de $\mathcal{L}$ a été étudiée, nous pouvons étudier plus en détail le phénomène de réduction.

3.3 Étude de la réduction

3.3.1 La classe des entièrement réductibles

Nous rappelons qu'une expression T est *entièrement réductible* si $\sigma(T) = \mathcal{P}$. Nous notons $\mathcal{R} \subset \mathcal{L}$ l'ensemble des arbres entièrement réductibles, et $R(z)$ sa série

génératrice.

Nous posons $p = |\mathcal{P}|$, et nous rappelons que $\mathfrak{a}$ désigne l'arité que doit avoir $\circledast$ pour pouvoir appliquer la règle d'absorption. Un élément de $\mathcal{R}$ est donc soit $\mathcal{P}$, soit un arbre de racine $\circledast$, avec $\mathfrak{a}$ fils, dont au moins un est complètement réductible. Cela se traduit facilement en l'équation suivante sur les séries génératrices :

$$R(z) = z^p + z\left((L(z))^{\mathfrak{a}} - (L(z) - R(z))^{\mathfrak{a}}\right) . \tag{3.3}$$

Nous déduisons de cette équation le lemme suivant sur la série $R(z)$:

Lemme 3.8.
▶ Les rayons de convergence de $R(z)$ et $L(z)$ sont égaux. De plus, il existe un entier $N_R \in \mathbb{N}$ tel que pour tout $n \geqslant N_R$, $[z^n]R(z) > 0$. ◀

Démonstration. Soient ρ_R et ρ_L les rayons de convergence des séries $R(z)$ et $L(z)$. Comme $\mathcal{R} \subset \mathcal{L}$, $[z^n]R(z) \leqslant [z^n]L(z)$ et donc $\rho_L \leqslant \rho_R$.

Pour l'inégalité inverse, considérons la famille $\tilde{\mathcal{R}} \subset \mathcal{R}$ d'expressions de la forme $(\circledast, \mathcal{P}, L, \ell_0, \ldots, \ell_0)$, avec $L \in \mathcal{L}$, et ℓ_0 une feuille fixée. La série génératrice $\tilde{R}(z)$ de $\tilde{\mathcal{R}}$ est $\tilde{R}(z) = z^{p+\mathfrak{a}-1}L(z)$, et a donc le même rayon de convergence que $L(z)$. De l'inclusion $\tilde{\mathcal{R}} \subset \mathcal{R}$, nous déduisons $\rho_{\tilde{R}} \geqslant \rho_R$, et donc $\rho_L \geqslant \rho_R$.

La seconde partie du lemme est aussi une conséquence de l'inclusion $\tilde{\mathcal{R}} \subset \mathcal{R}$: l'égalité $\tilde{R}(z) = z^{p+\mathfrak{a}-1}L(z)$ implique que $[z^n]\tilde{R}(z) = [z^{n-p-\mathfrak{a}+1}]L(z)$, pour n assez grand. Comme $[z^n]L(z)$ est non nul à partir d'un certain rang (par apériodicité de ϕ), nous en déduisons que c'est aussi le cas pour $[z^n]\tilde{R}(z)$, et donc pour $[z^n]R(z)$. ■

La proposition suivante précise le comportement de $R(z)$:

Proposition 3.9.
▶ La fonction $R(z)$ a une unique singularité dominante en $z = \rho$. De plus, autour de $z = \rho$, la fonction $R(z)$ peut s'écrire

$$R(z) = g_R(z) - h_R(z)\sqrt{1 - \frac{z}{\rho}}\,, \tag{3.4}$$

où ρ est le rayon de convergence commun à $L(z)$ et $R(z)$, et les fonctions $g_R(z)$ et $h_R(z)$ sont analytiques autour de $z = \rho$, et $h_R(\rho) \neq 0$. ◀

Démonstration. Nous introduisons la classe complémentaire des entièrement réductibles, afin de nous ramener à des équations à coefficients positifs. L'ensemble $\mathcal{G} := \mathcal{L} \setminus \mathcal{R}$ rassemble ainsi les arbres qui ne se réduisent pas complètement à $\mathcal{P}$. Nous notons $G(z)$ sa série génératrice. Les séries $R(z)$ et $G(z)$ satisfont facilement le système :

$$\begin{cases} R(z) = & z^p + z\Big((G(z) + R(z))^{\mathfrak{a}} - (G(z))^{\mathfrak{a}}\Big)\,, \\ G(z) = & -z^p + z\,\underline{\phi}\big(G(z) + R(z)\big) + zG(z)^{\mathfrak{a}}\,, \end{cases} \tag{3.5}$$

où $\underline{\phi}(x) = \phi(x) - x^{\mathfrak{a}}$. En effet, la première équation est simplement une réécriture de l'Eq. (3.3), tandis que la seconde vient du fait qu'un arbre n'est pas entièrement réductible si et seulement si il est différent de $\mathcal{P}$ (d'où le $-z^p$) et vérifie l'une des conditions suivantes :

— il commence par un nœud différent de $\circledast$, ou par $\circledast$ mais avec une arité différente de $\mathfrak{a}$; cela donne le terme en $\underline{\phi}$;

— il commence par $\circledast$ d'arité $\mathfrak{a}$, mais aucun de ses $\mathfrak{a}$ fils n'est entièrement réductible (d'où le terme $zG(z)^{\mathfrak{a}}$).

Malheureusement ce système n'est pas positif (il y a le terme $-z^p$). Il faut d'abord le transformer en système positif pour pouvoir appliquer le théorème de Drmota.

Pour cela, nous allons développer le système pour faire disparaître le $-z^p$. Tout d'abord, remarquons qu'à partir d'un certain rang q, $[z^n]R(z) > 0$ et $[z^n]G(z) > 0$ pour $n \geqslant q$. Cela vient du Lemme 3.8 pour $R(z)$. Pour les coefficients de $G(z)$ il suffit de remarquer que comme $L(z)$ est apériodique, il existe un autre opérateur dans la spécification de $\mathcal{L}$, $op \neq \circledast$, d'arité $\tilde{\mathfrak{a}}$. Toutes les expressions de la forme $op(\mathcal{L}, \ell_0, \dots, \ell_0)$ où ℓ_0 est une feuille sont donc dans G, et ainsi $[z^n]G(z) \geqslant [z^n]z^{\tilde{\mathfrak{a}}}L(z) = [z^{n-\tilde{\mathfrak{a}}}]L(z) > 0$ pour n assez grand.

Nous pouvons supposer sans perte de généralité que $q > p \geqslant 1$ et $q > \tilde{\mathfrak{a}} \geqslant 1$.

Posons ainsi $R(z) = P_R(z) + z^q R_q(z)$, $G(z) = P_G(z) + z^q G_q(z)$, avec $P_G(z) = [z^{\leqslant q}]G(z)$, $P_R = [z^{\leqslant q}]R(z)$, qui sont des polynômes de degré q, nuls en $z = 0$, et $G_q(0) = R_q(0) = 0$ (G_q et R_q ont un z en facteur). De même, nous posons $\underline{\phi} = P_\phi(y) + y^q \underline{\phi}_q(y)$ avec $\underline{\phi}_p(0) = 0$, et de plus P_ϕ est un polynôme de degré au moins $\tilde{\mathfrak{a}} \geqslant 1$. Ainsi

$$\begin{cases} R(z) = z^p + z\displaystyle\sum_{k=0}^{\mathfrak{a}-1} \binom{\mathfrak{a}}{k} (P_G(z) + z^q G_q(z))^k (P_R(z) + z^q R_q(z))^{\mathfrak{a}-k}\,, \\ G(z) = -z^p + z\,P_\phi\Big(P_G(z) + P_R(z) + z^q\big(G_q(z) + R_q(z)\big)\Big) \\ \qquad\quad + z(G(z) + R(z))^q \underline{\phi}_q(G(z) + R(z)) \\ \qquad\quad + z(P_G(z) + z^q G_p(z))^{\mathfrak{a}}\,. \end{cases} \tag{3.6}$$

En développant $(G(z) + R(z))^q$ avec la formule du binôme et en utilisant $R(z) = P_R(z) + z^q R_q(z)$, $G(z) = P_G(z) + z^q G_q(z)$, nous pouvons écrire $(G(z) + R(z))^q$ sous la forme $z^q P_1(z, G_p(z), R_p(z))$, avec $P_1(z, G_p, R_p)$ qui est un polynôme à coefficients positifs.

De même, $P_\phi\Big(P_G(z) + P_R(z) + z^q\big(G_q(z) + R_q(z)\big)\Big) + (P_G(z) + z^q G_q)^{\mathfrak{a}}$ est un polynôme en z, G_p, et R_p. Notons $z^q P_2(z, G_p, R_p)$ le polynôme obtenu à partir de celui-ci en ne gardant que les monômes dont le degré en z est supérieur à q. C'est un polynôme à coefficients positifs. Remarquons que $P_2(z, 0, 0) \neq 0$, car $P_G(z)$ est de degré q et $\mathfrak{a} \geqslant 2$, donc en développant $P_G(z)^{\mathfrak{a}}$, il y a au moins un monôme $z^{\mathfrak{a}q}$, avec $\mathfrak{a}q \geqslant q + 1$. De plus P_2 dépend bien de G_q et R_q car P_ϕ est un polynôme de degré au moins $\tilde{\mathfrak{a}} \geqslant 1$.

En identifiant les termes de degré plus grand que $q + 1$ en z, nous obtenons une équation de la forme :

$$z^q G_q(z) = z^q\big(zP_2(z, G_q(z), R_q(z)) + zP_1(z, G_q(z), R_q(z))F_1(z, G_q(z), R_q(z))\big)\,.$$

avec $F_1(z, G_q, R_q) = \underline{\phi}_q(P_G(z) + P_R(z) + z^q(G_q + R_q))$, qui est une série entière en z, G_q, R_q à coefficients positifs.

De même, en développant la somme $\sum_{k=0}^{\mathfrak{a}-1} \binom{\mathfrak{a}}{k}(P_G(z) + z^q G_q(z))^k (P_R(z) + z^q R_q(z))^{\mathfrak{a}-k}$, nous remarquons que pour $k = 0$, le développement de $(P_R(z) + z^q R_q(z))^{\mathfrak{a}}$ donne un terme $P_R(z)^{\mathfrak{a}}$ qui fournit le monôme $z^{\mathfrak{a}q}$, avec $\mathfrak{a}q \geqslant q + 1$.

Pour $k = \mathfrak{a} - 1$, le terme $z^{\mathfrak{a}q} G_q(z)^{\mathfrak{a}-1} R_q(z)$ dépend à la fois de G_q et R_q. Nous pouvons donc regrouper les termes en z^k avec $k \geqslant q > p$ de cette somme sous la forme $z^q P_3(z, G_q(z), R_q(z))$ avec $P_3(z, G_q, R_q)$ qui est un polynôme à coefficients positifs, qui dépend bien de G_q et R_q, et qui a le monôme $z^{(\mathfrak{a}-1)q}$ dans son développement.

En identifiant les termes de degré plus grand que $q+1$ en z, nous obtenons l'équation

$$z^q R_q(z) = z^{q+1} P_3(z, G_q(z), R_q(z))\,,$$

et ainsi $R_q(z)$ et $G_q(z)$ satisfont le système positif :

$$\begin{cases} R_q(z) = & zP_3(z, G_q(z), R_q(z)) \\ G_q(z) = & zP_2(z, G_q(z), R_q(z)) + zP_1(z, G_q(z), R_q(z)) \cdot F_1(z, G_q(z), R_q(z)) \end{cases} \tag{3.7}$$

avec P_1, P_2, P_3, F_1 qui sont à coefficients positifs. Comme $P_G(0) = P_R(0) = 0$, $F_1(z, G_q, R_q)$ est analytique en $(z, G_q, R_q) = (0,0,0)$ par composition. On pose

$$\boldsymbol{F}(z, G_p, R_p) = \begin{bmatrix} zP_3(z, G_q, R_q) \\ zP_2(z, G_q, R_q) + zP_1(z, G_q, R_q) \cdot F_1(z, G_q, R_q) \end{bmatrix}.$$

Par ce qui précède, $\boldsymbol{F}$ est analytique en $(0,0,0)$ et ses coefficients sont positifs. Par ailleurs, $\boldsymbol{F}(0, G_p, R_p) \equiv \mathbf{0}$ à cause du z en facteur, et $F_1(z,0,0) \neq 0$ et $F_2(z,0,0) \neq 0$ grâce au monôme $z^{(\mathfrak{a}-1)q}$ présent dans P_2 et P_3. Le système n'est pas linéaire car P_2 contient le monôme $z^{q(\mathfrak{a}-1)} G_q^{\mathfrak{a}}$ avec $\mathfrak{a} \geqslant 2$. Il est fortement connexe car P_2 et P_3 dépendent à la fois de R_q et G_q.

Enfin, nous rappelons que les coefficients des solutions $G_q(z)$ et $R_q(z)$ sont strictement positifs à partir d'un certain rang, car c'est le cas pour $G(z)$ et $R(z)$.

Nous pouvons enfin appliquer le théorème de Drmota : les fonctions $R_q(z)$ et $G_q(z)$ ont une unique singularité dominante $\tilde{\rho}$. Comme $R(z) = P_R(z) + z^q R_q(z)$, par le théorème de Pringsheim et le Lemme 3.8, nous en déduisons que $R(z)$ a une unique singularité dominante en $\tilde{\rho} = \rho$.

voir Théorème 1.89

De plus, $(\rho, G_q(\rho), R_q(\rho))$ est bien dans le domaine de convergence de $\boldsymbol{F}(z, G_q, R_q)$. En effet, les autres termes étant polynomiaux et ne posant pas de problème, toute la question repose sur $F_1(z, G_q, R_q) = \underline{\phi}_q(P_G(z) + P_R(z) + z^q(G_q + R_q))$. Or, nous avons l'égalité $P_G(\rho) + P_R(\rho) + \rho^q(G_q(\rho) + R_q(\rho)) = L(\rho) = \tau$, et par le *schéma d'inversion lisse*, τ est dans le disque de convergence de ϕ, donc de $\underline{\phi}_q$. Ainsi nous avons bien $\underline{\phi}_q(\tau) < +\infty$.

Donc toujours par le théorème de Drmota, au voisinage de $z = \rho$, $R_q(z) = g_q(z) - h_q(z)\sqrt{1 - \frac{z}{\rho}}$, avec $g_q(z), h_q(z)$ analytiques en ρ, et $h_q(\rho) \neq 0$

Ainsi au voisinage de $z = \rho$, la fonction $R(z)$ peut s'écrire $R(z) = g_R(z) - h_R(z)\sqrt{1 - \frac{z}{\rho}}$, où les fonctions $g_R(z)$ et $h_R(z)$ sont analytiques autour de $z = \rho$, et $h_R(\rho) = \rho^q h_q(\rho) \neq 0$. ■

Nous déduisons du comportement en racine carrée en $z = \rho$ que la proportion d'arbres entièrement réductibles tend vers une constante :

Corollaire 3.10 .
▶ La probabilité qu'une expression uniforme de $\mathcal{L}$ de taille n soit entièrement réductible tend vers une constante $\gamma > 0$ lorsque $n \to \infty$. ◀

Démonstration. Cette probabilité est donnée par la formule $\frac{[z^n]R(z)}{[z^n]L(z)}$. Nous appliquons alors le Théorème 1.81 de Transfert. La Proposition 3.9 nous indique que $R(z)$ a la même singularité ρ que $L(z)$, ainsi que le même exposant critique en racine carrée. Les termes en $\rho^n n^{-3/2}$ se simplifient dans la fraction. Donc la probabilité qu'une expression uniforme de $\mathcal{L}$ de taille n soit entièrement réductible tend vers une constante $\gamma > 0$ lorsque $n \to \infty$. ∎

3.3.2 Série bivariée et espérance de la taille après réduction

Notons $L(z,u) = \sum_{T\in\mathcal{L}} z^{|T|}u^{|\sigma(T)|}$ la série bivariée qui compte en u la taille des arbres après réduction. Nous obtenons alors facilement l'équation suivante pour $L(z,u)$, analogue au Lemme 2.39 pour les systèmes :

Lemme 3.11.
▶ La série $L(z,u)$ satisfait l'équation fonctionnelle

$$L(z,u) = (R(z) - z^p)u^p + zu\left(\underline{\phi}(L(z,u)) + \big(L(z,u) - u^p R(z)\big)^{\mathfrak{a}}\right), \quad (3.8)$$

où $\underline{\phi}(x) = \phi(x) - x^{\mathfrak{a}}$. ◀

Démonstration. Nous étudions tous les cas possibles, selon la racine des arbres considérés. Si la racine n'est pas $\circledast$, alors elle reste après la réduction, ce qui donne le terme $zu\underline{\phi}(L(z,u)$ dans la somme. Dans le cas où la racine de l'arbre est $\circledast$, alors soit l'arbre est entièrement réductible, ce qui contribue au terme $u^pR(z)$, soit il n'est pas entièrement réductible, auquel cas sa racine reste après réduction, et aucun de ses fils n'est entièrement réductible. Cela contribue au terme $(L(z,u) - u^pR(z))^{\mathfrak{a}}$. Enfin, le terme $-(zu)^p$ vient du fait que l'arbre $\mathcal{P}$ est compté deux fois dans ce raisonnement. ∎

Rappelons que l'espérance de la taille après réduction est donnée par la formule :

$$\mathbb{E}_n[|\sigma|] = \frac{[z^n]\partial_u L(z,u)\big|_{u=1}}{[z^n]L(z)}.$$

Nous cherchons donc classiquement à obtenir le comportement asymptotique de $[z^n]\partial_u L(z,u)\big|_{u=1}$, en étudiant la fonction $\partial_u L(z,u)\big|_{u=1}$ autour de sa singularité :

Lemme 3.12.
▶ La fonction $\partial_u L(z,u)\big|_{u=1}$ a une unique singularité dominante en $z=\rho$. De plus il existe deux fonctions $\tilde{g}(z)$ et $\tilde{h}(z)$, analytiques en $z=\rho$, telles que la fonction $\partial_u L(z,u)\big|_{u=1}$ s'écrive sous la forme $\partial_u L(z,u)\big|_{u=1} = \tilde{g}(z) - \tilde{h}(z)\sqrt{1-z/\rho}$ autour de $z=\rho$. ◀

Démonstration. En dérivant l'Équation 3.8 par rapport à u, nous obtenons :

$$\begin{aligned}\partial_u L(z,u)\big|_{u=1} &= pR(z) - pz^p + z\big(\phi(L(z)) - L(z)^{\mathfrak{a}} + (L(z)-R(z))^{\mathfrak{a}}\big)\\ &\quad + z\big((\phi'(L(z)) - \mathfrak{a}L(z)^{\mathfrak{a}-1})\partial_u L(z,u)\big|_{u=1}\\ &\quad + \mathfrak{a}(L(z)-R(z))^{\mathfrak{a}-1}(\partial_u L(z,u)\big|_{u=1} - pR(z))\big)\end{aligned}$$

Ainsi :

$$\partial_u L(z,u)\big|_{u=1} = \frac{p(R(z)-z^p) + L(z) + z\big(-L(z)^{\mathfrak{a}} + (L(z)-R(z))^{\mathfrak{a}} - p\mathfrak{a}(L(z)-R(z))^{\mathfrak{a}-1}R(z)\big)}{1 - z\left(\phi'(L(z)) - \mathfrak{a}L(z)^{\mathfrak{a}-1} + \mathfrak{a}(L(z)-R(z))^{\mathfrak{a}-1}\right)}.$$

Nous rappelons que la condition du *schéma d'inversion lisse* implique que $\tau = L(\rho)$ vérifie $1 = \rho\phi'(\tau)$, et que τ est dans le disque de convergence de ϕ. En particulier ϕ, et donc ϕ', sont analytiques en $\tau = L(\rho)$, et $\phi''(\tau) \neq 0$ car $\circledast$ est d'arité au moins 2.

Par ailleurs le dénominateur évalué en $z = \rho$ vaut

$$\rho\mathfrak{a}(\tau^{\mathfrak{a}-1} - (\tau - R(\rho))^{\mathfrak{a}-1}) > 0$$

qui est strictement positif car $\tau = L(\rho) > R(\rho) > 0$.

Rappelons que $L(z)$ et $R(z)$, par les Propositions 3.7 et 3.9, ont une unique singularité dominante en ρ et ont un comportement asymptotique en racine carrée autour de cette singularité. Nous pouvons alors appliquer les propriétés de clôture par somme, produit, composition et division du Lemme 2.45 des préliminaires au numérateur et dénominateur de $L(z,u)\big|_{u=1}$: la fonction $\partial_u L(z,u)\big|_{u=1}$ s'écrit alors sous la forme $\partial_u L(z,u)\big|_{u=1} = \tilde{g}(z) + \tilde{h}(z)\sqrt{1 - z/\rho}$ localement autour de $z = \rho$, avec $\tilde{g}(z)$ et $\tilde{h}(z)$ analytiques sur une voisinage de ρ.

Par ailleurs, la série $\partial_u L(z,u)\big|_{u=1}$ n'a aucune autre singularité dans le disque $|z| \leqslant \rho$. En effet, son rayon de convergence est ρ, puisque $[z^n]\{\partial_u L(z,u)\big|_{u=1}\} \leqslant n[z^n]L(z)$, donc ρ est une singularité. Comme $L(z)$, $R(z)$ et donc $\phi'(L(z))$ sont analytiques dans un voisinage de tout point z_0 du cercle $|z| = \rho$, tel que $z_0 \neq \rho$, il en est de même de $\partial_u L(z,u)\big|_{u=1}$. Donc $\partial_u L(z,u)\big|_{u=1}$ n'a pas d'autre singularité sur le cercle $|z| = \rho$. ■

Nous pouvons ainsi conclure par la proposition suivante, qui montre la moitié du théorème annoncé en début de section :

Proposition 3.13 .
▶ L'espérance de la taille, après simplification par la réduction σ, d'une expression uniforme de taille n tend vers une constante quand n tend vers l'infini :

$$\lim_{n\to\infty} \mathbb{E}_n[|\sigma|] = \delta\,,$$

avec $\delta > 0$. ◀

Démonstration. Le théorème de Transfert appliqué au résultat du Lemme 3.12 montre que, si $\tilde{h}(\rho) > 0$, alors $[z^n]\partial_u L(z,u)|_{u=1} \sim \frac{\tilde{h}(\rho)}{2\sqrt{\pi}}\rho^{-n}n^{-3/2}$. En combinant cela au Théorème 3.7, $\mathbb{E}_n[|\sigma|] = \frac{[z^n]\partial_u L(z,u)\big|_{u=1}}{[z^n]L(z)}$ tend vers une constante $\delta = \frac{\tilde{h}(\rho)}{2C_L\sqrt{\pi}}$, car $\rho = \rho_L$. Les autres valeurs pour le signe de $\tilde{h}(\rho)$ ne sont pas possibles : $\tilde{h}(\rho) < 0$ donnerait un équivalent asymptotique négatif, et $\tilde{h}(\rho) = 0$ impliquerait que la taille moyenne après réduction serait en $O(1/n)$, et donc tendrait vers 0, mais c'est impossible car cette taille moyenne est minorée par $p > 0$. ■

3.3.3 Convergence des moments d'ordre supérieur

Dans cette dernière section, nous montrons la convergence des moments d'ordre supérieur associés à la taille de la réduction.

Proposition 3.14 .
▶ Tous les moments de $|\sigma|$ convergent aussi : pour tout $i \in \mathbb{N}_{\geqslant 1}$, $\lim_{n\to\infty} \mathbb{E}_n[|\sigma|^i] = \delta_i$, avec $\delta_i > 0$. ◀

Démonstration. On montre le résultat pour $\sigma(\sigma-1)\dots(\sigma-k+1)$, car ces quantités sont reliées aux séries génératrices par la formule :

$$\lim_{n\to\infty} \mathbb{E}_n\Big[|\sigma|(|\sigma|-1)\dots(|\sigma|-k+1)\Big] = \frac{[z^n]\partial_u^k L(z,u)\big|_{u=1}}{[z^n]L(z)}\,.$$

Le résultat de la proposition suivra, car $|\sigma|^k$ peut s'écrire comme une combinaison linéaire d'éléments de la forme $|\sigma|(|\sigma|-1)\dots(|\sigma|-j+1)$ pour $j\leqslant k$.

Montrons que pour tout $k\in\mathbb{N}$, $\partial_u^k L(z,u)\big|_{u=1}$ est encore de la forme $\tilde{g}(z)+\tilde{h}(z)\sqrt{1-z/\rho}$ autour de $z=\rho$, avec $z\notin[\rho,\infty)$, tel que $\tilde{g}(z)$ et $\tilde{h}(z)$ soient analytiques en $z=\rho$. C'est le cas pour $k=0$ et $k=1$ (par le Théorème 3.7 et le Lemme 3.12).

En dérivant l'équation

$$L(z,u) = (R(z)-z^p)u^p + zu\big(\underline{\phi}(L(z,u)) + (L(z,u)-u^pR(z))^{\mathfrak{a}}\big), \qquad (3.8)$$

nous obtenons une expression de la forme :

$$\begin{aligned}\big(1 - zu\big(\phi'(L(z,u)) - \mathfrak{a}L(z,u)^{\mathfrak{a}-1} + \mathfrak{a}(L(z,u)-u^pR(z))^{\mathfrak{a}-1}\big)\big)\,\partial_u L(z,u)\\ = P_1\big(z,R(z),u,L(z,u),\phi(L(z,u))\big)\end{aligned}$$

où P_1 est un polynôme à coefficient entiers. En itérant ces dérivations, nous obtenons des expressions similaires pour les dérivées suivantes :

$$\begin{aligned}&\big(1 - zu\big(\phi'(L(z,u)) - \mathfrak{a}L(z,u)^{\mathfrak{a}-1} + \mathfrak{a}(L(z,u)-u^pR(z))^{\mathfrak{a}-1}\big)\big)\,\partial_u^k L(z,u)\\ &= P_k\big(z,R(z),u,L(z,u),\dots,\partial_u^{k-1}L(z,u),\phi(L(z,u)),\dots,(D_{k-1}\phi)(L(z,u))\big)\end{aligned}$$

où P_k est un polynôme à coefficients entiers.

En $u=1$, nous obtenons finalement :

$$\partial_u^k L(z,u)\big|_{u=1} = \frac{P_k\left(z,R(z),1,L(z,1),\dots,\partial_u^{k-1}L(z,u)|_{u=1},\phi(L(z)),\dots,(D_{k-1}\phi)(L(z)\right)}{1-z\left(\phi'(L(z))-\mathfrak{a}L(z)^{\mathfrak{a}-1}+\mathfrak{a}(L(z)-R(z))^{\mathfrak{a}-1}\right)}$$

La fonction $(1-z\big(\phi'(L(z))-\mathfrak{a}L(z)^{\mathfrak{a}-1}+\mathfrak{a}(L(z)-R(z))^{\mathfrak{a}-1}\big))^{-1}$ a déjà été étudiée dans le cas $k=1$, et nous avons prouvé qu'elle avait une unique singularité dominante en $z=\rho$, autour de laquelle elle admettait une expansion en racine carrée.

Par hypothèse de récurrence, tous les termes $\partial_u^j L(z,u)|_{u=1}$ pour $j\leqslant k-1$ ont aussi une unique singularité dominante en $z=\rho$, et une expansion en racine carrée. De plus $L(\rho)$ se trouve dans le rayon de convergence de ϕ, donc ϕ et toutes ses dérivées sont analytiques dans un voisinage de ρ.

Nous concluons en appliquant les propriétés de clôture par somme, produit, composition et division du Lemme 2.45 au numérateur et dénominateur de $\partial_u^k L(z,u)\big|_{u=1}$. Nous trouvons alors que la fonction $\partial_u^k L(z,u)\big|_{u=1}$ s'écrit sous la forme $\partial_u^k L(z,u)\big|_{u=1} = \tilde{g}_k(z)+\tilde{h}_k(z)\sqrt{1-z/\rho}$ localement autour de $z=\rho$, et que ρ est son unique singularité dominante.

Ainsi, en appliquant le théorème de Transfert, nous démontrons comme à la Proposition 3.13 que les termes qui dépendent de n se simplifient dans l'asymptotique de $\frac{[z^n]\partial_u^k L(z,u)\big|_{u=1}}{[z^n]L(z)}$. Ainsi les moments convergent vers une constante. ∎

3.4 Conclusion et expérimentation sur les langages réguliers

Dans ce chapitre, nous avons adapté les résultats du chapitre précédent au cas des expressions spécifiées par une équation analytique unidimensionnelle. Il s'agit à la fois d'une restriction et d'une généralisation des résultats du chapitre précédent, qui justifient un traitement à part, dans un chapitre dédié, même si les techniques sont similaires. Remarquons que si dans ce chapitre nous avons pu montrer la convergence des moments vers une constante (dans le chapitre précédent nous avions juste montré qu'ils étaient bornés par une constante), cela revient au même vis-à-vis de la dégénérescence de la distribution uniforme pour les arbres d'expressions. En effet, la réduction considérée ne prend pas en compte toutes les simplifications et équivalences sémantiques possibles, donc les tailles moyennes obtenues dans ce chapitre et au chapitre précédent sont des bornes supérieures des "vraies" tailles des expressions tirées au hasard.

Comme transition pour le chapitre suivant, nous développons le cas des expressions régulières aléatoires de taille n suivant la spécification :

$$\mathcal{L}_{\mathcal{R}} = a + b + \varepsilon + \overset{\star}{|}_{\mathcal{L}_{\mathcal{R}}} + \overset{\bullet}{/\!\backslash}_{\mathcal{L}_{\mathcal{R}}\ \mathcal{L}_{\mathcal{R}}} + \overset{+}{/\!\backslash}_{\mathcal{L}_{\mathcal{R}}\ \mathcal{L}_{\mathcal{R}}}.$$

Alors, par la Proposition 3.13, la taille moyenne après réduction des arbres originellement de taille n tend vers une constante lorsque n tend vers l'infini.

Nous avons voulu observer ce phénomène expérimentalement. Nous avons tiré des expressions aléatoires de taille n de plus en plus grande, puis nous avons calculé la moyenne de leurs tailles après réduction. Nous nous attendions alors à pouvoir observer la convergence, en allant assez loin dans les tailles des arbres générés.

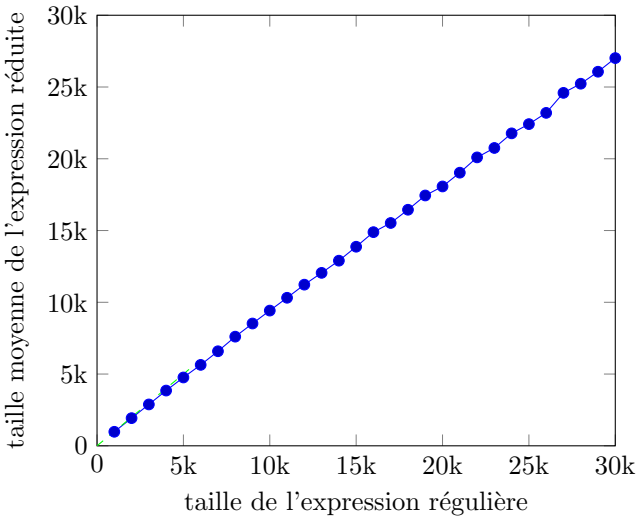


Figure 3.1. Tentative d'observation expérimentale de la convergence en moyenne

En utilisant la méthode récursive, nous avons ainsi fait l'expérience (voir Figure 3.1), pour des arbres de taille jusqu'à 30 000. La courbe observée semble linéaire en n, et absolument pas converger vers une constante ! En allant un peu plus loin dans la taille des arbres générés, en utilisant la méthode de Devroye [Dev12], nous pouvons voir que la courbe se détache un peu de la droite linéaire (cf Figure 3.2), sans observer

Je ne pouvais raisonnablement aller plus loin que 30 000 avec la méthode récursive sur mon ordinateur personnel.

cependant une convergence nette. Dans cette section, nous expliquons rapidement ce phénomène.

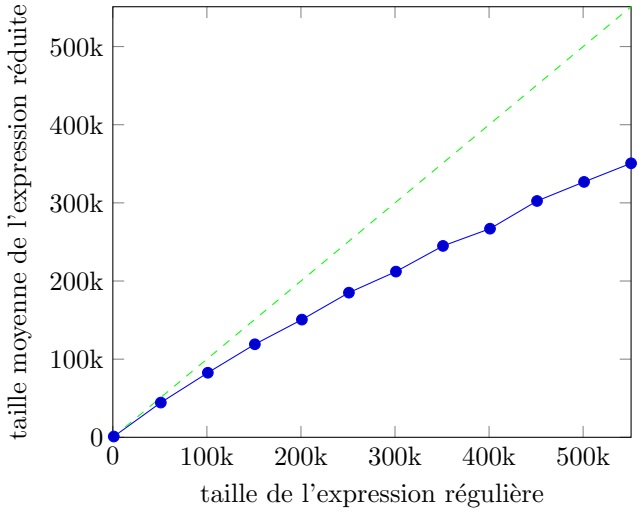


Figure 3.2. Tentative bis d'observation expérimentale de la convergence en moyenne

Les propositions démontrées aux sections précédentes nous disent que toutes les fonctions d'intérêt ont une unique singularité dominante en $\rho = \rho_L$, et ont un comportement de la forme $g(z) - h(z)\sqrt{1 - z/\rho}$, avec $g(z), h(z)$ analytiques en ρ. Or, si $L(z) = g(z) - h(z)\sqrt{1 - z/\rho}$, alors $L'(z) = g'(z) - h'(z)\sqrt{1 - z/\rho} + \frac{h(z)}{2\rho\sqrt{1-z\rho}}$, et ainsi :

$$\lim_{z\to\rho} L'(z)\sqrt{1 - z/\rho} = \frac{h(\rho)}{2\rho}\,.$$

On utilisera cette égalité pour calculer numériquement la constante δ annoncée.

La série génératrice des expressions régulières vérifie l'équation $L(z) = 3z + zL(z) + 2zL(z)^2$, ce qui donne après résolution :

$$L(z) = \frac{1 - z - \sqrt{-23\,z^2 - 2\,z + 1}}{4z}$$

Sa singularité dominante est en $\rho = \frac{-1+2\sqrt{6}}{23}$, la taille de $\mathcal{P}$ est $p = 4$, $\mathfrak{a} = 2$, enfin $\phi(y) = 3 + y + 2y^2$, et $\underline{\phi}(y) = 3 + y + y^2$.

Les séries $R(z)$ et $G(z)$ satisfont le système :

$$\begin{cases} R(z) = & z^4 + z\Big(L(z)^2 - (G(z))^2\Big)\,, \\ G(z) = & -z^4 + z\,\underline{\phi}\big(L(z)\big) + zG(z)^2\,. \end{cases} \tag{3.9}$$

Nous pouvons utiliser ce système (qui n'est pas positif) pour les calculs car nous sommes assurés par la théorie que ces fonctions ont bien le bon comportement en racine carrée autour de ρ. Il est possible en fait de résoudre $G(z)$ explicitement :

$$G(z) = \frac{1 - \sqrt{4\,z^5 + 1 - 4z^2(L(z)^2 + L(z) + 3)}}{2z}$$

si bien que nous connaissons aussi $R(z) = L(z) - G(z)$. Ainsi la formule suivante donne une expression explicite exacte pour $\partial_u L(z,u)\big|_{u=1}$:

$$\partial_u L(z,u)\big|_{u=1} = \frac{4(R(z)-z^4) + L(z) + z\big(-L(z)^2 + (L(z)-R(z))^2 - 8(L(z)-R(z))R(z)\big)}{1 - z\left(\phi'(L(z)) - 2L(z) + 2(L(z)-R(z))\right)}.$$

Nous savons que $\partial_u L(z,u)\big|_{u=1}$ a une unique singularité dominante en $z=\rho$ et est de la forme $\tilde{g}(z) - \tilde{h}(z)\sqrt{1-z\rho}$ autour de ρ, nous pouvons alors calculer avec Maple la valeur de $\tilde{h}(\rho)$. Par le théorème de Transfert, la taille moyenne, après réduction, des arbres de taille n tend vers la constante suivante lorsque n tend vers l'infini :

$$\delta = \frac{\tilde{h}(\rho)}{h(\rho)} \approx 3\,624\,217\,.$$

Cette constante est très grande ! C'est pour cela que nous ne pouvions pas l'observer expérimentalement avec des arbres de taille 30 000 seulement.

Ce calcul numérique montre les limites du résultat de ce chapitre (et du précédent) : nous avons délibérément choisi une réduction très simple, pour nous abstraire au maximum de la sémantique des arbres étudiés, afin d'obtenir un résultat général qui s'applique à de nombreuses classes d'expressions. Comme contrepartie de sa généralité, notre résultat ne permet pas d'analyser suffisamment finement la sémantique des arbres d'expression étudiés pour obtenir des constantes "petites".

Dans le cas des expressions régulières, nous pouvons remarquer par exemple que $(b+a)^*$ est aussi absorbant pour l'union. Notre réduction ne le prend pas en compte, car elle étudie un seul élément absorbant. Pourtant il existe de nombreuses autres expressions régulières équivalentes à $(a+b)^*$ faciles à détecter.

Par exemple, en parcourant une expression, nous pouvons détecter pour tout sous-arbre s'il reconnaît les lettres a, b ou encore le mot vide ε. Nous pouvons alors identifier un plus grand nombre de sous-arbres universels : si un arbre qui reconnaît à la fois les lettres a et b est mis à l'étoile, il est universel. De plus un arbre universel peut être considéré comme absorbant pour la concaténation si l'arbre en face reconnaît ε. En appliquant ces règles à des arbres uniformes, nous avons observé expérimentalement une convergence apparente vers une constante de l'ordre de 75 (voir Figure 3.3).

Le but du chapitre suivant est d'ajouter ces règles plus fines de réduction dans le cas des expressions régulières, afin de faire diminuer cette constante et de pouvoir observer expérimentalement la dégénérescence de la distribution uniforme pour les expressions régulières.

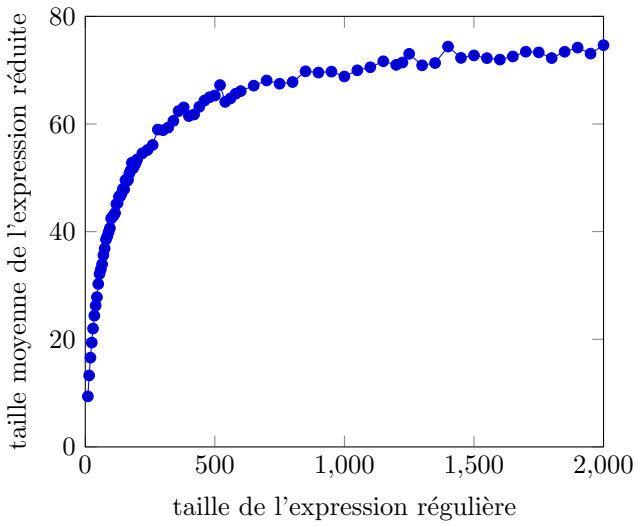


Figure 3.3. Observation expérimentale de la convergence, dans le cas des expressions régulières, en ajoutant plus de règles de réduction

Ce chapitre est tiré de l'article de conférence [KR21b].

4

Détection heuristique de sous-arbres universels pour les expressions régulières uniformes

4.1 Introduction

Les langages réguliers sont omniprésents en informatique : pour vérifier le bon formatage des données entrées dans un formulaire, pour rechercher/remplacer dans un éditeur de texte, rechercher dans un fichier avec `grep`... On les décrit alors naturellement par une expression régulière. Ainsi de nombreux programmes prennent en entrée une expression régulière, et il existe différents outils pour les traiter. Nous pouvons notamment citer les algorithmes de compilation d'une expression régulière en automate, comme la construction de Thompson, l'automate de Glushkov, l'automate des dérivées d'Antimirov, l'automate des préfixes de Yamamoto [AM06, Ant95, Yam14].

Si les arbres d'expressions régulières ont été utilisés avec succès pour étudier certains algorithmes de construction d'automates [BMMR12], ces arbres présentent néanmoins un élément absorbant pour l'opérateur d'union $+$, qui est $(a + b)^*$ pour un alphabet à deux lettres. Ainsi, le résultat principal des deux chapitres précédents montre que, pour la distribution uniforme, la complexité moyenne des algorithmes précédemment cités est asymptotiquement linéaire, si l'on commence par réduire l'expression régulière en entrée. Par ailleurs, nous avons montré au chapitre précédent que dans le cas d'un alphabet à deux lettres, une expression aléatoire uniforme de taille n est équivalente à une expression de taille constante $3\,624\,217$ en moyenne, quand n tend vers l'infini.

Cette taille moyenne de trois millions semble bien grande. Ceci est dû au fait que la réduction appliquée est à la fois très simple et très générale, dans le sens où elle s'applique à tout ensemble d'arbres dont la sémantique admet un élément absorbant. Cette constante ne se voit pas facilement expérimentalement, et on serait tenté d'objecter alors que pour des petites tailles d'arbres, la distribution uniforme pourrait encore être pertinente. Le but de ce chapitre est de spécialiser la réduction

des précédents chapitres au cas particulier des expressions régulières, pour ainsi faire baisser cette constante théorique. En prenant mieux en compte la sémantique particulière des expressions étudiées – ici les expressions régulières – nous allons pouvoir généraliser la réduction des chapitres précédents et obtenir des résultats plus fins sur la taille après réduction (cette démarche s'est avérée fructueuse par exemple pour étudier la distribution des expressions booléennes [CGM11, Gar05]). Ce chapitre peut être vu comme une application pratique et numérique aux expressions régulières des techniques développées aux précédents chapitres.

Le point de départ pour améliorer la réduction vient des deux observations suivantes sur la réduction étudiée jusqu'alors :

— plutôt que de reconnaître un seul élément absorbant $\mathcal{P}$ fixé à l'avance, nous pourrions augmenter la réduction en identifiant un sous-ensemble plus gros (infini) d'éléments absorbants de base, pour une opération $\circledast$ donnée ;

— nous n'exploitons pas du tout l'interaction des éléments absorbants avec les autres opérateurs, qui sont tous considérés par défaut comme bloquant la propagation des réductions par élément absorbant.

Dans ce chapitre nous étudions donc des règles de simplification spécifiques aux expressions régulières ; ces règles ont pour objectif d'identifier et réduire certains sous-arbres universels, qui acceptent tous les mots sur l'alphabet. Cette réduction linéaire généralise celle par élément absorbant des chapitres précédents. L'idée principale est d'identifier pour chaque nœud de l'arbre le sous-ensemble de $\Sigma \cup \{\varepsilon\}$ qui est reconnu par la sous-expression correspondante (en bleu sur la figure 4.1). En effet :

— l'expression T^* est universelle pour toute expression T qui reconnaît toutes les lettres de Σ ; il s'agit de notre ensemble infini d'éléments absorbants de base

— si une expression T est identifiée comme universelle, alors T^* est aussi universelle, l'union de cette expression avec n'importe quelle autre expression est aussi universelle, et enfin sa concaténation avec une expression qui reconnaît le mot vide est aussi universelle ; on prend ainsi en compte l'interaction des éléments absorbants avec les autres opérateurs.

Dans la figure 4.2, nous donnons les règles de réduction utilisées pour reconnaître l'universalité. Lorsque la réduction détecte qu'une expression est universelle [1], elle la remplace par un arbre fixé $\mathcal{U}$ représentant le langage universel Σ^*. Typiquement $\mathcal{U}$ est un arbre universel de taille minimale (l'arbre associé à $(a+b)^\star$ pour deux lettres).

Il est aussi possible d'introduire $\mathcal{U}$ comme un nouveau symbole spécial, de taille 1.

Dans ce chapitre, nous allons prouver que pour cet algorithme de réduction, toute expression régulière aléatoire uniforme de taille n se réduit en une expression de taille constante en moyenne lorsque $n \to \infty$; nous montrons par ailleurs que cette constante est suffisamment petite pour remettre en question la distribution uniforme des expressions régulières, même sur des petites tailles. Nous donnons pour $k = |\Sigma|$ allant de 2 à 5 les valeurs des constantes en question, dans la Table 4.1 (nous avons pris pour $\mathcal{U}$ un arbre universel minimal de taille $2k$). Dans le cas particulier de deux lettres, la constante ~ 77.797 est étonnamment petite, et est confirmée par nos expériences (voir Fig. 4.3).

Pour les simulations, les arbres uniformes ont été générés avec l'algorithme de [Dev12].

Le résultat principal de ce chapitre est le suivant :

1. Comme le problème de l'universalité est PSPACE-complet [MS72], cette détection en un temps linéaire est forcément partielle.

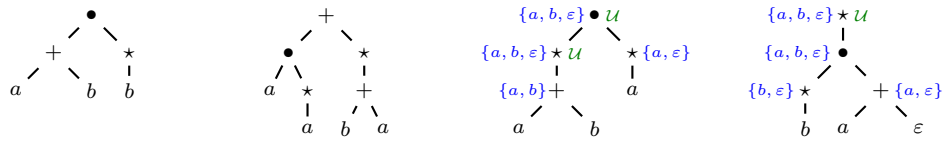


Figure 4.1. Quatre arbres d'expressions régulières. Les trois dernières expressions sont universelles – elles reconnaissent tous les mots sur l'alphabet $\{a, b\}$. Notre algorithme de simplification (voir la figure 4.2) sera capable de le détecter. Par exemple, pour les deux derniers arbres, les sous-ensembles de symboles reconnus parmi $\{a, b, \varepsilon\}$ apparaissent en bleu ; nous avons aussi annoté en vert si l'expression associée à un nœud est détectée comme étant universelle.

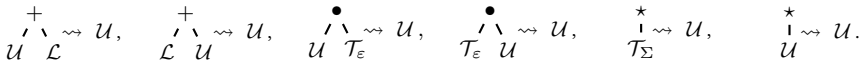


Figure 4.2. L'ensemble des règles de réductions, à appliquer de bas en haut. Ici $\mathcal{U}$ est un arbre spécial fixé, représentant les arbres identifiés comme étant universels ; la classe $\mathcal{L}$ désigne tous les arbres, $\mathcal{T}_\Sigma$ les arbres reconnaissant toutes les lettres de l'alphabet Σ, et $\mathcal{T}_\varepsilon$ représente la classe des arbres reconnaissant le mot vide ε. Par exemple, $\mathcal{U} \in \mathcal{T}_\varepsilon \cap \mathcal{T}_\Sigma$. La dernière règle est donc redondante, mais a été ajoutée pour améliorer la compréhension de la logique des règles de réécriture.

Théorème 4.1.
▶ Considérons l'ensemble des arbres d'expressions régulières pour un alphabet de taille fixée $|\Sigma| = k$, et σ la réduction en temps linéaire induite par les règles de simplification de la Figure 4.2. Alors la taille réduite d'une expression aléatoire uniforme de taille n tend vers une constante quand n tend vers l'infini, qui est donnée dans le tableau suivant pour k allant de 2 à 5 :

$\lvert\Sigma\rvert$	2	3	4	5
$\lim \mathbb{E}_n[\lvert\sigma(T)\rvert]$	77.79724…	495.59151…	2 518.20513…	11 694.43727…

Table 4.1. Tailles moyennes asymptotiques après réduction

◀

Le cheminement de la preuve nous permet par ailleurs de fournir un encadrement asymptotique de la proportion d'arbres uniformes universels. Dans la Proposition 4.24, nous montrons qu'il y a ainsi une proportion non négligeable d'arbres qui représentent le langage universel. En particulier cette proportion est comprise entre 31% et 46% pour un alphabet de deux lettres.

Remarque 4.2 (Autres réductions de la littérature).
▶ Il existe d'autres réductions dans la littérature [BK93, GG10, IY03, LS05] dont l'objectif est de réécrire les expressions sous une forme normale, afin de limiter certaines redondances, et de les transformer ensuite en automates. Ces réductions ont un réel intérêt pratique car elles s'intéressent à des expressions régulières de la vraie vie, écrites par des humains, et qui ont donc peu de chances d'avoir autant de redondances qu'une expression typique uniforme.

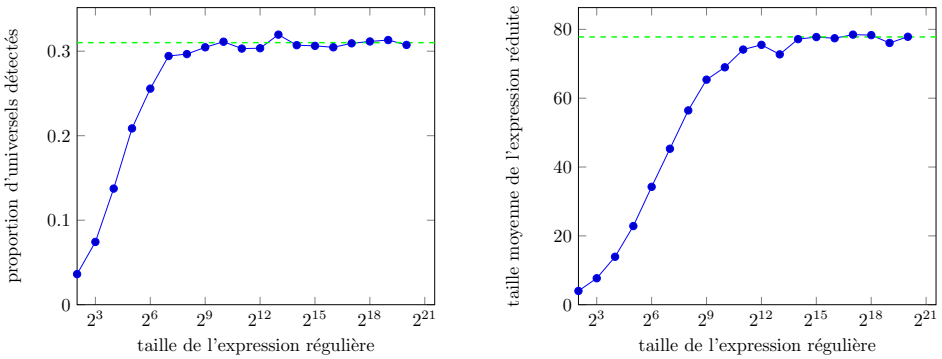


Figure 4.3. La proportion d'expressions universelles détectées par l'algorithme, et la taille moyenne après réduction, observées expérimentalement sur des expressions régulières à deux lettres, avec un échantillon de 10 000 expressions pour chaque taille. L'échelle des courbes est logarithmique en la taille des arbres. Les limites théoriques sont représentées en vert.

La réduction que nous étudions dans ce chapitre, quant à elle, a pour but d'illustrer la dégénérescence de la distribution uniforme sur les expressions régulières ; son objectif en soi n'est donc pas tant de réduire intelligemment les expressions, mais de montrer qu'au contraire de simples réductions réduisent beaucoup trop les expressions uniformes. Les deux démarches se rejoignent cependant dans le sens où nous montrons que l'étude en moyenne des réductions de la littérature n'a *a priori* pas d'intérêt si la distribution utilisée est uniforme. ◀

Plan du chapitre. Dans la Section 4.2, nous introduisons la réduction générale que nous étudions, puis dans la Section 4.3 nous analysons avec les outils de la combinatoire analytique le phénomène de réduction sur des arbres d'expressions régulières uniformes. Enfin, dans la Section 4.4, nous expliquons comment calculer en pratique les constantes annoncées à la section précédente.

4.2 Modèle et définitions

4.2.1 Définitions

Soit $k \geqslant 1$ un entier, et $\Sigma = \{a_1, \dots, a_k\}$ un alphabet de taille k. On considère les arbres d'expressions régulières définis par la syntaxe suivante :

Définition 4.3 .
▶ La classe $\mathcal{L}_\mathcal{R}$ des arbres d'expressions régulières est définie inductivement par l'équation

$$\mathcal{L}_\mathcal{R} = a_1 + \ldots + a_k + \varepsilon + \overset{\star}{|}_{\mathcal{L}_\mathcal{R}} + \overset{\bullet}{/\backslash}_{\mathcal{L}_\mathcal{R}\ \mathcal{L}_\mathcal{R}} + \overset{+}{/\backslash}_{\mathcal{L}_\mathcal{R}\ \mathcal{L}_\mathcal{R}} . \tag{4.1}$$

Autrement dit avec les notations du chapitre 2, $\mathcal{L}_\mathcal{R} = \mathcal{T}(S, a)$ où $S = \Sigma \cup \{\varepsilon, +, \bullet, \star\}$, et $a(\Sigma) = a(\varepsilon) = 0$, $a(\star) = 1$, et $a(+) = a(\bullet) = 2$. ◀

On rappelle que la taille $|T|$ d'un arbre $T \in \mathcal{L}_\mathcal{R}$ est son nombre de nœuds. On note $\mathcal{L}_n$ l'ensemble des arbres de $\mathcal{L}_\mathcal{R}$ de taille $n \in \mathbb{N}$, et $\ell_n = \mathrm{card}(\mathcal{L}_n)$.

On note $L(z) = \sum_n \ell_n z^n$ la série associée à $\mathcal{L}_\mathcal{R}$, qui vérifie alors l'équation :

$$L(z) = (k+1)z + zL(z) + 2z(L(z))^2\,, \tag{4.2}$$

et donc

$$L(z) = \frac{1 - z - \sqrt{\Delta(z)}}{4z}$$

avec $\Delta(z) := -(8k+7)\,z^2 - 2\,z + 1$. Ainsi $L(z)$ présente une fausse singularité en $z = 0$, et son unique singularité dominante ρ est caractérisée par

$$L(\rho) = \sqrt{\frac{1+k}{2}}\,, \qquad \rho = \frac{1}{1 + 4L(\rho)}\,.$$

On en déduit que $L(z) = h_L - g_L\sqrt{1 - z/\rho} + O\left(\left|1 - \frac{z}{\rho}\right|\right)$ lorsque $z \to \rho$, avec $h_L = L(\rho)$, et g_L une constante qui vérifie $g_L = 2\rho \lim_{z\to\rho} L'(z) \cdot \sqrt{1 - z/\rho}$. Enfin par le Théorème 1.81 de Transfert nous obtenons :

$$\ell_n = [z^n]L(z) \sim \frac{g_L}{2\sqrt{\pi}}\rho^{-n}n^{-3/2}\,,$$

avec $g_L = \frac{\sqrt{8k+7}\,\sqrt[4]{2+2k}}{2\sqrt{-1+2\sqrt{2+2k}}}$.

Probabilité de reconnaître ε. La classe $\mathcal{T}_\varepsilon$ des expressions reconnaissant le mot vide ε est fondamentale pour définir et analyser la nouvelle réduction. Il se trouve que l'on peut calculer facilement sa série génératrice. En effet, une expression reconnaît le mot vide ε si et seulement si on est dans l'un des cas suivants :

— c'est la feuille ε ;
— c'est l'étoile d'une expression quelconque ;
— c'est la concaténation de deux expressions régulières qui reconnaissent ε ;
— c'est l'union de deux expressions dont une au moins reconnaît le mot vide.

La classe $\mathcal{T}_\varepsilon$ vérifie donc l'équation suivante :

$$\mathcal{T}_\varepsilon = \varepsilon + \overset{\star}{|}_{\mathcal{L}_\mathcal{R}} + \overset{\bullet}{/\!\backslash}_{\mathcal{T}_\varepsilon\ \mathcal{T}_\varepsilon} + \overset{+}{/\!\backslash}_{\mathcal{T}_\varepsilon\ \mathcal{L}_\mathcal{R}} + \overset{+}{/\!\backslash}_{\mathcal{L}_\mathcal{R}\ \mathcal{T}_\varepsilon}\,,$$

où chaque somme est disjointe. Sa série génératrice $T_\varepsilon(z)$ satisfait donc l'équation linéaire suivante :

Les termes en $T_\varepsilon(z)^2$ s'annulent.

$$T_\varepsilon(z) = z + zL(z) + 2zL(z)T_\varepsilon(z)\,. \tag{4.3}$$

Nous pouvons alors résoudre cette équation linéaire $T_\varepsilon(z) = \frac{z+zL(z)}{1-2zL(z)}$ et vérifier ainsi que $T_\varepsilon(z) = h_{T_\varepsilon} - g_{T_\varepsilon}\sqrt{1 - z/\rho} + O\left(1 - z/\rho\right)$ lorsque $z \to \rho$, avec une constante $g_{T_\varepsilon} \neq 0$. Ainsi le nombre d'expressions de taille n reconnaissant ε est asymptotiquement

$$[z^n]T_\varepsilon(z) \sim g_{T_\varepsilon}\rho^{-n}n^{-3/2}/(2\sqrt{\pi})\,.$$

En normalisant par ℓ_n, nous obtenons la proportion asymptotique d'expressions reconnaissant le mot vide :

$k = \lvert\Sigma\rvert$	2	3	4	5
g_{T_ε}/g_L	$0.777\ldots$	$0.663\ldots$	$0.590\ldots$	$0.538\ldots$

Table 4.2. Les probabilités asymptotiques de reconnaître ε pour une expression régulière uniforme.

Proposition 4.4.
▶ La probabilité qu'une expression régulière aléatoire uniforme de taille n reconnaisse le mot vide converge lorsque n tend vers l'infini vers une constante positive $g_{T_\varepsilon}/g_L = \frac{\sqrt{2k+2}+3/2}{k+\sqrt{2k+2}+3/2}$, dont les premières valeurs sont données dans le Tableau 4.2. ◀

Démonstration. Il suffit d'injecter $L(z) = h_L - g_L\sqrt{1-z/\rho} + O\left(\left|1-\frac{z}{\rho}\right|\right)$ dans $T_\varepsilon(z) = \frac{z+zL(z)}{1-2zL(z)}$.

En développant $(1-2zL(z))^{-1} = (1-2\rho h_L)^{-1}(1-\frac{2zg_L}{1-2zh_L}\sqrt{1-z/\rho}+O(1-z/\rho))$, nous obtenons finalement que $g_{T_\varepsilon}/g_L = \frac{4\rho(1+2\rho)}{(1+\rho)^2}$. Il suffit ensuite de remplacer ρ par sa valeur pour obtenir le résultat. On remarque qu'on n'a pas besoin de connaître la valeur de g_L pour calculer la proportion. ■

4.2.2 Le nouvel algorithme de simplification

Nous considérons désormais une classe d'expressions, notée $\mathcal{R}$, qui rassemble certains arbres qui reconnaissent tous les mots de $\Sigma^\star$:

Définition 4.5.
▶ La classe $\mathcal{R}$ est définie de façon inductive :

— si T reconnaît toutes les lettres de l'alphabet, alors $\overset{\star}{|}_T \in \mathcal{R}$;

— si l'un au moins des arbres T_1 ou T_2 est dans $\mathcal{R}$, alors $\overset{+}{\wedge}_{T_1\ T_2} \in \mathcal{R}$;

— si $T_1 \in \mathcal{R}$ et $T_2 \in \mathcal{T}_\varepsilon$, alors $\overset{\bullet}{\wedge}_{T_1\ T_2} \in \mathcal{R}$ et $\overset{\bullet}{\wedge}_{T_2\ T_1} \in \mathcal{R}$.

En particulier, si $T \in \mathcal{R}$, alors $\overset{\star}{|}_T \in \mathcal{R}$. ◀

La classe $\mathcal{R}$ désigne en fait la classe des sous-arbres qui vont être réduits à $\mathcal{U}$ par le nouvel algorithme de simplification. On remarque que l'arbre associé à l'expression $\Sigma \cdot \Sigma^\star + \varepsilon$ n'appartient par à $\mathcal{R}$, malgré son universalité. Cette classe ne désigne donc pas toutes les expressions universelles, ce qui n'est pas étonnant étant donné que déterminer si un arbre est dans la classe $\mathcal{R}$ se fait en temps linéaire, tandis que décider si un arbre est universel est un problème PSPACE-complet [MS72].

Reconnaissance des lettres. Pour déterminer si un arbre appartient à $\mathcal{R}$, on doit pouvoir décider quelles lettres sont reconnues par un arbre. Comme pour $\mathcal{R}$, cela se fait par induction en remontant l'arbre depuis les feuilles. Soit $a \in \Sigma$ une lettre arbitraire :

— si $|T| = 1$, alors T reconnaît a si et seulement si $T = a$,

— si la racine de T est $\star$, alors T reconnaît a si et seulement si le fils de la racine reconnaît la lettre a,

— si la racine de T est $+$, alors T reconnaît a si et seulement si l'un des fils de la racine reconnaît a,

— si la racine de T est $\bullet$, alors T reconnaît a si et seulement si l'un des fils de la racine reconnaît a et l'autre reconnaît le mote vide ε.

Exemple 4.6 .

▶ Si T_1 reconnaît ε et a, tandis que T_2 reconnaît ε et b, alors $\overset{\bullet}{\underset{T_1}{/}\underset{T_2}{\backslash}}$ reconnaît a, b et ε. ◀

Nous pouvons maintenant définir mathématiquement la réduction σ considérée :

Définition 4.7 (Algorithme de réduction σ)**.**

▶ Soit T un arbre d'expression régulière, et $\mathcal{U}$ un arbre fixé représentant $\Sigma^\star$. La réduction de T, notée $\sigma_{\mathcal{U}}(T)$, est l'arbre d'expression régulière obtenu par l'algorithme de réduction informel suivant : en partant des feuilles de l'arbre, on détermine en remontant les branches :

a. quelles lettres de Σ sont reconnues par chaque sous-arbre de T,

b. quels sous-arbres de T reconnaissent le mot vide ε,

c. quels sous-arbres de T sont dans la classe $\mathcal{R}$.

Ces trois informations se calculent directement en remontant les branches, comme nous l'avons expliqué précédemment. À chaque fois qu'un sous-arbre est identifié dans la classe $\mathcal{R}$, on le remplace par l'arbre constant $\mathcal{U}$. Nous notons $\sigma_{\mathcal{U}}(T)$ l'arbre obtenu à la fin de l'algorithme.

Quand le contexte est clair sur la valeur de $\mathcal{U}$, nous notons simplement $\sigma(T)$. Le pseudo-code de l'algorithme est donnée dans l'Algorithme 2. ◀

4.2.3 Série génératrice associée à la réduction

Nous considérons la série génératrice *bivariée* qui compte en z la taille d'une expression, et en u la taille de sa réduction :

$$L(z,u) = \sum_{T\in\mathcal{L}_{\mathcal{R}}} z^{|T|}u^{|\sigma(T)|}\,. \tag{4.4}$$

Nous rappelons que la taille moyenne après réduction d'un arbre uniforme de taille n est donnée par la formule :

$$\mathbb{E}_n[|\sigma|] = \frac{[z^n]\partial_u L(z,u)\big|_{u=1}}{[z^n]L(z)}\,. \tag{4.5}$$

Pour calculer cette moyenne, nous devons donc déterminer $L(z,u)$. Pour cela nous allons partitionner $\mathcal{L}$ en plusieurs classes, correspondant aux différentes étapes de l'algorithme d'identification des arbres qui sont dans $\mathcal{R}$. Cela nous permettra notamment de déterminer la taille des arbres après réduction (voir Section 4.3.1).

4.3 Caractérisation analytique de la limite

Le but de cette section est de montrer que l'espérance de la taille après réduction d'un arbre uniforme de taille n converge lorsque n tend vers l'infini. Pour le montrer,

```
1  function reduce(T) :
2  if |T| = 1 then
3  |  return (T, {T});
4  if T = +(T_L, T_R) then
5  |  (T'_L, S_L) := reduce(T_L);
6  |  (T'_R, S_R) := reduce(T_R);
7  |  if T'_L = U or T'_R = U then
8  |  |  return (U, S_L ∪ S_R) ;
9  |  return (+(T'_L, T'_R), S_L ∪ S_R);
10 else if T = •(T_L, T_R) then
11 |  (T'_L, S_L) := reduce(T_L);
12 |  (T'_R, S_R) := reduce(T_R);
13 |  S := ∅;
14 |  if ε ∈ S_R then
15 |  |  if T'_L = U then
16 |  |  |  return (U, S_L) ;
17 |  |  S := S ∪ S_L ;
18 |  if ε ∈ S_L then
19 |  |  if T'_R = U then
20 |  |  |  return (U, S_R) ;
21 |  |  S := S ∪ S_R ;
22 |  return (•(T'_L, T'_R), S);
23 else if T = *(T_0) then
24 |  (T', S') := reduce(T_0);
25 |  if Σ ⊆ S' then
26 |  |  return (U, {ε} ∪ Σ) ;
27 |  return (*(T'), {ε} ∪ S');
```

Algorithme 2 : Pseudo-code de l'algorithme de réduction sur l'alphabet Σ. L'algorithme renvoie un couple $\texttt{reduce}(T) = (\sigma(T), S_T)$, où $\texttt{reduce}(T)$ correspond à la réduction de l'arbre T donné en entrée, et $S_T \subseteq \{\varepsilon\} \cup \Sigma$ représente les feuilles reconnues par T.

on étudie les propriétés analytiques de $\partial_u L(z,u)|_{u=1}$. Comme dans les précédents chapitres, nous allons montrer que cette fonction présente une unique singularité dominante en $z=\rho$ avec un comportement en racine carrée. Remarquons que la convergence n'est pas une conséquence directe des résultats principaux des chapitres précédents, car la réduction est plus compliquée qu'un simple élément absorbant (la taille pourrait osciller par exemple sans converger). Néanmoins cette section n'introduit pas de nouveauté technique : nous allons simplement réutiliser les outils sur les systèmes développés aux chapitres précédents. La contribution principale de cette partie consiste majoritairement à identifier la partition de $\mathcal{L}$ qui permette de décrire $L(z,u)$ pour cette réduction.

Les propriétés analytiques présentées dans cette partie seront exploitées dans la Section 4.4 pour obtenir une façon rapide de calculer la limite de l'espérance, à n'importe quelle précision (et en fait même la valeur exacte).

4.3.1 Système combinatoire induit par la réduction

En suivant la construction de la classe $\mathcal{R}$, on introduit les notations suivantes : pour tout sous-ensemble de lettres $X \subseteq \Sigma$, $\mathcal{T}_{X,\varepsilon}$ représente l'ensemble des arbres qui reconnaissent le mot vide, reconnaissent toutes les lettres dans X, mais aucune lettre de $\Sigma\backslash X$. De façon similaire, on note $\mathcal{T}_{X,\overline{\varepsilon}}$ l'ensemble des arbres d'expressions régulières qui reconnaissent toutes les lettres dans X, mais aucune lettre de $\Sigma\backslash X$, ni le mot vide ε.

Exemple 4.8.
▶ Par exemple, $\mathcal{T}_{\{a\},\varepsilon}$ contient les arbres correspondant à $a^\star$ et $(a\cdot(b^\star)+\varepsilon)$, mais pas l'arbre a. ◀

Théorème 4.9.
▶ Les classes combinatoires $(\mathcal{T}_{X,\varepsilon})_{X\subseteq\Sigma}$ et $(\mathcal{T}_{X,\overline{\varepsilon}})_{X\subseteq\Sigma}$ satisfont le système combinatoire suivant :

$$\begin{aligned}\mathcal{T}_{X,\varepsilon} &= \varepsilon\mathbf{1}_{X=\emptyset} + \overset{\star}{|}_{\mathcal{T}_{X,\varepsilon}} + \overset{\star}{|}_{\mathcal{T}_{X,\overline{\varepsilon}}} + \sum_{(S,S'):S\cup S'=X} \mathcal{T}_{S,\varepsilon} \overset{\bullet}{\wedge} \mathcal{T}_{S',\varepsilon} \\ &\quad + \sum_{(S,S'):S\cup S'=X} \mathcal{T}_{S,\varepsilon} \overset{+}{\wedge} \mathcal{T}_{S',\varepsilon} + \sum_{(S,S'):S\cup S'=X} \mathcal{T}_{S,\varepsilon} \overset{+}{\wedge} \mathcal{T}_{S',\overline{\varepsilon}} + \sum_{(S,S'):S\cup S'=X} \mathcal{T}_{S,\overline{\varepsilon}} \overset{+}{\wedge} \mathcal{T}_{S',\varepsilon},\end{aligned}$$

$$\begin{aligned}\mathcal{T}_{X,\overline{\varepsilon}} &= X\mathbf{1}_{|X|=1} + \sum_{S\subseteq\Sigma} \mathcal{T}_{X,\overline{\varepsilon}} \overset{\bullet}{\wedge} \mathcal{T}_{S,\varepsilon} + \sum_{S\subseteq\Sigma} \mathcal{T}_{S,\varepsilon} \overset{\bullet}{\wedge} \mathcal{T}_{X,\overline{\varepsilon}} + \mathbf{1}_{X=\emptyset} \sum_{S,S'\subseteq\Sigma} \mathcal{T}_{S,\overline{\varepsilon}} \overset{\bullet}{\wedge} \mathcal{T}_{S',\overline{\varepsilon}} \\ &\quad + \sum_{(S,S'):S\cup S'=X} \mathcal{T}_{S,\overline{\varepsilon}} \overset{+}{\wedge} \mathcal{T}_{S',\overline{\varepsilon}},\end{aligned}$$

où la notation $\mathbf{1}_{\text{condition}}$ est un raccourci d'écriture qui signifie que le terme qui lui est accolé apparaît uniquement si la condition en indice est remplie. Il faut faire attention au fait que les décompositions dans les sommes, de la forme $S\cup S'=X$, ne sont pas des partitions : il est donc possible d'avoir $S\cap S'\neq\emptyset$. ◀

Démonstration. Soit $X\subseteq\Sigma$. Une feuille, qui est toujours étiquetée par un élément de $\Sigma\cup\{\varepsilon\}$, est dans $\mathcal{T}_{X,\varepsilon}$ si et seulement si $X=\emptyset$ et la feuille est étiquetée par ε. D'où le terme $\varepsilon\mathbf{1}_{X=\emptyset}$ dans la somme. Les autres termes s'obtiennent en regardant les racines des arbres de $\mathcal{T}_{X,\varepsilon}$:

— un arbre $\overset{\star}{|}_{T}$ appartient à $\mathcal{T}_{X,\varepsilon}$ si et seulement si T reconnaît chaque lettre de X et aucune autre (peu importe si T reconnaît ε ou non), i.e. $T \in \mathcal{T}_{X,\varepsilon}$ ou $T \in \mathcal{T}_{X,\overline{\varepsilon}}$. Les deux cas sont bien disjoints, d'où le terme $\overset{\star}{|}_{\mathcal{T}_{X,\varepsilon}} + \overset{\star}{|}_{\mathcal{T}_{X,\overline{\varepsilon}}}$;

— pour qu'un arbre $T = \overset{\bullet}{\wedge}_{T_1\ T_2}$ soit dans $\mathcal{T}_{X,\varepsilon}$, il faut que T_1 et T_2 reconnaissent ε, sinon T ne pourrait pas reconnaître ε. Donc $T_1 \in \mathcal{T}_{S,\varepsilon}$ et $T_2 \in \mathcal{T}_{S',\varepsilon}$ pour deux ensembles $S, S' \subseteq \Sigma$. Comme ε est reconnu par les deux arbres, nous déduisons que $X = S \cup S'$. Réciproquement, la concaténation de deux arbres $T_1 \in \mathcal{T}_{S,\varepsilon}$ et $T_2 \in \mathcal{T}_{S',\varepsilon}$ pour deux ensembles quelconques S, S' tels que $X = S \cup S'$ appartient à $\mathcal{T}_{X,\varepsilon}$. En remarquant que chaque paire (S, S') représente un cas disjoint, nous obtenons exactement le terme $\sum_{(S,S'):S\cup S'=X} \overset{\bullet}{\wedge}_{\mathcal{T}_{S,\varepsilon}\ \mathcal{T}_{S',\varepsilon}}$;

— enfin pour qu'un arbre $T = \overset{+}{\wedge}_{T_1\ T_2}$ appartienne à $\mathcal{T}_{X,\varepsilon}$, il faut qu'au moins un de ses fils T_1 ou T_2 reconnaisse ε. La disjonction selon lequel de T_1, T_2 (ou les deux) reconnaît ε donne le dernier terme de l'équation pour $\mathcal{T}_{X,\varepsilon}$.

Regardons maintenant l'équation pour $\mathcal{T}_{X,\overline{\varepsilon}}$. Une feuille est dans $\mathcal{T}_{X,\overline{\varepsilon}}$ si et seulement si X est un singleton de la forme $X = \{x\}$ avec $x \in \Sigma$ qui étiquette la feuille. Ceci donne le terme $X\mathbf{1}_{|X|=1}$. Les autres termes sont obtenus en considérant les racines des arbres de $\mathcal{T}_{X,\overline{\varepsilon}}$:

— un arbre qui a une étoile $\star$ à sa racine reconnaît toujours ε, et donc ne peut jamais être dans $\mathcal{T}_{X,\overline{\varepsilon}}$;

— pour qu'un arbre $T = \overset{+}{\wedge}_{T_1\ T_2}$ soit dans $\mathcal{T}_{X,\overline{\varepsilon}}$, il faut qu'aucun de ses fils ne reconnaisse ε. Donc $T_1 \in \mathcal{T}_{S,\overline{\varepsilon}}$ et $T_2 \in \mathcal{T}_{S',\overline{\varepsilon}}$ pour deux ensembles $S, S' \subseteq \Sigma$ tels que $S \cup S' = X$. Réciproquement, tout arbre de cette forme appartient à $\mathcal{T}_{X,\overline{\varepsilon}}$, et la partition suivant les paires S, S' énumère des cas disjoints. D'où le terme $\sum_{(S,S'):S\cup S'=X} \overset{+}{\wedge}_{\mathcal{T}_{S,\overline{\varepsilon}}\ \mathcal{T}_{S',\overline{\varepsilon}}}$ dans l'équation ;

— dans le cas où la racine est $\bullet$, on remarque que pour qu'un arbre $T = \overset{\bullet}{\wedge}_{T_1\ T_2}$ appartienne à $\mathcal{T}_{X,\overline{\varepsilon}}$, il est impossible qu'à la fois T_1 et T_2 reconnaissent ε, autrement T reconnaîtrait aussi ε. On considère donc les trois cas disjoints suivants :

  ◦ si $T_1 \in \mathcal{T}_{S',\overline{\varepsilon}}$ et $T_2 \in \mathcal{T}_{S,\varepsilon}$ avec $S, S' \subseteq \Sigma$, alors comme leur concaténation appartient à $\mathcal{T}_{X,\overline{\varepsilon}}$, nous en déduisons que $S' = X$. De façon réciproque, la concaténation de $T_1 \in \mathcal{T}_{X,\overline{\varepsilon}}$ et $T_2 \in \mathcal{T}_{S,\varepsilon}$ pour tout ensemble $S \subseteq \Sigma$ appartient bien à $\mathcal{T}_{X,\overline{\varepsilon}}$. D'où le terme $\sum_{S\subseteq\Sigma} \overset{\bullet}{\wedge}_{\mathcal{T}_{X,\overline{\varepsilon}}\ \mathcal{T}_{S,\varepsilon}}$;

  ◦ si $T_1 \in \mathcal{T}_{S,\varepsilon}$ et $T_2 \in \mathcal{T}_{S',\overline{\varepsilon}}$ avec $S, S' \subseteq \Sigma$, nous avons le terme symétrique du cas précédent ;

  ◦ enfin, si ni T_1 ni T_2 ne reconnaissent ε, c'est-à-dire $T_1 \in \mathcal{T}_{S,\overline{\varepsilon}}$ et $T_2 \in \mathcal{T}_{S',\overline{\varepsilon}}$ avec $S, S' \subseteq \Sigma$, alors leur concaténation ne peut reconnaître aucune lettre ni le mot vide ε. Donc X doit être l'ensemble vide. Réciproquement, pour $X = \emptyset$, la concaténation de n'importe quel arbre $T_1 \in \mathcal{T}_{S,\overline{\varepsilon}}$ avec $T_2 \in \mathcal{T}_{S',\overline{\varepsilon}}$, avec S, S' quelconques appartient bien à $\mathcal{T}_{\emptyset,\overline{\varepsilon}}$. D'où le terme final $\mathbf{1}_{X=\emptyset} \sum_{S,S'\subseteq\Sigma} \overset{\bullet}{\wedge}_{\mathcal{T}_{S,\overline{\varepsilon}}\ \mathcal{T}_{S',\overline{\varepsilon}}}$.

■

Insertion de la classe $\mathcal{R}$ dans le système. La classe des arbres entièrement réductibles $\mathcal{R}$ satisfait l'équation :

$$\mathcal{R} = \overset{\star}{\mid}_{\mathcal{T}_{\Sigma,\overline{\varepsilon}}} + \overset{\star}{\mid}_{\mathcal{T}_{\Sigma,\varepsilon}} + \overset{+}{\wedge}_{\mathcal{R}\ \mathcal{L}} + \overset{+}{\wedge}_{\mathcal{L}\setminus\mathcal{R}\ \mathcal{R}} + \overset{\bullet}{\wedge}_{\mathcal{R}\ \mathcal{T}_{\varepsilon}} + \overset{\bullet}{\wedge}_{\mathcal{T}_{\varepsilon}\setminus\mathcal{R}\ \mathcal{R}}, \tag{4.6}$$

Nous souhaitons ajouter cette équation dans notre système. Pour que les coefficients des polynômes restent positifs, nous introduisons aussi la classe $\mathcal{T}_G := \mathcal{T}_{\Sigma,\varepsilon} \setminus \mathcal{R}$, qui rassemble les arbres qui reconnaissent toutes les lettres, le mot vide ε, mais qui ne sont pas identifiés (à tort ou à raison) comme universels par l'algorithme σ. L'union $\mathcal{T}_{\Sigma,\varepsilon} = \mathcal{R} + \mathcal{T}_G$ est ainsi disjointe. Nous pouvons alors transformer l'Eq. (4.6) en une équation sans variable $\mathcal{T}_{\Sigma,\varepsilon}$ ni coefficient négatif, à l'aide des égalités suivantes :

- $\mathcal{T}_{\Sigma,\varepsilon} = \mathcal{R} + \mathcal{T}_G$;
- $\mathcal{L} = \sum_{X\subsetneq\Sigma} \mathcal{T}_{X,\varepsilon} + \mathcal{T}_G + \mathcal{R} + \sum_{X\subseteq\Sigma} \mathcal{T}_{X,\overline{\varepsilon}}$;
- $\mathcal{T}_{\varepsilon} = \sum_{X\subsetneq\Sigma} \mathcal{T}_{X,\varepsilon} + \mathcal{R} + \mathcal{T}_G$.

En particulier, $\mathcal{L}\setminus\mathcal{R} = \mathcal{T}_G + \sum_{X\subsetneq\Sigma} \mathcal{T}_{X,\varepsilon} + \sum_{X\subseteq\Sigma} \mathcal{T}_{X,\overline{\varepsilon}}$, et nous avons une équation similaire pour $\mathcal{T}_{\varepsilon} \setminus \mathcal{R}$.

Équation satisfaite par $\mathcal{T}_G$. Nous avons donc introduit deux nouvelles classes, $\mathcal{R}$ et $\mathcal{T}_G$, qui viennent remplacer $\mathcal{T}_{\Sigma,\varepsilon}$ dans le système. Il nous manque ainsi une équation sur $\mathcal{T}_G$ pour avoir un système de la bonne dimension.

De façon intuitive, l'équation pour $\mathcal{T}_G$ s'obtient en développant $\mathcal{T}_{\Sigma,\varepsilon} = \mathcal{R} + \mathcal{T}_G$ dans l'équation de $\mathcal{T}_{\Sigma,\varepsilon}$, et en éliminant ensuite les termes qui sont des éléments de $\mathcal{R}$. Si l'équation est longue à écrire, elle est formellement simple à obtenir, il suffit de retirer de l'équation les arbres commençant par $\star$ (car aucun arbre dans $\mathcal{T}_G$ ne peut avoir $\star$ pour racine), et de remplacer dans le reste des termes chaque occurrence de $\mathcal{T}_{\Sigma,\varepsilon}$ par $\mathcal{T}_G$.

Par exemple, la contribution de la concaténation dans l'équation de $\mathcal{T}_G$ est, en prenant une notation *à la maple*, $\texttt{subs}\Big(\mathcal{T}_{\Sigma,\varepsilon} = \mathcal{T}_G, \sum_{(S,S'):S\cup S'=\Sigma} \overset{\bullet}{\wedge}_{\mathcal{T}_{S,\varepsilon}\ \mathcal{T}_{S',\varepsilon}}\Big)$, soit :

$$\sum_{\substack{S\subsetneq\Sigma, S'\subsetneq\Sigma:\\ S\cup S'=\Sigma}} \overset{\bullet}{\wedge}_{\mathcal{T}_{S,\varepsilon}\ \mathcal{T}_{S',\varepsilon}} + \sum_{S\subsetneq\Sigma} \overset{\bullet}{\wedge}_{\mathcal{T}_G\ \mathcal{T}_{S,\varepsilon}} + \overset{\bullet}{\wedge}_{\mathcal{T}_G\ \mathcal{T}_G} + \sum_{S\subsetneq\Sigma} \overset{\bullet}{\wedge}_{\mathcal{T}_{S,\varepsilon}\ \mathcal{T}_G}.$$

En effectuant la même démarche de substitution pour l'union, nous obtenons l'équation satisfaite par $\mathcal{T}_G$:

$$\begin{aligned}
\mathcal{T}_G = &\sum_{\substack{S\subsetneq\Sigma, S'\subsetneq\Sigma:\\ S\cup S'=\Sigma}} \overset{\bullet}{\wedge}_{\mathcal{T}_{S,\varepsilon}\ \mathcal{T}_{S',\varepsilon}} + \sum_{S\subsetneq\Sigma} \overset{\bullet}{\wedge}_{\mathcal{T}_G\ \mathcal{T}_{S,\varepsilon}} + \overset{\bullet}{\wedge}_{\mathcal{T}_G\ \mathcal{T}_G} + \sum_{S\subsetneq\Sigma} \overset{\bullet}{\wedge}_{\mathcal{T}_{S,\varepsilon}\ \mathcal{T}_G}\\
&+ \sum_{\substack{S\subsetneq\Sigma, S'\subsetneq\Sigma:\\ S\cup S'=\Sigma}} \overset{+}{\wedge}_{\mathcal{T}_{S,\varepsilon}\ \mathcal{T}_{S',\varepsilon}} + \sum_{S\subsetneq\Sigma} \overset{+}{\wedge}_{\mathcal{T}_G\ \mathcal{T}_{S,\varepsilon}} + \overset{+}{\wedge}_{\mathcal{T}_G\ \mathcal{T}_G} + \sum_{S\subsetneq\Sigma} \overset{+}{\wedge}_{\mathcal{T}_{S,\varepsilon}\ \mathcal{T}_G}\\
&+ \sum_{\substack{S\subsetneq\Sigma, S'\subseteq\Sigma:\\ S\cup S'=\Sigma}} \overset{+}{\wedge}_{\mathcal{T}_{S,\varepsilon}\ \mathcal{T}_{S',\overline{\varepsilon}}} + \sum_{S\subseteq\Sigma} \overset{+}{\wedge}_{\mathcal{T}_G\ \mathcal{T}_{S,\overline{\varepsilon}}} + \sum_{\substack{S\subseteq\Sigma, S'\subsetneq\Sigma:\\ S\cup S'=\Sigma}} \overset{+}{\wedge}_{\mathcal{T}_{S,\overline{\varepsilon}}\ \mathcal{T}_{S',\varepsilon}} + \sum_{S\subseteq\Sigma} \overset{+}{\wedge}_{\mathcal{T}_{S,\overline{\varepsilon}}\ \mathcal{T}_G}.
\end{aligned}$$

4.3.2 Séries génératrices et probabilité d'être entièrement réductible

Nous avons transformé le système combinatoire du Théorème 4.9, sur les classes $(\mathcal{T}_{X,\overline{\varepsilon}}, \mathcal{T}_{X,\varepsilon})_{X\subseteq\Sigma}$, en un nouveau système sur les classes

$$(\mathcal{R}, \mathcal{T}_G, \mathcal{T}_{\Sigma,\overline{\varepsilon}}, (\mathcal{T}_{X,\overline{\varepsilon}}, \mathcal{T}_{X,\varepsilon})_{X\subsetneq\Sigma}) .$$

Cela nous permet d'isoler la classe $\mathcal{R}$ qui est cruciale dans l'étude de la réduction. Dans cette section, nous étudions les premières propriétés des séries génératrices associées à ce système.

Pour $X \subseteq \Sigma$, nous notons $y_{X,\varepsilon}(z)$ (resp. $y_{X,\overline{\varepsilon}}(z)$) la série génératrice associée à $\mathcal{T}_{X,\varepsilon}$ (resp. $\mathcal{T}_{X,\overline{\varepsilon}}$). On note $R(z)$ et $y_G(z)$ les séries génératrices de $\mathcal{R}$ et $\mathcal{T}_G$. En particulier, $y_{\Sigma,\varepsilon}(z) = R(z) + y_G(z)$. Dans la suite, nous allons représenter ces séries dans un vecteur colonne de séries :

$$\boldsymbol{y}(z) = [R(z), y_G(z), y_{\Sigma,\overline{\varepsilon}}(z), (y_{X,\overline{\varepsilon}}(z), y_{X,\varepsilon}(z))_{X\subsetneq\Sigma}] .$$

Proposition 4.10 .
▶ Le vecteur $\boldsymbol{y}(z)$ satisfait un système d'équation de la forme :

$$\boldsymbol{y}(z) = \boldsymbol{\Phi}(z; \boldsymbol{y}(z)) \tag{4.7}$$

où chaque composante de $\boldsymbol{\Phi}(z; \boldsymbol{y})$ est un polynôme de degré 2 en $\boldsymbol{y}$, de degré 1 en z, tel que $\boldsymbol{\Phi}(0; \boldsymbol{y}) = \mathbf{0}$. ◀

Démonstration. La traduction du système combinatoire du Théorème 4.9 élargi avec les équations pour $\mathcal{R}$ and $\mathcal{T}_G$, en un système pour les séries génératrices est directe. Par souci de complétude, nous donnons l'équation pour $y_{X,\overline{\varepsilon}}(z)$, avec $X \subseteq \Sigma$, dont l'équation combinatoire :

$$\begin{aligned}\mathcal{T}_{X,\overline{\varepsilon}} = X\mathbf{1}_{|X|=1} &+ \sum_{(S,S'):S\cup S'=X} \mathcal{T}_{S,\overline{\varepsilon}} \overset{+}{\wedge} \mathcal{T}_{S',\overline{\varepsilon}} \\ &+ \sum_{S\subseteq\Sigma} \mathcal{T}_{X,\overline{\varepsilon}} \overset{\bullet}{\wedge} \mathcal{T}_{S,\varepsilon} + \sum_{S\subseteq\Sigma} \mathcal{T}_{S,\varepsilon} \overset{\bullet}{\wedge} \mathcal{T}_{X,\overline{\varepsilon}} + \mathbf{1}_{X=\emptyset} \sum_{S,S'\subseteq\Sigma} \mathcal{T}_{S,\overline{\varepsilon}} \overset{\bullet}{\wedge} \mathcal{T}_{S',\overline{\varepsilon}}\end{aligned}$$

se traduit en l'équation fonctionnelle :

$$\begin{aligned}y_{X,\overline{\varepsilon}}(z) = z\mathbf{1}_{|X|=1} &+ z \sum_{(S,S'):S\cup S'=X} y_{S,\overline{\varepsilon}}(z) y_{S',\overline{\varepsilon}}(z) \\ &+ z \sum_{S\subseteq\Sigma} y_{X,\overline{\varepsilon}}(z) y_{S,\varepsilon}(z) + z \sum_{S\subseteq\Sigma} y_{S,\varepsilon} y_{X,\overline{\varepsilon}}(z) + z\mathbf{1}_{X=\emptyset} \sum_{S,S'\subseteq\Sigma} y_{S,\overline{\varepsilon}}(z) y_{S',\overline{\varepsilon}}(z)\end{aligned}$$

Il suffit ensuite de remplacer $y_{\Sigma,\varepsilon}(z)$ par $R(z) + T_G(z)$ pour obtenir une équation de la forme

$$y_{X,\overline{\varepsilon}}(z) = \boldsymbol{\Phi}_{X,\overline{\varepsilon}}(z, \boldsymbol{y}(z))$$

où $\boldsymbol{\Phi}_{X,\overline{\varepsilon}}(z, \boldsymbol{y})$ est un polynôme en z et $\boldsymbol{y}$, de degré 1 en z, avec z en facteur, et de degré 2 en $\boldsymbol{y}$.

Nous voyons facilement que cette dernière propriété est vérifiée pour les autres polynômes $\mathbf{\Phi}_{X,\varepsilon}, \mathbf{\Phi}_R, \mathbf{\Phi}_G$ associés aux séries génératrices restantes. Nous obtenons donc un système de la forme :

$$\boldsymbol{y}(z) = \mathbf{\Phi}(z; \boldsymbol{y}(z))$$

qui vérifie les propriétés annoncées dans la proposition. ■

Maintenant que le système est établi, nous devons vérifier les hypothèses du Théorème 1.89 de Drmota pour pouvoir affirmer le comportement en racine carrée voulu en ρ. En particulier, nous voulons montrer que :

— le graphe de dépendance $G_{\mathbf{\Phi}}$ associé au système est fortement connexe, voir le Lemme 4.11.

— les solutions du système sont apériodiques, voir le Lemme 4.12.

— le rayon de convergence commun aux séries solutions du système, par le théorème de Drmota, coïncide avec le rayon de convergence de $L(z)$, c'est-à-dire $\rho = \rho_L$, voir le Lemme 4.13.

Pour simplifier les notations, et uniquement pour les preuves de cette section, nous allons renommer les coordonnées des vecteurs en les indiçant par des nombres, et ainsi écrire $\boldsymbol{y} = (y_i)_{i=1\dots m}$ et $\boldsymbol{\phi} = (\mathbf{\Phi}_i)_{i=1\dots m}$ où $m = 2^{k+1} + 1$.

On rappelle que le graphe de dépendance $G_{\mathbf{\Phi}}$ associé à $\mathbf{\Phi}$ a m sommets $y_1, \dots, y_m$, et est tel qu'il existe un arc orienté du sommet y_i au sommet y_j si le degré en y_j du polynôme $\mathbf{\Phi}_i(z, \boldsymbol{y})$ est non nul. Autrement dit $y_i \to y_j$ dès que $\deg_{y_j}(\Phi) \geqslant 1$.

Lemme 4.11.

▶ Le graphe $G_{\mathbf{\Phi}}$ est fortement connexe. ◀

Démonstration. Regardons le système du Théorème 4.9 :

— Pour chaque $X \subsetneq \Sigma$, l'équation de $\mathcal{T}_{X,\varepsilon}$ dépend de $\mathcal{T}_{\emptyset,\bar{\varepsilon}}$ (en prenant $S = \emptyset$ dans la dernière somme). L'équation de $\mathcal{T}_G$ et $\mathcal{R}$ dépend aussi de $\mathcal{T}_{\emptyset,\bar{\varepsilon}}$. Ainsi, pour tout sommet $y \in \{y_G, y_R, (y_{X,\varepsilon})_{X\subsetneq\Sigma}\}$, il y a un arc de y au sommet $y_{\emptyset,\bar{\varepsilon}}$ dans $G_{\mathbf{\Phi}}$.

— Comme dans l'expression de $\mathbf{\Phi}_{\emptyset,\bar{\varepsilon}}$, on trouve le polynôme $z\sum_{S,S'\subseteq\Sigma} y_{S,\bar{\varepsilon}}y_{S',\bar{\varepsilon}}$, il y a dans $G_{\mathbf{\Phi}}$ un arc du sommet $y_{\emptyset,\bar{\varepsilon}}$ à chaque sommet $y_{S,\bar{\varepsilon}}$, pour tout $S \subseteq \Sigma$.

— Finalement, pour tout $X \subseteq \Sigma$, on trouve dans $\mathbf{\Phi}_{X,\bar{\varepsilon}}$ l'expression $zy_{X,\bar{\varepsilon}}(y_R + y_G) + z\sum_{S\subsetneq\Sigma} y_{X,\bar{\varepsilon}}y_{S,\varepsilon}$, ce qui implique des arcs partant de $y_{X,\bar{\varepsilon}}$ jusqu'à y_R, y_G et tout $y_{S,\varepsilon}$ avec $S \subsetneq \Sigma$.

En conclusion le graphe $G_{\mathbf{\Phi}}$ est fortement connexe. ■

Lemme 4.12.

▶ Pour n assez grand, toutes les entrées de $[z^n]\boldsymbol{y}(z)$ sont strictement positives. ◀

Démonstration. Puisque $\overset{\star}{\underset{\mathcal{R}}{|}} \subset \mathcal{R}$, et $\overset{\star}{\underset{\mathcal{T}_{X,\varepsilon}}{|}} \subset \mathcal{T}_{X,\varepsilon}$ pour tout $X \subseteq \Sigma$, nous déduisons que $[z^{n+1}]y_R(z) \geqslant [z^n]y_R(z)$ et $[z^{n+1}]y_{X,\varepsilon}(z) \geqslant [z^n]y_{X,\varepsilon}(z)$ pour tout $X \subseteq \Sigma$.

De plus, comme $\overset{\bullet}{\underset{\mathcal{T}_{X,\bar{\varepsilon}}}{\wedge}}_{\varepsilon} \subset \mathcal{T}_{X,\bar{\varepsilon}}$, pour tout $X \subseteq \Sigma$, il suit que $[z^{n+2}]y_{X,\bar{\varepsilon}}(z) \geqslant [z^n]y_{X,\bar{\varepsilon}}(z)$ pour tout $X \subseteq \Sigma$. Nous obtenons la même inégalité avec $y_G(z)$, pour les mêmes raisons.

La stricte positivité des coefficients de Taylor de la solution $\boldsymbol{y}(z)$ s'ensuit par récurrence, après avoir calculé les premiers termes à la main. ■

Lemme 4.13 .
▶ Les séries $y_R(z)$ et $L(z)$ ont le même rayon de convergence ρ. ◀

Démonstration. L'expression de $L(z)$ montre qu'elle a une unique singularité dominante en $z = \rho$. Par le théorème de Pringsheim [FS09, Theorem IV.6]), elle coincide avec le rayon de convergence de la série.

Comme $\mathcal{R} \subset \mathcal{L}$, on sait déjà que $[z^n]y_R(z) \leqslant [z^n]L(z)$. Donc le rayon de convergence de $y_R(z)$ est plus grand que ρ. Réciproquement, soit $T \in \mathcal{R}$ un arbre universel de taille $2k$; nous pouvons prendre celui représentant $(a_1 + \ldots + a_k)^*$. Puisque $\overset{+}{\underset{T}{\wedge}}_{\mathcal{L}} \subseteq \mathcal{R}$, nous en déduisons que $[z^{n-2k-1}]L(z) \leqslant [z^n]y_R(z)$. Donc le rayon de convergence de $y_R(z)$ est plus petit que ρ. ■

Nous pouvons maintenant appliquer le Théorème 1.89 de Drmota pour trouver le comportement attendu en racine carrée :

Proposition 4.14 .
▶ Chaque coordonnée de la solution $\boldsymbol{y}(z)$ du système a une unique singularité dominante en ρ, si bien qu'au voisinage de $z = \rho$:

$$\boldsymbol{y}(z) = \boldsymbol{h}(z) - \boldsymbol{g}(z)\sqrt{1 - z/\rho} \tag{4.8}$$

où $\boldsymbol{h}(z)$ et $\boldsymbol{g}(z)$ sont deux vecteurs de fonctions analytiques dans un voisinage de $z = \rho$. De plus, chaque coordonnée du vecteur $\boldsymbol{g}(\rho)$ est strictement positive. ◀

Démonstration. Comme annoncé, nous allons rapidement vérifier les hypothèses du théorème de Drmota. Nous avons déjà montré que G_Φ est fortement connexe, les coefficients de Taylor de la solution $\boldsymbol{y}(z)$ sont positifs en tant que séries de comptage, et strictement positifs à partir d'un certain rang par le Lemme 4.12. Comme chaque composante de $\boldsymbol{\Phi}(z, \boldsymbol{y})$ est un polynôme, ce sont des fonctions analytiques en $(z, \boldsymbol{y}) = (0, \mathbf{0})$. De plus, $\boldsymbol{\Phi}(0, \mathbf{0}) \equiv \mathbf{0}$. Grâce au z en facteur, $\boldsymbol{\Phi}(0, \boldsymbol{y}) \equiv \mathbf{0}$. Ensuite, on vérifie que $\Phi_{\emptyset,\varepsilon}(z, \mathbf{0}) = z$, et par conséquent $\boldsymbol{\Phi}(z, \mathbf{0}) \not\equiv \mathbf{0}$. Le système n'est pas linéaire en $\boldsymbol{y}$, puisque chaque composante de $\boldsymbol{\Phi}(z, \boldsymbol{y})$ a un degré 2 en $\boldsymbol{y}$.

Par le théorème de Drmota (voir 1.89), tous les $y_j(z)$ ont une unique singularité dominante $\tilde{\rho}$, qui coïncide avec leur rayon de convergence commun. Donc $\tilde{\rho} = \rho$ par le Lemme 4.13.

De plus, le théorème énonce que $y_j(\rho) := \tau_j < \infty$. Puisque $\boldsymbol{\phi}$ est un polynôme, $(\rho, \boldsymbol{\tau})$ est un point caractéristique qui se situe à l'intérieur du disque de convergence de $\boldsymbol{\phi}$, de telle sorte que $\boldsymbol{\tau} = \boldsymbol{\phi}(\rho; \boldsymbol{\tau})$ et $0 = \det(\mathtt{Id} - \mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{\phi}](\rho; \boldsymbol{\tau}))$.

Enfin, le théorème énonce que pour tout j, nous pouvons développer $y_j(z) = h_j(z) - g_j(z)\sqrt{1 - z/\rho}$ localement autour de $z = \rho$, avec $z \notin [\rho, +\infty)$, où $h_j(z)$ et $g_j(z)$ sont analytiques en $z = \rho$, et $g_j(\rho) \neq 0$. Le Théorème 1.81 de Transfert nous donne alors l'équivalence $[z^n]y_j(z) \sim_{n\to\infty} g_j(\rho)\rho^{-n}n^{-3/2}/\Gamma(-1/2)$. Comme par le Lemme 4.12, $[z^n]y_j(z) > 0$ à partir d'un certain rang, nous en déduisons que $g_j(\rho) > 0$. ■

Théorème 4.15 .
▶ La probabilité qu'un arbre aléatoire uniforme T de taille n appartienne à une classe $\mathcal{C}$ de notre système combinatoire étendu tend vers la constante strictement positive $g_{\mathcal{C}}(\rho)/g_L(\rho)$ lorsque $n \to \infty$. En particulier, pour $\mathcal{C} = \mathcal{R}$, la proportion limite des arbres entièrement réductibles par l'algorithme de réduction est strictement positive. ◀

Démonstration. La preuve de la Proposition 4.14 nous permet d'appliquer le théorème de Transfert aux coordonnées de $\boldsymbol{y}(z)$.

Soit $\mathcal{C}$ une classe de notre système. Le nombre d'arbres de taille n dans $\mathcal{C}$ est équivalent par le théorème de Transfert à $g_j(\rho)\rho^{-n}n^{-3/2}/\Gamma(-1/2)$ lorsque $n \to \infty$. Comme $[z^n]L(z) \sim_{n\to\infty} g_L(\rho)\rho^{-n}n^{-3/2}/\Gamma(-1/2)$, la probabilité pour un arbre uniforme de taille n d'être dans la classe $\mathcal{C}$ converge vers $g_{\mathcal{C}}(\rho)/g_L(\rho) > 0$, lorsque $n \to \infty$. ■

4.3.3 Système bivarié et spécification de la réduction

Pour obtenir des informations sur la taille après réduction, comme nous l'avons expliqué à la Section 4.2.3, nous introduisons une nouvelle variable u qui va marquer les nœuds de l'expression réduite. Nous introduisons naturellement les séries bivariées associées au système $\boldsymbol{y}(z,u)$. Comme la réduction est entièrement décrite par le système, le passage aux séries bivariées est aisé. Il est d'abord immédiat que $R(z,u) = u^{|\mathcal{U}|}R(z)$. Ensuite, pour toutes les autres classes d'arbres du système, comme ces classes ne sont pas universelles pour notre algorithme, la racine des arbres de ces classes est toujours présente après réduction. Ainsi les équations à deux variables dérivent directement des équations à une variable, avec seulement un facteur additionnel u au niveau de la racine des règles. La proposition suivante formalise cette discussion :

Le système a été construit spécialement dans l'optique de pouvoir modéliser facilement la réduction.

Proposition 4.16 .
▶ Posons $\boldsymbol{y} = (R, \tilde{\boldsymbol{y}})$, et $\boldsymbol{\Phi} = (\Phi_R, \tilde{\boldsymbol{\Phi}})$. Le vecteur de séries génératrices bivariées $\tilde{\boldsymbol{y}}(z,u)$ satisfait le système suivant :

$$\tilde{\boldsymbol{y}}(z,u) = \tilde{\boldsymbol{\Phi}}(zu; u^p R(z), \tilde{\boldsymbol{y}}(z,u))$$

où $\boldsymbol{\Phi} = (\Phi_R, \tilde{\boldsymbol{\Phi}})$ est défini dans l'Eq. 4.7, et $p := |\mathcal{U}|$. ◀

Remarquons qu'avec ces notations, nous avons l'égalité

$$L(z,u) = u^p R(z) + (1,\dots,1)\cdot\tilde{\boldsymbol{y}}(z,u)\,.$$

Pour appliquer la formule de l'Eq (4.5) donnant la taille moyenne après réduction :

$$\mathbb{E}_n[|\sigma|] = \frac{[z^n]\partial_u L(z,u)\big|_{u=1}}{[z^n]L(z)}\,, \tag{4.5}$$

il faut dériver l'équation de $L(z,u)$ par rapport à u, et ensuite spécialiser en $u=1$. Pour simplifier les notations, nous noterons $Q\tilde{\boldsymbol{y}}(z) := \partial_u\tilde{\boldsymbol{y}}(z,u)\big|_{u=1}$ la "quantité cumulée" associée au système.

Proposition 4.17 .
▶ Le vecteur $Q\tilde{\boldsymbol{y}}(z) = \partial_u\tilde{\boldsymbol{y}}(z,u)\big|_{u=1}$ satisfait le système linéaire :

$$(\mathtt{Id} - \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\boldsymbol{\Phi}}](z; R(z), \tilde{\boldsymbol{y}}(z)))Q\tilde{\boldsymbol{y}}(z) = \tilde{\boldsymbol{\Phi}}(z; R(z), \tilde{\boldsymbol{y}}(z)) + p\partial_R\tilde{\boldsymbol{\Phi}}(z; R(z), \tilde{\boldsymbol{y}}(z))R(z)$$

◀

Démonstration. Dérivons par rapport à u l'équation de la Proposition 4.16 :

$$\tilde{\boldsymbol{y}}(z,u) = \tilde{\boldsymbol{\Phi}}(zu; u^p R(z), \tilde{\boldsymbol{y}}(z,u))\,.$$

Ceci nous donne :

$$\begin{aligned}\partial_u \tilde{\boldsymbol{y}}(z,u) = {} & z\partial_z \tilde{\boldsymbol{\Phi}}(zu; u^p R(z), \tilde{\boldsymbol{y}}(z,u)) \\ & + pu^{p-1} R(z) \partial_R \tilde{\boldsymbol{\Phi}}(zu; u^p R(z), \tilde{\boldsymbol{y}}(z,u)) \\ & + \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\boldsymbol{\Phi}}](zu; u^p R(z), \tilde{\boldsymbol{y}}(z,u)) \partial_u \tilde{\boldsymbol{y}}(z,u)\end{aligned}$$

En évaluant $u = 1$ et en remarquant que comme $\tilde{\boldsymbol{\Phi}}$ est un polynôme de degré 1 en z, avec z en facteur, nous avons l'égalité $z\partial_z \tilde{\boldsymbol{\Phi}} = \tilde{\boldsymbol{\Phi}}$, nous obtenons finalement la relation :

$$(\mathtt{Id} - \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\boldsymbol{\Phi}}](z; R(z), \tilde{\boldsymbol{y}}(z))) Q\tilde{\boldsymbol{y}}(z) = \tilde{\boldsymbol{\Phi}}(z; R(z), \tilde{\boldsymbol{y}}(z)) + p\partial_R \tilde{\boldsymbol{\Phi}}(z; R(z), \tilde{\boldsymbol{y}}(z)) R(z)$$

■

Maintenant nous pouvons démontrer que la quantité cumulée $Q\tilde{\boldsymbol{y}}(z)$ a bien un comportement en racine carrée en $z = \rho$:

Proposition 4.18 .
▶ Chaque coordonnée du vecteur $Q\tilde{\boldsymbol{y}}(z) = \partial_u \tilde{\boldsymbol{y}}(z,u)\big|_{u=1}$ a une unique singularité dominante, en $z = \rho$. De plus, dans un voisinage de $z = \rho$ nous pouvons écrire :

$$Q\tilde{\boldsymbol{y}}(z) = \boldsymbol{h}_{Q\tilde{\boldsymbol{y}}}(z) - \boldsymbol{g}_{Q\tilde{\boldsymbol{y}}}(z)\sqrt{1 - z/\rho}$$

où $\boldsymbol{h}_{Q\tilde{\boldsymbol{y}}}(z)$ et $\boldsymbol{g}_{Q\tilde{\boldsymbol{y}}}(z)$ sont deux vecteurs de fonctions analytiques dans un voisinage de $z = \rho$, tels que chaque coordonnée du vecteur $\boldsymbol{g}_{Q\tilde{\boldsymbol{y}}}(\rho)$ est strictement positive. ◀

Démonstration. Posons $M(z; \boldsymbol{y}) := \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\phi}](z; \boldsymbol{y})$, qui est une matrice carrée de polynômes en z et $\boldsymbol{y}$, à coefficients positifs.

La matrice $M(\rho, \boldsymbol{L}(\rho))$ est une matrice positive obtenue à partir de $\mathtt{Jac}_{\boldsymbol{y}}[\phi](\rho; \boldsymbol{L}(\rho))$ en supprimant sa première ligne et sa première colonne. Par le corollaire 2.58 présenté dans une section technique du chapitre 2, nous en déduisons que $\mathtt{Sp}(M(\rho; \boldsymbol{L}(\rho))) < 1$.

À partir de là nous pouvons appliquer une autre proposition technique, la Proposition 2.53 :

a. $(\mathtt{Id} - \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\boldsymbol{\Phi}}](z; R(z), \tilde{\boldsymbol{y}}(z)))$ est inversible pour tout $|z| \leqslant \rho$

b. les entrées de la matrice $\boldsymbol{J}(z) := (\mathtt{Id} - \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\boldsymbol{\Phi}}](z; R(z), \tilde{\boldsymbol{y}}(z)))^{-1}$ sont des fonctions analytiques pour $|z| < \rho$, continues sur $[0, \rho]$, ont au plus une singularité dominante sur le cercle $|z| = \rho$, située en $z = \rho$, et sont localement de la forme $g(z) - h(z)\sqrt{1 - z/\rho}$, avec $g(z)$ et $h(z)$ analytiques en ρ.

Ainsi les composantes du vecteur

$$Q\tilde{\boldsymbol{y}}(z) = \boldsymbol{J}(z) \cdot \big(\tilde{\boldsymbol{\Phi}}(z; R(z), \tilde{\boldsymbol{y}}(z)) + p\partial_R \tilde{\boldsymbol{\Phi}}(z; R(z), \tilde{\boldsymbol{y}}(z)) R(z)\big)$$

sont bien analytiques sur $|z| < \rho$, et la seule singularité dominante possible sur le cercle $|z| = \rho$ se trouve en $z = \rho$.

Comme $[z^n]L_i(z) \leqslant [z^n]Q\tilde{\boldsymbol{y}}_i(z)$ pout tout n, on en déduit que le rayon de convergence de $Q\tilde{\boldsymbol{y}}_i(z)$ est inférieur à ρ. Donc par le théorème de Pringsheim [FS09], ρ est bien l'unique singularité dominante de $Q\tilde{\boldsymbol{y}}_i(z)$. ■

En utilisant l'Eq (4.5) et le Théorème 1.81 de Transfert, nous pouvons finalement conclure :

Théorème 4.19 (Limite de la taille moyenne après réduction).
▶ On considère la simplification linéaire σ appliquée à des arbres d'expressions régulières sur un alphabet de taille fixée $|\Sigma| = k$ définis par la spécification

$$\mathcal{L}_{\mathcal{R}} = a_1 + \ldots + a_k + \varepsilon + \overset{\star}{\underset{\mathcal{L}_{\mathcal{R}}}{|}} + \overset{\bullet}{\underset{\mathcal{L}_{\mathcal{R}}}{/\!\backslash}}\,{}_{\mathcal{L}_{\mathcal{R}}} + \overset{+}{\underset{\mathcal{L}_{\mathcal{R}}}{/\!\backslash}}\,{}_{\mathcal{L}_{\mathcal{R}}} . \tag{4.1}$$

Alors la taille moyenne, après réduction par σ, d'une expression régulière aléatoire uniforme de taille n tend vers une constante lorsque n tend vers l'infini :

$$\lim_{n\to+\infty} \mathbb{E}_n[|\sigma(T)|] = \frac{|\mathcal{U}| g_R(\rho) + \|\boldsymbol{g}_{Q\tilde{\boldsymbol{y}}}(\rho)\|_1}{g_L(\rho)} \tag{4.9}$$

où $\|(v_1, \ldots, v_s)\|_1 = |v_1| + \ldots + |v_s|$. ◀

Remarque 4.20 (Taille de $\mathcal{U}$).
▶ La taille de $\mathcal{U}$ est un paramètre qui entre en jeu dans la taille moyenne. L'arbre $\mathcal{U}$ est un arbre minimal utilisé par σ pour remplacer un arbre détecté comme universel. Une valeur naturelle pour la taille de $\mathcal{U}$ est donc $|\mathcal{U}| = 2k$ si on utilise un arbre de taille minimale unaire-binaire représentant l'expression régulière $\Sigma^\star$; on peut aussi choisir $|\mathcal{U}| = 1$ en utilisant un symbole spécial, il faut toutefois adapter la réduction σ. Attention, dans la formule précédente, des dépendances en $|\mathcal{U}|$ sont cachées dans le vecteur $(g_R(\rho), \boldsymbol{g}_{Q\tilde{\boldsymbol{y}}}(\rho))$, dont les composantes dépendent bien de $|\mathcal{U}|$. ◀

4.4 Calcul pratique de la limite et analyse numérique

Dans cette section, nous cherchons un moyen efficace de calculer la limite annoncée dans le Théorème 4.19, pour une taille d'alphabet k fixée. Pour ce faire nous utilisons l'Eq (4.9), qui nous dit que nous devons calculer $g_R(\rho)$ et $\boldsymbol{g}_{Q\tilde{\boldsymbol{y}}}(\rho)$. Le point de départ pour calculer ces valeurs est la proposition suivante :

Proposition 4.21 .
▶ Le vecteur $g_{Q\tilde{\boldsymbol{y}}}(\rho)$ satisfait une équation de la forme :

$$g_{Q\tilde{\boldsymbol{y}}}(\rho) = \Big(\mathtt{Id} - \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\boldsymbol{\Phi}}](\rho; \boldsymbol{y}(\rho))\Big)^{-1} \cdot \boldsymbol{K}_{\boldsymbol{\Phi}}(\rho; \boldsymbol{y}(\rho), g_{\boldsymbol{y}}(\rho), Q\tilde{\boldsymbol{y}}(\rho))$$

où $\boldsymbol{K}_{\boldsymbol{\Phi}}(z; \boldsymbol{y}, \boldsymbol{g}, \boldsymbol{h_Q})$ dépend des dérivées de $\boldsymbol{\Phi}$, de $p = |\mathcal{U}|$, et est un polynôme en ses entrées. ◀

Démonstration. Dans un premier temps, remarquons que pour toute fonction analytique de la forme $w(z) = h(z) - g(z)\sqrt{1 - z/\rho}$, alors $w'(z) = O(1) + \frac{g(\rho)}{2\rho\sqrt{1-z/\rho}}$ pour $z \sim \rho$.

Nous allons donc dériver le système de la Prop. 4.17 par rapport à z :

$$(\mathtt{Id} - \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\boldsymbol{\Phi}}](z; R(z), \tilde{\boldsymbol{y}}(z))) \cdot Q\tilde{\boldsymbol{y}}(z) = \tilde{\boldsymbol{\Phi}}(z; R(z), \tilde{\boldsymbol{y}}(z)) + p\partial_R \tilde{\boldsymbol{\Phi}}(z; R(z), \tilde{\boldsymbol{y}}(z)) R(z) .$$

et identifier les termes en $\frac{1}{2\rho\sqrt{1-z/\rho}}$ au voisinage de ρ.

Rappelons aussi que $\tilde{\mathbf{\Phi}}$ est un polynôme, de degré 1 en z, avec z qui est en facteur, et de degré 2 en les autres variables. Donc les entrées de la matrice $\mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\mathbf{\Phi}}](z; R, \tilde{\boldsymbol{y}})$ sont simplement des combinaisons affines à coefficients réels de $R, \tilde{\boldsymbol{y}}$, multipliées par un facteur z. Nous avons donc l'égalité :

$$\partial_z(\mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\mathbf{\Phi}}](z; R(z), \tilde{\boldsymbol{y}}(z))) = \tfrac{1}{z}\mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\mathbf{\Phi}}](z; R(z), \tilde{\boldsymbol{y}}(z)) + \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\mathbf{\Phi}}](z; R'(z), \tilde{\boldsymbol{y}}'(z))$$

où $\tilde{\boldsymbol{y}}'(z)$ désigne le vecteur obtenu à partir du vecteur $\tilde{\boldsymbol{y}}(z)$ en dérivant chaque coordonnée par rapport à z. Ainsi :

— En dérivant $(\mathtt{Id} - \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\mathbf{\Phi}}](z; R(z), \tilde{\boldsymbol{y}}(z))) \cdot Q\tilde{\boldsymbol{y}}(z)$ nous obtenons donc :

  ◦ un premier terme $(\mathtt{Id} - \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\mathbf{\Phi}}](z; R(z), \tilde{\boldsymbol{y}}(z))) \cdot \partial_z(Q\tilde{\boldsymbol{y}}(z))$, dont le coefficient en $(2\rho\sqrt{1 - z/\rho})^{-1}$ au voisinage de ρ est $(\mathtt{Id} - \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\mathbf{\Phi}}](\rho; \boldsymbol{y}(\rho))) \cdot g_{Q\tilde{\boldsymbol{y}}}(\rho)$;

  ◦ un terme $\frac{1}{z}\mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\mathbf{\Phi}}](z; R(z), \tilde{\boldsymbol{y}}(z))) \cdot Q\tilde{\boldsymbol{y}}(z)$ qui n'apporte rien en $(1 - \frac{z}{\rho})^{-1/2}$;

  ◦ un dernier terme $-\mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\mathbf{\Phi}}](z; R'(z), \tilde{\boldsymbol{y}}'(z))) \cdot Q\tilde{\boldsymbol{y}}(z)$. Au voisinage de ρ, les dérivées dans la matrice introduisent un coefficient en $(2\rho\sqrt{1 - z/\rho})^{-1}$, qui est $-\tilde{J}(\rho; g_{\boldsymbol{y}}(\rho)) \cdot Q\tilde{\boldsymbol{y}}(\rho)$, où $\tilde{J}(z; \boldsymbol{y}) := \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\mathbf{\Phi}}](z; \boldsymbol{y}) - \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\mathbf{\Phi}}](z; \mathbf{0})$ (les termes qui ne dépendent que de z dans la matrice jacobienne n'interviennent pas pour les coefficients en $(2\rho\sqrt{1 - z/\rho})^{-1}$) ;

  avec, on le rappelle, la notation $\boldsymbol{y} = (R, \tilde{\boldsymbol{y}})$.

— Nous procédons de même en dérivant $\tilde{\mathbf{\Phi}}(z; R(z), \tilde{\boldsymbol{y}}(z))$ par rapport à z dans le membre droit, ce qui nous donne $\mathtt{Jac}_{\boldsymbol{y}}[\tilde{\mathbf{\Phi}}](\rho; \boldsymbol{y}(\rho)) \cdot g_{\boldsymbol{y}}(\rho)$ comme coefficient des termes en $(2\rho\sqrt{1 - z/\rho})^{-1}$.

— Enfin en dérivant $p\partial_R\tilde{\mathbf{\Phi}}(z; R(z), \tilde{\boldsymbol{y}}(z))R(z)$ et en identifiant les coefficients en $(2\rho\sqrt{1 - z/\rho})^{-1}$, nous obtenons $p \cdot \partial_R\tilde{\mathbf{\Phi}}(\rho; y(\rho)) \cdot g_R(\rho) + pR(\rho) \cdot \partial_R\mathtt{Jac}_{\boldsymbol{y}}[\tilde{\mathbf{\Phi}}](\rho, y(\rho)) \cdot g_y(\rho)$.

En fin de compte, nous avons l'égalité :

$$\begin{aligned}
&(\mathtt{Id} - \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\mathbf{\Phi}}](\rho; \boldsymbol{y}(\rho))) \cdot g_{Q\tilde{\boldsymbol{y}}}(\rho) \\
= \;& \mathtt{Jac}_{\boldsymbol{y}}[\tilde{\mathbf{\Phi}}](\rho; y(\rho)) \cdot g_{\boldsymbol{y}}(\rho) + p\partial_R\tilde{\mathbf{\Phi}}(\rho; y(\rho)) \cdot g_R(\rho) \\
&+ pR(\rho) \cdot \partial_R\mathtt{Jac}_{\boldsymbol{y}}[\tilde{\mathbf{\Phi}}](\rho, y(\rho)) \cdot g_y(\rho) \\
&+ \tilde{J}(\rho; g_{\boldsymbol{y}}(\rho)) \cdot Q\tilde{\boldsymbol{y}}(\rho)
\end{aligned}$$

avec $\tilde{J}(z; \boldsymbol{y}) := \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\mathbf{\Phi}}](z; \boldsymbol{y}) - \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\mathbf{\Phi}}](z; \mathbf{0})$.
Nommons $\boldsymbol{K}_{\mathbf{\Phi}}(\rho; \boldsymbol{y}(\rho), g_{\boldsymbol{y}}(\rho), Q\tilde{\boldsymbol{y}}(\rho))$ le terme droit de l'égalité. Il dépend des dérivées de $\mathbf{\Phi}$, de $p = |\mathcal{U}|$, et est bien un polynôme en ses entrées. ■

Pour connaître $g_{Q\tilde{\boldsymbol{y}}}(\rho)$, nous avons donc besoin de calculer $\boldsymbol{y}(\rho)$, $g_{\boldsymbol{y}}(\rho)$ et $Q\tilde{\boldsymbol{y}}(\rho)$:

a. Pour calculer $\boldsymbol{y}(\rho)$, nous allons réécrire le système sous une forme triangulaire, en exploitant les fonctions auxiliaires $L(z)$ et $T_\varepsilon(z)$. Nous pourrons alors calculer $\boldsymbol{y}(\rho)$ efficacement par programmation dynamique. Les détails se trouvent dans la Section 4.4.1.

b. Ensuite, dans la Section 4.4.2, $\boldsymbol{g}_{\boldsymbol{y}}(\rho)$ est calculé en résolvant un système linéaire, car c'est un vecteur propre de la matrice $\mathtt{Jac}_{\boldsymbol{y}}[\phi](\rho, \boldsymbol{y}(\rho))$.

c. Enfin, en posant $z = \rho$ dans la Prop. 4.17, nous pouvons calculer $Q\tilde{\boldsymbol{y}}(\rho)$ en résolvant le système linéaire :

$$(\mathtt{Id} - \mathtt{Jac}_{\tilde{\boldsymbol{y}}}[\tilde{\boldsymbol{\Phi}}](\rho; \boldsymbol{y}(\rho))) \cdot Q\tilde{\boldsymbol{y}}(\rho) = \tilde{\boldsymbol{\Phi}}(z; \boldsymbol{y}(\rho)) + p\partial_R \tilde{\boldsymbol{\Phi}}(\rho; R(\rho), \tilde{\boldsymbol{y}}(\rho))R(\rho)$$

La matrice est bien inversible en ρ, comme nous l'avons montré dans la Proposition 4.18.

4.4.1 Forme triangulaire du système (calcul de $\boldsymbol{y}(\rho)$)

Le système combinatoire que nous avons introduit dans le Théorème 4.9 a l'avantage d'être fortement connexe et donc de bien se prêter aux théorèmes de Drmota. Néanmoins, on se rend compte vite que certains termes des équations $\mathcal{T}_{X,\bar{\varepsilon}}$ interviennent ensemble dans la spécification, et peuvent se regrouper, en introduisant dans la spécification les classes combinatoires $\mathcal{L}_{\mathcal{R}}$, $\mathcal{T}_{\varepsilon}$ et $\mathcal{T}_{\bar{\varepsilon}} := \mathcal{L}_{\mathcal{R}} \backslash \mathcal{T}_{\varepsilon}$. La proposition suivante détaille cette factorisation :

Proposition 4.22 .
▶ La classe combinatoire $(\mathcal{T}_{X,\bar{\varepsilon}})_{X \subseteq \Sigma}$ satisfait la spécification suivante :

$$\mathcal{T}_{X,\bar{\varepsilon}} = X\mathbf{1}_{|X|=1} + \mathcal{T}_{X,\bar{\varepsilon}} \overset{\bullet}{\wedge} \mathcal{T}_{\varepsilon} + \mathcal{T}_{\varepsilon} \overset{\bullet}{\wedge} \mathcal{T}_{X,\bar{\varepsilon}} + \mathbf{1}_{X=\emptyset} \mathcal{T}_{\bar{\varepsilon}} \overset{\bullet}{\wedge} \mathcal{T}_{\bar{\varepsilon}} + \sum_{(S,S'):S\cup S'=X} \mathcal{T}_{S,\bar{\varepsilon}} \overset{+}{\wedge} \mathcal{T}_{S',\bar{\varepsilon}} . \tag{4.10}$$

◀

On rappelle que les séries $L(z)$, $T_{\varepsilon}(z)$, et par conséquent $T_{\bar{\varepsilon}}(z)$ aussi, sont connues et ont été calculées dans la section 4.2.1.

On observe alors que le système sur les séries génératrices déduit de l'équation 4.10 est triangulaire : l'équation définissant $y_{X,\bar{\varepsilon}}(z)$ est une équation du second degré en $y_{X,\bar{\varepsilon}}(z)$, dont les coefficients ne dépendent que des fonctions $y_{S,\bar{\varepsilon}}$ pour $S \subsetneq X$.

Pour les équations définissant les $y_{X,\varepsilon}(z)$, le système du Théorème 4.9 était déjà triangulaire pour les séries génératrices reconnaissant ε : nous avions déjà une équation du second degré en $y_{X,\varepsilon}(z)$, dont les coefficients dépendaient uniquement de $y_{S,\bar{\varepsilon}}$ avec $S \subsetneq X$, et de $y_{S,\bar{\varepsilon}}$ avec $S \subseteq X$.

Nous pouvons donc à chaque étape résoudre de façon exacte le système triangulaire et obtenir chacune des fonctions génératrices $y_{X,\bar{\varepsilon}}(z)$, puis obtenir chaque $y_{X,\varepsilon}(z)$, et enfin $R(z)$ en utilisant l'équation provenant de la spécification de $\mathcal{R}$ dans l'équation (4.6) :

$$R(z) = zy_{\Sigma,\bar{\varepsilon}}(z) + y_{\Sigma,\varepsilon}(z) + 2(L(z) + T_{\varepsilon}(z))R(z) - 2R(z)^2 . \tag{4.11}$$

Si pouvoir résoudre de façon exacte un tel système de séries génératrices est appréciable, les séries solutions ne sont pas forcément très lisibles pour un être humain. Par exemple, pour $\Sigma = \{a, b\}$, les expressions deviennent déjà grandes : $y_{\Sigma,\bar{\varepsilon}}(z) = \frac{1}{4z}\Big(-\sqrt{\Delta(z)} + 2\sqrt{(2z+2)\sqrt{\Delta(z)} - 6z^2 + 2} - \sqrt{(2z+2)\sqrt{\Delta(z)} + 10z^2 + 2} - z - 1\Big)$, avec $\Delta(z) = -23z^2 - 2z + 1$. Néanmoins ce système spécialisé en $z = \rho$ donne aussi un système triangulaire satisfait par $\boldsymbol{y}(\rho)$, qui se prête bien à la résolution exacte ou approchée.

Procédure pour calculer $\boldsymbol{y}(\rho)$. Nous calculons d'abord $L(\rho)$, $T_\varepsilon(\rho)$ et $T_{\overline{\varepsilon}}(\rho)$ comme expliqué en Section 4.2.1. Ensuite, étant donné le système triangulaire décrit plus haut, nous calculons les valeurs de chaque $y_{X,\overline{\varepsilon}}(\rho)$ pour tout $X \subseteq \Sigma$ par programmation dynamique. Ensuite, nous calculons de même chaque $y_{X,\varepsilon}(\rho)$ pour tout $X \subseteq \Sigma$. Chaque étape de l'algorithme demande d'effectuer des opérations simples (somme, produit, et une seule racine carrée) sur des valeurs précédemment calculées. Enfin $R(\rho)$ est calculé à l'aide de l'Eq. (4.11), et $y_G(\rho) = y_{\Sigma,\varepsilon}(\rho) - R(\rho)$.

4.4.2 Vecteur propre associé aux probabilités limites

Le vecteur $\boldsymbol{g_y}(\rho)$ est caractérisé par un système d'équations linéaires.

Proposition 4.23 .
▶ Le vecteur $\boldsymbol{g_y}(\rho)$ est un vecteur propre pour la valeur propre $\lambda = 1$ de la matrice Jacobienne $\mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{\Phi}](\rho; \boldsymbol{y}(\rho))$:

$$\mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{\Phi}](\rho; \boldsymbol{y}(\rho)) \cdot \boldsymbol{g_y}(\rho) = \boldsymbol{g_y}(\rho)\,.$$

De plus, le sous-espace propre associé à la valeur propre $\lambda = 1$ est de dimension 1 et $\boldsymbol{g_y}(\rho)$ est le seul vecteur propre satisfaisant $\|\boldsymbol{g_y}(\rho)\|_1 = g_L(\rho)$. ◀

Démonstration. En dérivant par rapport à z l'équation $\boldsymbol{y}(z) = \boldsymbol{\Phi}(z; \boldsymbol{y}(z))$, nous obtenons $\boldsymbol{y}'(z) = \mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{\Phi}](z; \boldsymbol{y}(z))\boldsymbol{y}'(z) + \partial_z \boldsymbol{\Phi}(z; \boldsymbol{y}(z))$. Comme pour toute coordonnée i, $y_i(z) = y_i(\rho) + o(1)$ et $y_i'(z) = \frac{g_i(\rho)}{2\rho}(1 - z/\rho)^{-1/2} + O(1)$ en $z = \rho$, on identifie comme on l'a déjà fait les termes en $(1 - z/\rho)^{-1/2}$, ce qui donne :

$$\mathtt{Jac}_{\boldsymbol{y}}[\boldsymbol{\Phi}](\rho; \boldsymbol{y}(\rho)) \cdot \boldsymbol{g_y}(\rho) = \boldsymbol{g_y}(\rho)\,.$$

La dimension du sous-espace propre est une conséquence du théorème de Perron-Frobenius, introduit au chapitre 2 (voir Théorème 2.56 et Proposition 2.54). ■

Le calcul des $\boldsymbol{g_y}(\rho)$ ne sert pas uniquement au calcul de la taille moyenne après réduction ; il apporte aussi des informations sur la proportion d'expressions régulières universelles, en appliquant le Théorème 4.15. En particulier, sur un alphabet à deux lettres, nous obtenons $\lim_n \Pr_n(\mathcal{R}) \doteq 0.310122\ldots$, et $\lim_n \Pr_n(\mathcal{T}_{\Sigma,\varepsilon}) \doteq 0.457051\ldots$ Nous en déduisons un encadrement de la proportion d'expressions universelles. Les calculs effectués sur différentes tailles d'alphabet sont présentés dans la proposition suivante :

Nous n'avons cependant pas démontré que cette proportion d'expressions universelles converge lorsque $n \to \infty$.

Proposition 4.24 .
▶ Pour n suffisamment grand, la proportion $\Pr_n(\mathtt{univ.})$ d'expressions régulières universelles sur un alphabet à k lettres appartient à l'intervalle :

k	2	3	4	5
intervalle	$[31\%, 46\%]$	$[13\%, 27\%]$	$[6.2\%, 15\%]$	$[2.8\%, 7.7\%]$

◀

4.4.3 Réduction de la taille du système et calcul de la taille réduite

Le système combinatoire du Théorème 4.9 décrivant les classes d'arbres pertinentes pour l'étude de la réduction est de taille exponentielle en $k = |\Sigma|$. Lorsqu'on le traduit

en système sur les séries génératrices, on obtient un système de $1 + 2^{k+1}$ équations (avec $R(z)$ et $T_G(z)$).

En fait, certaines classes ont les mêmes séries génératrices, par un argument de symétrie. En effet, si $X, Y \subseteq \Sigma$ ont le même cardinal $s = \mathrm{card}(X) = \mathrm{card}(Y)$, alors $y_{X,\varepsilon}(z) = y_{Y,\varepsilon}(z) := y_{s,\varepsilon}(z)$ et $y_{X,\bar{\varepsilon}}(z) = y_{Y,\bar{\varepsilon}}(z) := y_{s,\bar{\varepsilon}}(z)$. Cela peut se voir en prenant une permutation des lettres de Σ envoyant X sur Y, qui envoie de façon bijective $\mathcal{T}_{X,\varepsilon}$ (resp. $\mathcal{T}_{X,\bar{\varepsilon}}$) sur $\mathcal{T}_{Y,\varepsilon}$ (resp. $\mathcal{T}_{Y,\bar{\varepsilon}}$).

Ainsi, si on identifie les séries génératrices qui sont égales, le système est équivalent à un système à seulement $1 + 2(k + 1)$ équations. Si le système exponentiel avait l'avantage d'être une traduction directe du système combinatoire, pour les calculs numériques, le système de taille linéaire en k est bien plus pratique et permet d'éviter de recalculer sans arrêt les mêmes valeurs :

Proposition 4.25.
▶ Pour $0 \leqslant i \leqslant k$, nous avons le système :

$$y_{i,\varepsilon}(z) = z\mathbf{1}_{i=0} + zy_{i,\bar{\varepsilon}}(z) + zy_{i,\varepsilon}(z) + 2z\sum_{l=0}^{i}\sum_{j=0}^{l}\binom{i}{l}\binom{l}{j}y_{l,\varepsilon}(z)y_{i+j-l,\varepsilon}(z)$$
$$+ 2z\sum_{l=0}^{i}\sum_{j=0}^{l}\binom{i}{l}\binom{l}{j}y_{l,\varepsilon}(z)y_{i+j-l,\bar{\varepsilon}}(z)$$
$$y_{i,\bar{\varepsilon}}(z) = z\mathbf{1}_{i=1} + z\mathbf{1}_{i=0}\left(\sum_{j=0}^{k}\binom{k}{j}y_{j,\bar{\varepsilon}}(z)\right)^2 + z\sum_{l=0}^{i}\sum_{j=0}^{l}\binom{i}{l}\binom{l}{j}y_{l,\bar{\varepsilon}}(z)y_{i+j-l,\bar{\varepsilon}}(z)$$
$$+ 2zy_{i,\bar{\varepsilon}}(z)\sum_{j=0}^{k}\binom{k}{j}y_{j,\varepsilon}(z)$$

◀

Ce système s'obtient simplement à partir du système exponentiel du Théorème 4.9 en rassemblant les séries égales, et en éliminant les lignes répétées. Il faut bien voir qu'il s'agit simplement d'une réduction des redondances du système originel, si bien que toutes les techniques présentées dans les sections précédentes s'appliquent (forte connexité, Drmota, triangularisation, etc.).

Les seuls ajustements concernent la multiplicité des séries, lorsqu'on regroupe les valeurs calculées. Cela intervient à deux endroits :

— Lors de la triangularisation, il ne faut pas oublier les termes binomiaux : $L(z) = \sum_{j=0}^{k}\binom{k}{j}(y_{j,\bar{\varepsilon}}(z) + y_{j,\bar{\varepsilon}}(z))$ et $T_{\varepsilon}(z) = \sum_{j=0}^{k}\binom{k}{j}y_{j,\bar{\varepsilon}}(z)$.
— L'expression de la taille réduite doit aussi être ajustée avec les multiplicités. Dans l'équation (4.9), il faut remplacer $\|\boldsymbol{g}_{Q\tilde{\boldsymbol{y}}}(\rho)\|_1$ par $\sum_{j=0}^{k}\binom{k}{j}g_{Q\tilde{y}_j}(\rho)$.

À partir de ce système et des sections précédentes, nous avons pu calculer numériquement $\boldsymbol{y}(\rho)$, $g_{\boldsymbol{y}}(\rho)$ et $Q\tilde{\boldsymbol{y}}(\rho)$, pour les rentrer dans l'équation de la Proposition 4.21, et enfin calculer $g_{Q\tilde{\boldsymbol{y}}}(\rho)$. Nous obtenons finalement les valeurs suivantes pour la taille moyenne limite d'une expression régulière après réduction :

Théorème 4.26.
▶ Nous considérons l'ensemble des arbres d'expressions régulières pour un alphabet de taille fixée $|\Sigma| = k$, et σ la réduction en temps linéaire induite par les règles

de simplification de la Figure 4.2. Alors la taille réduite d'une expression aléatoire uniforme de taille n tend vers une constante quand n tend vers l'infini, qui est donnée dans le tableau suivant pour k allant de 2 à 5 :

$\lvert\Sigma\rvert$	2	3	4	5
$\lim \mathbb{E}_n[\lvert\sigma(T)\rvert]$	77.79724...	495.59151...	2 518.20513...	11 694.43727...

Table 4.3. Tailles moyennes asymptotiques après réduction

◀

Remarque 4.27 .
▶ Les calculs ont été effectués à l'aide de Maple, et sont disponibles en Appendice B.
◀

4.5 Conclusion

Dans ce chapitre nous avons étendu, dans le cas des expressions régulières, la règle de simplification induite par le fait que Σ^* soit absorbant pour l'union. Cela nous a permis de prédire des tailles asymptotiques moyennes après réduction bien plus petites qu'aux chapitres précédents. Nous avons également observé cette convergence en moyenne expérimentalement. Les courbes expérimentales confirment que le phénomène général de dégénérescence de la distribution uniforme n'est pas seulement théorique, mais s'observe en pratique sur des arbres de taille raisonnable. De plus, en expérimentant la réduction sur des expressions régulières de très grande taille, nous avons pu observer que la convergence est rapide, et que les tailles moyennes ne s'écartent en pratique pas beaucoup de la limite théorique.

Ce chapitre est issu de l'article de conférence [KR21a].

5

Éléments absorbants et expressions suivant la distribution ABR

5.1 Introduction

Nous quittons les spécificités des expressions régulières du chapitre précédent, et revenons sur la réduction générale étudiée tout au long des trois premiers chapitres de cette partie. Dans ce chapitre, la seule information sémantique que nous utiliserons sur les arbres est donc la présence d'un élément absorbant $\mathcal{P}$ pour un opérateur $\circledast$.

Dans les chapitres précédents, nous avons montré que la distribution uniforme n'était pas adaptée à la génération aléatoire d'arbres d'expressions, ni à l'analyse en moyenne des algorithmes qui les reçoivent en entrée, en raison de l'élément absorbant $\mathcal{P}$.

La question naturelle qui se pose après ce constat est : quelle distribution utiliser alors pour générer des expressions aléatoires ? Si on s'intéresse de plus près aux outils de génération de formules LTL utilisés en vérification, on se rend compte qu'ils génèrent des arbres aléatoires qui ne suivent pas la distribution uniforme. L'Algorithme 3 présente le pseudo-code utilisé par les outils *LTL-to-Büchi translator testbench* (`ltl2tgba`) de TCS [Tau00] (on peut citer aussi [DGV99] et l'outil `Spot` [DLLF+16] pour d'autres exemples).

On peut résumer la procédure comme suit : pour générer un arbre aléatoire d'expression LTL de taille n,
(1) si $n = 1$ alors on tire une feuille uniformément au hasard parmi toutes les feuilles possibles
(2) si $n = 2$, on tire un opérateur d'arité 1 uniformément au hasard, puis une feuille au hasard en suivant le cas $n = 1$
(3) si $n \geqslant 3$, alors on tire un opérateur uniformément au hasard parmi tous les opérateurs unaires et binaires. Si l'opérateur tiré est unaire, alors on construit récursivement un arbre aléatoire de taille $n - 1$; si l'opérateur est binaire, on tire uniformément au hasard la taille k du fils gauche entre 1 et $n - 1$, puis on génère aléatoirement, de façon récursive, un fils gauche de taille k et un fils droit de taille $n - 1 - k$.

```
1  function RandomFormula(n) :
2  if n = 1 then
3  |   p := random symbol in AP ∪ {⊤, ⊥};
4  |   return p;
5  else if n = 2 then
6  |   op := random operator in {¬, X, □, ◇};
7  |   f := RandomFormula(1);
8  |   return op f;
9  else
10 |   op := random operator in {¬, X, □, ◇, ∧, ∨, →, ↔, U, R};
11 |   if op in {¬, X, □, ◇} then
12 |   |   f := RandomFormula(n − 1);
13 |   |   return op f;
14 |   else
15 |   |   x := uniform integer in [1, n − 2];
16 |   |   f1 := RandomFormula(x);
17 |   |   f2 := RandomFormula(n − x − 1);
18 |   |   return (f1 op f2);
```

Algorithme 3 : Le pseudo-code utilisé par l'outil `ltbt` [Tau00, p.46] pour tirer au hasard une formule LTL aléatoire.

Figure 5.1. La taille moyenne d'un arbre uniforme de taille n est $\Theta(\sqrt{n})$.

BST-like en anglais, car un ABR est binaire par définition

La distribution qui correspond à cet algorithme de génération s'appelle la distribution *façon Arbre Binaire de Recherche*. Dans la suite, nous étudierons une généralisation de cet algorithme en donnant des poids (probabilités) différents pour chaque opérateur ; nous resterons cependant sur des arbres unaire-binaires, c'est-à-dire que nous ne considèrerons que des opérateurs d'arité 1 ou 2.

L'introduction à la partie développe plus en détail les différences entre les deux distributions

Si la distribution uniforme donne la même probabilité pour chaque arbre de taille n, ce n'est plus vrai pour la distribution ABR, qui favorise les arbres équilibrés. Ainsi, la forme typique d'un arbre aléatoire ABR et celle d'un arbre aléatoire uniforme sont complètement différentes (comparer les Figures 5.2 et 5.1).

Comme nous l'avions évoqué dans l'introduction de la partie, l'atout principal de l'Algorithme 3 est qu'il est extrêmement simple à implémenter. Il permet de générer efficacement des arbres en temps linéaire. Par ailleurs, il s'agit d'un modèle

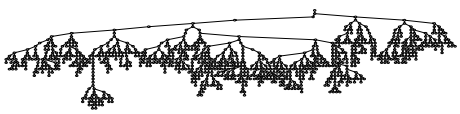


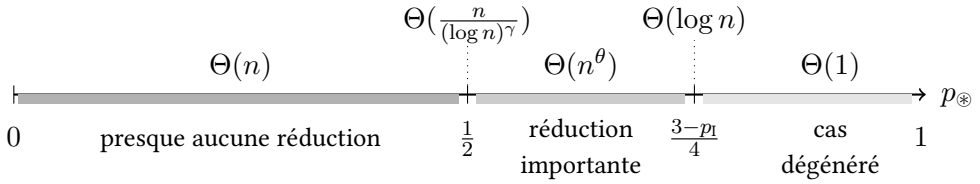
Figure 5.2. La taille moyenne d'un arbre ABR de taille n est $\Theta(\log n)$.

modulable, puisqu'on peut ajuster facilement les probabilités de chaque opérateur.

Dans ce chapitre, nous étudions l'effet de la réduction induite par la présence d'un élément absorbant sur les arbres aléatoires suivant la distribution ABR. Nous démontrons l'existence de deux valeurs seuils dans le choix de la probabilité d'apparition $p_\circledast$ de l'opérateur absorbant ; il y a ainsi cinq régimes asymptotiques différents pour la taille moyenne après réduction. Les trois principaux régimes des tailles moyennes après réduction sont montrés expérimentalement dans la Figure 5.3.

Théorème 5.1 (Résultat principal).
▶ Considérons une famille d'arbres d'expressions définie avec des opérateurs unaires et binaires. Nous supposons qu'il existe un élément absorbant $\mathcal{P}$, de taille au moins 3, pour un opérateur binaire $\circledast$. Nous introduisons la simplification qui consiste à remplacer de bas en haut tout arbre de racine $\circledast$ par $\mathcal{P}$, dès que l'un de ses fils est $\mathcal{P}$. Alors la taille réduite moyenne d'un arbre aléatoire de taille n suivant la distribution ABR, a un comportement asymptotique décrit par le schéma suivant, selon la valeur de la probabilité $p_\circledast$ d'apparition de l'opérateur absorbant :

Voir Section 5.2.2 pour une discussion sur la contrainte de taille.



Nous observons deux seuils critiques, en $p_\circledast = 1/2$ et en $p_\circledast = (3-p_\mathrm{I})/4$, où p_I est la probabilité de tirer un opérateur unaire. Nous obtenons ainsi cinq régimes différents, couvrant toutes les possibilités, d'une réduction très faible en $\Theta(n)$ à une réduction complète en $\Theta(1)$. Les exposants γ et θ dépendent des probabilités p_I et $p_\circledast$: ainsi $\gamma = \frac{2}{1-p_\mathrm{I}}$ et $\theta = 1 - \frac{4p_\circledast - 2}{1-p_\mathrm{I}}$. ◀

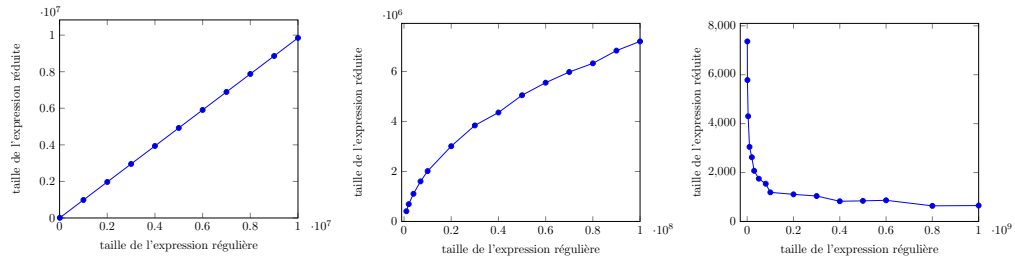


Figure 5.3. Les trois régimes principaux observés expérimentalement sur des expressions régulières à deux lettres, la moyenne étant faite sur 10 000 arbres générés pour chaque taille : (de gauche à droite) le régime linéaire ($p_+ = p_\star = p_\cdot = 1/3$), sous-linéaire ($p_+ = 19/29$, $p_\star = p_\cdot = 5/29$) et constant ($p_+ = 8/10$, $p_\star = p_\cdot = 1/10$).

Méthode. Contrairement au cas des arbres d'expression suivant la distribution uniforme, dans le cas de la distribution ABR, il n'y a pas de spécification combinatoire qui se traduise automatiquement en une équation sur des séries génératrices pertinentes pour analyser la taille après réduction. Nous commençons donc par identifier deux suites qui nous intéressent : la suite des probabilités des arbres entièrement

réductibles, et la suite des tailles moyennes réduites. Nous établissons alors des récurrences sur ces suites, qui se traduisent en équations différentielles sur leurs séries génératrices. Les techniques utilisées dans ce chapitre sont donc très différentes de celles des chapitres précédents, dans lesquels la spécification des arbres d'expression se traduisait en un système polynomial vérifié par les séries génératrices.

Comparaison avec la littérature. Comme nous l'avons annoncé dans l'introduction de la partie, la distribution ABR sur des arbres d'expressions a été très étudiée dans la littérature, notamment dans le cadre de l'étude d'algorithmes ou de réductions d'arbres. Dans [SCFC06], les auteurs se sont intéressés à des arbres binaires suivant la distribution ABR, dont les feuilles sont étiquetées par a ou b avec probabilité $1/2$, et dont les noeuds internes sont étiquetés par un unique symbole $\circ$: ils démontrent que si les noeuds internes $\circ$ sont idempotents (resp. nilpotents), alors la taille des arbres suivant la distribution ABR, après réduction par la règle d'idempotence (resp. de nilpotence), est linéaire en moyenne : cette réduction réduit peu les arbres, tout comme ce qui était observé dans le cas uniforme [CFCS90, NT04]. Cependant, les comportements pour la distribution uniforme et ABR diffèrent considérablement dans d'autres situations : par exemple, tester l'égalité de deux arbres binaires non étiquetés de taille totale n se fait en $O(1)$ en moyenne pour la distribution uniforme, mais est en $\Theta(\log(n))$ en moyenne pour la distribution ABR [Mar91] ; la taille moyenne de l'intersection de deux arbres binaires uniformes de taille totale n est en $O(1)$ en moyenne, contre du $O(n^{2\sqrt{2}-2}/\log(n))$ pour la distribution ABR [BYCDM92]. Je renvoie le lecteur à [CDM91] pour un survey sur les nombreuses statistiques qui différencient les deux distributions, ainsi que les méthodes usuelles pour les capturer. D'un point de vue plus technique, les formes des équations différentielles que nous allons retrouver dans ce chapitre (équations de Riccati) sont souvent rencontrées avec la distribution ABR, que ce soit pour l'étude de réductions [SCFC06] ou dans le contexte de la recherche de motifs [SCFC94, FGM97].
Le contexte de ce chapitre est un peu différent des travaux cités précédemment, dans le sens où nous étudions une distribution ABR adaptée aux expressions. Ainsi, nous n'autorisons pas de fils vide sous un noeud interne, et nous considérons des arbres unaire-binaires, pour autoriser des noeuds d'arité 1. De plus, la distribution que nous considérons est paramétrable : tous les noeuds des arbres sont étiquetés par des symboles, et ont une probabilité qui dépend de leur symbole. Notre cadre d'étude est ainsi plus proche de celui étudié par [CDM93] dans le cadre des arbres m-aires équilibrés.

5.2 Modèle, définitions et probabilité des entièrement réductibles

Nous introduisons en détail les arbres unaire-binaires que nous allons étudier, ainsi que leur distribution.

5.2.1 Le modèle des expressions ABR

Dans ce chapitre nous nous intéressons à des arbres d'expressions utilisant des opérateurs d'arité au plus 2, décrits par une équation unidimensionnelle.

En reprenant les notations du chapitre 2, nous considérons S un ensemble *fini*

d'opérateurs, et $a : S \to \{0,1,2\}$ une fonction d'arité associée, qui à chaque opérateur associe une arité inférieure à deux. Ainsi nous pouvons écrire $S = \mathcal{A}_0 \cup \mathcal{A}_1 \cup \mathcal{A}_2$, où $\mathcal{A}_0, \mathcal{A}_1$, et $\mathcal{A}_2$ correspondent respectivement aux feuilles, aux opérateurs unaires et aux opérateurs binaires de S. Rappelons ainsi que pour l'exemple des expressions régulières sur $\{a,b\}$, nous avons $\mathcal{A}_0 = \{\varepsilon, a, b\}$, $\mathcal{A}_1 = \{\star\}$ et $\mathcal{A}_2 = \{\cdot, +\}$.

Dans la suite du chapitre nous supposerons qu'aucun des ensembles $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2$ n'est vide : il y a donc des feuilles, des opérateurs unaires et binaires. L'ensemble d'arbres d'expressions étudié dans ce chapitre est $\mathcal{T}(S,a)$ tout entier, qui est constitué d'arbres unaire-binaires étiquetés.

Le cas $\mathcal{A}_1 = \emptyset$ sera traité en section 5.5.

La distribution sur $\mathcal{T}(S,a)$ que nous considérons dans ce chapitre est la distribution ABR (par opposition à la distribution uniforme considérée jusqu'alors). Pour $n \in \mathbb{N}$, nous notons $\mathcal{E}_n$ l'ensemble des expressions de taille n.

Probabilités des opérateurs. Pour définir le modèle ABR, nous associons à chaque feuille a de $\mathcal{A}_0$ une probabilité, notée p_a, de telle sorte que $(p_a)_{a\in\mathcal{A}_0}$ soit une distribution de probabilité, et ainsi $\sum_{a\in\mathcal{A}_0} p_a = 1$. De même, du côté des opérateurs unaires ou binaires, nous associons à chaque opérateur op de $\mathcal{A}_{\mathtt{ops}} := \mathcal{A}_1 \cup \mathcal{A}_2$ une probabilité p_{op}, telle que $\sum_{op\in\mathcal{A}_{\mathtt{ops}}} p_{op} = 1$.

Notons p_{I} la probabilité totale des opérateurs unaires, i.e., $p_{\mathrm{I}} = \sum_{op_1\in\mathcal{A}_1} p_{op_1}$.

Définition 5.2 (Arbre d'expression ABR).

▶ Un arbre aléatoire d'expression ABR de taille $n \in \mathbb{N}^*$ se construit récursivement de la façon suivante :

- si $n = 1$, on tire une feuille dans $\mathcal{A}_0$ avec la distribution $(p_a)_{a\in\mathcal{A}_0}$.
- si $n = 2$, on tire un opérateur unaire op_1 selon la distribution normalisée $\left(\frac{1}{p_{\mathrm{I}}} p_{op_1}\right)_{op_1\in\mathcal{A}_1}$, puis on construit indépendamment une feuille $a_0 \in \mathcal{A}_0$ de taille 1, et on retourne $\overset{op_1}{|}_{a_0}$.
- Si $n \geqslant 3$, on tire un opérateur $\oplus \in \mathcal{A}_{\mathtt{ops}}$ avec la distribution $(p_{op})_{op\in\mathcal{A}_{\mathtt{ops}}}$.
  - ($\star$) Si l'opérateur tiré est un opérateur unaire $\oplus \in \mathcal{A}_1$, on construit de façon récursive et indépendamment un arbre T de taille $n-1$ et on retourne $\overset{\oplus}{|}_{T}$.
  - ($\star\star$) Sinon, l'opérateur tiré est un opérateur binaire $\oplus \in \mathcal{A}_2$. Dans ce cas on tire une taille k uniformément au hasard dans $\{1,\dots,n-2\}$. On construit indépendamment deux arbres T_L et T_R de tailles respectives k et $n-1-k$. Enfin on retourne $\overset{\oplus}{/\!\backslash}_{T_L\ T_R}$.

◀

Pour que le cas $n = 2$ de la Définition 5.2 soit bien défini, nous supposons dans tout le chapitre que $p_{\mathrm{I}} > 0$. Si $p_{\mathrm{I}} = 0$, nous pouvons toujours définir sur $\mathcal{T}(S,a)$ une distribution ABR, mais dans ce cas la procédure ne peut produire que des arbres binaires de taille impaire. Cette restriction d'imposer $p_{\mathrm{I}} > 0$ dans un premier temps n'est pas contraignante, les résultats sont les mêmes si $p_{\mathrm{I}} = 0$ à la différence près qu'il faudra adapter un peu les calculs (voir Section 5.5).

La procédure de la Définition 5.2 définit une distribution de probabilité sur les arbres d'expression : la probabilité $\Pr_n(T)$ d'un arbre T de taille n est la probabilité que la procédure retourne cet arbre en ayant reçu en entrée la taille n. Cette distribution n'est pas uniforme, comme on peut le voir à la Figure 5.4.

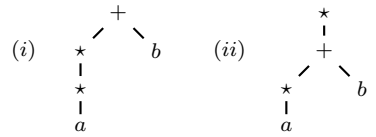


Figure 5.4. Exemple de deux arbres de taille $n = 5$ qui ont des probabilités différentes, peu importe le choix (non nul) de $(p_+, p_\star, p_a, p_b)$. Leurs probabilités sont respectivement $\frac{1}{3}p_+p_\star p_a p_b$ et $\frac{1}{2}p_\star p_+ p_a p_b$.

5.2.2 Élément absorbant et restriction technique

Nous nous intéressons à la réduction par élément absorbant déjà étudiée aux chapitres 2 et 3 dans le cas de la distribution uniforme. Pour la suite, nous fixons :

- un opérateur binaire $\circledast \in \mathcal{A}_2$ appelé opérateur absorbant
- un arbre $\mathcal{P}$ absorbant pour $\circledast$, de taille $s := |\mathcal{P}|$.

On notait p la taille de $\mathcal{P}$ jusqu'à présent, mais la présence de probabilités dans ce chapitre nous a incité à changer de notation.

Nous faisons de plus les hypothèses de travail suivantes :

- nous supposons que $\gamma_s := \Pr_s(\mathcal{P}) > 0$, c'est-à-dire que $\mathcal{P}$ a une probabilité non nulle d'être généré, sinon nous pouvons déjà annoncer sans calcul qu'il n'y aura pas de réduction ;
- nous supposons que $p_\circledast > 0$, c'est-à-dire que l'opérateur absorbant apparaît bien dans les arbres générés, sinon il n'y a pas non plus de réduction possible ;
- pour des raisons techniques, nous imposerons aussi $s \geqslant 3$. Cela interdit les arbres absorbants trop petits ($\bot$ pour $\wedge$ par exemple), mais ce n'est pas une grande restriction, car nous pouvons toujours construire à partir d'un arbre absorbant de taille $s \in \{1, 2\}$ un autre arbre absorbant de taille $s + 2 \geqslant 3$ en considérant $\mathcal{P}' = \overset{\circledast}{\underset{\mathcal{P}\ a}{/\backslash}}$, avec a une feuille telle que $p_a > 0$ (par exemple, si $\bot$ est absorbant pour $\wedge$, alors $\overset{\wedge}{\underset{\top\ \bot}{/\backslash}}$ est de taille 3 et est absorbant pour $\wedge$) ; de plus $\mathcal{P}'$ vérifie bien que $\Pr_{s+2}(\mathcal{P}') = p_\circledast \frac{1}{s}\gamma_s p_a > 0$. La réduction induite par $\mathcal{P}'$ est une sous-réduction de celle induite par $\mathcal{P}$, si bien que les expressions en Θ obtenus dans ce chapitre pour la réduction avec $\mathcal{P}'$ donneront des bornes en O pour la réduction avec $\mathcal{P}$.

La réduction σ étudiée est la même que dans les chapitres 2 et 3, et consiste à appliquer de bas en haut la substitution :

$$\overset{\circledast}{\underset{T_1 \quad T_2}{/\ \backslash}} \rightsquigarrow \mathcal{P}\,, \text{ dès que } T_1 \text{ ou } T_2 \text{ est égal à } \mathcal{P}.$$

Rappelons qu'un arbre d'expression T est dit *entièrement réductible* si $\sigma(T) = \mathcal{P}$.

5.2.3 Récurrences pour l'espérance de la taille après réduction

Dans les chapitres précédents, la spécification des arbres se transformait automatiquement en une équation sur la série génératrice adéquate pour étudier la distribution uniforme. Dans le cas de la distribution ABR, les arbres de même taille n'ont pas tous la même probabilité, il ne suffit donc pas de décrire les arbres possibles pour les compter, il faut aussi prendre en compte les probabilités qui entrent en jeu. Nous allons donc procéder *à la main*, en identifiant les suites qui nous intéressent, en établissant des relations de récurrence sur ces suites, et enfin en utilisant ces récurrences pour obtenir des équations sur les séries génératrices associées. Cette approche *à la main* est courante lorsque la distribution sur la classe combinatoire étudiée n'est pas uniforme [FS09, PSS12]).

Définition 5.3 .
▶ Pour $n \in \mathbb{N}$, nous définissons les suites suivantes :

— pour $k \leqslant n$, $p_{n,k}$ désigne la probabilité qu'un arbre de taille n suivant la distribution ABR ait une taille k après réduction par σ :

$$p_{n,k} = \sum_{T\,:\,|\sigma(T)|=k} \Pr_n(T)$$

— e_n désigne l'espérance de la taille après réduction des arbres aléatoires de taille n suivant la distribution ABR [1]. Par définition de l'espérance :

$$e_n := \sum_{k=0}^{n} k \cdot p_{n,k}$$

— enfin γ_n désigne la probabilité qu'un arbre d'expression aléatoire de taille n soit entièrement réductible.

Par ailleurs, nous notons $p_{\text{II}} := 1 - p_{\text{I}} - p_{\circledast}$ la probabilité de tirer un opérateur binaire différent de $\circledast$ lors de la procédure de tirage des arbres ABR. ◀

Comme dans les chapitres précédents, le comportement des arbres entièrement réductibles est fondamental pour comprendre la réduction. Nous démontrons dans un premier temps une relation de récurrence satisfaite par γ_n.

Proposition 5.4 (Probabilité des entièrement réductibles).
▶ La suite (γ_n) vérifie la relation de récurrence

$$\gamma_{n+1} = p_{\circledast} \cdot \frac{1}{n-1} \sum_{k=1}^{n-1} (\gamma_k + \gamma_{n-k} - \gamma_k\gamma_{n-k}) \ , \quad \text{pour tout } n \geqslant s, \tag{5.1}$$

avec $\gamma_n = 0$ pour $n < s$ et $\gamma_s = \Pr_s(\mathcal{P})$. ◀

Démonstration. Un arbre de taille $n+1 > s$ (donc différent de $\mathcal{P}$) est entièrement réductible si et seulement si il commence par $\circledast$ et l'un de ses fils est entièrement réductible.

Si nous considérons les arbres entièrement réductibles avec un fils gauche de taille k, leur probabilité est donc $\frac{p_{\circledast}}{n-1}(\gamma_k + (1-\gamma_k)\gamma_{n-k})$. Nous obtenons ainsi la formule annoncée. ■

Remarque 5.5 (Récurrence pour tout n).
▶ En remarquant que $\gamma_n = 0$ pour $n < s$, nous pouvons en fait écrire pour tout $n \in \mathbb{N}$:

$$(n-1)\gamma_{n+1}\mathbf{1}_{n+1\neq s} = p_{\circledast} \cdot \sum_{k=1}^{n-1} (\gamma_k + \gamma_{n-k} - \gamma_k\gamma_{n-k}) \ ,$$

où $\mathbf{1}_{n+1\neq s}$ vaut 1 si $n+1 \neq s$ et 0 sinon. ◀

1. Dans toute la suite, nous dirons juste aléatoire, sans repréciser suivant la distribution ABR

Remarque 5.6 (Limite possible de γ_n).
▶ Nous montrerons dans la suite du chapitre que la suite $(\gamma_n)_{n\geqslant 1}$ converge (Théorème 5.15). Nous pouvons déjà remarquer que si $(\gamma_n)_n$ converge, il n'y a qu'un nombre restreint de candidats pour sa limite $L = \lim \gamma_n$. En effet, les moyennes de Cesàro impliquent que si une suite $(a_n)_{n\geqslant 1}$ converge vers un réel L, alors $\lim_n \frac{1}{n}\sum_{k=1}^n a_k = L$, et $\lim_n \frac{1}{n}\sum_{k=1}^n a_k\, a_{n+1-k} = L^2$. À partir de l'Eq. (5.1), et en supposant que γ_n converge, nous en déduisons que sa limite vérifie l'équation

$$L = p_{\circledast}\cdot(2L - L^2)\,.$$

Ainsi si la limite existe, elle ne peut être que 0 ou $\gamma_\infty := 2 - 1/p_{\circledast}$. Si $p_{\circledast} < 1/2$, alors $\gamma_\infty < 0$ et donc la limite ne peut être que $L = 0$. Pour $p_{\circledast} > 1/2$, le Théorème 5.15 montrera que $L = \gamma_\infty$. Nous voyons ainsi déjà apparaître des phénomènes de seuil selon la valeur de $p_{\circledast}$ par rapport à $\frac{1}{2}$. ◀

La proposition suivante vient de la nature récursive de la génération des arbres aléatoires et relie la suite γ_n à l'espérance des tailles après réduction :

Proposition 5.7.
▶ La suite (e_n) des espérances des tailles après réduction satisfait pour tout $n > 1$ la récurrence suivante :

$$\begin{aligned} e_{n+1} = 1 &+ (s-1)\gamma_{n+1}\mathbf{1}_{n+1\neq s} + p_{\mathrm{I}}e_n \\ &+ \frac{2p_{\mathrm{II}}}{n-1}\sum_{j=1}^{n-1} e_j + \frac{2p_{\circledast}}{n-1}\sum_{j=1}^{n-1}(e_j - s\gamma_j)(1-\gamma_{n-j})\,. \end{aligned} \tag{5.2}$$

◀

Démonstration. Nous cherchons dans un premier temps, pour $n > 1$ et k fixés, une récurrence sur $p_{n,k}$. Si nous regardons la probabilité des arbres de taille $n+1$ dont la réduction est de taille k :

- Les arbres qui commencent par un opérateur unaire ont une probabilité $p_{\mathrm{I}}p_{n,k-1}$, car leur racine reste après réduction
- Les arbres qui commencent par un opérateur binaire autre que $p_{\circledast}$ ont une probabilité $p_{\mathrm{II}}\frac{1}{n-1}\sum_{j=1}^{n-1}\sum_{\ell_1+\ell_2=k-1} p_{j,\ell_1}p_{n-j,\ell_2}$.
- Les arbres qui commencent par $p_{\circledast}$ mais n'ont aucun fils entièrement réductible ont une probabilité $p_{\circledast}\frac{1}{n-1}\sum_{j=1}^{n-1}\sum_{\ell_1+\ell_2=k-1}(p_{j,\ell_1} - \mathbf{1}_{\ell_1=s}\gamma_j)(p_{n-j,\ell_2} - \mathbf{1}_{\ell_2=s}\gamma_{n-j})$
- Enfin, si $k = s$, les arbres entièrement réductibles ont une probabilité $\gamma_{n+1}\mathbf{1}_{n+1\neq s}$ (en effet, si $n+1 = s$, $\mathcal{P}$ a déjà été énuméré dans les cas précédents)

Nous avons donc pour $n > 1$ la récurrence suivante :

$$\begin{aligned} p_{n+1,k} =& \gamma_{n+1}\mathbf{1}_{n+1\neq s}\mathbf{1}_{k=s} + p_{\mathrm{I}}p_{n,k-1} + \frac{p_{\mathrm{II}}}{n-1}\sum_{j=1}^{n-1}\sum_{\ell_1+\ell_2=k-1} p_{j,\ell_1}p_{n-j,\ell_2} \\ &+ \frac{p_{\circledast}}{n-1}\sum_{j=1}^{n-1}\sum_{\ell_1+\ell_2=k-1}(p_{j,\ell_1} - \mathbf{1}_{\ell_1=s}\gamma_j)(p_{n-j,\ell_2} - \mathbf{1}_{\ell_2=s}\gamma_{n-j}) \end{aligned}$$

Nous introduisons alors les polynômes $F_n(u) = \sum_{k=0}^n p_{n,k}u^k$ pour tout $n > 1$. Nous remarquons dans un premier temps que $F_n(1) = 1$, que $F_n'(1) = e_n$. Par

ailleurs, comme $s \geqslant 3$, la somme peut partir de $k = 2$ lorsque $n \geqslant 2$. En multipliant la récurrence pour $p_{n,k}$ par u^k puis en sommant pour k allant de 2 à $n+1$, nous obtenons la récurrence suivante pour $F_n(u)$: *car $|\sigma(T)| \geqslant s$ si $|T| \geqslant s$*

$$F_{n+1}(u) = \gamma_{n+1}\mathbf{1}_{n+1\neq s}u^s + p_{\mathrm{I}}uF_n(u) + u\frac{p_{\mathrm{II}}}{n-1}\sum_{j=1}^{n-1}F_j(u)F_{n-j}(u) + u\frac{p_{\circledast}}{n-1}\sum_{j=1}^{n-1}\left(F_j(u)-\gamma_j u^s\right)\left(F_{n-j}(u)-\gamma_{n-j}u^s\right).$$

En dérivant alors cette équation en u, puis en évaluant en $u = 1$:

$$e_{n+1} = \gamma_{n+1}\mathbf{1}_{n+1\neq s}s + p_{\mathrm{I}} + p_{\mathrm{I}}e_n + p_{\mathrm{II}} + \frac{2p_{\mathrm{II}}}{n-1}\sum_{j=1}^{n-1}e_j + \frac{p_{\circledast}}{n-1}\sum_{j=1}^{n-1}(1-\gamma_j)(1-\gamma_{n-j}) + \frac{2p_{\circledast}}{n-1}\sum_{j=1}^{n-1}(e_j - s\gamma_j)(1-\gamma_{n-j}).$$

En développant la deuxième somme, nous faisons apparaître la récurrence de γ_n :

$$\frac{p_{\circledast}}{n-1}\sum_{j=1}^{n-1}(1-\gamma_j)(1-\gamma_{n-j}) = p_{\circledast} - \frac{p_{\circledast}}{n-1}\sum_{j=1}^{n-1}(\gamma_j + \gamma_{n-j} - \gamma_j\gamma_{n-j}).$$

Par la Remarque 5.5, $\frac{p_{\circledast}}{n-1}\sum_{j=1}^{n-1}(\gamma_j + \gamma_{n-j} - \gamma_j\gamma_{n-j}) = \gamma_{n+1}\mathbf{1}_{n+1\neq s}$. Enfin en utilisant la relation $p_{\mathrm{I}} + p_{\mathrm{II}} + p_{\circledast} = 1$, nous obtenons bien la relation de récurrence annoncée. ■

5.2.4 Feuille de route pour l'étude de la réduction

Le but de cette section est d'annoncer les grandes lignes de la démarche que nous avons entreprise pour étudier la suite (e_n).

Comme la récurrence satisfaite par (e_n) à la Proposition 5.7 dépend de la suite (γ_n), il faut d'abord étudier cette suite. Nous introduisons pour cela les deux séries génératrices :

$$A(z) := \sum_{n=0}^{\infty}\gamma_n z^n, \quad E(z) := \sum_{n=0}^{\infty}e_n z^n.$$

Les récurrences obtenues sur les suites vont se traduire en équations différentielles sur ces séries. En effet, la dérivée de la série $F(z) = \sum a_n z^n$ est $F'(z) = \sum(n+1)a_n z^n$. Donc en multipliant l'Équation (5.1) par $(n-1)z^n$ et en sommant sur $n \in \mathbb{N}$, nous introduisons la dérivée de $A(z)$. Le même phénomène apparaît pour l'équation portant sur $E(z)$. Ainsi les deux récurrences de la section précédente se traduisent en deux équations différentielles sur les fonctions génératrices, une équation de Riccati pour $A(z)$, et une équation différentielle linéaire pour $E(z)$, dont les coefficients font intervenir $A(z)$:

$$A'(z) = (s-2)\gamma_s z^{s-1} + \left(\frac{2}{z} + 2p_{\circledast}\frac{z}{1-z}\right)A(z) - p_{\circledast}\cdot(A(z))^2, \qquad (5.3)$$

et, pour une certaine fonction $F(x, y)$ qu'on explicitera plus tard,

$$E'(z) = F(z, A(z)) + \tfrac{1}{1-p_{\mathrm{I}}z}\left(\tfrac{2}{z} - p_{\mathrm{I}} + 2\,(1-p_{\mathrm{I}})\,\frac{z}{1-z} - 2p_{\circledast}A(z)\right)\cdot E(z)\,.$$

Comme l'équation portant sur $E(z)$ est linéaire, nous savons la résoudre, avec la méthode de variation des constantes [Apo69, Th. 6.1] : nous obtenons une expression de $E(z)$ dépendant de la fonction $A(z)$. Nous étudions alors plus précisément $A(z)$, et notamment son domaine d'analyticité. Nous prouvons dans la section 5.3 que $z = 1$ est une singularité dominante pour $A(z)$, et que la fonction se prolonge analytiquement au domaine $\Omega = \mathbb{C} \setminus [1, \infty)$. Par ricochet, nous obtiendrons le même résultat pour $E(z)$.

À partir de là, le cadre pour appliquer le théorème de Transfert est respecté, et il reste à étudier le comportement asymptotique de $E(z)$ au voisinage de sa singularité dominante, $z = 1$. La solution pour $E(z)$ est de la forme suivante :

$$E(z) \approx \frac{C}{(1-z)^2}\exp\left(-2p_{\circledast}\int_0^z \frac{A(w)}{1-p_{\mathrm{I}}w}dw\right)\times\left(2 + \int_0^z G(\zeta)\exp\left(2p_{\circledast}\int_0^\zeta \frac{A(w)}{1-p_{\mathrm{I}}w}dw\right)d\zeta\right)$$

lorsque $z \to 1$, pour une constante $C > 0$ et une fonction bornée $G(z)$.

Nous voyons ici que nous aurons besoin d'asymptotiques précis pour $A(z)$ en $z = 1$, qui puissent se transposer à l'intérieur des intégrales, à l'aide de théorèmes d'intégration singulière.

Analyse de $A(z)$ autour de sa singularité dominante. Dans un premier temps, nous transformons l'équation de Riccati (5.3) sastisfaite par $A(z)$ en une équation linéaire homogène du second ordre, par un changement classique de fonction inconnue $p_{\circledast}A(z) = v'(z)/v(z)$. Nous étudions ensuite la fonction $v(z)$ par la méthode de Frobenius pour obtenir le comportement local de $v(z)$ autour de sa singularité $z = 1$. Cette analyse est résumée dans la Proposition 5.14, qui montre la présence de 3 régimes pour $A(z)$, qui dépendent de la position de $p_{\circledast}$ par rapport à $1/2$. Comme conséquence directe de cette proposition, par le théorème de Transfert, nous montrons dans le Théorème 5.15 que γ_n tend vers 0 pour $p_{\circledast} \leqslant 1/2$ et tend vers la constante $\gamma_\infty > 0$ lorsque $p_{\circledast} > 1/2$. Les détails de cette partie sont expliqués dans la Section 5.3.

Analyse de $E(z)$ autour de sa singularité dominante. Les différents comportements de $A(z)$ de la Proposition 5.14 expliquent déjà le seuil en $1/2$. Un nouveau seuil apparaît à cause du terme suivant du développement asymptotique de $E(z)$:

$$2 + \int_0^z G(\zeta)\exp\left(2p_{\circledast}\int_0^\zeta \frac{A(w)}{1-p_{\mathrm{I}}w}dw\right)d\zeta\,.$$

Ce nouveau seuil correspond exactement au point où l'intégrale cesse d'être convergente lorsque $z \to 1$.

5.3 Étude des arbres entièrement réductibles

Dans cette section, nous étudions en détail la probabilité $\gamma_n = \Pr_n\{\sigma(T) = \mathcal{P}\}$ des arbres entièrement réductibles de taille n, qui apparaît dans la récurrence de e_n.

Le point de départ de l'étude est la récurrence de la Proposition 5.4 :

$$\gamma_{n+1} = p_\circledast \cdot \frac{1}{n-1} \sum_{k=1}^{n-1} (\gamma_k + \gamma_{n-k} - \gamma_k \gamma_{n-k}) \text{ pour tout } n \geqslant s. \tag{5.1}$$

5.3.1 Série génératrice et équation de Riccati

Le but de cette section est d'établir une équation différentielle satisfaite par $A(z) = \sum_{n=0}^{\infty} \gamma_n z^n$, et de l'utiliser pour déterminer notamment le domaine d'analyticité de $A(z)$. Comme $0 \leqslant \gamma_n \leqslant 1$, son rayon de convergence est au moins 1. La proposition suivante montre que c'est exactement 1 :

Proposition 5.8.
▶ Le rayon de convergence de $A(z)$ est exactement 1. ◀

Démonstration. Par l'absurde. Supposons que la série $\sum_k \gamma_k$ est convergente. Alors en appliquant l'inégalité $\gamma_k + \gamma_{n-k} - \gamma_k \gamma_{n-k} \geqslant \gamma_k$, valide pour tout k, à la récurrence de l'Eq (5.1), nous obtenons la minoration $\gamma_n \geqslant \frac{p_\circledast}{n-1} \sum_{k=1}^{n-1} \gamma_k = \Omega(1/n)$. C'est en contradiction avec la convergence de la série $\sum_k \gamma_k$, car la série harmonique diverge. ■

La proposition suivante montre que $A(z)$ vérifie une équation différentielle d'une certaine forme, appelée équation de Riccati, qui est souvent rencontrée avec la distribution ABR : les équations de Riccati interviennent par exemple dans les réductions d'arbres binaires de [SCFC06] par élément idempotent ou nilpotent, ou encore dans le contexte de la recherche de motifs [FGM97].

Proposition 5.9.
▶ La série $A(z)$ satisfait l'équation de Riccati :

$$A'(z) = (s-2)\gamma_s z^{s-1} + \left(\frac{2}{z} + 2p_\circledast \frac{z}{1-z}\right) A(z) - p_\circledast \cdot (A(z))^2 . \tag{5.3}$$

◀

Démonstration. Nous multiplions l'Équation 5.1 par $n-1 = (n+1)-2$, puis le tout par z^n et nous sommons sur tous les $n \geqslant s$. Nous obtenons alors l'équation

$$\sum_{n \geqslant s} (n+1)\gamma_{n+1} z^n - 2 \sum_{n \geqslant s} \gamma_{n+1} z^n = 2p_\circledast \sum_{n \geqslant s} \sum_{k=1}^{n-1} \gamma_k z^n - p_\circledast \sum_{n \geqslant s} \sum_{k=0}^{n} \gamma_k \gamma_{n-k} z^n .$$

En rappelant que pour $k < s$, $\gamma_k = 0$, nous reconnaissons à gauche $A(z)$ et sa dérivée, et à droite le produit de Cauchy de $A(z)$ avec $\frac{1}{1-z}$ et avec lui-même :

$$A'(z) - s\gamma_s z^{s-1} - \tfrac{2}{z}(A(z) - \gamma_s z^s) = 2p_\circledast \frac{z}{1-z} A(z) - p_\circledast A(z)^2 .$$

■

Pour étudier cette équation différentielle, nous effectuons le changement de fonction inconnue $v(z) = \exp\left(p_\circledast \int_0^z A(w)dw\right)$. En particulier $A(z) = 1/p_\circledast \cdot v'(z)/v(z)$. Ce changement de fonction inconnue est une méthode classique pour transformer une

Le symbole $\int_0^z$ signifie qu'on intègre sur le segment $[0, z]$ du plan complexe.

équation de Riccati en équation différentielle linéaire. Ainsi $v(z)$ vérifie l'équation différentielle linéaire d'ordre 2 :

$$v''(z) = p_{\circledast} \cdot (s-2)\gamma_s z^{s-1} v(z) + \left(\frac{2}{z} + 2p_{\circledast}\frac{z}{1-z}\right) v'(z)\,. \tag{5.4}$$

Nous remarquons que $v(z)$ est analytique dans le disque $|z| < 1$ car $A(z)$ l'est.

Les singularités et le comportement local des solutions d'équations différentielles linéaires de la forme de l'Eq. (5.4) sont bien comprises [Apo69, Sta80, Was18]. Dans la proposition suivante, nous montrons que $A(z)$ est analytique sur $\Omega = \mathbb{C} \setminus [1, \infty)$.

Proposition 5.10.

Domaine = ouvert connexe.

▶ La série entière $A(z)$ admet un prolongement analytique sur le domaine $\Omega = \mathbb{C} \setminus [1, \infty)$. En particulier, $z = 1$ est la seule singularité dominante de $A(z)$. ◀

Démonstration. La fonction $v(z)$ satisfait une équation différentielle linéaire dont les coefficients sont analytiques sur $\mathbb{C} \setminus \{0, 1\}$. Nous savons par la Proposition 5.8 qu'à la fois $A(z)$ et $v(z) = \exp\left(p_{\circledast} \int_0^z A(w)dw\right)$ sont analytiques sur le disque $|z| < 1$.

Voir annexe A.2.

Le théorème d'unicité des solutions des équations différentielles nous assure qu'il existe une unique solution analytique à l'équation (5.4) sur un domaine Ω_0, si les conditions suivantes sont respectées :

a. les conditions initiales sont données en un point $z_0 \in \Omega_0$,

b. les coefficients de l'équation différentielle sont analytiques sur Ω_0,

c. le coefficient de tête en face de $v''(z)$ ne s'annule pas sur Ω_0,

d. le domaine Ω_0 est simplement connexe (en pratique il suffit que Ω_0 soit étoilé).

Malheureusement, le coefficient $\frac{2}{z}$ nous demande de ruser un peu pour appliquer ce théorème ; nous allons donc découper $\mathbb{C} \setminus \{0, 1\}$ en plusieurs régions pour éviter la fausse singularité en $z = 0$:

— Nous savons déjà que $v(z)$ est analytique sur $D(0, 1) = \{z \,|\, |z| < 1\}$.

— Par ce qui précède, il existe une unique solution à l'équation (5.4) sur le domaine étoilé $\Omega_1 := \{\Im(z) > 0\}$, qui coïncide avec $v(z)$ et ses dérivées au point $z = i/2$. Par unicité du prolongement analytique, $v(z)$ est analytique sur $D(0, 1) \cup \Omega_1$.

— Le même argument appliqué à $\Omega_2 := \{\Im(z) < 0\}$ et $\Omega_3 := \{\Re(z) < 0\}$, montre que $v(z)$ est analytique sur $\Omega := \Omega_1 \cup \Omega_2 \cup \Omega_3 \cup D(0, 1) = \mathbb{C} \setminus [1, \infty)$.

Comme $v(z) \neq 0$ par définition (c'est une exponentielle), nous en déduisons que $A(z) = \frac{1}{p_{\circledast}} v'(z)/v(z)$ est aussi analytique sur Ω.

Comme le rayon de convergence de $A(z)$ est 1, par la Proposition 5.8, le théorème de Pringsheim permet de conclure que $z = 1$ est l'unique singularité dominante de $A(z)$. ■

5.3.2 Comportements asymptotiques des probabilités des arbres entièrement réductibles

Nous pouvons maintenant nous attaquer au comportement asymptotique de $v(z)$, où nous rappelons que $v(z)$ satisfait l'équation

$$v''(z) - \left(\frac{2}{z} + 2p_{\circledast}\frac{z}{1-z}\right) v'(z) - p_{\circledast} \cdot (s-2)\gamma_s z^{s-1} v(z) = 0\,. \tag{5.4}$$

Nous utilisons pour cela la méthode de Frobenius (voir [Apo69, pp.181-182] et [Tes12, Theorem 4.5]), que nous énonçons dans le cas des équations différentielles linéaires d'ordre 2.

Définition 5.11 (Singularité régulière et polynôme indiciel).
▶ Considérons une équation différentielle linéaire homogène du second ordre de la forme $y''(z) + d_1(z)y'(z) + d_2(z)y(z) = 0$, où $d_1(z)$ et $d_2(z)$ sont méromorphes sur un domaine étoilé $\tilde{\Omega}$.

Un complexe $\zeta \in \tilde{\Omega}$ est une *singularité régulière* pour l'équation si c'est une singularité de $d_1(z)$ ou $d_2(z)$, ou des deux, telle que les limites $\delta_j := \lim_{z\to\zeta}(z-\zeta)^j d_j(z)$ existent et sont finies, pour $j = 1$ et 2. Dans ce cas, le *polynôme indiciel* $\mathcal{I}(\theta)$ en la singularité régulière $z = \zeta$ est défini par $\mathcal{I}(\theta) = \theta(\theta-1) + \delta_1\theta + \delta_2$. ◀

Le théorème suivant est le théorème clef de la méthode de Frobenius, et explique comment le polynôme indiciel permet de trouver les asymptotiques des solutions de l'équation différentielle [2] :

Théorème 5.12 ([Apo69, pp.181-182], [Tes12, Theorem 4.5]).
▶ Considérons l'équation différentielle linéaire homogène d'ordre 2

$$y''(z) + d_1(z)y'(z) + d_2(z)y(z) = 0\,,$$

où $d_1(z)$ et $d_2(z)$ sont méromorphes sur un domaine étoilé $\tilde{\Omega}$, et ζ une singularité régulière de l'équation. Alors :

— Si les deux racines θ_1 et θ_2 du polynôme indiciel associé à ζ ne sont pas séparées d'un entier (y compris 0 pour les racines doubles), alors, dans un voisinage de ζ inclus dans $\tilde{\Omega}$, toute solution $y(z)$ est de la forme *Autrement dit $\theta_1 - \theta_2 \notin \mathbb{Z}$.*

$$c_1(\zeta-z)^{\theta_1}H_1(\zeta-z) + c_2(\zeta-z)^{\theta_2}H_2(\zeta-z)\,,$$

où $c_1, c_2 \in \mathbb{C}$, $H_1(z), H_2(z)$ sont analytiques en $z = 0$ et $H_1(0) \neq 0$, $H_2(0) \neq 0$.

— Si le polynôme indiciel a une racine double θ_0, alors dans un voisinage de ζ inclus dans $\tilde{\Omega}$, toute solution $y(z)$ est de la forme

$$(\zeta-z)^{\theta_0}(c_1H_1(\zeta-z) + c_2\log(\zeta-z)H_2(\zeta-z))\,,$$

où $c_1, c_2 \in \mathbb{C}$, $H_1(z), H_2(z)$ sont analytiques en $z = 0$ et $H_1(0) \neq 0$, $H_2(0) \neq 0$.

◀

Ce théorème nous permet de déduire le comportement asymptotique de $v(z)$ et de $v'(z)$ autour de $z = 1$:

Proposition 5.13 .
▶ Autour de $z = 1$, la fonction $v(z)$ s'écrit sous les formes suivantes :

— si $p_\circledast > 1/2$, alors $v(z) = c_1H_1(1-z) + c_2(1-z)^{1-2p_\circledast}H_2(1-z)$,

— si $p_\circledast = 1/2$, alors $v(z) = c_1H_1(1-z) + c_2\log\left(\frac{1}{1-z}\right)H_2(1-z)$,

2. Dans la littérature, le théorème est souvent annoncé avec $|\zeta - z|$, ce qui permet d'éviter de restreindre le domaine de définition ; ici nous utilisons $(\zeta - z)$ car nous avons choisi une détermination de $\log(1-z)$.

– si $p_\circledast < 1/2$, alors $v(z) = c_1 H_1(1-z) + c_2(1-z)^{1-2p_\circledast} H_2(1-z)$ avec $c_1 \neq 0$, où c_1, c_2, H_1, H_2 dépendent de $p_\circledast$, et pour les trois cas, $c_2 \neq 0$, $H_1(z)$ et $H_2(z)$ sont analytiques en $z = 0$, avec $H_1(0) = H_2(0) = 1$. ◀

Démonstration. Rappelons que $v(z)$ est analytique sur $\Omega = \mathbb{C} \setminus [1, +\infty)$. Comme nous nous intéressons uniquement au comportement en $z = 1$, qui la singularité dominante de $A(z)$, nous introduisons le domaine étoilé $\tilde{\Omega} = \Omega \cap D(3/4, 1/2)$, qui évite soigneusement $z = 0$. La preuve consiste à appliquer le Théorème 5.12 à l'Eq. (5.4), sur le domaine $\tilde{\Omega}$:

Avec $D(3/4, 1/2) := \{z \in \mathbb{C} \,|\, |z - 3/4| < 1/2\}$

$$v''(z) - \left(\frac{2}{z} + 2p_\circledast \frac{z}{1-z}\right) v'(z) - p_\circledast \cdot (s-2)\gamma_s z^{s-1} v(z) = 0\,. \tag{5.4}$$

Comme $0 \notin \tilde{\Omega}$, l'équation a une unique singularité dans $\tilde{\Omega}$, en $z = 1$, et le polynôme indiciel associé est $\mathcal{I}(\theta) = \theta(\theta-1) + 2p_\circledast\theta = \theta(\theta - 1 + 2p_\circledast)$. Les racines de ce polynôme sont $\theta_1 = 0$ and $\theta_2 = 1 - 2p_\circledast$.

En appliquant le Théorème 5.12, nous obtenons ainsi les formes annoncées dans la proposition. Les conditions sur les constantes viennent du fait que si $c_2 = 0$, alors $A(z)$ n'aurait pas de singularité en $z = 1$. De plus, si $c_1 = 0$ pour le cas $p_\circledast < 1/2$, alors $A(z) = \frac{1}{p_\circledast} v'(z)/v(z)$ satisfait $A(z) = \gamma_\infty/(1-z)$ autour de $z = 1$. Or, dans ce cas, $\gamma_\infty := (2p_\circledast - 1)/p_\circledast < 0$ tandis que $A(z)$ est positif sur $[0, 1)$, contradiction. ■

Il ne reste plus qu'à reporter l'expansion locale de $v(z)$ dans l'égalité $A(z) = \frac{1}{p_\circledast} v'(z)/v(z)$, autour de la singularité $z = 1$, pour obtenir le comportement de $A(z)$:

Proposition 5.14 .

▶ La série génératrice $A(z)$ de $(\gamma_n)_{n \geqslant 1}$ a les comportements asymptotiques suivants lorsque $z \to 1$ sur Ω :

– si $p_\circledast < \frac{1}{2}$, $A(z) \sim \frac{D}{(1-z)^{2p_\circledast}}$,

– si $p_\circledast = \frac{1}{2}$, $A(z) = \frac{2}{1-z}\left(\log\left(\frac{1}{1-z}\right)\right)^{-1} + O\left(\frac{1}{1-z}\left(\log\left(\frac{1}{1-z}\right)\right)^{-2}\right)$

– si $p_\circledast > \frac{1}{2}$, $A(z) = \frac{\gamma_\infty}{1-z} + O((1-z)^{2p_\circledast - 2})$,

où nous rappelons que $\gamma_\infty := (2p_\circledast - 1)/p_\circledast$ et où $D > 0$ est une constante qui dépend de $p_\circledast$ et de s. ◀

Démonstration. Comme annoncé, nous développons $v(z)$ autour de $z = 1$ dans l'égalité $A(z) = \frac{v'(z)}{p_\circledast v(z)}$.

– Pour $p_\circledast > \frac{1}{2}$. Rappelons que dans ce cas, par la Proposition 5.13, $v(z) = c_1 H_1(1-z) + c_2(1-z)^{1-2p_\circledast} H_2(1-z)$ au voisinage de $z = 1$ dans Ω, avec $H_1(z), H_2(z)$ analytiques en $z = 0$, $H_1(0) = H_2(0) = 1$ et $c_2 \neq 0$. Nous en déduisons que

$$v(z) = (1-z)^{1-2p_\circledast}\left(c_2 + O((z-1)^{2p_\circledast - 1})\right),$$
$$\text{et } v'(z) = (1-z)^{-2p_\circledast}\left(c_2(1-2p_\circledast) + O((z-1)^1)\right).$$

Ainsi, en normalisant :

$$A(z) = \frac{\gamma_\infty}{1-z} \times \frac{1 + O((z-1)^1)}{1 + O((z-1)^{2p_\circledast - 1})} = \frac{\gamma_\infty}{1-z} + O((1-z)^{2p_\circledast - 2})\,,$$

en remarquant que $\left(1 + O((1-z)^{2p_\circledast - 1})\right)^{-1} = 1 + O\left((1-z)^{2p_\circledast - 1}\right)$.

— Dans le cas où $p_\circledast < \frac{1}{2}$, alors $v(z) = c_1 H_1(1-z) + c_2(1-z)^{1-2p_\circledast} H_2(1-z)$ au voisinage de $z=1$ dans Ω, avec $H_1(z), H_2(z)$ analytiques en $z=0$, $H_1(0) = H_2(0) = 1$ et $c_1, c_2 \neq 0$.
Donc dans ce cas $v(z) = c_1 + O((1-z)^{1-2p_\circledast})$ lorsque $z \to 1$, avec $c_1 \neq 0$, tandis que $v'(z) = (z-1)^{-2p_\circledast}((1-2p_\circledast)c_2 + O((1-z)^{2p_\circledast}))$. Ainsi

$$A(z) \sim (\gamma_\infty c_2/c_1) \times (1-z)^{-2p_\circledast}\,.$$

— Enfin, si $p_\circledast = \frac{1}{2}$, la Proposition 5.13 implique que $v(z) = c_1 H_1(1-z) + c_2 \log\left(\frac{1}{1-z}\right) H_2(1-z)$ au voisinage de $z=1$ dans Ω, avec $H_1(z), H_2(z)$ analytiques en $z=0$, $H_1(0) = H_2(0) = 1$ et $c_2 \neq 0$. En dérivant cette égalité nous obtenons :

$$v'(z) = \frac{c_2}{1-z} + c_2 H_2'(0) \log\left(\tfrac{1}{1-z}\right) + O(1)\,,$$

tandis que $v(z) = c_2 \log\left(\frac{1}{1-z}\right) + O(1)$. En divisant l'un par l'autre :

$$A(z) = p_\circledast^{-1} \frac{1}{1-z}\left(\log\left(\tfrac{1}{1-z}\right)\right)^{-1} + O\left(\frac{1}{1-z}\left(\log\left(\tfrac{1}{1-z}\right)\right)^{-2}\right)$$

en utilisant le fait que $(1+O(r(z))^{-1} = 1 + O(r(z))$ si $r(z) \to 0$. ■

Comme corollaire de cette proposition, nous prouvons la convergence de la suite (γ_n) :

Théorème 5.15.
▶ La probabilité γ_n d'être entièrement réductible tend vers la constante $\gamma_\infty := (2p_\circledast - 1)/p_\circledast$ si $p_\circledast > \frac{1}{2}$ et vers zéro sinon. Plus précisément, lorsque $p_\circledast = \frac{1}{2}$, $\gamma_n \sim \frac{2}{\log n}$, tandis que lorsque $p_\circledast < \frac{1}{2}$, $\gamma_n \sim D \cdot n^{2p_\circledast - 1}/\Gamma(2p_\circledast)$, où D est la constante de la Proposition 5.14. ◀

Remarque 5.16.
▶ Une approche complémentaire du problème, par Pablo Rotondo, du cas $p_\circledast < 1/2$ permet d'exprimer la valeur de la constante D en fonction de $A(z)$, si $s \geqslant 3$:

$$D = e^{-2p_\circledast} \cdot \left((s-2)\gamma_s \int_0^1 t^{s-3}(1-t)^{2p_\circledast} e^{2p_\circledast t}dt - p_\circledast \int_0^1 (A(t))^2 (1-t)^{2p_\circledast} e^{2p_\circledast t}dt\right).$$

On en déduit notamment que $D \leqslant e^{-2p_\circledast}(s-2)\gamma_s \int_0^1 t^{s-3}(1-t)^{2p_\circledast} e^{2p_\circledast t}dt$. ◀

Pablo avait en effet réussi à aborder le problème à la main, dans le cas $p_\circledast < 1/2$. La méthode ne se généralisait pas aux autres cas, donc nous nous sommes intéressés aux équations différentielles et à la méthode de Frobenius.

5.4 Résultat principal : tailles moyennes des arbres après réduction

Le but de cette section est de démontrer le théorème principal de ce chapitre :

Théorème 5.17 (Comportement asymptotique des tailles moyennes e_n).
▶ Si la probabilité p_{I} des opérateurs unaires n'est pas 0, alors l'espérance e_n de la taille après réduction par σ d'un arbre d'expressions aléatoire de taille n suivant la distribution ABR, satisfait les propriétés suivantes :

— si $p_\circledast < 1/2$, alors $e_n \sim c_1\, n$;

— si $p_\circledast = 1/2$, alors $e_n \sim c_2\, n \log(n)^{-2/(1-p_\mathrm{I})}$;
— si $\frac{1}{2} < p_\circledast < \frac{3-p_\mathrm{I}}{4}$, alors $e_n \sim c_3\, n^{1-\frac{4p_\circledast-2}{1-p_\mathrm{I}}}$;
— si $p_\circledast = \frac{3-p_\mathrm{I}}{4}$, alors $e_n \sim c_4\, \log(n)$;
— si $p_\circledast > \frac{3-p_\mathrm{I}}{4}$, alors $e_n \to e_\infty$, où e_∞ est une constante strictement positive ;
avec $c_1, \dots, c_4$ des constantes positives non nulles. ◀

Remarque 5.18 .
▶ Attention, dans le théorème précédent, les constantes $e_\infty, c_1, \dots, c_4$ sont des constantes par rapport à n, mais *elles dépendent de* p_I, $p_\circledast$ *et* s. ◀

Ce théorème se prouve en déterminant le comportement asymptotique de la série génératrice $E(z) = \sum e_n z^n$ autour de sa singularité dominante $z = 1$. En résolvant une équation différentielle satisfaite par $E(z)$, nous prouvons que lorsque z tend vers 1

$$E(z) \sim 1 + (1-p_\mathrm{I})^{2/p_\mathrm{I}-1} K(z)^{-1} \left(2 + \int_0^z G(w)K(w)dw\right) \times (1-z)^{-2}\,,$$

où $G(z)$ tend vers $G(1) > 0$ lorsque z tend vers 1, et $K(z) := \exp\left(2p_\circledast \int_0^z \frac{A(w)}{1-p_\mathrm{I} w} dw\right)$.

Pour déterminer le comportement asymptotique précis de $E(z)$, il nous faut étudier $K(z)$ et l'intégrale $J(z) := \int_0^z G(w)K(w)dw$, dont le comportement est intuitivement déterminé par celui de l'intégrale $\int_0^z K(w)dw$. En effet, $\int_0^z K(w)dw$ converge si et seulement si $\int_0^z G(w)K(w)dw$ converge, et si les intégrales divergent en $z = 1$, $\int_0^z G(w)K(w)dw \sim G(1) \int_0^z K(w)dw$.

Le comportement asymptotique de $K(z)$ est obtenu par intégration singulière de celui de $A(z)$ (voir [FS09, Theorem VI.9]).

5.4.1 Forme close pour la série $E(z)$

La Proposition 5.7 fournit une récurrence satisfaite par la suite (e_n), dépendant de la probabilité γ_n des arbres entièrement réductibles. Pour $n > 1$,

$$e_{n+1} = 1 + (s-1)\gamma_{n+1}\mathbf{1}_{n+1\neq s} + p_\mathrm{I} e_n + \frac{2p_\mathrm{II}}{n-1}\sum_{j=1}^{n-1} e_j + \frac{2p_\circledast}{n-1}\sum_{j=1}^{n-1}(e_j - s\gamma_j)(1-\gamma_{n-j})\,. \tag{5.2}$$

où nous rappelons que $s \geqslant 3$ est la taille de l'élément absorbant $\mathcal{P}$. Nous transformons alors cette récurrence en une équation différentielle satisfaite par $E(z)$:

Proposition 5.19 .
▶ La série $E(z)$ satisfait l'équation différentielle linéaire du premier ordre suivante :

$$E'(z) = F(z, A(z)) + \tfrac{1}{1-p_\mathrm{I} z}\left(\tfrac{2}{z} - p_\mathrm{I} + 2\,(1-p_\mathrm{I})\,\frac{z}{1-z} - 2p_\circledast A(z)\right)\cdot E(z)\,,$$

avec $F(z,w) = \frac{1}{1-p_\mathrm{I} z}\left(-1 + \frac{z^2}{(1-z)^2} - 2p_\circledast \frac{z}{1-z} w + p_\circledast(s+1)w^2\right)$. ◀

Le calcul est élémentaire, mais minutieux, j'ai décidé de l'écrire car il est très facile d'oublier un terme.

Démonstration. En multipliant l'équation de récurrence par $(n-1)z^n = (n+1-2)z^n$ puis en sommant sur tous les $n > 1$, nous obtenons plusieurs termes :

- $\sum_{n>1}(n-1)z^n = \frac{z^2}{(1-z)^2} - 1$
- $\sum_{n>1}(n+1)e_{n+1}z^n - 2\sum_{n>1} e_{n+1}z^n = E'(z) - 2z - \frac{2}{z}(E(z)-z)$ — *Comme $s \geqslant 3$, $e_0 = 0$, $e_1 = 1$, $e_2 = 2$*
- De même, $\sum_{n>1}(n+1-2)\gamma_{n+1}\mathbf{1}_{n+1\neq s}z^n = A'(z) - s\gamma_s z^{s-1} - \frac{2}{z}(A(z)-\gamma_s z^{s+1})$
- $\sum_{n>1}(n-1)e_n z^n = z\sum_{n>1} n e_n z^{n-1} - \sum_{n>1} e_n z^n = zE'(z) - z - (E(z)-z)$
- $\sum_{n>1} z^n \sum_{j=1}^{n-1} e_j = z\sum_{n>0} z^{n-1}\sum_{j=0}^{n-1} e_j = \frac{z}{1-z}E(z)$
- De même $\sum_{n>1} z^n \sum_{j=1}^{n-1}(e_j - s\gamma_j) = \frac{z}{1-z}(E(z) - sA(z))$. — *Car $\gamma_0 = 0$.*
- Enfin, $\sum_{n>1} z^n \sum_{j=1}^{n-1}(e_j - s\gamma_j)\gamma_{n-j} = \sum_{n>1} z^n \sum_{j=0}^{n}(e_j - s\gamma_j)\gamma_{n-j} = (E(z) - sA(z))A(z)$. — *Le terme pour $n = 1$ est nul.*

Ainsi en mettant tout bout à bout, nous obtenons l'équation suivante :

$$\begin{aligned}(1-p_{\mathrm{I}}z)E'(z) = &\frac{z^2}{(1-z)^2} + \frac{2}{z}E(z) - p_{\mathrm{I}}E(z)\\ &+ 2p_{\mathrm{II}}\frac{z}{1-z}E(z) + 2p_{\circledast}\frac{z}{1-z}E(z) - 2p_{\circledast}E(z)A(z)\\ &- 2p_{\circledast}s\frac{z}{1-z}A(z) + 2p_{\circledast}sA(z)^2\\ &+ (s-1)(A'(z) - \tfrac{2}{z}A(z) - (s-2)\gamma_s z^{s-1})\end{aligned}$$

Rappelons que par l'Équation 5.3, $A'(z) = (s-2)\gamma_s z^{s-1} + \left(\frac{2}{z} + 2p_{\circledast}\frac{z}{1-z}\right)A(z) - p_{\circledast}\cdot(A(z))^2$, si bien que des deux dernières lignes, il ne reste que $-2p_{\circledast}\frac{z}{1-z}A(z) + (s+1)p_{\circledast}A(z)^2$.

Nous concluons en remarquant que $p_{\mathrm{II}} + p_{\circledast} = 1 - p_{\mathrm{I}}$. ■

En résolvant cette équation linéaire du premier ordre non homogène, nous obtenons une forme close pour $E(z)$, dépendant de $A(z)$:

Proposition 5.20 .
▶ La série génératrice des tailles réduites moyennes $E(z)$ admet la forme close suivante :

$$E(z) = z + z^2 I(z)\left(2 + \int_0^z (f(\zeta) + g(\zeta))\, I(\zeta)^{-1} d\zeta\right),$$

où $(1-p_{\mathrm{I}}z)f(z) = -p_{\mathrm{I}}^2 + \frac{1}{(1-z)^2} - 2p_{\circledast}\frac{A(z)/z}{1-z} + p_{\circledast}(s+1)(A(z)/z)^2$ et

$$I(z) = \exp\left(p_{\mathrm{I}}z + \int_0^z \zeta g(\zeta) d\zeta\right) \quad \text{avec } g(z) = \frac{p_{\mathrm{I}}^2 - 2p_{\mathrm{I}}}{1-p_{\mathrm{I}}z} + \frac{2}{1-z} - 2p_{\circledast}\frac{A(z)/z}{1-p_{\mathrm{I}}z}.$$

◀

Démonstration. Avec les premières valeurs de e_n, nous calculons les valeurs initiales $E(0) = 0$, $E'(0) = 1$ et $E''(0) = 4$. En écrivant $\frac{1}{1-p_{\mathrm{I}}z} = 1 + p_{\mathrm{I}}z + \frac{(p_{\mathrm{I}}z)^2}{1-p_{\mathrm{I}}z}$, et en remarquant que $A(z)/z$ est analytique sur Ω, — *En effet $e_i = i$ pour $0 \leqslant i \leqslant 4$*

$$\begin{aligned}F(z, A(z)) &= -1 - p_{\mathrm{I}}z + \frac{z^2}{1-p_{\mathrm{I}}z}\left(-p_{\mathrm{I}}^2 + \frac{1}{(1-z)^2} - 2p_{\circledast}\frac{A(z)/z}{1-z} + p_{\circledast}(s+1)(A(z)/z)^2\right)\\ &= -1 - p_{\mathrm{I}}z + z^2 f(z)\end{aligned}$$

De même le coefficient linéaire de l'équation se réécrit :

$$\begin{aligned}&\tfrac{1}{1-p_{\mathrm{I}}z}\left(\tfrac{2}{z} - p_{\mathrm{I}} + 2\,(1-p_{\mathrm{I}})\,\frac{z}{1-z} - 2p_{\circledast}A(z)\right)\\ &= \tfrac{2}{z} + (2p_{\mathrm{I}} - p_{\mathrm{I}}) + \frac{z}{1-p_{\mathrm{I}}z}(2p_{\mathrm{I}}^2 - p_{\mathrm{I}}^2 + \frac{2(1-p_{\mathrm{I}})}{1-z} - 2p_{\circledast}A(z)/z)\\ &= \tfrac{2}{z} + p_{\mathrm{I}} + zg(z)\end{aligned}$$

On doit en effet adapter légèrement la formule classique de la solution d'une équation différentielle linéaire d'ordre 1, car $2/z$ n'est pas analytique en zéro

en utilisant l'égalité $(1-p_{\mathrm{I}}) = (1-p_{\mathrm{I}}z) - p_{\mathrm{I}}(1-z)$. L'équation différentielle pour $E(z)$ est donc sous la forme étudiée par le Corollaire A.4 en annexe, avec les paramètres $u_1 = 1$, $u_2 = 2$ et $c = -p_{\mathrm{I}}$. La formule du Corollaire A.4 donne exactement la solution annoncée pour $E(z)$. ■

5.4.2 Comportement local de $E(z)$ en $z = 1$: angle d'attaque et intégration singulière

La forme close de $E(z)$, qui dépend de $A(z)$, dans la Proposition 5.20 montre que $E(z)$ est analytique sur le domaine $\Omega := \mathbb{C} \setminus [1, +\infty)$. Comme la série $\sum e_n$ diverge, $E(z)$ a une unique singularité dominante en $z = 1$.

Le but de cette section est d'étudier le comportement local de $E(z)$ au voisinage de sa singularité dominante $z = 1$.

Comme l'expression de $E(z)$ est assez compliquée, nous la coupons en morceaux plus petits que nous allons étudier séparément. Dans un premier temps, remarquons qu'on peut calculer directement l'intégrale $I(z)$:

$$I(z) = (1-p_{\mathrm{I}}z)^{2/p_{\mathrm{I}}-1} \exp\left(-2p_{\circledast}\int_0^z \frac{A(w)}{1-p_{\mathrm{I}}w}dw\right) \times \frac{1}{(1-z)^2}\,. \tag{5.5}$$

Nous définissons ainsi les fonctions :

$$\begin{aligned}K(z) &:= \exp\left(2p_{\circledast}\int_0^z \frac{A(w)}{1-p_{\mathrm{I}}w}dw\right)\,,\\ G(z) &:= (1-p_{\mathrm{I}}z)^{1-2/p_{\mathrm{I}}} \cdot (f(z)+g(z)) \cdot (1-z)^2\,.\end{aligned}$$

Avec ces fonctions, l'expression de $E(z)$ s'écrit désormais :

$$E(z) = z + z^2(1-p_{\mathrm{I}}z)^{2/p_{\mathrm{I}}-1}K(z)^{-1}\left(2 + \int_0^z G(w)K(w)dw\right) \times (1-z)^{-2}\,.$$

Nous étudions dans un premier temps la fonction $K(z)$ en $z = 1$, dans la Section 5.4.3, puis nous étudions l'intégrale $J(z) = \int_0^z G(w)K(w)dw$ dans la Section 5.4.4. Nous concluons la preuve du théorème principal dans la Section 5.4.5, en recollant les morceaux. Pour étudier $K(z)$ et $J(z)$, nous allons utiliser à de nombreuses reprises des théorèmes d'intégration singulière, qui permettent d'intégrer les comportements asymptotiques de certaines fonctions analytiques. Les théorèmes d'intégration singulière sont énoncés ci-après ; leur démonstration est donnée en annexe, à la section A.1.

Lemme 5.21 ([FS09, Theorem VI.9]).
▶ Soit $f(z)$ une fonction analytique sur Ω telle que dans un voisinage de $z = 1$, $f(z) = O((1-z)^s)$, avec $s > -1$. Alors

$$\int_0^z f(\zeta)d\zeta = \int_0^1 f(t)dt + O((1-z)^{s+1})\,. \qquad ◀$$

Le lemme suivant est une adaptation du précédent dans le cas où la fonction est bornée par un terme logarithmique :

Lemme 5.22 ([FS09, Theorem VI.9]).
▶ Soit $f(z)$ une fonction analytique sur Ω telle que dans un voisinage de $z = 1$, $f(z) = O((1-z)^{-1}(-\log(1-z))^{-\beta})$, avec $\beta > 1$. Soit $a_0 > 0$. Alors

$$\int_{a_0}^{z} f(\zeta)d\zeta = \int_{a_0}^{1} f(t)dt + O((-\log(1-z))^{1-\beta})\,. \quad \blacktriangleleft$$

Lemme 5.23 (Intégration des équivalents).
▶ Soit $f(z)$ une fonction analytique sur Ω telle qu'au voisinage de $z = 1$, $f(z) \sim_{z=1} (1-z)^s$. Alors :

a. si $s > -1$, l'intégrale converge et $\int_0^z f(\zeta)d\zeta \to_{z=1} \int_0^1 f(\zeta)d\zeta$;
b. si $s < -1$, $\int_0^z f(\zeta)d\zeta \sim_{z=1} \frac{1}{-s-1}(1-z)^{s+1}$;
c. si $s = 1$, $\int_0^z f(\zeta)d\zeta \sim_{z=1} -\log(1-z)$. ◀

5.4.3 Étude de $K(z)$

Le lemme suivant caractérise le comportement asymptotique de $K(z)$, à partir de celui de $A(z)$ décrit dans la Proposition 5.14 :

Proposition 5.24.
▶ La fonction $K(z) := \exp\left(2p_\circledast \int_0^z \frac{A(w)}{1-p_I w}dw\right)$ admet les asymptotiques suivantes lorsque $z \to 1$ dans Ω :

— si $p_\circledast < 1/2$ alors $K(z) \sim_{z\to 1} \exp\left(2p_\circledast \int_0^1 \frac{A(t)}{1-p_I t}dt\right)$,

— si $p_\circledast = 1/2$ alors $K(z) \sim_{z\to 1} C_K \times \left(\log \frac{1}{1-z}\right)^{\frac{2}{1-p_I}}$,

— si $p_\circledast > 1/2$ alors $K(z) \sim_{z\to 1} C_K \times \left(\frac{1}{1-z}\right)^{\frac{4p_\circledast - 2}{1-p_I}}$,

où $C_K > 0$ est une constante qui dépend de $p_I, p_\circledast$ et de s. ◀

Autrement dit le C_K du point 2 et du point 3 ne sont pas les mêmes !

Démonstration. Nous allons appliquer les lemmes 5.21 et 5.22 d'intégration singulière à la fonction $\frac{A(z)}{1-p_I z}$, pour ensuite passer à l'exponentielle. Le comportement de $K(z)$ change naturellement aux mêmes seuils que $A(z)$.

— Dans le cas $p_\circledast < 1/2$, nous savons par la Proposition 5.14 que :

$$A(z) \sim_{z\to 1} \frac{C_A}{(1-z)^{2p_\circledast}}$$

lorsque $z \to 1$ dans Ω. Comme $2p_\circledast < 1$, l'intégrale $\int_0^z \frac{A(w)}{1-p_I w}dw$ converge lorsque $z \to 1$ dans Ω. De plus, par le Lemme 5.21 d'intégration singulière :

$$\int_0^z \frac{A(w)}{1-p_I w}dw = \int_0^1 \frac{A(t)}{1-p_I t}dt + O((z-1)^{1-2p_\circledast})\,,$$

ce qui nous donne l'asymptotique suivante, après être passé à l'exponentielle :

$$\exp\left(2p_\circledast \int_0^z \frac{A(w)}{1-p_I w}dw\right) = \exp\left(2p_\circledast \int_0^1 \frac{A(t)}{1-p_I t}dt\right) \times \left(1 + O((z-1)^{1-2p_\circledast})\right)\,,$$

car $\exp(O((1-z)^{1-2p_\circledast})) = 1 + O((1-z)^{1-2p_\circledast})$.
Nous avons donc l'asymptotique annoncée dans le cas $p_\circledast < 1/2$, car l'intégrale $C_K := \exp\left(2p_\circledast \int_0^1 \frac{A(t)}{1-p_\mathrm{I} t} dt\right)$ est bien strictement positive et finie.

— Intéressons-nous maintenant au cas $p_\circledast > 1/2$. Par la Proposition 5.14, nous savons que

$$A(z) = \frac{\gamma_\infty}{1-z} + O((1-z)^{2p_\circledast - 2}) \,.$$

Remarquons alors que $\frac{1}{1-p_\mathrm{I} z}\left(A(z) - \frac{\gamma_\infty}{1-z}\right) = O((1-z)^{2p_\circledast-2})$. En effet, $1/p_\mathrm{I} > 1$ donc le terme $\frac{1}{1-p_\mathrm{I} z}$ est borné dans un voisinage de 1.
Comme $2p_\circledast - 2 > -1$, nous appliquons le Lemme 5.21 d'intégration singulière :

$$\int_0^z \frac{1}{1-p_\mathrm{I} w}\left(A(w) - \frac{\gamma_\infty}{1-w}\right) dw = c_0 + O((1-z)^{2p_\circledast - 1}) \,,$$

où $c_0 = \int_0^1 \frac{1}{1-p_\mathrm{I} t}\left(A(t) - \frac{\gamma_\infty}{1-t}\right) dt$. Par une décomposition en éléments simples nous calculons :

$$\int_0^z \frac{\gamma_\infty}{1-w}\frac{dw}{1-p_\mathrm{I} w} = \frac{\gamma_\infty}{1-p_\mathrm{I}} \log\left(\frac{1}{1-z}\right) + \frac{\gamma_\infty}{p_\mathrm{I}-1} \log\left(\frac{1}{1-p_\mathrm{I} z}\right) \,,$$

et nous remarquons que $\frac{\gamma_\infty}{p_\mathrm{I}-1}\log\left(\frac{1}{1-p_\mathrm{I} z}\right) = \frac{\gamma_\infty}{p_\mathrm{I}-1}\log\left(\frac{1}{1-p_\mathrm{I}}\right) + O((z-1)^1)$.
Ainsi, tout mis bout à bout :

$$2p_\circledast \int_0^z \frac{A(z)}{1-p_\mathrm{I} w} dw = 2p_\circledast \frac{\gamma_\infty}{1-p_\mathrm{I}} \log\left(\frac{1}{1-z}\right) + c + O((1-z)^{2p_\circledast - 1}) \,,$$

où $c = 2p_\circledast\left(c_0 + \frac{\gamma_\infty}{p_\mathrm{I}-1}\log\left(\frac{1}{1-p_\mathrm{I}}\right)\right)$. Nous obtenons le résultat annoncé en passant à l'exponentielle, et en utilisant le fait que $\exp(O((1-z)^{2p_\circledast-1})) = 1 + O((1-z)^{2p_\circledast-1})$.

— Enfin, nous nous intéressons au cas $p_\circledast = 1/2$, dont la preuve est très similaire au cas $p_\circledast > \frac{1}{2}$. Par la Proposition 5.14, nous savons que

$$A(z) \sim_{z\to 1} \frac{2}{1-z}\left(\log\left(\frac{1}{1-z}\right)\right)^{-1} \,.$$

Ainsi, l'intégrale $\int_0^1 \frac{A(w)}{1-p_\mathrm{I} w} dw$ n'est pas convergente en 1 (on reconnaît à droite une intégrale de Bertrand divergente [3]). Cependant, l'équivalent asymptotique de $A(z)$ au voisinage de 1 n'est pas intégrable en 0, il faut donc couper l'intégrale avant d'utiliser l'intégration singulière.
Pour un $a_0 \in]0,1[$ arbitraire, nous considérons donc l'intégrale de a_0 à z, en évitant ainsi les problèmes d'intégration de $\left(\log\left(\frac{1}{1-z}\right)\right)^{-1}$ en $z = 0$.
Nous écrivons donc :

$$\int_0^z \frac{A(w)}{1-p_\mathrm{I} w} dw = \int_0^{a_0} \frac{A(w)}{1-p_\mathrm{I} w} dw + \int_{a_0}^z \frac{A(w)}{1-p_\mathrm{I} w} dw \,,$$

3. Une intégrale de Bertrand est une intégrale de la forme $\int_0^a \frac{dx}{x^\alpha |\log x|^\beta}$ avec $0 < a < 1$, qui converge en 0 si et seulement si $\alpha < 1$ ou ($\alpha = 1$ et $\beta > 1$).

où la première intégrale est juste une constante positive. Nous nous intéressons donc à l'intégrale :

$$\int_{a_0}^{z} \frac{1}{1-p_{\mathrm{I}}w}\left(A(w) - \frac{2}{1-w}\left(\log\left(\tfrac{1}{1-w}\right)\right)^{-1}\right) dw\,,$$

qui converge en $z=1$, car l'intégrande est borné en $O\left(\frac{1}{1-w}\left(\log\left(\frac{1}{1-w}\right)\right)^{-2}\right)$ lorsque $w \to 1$, par la Proposition 5.14. Ainsi le Lemme 5.22 d'intégration singulière implique l'égalité :

$$\int_{a_0}^{z} \frac{1}{1-p_{\mathrm{I}}w}\left(A(w) - \frac{2}{1-w}\left(\log\left(\tfrac{1}{1-w}\right)\right)^{-1}\right) dw = c_0 + O((\log(\tfrac{1}{1-z}))^{-1})\,,$$

pour une certaine constante c_0. Par ailleurs, en intégrant par partie :

$$\int_{a_0}^{z} \tfrac{1}{1-p_{\mathrm{I}}w}\tfrac{1}{1-w}\left(\log\left(\tfrac{1}{1-w}\right)\right)^{-1} dw = \tfrac{1}{1-p_{\mathrm{I}}z}\log\log(\tfrac{1}{1-z}) + c_1 + \int_{a_0}^{z} \tfrac{p_{\mathrm{I}}}{(1-p_{\mathrm{I}}w)^2}\log(\log(\tfrac{1}{1-w}))dw$$

pour une certaine constante c_1. Comme $(1-p_{\mathrm{I}}w)^{-2}\log(\log(\frac{1}{1-w})) = O((1-w)^{-1/2})$, par intégration singulière, $\int_{a_0}^{z} \frac{p_{\mathrm{I}}}{(1-p_{\mathrm{I}}w)^2}\log(\log(\frac{1}{1-w}))dw = c_2 + o(1)$ pour une constante c_2.
Ainsi :

$$\int_{0}^{z} \frac{A(w)}{1-p_{\mathrm{I}}w}dw = \frac{2}{1-p_{\mathrm{I}}}\log\log\left(\tfrac{1}{1-z}\right) + \texttt{constante} + o(1)\,,$$

et nous concluons en passant à l'exponentielle.

■

Nous avons ainsi déterminé le comportement de $K(z)$, et nous pouvons désormais nous attaquer à l'étude de l'intégrale $J(z) = \int_0^z G(w)K(w)dw$.

Remarque 5.25 .
▶ Dans l'article initial publié à STACS, en 2021, nous avons utilisé dans les parties qui suivent des développements asymptotiques, à la place d'équivalents, pour finalement supprimer tous les termes d'erreurs et ne garder qu'un équivalent dans le résultat final. Il nous manquait notamment un théorème d'intégration singulière qui porte uniquement sur les équivalents pour pouvoir éviter l'introduction des termes d'erreurs (cf Lemme 5.23). J'ai décidé dans ma thèse de ne garder que les termes nécessaires à la preuve du résultat principal, ce qui allège légèrement les calculs. ◀

5.4.4 Étude de $J(z)$

Nous rappelons que nous avons défini précédemment les fonctions suivantes :

$$f(z) := (1-p_{\mathrm{I}}z)^{-1}\left(-p_{\mathrm{I}}^2 + \frac{1}{(1-z)^2} - 2p_{\circledast}\frac{A(z)/z}{1-z} + p_{\circledast}(s+1)(A(z)/z)^2\right)$$

$$g(z) := \frac{p_{\mathrm{I}}^2 - 2p_{\mathrm{I}}}{1-p_{\mathrm{I}}z} + \frac{2}{1-z} - 2p_{\circledast}\frac{A(z)/z}{1-p_{\mathrm{I}}z}$$

$$G(z) := (1-p_{\mathrm{I}}z)^{1-2/p_{\mathrm{I}}} \cdot (f(z)+g(z)) \cdot (1-z)^2$$

Le but de cette section est de trouver les asymptotiques en $z = 1$ de l'intégrale $J(z) = \int_0^z G(w)K(w)dw$.

Comme $K(z)$ a déjà été étudiée à la section précédente, intéressons-nous à $G(z) = (1 - p_{\mathrm{I}}z)^{1-2/p_{\mathrm{I}}} \cdot (f(z) + g(z)) \cdot (1 - z)^2$, qui se trouve être en réalité analytique en $z = 1$. En effet nous montrons que $f(z) + g(z)$ est de la forme $c(1 - z)^{-2}$ lorsque $z \to 1$:

Lemme 5.26.
▶ La fonction $f(z) + g(z)$ se comporte en $\Theta((1 - z)^{-2})$ en $z = 1$. Plus précisément, il existe une constante c qui dépend de $p_\circledast, p_{\mathrm{I}}$ et s, telle qu'au voisinage de $z = 1$ dans Ω,

$$f(z) + g(z) \sim_{z\to 1} \frac{c}{(1 - z)^2}\,.$$

Par conséquent la fonction $G(z) = (1 - p_{\mathrm{I}}z)^{1-2/p_{\mathrm{I}}} \cdot (f(z) + g(z)) \cdot (1 - z)^2$ admet un prolongement analytique sur un voisinage de 1. ◀

Démonstration. Calculons :

$$f(z) + g(z) = \frac{2}{1 - z} + (1 - p_{\mathrm{I}}z)^{-1}\Big(- 2p_{\mathrm{I}} + \frac{1}{(1 - z)^2} - 2p_\circledast \frac{A(z)}{z}\frac{2 - z}{1 - z} + p_\circledast(s + 1)(A(z)/z)^2\Big)$$

— Dans le cas où $p_\circledast < 1/2$, $A(z) \sim C_A(1 - z)^{-2p_\circledast}$ par la Proposition 5.14.
Ainsi dans la somme $f(z)+g(z)$, le terme dominant est $(1-z)^{-2}$, car $A(z)/(1-z) = O((1 - z)^{-1-2p_\circledast})$ et $A(z)^2 = O((1 - z)^{-4p_\circledast})$, et $4p_\circledast < 1 + 2p_\circledast < 2$. Ainsi :

$$f(z) + g(z) \sim_{z=1} \frac{1/(1 - p_{\mathrm{I}})}{(1 - z)^2}\,.$$

— Le cas $p_\circledast = 1/2$ est similaire : dans ce cas $A(z) = o((1 - z)^{-1})$, par la Proposition 5.14, et donc le terme dominant est toujours le terme en $(1 - z)^{-2}$, si bien que

$$f(z) + g(z) \sim \frac{1/(1 - p_{\mathrm{I}})}{(1 - z)^2}\,.$$

— Dans le cas où $p_\circledast > 1/2$, $A(z) \sim \frac{\gamma_\infty}{1-z}$ par la Proposition 5.14. Ainsi

$$f(z) + g(z) \sim_{z=1} \left(\frac{1}{1 - p_{\mathrm{I}}} - \frac{2p_\circledast\gamma_\infty}{1 - p_{\mathrm{I}}} + \frac{(s + 1)p_\circledast}{1 - p_{\mathrm{I}}}\gamma_\infty^2\right)\frac{1}{(1 - z)^2}\,,$$

pourvu que la constante devant $(1 - z)^{-2}$ soit non nulle. Nous rappelons que la limite γ_∞ vérifie $2p_\circledast\gamma_\infty - p_\circledast\gamma_\infty^2 = \gamma_\infty$. Ainsi

$$1 - 2p_\circledast\gamma_\infty + (s + 1)p_\circledast\gamma_\infty^2 \geqslant 1 - 2p_\circledast\gamma_\infty + p_\circledast\gamma_\infty^2 = 1 - \gamma_\infty\,,$$

qui est strictement positif car $p_\circledast < 1$ et donc $\gamma_\infty < 1$. ■

Rappelons que $p_{\mathrm{I}} > 0$ *et que* $p_{\mathrm{I}} + p_\circledast \leqslant 1$

Maintenant, nous sommes en mesure de trouver les asymptotiques de l'intégrale $J(z) := \int_0^z G(w)K(w)dw$ lorsque $z \to 1$.

Proposition 5.27 .
▶ La fonction $J(z)$ satisfait les asymptotiques suivantes lorsque $z \to 1$ dans Ω :
— si $4p_\circledast < 3 - p_\mathrm{I}$ alors $J(z) \to C_J$, avec $2 + C_J > 0$
— si $4p_\circledast = 3 - p_\mathrm{I}$ alors $J(z) \sim C_J \times \log\left(\frac{1}{1-z}\right)$, avec $C_J > 0$
— si $4p_\circledast > 3 - p_\mathrm{I}$, alors $J(z) \sim C_J \times (1-z)^{1-\frac{4p_\circledast-2}{1-p_\mathrm{I}}}$, avec $C_J > 0$
où C_J est une constante qui dépend de $p_\mathrm{I}, p_\circledast, s$. ◀

Démonstration. Nous rappelons que $G(z)$ est analytique sur Ω et que par ailleurs $G(z)$ admet une limite finie lorsque $z \to 1$, par le Lemme 5.26. Ainsi, $|G(w)K(w)| \sim G(1)|K(w)|$ lorsque $w \to 1$, et la convergence de l'intégrale $J(z) = \int_0^z G(w)K(w)dw$ en $z \to 1$ est équivalente à celle de $K(w)$.

— Par les équivalents de $K(z)$ en 1 de la Proposition 5.24, $K(z)$ est intégrable en $z = 1$ pour $p_\circledast < 1/2$ et pour $p_\circledast = 1/2$ (car $(-\log(1-z))^{\frac{2}{1-p_\mathrm{I}}} = o((1-z)^{-1/2})$). Par ailleurs dans le cas où $p_\circledast > 1/2$, $K(z)$ est aussi intégrable si $\frac{4p_\circledast-2}{1-p_\mathrm{I}} < 1$, ce qui est équivalent à $4p_\circledast < 3 - p_\mathrm{I}$. Remarquons que cette dernière condition englobe le cas $p_\circledast \leqslant 1/2$.
Ainsi, si $4p_\circledast < 3 - p_\mathrm{I}$, alors $J(z)$ converge en $z = 1$, vers une constante C_J, qui est a priori de signe quelconque, pour l'instant.
— Dans le cas où $\frac{4p_\circledast-2}{1-p_\mathrm{I}} > 1$, alors l'intégrale diverge. Par ailleurs $p_\circledast > 1/2$, donc le Lemme 5.26 nous dit que lorsque $w \to 1$:

$$G(w)K(w) \sim c(1-p_\mathrm{I})^{2/p_\mathrm{I}-1} C_K (1-w)^{-\frac{4p_\circledast-2}{1-p_\mathrm{I}}},$$

lorsque $w \to 1$. Par le Lemme 5.23 d'intégration des équivalents (cas divergent),

$$J(z) \sim_{z\to 1} c(1-p_\mathrm{I})^{2/p_\mathrm{I}-1} C_K (1-z)^{1-\frac{4p_\circledast-2}{1-p_\mathrm{I}}}.$$

— Lorsque $\frac{4p_\circledast-2}{1-p_\mathrm{I}} = 1$, ou de façon équivalente $4p_\circledast = 3 - p_\mathrm{I}$, nous avons le comportement asymptotique $K(z) \sim \frac{C_K}{1-z}$. Par le Lemme 5.23,

$$J(z) \sim_{z\to 1} c(1-p_\mathrm{I})^{2/p_\mathrm{I}-1} C_K \log\left(\frac{1}{1-z}\right).$$

Enfin, nous expliquons pourquoi $2 + C_J > 0$ dans le cas convergent, par un argument combinatoire. Rappelons que

$$(E(z) - z)/z^2 = I(z) \cdot (2 + J(z)),$$

avec $I(x)$ qui est le résultat d'une exponentielle, à valeur réelle pour $x \in [0, 1[$, donc strictement positive sur $[0, 1[$. De plus, par le Lemme 5.26, $f(x) + g(x) \sim c/(1-x)^2$ et donc $f(x) + g(x) > 0$ pour $x \in [\alpha, 1[$ avec α suffisamment proche de 1. Ainsi l'intégrale $J(x) = \int_0^x (f(\zeta) + g(\zeta))I(\zeta)^{-1}d\zeta$ est strictement croissante sur l'intervalle $[\alpha, 1[$.

Sa dérivée est strictement positive.

Or, $(E(\alpha) - \alpha)/\alpha^2 = \sum_{n=0}^{\infty} e_{n+2}\alpha^n \geqslant \frac{1}{1-\alpha} > 0$ car $e_n \geqslant 1$ pour $n \geqslant 1$. Ainsi $2 + J(\alpha) > 0$, et donc par croissance $2 + J(x) > 2 + J(\alpha) > 0$ pour tout $x \in [\alpha, 1[$. Donc en passant à la limite $2 + C_J > 0$. ■

5.4.5 Preuve du théorème principal

Il ne reste plus qu'à combiner les comportements des briques $K(z)$ et $J(z)$ pour déterminer celui de $E(z)$, et donc de la suite (e_n).

Théorème 5.17 (Comportement asymptotique des tailles moyennes e_n).
▶ Si la probabilité p_{I} des opérateurs unaires n'est pas 0, alors l'espérance e_n de la taille après réduction par σ d'un arbre d'expressions aléatoire de taille n suivant la distribution ABR, satisfait les propriétés suivantes :

- si $p_{\circledast} < 1/2$, alors $e_n \sim c_1\, n$;
- si $p_{\circledast} = 1/2$, alors $e_n \sim c_2\, n \log(n)^{-2/(1-p_{\mathrm{I}})}$;
- si $\frac{1}{2} < p_{\circledast} < \frac{3-p_{\mathrm{I}}}{4}$, alors $e_n \sim c_3\, n^{1-\frac{4p_{\circledast}-2}{1-p_{\mathrm{I}}}}$;
- si $p_{\circledast} = \frac{3-p_{\mathrm{I}}}{4}$, alors $e_n \sim c_4\, \log(n)$;
- si $p_{\circledast} > \frac{3-p_{\mathrm{I}}}{4}$, alors $e_n \to e_\infty$, où e_∞ est une constante strictement positive ;

avec $c_1, \ldots, c_4$ des constantes positives non nulles. ◀

Démonstration. Nous rappelons que $E(z) = z + z^2 I(z) \cdot (2 + J(z))$, avec $I(z) = z^2(1 - p_{\mathrm{I}} z)^{2/p_{\mathrm{I}}-1} K(z)^{-1}(1-z)^{-2}$. Ainsi :

$$E(z) \sim_{z\to 1} \frac{(1-p_{\mathrm{I}})^{2/p_{\mathrm{I}}-1}}{(1-z)^2} K(z)^{-1} \cdot (2 + J(z))\,, \tag{5.6}$$

et il ne reste plus qu'à appliquer les Propositions 5.24 et 5.27 et le théorème de Transfert (rappelons que $E(z)$ est analytique sur $\Omega = \mathbb{C} \setminus \{1\}$ qui est un Δ-domaine autour de sa singularité dominante $z = 1$).

- Si $p_{\circledast} < \frac{1}{2}$, alors $K(z)$ et $2 + J(z)$ convergent vers des constantes strictement positives lorsque $z \to 1$. Ainsi $E(z) \sim_{z\to 1} C_E(1-z)^{-2}$ avec $C_E > 0$. Le théorème de Transfert implique donc que e_n est asymptotiquement linéaire en n.
- Si $p_{\circledast} = \frac{1}{2}$, alors $K(z) \sim C_K \times \left(\log \frac{1}{1-z}\right)^{\frac{2}{1-p_{\mathrm{I}}}}$, avec $C_K > 0$, et nous avons toujours $2 + J(z) \sim 2 + C_J > 0$. Le théorème de Transfert implique donc que $e_n \sim c_2 n \log(n)^{-2/(1-p_{\mathrm{I}})}$ avec $c_2 > 0$.
- Si $p\circledast > \frac{1}{2}$, alors $K(z)^{-1} \sim C_K^{-1}(1-z)^{\frac{4p_{\circledast}-2}{1-p_{\mathrm{I}}}}$. De plus :
  - Si $4p_{\circledast} < 3 - p_{\mathrm{I}}$, alors nous avons toujours $2 + J(z) \sim 2 + C_J > 0$. Ainsi
    $$E(z) \sim_{z\to 1} C_E \left(\frac{1}{1-z}\right)^{2-\frac{2p_{\circledast}\gamma_\infty}{1-p_{\mathrm{I}}}}$$
    pour une constante $C_E > 0$. Ainsi par le théorème de Transfert, $e_n \sim c_3 n^{1-\frac{4p_{\circledast}-2}{1-p_{\mathrm{I}}}}$ avec $c_3 > 0$.
  - Si $4p_{\circledast} = 3 - p_{\mathrm{I}}$, cette fois $2 + J(z) \sim C_J \times \log\left(\frac{1}{1-z}\right)$ et ainsi
    $$E(z) \sim C_E \frac{1}{1-z} \log\left(\frac{1}{1-z}\right)$$
    pour une constante $C_E > 0$. Ainsi par le théorème de Transfert, $e_n \sim c_4 \log n$.

— Ensin si $4p_{\circledast} > 3 - p_{\mathrm{I}}$, alors $2 + J(z) \sim C_J \times (1-z)^{1-\frac{4p_{\circledast}-2}{1-p_{\mathrm{I}}}}$, si bien que :

$$E(z) \sim \frac{C_E}{1-z}$$

pour une constante $C_E > 0$. Ainsi par le théorème de Transfert, $e_n \to e_\infty = C_E$.

■

Remarque 5.28 .
▶ Nous pouvons calculer facilement la limite e_∞ dans le cas $4p_{\circledast} > 3 - p_{\mathrm{I}}$ à partir de la récurrence de e_n :

$$e_{n+1} = 1+(s-1)\gamma_{n+1}\mathbf{1}_{n+1\neq s}+p_{\mathrm{I}}e_n+\frac{2p_{\mathrm{II}}}{n-1}\sum_{j=1}^{n-1}e_j+\frac{2p_{\circledast}}{n-1}\sum_{j=1}^{n-1}(e_j-s\gamma_j)(1-\gamma_{n-j})\,.$$

Dans ce cas, $e_n \to e_\infty$ et $\gamma_n \to \gamma_\infty$. En particulier, en utilisant les sommes de Cesàro suivantes :

$$\frac{1}{n-1}\sum_{j=1}^{n-1}e_j \to e_\infty\,, \quad \frac{1}{n-1}\sum_{j=1}^{n-1}(e_j-s\gamma_j)(1-\gamma_{n-j}) \to (e_\infty - s\gamma_\infty)(1-\gamma_\infty)\,.$$

Ainsi, en passant à la limite, nous obtenons l'équation suivante pour e_∞ :

$$e_\infty = 1 + (s-1)\gamma_\infty + p_{\mathrm{I}}e_\infty + 2p_{\mathrm{II}}e_\infty + 2p_{\circledast}(e_\infty - s\gamma_\infty)(1-\gamma_\infty)\,.$$

Nous obtenons finalement, après avoir remplacé γ_∞ par sa valeur $\gamma_\infty = (2p_{\circledast} - 1)/p_{\circledast}$:

$$e_\infty = \frac{(2p_{\circledast}-1)^2 s + 1 - p_{\circledast}}{p_{\circledast}(4p_{\circledast} - 3 + p_{\mathrm{I}})}\,.$$

Nous remarquons notamment que e_∞ tend vers l'infini lorsque $p_{\circledast}$ s'approche du seuil $(3-p_{\mathrm{I}})/4$, tandis que e_∞ tends vers s lorsque $p_{\mathrm{I}} \to 0$ et $p_{\circledast} \to 1$ simultanément. ◀

5.5 Le cas binaire $p_{\mathrm{I}} = 0$

Le cas où p_{I} est nul, donc avec des arbres binaires, est très similaire au cas unaire-binaire précédent. Les calculs sont même plus simples. Même si le résultat final revient à peu près à dire qu'il suffit, dans le théorème principal de la section précédente, de poser $p_{\mathrm{I}} = 0$, les équations intermédiaires sont tout de même différentes, si bien qu'il s'agit bien d'un cas à part entière. Il faut notamment changer la définition de la taille d'un arbre d'expression.

Définition 5.29 (Taille d'un arbre binaire).
▶ Nous considérons une famille $\mathcal{T}(\mathcal{A}_0 \cup \mathcal{A}_2, a)$ d'arbres d'expressions binaires, construits sur un ensemble $\mathcal{A}_0$ de feuilles et un ensemble $\mathcal{A}_2$ d'opérateurs binaires.

Dans toute cette section, pour $T \in \mathcal{T}(\mathcal{A}_0 \cup \mathcal{A}_2, a)$, la taille $|T|$ de T désignera son nombre d'*opérateurs binaires*. Autrement dit nous ne comptons plus les feuilles dans la taille d'un arbre. ◀

Nous adaptons donc la définition d'un arbre d'expression aléatoire ABR à cette nouvelle notion de taille, et à l'absence d'opérateur unaire :

Définition 5.30 (Arbre d'expression ABR).
▶ Un arbre aléatoire d'expression ABR de taille $n \in \mathbb{N}$ se construit de façon récursive :

- si $n = 0$, on tire une feuille dans $\mathcal{A}_0$ avec la distribution $(p_a)_{a\in\mathcal{A}_0}$.
- Si $n \geqslant 1$, on tire un opérateur $\oplus \in \mathcal{A}_2$ avec la distribution $(p_{op})_{op\in\mathcal{A}_2}$. Puis on tire une taille k uniformément au hasard dans $\{0, \dots, n-1\}$. On construit indépendamment deux arbres T_L et T_R de tailles respectives k et $n-1-k$. Enfin on retourne $\overset{\oplus}{\underset{T_L\ T_R}{/\backslash}}$.

◀

Nous définissons $s = |\mathcal{P}|$ la taille de l'élément absorbant, et de même qu'il était demandé que la taille en nombre total de nœud soit plus grande que 3, nous demandons ici que $s \geqslant 1$.

Nous définissons alors les suites $p_{n,k}$, γ_n, e_n de la même façon qu'aux sections précédentes, en changeant juste la définition de la taille qui ne prend plus en compte les feuilles. Nous définissons de même $A(x) = \sum_{n\in\mathbb{N}} \gamma_n x^n$ et $E(x) = \sum_{n\in\mathbb{N}} e_n x^n$.

5.5.1 Étude rapide de $A(z)$

Nous avons alors facilement la récurrence suivante (il s'agit de la même preuve que pour le cas unaire-binaire, adaptée à la nouvelle définition de la taille)

Proposition 5.31 (Probabilité des entièrement réductibles).
▶ La suite (γ_n) vérifie la relation de récurrence

$$\gamma_{n+1}\mathbf{1}_{n+1\neq s} = \frac{p_{\circledast}}{n+1} \cdot \sum_{k=0}^{n} (\gamma_k + \gamma_{n-k} - \gamma_k\gamma_{n-k}) \quad \text{pour tout } n \in \mathbb{N},$$

avec $\gamma_n = 0$ pour $n < s$ et $\gamma_s = \Pr_s(\mathcal{P})$. ◀

Remarque 5.32 (Choix de la définition de la taille).
▶ Le terme $\frac{1}{n+1}$ des récurrences précédentes, dû au changement de définition de la taille, à la place du $\frac{1}{n-1}$ qu'on trouvait dans les sections précédentes, va rendre les preuves beaucoup plus naturelles : il n'y aura notamment plus de terme en $2/z$ dans les équations différentielles, qui gênait artificiellement les preuves. Les deux notions de taille sont reliées facilement par la relation $|T|_{\text{int}} \leqslant |T|_{\text{int+feuilles}} \leqslant 2|T|_{\text{int}} + 1$ (dans le cas binaire, la dernière inégalité est une égalité). Dans ce cas, pourquoi ne pas avoir choisi cette notion de taille dans le cas unaire-binaire ? La première raison est purement temporelle, nous n'avions pas étudié le cas binaire pour notre article accepté à STACS, et c'est dans le cas binaire que ce choix de taille s'impose naturellement. Par ailleurs, nous sommes partis des algorithmes de génération aléatoire utilisés dans la littérature de vérification, et avons ainsi naturellement utilisé la même notion de taille d'expression. Notre but étant de pouvoir parler aux personnes utilisant ces algorithmes, il nous paraissait important de garder la même notion de taille. ◀

La preuve de la proposition suivante est semblable au cas unaire-binaire :

Proposition 5.33.
▶ Le rayon de convergence de $A(z)$ est exactement 1. ◀

Et la série $A(z)$ satisfait toujours une équation de Riccati :

Proposition 5.34.
▶ La série $A(z)$ satisfait l'équation de Riccati :

$$A'(z) = s\gamma_s z^{s-1} + \frac{2p_{\circledast}}{1-z}A(z) - p_{\circledast} \cdot (A(z))^2 \,. \tag{5.7}$$

◀

Démonstration. Il suffit de multiplier la récurrence de γ par $(n+1)z^n$, et de sommer pour $n \geqslant s$. ■

Pour étudier cette équation différentielle, nous effectuons le changement de fonction inconnue $v(z) = \exp\left(p_{\circledast} \int_0^z A(w)dw\right)$. Ainsi $v(z)$ vérifie l'équation différentielle linéaire d'ordre 2 :

$$v''(z) = p_{\circledast} s\gamma_s z^{s-1} v(z) + \frac{2p_{\circledast}}{1-z}v'(z) \,. \tag{5.8}$$

Nous remarquons que $v(z)$ est analytique dans le disque $|z| < 1$ car $A(z)$ l'est, et de plus :

Proposition 5.35.
▶ La série entière $A(z)$ admet un prolongement analytique sur le domaine $\Omega = \mathbb{C} \setminus [1, \infty)$. En particulier, $z = 1$ est la seule singularité dominante de $A(z)$. ◀

Démonstration. Même argument que dans le cas unaire-binaire, la preuve est même plus simple car les coefficients sont analytiques sur Ω, il n'y a pas de fausse singularité en $z = 0$. ■

Nous pouvons toujours appliquer la méthode de Frobenius à $v(z)$ (cf Théorème 5.12). Le polynôme indiciel associé est $\mathcal{I}(\theta) = \theta(\theta-1) + 2p_{\circledast}\theta = \theta(\theta - 1 + 2p_{\circledast})$. Il s'agit du même polynôme indiciel que dans le cas unaire-binaire. Nous en déduisons que les formes des asymptotiques de $v(z)$ de la Proposition 5.13 et de $A(z)$ de la Proposition 5.14 sont toujours valides, ce qui conclut l'étude de $A(z)$.

5.5.2 Étude rapide de $E(z)$ et asymptotiques des tailles réduites

Dans cette partie, je démontre soigneusement la récurrence satisfaite par (e_n), même si elle ressemble au cas unaire-binaire, puis j'explique en quoi les arguments du cas unaire-binaire s'adaptent facilement pour obtenir à la fin le même type de résultat que dans le cas unaire-binaire.

Proposition 5.36.
▶ La suite (e_n) des espérances des tailles après réduction satisfait pour tout $n \geqslant 0$ la récurrence suivante :

$$\begin{aligned} e_{n+1} = 1 &+ (s-1)\gamma_{n+1}\mathbf{1}_{n+1\neq s} \\ &+ \frac{2p_{\mathrm{II}}}{n+1}\sum_{j=0}^{n} e_j + \frac{2p_{\circledast}}{n+1}\sum_{j=0}^{n}(e_j - s\gamma_j)(1-\gamma_{n-j}) \,. \end{aligned}$$

◀

Démonstration. Je cherche dans un premier temps, pour n et k fixés une récurrence sur $p_{n,k}$. Si on regarde la probabilité des arbres de taille $n+1$ dont la réduction est de taille k :

- Les arbres qui commencent par un opérateur binaire autre que $p_{\circledast}$ ont une probabilité $p_{\text{II}}\frac{1}{n+1}\sum_{j=0}^{n}\sum_{\ell_1+\ell_2=k-1} p_{j,\ell_1}p_{n-j,\ell_2}$ car la racine reste après réduction.
- Les arbres qui commencent par $p_{\circledast}$ mais n'ont aucun fils entièrement réductible ont une probabilité $p_{\circledast}\frac{1}{n+1}\sum_{j=0}^{n}\sum_{\ell_1+\ell_2=k-1}(p_{j,\ell_1}-\mathbf{1}_{\ell_1=s}\gamma_j)(p_{n-j,\ell_2}-\mathbf{1}_{\ell_2=s}\gamma_{n-j})$
- Enfin, si $k=s$, les arbres entièrement réductibles ont une probabilité $\gamma_{n+1}\mathbf{1}_{n+1\neq s}$ (rappelons que si $n+1=s$, $\mathcal{P}$ a déjà été énuméré dans les cas précédents)

Nous avons donc pour $0\leqslant k\leqslant n$ la récurrence suivante :

$$\begin{aligned}p_{n+1,k} =&\gamma_{n+1}\mathbf{1}_{n+1\neq s}\mathbf{1}_{k=s}+\frac{p_{\text{II}}}{n+1}\sum_{j=0}^{n}\sum_{\ell_1+\ell_2=k-1}p_{j,\ell_1}p_{n-j,\ell_2}\\&+\frac{p_{\circledast}}{n+1}\sum_{j=0}^{n}\sum_{\ell_1+\ell_2=k-1}(p_{j,\ell_1}-\mathbf{1}_{\ell_1=s}\gamma_j)(p_{n-j,\ell_2}-\mathbf{1}_{\ell_2=s}\gamma_{n-j})\end{aligned}$$

Introduisons alors les polynômes $F_n(u)=\sum_{k=0}^{n}p_{n,k}u^k$ pour tout n.Là encore, $F_n(1)=1$, et $F_n'(1)=e_n$. En multipliant la récurrence pour $p_{n,k}$ par u^k puis en sommant pour k allant de 0 à $n+1$, nous obtenons la récurrence suivante pour $F_n(u)$:

$$\begin{aligned}F_{n+1}(u) =&\gamma_{n+1}\mathbf{1}_{n+1\neq s}u^s+u\frac{p_{\text{II}}}{n+1}\sum_{j=0}^{n}F_j(u)F_{n-j}(u)\\&+u\frac{p_{\circledast}}{n+1}\sum_{j=0}^{n}\left(F_j(u)-\gamma_j u^s\right)\left(F_{n-j}(u)-\gamma_{n-j}u^s\right).\end{aligned}$$

En dérivant alors cette équation en u, puis en évaluant en $u=1$:

$$\begin{aligned}e_{n+1} =&\gamma_{n+1}\mathbf{1}_{n+1\neq s}s+p_{\text{II}}+\frac{2p_{\text{II}}}{n+1}\sum_{j=0}^{n}e_j+\frac{p_{\circledast}}{n+1}\sum_{j=0}^{n}\left(1-\gamma_j\right)\left(1-\gamma_{n-j}\right)\\&+\frac{2p_{\circledast}}{n+1}\sum_{j=0}^{n}\left(e_j-s\gamma_j\right)\left(1-\gamma_{n-j}\right).\end{aligned}$$

En utilisant la récurrence sur γ_n, et $p_{\text{II}}+p_{\circledast}=1$, nous trouvons la récurrence annoncée pour e_n. ■

Proposition 5.37 .
▶ La série $E(z)$ satisfait l'équation différentielle linéaire du premier ordre suivante :

$$E'(z)=\tfrac{1}{(1-z)^2}-2p_{\circledast}\tfrac{A(z)}{1-z}+p_{\circledast}(s+1)A(z)^2+\Big(\frac{2}{1-z}-2p_{\circledast}A(z)\Big)\cdot E(z)\,,$$

qu'on peut résoudre :

$$E(z)=\frac{K(z)^{-1}}{(1-z)^2}\int_0^z G(w)K(w)dw$$

avec $K(z)=\exp(2p_{\circledast}\int_0^z A(w)dw)=v(z)^2$,
et $G(w)=1-2p_{\circledast}A(w)(1-w)+(s+1)p_{\circledast}A(w)^2(1-w)^2$. ◀

Démonstration. Il s'agit de la même technique de preuve que pour le cas unaire-binaire : nous traduisons minutieusement la récurrence en équation différentielle (il y a moins de complications grâce au terme $n+1$ qui remplace le terme $n-1$ du cas unaire-binaire).

Nous pouvons alors résoudre directement l'équation par la formule (cf Proposition A.3) de résolution d'un équation différentielle non homogène linéaire d'ordre 1, avec comme condition initiale $E(0) = 0$ (cette fois les coefficients de l'équation sont analytiques en 0 donc il n'y a pas de complication non plus, la formule classique suffit). ■

Nous retrouvons alors des propriétés similaires au cas unaire-binaire : $G(z)$ est prolongeable analytiquement en $z = 1$. De plus, l'asymptotique de $K(z) = v(z)^2$ au voisinage de 1 est simplement celle de $v(z)$ mise au carré.

Lemme 5.38.
▶ Nous retrouvons les propriétés suivantes :

a. La fonction $G(z)$ est prolongeable analytiquement en $z = 1$.

b. Les asymptotiques de $K(z) = v(z)^2$ sont obtenues en reprenant celles du cas unaire-binaire de la Proposition 5.24 en posant $p_{\mathrm{I}} = 0$. ◀

Démonstration. Pour $G(w)$, il s'agit à peu près des mêmes arguments que dans le cas unaire-binaire, car les asymptotiques de $A(z)$ et de $v(z)$ sont de la même forme : pour $p_{\circledast} \leqslant 1/2$ nous obtenons $G(w) \sim 1$, et pour $p_{\circledast} > 1/2$, nous avons $G(w) \sim 1 - 2p_{\circledast}\gamma_\infty + (s+1)p_{\circledast}\gamma_\infty^2$, et nous avons montré que cette constante était strictement positive.

Pour les asymptotiques de $v(z)$, il suffit de remarquer qu'effectivement les asymptotiques au carré de $v(z)$, de la Proposition 5.13, coïncident avec les asymptotiques de $K(z)$ de la Proposition 5.24 en posant $p_{\mathrm{I}} = 0$. ■

Nous obenons alors un théorème similaire au cas unaire-binaire :

Théorème 5.39 (Comportement asymptotique des tailles moyennes e_n).
▶ Si la probabilité p_{I} des opérateurs unaires est nulle, alors l'espérance e_n de la taille après réduction par σ d'un arbre d'expression aléatoire de taille n suivant la distribution ABR, satisfait les propriétés suivantes :

Je rappelle que dans cette section la taille d'un arbre est son nombre de nœuds internes.

- si $p_{\circledast} < 1/2$, alors $e_n \sim c'_1\, n$;
- si $p_{\circledast} = 1/2$, alors $e_n \sim c'_2\, n \log(n)^{-2}$;
- si $\frac{1}{2} < p_{\circledast} < \frac{3}{4}$, alors $e_n \sim c'_3\, n^{3-4p_{\circledast}}$;
- si $p_{\circledast} = \frac{3}{4}$, alors $e_n \sim c'_4\, \log(n)$;
- si $p_{\circledast} > \frac{3}{4}$, alors $e_n \to e_\infty$, où e_∞ est une constante strictement positive ;

avec $c'_1, \ldots, c'_4$ des constantes positives non nulles. ◀

Les constantes sont des constantes de n, mais dépendent de $s, p_{\mathrm{I}}, p_{\circledast}$.

Démonstration. Il s'agit presque de la même preuve que dans le cas unaire binaire, en posant $J(z) = \int_0^z G(w)K(w)dw$. Les raisonnements sur les asymptotiques sont transposables car les asymptotiques de $J(z)$ et de $K(z)$ ont la même forme que dans le cas unaire-binaire.

Il faut juste adapter la preuve que $\lim_{x\to 1} J(x) > 0$ lorsque $p_{\circledast} < 3/4$. Il suffit de remarquer que

$$G(x) = (1-p_{\circledast}) + p_{\circledast}(A(x)(1-x)-1)^2 + sp_{\circledast}A(x)^2(1-x)^2 \geqslant 1 - p_{\circledast} > 0$$

est strictement positif sur $[0,1[$, et $K(x) > 0$ sur $[0,1[$, donc nous avons bien $J(1) > 0$ en tant qu'intégrale de fonction continue strictement positive sur $[0,1[$. ■

Remarque 5.40.
▶ Il est possible à nouveau de calculer la limite e_∞ dans le cas $4p_\circledast > 3 - p_\mathrm{I}$ à partir de la récurrence de e_n ; nous trouvons alors :

$$e_\infty = \frac{(2p_\circledast - 1)^2 s + 1 - p_\circledast}{p_\circledast(4p_\circledast - 3)}.$$

Là encore e_∞ tend vers l'infini lorsque $p_\circledast$ s'approche du seuil $3/4$, tandis que e_∞ tends vers s lorsque $p_\circledast \to 1$. ◀

5.6 Conclusion

Dans ce chapitre, nous avons montré que les expressions aléatoires ABR ont un comportement plus riche que les uniformes vis-à-vis de la réduction liée à la présence d'un élément absorbant. D'un point de vue théorique, il est satisfaisant d'observer des phénomènes de seuils dans le choix des probabilités de l'opérateur absorbant ; les résultats obtenus donnent naturellement naissance aux extensions techniques possibles suivantes :

— Adapter tous les calculs pour le cas où l'élément absorbant est une feuille (cas $s = 1$). Nous nous attendons à obtenir le même résultat.

— S'intéresser à des spécifications qui autorisent des opérateurs d'arités plus que deux. En effet, la distribution ABR peut être adaptée pour des opérateurs ternaires ou plus. Les équations qui en découlent risquent d'être cependant bien plus techniques (l'équation de Ricatti est due à l'arité maximale 2 des opérateurs).

— Étudier des spécifications d'expressions données sous la forme de systèmes, comme dans le chapitre 2, mais en utilisant la distribution ABR.

Si nous revenons à l'objectif principal de toute cette partie, à savoir questionner la pertinence des distributions usuelles (uniformes et ABR) utilisées pour les benchmarks ou l'analyse en moyenne, le résultat de ce chapitre ne répond que partiellement à la question pour la distribution ABR. Ce n'est pas parce que notre réduction par élément absorbant n'arrive pas à réduire les arbres que la distribution n'est pas dégénérée en prenant en compte toute la sémantique des objets considérés. Par exemple, en appliquant notre résultat aux arbres booléens $\vee/\wedge$, avec comme élément absorbant $x_1 \vee \overline{x_1}$ pour l'opérateur $\vee$, nous obtenons pour $p_\vee < 1/2$ très peu de réductions. Mais il y en a en fait d'autres : en effet si $p_\vee < 1/2$ alors $p_\wedge > 1/2$, et notre résultat s'applique pour une autre réduction, concernant l'opérateur $\vee$ et son élément absorbant $x_1 \wedge \overline{x_1}$. Notons que pour $p_\vee = p_\wedge = 1/2$, nous obtenons un nombre modéré de simplifications, taille moyenne après réduction étant en $n/\log(n)^2$. Mais les travaux de [CGM11] sur la distribution ABR des arbres booléens ont montré que la distribution était en réalité complètement dégénérée, pour toutes les valeurs de $p_\vee$ [4], en précisant plus finement ce que dégénérée signifie : la distribution ABR

4. Dans l'article que nous avons publié à STACS, nous avons cité les travaux de [CGM11] en disant à tort que l'article étudiait uniquement le cas $p_\vee = p_\wedge = 1/2$; nous n'avons pas été suffisamment été attentifs lors du travail de bibliographie, les auteures étudient bien dans son ensemble la distribution ABR des arbres booléens, pour tout choix de probabilité des opérateurs.

charge uniquement les fonctions $True$ et $False$, quand la taille des arbres tend vers l'infini.

Il serait donc intéressant de prendre en compte plus de règles de sémantique pour affiner notre résultat, en autorisant par exemple plusieurs éléments absorbants, ou d'étudier des outils concrets utilisés en vérification, comme ceux utilisés dans `Spot` (ces outils utilisent notamment de nombreuses règles de réécriture pour simplifier les expressions LTL produites). Cela permettrait d'affiner la compréhension de la distribution ABR sur ces objets.

Deuxième partie

Automates de Parikh faiblement non ambigus

Introduction à la partie

En informatique théorique, un mot sur un alphabet fini désigne une suite finie ordonnée de symboles de l'alphabet. Un langage formel est un ensemble de mots sur un alphabet fini. L'informatique s'intéresse en général à des langages particuliers, qui sont décrits par des structures finies, comme des automates (automates finis, automate à pile, machines à compteurs ...), des systèmes de réécriture (grammaires, lambda-calcul), ou d'autres modèles de machines (comme les machines de Turing).

On peut voir les différents modèles de machines comme des modélisations d'ordinateurs ou de programmes avec des restrictions sur leur mémoire ou les opérations qu'ils peuvent effectuer : par exemple un automate représente un programme très simple qui se contente de lire son entrée avec une mémoire finie ; un automate de Parikh possède en plus un nombre fini de compteurs pour pouvoir faire un nombre fini d'additions ; un automate à pile est un automate qui possède une mémoire infinie accessible sous la forme d'une pile.

Tous les automates précités ne sont pas capables de reconnaître les mêmes langages. Il existe une hiérarchie entre eux, selon leur puissance de calcul, c'est-à-dire la complexité des langages qu'ils sont capables de reconnaître : c'est la *hiérarchie de Chomsky* [Cho56]. Ainsi, la classe des langages réguliers, reconnaissables par un automate fini, est strictement incluse dans la classe des langages algébriques, reconnaissables par une grammaire hors-contexte, qui est strictement incluse dans la classe des langages contextuels, reconnaissables par une grammaire contextuelle. Cette dernière classe est elle-même strictement incluse dans la classe des langages récursivement énumérables, reconnaissables par une machine de Turing.

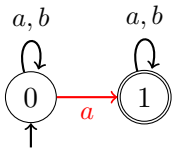
Cette hiérarchie a été étendue et largement complétée par des modèles d'automates divers (automates d'arbres [CDG+07], automate de Parikh [KR02], machines à compteurs [BB74, Iba78, Iba16], machines à registre, etc.), au point qu'on ne peut plus vraiment parler de hiérarchie, car les différents modèles ne sont plus strictement emboîtés les uns dans les autres, et certains modèles sont incomparables. Dans cette partie de la thèse, nous nous sommes intéressés à une petite portion de la hiérarchie de Chomsky, constituée des langages algébriques et des modèles de machines à compteurs, et plus précisément à leurs variantes non ambiguës (que je vais préciser dans la suite).

Calcul d'un automate et non ambiguïté

Une exécution d'un automate sur un mot w en entrée est généralement appelé un *calcul* de l'automate sur w (pour les modèles décrits par des systèmes de réécriture, comme les grammaires, on parlera plutôt de dérivation) ; un calcul est dit acceptant s'il finit dans une configuration particulière, dite acceptante, il est rejetant sinon. Les configurations acceptantes varient selon les modèles : avoir lu le mot en entier et être dans un état final pour les automates finis ou les automates à pile, ou la variante d'avoir lu le mot et de finir avec une pile vide pour les automates à pile, ou encore simplement être dans un état acceptant pour les machines de Turing…Il s'agit bien *d'un* calcul de l'automate sur w, et non pas *du* calcul, car en toute généralité les automates peuvent avoir plusieurs exécutions sur une même entrée : ces automates sont appelés non déterministes, car leur comportement n'est pas déterminé au cours de la lecture de l'entrée par leur configuration et la lettre courante du mot en cours de lecture.

Les automates sont ainsi un peu plus que de simples accepteurs de langages prenant un mot en entrée et renvoyant `accept` si le mot est accepté, et `reject` sinon. Un automate associe plutôt à chaque mot donné en entrée l'ensemble (potentiellement infini) de ses calculs sur cette entrée. Le mot est alors accepté si son ensemble de calculs contient un calcul acceptant, refusé si toutes ses calculs sont rejetants.

On peut voir en quelque sorte un calcul acceptant d'un automate comme une preuve d'appartenance au langage, car il calcule une décomposition de ce mot à l'aide des transitions de l'automate, et cette décomposition prouve son appartenance au langage. Prenons un exemple très simple, considérons le langage des mots qui contiennent au moins une lettre a. L'automate suivant reconnaît ce langage :



Un calcul acceptant de cet automate prouve que le mot en entrée appartient bien au langage en mettant en évidence, par la transition rouge, une lettre a présente dans le mot. On retrouve ce phénomène dans d'autres classes de langages, par exemple, les arbres de dérivation des grammaires reconnaissant les mots bien parenthésés fournissent une décomposition d'un mot en appariant les différentes parenthèses du mot.

Un automate déterministe, par opposition aux automates non déterministes, est un automate dont le comportement est entièrement déterminé par sa configuration et la lettre lue en entrée : ainsi un automate déterministe complet peut être vu comme une fonction qui a tout mot associe un unique calcul, qu'il soit acceptant ou non ; il n'y a qu'une seule exécution possible pour tout mot donné en entrée. Généralement les versions déterministes jouissent de propriétés de clôture plus étendues, ont parfois plus de propriétés décidables, et dans certains cas particuliers (comme pour les automates finis, ou les machines de Turing) sont équivalentes aux versions non déterministes. Cependant, les modèles déterministes présentent en général plusieurs inconvénients :

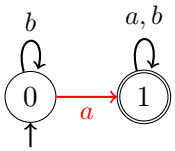
— ils sont parfois beaucoup trop peu expressifs par rapport à leur version non déterministe ; par exemple le langage des palindromes n'est pas reconnaissable

par un automate à pile déterministe

— ils peuvent être beaucoup plus gros : l'automate fini déterministe et complet reconnaissant le langage $\Sigma^* a \Sigma^n$ a au moins 2^n états.

Une autre sous-classe naturelle associée à un modèle d'automate est son ensemble d'automates non ambigus : ce sont des automates du modèle qui ne sont pas forcément déterministes, mais qui ne peuvent accepter un mot que d'une seule manière : tout mot accepté par l'automate possède un unique calcul acceptant. Contrairement aux automates déterministes, le comportement local d'un automate non ambigu n'est pas déterminé par la portion locale du mot qu'il est en train de lire. Par contre, si le mot en entrée est accepté, le comportement global de l'automate sur tout le mot est uniquement déterminé. Il s'agit d'un bon compromis à mi-chemin entre les versions déterministes trop rigides, et les versions non déterministes trop complexes à étudier : le langage des palindromes est ainsi reconnaissable par un automate à pile non ambigu, le langage $\Sigma^* a \Sigma^n$ est reconnaissable par un automate fini non ambigu de taille n, et certains problèmes indécidables dans le cas non déterministe deviennent décidables dans le cas non ambigu (par exemple, l'universalité est indécidable pour les langages algébriques, mais décidable pour les langages algébriques non ambigus).

Un automate non ambigu peut donc être vu comme une fonction qui à un mot de son langage associe *son* calcul acceptant. En interprétant un calcul comme une preuve d'appartenance, cela signifie que l'automate est capable de décomposer le mot d'une unique façon pour prouver son appartenance au langage. Ce modèle est plus souple que les version déterministes, car il autorise plusieurs calculs rejetants pour un même mot. Si on reprend l'exemple de l'automate précédent qui accepte les mots qui ont au moins une lettre a, il y a autant de calculs acceptants pour un mot w que d'occurrences de la lettre a. L'automate suivant est non ambigu (et même déterministe) :



L'automate associe ainsi à tout mot de son langage une unique décomposition, en marquant la première occurrence de la lettre a. On retrouve cette idée pour d'autres modèles de calcul, par exemple le langage des mots qui ont un a en leur milieu est reconnu par un automate à pile non ambigu, et un calcul acceptant de cet automate fournit une décomposition du mot qui identifie exactement le milieu du mot.

Intrinsèque ambiguïté

Un langage est dit *intrinsèquement ambigu* pour une classe d'automates s'il n'est reconnaissable par aucun automate non ambigu de cette classe. La notion d'ambiguïté, et donc aussi l'intrinsèque ambiguïté, dépendent naturellement de la classe d'automate étudiée. Trouver des langages intrinsèquement ambigus est très utile pour comprendre les limites d'une classe d'automates, et les décompositions qu'elle est capable d'effectuer sur des mots.

L'intrinsèque ambiguïté est une notion compliquée à étudier. Savoir pour une classe d'automates si un langage est intrinsèquement ambigu est généralement indécidable

([Gre68]) : c'est le cas pour les langages algébrique, ou encore les automates de Parikh que nous verrons dans la suite. Pour les langages algébriques, elle est doublement indécidable, car savoir si un automate à pile donné est ambigu est aussi un problème indécidable.

Prouver qu'un langage donné est intrinsèquement ambigu pour un modèle s'avère en général difficile, car il faut être capable de montrer que le modèle est incapable de calculer une décomposition unique pour tous les mots du langage. Les premières preuves d'intrinsèque ambiguïté pour les langages algébriques étudient ainsi la forme des dérivations de toutes les grammaires hors-contexte reconnaissant le langage. Ces preuves sont généralement très techniques (voir par exemple [Cre72, GU66, Ogd68]), mais elles prouvent souvent plus de choses que la simple intrinsèque ambiguïté : elles arrivent généralement à trouver la forme des différentes décompositions calculées par une grammaire pour un mot du langage, ce qui est très intéressant pour mieux comprendre les calculs des langages algébriques. Malheureusement, la technicité de ces preuves les limitent généralement à des langages suffisamment simples.

En 1987, [Fla87] trouve une nouvelle technique simple pour prouver l'intrinsèque ambiguïté de certains langages algébriques complexes, qui vient compléter les techniques précédentes. Flajolet utilise la contraposée du théorème de Chomsky-Schützenberger [CS63], qui énonce que la série génératrice d'un langage algébrique non ambigu est algébrique, et développe des critères pour montrer qu'un langage algébrique est intrinsèquement ambigu : si on peut prouver que la série génératrice d'un langage n'est pas algébrique, alors il est intrinsèquement ambigu. Ses critères sont si efficaces que l'article [Fla87] montre en quelques pages l'intrinsèque ambiguïté de très nombreux langages, et démontre plusieurs conjectures de l'époque. Les critères de Flajolet sont un formidable raccourci de preuve pour montrer l'intrinsèque ambiguïté de certains langages algébriques. En revanche ils ne donnent pas d'information sur les formes des différentes dérivations du langage étudié, et ils s'appliquent à des langages suffisamment complexes pour avoir une série non algébrique. En cela, ils ne sont pas supérieurs mais bien complémentaires aux techniques précédentes, qui étudiaient les dérivations.

Extension de l'approche de Flajolet

Le théorème de Chomsky-Schützenberger prouve en réalité que la série génératrice des calculs acceptants d'un langage algébrique non ambigu est algébrique, et c'est la condition de non ambiguïté qui permet de remonter à la série du langage. Rappelons qu'un automate non ambigu peut être vu comme une fonction qui a tout mot de son langage associe *son* calcul acceptant ; réciproquement tout calcul acceptant est étiqueté par un mot du langage par définition. Un automate non ambigu fournit donc une bijection entre les mots du langages et son ensemble de calculs acceptants. Si on définit la taille d'un calcul acceptant comme la longueur du mot qu'il reconnaît, alors la série génératrice des calculs acceptants d'un automate est égale, par cette bijection, à la série génératrice du langage accepté par l'automate.

Par exemple, si un langage L, de série génératrice $L(x)$ est reconnu par un automate non ambigu $\mathcal{A}$, dont la série génératrice des calculs acceptants est $A(x)$, alors :

$$L(x) = A(x)\,.$$

À gauche de l'égalité, nous avons généralement une série $L(x)$ que nous connaissons,

c'est la série génératrice du langage que nous étudions. À droite, il s'agit d'une série issue des calculs d'un modèle d'automates. Selon la façon dont ce modèle d'automates est décrit, les séries des calculs acceptants associés ont des propriétés particulières : par exemple, la série des calculs acceptants d'un automate fini est rationnelle, c'est-à-dire qu'elle s'exprime sous la forme $A(x) = P(x)/Q(x)$ avec P et Q deux polynômes ; la série des calculs d'un automate à pile (ou la série des dérivation d'une grammaire hors-contexte) est quant à elle algébrique, par le théorème de Chomsky-Schützenberger, c'est-à-dire qu'elle satisfait une équation polynomiale de la forme $P(x, A(x)) = 0$ avec $P(x, Y)$ un polynôme.

L'approche de Flajolet est générale, et peut s'appliquer à d'autres classes : si la série génératrice des calculs d'un modèle d'automates vérifie une propriété $\mathcal{P}$, alors si un langage donné a une série génératrice qui ne vérifie par $\mathcal{P}$, il est intrinsèquement ambigu pour ce modèle. Dans le cas des langages algébriques, la propriété utilisée par Flajolet est *être algébrique*. Au delà de la question de l'intrinsèque ambiguïté, la propriété $\mathcal{P}$ sur les séries peut avoir des conséquences algorithmiques : dans le cas des automates finis, la propriété d'avoir une série génératrice rationnelle a été utilisée (implicitement) pour concevoir des algorithmes en temps polynomial pour le test d'inclusion et l'universalité des automates finis non ambigus [SI85].

Pour généraliser ces approches à d'autres modèles que les automates finis et les automates à pile, il faut à la fois être capable d'identifier des classes d'automates dont les séries des calculs acceptants ont des propriétés suffisamment fines pour discriminer certains langages reconnus par le modèle, et être capable de fournir des critères qui permettent de démontrer qu'une série ne vérifie pas la propriété en question.

Une tentative d'extension partielle a été proposée par [Mas93] : en ajoutant des contraintes semilinéaires au nombre d'occurrences de lettres d'un langage algébrique non ambigu, l'auteur a formalisé une famille de langages LCL spécialement conçue pour avoir une série holonome, c'est-à-dire qui vérifie une équation différentielle linéaire à coefficients polynomiaux. Les séries holonomes bénéficient de nombreuses propriétés de clôture [Sta80, Lip89, Lip89], et sont largement étudiées en calcul formel (voir par exemple [SZ94, Chy94]) ; ainsi il existe plusieurs critères permettant de montrer qu'une série n'est pas holonome. Nous pouvons en fait retrouver l'idée de Massazza [Mas93] dans l'article de [Lip88], qui propose, comme application de sa preuve que les séries holonomes sont closes par diagonale, de contraindre le support de certaines séries holonomes par des ensembles de contraintes linéaires sur les occurrences de chaque variable.

Si l'extension proposée par [Mas93] est prometteuse du côté des séries, elle pêche du côté de la non ambiguïté : la classe LCL n'est reliée à aucune classe d'automate standard. Plus récemment, Castiglione et Massazza [CM17] ont essayé d'aborder la question en introduisant une autre classe de langages, la classe RCM, elle aussi conçue dans le but d'avoir une série génératrice holonome. Mais les auteurs n'ont pas réussi à trouver un modèle d'automates capturant leur famille de langages, même s'ils ont conjecturé que leur classe RCM contenait les machines unidirectionnelles à inversion bornée déterministes (RBCM en abrégé). Massazza a ensuite prouvé la conjecture pour deux sous-classes particulières de RBCM déterministes unidirectionnelles [Mas17, Mas18].

Par conséquent, aussi bien la classe LCL que la classe RCM sont des extensions partielles de l'approche de Flajolet, car il leur manque un modèle d'automate sous-jacent, dont les calculs acceptants ont une série holonome. Dans cette partie, nous

comblons ce vide en étudiant deux modèles d'automates bien connus de la littérature (les automates de Parikh, ou leur modèle équivalent les RBCM, ainsi que leur extension avec une pile [KR02, Iba78]) ; nous prouvons notamment que la série génératrice des calculs acceptants de ces deux modèles est holonome, puis nous étudions les conséquences de cette propriété, pour prouver l'intrinsèque ambiguïté de certains langages, ou encore pour étudier l'algorithmique du problème de l'inclusion pour les modèles non ambigus des automates de Parikh.

Plan de la partie

Une grande partie du contenu des trois chapitres de cette partie a été publiée à ICALP en 2020, sous une forme plus condensée. Cette partie est une version développée de l'article initial, qui faisait 60 pages avec annexes. Elle s'organise comme suit :

— Dans le chapitre 6, nous introduisons différents modèles de machines à compteurs non ambiguës, leurs propriétés de clôtures, et leurs extensions. Ce chapitre technique prend un point de vue purement automate ; il a pour but de préciser les liens entre différents modèles d'automates non ambigus. Nous démontrons par exemple la conjecture de [CM17], en montrant que la classe RCM coïncide avec la classe des RBCM non ambiguës, ou encore des automates de Parikh faiblement non ambigus.

— Dans le chapitre 7, nous nous intéressons principalement aux techniques permettant de prouver l'intrinsèque ambiguïté d'un langage. En étudiant le lien entre séries holonomes et langages, nous étendons notamment les critères d'intrinsèque ambiguïté de Flajolet à la classe des langages algébriques de Parikh. De façon orthogonale, nous démontrons aussi l'intrinsèque faible ambiguïté d'un langage de Parikh dont la série génératrice est rationnelle, par des arguments d'itération.

— Enfin, le chapitre 8 étudie les conséquences algorithmiques de ce lien entre séries holonomes et langages formels. Nous nous concentrons sur le problème d'inclusion des automates de Parikh faiblement non-ambigus, dont la décidabilité peut être déduite de travaux de Castiglione et Massazza [CM17]. La contribution du chapitre est un résultat de décidabilité effective : nous arrivons à trouver une taille concrète B qui borne la longueur du plus petit mot qui réfute l'inclusion des deux langages. Cette borne B est obtenue par une analyse minutieuse des preuves établissant les propriétés de clôture des séries holonomes (à plusieurs variables), notamment la clôture par produit d'Hadamard.

Ce chapitre est en partie issu de l'article de conférence [BCKN20].

6

Modèles de machines à compteurs non ambiguës

6.1 Introduction

Dans ce chapitre, nous présentons différents modèles de machines à compteurs existant dans la littérature, et nous nous intéressons plus particulièrement à leurs variantes non ambiguës. Dans la section 6.2, nous introduisons la classe des automates de Parikh faiblement non ambigus, puis nous étudions dans la section 6.3 plusieurs modèles d'automates équivalents, comme les Reversal Bounded Counter Machines non ambiguës, ou encore la classe RCM. Nous insérons ainsi précisément la classe RCM dans la hiérarchie des machines à compteurs. Ceci démontre une version plus forte de la conjecture de [CM17], qui annonçait que les langages reconnaissables par des RBCM déterministes unidirectionnelles étaient inclus dans la classe RCM. Dans la section 6.4 et la section 6.5, nous effectuons la même démarche en ajoutant une pile au modèle. Si la plupart des équivalences de modèles de ce chapitre ne sont pas surprenantes au vu des équivalences déjà existantes dans la littérature pour les modèles non déterministes, elles demandent toutefois à être vérifiées avec soin. Par ailleurs, elles confirment la robustesse de la notion de non-ambiguïté que nous avons étudiée sur ces modèles.

6.2 Automates de Parikh faiblement non ambigus

6.2.1 Automate de Parikh, calcul acceptant, faible non-ambiguïté

Nous introduisons dans cette section les automates de Parikh étudiés dans [KR02], ainsi que leur version faiblement non ambiguë. De façon informelle, un automate de Parikh est un automate dont les transitions sont étiquetées par un couple lettre-vecteur. Lors d'un calcul de l'automate, on concatène les lettres rencontrées dans les transitions, et on additionne les vecteurs, composante par composante. Un chemin de l'automate est donc étiqueté par un couple mot-vecteur, et est appelé calcul acceptant s'il commence dans un état initial, termine dans un état final, et si le vecteur somme

Le lecteur est invité à lire le chapitre 1 des préliminaires pour la définition d'un ensemble semilinéaire.

appartient à un ensemble semilinéaire d'acceptation associé à l'automate. De façon plus formelle :

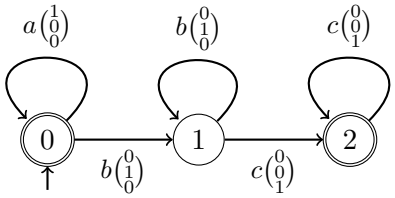
Définition 6.1 (Automate de Parikh [KR02, KR03]).
▶ Un *automate de Parikh* de dimension $d \geqslant 1$ est un tuple $\mathcal{A} = (\Sigma, Q, q_I, F, C, \Delta)$, où Σ désigne l'alphabet, Q l'ensemble d'états, $q_I \in Q$ est l'état initial, et $F \subseteq Q$ l'ensemble des états finaux. Enfin $C \subseteq \mathbb{N}^d$ est l'ensemble semilinéaire d'acceptation, et $\Delta \subseteq Q \times (\Sigma \times \mathbb{N}^d) \times Q$ est l'ensemble des transitions.

Voir la sous-section 1.1.3 des préliminaires pour la définition d'un ensemble semilinéaire.

Un *calcul* de l'automate est une séquence $q_0 \xrightarrow{a_1,\boldsymbol{v_1}} q_1 \xrightarrow{a_2,\boldsymbol{v_2}} q_2 \cdots q_{n-1} \xrightarrow{a_n,\boldsymbol{v_n}} q_n$ telle que pour tout $i \in [1, n]$, $(q_{i-1}, (a_i, \boldsymbol{v_i}), q_i)$ est une transition de Δ. Un calcul de cette forme est étiqueté par le couple $(a_1 \cdots a_n, \boldsymbol{v_1} + \cdots + \boldsymbol{v_n}) \in \Sigma^* \times \mathbb{N}^d$. Il est acceptant si $q_0 = q_I$, si l'état q_n est final et si le vecteur somme $\boldsymbol{v_1} + \cdots + \boldsymbol{v_n}$ appartient à l'ensemble semilinéaire C. On dit alors que le mot w est accepté par $\mathcal{A}$. Le *langage accepté* par $\mathcal{A}$ désigne l'ensemble des mots acceptés par l'automate, et est noté $\mathcal{L}(\mathcal{A})$.

On appellera *langages de Parikh* l'ensemble des langages reconnus par un automate de Parikh. ◀

Exemple 6.2 (Le langage $a^n b^n c^n$).
▶ Le langage $L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ (qui n'est pas algébrique) est accepté par l'automate de Parikh suivant :



$$C = \{(n_1, n_2, n_3) \in \mathbb{N}^3 \mid n_1 = n_2 \text{ et } n_2 = n_3\}$$

◀

Nous définissons alors la notion de non-ambiguïté qui nous a intéressé dans toute cette partie de la thèse :

Définition 6.3 (Automate de Parikh faiblement non ambigu).
▶ Un automate de Parikh est *faiblement non ambigu* si pour tout mot de Σ^*, il existe au plus un calcul acceptant étiqueté par ce mot.

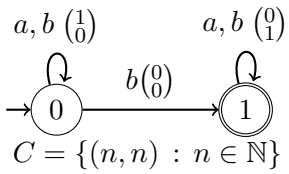
Un langage de Parikh faiblement non ambigu est un langage reconnaissable par un automate de Parikh faiblement non ambigu. On note wuPA la classe des langages de Parikh faiblement non ambigus.

Weakly Unambiguous Parikh Automata

Un langage de Parikh qui n'est pas faiblement non ambigu est dit *intrinsèquement faiblement ambigu*, c'est-à-dire qu'aucun automate de Parikh qui le reconnaît n'est faiblement non ambigu. ◀

Remarque 6.4 (À propos du terme faiblement non ambigu).
▶ La définition de la faible non-ambiguïté coïncide en fait avec la définition standard de la non-ambiguïté. Mais le nom d'automates de Parikh non ambigus est déjà utilisé dans la littérature pour désigner une classe moins expressive, définie par une contrainte plus forte que la non-ambiguïté classique. Nous renvoyons le lecteur à la sous-section 6.2.3 pour une explication plus détaillée de cette classe. ◀

Exemple 6.5 (Langage des mots de longueur impaire qui ont un b au milieu).
▶ Considérons l'automate suivant, sur l'alphabet $\Sigma = \{a, b\}$:



Un mot w qui contient la lettre b a autant de calculs joignant l'état initial à l'état final qu'il a de décompositions de la forme $w = ubv$, avec $u, v \in \Sigma^*$. Chaque calcul associé à la décomposition $w = ubv$ est par ailleurs étiqueté par le vecteur $(|u|, |v|)$.

Un mot w a donc un calcul acceptant si et seulement si w est de la forme $w = ubv$ avec $|u| = |v|$, et si c'est le cas, c'est le seul calcul acceptant pour w. Donc le langage de l'automate ci-dessus est le langage des mots de longueur impaire qui ont un b au milieu, et cet automate est faiblement non ambigu. ◀

Remarque 6.6 (Représentation du semilinéaire).
▶ Dans la définition d'un automate de Parikh, nous n'avons pas précisé sous quelle forme était représenté le semilinéaire C – par une union finie d'ensembles linéaires, par un automate, par une formule de Presburger ? Le choix de la représentation (notamment si le semilinéaire est sous une forme non ambiguë) peut modifier la complexité des problèmes associés à l'automate. Dans ce chapitre nous nous intéressons cependant uniquement au pouvoir de calcul des automates présentés, donc nous choisirons librement une représentation adaptée à chaque preuve. Il s'agira souvent d'une formule de Presburger. ◀

6.2.2 Quelques propriétés de clôture

Cette section arrive un peu trop tôt dans le chapitre, ou la thèse, car nous ne disposons pas à ce stade de suffisamment d'outils pour faire certaines preuves. J'ai décidé cependant de rassembler ici quelques propriétés de (non) clôture des automates de Parikh faiblement non ambigus, car elle seraient sinon dispersées tout au long du chapitre et du suivant. Il faut donc voir cette section comme une annonce de résultats, qui répond à certaines interrogations classiques que l'on se pose lorsque l'on introduit une classe d'automates. Par ailleurs, elle permet aussi de mieux comparer la classe des automates de Parikh faiblement non ambigus avec celle des automates de Parikh non ambigus présentés à la section suivante. Bien entendu, nous avons fait soigneusement attention à ne pas insérer par erreur des boucles d'interdépendances avec les preuves du prochain chapitre.

Dans un premier temps, nous énonçons quelques propriétés de clôture élémentaires :

Proposition 6.7 (Quelques propriétés de clôture).
▶ Soient $\mathcal{L}_1$ et $\mathcal{L}_2$ deux langages de Parikh faiblement non ambigus sur un alphabet Σ. Soit w un mot dans Σ^*. Alors :

a. L'intersection $\mathcal{L}_1 \cap \mathcal{L}_2$ est un langage de Parikh faiblement non ambigu.

b. Le langage quotient $w^{-1}\mathcal{L}_1 := \{v \in \Sigma^* \mid wv \in \mathcal{L}_1\}$ est un langage de Parikh faiblement non ambigu.

c. Si L_1 et L_2, sont disjoints alors l'union disjointe $L_1 \uplus L_2$ est un langage de Parikh faiblement non ambigu.

d. Si le complémentaire $\overline{L_1}$ est un langage de Parikh faiblement non ambigu, alors $L_1 \cup L_2$ est un langage de Parikh faiblement non ambigu.

◀

Démonstration. **a.** Soit $\mathcal{A} = (\Sigma, Q_\mathcal{A}, q_{0,\mathcal{A}}, F_\mathcal{A}, C_\mathcal{A}, \Delta_\mathcal{A})$ un automate de Parikh faiblement non ambigu , de dimension $d_\mathcal{A}$, tel que $\mathcal{L}_1 = \mathcal{L}(\mathcal{A})$.
De même, soit $\mathcal{B} = (\Sigma, Q_\mathcal{B}, q_{0,\mathcal{B}}, F_\mathcal{B}, C_\mathcal{B}, \Delta_\mathcal{B})$ un automate de Parikh faiblement non ambigu, de dimension $d_\mathcal{B}$, reconnaissant $\mathcal{L}_2$.
Nous introduisons l'automate produit $\mathcal{C}$, de dimension $d_\mathcal{C} = d_\mathcal{A} + d_\mathcal{B}$, d'ensemble d'états $Q_\mathcal{C} = Q_\mathcal{A} \times Q_\mathcal{B}$, et tel que pour tout couple de transitions $(t_1, t_2) \in \Delta_\mathcal{A} \times \Delta_\mathcal{B}$, étiquetées par la même lettre $a \in \Sigma$, avec $t_1 = (q_1, (a, \boldsymbol{v_1}), q_1')$ et $t_2 = (q_2, (a, \boldsymbol{v_2}), q_2')$, alors $((q_1, q_2), (a, (\boldsymbol{v_1}, \boldsymbol{v_2})), (q_1', q_2')) \in \Delta_\mathcal{C}$. Nous définissons aussi l'ensemble des états finaux $F_\mathcal{C} = F_\mathcal{A} \times F_\mathcal{B}$, et l'état initial est $q_{0,\mathcal{C}} = (q_{0,\mathcal{A}}, q_{0,\mathcal{B}})$. Enfin, le semilinéaire $C_\mathcal{C}$ est défini par

Le vecteur $(\boldsymbol{v_1}, \boldsymbol{v_2})$ représente le vecteur de $\mathbb{N}^{d_\mathcal{A}+d_\mathcal{B}}$ dont les $d_\mathcal{A}$ premières coordonnées sont celles de $\boldsymbol{v_1}$, et les $d_\mathcal{B}$ dernières sont celles de $\boldsymbol{v_2}$.

$$C_\mathcal{C} := \{(x_1, \dots, x_{d_\mathcal{A}}, x_{d_\mathcal{A}+1}, \dots, x_{d_\mathcal{C}}) \mid (x_1, \dots, x_{d_\mathcal{A}}) \in C_\mathcal{A} \text{ et } (x_{d_\mathcal{A}+1}, \dots, x_{d_\mathcal{C}}) \in C_\mathcal{B}\}$$

Les calculs de $\mathcal{C}$ sont classiquement en bijection avec les couples de calculs de $\mathcal{A}$ et $\mathcal{B}$ étiquetés par le même mot. Les $d_\mathcal{A}$ premières coordonnées des vecteurs de $\mathcal{C}$ contiennent les vecteurs des calculs de $\mathcal{A}$, tandis que les $d_\mathcal{B}$ dernières coordonnées contiennent ceux des calculs de $\mathcal{B}$. Ainsi $\mathcal{C}$ est bien aussi faiblement non ambigu, et reconnaît l'intersection des deux langages.
Nous notons $\mathbf{0}_k$ le vecteur $\underbrace{(0, \dots, 0)}_{k \text{ fois}}$ pour tout $k \geqslant 1$.

Il est en fait possible de prendre $d_\mathcal{C} = \max(d_\mathcal{A}, d_\mathcal{B}) + 1$, en ajoutant juste un bit dans la dernière dimension pour préciser quel automate est simulé.

Nous aurons besoin de cette construction pour le chapitre 8.

Remarquons que si $C_\mathcal{A}$ et $C_\mathcal{B}$ sont donnés sous une présentation non ambiguë :

$$\begin{aligned} C_\mathcal{A} &= \biguplus_{i \in [1, k_\mathcal{A}]} \boldsymbol{c_i} + P_i^* \\ C_\mathcal{B} &= \biguplus_{i \in [1, k_\mathcal{B}]} \boldsymbol{d_i} + R_i^* \end{aligned}$$

alors on obtient facilement une présentation non ambiguë de $C_\mathcal{C}$:

$$C_\mathcal{C} = \biguplus_{i \in [1, k_\mathcal{A}] j \in [1, k_\mathcal{B}]} (\boldsymbol{c_i}, \boldsymbol{d_j}) + Q_{i,j}^*$$

avec pour $i \in [1, k_\mathcal{A}]$ et $j \in [1, k_\mathcal{B}]$,

$$Q_{i,j} = \{(\boldsymbol{p}, \mathbf{0}_{d_\mathcal{B}}) \mid \boldsymbol{p} \in P_i\} \cup \{(\mathbf{0}_{d_\mathcal{A}}, \boldsymbol{r}) \mid \boldsymbol{r} \in R_j\ .\}$$

b. Le second point sera très facile à démontrer une fois prouvée en sous-section 6.3.3 l'équivalence avec la variante des automates de Parikh faiblement non ambigus à ε-transitions (voir Proposition 6.24) .

c. Il s'agit de la construction classique d'union d'automates. En reprenant les notations de la preuve du **a.**, nous introduisons l'automate union $\mathcal{C}$, de dimension $d_\mathcal{C} = d_\mathcal{A} + d_\mathcal{B}$, d'ensemble d'états $Q_\mathcal{C} = Q_\mathcal{A} \uplus Q_\mathcal{B} \uplus \{q_{0,\mathcal{C}}\}$ (on suppose quitte à renommer les états que ces ensembles sont disjoints), avec $q_{0,\mathcal{C}}$ un nouvel état que l'on désigne comme initial. Pour toute transition $t_1 =$

$(q_1,(a,\boldsymbol{v_1}),q_1') \in \Delta_{\mathcal{A}}$, alors $(q_1,(a,(\boldsymbol{v_1},\mathbf{0}_{d_{\mathcal{B}}})),q_1') \in \Delta_{\mathcal{C}}$, et si $q_1 = q_{0,\mathcal{A}}$, on ajoute aussi la transition $(q_{0,\mathcal{C}},(a,(\boldsymbol{v_1},\mathbf{0}_{d_{\mathcal{B}}})),q_1')$ à $\Delta_{\mathcal{C}}$. De même, pour toute transition $t_2 = (q_2,(a,\boldsymbol{v_2}),q_2') \in \Delta_{\mathcal{B}}$, alors $(q_2,(a,(\mathbf{0}_{d_{\mathcal{A}}},\boldsymbol{v_2})),q_2') \in \Delta_{\mathcal{C}}$, et si $q_2 = q_{0,\mathcal{B}}$, alors $(q_{0,\mathcal{C}},(a,(\mathbf{0}_{d_{\mathcal{A}}},\boldsymbol{v_2})),q_2') \in \Delta_{\mathcal{C}}$. Nous définissons aussi l'ensemble des états finaux $F_{\mathcal{C}} = F_{\mathcal{A}} \cup F_{\mathcal{B}}$ ($\cup \{q_{0,\mathcal{C}}\}$ si $\varepsilon \in L_1 \cup L_2$). Enfin, le semilinéaire $C_{\mathcal{C}}$ est défini par

$$C_{\mathcal{C}} := \{(\boldsymbol{v},\mathbf{0}_{d_{\mathcal{B}}}) \mid \boldsymbol{v} \in C_{\mathcal{A}}\} \cup \{(\mathbf{0}_{d_{\mathcal{A}}},\boldsymbol{v}) \mid \boldsymbol{v} \in C_{\mathcal{B}}\}.$$

L'automate $\mathcal{C}$ reconnaît facilement $L_1 \cup L_2$ en décidant de façon non déterministe de lancer un calcul dans l'un des automates $\mathcal{A}$ ou $\mathcal{B}$. Les états et transitions étant bien disjointes, et les dimensions bien séparées, les calculs de $\mathcal{A}$ et $\mathcal{B}$ ne se mélangent pas dans $\mathcal{C}$. L'automate $\mathcal{C}$ est faiblement non ambigu : tout mot a au plus un calcul acceptant dans $\mathcal{A}$, au plus un calcul acceptant dans $\mathcal{B}$, et donc comme l'union des langages est disjointe, au plus un calcul acceptant dans $\mathcal{C}$.

d. Immédiat en utilisant l'union disjointe $L_1 \cup L_2 = L_1 \uplus (\overline{L_1} \cap L_2)$. ■

Remarque 6.8 (Stabilité par union et complémentaire).
▶ Nous ne savons pas si les automates de Parikh faiblement non ambigus sont stables par complémentaire ou union. Si jamais ils le sont par complémentaire, alors ils sont stables par union, par la proposition précédente (on peut aussi le voir par l'égalité $L_1 \cup L_2 = \overline{\overline{L_1} \cap \overline{L_2}}$). ◀

Nous admettons dans un premier temps la proposition suivante, qui sera démontrée au prochain chapitre :

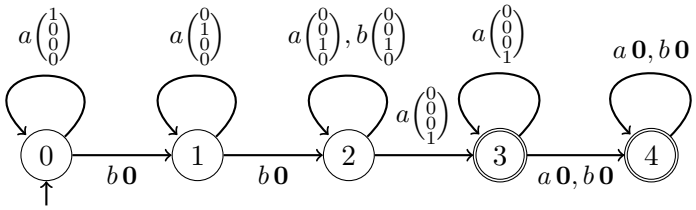
Proposition 6.9 (Le langage de Shamir est intrinsèquement faiblement ambigu).
▶ Le langage $\mathcal{S} = \{a^n b v_1 a^n v_2 : n \geqslant 1, v_1, v_2 \in \{a,b\}^*\}$ est un langage de Parikh intrinsèquement faiblement ambigu. ◀

Cf. la Proposition 7.55 du chapitre suivant.

Ce contre-exemple a comme corollaire la proposition suivante, qui contraste avec la stabilité de la classe par quotient à gauche par un mot fixé :

Proposition 6.10 (Non stabilité par quotient à gauche).
▶ La classe des automates de Parikh faiblement non ambigus n'est pas stable par quotient à gauche par un langage rationnel. ◀

Démonstration. Il suffit pour la montrer de considérer le langage de Parikh faiblement non ambigu $L = \{a^m b a^n b v a^n w \ : \ v,w \in \{a,b\}^*, |v| = m \in \mathbb{N}, \text{ et } n \geqslant 1\}$. Il est en effet reconnu par l'automate faiblement non ambigu suivant :



$$C = \{(m,n,m',n') \in \mathbb{N}^4 \mid m = m' \text{ et } n = n'\}$$

La transition 2 vers 3 force $n \geqslant 1$.

Mais $(a^*b)^{-1}L = \mathcal{S}$ est intrinsèquement faiblement ambigu. ■

6.2.3 Comparaison avec les automates de Parikh non ambigus

La condition de non-ambiguïté que nous avons introduite pour les automates de Parikh est standard : tout mot est accepté par au plus un calcul acceptant. Si tout le monde semble s'accorder sur cette formulation de la non-ambiguïté, il y a une imprécision sur ce qu'on appelle un calcul acceptant pour les automates de Parikh.

Dans cette thèse, un calcul acceptant est un calcul qui accepte un mot du langage, autrement dit une preuve de l'appartenance d'un mot au langage de Parikh : il s'agit donc d'un chemin de l'automate, partant d'un état initial jusqu'à un état final, et *étiqueté par un vecteur appartenant au semilinéaire* C.

Il existe dans la littérature une autre notion de non-ambiguïté pour les automates de Parikh, pour laquelle un calcul acceptant est simplement un calcul joignant un état initial à un état final, indépendamment du vecteur qui l'étiquette. Avec cette définition, un calcul acceptant peut donc être étiqueté par un mot qui n'est pas dans le langage accepté par l'automate.

Cette classe d'automates est appelée dans la littérature *automates de Parikh non ambigus*. Un automate de Parikh est non ambigu si pour tout mot w, il existe un seul calcul joignant un état initial à un état final étiqueté par w : autrement dit l'automate fini obtenu en effaçant les vecteurs est non ambigu. Il s'agit d'une contrainte assez forte, et tout automate de Parikh non ambigu est faiblement non ambigu (l'inclusion est stricte, voir Proposition 6.11).

Les automates de Parikh non ambigus ont été introduits et étudiés par Cadilhac et al. dans [CFM13]. Pour être exact, il s'agissait d'une variante équivalente aux automates de Parikh, appelée *Unambiguous Constrained Automata*. Dans les discussions de sa thèse, Cadilhac propose une définition équivalente à cette classe dans le modèle des automates de Parikh, qu'il appelle automates de Parikh non ambigus.

Ces restrictions fortes dans la définition des automates de Parikh sont contrebalancées par la stabilité remarquable de la classe sous de nombreuses opérations, dont les opérations booléennes (union, intersection, complémentaire) et les quotients à gauche et à droite. Cette stabilité s'explique par leur équivalence avec des modèles déterministes de machines (dans [CFM13], les auteurs montrent l'équivalence de la classe avec un modèle déterministe d'automate à registre, et plus récemment les auteurs de [FGM19] ont montré l'équivalence avec un modèle d'automates de Parikh déterministe two-way).

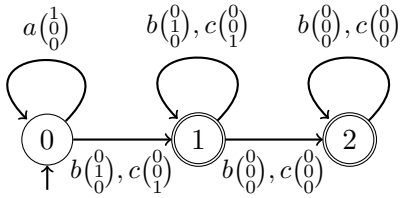
Cadhilac, toujours dans les discussions de sa thèse [Cad13], évoque les automates de Parikh faiblement non ambigus, en les appelant des automates *one-success (OneCA)*. Il montre l'inclusion stricte des langages de Parikh non ambigus dans les langages faiblement non ambigus :

Proposition 6.11 (Faible non-ambiguïté $\neq$ non-ambiguïté, [CFM13, Cad13]).
▶ Le langage $L = \{a^n w \mid w \in \{b,c\}^* \wedge |w_1 \dots w_n|_b < |w_1 \dots w_n|_c\}$ des mots de la forme $a^n w$ avec strictement moins de b que de c dans les n premières lettres de w, est un langage de Parikh faiblement non ambigu, mais n'est pas un langage de Parikh non ambigu. ◀

Ébauche. Supposons que $\mathcal{L}$ est reconnu par un automate de Parikh non ambigu. Alors, par stabilité des langages de Parikh non ambigus par intersection, et quotient à gauche par un langage de Parikh (en particulier par un langage régulier), $(\{a^*\}^{-1}L) \cap \{b,c\}^*$ est non ambigu. Mais ce dernier langage n'est pas un langage de Parikh non ambigu, par la Proposition 11 de [CFM13] (l'argument est que son complémentaire, le langage

de Dyck, n'est pas un langage de Parikh, alors que les langages des automates de Parikh non ambigus sont stables par complémentaire).

Par ailleurs, l'automate de Parikh faiblement non ambigu suivant reconnaît le langage L :



$$C = \{(n_1, n_2, n_3) \mid n_1 = n_2 + n_3 \text{ et } n_2 < n_3\}$$

■

Nous proposons un autre exemple, peut-être encore plus frappant, qui sépare les deux classes. Il montre que les automates de Parikh non ambigus ne sont pas capables de détecter le milieu d'un mot :

Proposition 6.12 .
▶ Le langage $L = \{vbw \mid v, w \in \{a, b\}^* \wedge |v| = |w|\}$ des mots de longueur impaire qui ont un b au milieu est un langage de Parikh faiblement non ambigu, mais n'est pas un langage de Parikh non ambigu. ◀

Démonstration. Supposons que L est reconnu par un automate de Parikh non ambigu $\mathcal{A} = (\{a, b\}, Q, q_I, F, C, \Delta)$.

Afin de ne pas mélanger les deux notions, j'appellerai *calcul final* un calcul de l'automate partant de l'état initial et finissant dans un état final, et je garderai l'appellation calcul acceptant pour les calculs finaux étiquetés par un vecteur appartenant au semilinéaire C.

La preuve repose sur la propriété suivante d'un automate de Parikh non ambigu : comme tout mot possède au plus un unique calcul final, tout calcul final étiqueté par un mot dans le langage est forcément acceptant, c'est-à-dire qu'il est étiqueté par un vecteur du semilinéaire C.

Soit n un multiple commun à toutes les longueurs des cycles élémentaires de l'automate, tel que $n - 1$ est plus grand que le nombre d'états de l'automate.

Cycle élémentaire = cycle qui ne possède pas de sous-cycle non trivial.

Soit m un entier strictement plus grand que le nombre de cycles élémentaires de l'automate.

Regardons le mot $w = (a^{n-1}b)^m a^{mn-1}$. Ce mot est constitué de m séquences de a de taille $n - 1$, séparées par des b, et finit par une longue séquence de a, de taille $mn - 1$. C'est la première moitié du mot qui nous intéresse, la longue séquence finale de a est juste là pour que le dernier b de la séquence soit au milieu du mot. Comme ce mot appartient au langage L, il existe un calcul acceptant π étiqueté par ce mot.

L'idée de la preuve est la suivante : nous identifions deux facteurs a^n de $w = \ldots a^n b \ldots a^n b \ldots$ tels que π passe par un même cycle commun en lisant chacun des deux facteurs. En itérant ce cycle dans le facteur de gauche, nous nous arrangeons pour obtenir un calcul d'un mot tel que le b qui clôt le premier facteur se trouve au milieu du mot. Par non-ambiguïté, ce calcul est étiqueté par un vecteur du semilinéaire. Mais en itérant le cycle une fois de plus dans le premier facteur, et une fois de moins dans le second facteur, qui se trouve à droite du milieu, nous obtenons un calcul

final étiqueté par un vecteur du semilinéaire, mais pour un mot qui n'est pas dans le langage car il a un a en son milieu. Contradiction.

Formellement, nous décomposons π selon les transitions effectuées dans l'automate, sous la forme $\pi = \pi_1 t_1 \dots \pi_m t_m \pi_f$, où $t_1, \dots, t_m$ sont des transitions de Δ qui lisent un b, et $\pi_1, \dots, \pi_m$ sont des calculs de l'automate étiquetés par a^{n-1}, et π_f par a^{mn-1}.

Comme $n-1$ est plus grand que le nombre d'états, chaque π_i avec $1 \leqslant i \leqslant m$ passe au moins deux fois par un même état, donc passe au moins une fois dans un cycle élémentaire, que l'ont note σ_i. Comme m est plus grand que le nombre total de cycles élémentaires, il existe deux indices $1 \leqslant i < j \leqslant m$ tels que $\sigma_i = \sigma_j$. Autrement dit, lors de la lecture des m premières séquences de a, π passe par un même cycle élémentaire dans deux séquences de a distinctes.

Notons σ ce cycle commun, et $\pi_i = \tau_1 \sigma \tau_2$, et $\pi_j = \tau_3 \sigma \tau_4$.

Modifions le calcul π en choisissant d'itérer k fois supplémentaires le cycle σ dans π_i. On obtient alors un calcul final (et non plus forcément acceptant, car le vecteur a changé est n'est peut-être plus dans C) :

$$\pi^k = \underbrace{\pi_1 \dots t_{i-1}}_{(a^{n-1}b)^{i-1}} \underbrace{\tau_1 \sigma^{k+1} \tau_2 t_i}_{a^{n-1+k|\sigma|}b} \underbrace{\pi_{i+1} \dots \pi_f}_{(a^{n-1}b)^{m-i}a^{mn-1}}$$

étiqueté par le mot $w_k = (a^{n-1}b)^{i-1} a^{n-1+k|\sigma|} b (a^{n-1}b)^{m-i} a^{mn-1}$.

Nous souhaitons ajouter suffisamment de a en itérant le cycle σ pour que le b qui clôt la i-ième séquence de a se retrouve au milieu du mot.

Il suffit de résoudre alors en k l'équation $n(i-1)+n-1+k|\sigma| = n(m-i)+mn-1$. On obtient alors $k|\sigma| = 2n(m-i)$. Comme $|\sigma|$ divise n, $k_0 = 2n(m-i)/|\sigma| > 0$ convient (car $i < m$).

Le calcul π^{k_0} est donc un calcul final étiqueté par un mot qui appartient au langage de $\mathcal{A}$: il est donc acceptant par non-ambiguïté. Par conséquent, il est étiqueté par un vecteur $\boldsymbol{v}$ appartenant au semilinéaire C.

Or le sous-calcul π_j est toujours présent dans π^{k_0}, et donc π^{k_0} passe à nouveau par le cycle σ en lisant la j-ième séquence de a de w_{k_0}. Ainsi :

$$\pi^{k_0} = \pi_1 \dots t_{i-1} \underbrace{\tau_1 \sigma^{k_0+1} \tau_2 t_i}_{a^{n-1+k_0|\sigma|}b} \pi_{i+1} \dots t_{j-1} \underbrace{\tau_3 \sigma \tau_4 t_j}_{a^{n-1}b} \pi_{j+1} \dots \pi_f$$

En retirant dans π^{k_0} le cycle σ de π_j, mais en ajoutant une itération supplémentaire du cycle σ dans π_i, on obtient un calcul final de la forme :

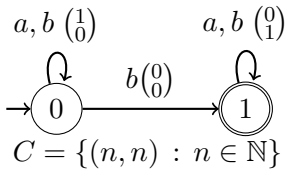
$$\pi' = \pi_1 \dots t_{i-1} \tau_1 \sigma^{k_0+2} \tau_2 t_i \pi_{i+1} \dots t_{j-1} \tau_3 \tau_4 t_j \pi_{j+1} \dots \pi_f$$

Remarquons que π' est étiqueté par le même vecteur $\boldsymbol{v} \in C$ que π^{k_0}, et il est final, donc c'est un calcul acceptant. Il étiquette donc un mot du langage L. Mais le mot qu'il étiquette est :

$$(a^{n-1}b)^{i-1} a^{n-1+(k_0+1)|\sigma|} b (a^{n-1}b)^{j-i-1} a^{n-1-|\sigma|} b (a^{n-1}b)^{m-j} a^{mn-1}$$

qui n'est pas dans L. En effet, le milieu du mot tombe dans la i-ième séquence de a, et n'est pas un b. Nous avons exhibé un calcul acceptant pour un mot qui n'est pas dans le langage. Contradiction.

Le langage L est reconnu par l'automate faiblement non ambigu de l'Exemple 6.5 :



■

Ces deux contre-exemples se comprennent bien si on considère le modèle des automates de Parikh déterministes two-way [FGM19], équivalent aux automates de Parikh non ambigus : l'idée est qu'un automate de Parikh déterministe two-way n'a pas accès à la valeur de ses vecteurs au cours du calcul. Il ne peut donc pas adapter son comportement à la valeur de la taille d'un préfixe (cas du langage de la Proposition 6.11) ni détecter le milieu du mot (cas du langage de la Proposition 6.12). À l'aide du langage précédent, nous proposons un langage un peu plus élémentaire que celui de la Proposition 6.11 mettant en évidence ce phénomène pour le préfixe :

Proposition 6.13 .
▶ Le langage $L = \{c^n w \mid w_n = b \text{ avec } n \in \mathbb{N},\ w \in \{a, b\}^*\}$, des mots de la forme $c^n w$ tels que la n-ième lettre de w est un b, est un langage de Parikh faiblement non ambigu, mais n'est pas un langage de Parikh non ambigu. ◀

Démonstration. Supposons que L est un langage de Parikh non ambigu. Le langage des mots de la forme $c^n w$ tel que $w \in \{a, b\}^*$ et $|w| = 2n + 1$ est non ambigu (il est même déterministe). Par conséquent, par stabilité par intersection, le langage $L' = \{c^n w \mid w = vbv',\ \text{avec } v, v' \in \{a, b\}^* \wedge |v| = |v'| = n\}$ serait non ambigu.

Par stabilité par quotient à gauche par c^*, puis par intersection par $\{a, b\}^*$, le langage $\{vbw \mid v, w \in \{a, b\}^* \wedge |v| = |w|\}$ des mots de longueur impaire qui ont un b au milieu serait non ambigu, contradiction avec la Proposition 6.12. ■

Le but de la prochaine section est de montrer que les automates de Parikh faiblement non ambigus s'insèrent naturellement dans le paysage des automates à compteurs non ambigus. Les automates de Parikh étant classiquement des modèles équivalents à des machines à compteurs appelées RBCM (Reversal Bounded Counter Machines), il est naturel de se demander si les deux modèles coïncident toujours sur leurs versions (faiblement) non ambiguës. Notons que ce n'est pas le cas pour les automates de Parikh non ambigus, qui sont incomparables avec les RBCM déterministes one-way [CFM13, Proposition 14].

Leur contre-exemple $\{a, b\}^* a^n b^n$ est par contre reconnaissable par une RBCM déterministe bidirectionnelle.

6.3 Modèles non ambigus équivalents aux automates de Parikh faiblement non ambigus

Dans cette section, nous introduisons différents modèles de machines non ambiguës, que nous montrons équivalents aux automates de Parikh faiblement non ambigus. Tous ces modèles, dans leurs versions non déterministes, sont connus pour être équivalents aux automates de Parikh non déterministes (hormis pour la dernière classe RCM, qui n'est pas reliée à des classes d'automates). Les preuves consistent donc en grande partie à vérifier que les constructions des preuves d'équivalences, dans le cas non déterministe, préservent la (faible) non-ambiguïté.

Dans la sous-section 6.3.2, nous généralisons les automates de Parikh faiblement non ambigus en étiquetant les transitions par des ensembles semilinéaires plutôt que des vecteurs, et nous montrons que les automates de ce modèle, que nous appelons automates de Parikh généralisés faiblement non ambigus, acceptent les mêmes langages que les automates de Parikh faiblement non ambigus standard. Ce modèle est implicite dans les preuves de [KR02, KR03]. Nous le rendons explicite pour deux raisons : premièrement ce modèle permet une présentation plus propre de la procédure d'élimination des ε-transitions de la sous-section 6.3.3 ; ensuite, ce modèle est assez générique et sera utilisé aussi pour les automates de Parikh à pile plus loin dans le chapitre, en section 6.4.

Dans la sous-section 6.3.3, nous généralisons les automates de Parikh en autorisant des ε-transitions, puis nous montrons comment les éliminer. La procédure d'élimination est essentiellement celle de [KR03], adaptée pour préserver la faible non-ambiguïté.

Pour les machines à compteurs on peut enfin dire non ambiguë sans préciser faiblement ; personne n'appelle calcul acceptant un calcul d'une RBCM qui ignore les valeurs de ses compteurs.

La sous-section 6.3.4 introduit les machines à compteur à inversion bornée (Reversal Bounded Counter Machines) non ambiguës, et montre l'équivalence avec les automates de Parikh faiblement non ambigus. Là encore, il s'agit principalement de s'assurer que les constructions de [Iba78] (réduction du nombre d'inversions des compteurs, passage d'une tête de lecture bidirectionnelle (two-way) à une unidirectionnelle (one-way)) et de [KR02, KR03] conservent la non-ambiguïté.

Enfin, dans la sous-section 6.3.5, nous introduisons la classe de langages RCM de [CM17]. Cette classe de langages, qui a été construite dans le but d'avoir une série génératrice holonome, n'était jusqu'alors pas reliée clairement à des classes d'automates, et les auteurs [CM17] conjecturaient certaines inclusions entre leur classe et des RBCM déterministes. Nous démontrons leur conjecture en montrant que la classe RCM reconnaît les mêmes langages que les automates de Parikh faiblement non ambigus.

6.3.1 Rappels et notations sur les calculs d'un automate de Parikh

Nous rappelons qu'un *calcul* d'un automate de Parikh $\mathcal{A} = (\Sigma, Q, q_I, F, C, \Delta)$ de dimension d est une séquence

$$\pi = p_0\,(a_0, \boldsymbol{v_0})\,p_1 \cdots p_{n-1}\,(a_{n-1}, \boldsymbol{v}_{n-1})\,p_n$$

avec $n \geqslant 0$, $p_i \in Q$, $a_i \in \Sigma$ et $\boldsymbol{v}_i \in \mathbb{N}^d$ telle que pour tout $i \in [0, n-1]$, $(p_i, (a_i, \boldsymbol{v_i}), p_{i+1})$ est une transition de $\mathcal{A}$. On notera ce calcul sous la forme plus lisible $p_0 \xrightarrow{a_0, \boldsymbol{v_0}} p_1 \cdots p_{n-1} \xrightarrow{a_{n-1}, \boldsymbol{v_{n-1}}} p_n$, et on dit qu'il commence en p_0, finit en p_n et est étiqueté par le couple $(a_0 \cdots a_{n-1}, \boldsymbol{v_0} + \cdots + \boldsymbol{v_{n-1}})$. Nous utiliserons une notation condensée $p_0 \xRightarrow[\pi]{w, \boldsymbol{v}} p_n$, avec $w = a_0 \cdots a_{n-1}$ et $\boldsymbol{v} = \boldsymbol{v_0} + \cdots + \boldsymbol{v_{n-1}}$.

Remarquons que $\pi = q$ est un calcul valide, qui commence en $q = p_0 = p_n$ (avec $n = 0$). Par convention un tel calcul de longueur nulle (il n'emprunte aucune transition) est étiqueté par le couple $(\varepsilon, \mathbf{0})$, où $\mathbf{0}$ est le vecteur nul de dimension d.

Si π_1 est un calcul commençant dans l'état p et finissant dans q, et π_2 est un calcul commençant en q et finissant en r, on note $\pi_1 \cdot \pi_2$ la concaténation des deux calculs, définie comme suit : si $\pi_1 = p_0 \xrightarrow{a_0, \boldsymbol{v_0}} p_1 \cdots p_{n-1} \xrightarrow{a_{n-1}, \boldsymbol{v_{n-1}}} p_n$ et $\pi_2 = q_0 \xrightarrow{b_0, \boldsymbol{w_0}} q_1 \cdots q_{m-1} \xrightarrow{b_{n-1}, \boldsymbol{v_{n-1}}} q_n$, alors :

$$\pi_1 \cdot \pi_2 = p_0 \xrightarrow{a_0, \boldsymbol{v_0}} p_1 \cdots p_{n-1} \xrightarrow{a_{n-1}, \boldsymbol{v_{n-1}}} p_n \xrightarrow{b_0, \boldsymbol{w_0}} q_1 \cdots q_{m-1} \xrightarrow{b_{n-1}, \boldsymbol{w_{n-1}}} q_n.$$

En particulier, si $p \xrightarrow[\pi_1]{u_1,\boldsymbol{v_1}} q$ et $q \xrightarrow[\pi_2]{u_2,\boldsymbol{v_2}} r$ alors $p \xrightarrow[\pi_1\cdot\pi_2]{u_1\cdot u_2,\boldsymbol{v_1}+\boldsymbol{v_2}} r$.

6.3.2 Automates de Parikh généralisés faiblement non ambigus

Nous introduisons une classe d'automates de Parikh dont les transitions sont étiquetées par des ensembles semilinéaires plutôt que des vecteurs de $\mathbb{N}^d$.

Définition 6.14 (Automate de Parikh généralisé).
▶ Un automate de Parikh généralisé de dimension d est un tuple $\mathcal{A} = (\Sigma, Q, q_I, F, C, \Delta)$ où Σ est l'alphabet d'entrée, Q est un ensemble fini d'états, $q_I \in Q$ est l'état initial, $F \subseteq Q$ est l'ensemble d'états finaux, C est un ensemble semilinéaire de $\mathbb{N}^d$ et Δ est l'ensemble de transitions, de la forme $p \xrightarrow{a,S} q$ avec p et $q \in Q$, $a \in \Sigma$ et S un *ensemble semilinéaire* de $\mathbb{N}^d$.

Un *calcul* $p_0 \xrightarrow{a_0,S_0} p_1 \cdots p_{n-1} \xrightarrow{a_{n-1},S_{n-1}} p_n$ est étiqueté par un couple (w, S) avec $w = a_0 \cdots a_{n-1}$, et $S = S_0 + \cdots + S_{n-1}$. Par convention, le calcul $\pi = q$ est étiqueté par $(\varepsilon, \{\mathbf{0}\})$.

Un calcul est *acceptant* s'il commence dans l'état initial (*i.e.* $p_0 = q_I$), finit dans un état final (*i.e.*, $p_n \in F$) et enfin $S \cap C \neq \emptyset$.

Un automate de Parikh généralisé est appelé faiblement non ambigu si tout mot possède étiquette au plus un calcul acceptant. ◀

Il n'existe pas d'automate de Parikh généralisé non ambigu dans la littérature, mais nous gardons le terme faiblement.

Un automate de Parikh standard peut être vu comme un automate de Parikh généralisé en remplaçant les transitions $(p, (a, \boldsymbol{v}), q)$ par $(p, (a, \{\boldsymbol{v}\}), q)$. Comme pour tous vecteurs $\boldsymbol{v_1}, \boldsymbol{v_2}$, $\{\boldsymbol{v_1}\} + \{\boldsymbol{v_2}\} = \{\boldsymbol{v_1} + \boldsymbol{v_2}\}$ et $\boldsymbol{v_1} \in S \Leftrightarrow \{\boldsymbol{v_1}\} \cap S \neq \emptyset$, cette transformation met en bijection les calculs acceptants des deux automates de Parikh considérés. Ainsi :

Proposition 6.15 .
▶ Soit $\mathcal{A}$ un automate de Parikh. Alors $\mathcal{L}(\mathcal{A})$ est reconnu par un automate de Parikh généralisé $\mathcal{A}'$. De plus si $\mathcal{A}$ est faiblement non ambigu, alors $\mathcal{A}'$ peut être choisi faiblement non ambigu. ◀

Le but de cette section est de montrer la réciproque, dans la Proposition 6.17, en convertissant un automate de Parikh généralisé en automate standard à vecteurs, tout en conservant la faible non-ambiguïté.

Pour cela, nous avons besoin d'un lemme technique sur les ensembles semilinéaires. Pour tout sous-ensemble L de $\mathbb{N}^d$, nous définissons L^r par $L^0 = \{\mathbf{0}\}$ et $L^{r+1} = L^r + L$ pour tout $r \geqslant 0$. Autrement dit, L^r contient tous les vecteurs de $\mathbb{N}^d$ qui s'expriment comme une somme de r vecteurs de L.

Lemme 6.16 ($\boldsymbol{v} \in S^r$ est semilinéaire, [KR02]).
▶ Soit S un ensemble semilinéaire dans $\mathbb{N}^d$. Alors la relation $\boldsymbol{v} \in S^r$ est semilinéaire, c'est-à-dire qu'il existe une formule de Presburger $\varphi(x, y_1, \ldots, y_d)$ telle que pour tout $r, v_1, \ldots, v_d \in \mathbb{N}$:

$$\varphi[r, v_1, \ldots, v_d] \text{ est vraie si et seulement si } (v_1, \ldots, v_d) \in S^r .$$

◀

Démonstration. Considérons d'abord le cas linéaire $S = \boldsymbol{c} + P^*$. Pour tout $\boldsymbol{v} \in \mathbb{N}^d$ et $r > 0$, alors $\boldsymbol{v} \in S^r$ si et seulement si $\boldsymbol{v} - r\boldsymbol{c} \in P^*$. Ceci est dû au fait que

$(P^*)^r = P^*$ dès que $r > 0$. Ainsi, la formule φ_S suivante définit bien la relation $\boldsymbol{v} \in S^r$:

$$\begin{aligned}\varphi_S(r, v_1, \ldots, v_d) \quad := \quad & r = 0 \to v_1 = \cdots = v_d = 0 \\ & \wedge\, r \neq 0 \to \varphi_P(v_1 - c_1 r, \ldots, v_d - c_d r)\end{aligned}$$

où $\varphi_P(x_1, \ldots, x_d)$ est une formule de Presburger définissant P. Notons que la multiplication de r par c_i est autorisée parce que $\boldsymbol{c}$ est une constante de la formule.

Considérons maintenant le cas général où S est semilinéaire, de la forme $S = \cup_{i=1}^{n} S_i$ où chaque S_i est un ensemble linéaire. Nous définissons

$$\begin{aligned}\varphi_S(r, \boldsymbol{v}) := \exists \ell_1, \ldots, \ell_n,\ \exists \boldsymbol{z_1}, \ldots, \boldsymbol{z_n}, \\ \ell_1 + \ldots + \ell_n = r \wedge \boldsymbol{v} = \boldsymbol{z_1} + \ldots + \boldsymbol{z_n} \\ \wedge \bigwedge_{i=1}^{n} \varphi_{S_i}(\ell_i, \boldsymbol{z_i})\end{aligned}$$

où chaque $\boldsymbol{z_i}$ designe un vecteur de d variables fraîches $(z_{i,1}, \ldots, z_{i,d})$. Montrons que cette formule définit bien la relation $\boldsymbol{v} \in S^r$.

La notation $\varphi_S[\boldsymbol{v}, r]$ indique qu'on évalue la formule en remplaçant les variables libres par des entiers de $\mathbb{N}$.

Soient $\boldsymbol{v} \in \mathbb{N}^d, r \in \mathbb{N}$ tels que $\boldsymbol{v} \in S^r$. Si $r = 0$ alors $\boldsymbol{v} = \boldsymbol{0}$ et $\varphi_S[\boldsymbol{v}, r]$ est vraie. Si $r > 0$, alors $\boldsymbol{v}$ est par définition la somme de r vecteurs de S : nous pouvons écrire $\boldsymbol{v} = \boldsymbol{v_1} + \ldots + \boldsymbol{v_r}$, avec $\boldsymbol{v_j} \in S$ pour tout $1 \leqslant j \leqslant r$.

Pour tout $1 \leqslant i \leqslant n$, nous notons D_i l'ensemble des vecteurs de $\boldsymbol{v_1}, \ldots, \boldsymbol{v_r}$ qui appartiennent à L_i, mais à aucun L_k pour $k < i$. Autrement dit $\boldsymbol{v_j}$ pour $j \in [r]$ appartient à D_i si et seulement si i est le plus petit entier tel que $\boldsymbol{v_j} \in L_i$. Les ensembles D_i sont naturellement disjoints. En réordonnant la somme $\boldsymbol{v} = \boldsymbol{v_1} + \ldots + \boldsymbol{v_r}$ de façon à regrouper les vecteurs selon leur ensemble D_i associé, nous pouvons réécrire $\boldsymbol{v} = \boldsymbol{z_1} + \ldots + \boldsymbol{z_n}$, où chaque $\boldsymbol{z_i} = \sum_{\boldsymbol{v} \in D_i} \boldsymbol{v}$ est la somme de tous les vecteurs de D_i (et est le vecteur nul si D_i est vide). Posons $\ell_i = |D_i|$. Comme $D_i \subseteq L_i$, $\boldsymbol{z_i} \in L_i^{\ell_i}$ et $\ell_1 + \ldots + \ell_n = r$. Donc $\varphi_S[r, \boldsymbol{v}]$ est vraie.

Montrons la réciproque. Soit $r \in \mathbb{N}$ et $\boldsymbol{v} \in \mathbb{N}^d$ tels que $\varphi_S[r, \boldsymbol{v}]$ soit vraie, c'est-à-dire qu'il existe des entiers $\ell_1, \ldots, \ell_n \in \mathbb{N}$ et des vecteurs $\boldsymbol{z_1}, \ldots, \boldsymbol{z_n} \in \mathbb{N}^d$, tels que $\ell_1 + \ldots + \ell_n = r$ et $\boldsymbol{v} = \boldsymbol{z_1} + \ldots + \boldsymbol{z_n}$ et $\boldsymbol{z_i} \in L_i^{\ell_i} \subseteq S^{\ell_i}$.

Si $r = 0$, alors chaque ℓ_i est nul, donc $\boldsymbol{z_i} = \boldsymbol{0}$ pour tout i. Ainsi $\boldsymbol{v} = \boldsymbol{0}$, et donc on a bien $\boldsymbol{v} \in S^r$.

Et si $r > 0$, alors certains ℓ_i sont non nuls. Chaque $\boldsymbol{z_i}$ s'écrit comme une somme de ℓ_i vecteurs de S, si bien que l'égalité $\boldsymbol{v} = \boldsymbol{z_1} + \ldots + \boldsymbol{z_n}$ se réécrit en une somme de r vecteurs de S. Donc $\boldsymbol{v} \in S^r$. ■

Nous pouvons maintenant proposer une traduction d'un automate généralisé vers un automate standard qui préserve le langage reconnu et la faible non-ambiguïté.

Proposition 6.17 (Équivalence avec les automates de Parikh).
▶ Soit $\mathcal{A}$ un automate de Parikh généralisé (faiblement non ambigu). Alors le langage $L(\mathcal{A})$ est reconnu par un automate de Parikh (faiblement non ambigu) $\mathcal{B}$ qui possède le même nombre d'états, et dont la dimension est égale au nombre d'ensembles semilinéaires distincts apparaissant dans les transitions de $\mathcal{A}$. ◀

Démonstration. Notons $\mathcal{A} = (Q, q_0, F, \Sigma, \Delta, S)$ l'automate de Parikh généralisé de l'énoncé. Soient $S_1, \ldots, S_r$ une énumération des différents ensembles semilinéaires apparaissant dans les transitions de $\mathcal{A}$.

Nous construisons un automate de Parikh $\mathcal{B} = (Q, q_0, F, \Sigma, \Delta', S')$ de dimension r acceptant $L(\mathcal{A})$. L'automate $\mathcal{B}$ a la même structure que $\mathcal{A}$ (*i.e.*, le même ensemble d'états, même état initial et mêmes états finaux), et ne diffère de $\mathcal{A}$ que par ses transitions et son ensemble semilinéaire d'acceptation.

À chaque transition $p \xrightarrow{a,S_i} q$ de Δ, nous associons une transition $p \xrightarrow{a,\boldsymbol{e_i}} q$ dans Δ', où $\boldsymbol{e_i}$ est le vecteur nul partout sauf pour sa i-ième coordonnée, qui vaut 1. La contrainte semilinéaire est $S' := \{(n_1, \ldots, n_r) \in \mathbb{N}^r \ : \ S \cap \sum_{i=1}^r S_i^{n_i} \neq \emptyset\}$. Elle est bien semilinéaire, car reconnue par la formule de Presburger suivante :

$$\varphi_{S'}(n_1, \ldots, n_r) = \exists \boldsymbol{v_1}, \ldots, \boldsymbol{v_r},\ \boldsymbol{v_1} + \ldots + \boldsymbol{v_r} \in S \ \wedge \ \bigwedge_{i=1}^{r} \boldsymbol{v_i} \in S_i^{n_i}$$

où $\boldsymbol{v_i} \in S_i^{n_i}$ est du sucre syntaxique pour la formule $\varphi_{S_i}(n_i, \boldsymbol{v_i})$ construite au Lemme 6.16.

De façon informelle, l'automate $\mathcal{B}$ simule les même calculs que $\mathcal{A}$, mais à la place d'additionner des ensembles semilinéaires pendant le calcul, il compte le nombre de fois que chaque semilinéaire apparaît au cours du calcul. L'ensemble S' se charge ensuite d'utiliser cette information pour en déduire que la somme des semilinéaires du calcul de l'automate $\mathcal{A}$ intersecte bien le semilinéaire S.

Nous notons τ la transformation qui traduit un calcul π de $\mathcal{A}$ en un calcul de $\mathcal{B}$ en remplaçant chaque transition de Δ par sa transition associée dans Δ'. L'application τ est une bijection des calculs de $\mathcal{A}$ dans les calculs de $\mathcal{B}$ qui préserve l'ensemble d'états et le mot étiquetant le calcul.

Pour montrer que $\mathcal{A}$ et $\mathcal{B}$ acceptent le même langage, nous allons prouver que pour chaque calcul π de $\mathcal{A}$, π est acceptant pour $\mathcal{A}$ si et seulement si $\tau(\pi)$ est acceptant pour $\mathcal{B}$. En particulier, comme τ est une bijection préservant les mots étiquetant les calculs, $L(\mathcal{A}) = L(\mathcal{B})$ et si $\mathcal{A}$ est faiblement non ambigu, alors $\mathcal{B}$ l'est aussi.

Soit π un calcul acceptant de $\mathcal{A}$ étiqueté par (w, S_π). Posons m_i le nombre d'occurrences du semilinéaire S_i dans les transitions de π, pour $1 \leqslant i \leqslant r$. Posons de plus $\boldsymbol{m}$ le vecteur $(m_i)_{i\in[1,r]}$. Par définition, l'ensemble semilinéaire S_π étiquetant le calcul π est $S_\pi = \sum_{i=1}^r S_i^{m_i}$, et comme π est acceptant, $S_\pi \cap S \neq \emptyset$.

Le calcul $\tau(\pi)$ dans $\mathcal{B}$ est lui étiqueté par le couple $(w, \boldsymbol{m})$, et emprunte exactement les mêmes états que π. Pour montrer que $\tau(\pi)$ est acceptant pour $\mathcal{B}$, nous devons montrer que $\boldsymbol{m} \in S'$. Par définition de S', la condition $\boldsymbol{m} \in S'$ est équivalente au fait que $S \cap \sum_{i=1}^r S_i^{m_i} = S \cap S_\pi \neq \emptyset$, qui est vérifiée car π est acceptant.

Réciproquement, soit π' un calcul acceptant de $\mathcal{B}$ étiqueté par $(w, \boldsymbol{m})$, avec $\boldsymbol{m} = (m_i)_i \in S'$. Alors $\pi = \tau^{-1}(\pi')$ est un calcul de $\mathcal{A}$ étiqueté par (w, S_π), commençant dans l'état initial, et finissant dans un état final. De plus m_i, par définition de τ, est le nombre d'occurrences du semilinéaire S_i dans les transitions de π. Donc $S_\pi = \sum_i S_i^{m_i}$. Pour montrer que π est acceptant, nous devons montrer que $S_\pi \cap S \neq \emptyset$, qui est vérifiée car $\boldsymbol{m} \in S'$. ■

Nous avons ainsi démontré l'équivalence des deux modèles :

Proposition 6.18 (Équivalence des modèles).
▶ Les automates de Parikh généralisés faiblement non ambigus et les automates de Parikh faiblement non ambigus reconnaissent les mêmes langages. ◀

Et nous pouvons déduire des constructions présentées dans les preuves le corollaire suivant :

Corollaire 6.19 .
▶ Tout langage de Parikh reconnu par un automate de Parikh faiblement non ambigu $\mathcal{A}$ est reconnu par un automate de Parikh faiblement non ambigu dont les transitions sont étiquetées par des vecteurs unitaires $\boldsymbol{e_i}$, de dimension $d \leqslant |\Delta_{\mathcal{A}}|$. ◀

Démonstration. Il s'agit d'un cas très particulier de la construction de la Proposition 6.17, dans le cas où les semilinéaires sont simplement des vecteurs. Il est tout à fait possible de le refaire plus simplement à la main, avec des vecteurs : si $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^m$ sont les vecteurs distincts apparaissant dans l'automate, on les remplace par les vecteurs canoniques $\boldsymbol{e_i}$ de dimension m. La coordonnée i compte donc le nombre de fois qu'une transition étiquetée par i a été empruntée. La nouvelle formule de Presburger, portant sur les m variables $y_1, \ldots, y_m$, est alors obtenue en remplaçant dans la formule de Presburger de l'automate de départ, chaque variable x_j pour $j \in [1, d]$ par l'expressions $\sum_{i=1}^{m} v_j^i y_i$, où v_j^i désigne la j-ième coordonnée du vecteur $\boldsymbol{v}^i$. ■

La proposition suivante a un intérêt théorique pour normaliser les transitions et le semilinéaire d'un automate de Parikh, et sera utile pour relier les séries génératrices des automates de Parikh avec les pavages de tuiles irrationnelles de [GP14]. Elle permettra aussi de démontrer facilement un des sens de l'équivalence avec les RBCM non ambiguës.

Proposition 6.20 (Normalisation d'un automate de Parikh).
▶ Tout langage de Parikh reconnu par un automate $\mathcal{A} = (\Sigma, Q, q_I, F, C, \Delta)$ faiblement non ambigu est reconnaissable par un automate de Parikh $\mathcal{B}$ qui vérifie :

- $\mathcal{B}$ est faiblement non ambigu, de dimension $2d$, avec $d \leqslant |\Delta|$;
- les transitions de $\mathcal{B}$ sont étiquetées par des vecteurs de la forme $(\boldsymbol{e_i}, \boldsymbol{e_j}) \in \mathbb{N}^{2d}$, où $\boldsymbol{e_i}$ désigne le vecteur unitaire de la base canonique de $\mathbb{N}^d$ qui possède un 1 en coordonnée i ;
- $C_{\mathcal{B}} = \{(n_1, \ldots, n_{2d}) \mid n_1 = n_{d+1} \wedge n_2 = n_{d+2} \wedge \ldots \wedge n_d = n_{2d}\}$.

◀

Démonstration. D'après le Corollaire 6.19, il existe un automate $\mathcal{A}' = (\Sigma, Q, q_I, F, C', \Delta')$ faiblement non ambigu qui reconnaît le même langage que $\mathcal{A}$, dont les transitions sont étiquetées par des vecteurs $\boldsymbol{e_i} \in \mathbb{N}^d$, où $d \leqslant |\Delta|$ est le nombre de vecteurs distincts qui apparaissent dans les transitions de $\mathcal{A}$.

Le semilinéaire C' est reconnaissable par un automate fini non ambigu dont les transitions sont étiquetées par des vecteurs de $\mathbb{N}^d$. On note $\mathcal{C} = (Q_C, q_{IC}, F_C, \Delta_C)$ cet automate. Quitte à ajouter des états, on peut supposer que chaque transition est étiquetée par un vecteur $\boldsymbol{e_j}$ (par exemple la transition $t = q \xrightarrow{(1,2,1)} q' \in \Delta$ est remplacée en créant de nouveaux états par une suite de transitions $q \xrightarrow{(1,0,0)} q_{t_1} \xrightarrow{(0,1,0)} q_{t_2} \xrightarrow{(0,1,0)} q_{t_3} \xrightarrow{(0,0,1)} q'$, avec de nouveaux états $q_{t_1}, q_{t_2}, q_{t_3}$ qui n'apparaissent que dans cette suite de nouvelles transitions).

On définit alors l'automate de Parikh produit de dimension d suivant :

$$\mathcal{B} = (\Sigma, Q \times Q_C, (q_I, q_{IC}), F \times F_C, C_{\mathcal{B}}, \Delta_{\mathcal{B}}) ,$$

où $C_{\mathcal{B}}$ est défini dans l'énoncé de la proposition, et

$$\Delta_{\mathcal{B}} = \{((p, p'), (a, (\boldsymbol{e_i}, \boldsymbol{e_j})), (q, q')) \mid (p, a, q) \in \Delta' \text{ et } (p', \boldsymbol{e_j}, q') \in \Delta_C\} .$$

L'automate de Parikh simule les calculs de $\mathcal{A}'$ et de $\mathcal{C}$ en parallèle. Comme chaque transition est étiquetée par un vecteur qui a une seule composante non nulle, égale à 1, tout calcul de $\mathcal{A}$ ou de $\mathcal{C}$ étiqueté par un vecteur $\boldsymbol{v}$ nécessite exactement $\|\boldsymbol{v}\|_1$ transitions, où $\|\boldsymbol{v}\|_1$ est la norme 1 de $\boldsymbol{v}$ (la somme de ses composantes).

Si π est un calcul acceptant de $\mathcal{A}'$, étiqueté par $(w, \boldsymbol{v})$, alors par définition d'un calcul acceptant, $\boldsymbol{v} \in C'$. Il existe donc un unique calcul acceptant dans l'automate $\mathcal{C}$ étiqueté par $\boldsymbol{v}$, et donc on peut associer à π un calcul acceptant dans $\mathcal{B}$ étiqueté par $(w, (\boldsymbol{v}, \boldsymbol{v}))$. Par non-ambiguïté de $\mathcal{C}$ et faible non-ambiguïté de $\mathcal{A}'$, il n'existe qu'un seul calcul acceptant dans $\mathcal{B}$ étiqueté par w.

Réciproquement, tout calcul acceptant dans $\mathcal{B}$ est étiqueté par un couple de la forme $(w, (\boldsymbol{v}, \boldsymbol{v}))$, avec par construction de $\mathcal{C}$, $\boldsymbol{v} \in C'$, et se projette donc en un calcul acceptant pour $\mathcal{A}'$ étiqueté par le même mot.

Donc $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{B})$, et $\mathcal{B}$ est faiblement non ambigu. ■

Remarque 6.21 .
▶ La proposition précédente montre qu'on peut "cacher" le semilinéaire dans l'automate sans changer l'expressivité du modèle. Le résultat a surtout un intérêt théorique, car sa construction en pratique augmente la dimension et le nombre d'états de l'automate. Par ailleurs, la preuve utilise de nombreuses constructions implicites non triviales (passage d'un semilinéaire sous forme d'automate à une formule de Presburger, et conversion inverse, construction d'un automate non ambigu reconnaissant un semilinéaire…) ◀

Au prix d'un semilinéaire un peu moins simple.

En appliquant la construction du Corollaire 6.19 à la normalisation précédente, nous obtenons un automate de Parikh avec uniquement des vecteurs canoniques :

Proposition 6.22 (Normalisation bis d'un automate de Parikh).
▶ Tout langage de Parikh faiblement non ambigu est reconnaissable par un automate de Parikh $\mathcal{B}$ qui vérifie :

- $\mathcal{B}$ est faiblement non ambigu ;
- les transitions de $\mathcal{B}$ sont étiquetées par des couples de la forme $(a, \boldsymbol{e_i})$ avec $\boldsymbol{e_i} \in \mathbb{N}^r$; de plus le vecteur $\boldsymbol{e_i} \in \mathbb{N}^r$ détermine la lettre a : si $(a, \boldsymbol{e_i})$ et $(b, \boldsymbol{e_i})$ étiquettent deux transitions de $\mathcal{B}$, alors $a = b$;
- $C_{\mathcal{B}}$ est de la forme :

$$C_{\mathcal{B}} = \{(n_1, \dots, n_r) \mid \bigwedge_{i=1}^{r} (\textstyle\sum_{j=1}^{r} a_{i,j} n_j = 0), \text{ avec } a_{i,j} \in \{-1, 0, 1\}\}\,.$$

◀

Démonstration. On suppose sans perte de généralité que le langage de l'énoncé est reconnu par un automate de Parikh faiblement non ambigu normalisé par la Proposition 6.20.

Soit f une bijection de $\Sigma \times [1, d]^2$ dans $[1, d^2|\Sigma|]$. On note pour simplifier $\boldsymbol{e_{a,i,j}} := \boldsymbol{e_{f(a,i,j)}}$, pour $a \in |\Sigma|$ et $1 \leqslant i, j \leqslant d$.

On utilise alors la construction du Corollaire 6.19, adaptée légèrement pour encoder aussi les lettres. L'automate $\mathcal{B}$ est obtenu à partir de $\mathcal{A}$ en remplaçant chaque étiquette de transition de la forme $(a, (\boldsymbol{e_i}, \boldsymbol{e_j}))$ par l'étiquette $(a, \boldsymbol{e_{a,i,j}})$. En notant pour simplifier $n_{a,i,j}$ la valeur de la $f(a, i, j)$-ième coordonnée de $\mathbb{N}^r$, le semilinéaire $\{\boldsymbol{n} \in$

$\mathbb{N}^{2d} \mid \bigwedge_{i=1}^{d} n_i = n_{d+i}\}$ est alors remplacé dans la construction du Corollaire 6.19 par le semilinéaire

$$\{\boldsymbol{n} \in \mathbb{N}^r \mid \bigwedge_{i=1}^{d} \sum_{j=1}^{d} \sum_{a\in\Sigma} n_{a,i,j} = \sum_{j=1}^{d} \sum_{a\in\Sigma} n_{a,j,i}\}\,.$$

■

6.3.3 Automates de Parikh faiblement non ambigus avec ε-transitions

Un automate de Parikh avec ε-transitions est défini comme un automate de Parikh standard, à la seule différence que les transitions de la forme $(p, (\varepsilon, \boldsymbol{v}), q)$ sont autorisées, où ε est le mot vide. Les notions de calculs et de langage accepté sont les mêmes que pour les automates de Parikh sans ε.

Dans les versions non déterministes, il est connu que les ε-transitions n'augmentent pas l'expressivité du modèle [KR03]. Cette partie consiste à s'assurer avec attention que la procédure d'élimination des ε-transitions préserve aussi la faible non-ambiguïté :

Proposition 6.23 (Élimination des ε-transitions).
▶ Soit $\mathcal{A}$ un automate de Parikh faiblement non ambigu, avec ε-transitions. Alors on peut construire un automate de Parikh faiblement non ambigu $\mathcal{B}$ équivalent, sans ε-transitions. De plus $\mathcal{B}$ peut être choisi avec $n+1$ états et de dimension $2(n+1)^2 t$, si $\mathcal{A}$ possède n états et t transitions. ◀

Démonstration. Soit $\mathcal{A} = (\Sigma, Q, q_I, F, C, \Delta)$ un automate de Parikh faiblement non ambigu, avec ε-transitions. Sans perte de généralité, nous supposons que l'état initial q_I n'a pas de transition entrante. Grâce à la Proposition 6.17, il suffit de construire un automate de Parikh généralisé faiblement non ambigu $\mathcal{B}$ sans ε-transitions qui accepte $L(\mathcal{A})$.

Quitte à dupliquer q_I, d'où le $n+1$ dans l'énoncé, et le $2t$.

L'idée principale derrière $\mathcal{B}$ est de court-circuiter les chemins d'ε-transitions, en les assemblant avec une transition qui lit une lettre. Une transition de $\mathcal{B}$ simule une transition lisant une lettre, suivie d'un chemin d'ε-transitions (sauf pour l'état initial, où il faut simuler aussi un chemin d'ε-transitions avant la première lecture d'une lettre). Pour simuler ces chemins d'ε-transitions en une seule étape, nous souhaitons ajouter les vecteurs étiquetant ces chemins aux transitions lisant des lettres. Le problème est qu'il y a potentiellement une infinité de vecteurs étiquetant un chemin d'ε-transitions entre deux états (imaginez par exemple une boucle sur un état). Comme ces ensembles sont semilinéaires, il est possible de résoudre ce problème en utilisant le formalisme des automates de Parikh généralisés.

Pour tout couple d'états (p, q) dans $\mathcal{A}$, nous considérons l'ensemble $C^{\varepsilon}_{p,q}$ des vecteurs $\boldsymbol{v}$ de $\mathbb{N}^d$ qui peuvent étiqueter un calcul π de la forme $p \overset{\varepsilon,\boldsymbol{v}}{\Longrightarrow} q$. Autrement dit $C^{\varepsilon}_{p,q}$ contient tous les vecteurs que l'on peut obtenir en joignant p à q en n'empruntant que des ε-transitions.

Remarquons que $C^{\varepsilon}_{p,q}$ est semilinéaire. En effet, il est reconnu par l'automate $(Q, \{p\}, \{q\}, \Delta_\varepsilon)$ sur $(\mathbb{N}^d, +)$, obtenu à partir de $\mathcal{A}$ en définissant comme état initial p, comme état final q, et en ne gardant que les ε-transitions de Δ (autrement dit $(p, \boldsymbol{v}, q) \in \Delta_\varepsilon$ si et seulement si $(p, (\varepsilon, \boldsymbol{v}), q) \in \Delta$).

L'automate $\mathcal{B}$ est alors défini comme ci-après : il a les mêmes états, le même état initial, et les mêmes états finaux que $\mathcal{A}$.

De plus, pour toute transition $(p,(a,\boldsymbol{u}),q) \in \Delta$, telle que $a \neq \varepsilon$ et $p \neq q_I$, et tout couple (q,r) d'états de $\mathcal{A}$, nous ajoutons à Δ' la transition $(p,(a,(\{\boldsymbol{u}\}+C^{\varepsilon}_{q,r}),r)$, et la transition $(q_I,(a,(\{\boldsymbol{u}\}+C^{\varepsilon}_{q_I,p}+C^{\varepsilon}_{q,r}),r)$. Comme q_I n'a pas de transition entrante, il ne peut y avoir qu'une seule transition de la forme $(q_I,(a,(\{\boldsymbol{u}\}+C^{\varepsilon}_{q_I,p}+C^{\varepsilon}_{q,r}),r)$ dans un calcul de $\mathcal{A}'$, à son tout début.

Montrons que $L(\mathcal{A}) \subseteq L(\mathcal{B})$. Soit $w = w_1 \dots w_r \in L(\mathcal{A})$, et $q_I \xRightarrow[\pi]{(w,\boldsymbol{u})} q_f$ son calcul acceptant dans $\mathcal{A}$, avec $\boldsymbol{u} \in S$. Nous pouvons décomposer le calcul sous la forme

$$\pi = \pi^0_\varepsilon \pi^1_{w_1} \dots \pi^{r-1}_\varepsilon \pi^r_{w_r} \pi^r_\varepsilon \,,$$

où chaque $\pi^i_{w_i}$ est une transition qui lit la lettre w_i, et chaque π^i_ε est un chemin (potentiellement vide) d'ε-transitions. Au tout début du calcul, nous avons un sous-calcul de la forme

$$q_I \xRightarrow[\pi^0_\varepsilon]{(\varepsilon,\boldsymbol{v_0})} p \xrightarrow[\pi^1_{w_1}]{(w_1,\boldsymbol{u_1})} q \xRightarrow[\pi^1_\varepsilon]{(\varepsilon,\boldsymbol{v_1})} r \,,$$

avec par définition $\boldsymbol{v_0} \in C^{\varepsilon}_{q_I,p}$ et $\boldsymbol{v_1} \in C^{\varepsilon}_{q,r}$. La transition $(q_I,(w_1,(\{\boldsymbol{u}\}+C^{\varepsilon}_{q_I,p}+C^{\varepsilon}_{q,r}),r)$ est dans Δ' par construction. Pour simplifier l'écriture, nous écrivons $L_0 = C^{\varepsilon}_{q_I,p}$ et $L_1 = C^{\varepsilon}_{q,r}$.

Pour tout $2 \leqslant j \leqslant r$, $\pi^j_{w_j}$ et π^j_ε sont de la forme $\pi^j_{w_j} = p \xrightarrow{w_j,\boldsymbol{u_j}} q$ et $\pi^j_\varepsilon = q \xRightarrow{(\varepsilon,\boldsymbol{v_j})} r$, où p, q, et r sont trois états et $\boldsymbol{v_j} \in C^{\varepsilon}_{q,r}$, de telle sorte que dans Δ' il y ait la transition correspondante $(p,(w_j,(\{\boldsymbol{u_j}\}+C^{\varepsilon}_{q,r}),r)$. Pour simplifier l'écriture, nous notons $L_j = C^{\varepsilon}_{p,q}$. Remarquons que cette décomposition est non ambiguë, car il n'y a qu'une seule façon de décomposer π sous cette forme.

Le calcul π se transpose donc en un calcul de $\mathcal{B}$ de la forme $q_I \xRightarrow[\pi_\mathcal{B}]{(w,L)} q_f$ avec $L = L_0 + \ldots + L_r + \{\boldsymbol{u_1}\} + \ldots + \{\boldsymbol{u_r}\}$. Il est acceptant si $L \cap S \neq \emptyset$, ce qui est immédiat puisque $\boldsymbol{v_0} + \ldots + \boldsymbol{v_r} + \boldsymbol{u_1} + \ldots + \boldsymbol{u_r} = \boldsymbol{u} \in S$, et $\boldsymbol{v_i} \in L_i$ pour tout $0 \leqslant i \leqslant r$.

Montrons maintenant l'inclusion inverse $L(\mathcal{B}) \subseteq L(\mathcal{A})$. Soit $w = w_1 \dots w_r \in L(\mathcal{B})$, et $q_I \xRightarrow[\pi']{(w,L)} q_f$ son calcul acceptant dans $\mathcal{B}$, avec $S \cap L \neq \emptyset$. Nous pouvons décomposer $\pi' = \pi'_1 \pi'_2 \dots \pi'_r$. La première transition π'_1 est de la forme $q_I \xrightarrow{w_1,\{\boldsymbol{u_1}\}+L_0+L_1} r$, avec p,q,r trois états, $L_0 = C^{\varepsilon}_{q_I,p}$ et $L_1 = C^{\varepsilon}_{q,r}$. De même, pour tout $j \leqslant 2$, chaque π'_j est de la forme $p \xrightarrow{w_j,\{\boldsymbol{u_j}\}+L_j} r$ et $L_j = C^{\varepsilon}_{q,r}$.

Comme $S \cap L \neq \emptyset$, nous fixons un vecteur $\boldsymbol{u} \in S \cap L$. Comme $L = L_0 + \ldots + L_r + \{\boldsymbol{u_1}\} + \ldots + \{\boldsymbol{u_r}\}$, il existe des vecteurs $\boldsymbol{v_i} \in L_i$ pour tout $0 \leqslant i \leqslant r$ tels que $\boldsymbol{u} = \boldsymbol{u_1} + \ldots + \boldsymbol{u_r} + \boldsymbol{v_0} + \ldots + \boldsymbol{v_r}$.

Or, $\boldsymbol{v_0} \in L_0$ et $\boldsymbol{v_1} \in L_1$, donc nous pouvons construire à partir de π'_1 un début de calcul dans $\mathcal{A}$ de la forme $\pi_1 = q_I \xRightarrow[\pi^0_\varepsilon]{(\varepsilon,\boldsymbol{v_0})} p \xrightarrow{(w_1,\boldsymbol{u_1})} q \xRightarrow[\pi^1_\varepsilon]{(\varepsilon,\boldsymbol{v_1})} r$.

Pour les mêmes raisons, pour tous $2 \leqslant j \leqslant r$, $\boldsymbol{v_j} \in L_j$, donc nous pouvons reconstruire à partir de π'_j un calcul de $\mathcal{A}$ de la forme $\pi_j = p \xrightarrow{w_j,\boldsymbol{u_j}} q \xRightarrow[\pi^\varepsilon_j]{(\varepsilon,\boldsymbol{v_j})} r$.

Le calcul $\pi_1 \dots \pi_r = q_I \xRightarrow[\pi]{w,\boldsymbol{u}} q_f$ est un calcul acceptant dans $\mathcal{A}$, puisque $\boldsymbol{u} \in S$. Donc $w \in L(\mathcal{A})$.

Nous remarquons que la construction que nous utilisons dans cette preuve transforme les calculs π de $\mathcal{A}$ en calculs π' de $\mathcal{B}$, obtenus en contractant les chemins d'ε-transitions de π. Cette contraction est bien définie, et il existe une seule façon de contracter un calcul de $\mathcal{A}$ en un calcul de $\mathcal{B}$. Réciproquement, nous avons montré que tout calcul de $\mathcal{B}$ est la contraction d'au moins un calcul de $\mathcal{A}$.

Supposons que $w \in L(\mathcal{B})$ est accepté par deux calculs différents π'_1, π'_2 de $\mathcal{B}$. Alors nous pouvons créer deux calculs acceptant π_1, π_2 de $\mathcal{A}$ étiquetés par w, tels que π'_1 est la contraction de π_1, et π'_2 est la contraction de π_2. Par faible non-ambiguïté de $\mathcal{A}$, $\pi_1 = \pi_2$. Comme il n'y a qu'une seule façon de contracter un calcul de $\mathcal{A}$ en un calcul de $\mathcal{B}$, $\pi'_1 = \pi'_2$. Donc $\mathcal{B}$ est aussi faiblement non ambigu. ■

L'élimination des ε-transitions a pour première conséquence une preuve simple de la stabilité des langages de Parikh faiblement non ambigus par quotient à gauche par un mot, annoncée à la Proposition 6.7.

Proposition 6.24 (Clôture par quotient à gauche par un mot).
▶ Soit $\mathcal{L}$ un langage de Parikh faiblement non ambigu sur un alphabet Σ. Soit w un mot dans Σ^*. Alors le langage quotient $w^{-1}\mathcal{L} := \{v \in \Sigma^* \mid wv \in \mathcal{L}\}$ est un langage de Parikh faiblement non ambigu. ◀

Preuve rapide. Soit $\mathcal{A} = (\Sigma, Q, q_I, F, C, \Delta)$ un automate de Parikh standard reconnaissant L.

Soit Π_w l'ensemble des calculs de $\mathcal{A}$ qui lisent le mot w en partant de l'état initial. Comme $\mathcal{A}$ ne contient pas d'ε-transitions, Π_w est fini (les calculs sont de longueur $|w|$).

On définit l'automate $\mathcal{A}' = (\Sigma, Q, q'_I, F, C, \Delta')$, avec q'_I qui est un nouvel état, et $\Delta' = \Delta \cup \{(q'_I, (\varepsilon, \boldsymbol{v}), q) \mid \pi = q_I \xrightarrow{(w,\boldsymbol{v})} q \in \Pi_w\}$.

L'automate $\mathcal{A}'$ simule un calcul de $\mathcal{A}$ qui lit le mot w depuis q_I, sans lire le mot w : s'il existe un calcul de $\mathcal{A}$ de q_I à un état q étiqueté par $(w, \boldsymbol{v})$, alors $\mathcal{B}$ va directement dans q depuis q'_I par une ε-transition étiquetée par le vecteur $\boldsymbol{v}$.

Il est facile de voir que les calculs acceptants de $\mathcal{A}'$ sur un mot v sont en bijection avec les calculs acceptants de $\mathcal{A}$ qui lisent le mot wv. Ainsi $\mathcal{L}(\mathcal{A}') = w^{-1}\mathcal{L}$ et $\mathcal{A}'$ est faiblement non ambigu comme $\mathcal{A}$. ■

6.3.4 RBCM non ambiguës

Une *machine à k compteurs* [Iba78] est de façon informelle une machine de Turing avec k compteurs positifs et un ruban en lecture seule, qui contient $¢w\$$, où w est le mot à lire en entrée, et $¢$ et $\$$ sont des délimiteurs de début et de fin de ruban. La machine, en lisant une lettre a sur le ruban de lecture, dans un état q, peut vérifier lesquels de ses compteurs valent 0, incrémenter ses compteurs, décrémenter ses compteurs qui ne sont pas nuls, changer d'état, et bouger sa tête de lecture d'un pas vers la gauche, la droite, ou rester sur place. La machine n'a pas accès à la valeur exacte de ses compteurs, mais sait uniquement s'ils sont nuls ou non. De façon plus formelle :

Définition 6.25 (Machine à compteurs, [Iba78]).
▶ Une *machine à k compteurs* $\mathcal{M}$ est un tuple $(\Sigma, Q, q_I, F, \Delta, ¢, \$)$, où Σ l'alphabet d'entrée, Q désigne l'ensemble des états de $\mathcal{M}$, q_I l'état initial, $¢, \$ \notin \Sigma$ les symboles

de début et de fin de ruban, F l'ensemble des états finaux, et Δ est l'ensemble des transitions.

Plus précisément, les transitions de Δ sont de la forme $(q, a, \boldsymbol{v}) \to (q', d, \boldsymbol{v'})$, avec q qui désigne l'état courant de la machine, $a \in \Sigma \cup \{¢, \$\}$ le symbole lu, et $\boldsymbol{v} \in \{0, 1\}^k$ l'états des k compteurs (1 pour non nul, 0 pour nul), et de l'autre côté q' désigne l'état dans lequel se retrouve la machine après avoir emprunté la transition, $d \in \{-1, 0, +1\}$ le déplacement qu'effectue la tête de lecture, et enfin $\boldsymbol{v'} \in \{-1, 0, +1\}^k$ désigne les incréments ou décréments à effectuer sur les compteurs.

On impose que Δ fasse en sorte qu'on ne puisse pas déplacer la tête de lecture à gauche du symbole ¢ ni à droite de \$, et qu'on ne puisse pas décrémenter un compteur nul.

Une *configuration* d'une machine à compteur est un tuple $(q, ¢w\$, i, \boldsymbol{v})$, où q est l'état courant de la machine, i est la position de la tête de lecture sur le ruban qui contient $¢w\$$, et $\boldsymbol{v} \in \mathbb{N}^k$ est la valeur des compteurs (on rappelle que la machine n'a pas accès à la valeur exacte de $\boldsymbol{v}$, mais juste à ses coordonnées nulles).

La *configuration initiale* de la machine sur un mot w est la configuration $(q_I, ¢w\$, 0, \mathbf{0})$. Une *configuration finale* de la machine pour un mot w est une configuration de la forme $(q_f, ¢w\$, i, \boldsymbol{v})$ où $q_f \in F$.

Un pas de la machine est un tuple de la forme (c, t, c') où

- $c = (q, ¢w\$, i, \boldsymbol{v})$ est la configuration de départ
- t est la transition empruntée, de la forme $t = (q, a, \boldsymbol{u}) \to (q', d, \boldsymbol{v'})$, et est compatible avec c, c'est-à-dire qu'en position i de $¢w\$$ se situe bien la lettre a, et pour $i \in [k]$, $u_i = 0$ si et seulement si $v_i = 0$.
- c' est la configuration $c' = (q', ¢w\$, i + d, \boldsymbol{v} + \boldsymbol{u})$

Un *calcul* de la machine est une suite de pas de la machine telle que la configuration d'arrivée d'un pas est la configuration d'entrée du pas suivant. Un calcul sur un mot w est *acceptant* s'il commence dans la configuration initiale sur le mot w et termine dans une configuration finale. On dit alors que w est accepté par $\mathcal{M}$ et on note $\mathcal{L}(\mathcal{M})$ le langage des mots acceptés par $\mathcal{M}$.

Une machine à compteurs $\mathcal{M}$ est *déterministe* si Δ est une fonction partielle, c'est-à-dire que tout tuple $(q, a, \boldsymbol{v})$ est associé à au plus un tuple $(q', d, \boldsymbol{v'})$ par Δ.

Une machine à compteurs $\mathcal{M}$ est appelée *non ambiguë* si tout mot qu'elle accepte possède exactement un calcul acceptant. ◀

Nous considérons une restriction des machines à compteurs, qui limite leur pouvoir de calcul :

Définition 6.26 ((m, n)-reversal, [Iba78]).
▶ Une machine à k compteurs est dite à (m, n)-inversion bornée si sa tête de lecture ne change de direction qu'au plus m fois (sans compter le fait de rester sur place), et si chaque compteur n'alterne entre l'incrémentation et la décrémentation qu'au plus n fois. On note $\mathrm{NFCM}(k, m, n)$ l'ensemble des machines à k compteurs à (m, n)-inversion bornée.

Une machine $\mathcal{M}$ est une machine à compteur à inversion bornée, notée RBCM (Reversal Bounded Counter Machine) s'il existe trois entiers k, m et n tels que $\mathcal{M}$ est dans $\mathrm{NFCM}(k, m, n)$.

On note $\mathrm{detRBCM}$ (resp. 2-$\mathrm{detRBCM}$) la classe des langages reconnaissables par une RBCM déterministe à tête de lecture unidirectionnelle ($m = 0$) (resp. bidirectionnelle ($m > 0$)).

On note uRBCM la classe des langages reconnaissables par une RBCM non ambiguë. ◀

Les différentes classes de RBCM dépendent de trois entiers (k, m, n). Ibarra montre qu'on ne change par leur expressivité en imposant $m = 0$ et $n = 1$:

La transformation augmente bien sûr le nombre de compteurs et d'états. La condition $n = 1$ sur les compteurs est prouvée dans [BB74].

Proposition 6.27 (Normalisation des RBCM, [Iba78]).
▶ Soit L un langage reconnu par une RBCM (non ambiguë). Alors L est reconnue par une RBCM (non ambiguë) qui lit l'entrée de gauche à droite une seule fois, et dont chaque compteur ne peut plus s'incrémenter après avoir été décrémenté. ◀

Les constructions d'Ibarra pour prouver cette proposition n'étudient pas la non-ambiguïté. Ibarra remarque que la plupart des constructions préservent le caractère déterministe des machines, sauf la simulation d'une machine à tête de lecture bidirectionnelle par une machine unidirectionnelle, qui a besoin du non déterminisme. Toutes ces constructions s'adaptent très bien pour préserver la non-ambiguïté ; pour le cas de la simulation d'une machine "two-way" par une machine "one-way", il faut juste s'assurer que le découpage des calculs au niveau des extrémités ¢ et $ du ruban est non ambigu.

Nous rappelons que $\mathrm{detRBCM} \subsetneq \text{2-detRBCM}$.

La proposition suivante établit le lien entre les RBCM et les automates de Parikh :

Proposition 6.28 (Lien entre RBCM et automates de Parikh, [KR02, KR03, CFM12a]).
▶ Les RBCM reconnaissent les mêmes langages que les automates de Parikh. ◀

La preuve de [CFM12a] est plus facile à comprendre, et corrige une erreur dans la preuve de [KR02].

Le but de cette section est de montrer l'équivalence analogue sur les versions non ambiguës :

Théorème 6.29 (Équivalence des RBCM et des automates de Parikh faiblement non ambigus).
▶ Les RBCM non ambiguës reconnaissent les mêmes langages que les automates de Parikh faiblement non ambigus. ◀

Démonstration. Ce sont les mêmes idées que pour le cas non déterministe, il faut juste faire attention à préserver la non-ambiguïté / faible non-ambiguïté.

De PA faiblement non ambigu vers RBCM non ambiguë. Soit $\mathcal{A}$ un automate de Parikh faiblement non ambigu, normalisé comme dans la Proposition 6.20, de dimension $2d$. On lui associe une RBCM $\mathcal{M}$ qui possède $2d$ compteurs, a le même ensemble d'états que $\mathcal{A}$, mais avec un état initial et un état final supplémentaires distincts de ceux de $\mathcal{A}$. Lorsque la RBCM lit le symbole ¢ depuis son état initial, elle bouge sa tête de lecture vers la droite et va dans l'état initial de $\mathcal{A}$. La machine ensuite simule un calcul de $\mathcal{A}$, chaque transition de l'automate de Parikh étiquetée par $(\boldsymbol{e_i}, \boldsymbol{e_j})$ étant traduite en une transition lisant la même lettre, allant dans le même état, et qui incrémente le i-ième compteur et le j-ième compteur de $\mathcal{M}$. Lorsque le mot est lu en entier et que la RBCM arrive sur le symbole de fin $, dans un état final pour $\mathcal{A}$, elle reste sur place sur le symbole $ et vérifie pour chaque $i \in [d]$ que le compteur i est égal au compteur $d + i$ (pour cela elle les décrémente simultanément et vérifie qu'ils deviennent nuls en même temps). Si c'est le cas elle va dans son état final et accepte le mot. Les calculs acceptants de $\mathcal{A}$ et de $\mathcal{M}$ sont en bijection, et étiquetés par les mêmes mots, car le comportement de $\mathcal{M}$ aux extrémités du ruban ¢ et $ est déterministe. Donc $\mathcal{M}$ reconnaît le même langage que $\mathcal{A}$ et est non ambiguë.

Pour rester non ambigu il faut faire la vérification des compteurs dans l'ordre : soit en ajoutant des états successifs, soit en empêchant la décrémentation des compteurs j et $d + j$ tant qu'il existe un compteur i ou $d + i$ non nul, avec $i < j$.

De RBCM non ambiguë vers PA faiblement non ambigu. Nous adaptons la preuve de [CFM12a] pour montrer qu'une RBCM non ambiguë peut être simulée par un automate de Parikh faiblement non ambigu, avec des ε-transitions. Soit $\mathcal{M}$ une RBCM normalisée non ambiguë (cf Proposition 6.27). Pour simplifier la preuve, nous considérons uniquement le cas $k = 1$, à un seul compteur, mais la preuve s'adapte facilement avec plusieurs compteurs. La machine $\mathcal{M}$ est donc non ambiguë, unidirectionnelle, à un seul compteur, qui ne peut plus être incrémenté après avoir été décrémenté.

L'automate de Parikh $\mathcal{A}$ associé est de dimension 3, et ses états sont de la forme (q, τ), avec q un état de $\mathcal{M}$ et $\tau \in \{0, 1, 2, 3\}$. Intuitivement, $\mathcal{A}$ simule $\mathcal{M}$ en utilisant la première coordonnée de ses vecteurs pour stocker le nombre d'incréments du compteur de $\mathcal{M}$, et en utilisant la seconde coordonnée pour stocker le nombre de décréments. En particulier, la valeur du compteur d'un calcul de $\mathcal{M}$ s'obtient en soustrayant la seconde coordonnée à la première.

La principale difficulté pour simuler $\mathcal{M}$ sur un automate de Parikh vient du fait qu'un automate de Parikh n'est pas capable de détecter en cours de calcul quand ces deux coordonnées sont égales, et donc ne peut tester en même temps que $\mathcal{M}$ la nullité du compteur : l'automate de Parikh doit deviner lorsque le compteur est nul, et vérifier à la fin du calcul, par la contrainte semilinéaire, que son choix était correct.

L'automate de Parikh stocke dans la composante τ de ses états (et aussi la troisième composante des vecteurs pour récupérer l'information dans le semilinéaire) l'état du compteur de la machine au cours de la simulation. Plus précisément :

- $\tau = 0$ si la machine $\mathcal{M}$ n'a jamais incrémenté son compteur. Dans ce cas son compteur est forcément nul ;
- $\tau = 1$ si la machine a déjà incrémenté son compteur mais ne l'a jamais décrémenté. Dans de cas, le compteur est forcément non nul ;
- $\tau = 2$ si la machine a déjà décrémenté son compteur mais le compteur est non nul ;
- $\tau = 3$ si la machine a déjà décrémenté son compteur, et son compteur est nul.

L'automate $\mathcal{A}$ reporte dans la troisième coordonnée de ses vecteurs la valeur de τ avant d'accepter.

Lorsque $\tau = 0$ ou $\tau = 3$, $\mathcal{A}$ est autorisé à simuler uniquement des transitions de $\mathcal{M}$ qui imposent que le compteur soit nul, et lorsque $\tau = 1$ ou $\tau = 2$, uniquement des transitions qui demandent que le compteur soit non nul. En particulier, les transitions qui simulent une décrémentation ne peuvent être prises que depuis les états $\tau = 1$ ou $\tau = 2$.

Toute RBCM vérifie que son compteur est non nul avant de le décrémenter.

L'automate $\mathcal{A}$ commence dans l'état $\tau = 0$ et passe de façon déterministe dans l'état $\tau = 1$ à la première incrémentation du compteur. Toute transition de l'automate qui simule une décrémentation du compteur (qui n'est possible que depuis les états $\tau = 1$ ou $\tau = 2$) choisit de façon non déterministe d'arriver dans l'état $\tau = 2$ (l'automate devine que le compteur n'est pas encore nul après la décrémentation) ou l'état $\tau = 3$ (l'automate devine que le compteur vient de s'annuler).

La contrainte semilinéaire vérifie que si la simulation termine dans l'état $\tau = 2$, alors le compteur n'a jamais atteint zéro, et que si elle termine dans l'état $\tau = 3$, alors le compteur est bien nul. La contrainte correspondante est $\phi(x_1, x_2, x_3) := (x_3 = 2 \Rightarrow x_1 > x_2) \wedge (x_3 = 3 \Rightarrow x_1 = x_2)$.

La seule source d'ambiguïté possible pendant la simulation de $\mathcal{M}$ par $\mathcal{A}$ vient du choix d'entrer ou non dans l'état 3. Nous remarquons que $\mathcal{A}$ ne peut entrer dans l'état 3 que par une transition qui décrémente le compteur, et ne peut ensuite plus simuler aucune transition qui diminue le compteur. Par conséquent, un calcul de $\mathcal{A}$, pour être acceptant, est obligé d'entrer dans l'état 3 par la transition qui fait passer le compteur de 1 à 0 dans le calcul de $\mathcal{M}$ qu'il simule.

Entrer dans l'état 3 par une transition qui décrémente le compteur est aussi nécessaire dans la preuve non déterministe, sinon l'automate de Parikh accepte des calculs non acceptants de la RBCM.

Les calculs acceptants de $\mathcal{A}$ et de $\mathcal{M}$ sont ainsi facilement en bijection, les deux machines reconnaissent le même langage, et finalement $\mathcal{A}$ est faiblement non ambiguë par non-ambiguïté de $\mathcal{M}$. ■

Corollaire 6.30 (Inclusion).
▶ Comme toute RBCM déterministe bidirectionnelle est non ambiguë, nous déduisons l'inclusion suivante :

$$\mathrm{detRBCM} \subsetneq \text{2-detRBCM} \subseteq \mathrm{wuPA} \ .$$

Nous conjecturons que la dernière inclusion est stricte. ◀

6.3.5 La classe RCM

Dans cette section nous présentons une classe de langages, appelée RCM, introduite par [CM17]. Nous montrons notamment que cette classe de langages représente exactement la classe des langages de Parikh faiblement non ambigus. Nous rappelons la notation du vecteur de Parikh d'un mot :

Définition 6.31 (Image de Parikh d'un mot).
▶ Soit $\Gamma = \{a_1, \ldots, a_d\}$ un alphabet fixé à d lettres, avec $d \geqslant 1$, et w un mot dans Γ^*. Alors l'image de Parikh du mot w est le vecteur $\boldsymbol{\pi}^{\boldsymbol{\Gamma}}_{\boldsymbol{kh}}(w) = (|w|_{a_1}, \ldots, |w|_{a_d}) \in \mathbb{N}^d$ du nombre d'occurrences de chaque lettre dans w.

L'image de Parikh (notamment sa dimension, et l'ordre des lettres) d'un mot dépend de l'alphabet Γ fixé au départ. Lorsque celui-ci est clair par le contexte, on l'omettra pour simplifier les notations. ◀

Dans un premier temps, nous introduisons la classe LCL_R :

Définition 6.32 (Linear Constrained Language - Regular, [BMS92]).
▶ Soit Γ un alphabet de taille $d \geqslant 1$. Un langage $L \subseteq \Gamma^*$ appartient à la classe LCL_R s'il existe un langage rationnel R sur l'alphabet Γ, et un ensemble semilinéaire $C \subseteq \mathbb{N}^d$, tel que

Cf. le caveat de la Remarque 6.44 sur la définition de [BMS92]

$$L = R \cap [C]$$

[CFM12a] utilisent la notation $R \restriction_C$. $[C] = \boldsymbol{\pi}^{\boldsymbol{\Gamma}}_{\boldsymbol{kh}}{}^{-1}(C)$

où $[C] = \{w \in \Gamma^* \mid \boldsymbol{\pi}^{\boldsymbol{\Gamma}}_{\boldsymbol{kh}}(w) \in C\}$ désigne l'ensemble des mots sur Γ qui vérifient les contraintes semilinéaires C portant sur le nombre d'occurrences de chaque lettre. ◀

Exemple 6.33 .
▶ Le langage $L = \{a^n b^n c^n \mid n \in \mathbb{N}\} \in \mathrm{LCL}_R$. En effet, $L = R \cap [C]$, avec $R = a^*b^*c^*$ et $C = \{(n, n, n) \mid n \in \mathbb{N}\}$. ◀

Définition 6.34 (Automates de Parikh sur les lettres, [CFM12a]).
▶ Un automate de Parikh de dimension d est appelé automate de Parikh sur les lettres si pour chaque lettre $a \in \Sigma$, toute transition étiquetée par la lettre a est étiquetée par le même vecteur $\boldsymbol{v}_a$. Autrement dit si $(a, \boldsymbol{v})$ et $(a, \boldsymbol{v'})$ sont deux étiquettes de deux transitions de l'automate, alors $\boldsymbol{v} = \boldsymbol{v'}$. ◀

À ne pas confondre avec les automates normalisés pour lesquels le vecteur caractérise la lettre.

Le lien entre la classe LCL_R et les automates de Parikh est bien compris :

Proposition 6.35 ([CFM12a, Mas17]).
▶ La classe LCL_R reconnaît les mêmes langages que les automates de Parikh sur les lettres. ◀

Cf. Remarque 6.44

Cette classe est ainsi assez peu expressive (encore moins que les langages de Parikh déterministes). Ces langages ont la contrainte très forte de devoir compter chaque lettre de la même façon au cours du mot, sans pouvoir différencier certains blocs. Ainsi :

Proposition 6.36 ($(a^nb^n)^2 \notin \mathrm{LCL}_R$, [CFM12a]).
▶ Le langage $L = \{a^nb^na^mb^m \mid n, m \in \mathbb{N}^*\}$ n'est pas dans LCL_R. ◀

Idée de preuve. La Proposition 5.4 de [CFM12a] permet de prouver la proposition tout en évitant un argument de pompage, à l'aide de l'introduction d'un langage non régulier (le pompage étant ainsi reporté dans la preuve de non régularité du langage).

Nous proposons, dans ce cas simple, une ébauche de preuve directe par pompage. Par l'absurde supposons que $L = R \cap [C]$ avec R un langage régulier sur $\{a, b\}$ et C une contrainte semilinéaire de dimension 2. Un argument de pompage permet de dire qu'il existe un entier n suffisamment grand, et un entier $0 < s < n$, tel que $a^nb^na^nb^n \in R$ et $a^nb^{n+s}a^nb^{n-s} \in R$. Comme $a^nb^na^nb^n \in L$, le vecteur $(2n, 2n) \in C$. L'image de Parikh du mot $a^nb^{n+s}a^nb^{n-s} \in R$ est donc $(2n, 2n) \in C$. Donc ce mot appartient à $R \cap [C] = L$, mais pourtant il n'est pas dans L. Contradiction. ■

Nous utiliserons la technique de [CFM12a] pour l'analogue hors-contexte de LCL_R, au Lemme 6.64.

Pour différencier le comptage de certaines lettres, et ainsi enrichir la classe de langages décrits de cette façon, Castiglione et Massazza ont introduit un morphisme :

Définition 6.37 (RCM, [CM17]).
▶ Soit Σ un alphabet non vide. Un langage $L \subseteq \Sigma^*$ appartient à la classe RCM s'il existe un alphabet Γ de taille $d \geqslant 1$, un langage rationnel R sur l'alphabet Γ, un ensemble semilinéaire $C \subseteq \mathbb{N}^d$, et enfin un morphisme lettre à lettre $\mu : \Gamma^* \to \Sigma^*$ *injectif sur* $R \cap [C]$, tel que

$$L = \mu(R \cap [C]) .$$

Autrement dit, L est l'image d'un langage L' de LCL_R par un morphisme lettre à lettre et injectif sur L'.

On note alors $L = \langle R, C, \mu \rangle$. ◀

Remarque 6.38 .
▶ La notation $L = \langle R, C, \mu \rangle$ explique le nom de la classe. Nous pouvons aussi proposer l'acronyme *Regular Constrained language with an injective Morphism.* ◀

Exemple 6.39 (Morphisme et renommage des lettres).
▶ Le langage $L = (a^nb^n)^2 := \{a^nb^na^mb^m \mid n, m \in \mathbb{N}^*\} \in \mathrm{RCM}$. En effet, posons $\Sigma = \{a, b\}$, $\Gamma = \{a_1, a_2, b_1, b_2\}$, et considérons le langage régulier $R = a_1^*b_1^*a_2^*b_2^*$. Alors $L = \mu(R \cap [C])$, où $C = \{(n, n, m, m) \mid n, m \in \mathbb{N}^*\}$ et μ est le morphisme lettre à lettre défini par $\mu(a_1) = \mu(a_2) = a$ et $\mu(b_1) = \mu(b_2) = a$. On vérifie facilement que μ est bien injectif sur $R \cap [C]$.

Ce langage justement n'appartient pas LCL_R, et le morphisme permet ainsi de différencier la première paire (a, b) de la seconde. ◀

Remarque 6.40 (Injectivité du morphisme).
▶ La contrainte d'injectivité du morphisme dans la définition de [CM17] a été initialement exploitée pour pouvoir établir une bijection entre les mots de $R \cap [C]$ de longueur n et ceux de L, afin que les séries génératrices de comptage des deux langages soient égales. ◀

Le Théorème 6.41 permet de se convaincre facilement que RCM *sans la contrainte d'injectivité est égale à la classe des langages de Parikh non déterministes.*

Nous pouvons désormais démontrer la proposition principale de cette section :

Théorème 6.41 (RCM = wuPA).
▶ La classe RCM est égale à la classe des langages de Parikh faiblement non ambigus. ◀

Démonstration. Nous montrons comment reconnaître un langage dans RCM par un automate de Parikh faiblement non ambigu, et comment faire la procédure inverse. Les deux directions sont assez syntaxiques, vu que la définition de RCM est finalement très proche de celle d'un automate de Parikh (les deux sont grossièrement un langage régulier avec des contraintes semilinéaires). Comme il ne s'agit pas de transformations déjà publiées dans la littérature, à simplement adapter dans le cas faiblement non ambigu, j'ai décidé de détailler les preuves.

L'idée principale consiste à faire correspondre la contrainte d'injectivité du morphisme de RCM avec celle de l'unicité d'un calcul acceptant d'un automate de Parikh faiblement non ambigu.

De RCM vers PA. Soit $L \in \mathrm{RCM}$. On introduit $\Gamma = \{a_1 \dots a_d\}$ et Σ deux alphabets, R un langage régulier sur Γ, C un ensemble semilinéaire de $\mathbb{N}^d$ et finalement $\mu : \Gamma^* \longrightarrow \Sigma^*$ un morphisme lettre à lettre injectif sur $R \cap [C]$, tels que $L = \langle R, \mu, C \rangle$.

Nous rappelons que pour $w \in \Gamma^*$, $\boldsymbol{\pi_{kh}}(w) = (|w|_{a_1}, \dots, |w|_{a_r})$ désigne son image de Parikh. Nous supposons que R est reconnu par un automate déterministe fini $\mathcal{A}$.

L'automate de Parikh $\mathcal{B}$ reconnaissant L s'obtient facilement en remplaçant dans $\mathcal{A}$ chaque transition de la forme (q_1, a_i, q_2) par la transition $(q_1, (\mu(a_i), \boldsymbol{e_{a_i}}), q_2)$ où $\boldsymbol{e_{a_i}}$ est le vecteur unitaire de dimension $|\Gamma| = r$ qui n'a qu'une seule coordonnée non nulle, la coordonnée i, qui contient un 1. Nous gardons la même contrainte semilinéaire C pour $\mathcal{B}$.

Les calculs de $\mathcal{A}$ et de $\mathcal{B}$ sont clairement en bijection, puisque $\mathcal{B}$ enregistre dans le vecteur de chaque transition la lettre de Γ correspondant à la transition associée dans $\mathcal{A}$. Ainsi pour tout calcul de $\mathcal{B}$ étiqueté par $(w \in \Sigma^*, \boldsymbol{d} \in \mathbb{N}^d)$, il existe un calcul de $\mathcal{A}$ passant par exactement les mêmes états, étiqueté par un mot $w_2 \in \Gamma^*$ tel que $\mu(w_2) = w$ et $\boldsymbol{d} = \boldsymbol{\pi_{kh}}(w_2)$. Réciproquement, tout calcul de $\mathcal{A}$ étiqueté par le mot w peut se traduire en calcul dans $\mathcal{B}$ étiqueté par $(\mu(w), \boldsymbol{\pi_{kh}}(w))$, et passant par les mêmes états.

Lemme 6.42.
▶ $L = \mathcal{L}(\mathcal{B})$ et $\mathcal{B}$ est faiblement non ambigu. ◀

Preuve du lemme. Par double inclusion.

$\subseteq$ Soit $w \in L$. Il existe un mot $w_2 \in R$ tel que $\mu(w_2) = w$ et $\boldsymbol{\pi_{kh}}(w_2) \in C$. Il existe donc un calcul acceptant dans $\mathcal{A}$ pour w_2, et donc par ce qui précède, un calcul passant par les mêmes états dans $\mathcal{B}$ étiqueté par $(w, \boldsymbol{\pi_{kh}}(w_2))$. Comme $\boldsymbol{\pi_{kh}}(w_2) \in C$, ce calcul est acceptant dans $\mathcal{B}$. Donc $w \in \mathcal{L}(\mathcal{B})$.

⊇ Soit $w = b_1 \dots b_s \in \mathcal{L}(\mathcal{B})$. Il existe un calcul acceptant dans $\mathcal{B}$ étiqueté par $((b_1, \boldsymbol{d_1}), \dots, (b_s, \boldsymbol{d_s}))$ tel que $\boldsymbol{d} \in C$, avec $\boldsymbol{d} = \boldsymbol{d_1} + \dots + \boldsymbol{d_s}$. Il existe donc un calcul dans $\mathcal{A}$ étiqueté par un mot $w_2 \in \Gamma^*$ tel que $\mu(w_2) = w$ et $\boldsymbol{d} = \boldsymbol{\pi_{kh}}(w_2)$, passant par les mêmes états. Ce calcul est donc acceptant pour $\mathcal{A}$, et $w_2 \in R$. Donc $w_2 \in R \cap [C]$ et finalement $w \in \mu(R \cap [C]) = L$.

Supposons maintenant qu'il existe un autre calcul acceptant dans $\mathcal{B}$ étiqueté par $((b_1, \boldsymbol{d'_1}), \dots, (b_s, \boldsymbol{d'_s}))$ tel que $\boldsymbol{d'} \in C$, avec $\boldsymbol{d'} = \boldsymbol{d'_1} + \dots + \boldsymbol{d'_s}$. Comme il y a une bijection entre les transitions de $\mathcal{A}$ et celles de $\mathcal{B}$, on peut mettre en bijection ce calcul avec un calcul de $\mathcal{A}$, étiqueté par un mot $w_3 \in R \cap [C]$. Comm $\mu(w_3) = \mu(w_2) = w$, l'injectivité de μ sur $R \cap [C]$ implique $w_3 = w_2$. Donc $\boldsymbol{d'_1} = \boldsymbol{d_1}, \dots, \boldsymbol{d_s} = \boldsymbol{d'_s}$. De plus, $\mathcal{A}$ est déterministe, donc les deux calculs empruntent les mêmes transitions dans $\mathcal{A}$ et passent donc par les mêmes chemins. Donc les deux calculs dans $\mathcal{B}$ sont identiques.

Ainsi pour tout mot, il y a au plus un calcul acceptant dans $\mathcal{B}$: $\mathcal{B}$ est faiblement non ambigu.

■

De PA vers RCM. Soit $\mathcal{B}$ un automate de Parikh faiblement non ambigu de dimension r, dont les transitions sont étiquetées par des couples $(a_i, \boldsymbol{e_i})$, par la Proposition 6.22, où chaque vecteur $\boldsymbol{e_i}$ est toujours associé à la même lettre a_i (par contre une lettre peut être associée à plusieurs vecteurs différents, ainsi pour $i \neq j$, on peut cependant avoir $a_i = a_j$).

On considère l'alphabet $\Gamma = \{(a_i, \boldsymbol{e_i}) \mid i \in [1, r]\} \subseteq \Sigma \times \{\boldsymbol{e_i}\}_{i \in [1,d]}$.

L'automate $\mathcal{A}$ est obtenu simplement en considérant $\mathcal{B}$ comme un automate fini sur l'alphabet Γ (en voyant les vecteurs comme des lettres qu'on concatène). Le morphisme μ est défini par la projection sur la première coordonnée : $\mu((a_i, \boldsymbol{e_i})) = a_i$.

Soit R le langage régulier reconnu par $\mathcal{A}$: il s'agit du langage des couples lettre-vecteur qui étiquettent les calculs de $\mathcal{B}$ qui joignent l'état initial à un état final. On remarque par ailleurs que $|\Gamma| = r$, et que le nombre d'occurrences de la lettre $(a_i, \boldsymbol{e_i})$ dans un calcul de $\mathcal{A}$ est égal à la coordonnée i du vecteur étiquetant le calcul correspondant dans $\mathcal{B}$. Ainsi, $R \cap [C]$ est le langage des couples lettre-vecteur qui étiquettent les calculs acceptants de $\mathcal{B}$. Donc $\mu(R \cap [C]) = \mathcal{L}(\mathcal{B})$.

Montrons que μ est injectif sur $R \cap [C]$. Soient $w_1, w_2 \in R \cap [C]$ tels que $\mu(w_1) = \mu(w_2)$. Comme w_1 et w_2 représentent deux calculs acceptants de $\mathcal{B}$, qui sont étiquetés par le même mot $\mu(w_1)$, par faible non-ambiguïté, $w_1 = w_2$. Donc μ est bien injectif.

■

Corollaire 6.43 (Corollaire de la preuve).
▶ Tout langage dans RCM peut être défini par un triplet $\langle R, C, \mu \rangle$, où C est une conjonction de contraintes linéaires de la forme :

$$\{(n_1, \dots, n_r) \mid \bigwedge_{i=1}^{r} (\textstyle\sum_{j=1}^{r} a_{i,j} n_j = 0), \text{ avec } a_{i,j} \in \{-1, 0, 1\}\}.$$

◀

Démonstration. C'est une conséquence de la construction du passage de wuPA à RCM, qui ne modifie pas le semilinéaire si l'automate est préalablement normalisé par la Proposition 6.22. ■

Remarque 6.44 (Différence de définition pour les contraintes semilinéaires).
▶ Les définitions de LCL_R et RCM dans cette section ne sont pas exactement celles de [CM17]. En effet, dans leurs articles, les auteurs utilisent une définition non standard des contraintes semilinéaires C : ils considèrent des combinaisons booléennes de contraintes linéaires de la forme

$$a_1 n_1 + \ldots + a_d n_d \triangle a_{d+1}\,,$$

où $\triangle \in \{\leqslant, \geqslant, =, \neq, <, >\}$ et $a_1, \ldots, a_{d+1}$ sont des entiers relatifs. Si ces contraintes C sont bien semilinéaires, elles ne recouvrent pas tous les ensembles semilinéaires, contrairement à ce qui est affirmé dans [CM17] (leur exemple 1 qui illustre la conversion d'un semilinéaire vers un ensemble de contraintes de cette forme est par ailleurs faux). En effet, rien qu'en dimension 1, il n'est pas possible d'exprimer l'ensemble semilinéaire $\{2n \mid n \in \mathbb{N}\}$ avec uniquement des inégalités de cette forme. C'est d'ailleurs pour cela que l'élimination des quantificateurs dans les formules de Presburger introduit des modulos. Nous appellerons dans la suite de cette remarque *contraintes semilinéaires sans modulo* un ensemble de contraintes de cette forme. Les contraintes semilinéaires sans modulo ne sont pas stables par projection : par exemple la projection du semilinéaire sans modulo $\{(n, m) \mid n = m + m\}$ sur la première coordonnée est l'ensemble des nombres pairs.

Si les contraintes semilinéaires sans modulo sont une sous-classe stricte des semilinéaires, cela n'implique pas forcément que les classes LCL_R et RCM définies avec un semilinéaire sans modulo diffèrent : en effet le langage régulier introduit dans la définition de LCL_R et de RCM est capable de calculer certains modulos. Par exemple, pour le langage $\{a^{2n} \mid n \in \mathbb{N}\} = a^* \cap [\{2n \mid n \in \mathbb{N}\}] = (a^2)^* \cap [\mathbb{N}]$, la première représentation utilise un semilinéaire avec modulo, tandis que la seconde utilise un semilinéaire sans modulo, la parité étant gérée par le langage régulier.

C'est le cas pour la classe RCM, comme nous l'avons démontré au corollaire précédent : les classes sont équivalentes, que l'on demande à la contrainte C d'être un semilinéaire ou une combinaison linéaire de contraintes linéaires sans modulo.

En revanche, ce n'est pas le cas pour la classe LCL_R de [BMS92, CM17] : l'absence de morphisme restreint la dimension au nombre de lettres du langage, et le formalisme est ainsi trop rigide pour faire communiquer la partie automate et la partie semilinéaire. La classe définie avec un semilinéaire sans modulo est strictement plus petite que celle définie avec un semilinéaire quelconque (et donc n'est pas égale aux langages de Parikh sur les lettres, contrairement à ce qui est annoncé dans [CM17]). Un exemple de langage qui sépare les deux classes est le langage $L = \{a^n b^m \mid (m \text{ pair}) \vee (n > m)\}$.

En effet, supposons que $L = \{a^n b^m \mid (m \text{ pair}) \vee (n > m)\}$ s'écrive $L = R \cap [C]$ avec R un langage régulier sur $\{a, b\}$ et C un ensemble de contraintes semilinéaires sans modulo, de dimension 2. On peut mettre C sous une forme normale disjonctive :

$$C = \bigvee_{j=1}^{r} C_j\,,$$

avec chaque C_j qui est une conjonction d'inégalités de la forme $a_1 n + a_2 m \triangle a_3$ avec $\triangle \in \{<, >, \leqslant, \geqslant\}$ (on remplace $x \neq y$ par $x < y \wedge y > x$ et $x = y$ par $x \leqslant y \wedge y \geqslant x$).

Le semilinéaire C est donc une union finie de polygones possiblement non bornés de $\mathbb{N}^2$. La preuve va utiliser le fait qu'aucune union finie d'intersections de demi-plans

n'est capable de ne sélectionner que des points d'ordonnée paire le long d'une droite verticale.

Supposons que R est reconnu par un automate fini déterministe $\mathcal{A}$. Soit n un entier pair strictement plus grand que le nombre d'états de $\mathcal{A}$. On s'intéresse aux mots $w_k = a^n b^{n+k} \in L$, avec k pair. L'ensemble des w_k lorsque k parcourt les nombres pairs est infini, donc il existe un polygone C_j qui valide l'image de Parikh d'une infinité de ces mots. Ainsi, on peut trouver une suite strictement croissante k_i de nombres pairs qui tend vers l'infini, telle que $\forall i \in \mathbb{N}, (n, n+k_i) \in C_j$.

Fixons une contrainte de C_j de la forme $a_1 n + a_2 m \triangle a_3$ avec $\triangle \in \{<, >, \leqslant, \geqslant\}$. Supposons que $a_2 > 0$. Alors comme cette contrainte est vérifiée par tous les vecteurs $(n, n+k_i)$, nous en déduisons que pour tout i, $(a_1 + a_2)n + a_2 k_i \triangle a_3$, avec n fixé et $a_2 k_i \to +\infty$. Par conséquent $\triangle \in \{>, \geqslant\}$, et donc il existe un rang m_0 tel que pour tout $m \geqslant m_0$, $(n, n+m) \in C_j$ (sans contrainte de parité sur m). Le raisonnement est le même si $a_2 < 0$, nous en déduisons que $\triangle \in \{<, \leqslant\}$ et que donc la contrainte est satisfaite pour tout vecteur $(n, n+m)$ pour m suffisamment grand. Et si $a_2 = 0$, la contrainte est vérifiée pour tout vecteur $(n, n+m)$, peu importe la valeur de m.

Comme C_j est une conjonction finie de contraintes de cette forme, nous pouvons donc trouver un entier $m > 0$ impair tel que $(n, n+m) \in C_j$. Comme $a^n b^{n+m} \notin L$ mais $(n, n+m) \in C$, cela signifie que $a^n b^{n+m} \notin R$. Comme R est déterministe, il existe un unique chemin dans $\mathcal{A}$ étiqueté par $a^n b^{n+m}$, allant de l'état initial à un état non acceptant. Mais comme n est supérieur au nombre d'états de $\mathcal{A}$, on peut itérer une boucle de ce chemin, au niveau de la lecture des a : il existe donc un calcul non acceptant dans $\mathcal{A}$ pour un mot de la forme $a^N b^{n+m}$ avec $N > n+m$. Comme $\mathcal{A}$ est déterministe, cela signifie que ce mot n'est pas dans R, donc pas dans L. Il s'agit d'une contradiction car il appartient bien à L. ◀

L'argument utilise fortement le fait que les langages réguliers sont stables par complémentaire ; il ne s'adaptera pas pour la classe LCL*, qui remplace R par un langage algébrique non ambigu.*

Nous finissons cette section par quelques corollaires de l'égalité $\mathrm{RCM} = \mathrm{wuPA}$. Dans [CM17], les auteurs conjecturent que la classe des langages reconnus par des RBCM unidirectionnelles déterministes est strictement incluse dans RCM. Cette conjecture a été démontrée dans deux cas très particuliers, pour les $\mathrm{detRBCM}$ avec un seul compteur qui ne s'inverse qu'une fois [Mas17], et les $\mathrm{detRBCM}$ sans cycle négatif [Mas18]. Nous prouvons une version plus générale de leur conjecture :

Corollaire 6.45 (Preuve de la conjecture de [CM17]).
▶ Tout langage reconnu par une RBCM déterministe bidirectionnelle est dans RCM :

$$\mathrm{detRBCM} \subsetneq \text{2-}\mathrm{detRBCM} \subseteq \mathrm{wuPA} = \mathrm{RCM} = \mathrm{uRBCM}\ .$$

◀

Nous conjecturons que la dernière inclusion est stricte.

Démonstration. Il s'agit d'une conséquence directe de l'inclusion déjà énoncée dans le cas des RBCM au Corollaire 6.30 et du Théorème 6.41. ■

Corollaire 6.46 (Inclusion stricte de RCM dans les PA, [CM17]).
▶ La classe RCM est strictement incluse dans les langages de Parikh. ◀

Démonstration. C'est une conséquence directe de l'équivalence des classes RCM et wuPA, et du fait que les langages de Parikh faiblement non ambigus sont strictement inclus dans les langages de Parikh, par la Proposition 6.9. Notons que la preuve de cette inclusion stricte dans [CM17] est inexacte, car leur argument permet uniquement de conclure que si RCM reconnaissait tous les langages de Parikh, alors il n'existerait

pas de conversion récursive d'un automate de Parikh quelconque dans le formalisme de RCM.

Nous verrons dans le chapitre 7 que la série génératrice du langage de Shamir présenté dans la Proposition 6.9 n'est pas holonome, ce qui permet de préciser le Théorème 12 de [CM17] : il existe bien des langages de Parikh qui ont une série génératrice non holonome. ■

Nous concluons cette section par une remarque sur la stabilité de RCM par union.

Remarque 6.47 (La question de la stabilité par union).
▶ Comme nous l'avons dit précédemment, nous ne savons pas si les langages de Parikh faiblement non ambigus sont stables par union. Dans [CM17, Theorem 4], les auteurs affirment que la classe RCM est close par union.

Transposée dans le formalisme des automates de Parikh, leur preuve consiste simplement à reconnaître l'union de deux automates de Parikh faiblement non ambigus par un automate produit. Il n'y a aucune raison, sans argument supplémentaire, que cette simple construction fournisse un automate faiblement non ambigu, et on peut construire facilement des exemples pour lesquels ce n'est pas le cas.

Cependant, leur preuve contient une erreur de raisonnement non rattrapable. Nous proposons dans cette remarque un contre-exemple à la construction de leur preuve.

Nous suivons donc leur construction dans le cas particulier de l'union de $L = (R, C, \mu)$ et de $L' = (R', C', \mu')$, définis comme suit :
- $R = \{a, A\}$ est un langage fini (donc régulier) sur l'alphabet $\Gamma = \{a, A\}$,
- $C = (\mathbb{N} \times \{0\}) \setminus (0, 0)$,
- $\Sigma = \{c\}$,
- $\mu : \Gamma^* \mapsto \Sigma^*$ défini par $\mu(a) = \mu(A) = c$.
- $R' = \{b\}$ est un langage fini (donc régulier) sur l'alphabet $\Gamma' = \{b\}$,
- $C' = \mathbb{N} \setminus \{0\}$,
- $\mu' : (\Gamma')^* \mapsto \Sigma^*$ défini par $\mu(b) = c$.

Ainsi $L = \mu(R \cap [C]) = \{c\}$ et $L' = \mu'(R' \cap [C']) = \{c\}$.

La construction de la preuve introduit alors les langages suivants :
- $M := \mu(R) \cap \mu(R') = \{c\}$
- $R_1 := R \cap \mu^{-1}(M) = R$
- $R'_1 := R' \cap \mu'^{-1}(M) = R'$
- $R_2 := R \setminus R_1 = \emptyset$
- $R'_2 := R \setminus R'_1 = \emptyset$

La preuve construit alors à partir de ces ensembles le langage $M' = \{\tau_{ab}, \tau_{Ab}\}$, le morphisme $\mu''(\tau_{ab}) = \mu''(\tau_{Ab}) = c$, et la contrainte semilinéaire $C'' = \mathbb{N}^2 \setminus (0, 0)$. On a donc $M' \cap [C''] = M'$.

Les auteurs annoncent alors que l'union de L et L' est $\mu''(M' \cap [C'']) = \{c\}$, ce qui est le cas, mais contrairement à ce qu'ils affirment, le morphisme n'est pas injectif sur $M' \cap [C''] = M'$, puisque $\mu''(\tau_{ab}) = \mu''(\tau_{Ab}) = c$.

Cette remarque invalide uniquement la preuve donnée dans [CM17], nous ne savons pas si RCM est clos par union. ◀

6.4 Automates de Parikh à pile faiblement non ambigus et modèles équivalents

6.4.1 Définition, propriétés de clôture

Dans cette section, nous étendons le modèle des automates de Parikh en ajoutant une pile.

Définition 6.48 (Automate de Parikh à pile, [Kar04]).
▶ Un automate de Parikh à pile $\mathcal{B}$ de dimension d est un couple $(\mathcal{A}, C)$, où $\mathcal{A}$ est un automate à pile sur un alphabet fini Σ, dont les transitions sont étiquetées par des couples $(a, \boldsymbol{v})$, avec $a \in \Sigma \cup \{\varepsilon\}$ et $\boldsymbol{v} \in \mathbb{N}^d$, et C est un ensemble semilinéaire de dimension d. En toute généralité, nous supposons que $\mathcal{A}$, vu comme un automate à pile, accepte par état final.

Les transitions étiquetées par un couple de la forme $(\varepsilon, \boldsymbol{v})$ sont appelées des ε-transitions.

Un calcul d'un automate de Parikh à pile est ainsi étiqueté par un couple mot-vecteur $(w, \boldsymbol{v})$, où w est la concaténation des lettres et $\boldsymbol{v}$ est la somme des vecteurs étiquetant les transitions empruntées.

Un calcul étiqueté par $(w, \boldsymbol{v})$ est dit acceptant dans $\mathcal{B}$ si c'est un calcul acceptant de l'automate à pile $\mathcal{A}$, et si de plus le vecteur $\boldsymbol{v}$ appartient au semilinéaire C.

Le langage accepté par $\mathcal{B}$, noté $\mathcal{L}(\mathcal{B})$ est l'ensemble des mots étiquetant des calculs acceptants de $\mathcal{B}$.

Un automate de Parikh à pile $\mathcal{B}$ est appelé *faiblement non ambigu* si tout mot étiquette au plus un calcul acceptant de $\mathcal{B}$.

On appelle *langage algébrique de Parikh* (faiblement non ambigu) un langage reconnu par un automate de Parikh à pile (faiblement non ambigu).

Nous notons $\mathbb{P}\mathrm{A}$ (resp. $\mathrm{wu}\mathbb{P}\mathrm{A}$) l'ensemble des langages algébriques de Parikh (resp. faiblement non ambigus). ◀

Pushdown Parikh Automata

Remarque 6.49.
▶ [Kar04] définit l'automate $\mathcal{A}$ comme un automate à pile fini sur l'alphabet $\Sigma \times D$, avec $D \subseteq \mathbb{N}^d$, le produit étant vu comme un alphabet à part entière, c'est-à-dire que chaque couple de la forme $(a, \boldsymbol{v})$ est considéré comme une lettre. Cette présentation formelle est concise mais a l'inconvénient de donner une définition un peu stricte des ε-transitions. En effet, une ε-transition sur un automate est généralement une transition qui ne lit pas de lettre de l'alphabet ; ainsi dans [Kar04], les ε-transitions ne sont pas étiquetées par des vecteurs. ◀

Les automates de Parikh à pile étendent strictement la classe des langages algébriques et des langages de Parikh :

Proposition 6.50.
▶ Les langages algébriques et les langages de Parikh sont strictement inclus dans les langages algébriques de Parikh. ◀

Démonstration. L'inclusion est immédiate, par définition, car un automate de Parikh est un automate de Parikh à pile sans pile, et un automate à pile est un automate de Parikh à pile avec des vecteurs nuls.

La stricte inclusion des langages de Parikh vient du fait que le langage des palindromes $L_1 := \{w\#w^R \mid w \in \{a,b\}^*\}$ avec w^R qui désigne le mot miroir de w est algébrique non ambigu (et même déterministe), mais n'est pas un langage de Parikh ([KR02, Kar04, CFM12a]). Celle des langages algébriques vient du fait que le langage $L_2 := \{a^n b^n c^n \mid n \in \mathbb{N}\}$ n'est classiquement pas algébrique.

[KR02] le prouvent pour le langage des mots de la forme ww, mais leur preuve s'adapte aux palindromes [Kar04]. La preuve de [CFM12a] est beaucoup plus élémentaire que l'originale de [KR02].

La concaténation $L_3 := L_1\#L_2$ n'est ni un langage de Parikh, ni un langage algébrique, mais est un langage de Parikh algébrique. En effet, si L_3 était un langage de Parikh, alors par stabilité par quotient à gauche, $L_1 = L_3\#^{-1}$ serait un langage de Parikh, et si L_3 était un langage algébrique, alors de même $L_2 = (\#\#)^{-1}L_3$ serait algébrique. ■

Remarque 6.51 .
▶ La preuve précédente permet de montrer de la même façon que les langages algébriques non ambigus et les langages de Parikh faiblement non ambigus sont strictement inclus dans les langages algébriques de Parikh faiblement non ambigus. ◀

Définition 6.52 .
▶ Un langage algébrique de Parikh est dit *intrinsèquement faiblement non ambigu* s'il n'est reconnu par aucun automate de Parikh à pile faiblement non ambigu. ◀

La proposition est démontrée à la Proposition 7.56 du chapitre suivant.

Nous annonçons la proposition suivante, qui sera démontrée au chapitre suivant :

Proposition 6.53 (Non clôture des langages algébriques de Parikh faiblement non ambigus).
▶ La classe des langages algébriques de Parikh faiblement non ambigus n'est *pas* close par intersection, union, concaténation, ni quotient à gauche. ◀

Remarque 6.54 (Quelques propriétés de clôture).
▶ En revanche, on adapte facilement les preuves sur les langages de Parikh faiblement non ambigus pour montrer que la classe des langages algébriques de Parikh faiblement non ambigus est close par intersection avec un langage de Parikh faiblement non ambigu, et par quotient à gauche avec un mot w fixé. ◀

Remarque 6.55 (Question ouverte).
▶ Les techniques développées au chapitre 7 ne nous permettent pas de répondre à la question de la clôture par complémentaire, qui reste ouverte. ◀

6.4.2 Élimination des ε-transitions

Dans la définition des automates de Parikh à pile, nous avons autorisé les ε-transitions : l'automate peut effectuer des calculs, additionner des vecteurs et changer d'état sans lire de lettre du mot. Il est possible, comme pour les automates à pile (cf par exemple [Car08, p.101]), d'éliminer les ε-transitions sur un automate de Parikh à pile faiblement non ambigu, tout en préservant la faible non-ambiguïté. La preuve, si elle n'est conceptuellement pas compliquée (elle suit les idées de la preuve pour les automates à pile), est assez fastidieuse, car elle introduit d'autres modèles d'acceptation pour les langages, et demande de vérifier soigneusement que les constructions n'introduisent pas d'ambiguïté. Pour ne pas interrompre la présentation des modèles équivalents aux wu$\mathbb{P}$A par une longue preuve technique, nous admettons dans cette section la proposition suivante, qui sera démontrée dans la section 6.5 :

Proposition 6.56 (Élimination des ε-transitions dans un wu$\mathbb{P}$A).
▶ Tout langage algébrique de Parikh faiblement non ambigu est reconnaissable par un automate de Parikh faiblement non ambigu qui ne possède aucune ε-transition (toutes les transitions lisent une lettre). ◀

La proposition suivante est l'analogue avec pile du Corollaire 6.19 et de la Proposition 6.20 :

Proposition 6.57 (Normalisation d'un wu$\mathbb{P}$A)**.**
▶ Tout langage algébrique de Parikh faiblement non ambigu est reconnaissable

— par un automate de Parikh à pile faiblement non ambigu sans ε-transition, de dimension paire $2d$, dont les transitions sont étiquetées par des vecteurs de la forme $(\boldsymbol{e_i}, \boldsymbol{e_j})$, où $\boldsymbol{e_i}, \boldsymbol{e_j} \in \mathbb{N}^d$, et le semilinéaire d'acceptation est $C = \{(n_1, \ldots, n_{2d}) \mid n_1 = n_{d+1} \wedge n_2 = n_{d+2} \wedge \ldots \wedge n_d = n_{2d}\}$.

— par un automate de Parikh à pile faiblement non ambigu sans ε-transition, de dimension r, dont les transitions sont étiquetées par des vecteurs de la forme $\boldsymbol{e_i}$, où $\boldsymbol{e_i} \in \mathbb{N}^r$, et le semilinéaire d'acceptation est de la forme :

$$C = \left\{(n_1, \ldots, n_r) \mid \bigwedge_{i=1}^{r} (\textstyle\sum_{j=1}^{r} a_{i,j} n_j = 0), \text{ avec } a_{i,j} \in \{-1, 0, 1\}\right\}.$$

De plus, chaque vecteur $\boldsymbol{e_i}$ est associé à une seule lettre, c'est-à-dire que si $(a, \boldsymbol{e_i})$ et $(b, \boldsymbol{e_i})$ étiquettent deux transitions, alors $a = b$.

◀

Démonstration. Il s'agit de la même preuve que pour les versions sans pile, présentée dans le Corollaire 6.19 et la Proposition 6.20. ■

6.4.3 Équivalence avec les RBCM à pile unidirectionnelles non ambiguës

Définition 6.58 (RBCM à pile, [Iba78])**.**
▶ La classe $\mathrm{NPCM}(k, m, n)$ désigne l'ensemble des machines à k compteurs à (m, n)-inversion bornée, étendues avec une pile.

Une RBCM à pile est non ambiguë si tout mot est accepté par au plus un calcul acceptant. ◀

Proposition 6.59 ([Iba78])**.**
▶ Tout langage reconnu par une RBCM à pile unidirectionnelle ($m = 0$) non ambiguë est reconnaissable par une RBCM à pile unidirectionnelle, non ambiguë, et dont chaque compteur ne peut plus s'incrémenter après avoir été incrémenté ($n = 1$). ◀

Contrairement aux RBCM sans pile, il n'est plus possible de simuler une machine bidirectionnelle par une machine unidirectionnelle. En effet, l'image de Parikh du langage d'une machine à pile unidirectionnelle est semilinéaire, tandis que les machines bidirectionnelles peuvent simuler des intersections de langages algébriques, dont l'image de Parikh n'est pas semilinéaire [Iba78].

Idée de la preuve. Il s'agit d'une simple adaptation de l'argument de [BB74, Iba78] pour simuler de façon déterministe, avec m compteurs à une seule inversion, un compteur à au plus $2m$ inversions. ■

Puisque nous avons autorisé les ε-transitions dans la définition des automates de Parikh à pile, la proposition suivante est naturelle :

Proposition 6.60 .
▶ Les RBCM à pile unidirectionnelles non ambiguës et les automates de Parikh à pile faiblement non ambigus reconnaissent les mêmes langages. ◀

Démonstration. Par une simple adaptation de la preuve du cas sans pile du Théorème 6.29. ■

6.4.4 Équivalence avec la classe LCM

Définition 6.61 (Linear Constrained Language (LCL), [Mas93]).
▶ Soit Γ un alphabet de taille $d \geqslant 1$. Un langage $L \subseteq \Gamma^*$ appartient à la classe LCL s'il existe un langage algébrique non ambigu G sur l'alphabet Γ, et un ensemble semilinéaire $C \subseteq \mathbb{N}^d$, tel que

$$L = G \cap [C]$$

où nous rappelons que $[C] = \{w \in \Gamma^* \mid \boldsymbol{\pi}_{\boldsymbol{kh}}^{\Gamma}(w) \in C\}$ désigne l'ensemble des mots sur Γ qui vérifient les contraintes semilinéaires C portant sur le nombre d'occurrences de chaque lettre. ◀

Remarque 6.62 .
▶ Nous avons choisi de définir la classe LCL avec C semilinéaire quelconque, tandis qu'historiquement la classe LCL est définie dans [Mas93] avec un semilinéaire sans modulo. Nous avons déjà vu avec la classe LCL_R que ces deux définitions n'ont pas de raison d'être équivalentes. Le contre-exemple que nous avions trouvé pour séparer les deux définitions de LCL_R ne s'applique pas pour LCL, car il était algébrique non ambigu, et de plus la technique de preuve reposait sur un passage au complémentaire, tandis que les langages algébriques ne sont pas clos par complémentaire. Nous n'avons pas trouvé pour l'instant de contre-exemple, mais nous pensons que les deux définitions ne sont pas équivalentes. La classe que nous appelons LCL est donc potentiellement plus générale que celle décrite dans [Mas93]. ◀

Exemple 6.63 .
▶ Le langage $L = \{w\#w^R \mid |w|_a = |w|_b\} \in \mathrm{LCL}$. En effet, $L = G \cap [C]$, avec G le langage non ambigu des palindromes sur $\{a, b\}$, et $C = \{(n, n) \mid n \in \mathbb{N}\}$. ◀

Lemme 6.64 (Version à pile du Lemme 5.4 de [CFM12a]).
▶ Soit $L = G \cap [C] \in \mathrm{LCL}$, avec G un langage algébrique non ambigu. Soit E un langage régulier tel que $L \cap E$ n'est pas algébrique. Alors il existe un mot w de $G \cap E$ tel que $w \notin [C]$. ◀

La preuve fonctionne encore en supposant que $L \cap E$ n'est pas un langage algébrique non ambigu.

Démonstration. Il s'agit d'une simple adaptation de la preuve de [CFM12a].

Comme $L \subseteq G$, nous déduisons que $L \cap E \subseteq G \cap E$. Comme $G \cap E$ est algébrique non ambigu (par stabilité des langages algébriques non ambigus par intersection avec un langage régulier), mais $L \cap E$ n'est pas algébrique, l'inclusion est forcément stricte.

Il existe donc un mot w qui est dans $G \cap E$ mais pas dans $L \cap E$. Donc w est dans G mais pas dans L. Donc $\boldsymbol{\pi}_{\boldsymbol{kh}}(w) \notin C$. ■

Tout comme LCL_R, la classe LCL est définie par une contrainte très forte de devoir compter chaque lettre de la même façon au cours du mot. Ainsi :

Proposition 6.65 ($a^n b^m a^n b^m \notin \mathrm{LCL}$).
▶ Le langage $L = \{a^n b^m a^n b^m \mid n, m \in \mathbb{N}^*\}$ n'est pas dans LCL. ◀

Démonstration. Afin d'éviter un argument de pompage un peu lourd, nous utilisons la version à pile du Lemme 5.4 de [CFM12a], démontrée au Lemme 6.64, qui relègue en fait le pompage dans la preuve qu'un langage n'est pas algébrique. Supposons que $L = G \cap [C]$ avec G un langage algébrique non ambigu.

Soit E le langage régulier $(a^2)^+(b^2)^+(a^2)^+(b^2)^+$. Comme le langage $L \cap E = \{a^{2n}b^{2m}a^{2n}b^{2m} \mid n, m \in \mathbb{N}^*\}$ n'est classiquement pas algébrique, par le Lemme 6.64 il existe un mot $w \in E$ tel que $\boldsymbol{\pi_{kh}}(w) = (|w|_a, |w|_b) \notin C$.

C'est plutôt $a^nb^ma^nb^m$ qui n'est classiquement pas algébrique, mais la preuve s'adapte facilement.

Comme $w \in E$, son nombre de a et de b est pair. Le mot

$$w' = a^{|w|_a/2}b^{|w|_b/2}a^{|w|_a/2}a^{|w|_b/2}$$

est dans L, mais il a la même image de Parikh que w. Donc $\boldsymbol{\pi_{kh}}(w') \notin C$, autrement dit $w' \notin [C]$. Contradiction. ■

Il manque à LCL un moyen de compter différemment certains groupes de lettres ; une approche naturelle pour le faire est d'utiliser un morphisme. Nous généralisons alors la classe LCL, en transposant l'idée de Castiglione et Massazza pour la classe RCM :

Définition 6.66 (LCM).
▶ Soit Σ un alphabet non vide. Un langage $L \subseteq \Sigma^*$ appartient à la classe LCM s'il existe un alphabet Γ de taille $d \geqslant 1$, un langage algébrique non ambigu G sur l'alphabet Γ, un ensemble semilinéaire $C \subseteq \mathbb{N}^d$, et enfin un morphisme lettre à lettre $\mu : \Gamma^* \to \Sigma^*$ *injectif sur* $G \cap [C]$, tel que

$$L = \mu(G \cap [C])\,.$$

Autrement dit, L est l'image d'un langage L' de LCL par un morphisme lettre à lettre et injectif sur L'. On note alors $L = \langle G, C, \mu\rangle$. ◀

Exemple 6.67 (Morphisme et renommage des lettres).
▶ Le langage $L = \{a^nb^ma^nb^m \mid n, m \in \mathbb{N}^*\}$ appartient à la classe LCM. En effet, il est dans RCM, qui est inclus dans LCM. ◀

Proposition 6.68.
▶ Dans la définition de LCM, on peut supposer sans perte de généralité que le langage algébrique G est reconnu par un automate à pile déterministe sans ε-transition. ◀

Démonstration. Soit $L = \langle G, C, \mu\rangle \in$ LCM, où G est un langage reconnu par un automate à pile $\mathcal{A}$ non ambigu, C est un ensemble semilinéaire de dimension $|\Gamma|$ reconnu par une formule de Presburger Φ, et $\mu : \Gamma \to \Sigma$ est un morphisme lettre à lettre injectif sur $G \cap [C]$.

Sans perte de généralité, nous pouvons supposer que $\mathcal{A}$ est non ambigu, sans ε-transitions. En effet la procédure d'élimination des ε-transitions pour les automates à pile préserve la non-ambiguïté [Che94].

Ce sera aussi une conséquence de la section 6.5.

Le langage des calculs acceptants de $\mathcal{A}$ est facilement reconnu par un automate à pile déterministe $\mathcal{A}'$: l'alphabet d'entrée de $\mathcal{A}'$ est l'ensemble des transitions Δ de $\mathcal{A}$, et en lisant une transition de $\mathcal{A}$, l'automate correspondant effectue les actions décrites par la transition.

On définit alors $\mu' : \Delta \to \Sigma$, par $\mu'(t) = \mu(\alpha(t))$, où $\alpha(t)$ désigne la lettre de Γ qui étiquette la transition $t \in \Delta$ (on rappelle que $\mathcal{A}$ n'a pas d'ε-transitions, donc le morphisme est bien lettre à lettre).

Enfin, on définit $\Phi'(\boldsymbol{y})$ pour $\boldsymbol{y} = (y_t)_{t\in\Delta}$, la formule de Presburger obtenue à partir de $\Phi(\boldsymbol{x})$ en remplaçant chaque variable x_a avec $a \in \Gamma$ par $\sum_{t\in\Delta:\alpha(t)=a} y_t$.

On vérifie facilement que $\mu'(\mathcal{L}(\mathcal{A}') \cap [C']) = \mu(G \cap [C]) = L$. Il reste à démontrer que μ' est bien injectif sur $\mathcal{L}(\mathcal{A}') \cap [C']$. Soient π_1 et π_2 deux mots de $\mathcal{L}(\mathcal{A}') \cap [C']$

tels que $\mu'(\pi_1) = \mu'(\pi_2)$. Par définition de $\mathcal{A}'$, π_1 et π_2 représentent deux calculs acceptants de $\mathcal{A}$, étiquetés par deux mots w_1 et w_2 de G. Ainsi $\mu'(\pi_1) = \mu(w_1)$ et $\mu'(\pi_2) = \mu(w_2)$. Par construction de C', si $\pi_1 \in C'$, alors $w_1 \in C$. Par injectivité de μ sur $G \cap [C]$, on a ainsi $w_1 = w_2$. Donc π_1 et π_2 représentent deux calculs acceptants de $\mathcal{A}$ étiquetés par le même mot. Par non-ambiguïté de $\mathcal{A}$, $\pi_1 = \pi_2$. Donc μ' est bien injectif. ■

Nous pouvons désormais démontrer la proposition principale de cette section :

Théorème 6.69 (LCM = wuℙA).
▶ La classe LCM est égale à la classe des langages algébriques de Parikh faiblement non ambigus. ◀

Démonstration. Il s'agit de la même preuve que dans le cas sans pile, donc je détaille un peu moins le raisonnement.

De LCM **vers** wuℙA**.** Soit $L \in \text{LCM}$. On introduit $\Gamma = \{a_1 \dots a_d\}$ et Σ deux alphabets, G un langage sur Γ, reconnu par un automate à pile déterministe $\mathcal{A}$ par la Proposition 6.68, C un ensemble semilinéaire de $\mathbb{N}^d$ et finalement $\mu : \Gamma^* \longrightarrow \Sigma^*$ un morphisme lettre à lettre injectif sur $G \cap [C]$, tel que $L = \langle G, C, \mu \rangle$.

L'automate de Parikh à pile est $\mathcal{B} = (\mathcal{A}', C)$, où C est inchangé, et $\mathcal{A}'$ est obtenu à partir de $\mathcal{A}$ en remplaçant, dans toutes les transitions, la lettre $a_i \in \Gamma$ par le couple $(\mu(a_i), \boldsymbol{e_{a_i}})$, pour tout $i = 1 \dots d$.

Les calculs de $\mathcal{A}$ et de $\mathcal{B}$ sont clairement en bijection, et on vérifie comme dans le cas sans pile que $\mathcal{B}$ reconnaît le langage L et est faiblement non ambigu.

De wuℙA vers LCM. Soit $\mathcal{B} = (\mathcal{A}, C)$ un automate de Parikh à pile faiblement non ambigu de dimension d normalisé par la Proposition 6.57, c'est-à-dire qu'il est sans ε-transition, et ses transitions sont étiquetées par des vecteurs $\boldsymbol{e_i}$ qui identifient de façon unique la lettre avec laquelle ils étiquettent des transitions. Autrement dit toute transition étiquetée par un vecteur $\boldsymbol{e_i}$ est étiquetée par le couple $(a_i, \boldsymbol{e_i})$ où $a_i \in \Sigma$ est entièrement déterminée par i. De plus le semilinéaire d'acceptation est de la forme :

$$C = \left\{(n_1, \dots, n_r) \mid \bigwedge_{i=1}^{r} (\textstyle\sum_{j=1}^{r} a_{i,j} n_j = 0), \text{ avec } a_{i,j} \in \{-1, 0, 1\}\right\}.$$

On considère l'automate $\mathcal{A}'$ égal à $\mathcal{A}$, vu comme un automate à pile sur l'alphabet $\Gamma = \{(a_i, \boldsymbol{e_i})\}_{i \in [1,d]}$, où l'on concatène lettres et vecteurs. Le morphisme μ est défini par la projection sur la première coordonnée : $\mu((a, \boldsymbol{e_i})) = a$.

Soit G le langage algébrique reconnu par $\mathcal{A}'$. Comme pour la preuve sans pile, on vérifie que $\mathcal{L}(\mathcal{B}) = \mu(G \cap [C])$ et que μ est injectif sur $G \cap [C]$. ■

Corollaire 6.70 (Corollaire de la preuve).
▶ Tout langage dans LCM peut être défini par un triplet $\langle G, C, \mu \rangle$, où C est une conjonction de contraintes linéaires de la forme

$$\{(n_1, \dots, n_r) \mid \bigwedge_{i=1}^{r} (\textstyle\sum_{j=1}^{r} a_{i,j} n_j = 0), \text{ avec } a_{i,j} \in \{-1, 0, 1\}\}.$$

◀

6.5 Preuve de l'élimination des ε-transitions dans un automate de Parikh à pile faiblement non ambigu

Pour prouver l'équivalence entre les classes $\mathrm{wu}\mathbb{P}\mathrm{A}$ et la classe LCM, à la section précédente, nous avons eu besoin d'utiliser un automate de Parikh à pile normalisé sans ε-transitions. Le but de cette section est de prouver qu'il est possible d'éliminer les ε-transitions, tout en préservant la faible non-ambiguïté. Nous souhaitons donc démontrer la proposition suivante, admise à la section précédente :

Proposition 6.56 (Élimination des ε-transitions dans un $\mathrm{wu}\mathbb{P}\mathrm{A}$).
▶ Tout langage algébrique de Parikh faiblement non ambigu est reconnaissable par un automate de Parikh faiblement non ambigu qui ne possède aucune ε-transition (toutes les transitions lisent une lettre). ◀

L'élimination des ε-transitions dans un automate à pile classique consiste à traduire l'automate à pile en grammaire hors-contexte équivalente, puis mettre cette grammaire sous forme normale de Greibach, c'est-à-dire une grammaire dont les règles de dérivation sont toutes de la forme $S \to aV_1 \dots V_r$ avec a une lettre terminale quelconque, et $V_1, \dots, V_r$ des symboles non terminaux, avec $r \geqslant 0$. Enfin, cette grammaire sous forme normale de Greibach se traduit en un automate à pile équivalent, sans ε-transitions. Je n'ai pas connaissance de procédure qui permette d'éliminer les ε-transitions d'un automate à pile sans passer par le formalisme des grammaires hors-contexte.

Comme nous voulons éliminer les ε-transitions dans un modèle qui généralise les automates à pile classiques, nous introduisons une classe de grammaires hors-contexte pondérées par des contraintes semilinéaires, qui généralise les grammaires hors-contexte, et qui est équivalente aux automates de Parikh à pile. Nous adaptons les preuves classiques des grammaires hors-contexte à ces grammaires hors-contexte pondérées, tout en nous assurant que les constructions classiques préservent la faible non-ambiguïté.

Dans d'autres contextes, la forme normale de Greibach a été étudiée pour les grammaires pondérées, dans le cadre des demi-anneaux [BRW12, Sta72], ou dans le cas particulier des grammaires probabilistes [AMP99, Pon12]. La persistance de la non-ambiguïté lors de la mise sous forme normale de Greibach a été étudiée dans le cadre des grammaires hors-contexte dans [Che94].

6.5.1 Plan de la preuve

Pour éliminer les ε-transitions d'un automate de Parikh à pile, nous étudions le formalisme équivalent des *grammaires hors-contexte contraintes*, notées $\mathbb{C}\mathrm{FG}$ (pour constrained context-free grammar). Tout au long de la preuve, nous allons étudier la mise sous forme normale de Greibach d'une grammaire dont les poids sont des vecteurs ou des ensembles semilinéaires de $\mathbb{N}^d$.

a. Nous montrons dans un premier temps, à la Proposition 6.79 que tout langage algébrique de Parikh faiblement non ambigu est reconnu par une $\mathbb{C}\mathrm{FG}$ non ambiguë.

b. Ensuite, nous montrons à la Proposition 6.95 que nous pouvons mettre une $\mathbb{C}\mathrm{FG}$ non ambiguë sous forme normale de Greibach tout en préservant la non-ambiguïté. Cette mise sous forme normale se fait en trois étapes, exactement comme pour les grammaires hors-contexte classiques :

a) La première étape consiste à éliminer les ε-productions, c'est-à-dire toutes les règles de la forme $A \xrightarrow{L} \varepsilon$, pour obtenir une grammaire qui reconnaît le même langage privé du mot vide. Cette construction est faite à la Proposition 6.81.

b) Ensuite, nous éliminons les règles unité, qui sont les règles de dérivation de la forme $A \xrightarrow{L} B$ où A et B sont deux symboles non terminaux. Cette construction est faite à la Proposition 6.83.

c) Enfin nous éliminons la récursivité gauche ; il s'agit de la dernière étape, qui est la plus difficile, de la mise sous forme normale de Greibach. Pour garantir la préservation de la non-ambiguïté par cette transformation, qui est plus délicate que les deux précédentes, nous prenons une autre approche que celle généralement présentée pour les grammaires hors-contexte. Nous introduisons d'abord une transformation Φ définie sur des arbres, qui transforme les arbres de dérivation de la grammaire en éliminant en grande partie la récursivité gauche. Puis nous décrivons une grammaire G_Φ dont les arbres de dérivation coïncident avec l'image de Φ. Cette présentation de la mise sous forme normale de Greibach, en introduisant d'abord la transformation Φ sur les arbres de dérivation, a l'avantage d'expliciter la bijection, en l'occurrence Φ, entre les arbres de dérivation de la grammaire originale et ceux de la grammaire équivalente sous forme normale de Greibach. Grâce à cela, nous pouvons garantir rigoureusement que la grammaire obtenue est toujours non ambiguë.

c. Enfin, nous montrons dans la Proposition 6.96 comment une $\mathbb{C}$FG non ambiguë sous forme normale de Greibach peut être traduite en automate de Parikh à pile faiblement non ambigu, qui n'utilise pas d'ε-transitions.

6.5.2 Grammaires hors-contexte contraintes

Définition 6.71 (Grammaire hors-contexte contrainte)**.**
▶ Une *grammaire hors-contexte contrainte* de dimension d est un tuple $(N, \Sigma, S, D, \mathcal{S})$ où :

a. N est l'ensemble des non terminaux, Σ est l'ensembles des lettres terminales,

b. $S \in N$ est appelé l'axiome,

c. $\mathcal{S} \subseteq \mathbb{N}^d$ est un ensemble semilinéaire,

d. D est l'ensemble des règles de production de la grammaire, qui sont de la forme $A \xrightarrow{\boldsymbol{v}} A_1 \dots A_r$, où $A \in N$, $r \geqslant 0$, $\boldsymbol{v} \in \mathbb{N}^d$, et $A_i \in N \cup \Sigma$ pour tout $1 \leqslant i \leqslant r$.

◀

Intuitivement, une grammaire hors-contexte contrainte est une grammaire hors-contexte dont les règles de dérivation sont étiquetées par un vecteur. Un mot w est dans le langage écrit par la grammaire s'il existe un arbre de dérivation de frontière w, tel que la somme de tous les vecteurs utiliser pour dériver w appartient à un ensemble semilinéaire d'acceptation. Avant de définir proprement les arbres de dérivation pour ces grammaires, nous définissons une variante dont nous aurons besoin :

Définition 6.72 (Grammaire hors-contexte contrainte généralisée)**.**
▶ Une *grammaire hors-contexte contrainte généralisée* de dimension d est une grammaire dont les règles de production sont étiquetées par des ensembles semilinéaires de $\mathbb{N}^d$. ◀

Remarque 6.73 .
▶ Toute grammaire hors-contexte contrainte peut être vue comme généralisée, en remplaçant tout vecteur $\boldsymbol{v}$ qui étiquette une règle de dérivation par le semilinéaire $\{\boldsymbol{v}\}$. ◀

Nous définissons les arbres de dérivation des grammaires hors-contexte contraintes généralisées. Pour cela nous introduisons brièvement les arbres étiquetés par des ensembles semilinéaires. Nous fixons $d \in \mathbb{N}^*$ une dimension.

Définition 6.74 .
▶ Soit P un ensemble de symboles. L'ensemble $Tree_P$ est défini par induction :

— tout symbole $v \in P$ est un arbre de $Tree_P$;
— si $r \geqslant 1$, $v \in P$, L est un ensemble semilinéaire de dimension d, et $t_1, \ldots, t_r$ sont des arbres de $Tree_P$, alors $(v, L, t_1, \ldots, t_r) \in Tree_P$.

Autrement dit, il s'agit de l'ensemble des arbres plans dont les noeuds sont étiquetés par des symboles de P, et de plus chaque noeud interne est étiqueté par un ensemble semilinéaire de $\mathbb{N}^d$. ◀

Définition 6.75 .
▶ Nous définissons par induction les notions suivantes :

— pour tout $v \in P$, $root(v) = v$; et si $t = (v, L, t_1, \ldots, t_r)$, $root(t) = v$ désigne la racine de t.
— pour tout $v \in P$, $\mathcal{S}(v) = \{0\}$; et si $t = (v, L, t_1, \ldots, t_r)$, $\mathcal{S}(t) = L + \sum_{i=1}^{r} \mathcal{S}(t_i)$ désigne la somme des ensembles semilinéaires qui apparaissent dans t.
— pour tout $v \in P$, $left(v) = v$; et si $t = (v, L, t_1, \ldots, t_r)$, $left(t) = left(t_1)$ désigne le symbole le plus en bas à gauche de l'arbre t.
— pour tout $v \in P$, $Fr(v) = v$; et si $t = (v, L, t_1, \ldots, t_r)$, $Fr(t) = Fr(t_1) \ldots Fr(t_r)$ désigne la frontière de l'arbre t. Autrement dit il s'agit du mot obtenu en lisant les feuilles de t, dans l'ordre de gauche à droite.
— pour tout $v \in P$, $subtrees(v) = \{v\}$; et si $t = (v, L, t_1, \ldots, t_r)$, $subtrees(t) = \{t\} \cup subtrees(t_1) \cup \ldots \cup subtrees(t_r)$ désigne l'ensemble des sous-arbres de t.
— pour tout $v \in P$, $ht(v) = 0$; et si $t = (v, L, t_1, \ldots, t_r)$, $ht(t)$ désigne la hauteur de l'arbre t, définie par $ht(t) = 1 + \max(ht(t_1), \ldots, ht(t_r))$.

◀

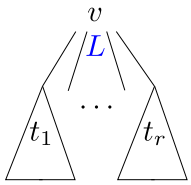


Figure 6.1. Représentation d'un arbre dont la racine v possède r fils, et est étiquetée par l'ensemble semilinéaire L

Définition 6.76 (Arbre de dérivation).
▶ Soit $G = (N, \Sigma, S, D, \mathcal{S})$ une grammaire hors-contexte contrainte généralisée. Pour tout $v \in N \cup \Sigma$, l'ensemble des arbres de dérivation de G enracinés en v, noté $Tree_G(v)$, est défini par induction :

— pour $v \in N \cup \Sigma$, $v \in Tree_G(v)$;

— si $v \in N$ et $v \xrightarrow{L} v_1 \dots v_r \in D$ est une règle de production, et si $t_i \in Tree_G(v_i)$ pour tout $1 \leqslant i \leqslant r$, alors $(v, L, t_1, \dots, t_r) \in Tree_G(v)$.

Nous notons $Tree_G = \bigcup_{v \in N \cup \Sigma} Tree_G(v)$ l'ensemble des arbres de dérivation de G. Enfin, un arbre de dérivation est appelé *terminal* si sa frontière est dans Σ^*. ◀

Définition 6.77 (Langage d'une grammaire, non-ambiguïté).
▶ Un mot $w \in \Sigma^*$ est généré par une grammaire contrainte généralisée G s'il existe un arbre de dérivation $t \in Tree_G(S)$ tel que $Fr(t) = w$ et tel que la somme des ensembles linéaires apparaissant dans l'arbre de dérivation t contient un vecteur dans $\mathcal{S}$ (autrement dit $\mathcal{S}(t) \cap \mathcal{S} \neq \emptyset$). Dans le cas particulier des grammaires contraintes étiquetées par des vecteurs, cette condition est équivalente au fait que la somme des vecteurs qui apparaissent dans l'arbre de dérivation t appartient au semilinéaire $\mathcal{S}$.

Une $\mathbb{C}$FG (généralisée) est *non ambiguë* si pour tout mot w il existe au plus un arbre de dérivation de G générant le mot w. ◀

Définition 6.78 (Dérivation gauche).
▶ Une dérivation gauche d'un mot $w \in (N \cup \Sigma)^*$ à partir d'un symbole $V \in N$ est une séquence de dérivations successives $w_1 \xrightarrow{r_1} \dots \xrightarrow{r_{n-1}} w_n$, où $w_1 = V$, $w_n = w$, et pour tout $i \in [n]$, $w_i \in (\Sigma \cup N)^*$ et w_{i+1} est obtenu à partir de w_i en appliquant une règle de dérivation $r_i \in D$ au symbole non terminal le plus à gauche de w_i.

Une dérivation est dite *terminale* si le dernier mot w_n de sa séquence ne contient aucun symbole non terminal, autrement dit si $w_n \in \Sigma^*$.

Pour tout symbole $V \in N$, nous définissons $D_{left}(V)$ l'ensemble des dérivations gauches commençant par V. On démontre facilement par induction que l'ensemble des dérivations gauches terminales commençant par V est en bijection avec l'ensemble des arbres de dérivation terminaux de G enracinés en V. ◀

La proposition suivante permet de traduire un automate de Parikh à pile dans le formalisme des grammaires hors-contexte contraintes :

Proposition 6.79 .
▶ Tout langage algébrique de Parikh faiblement non ambigu est reconnaissable par une $\mathbb{C}$FG non ambiguë. ◀

Démonstration. Nous adaptons la méthode classique des triplets de Ginsburg. Nous considérons $\mathcal{B} = (Q, q_I, F, \Sigma, \Gamma, \bot, \delta, \mathcal{S})$ un automate de Parikh à pile de dimension d, faiblement non ambigu, où Q désigne l'ensemble des états, q_I est l'état initial, F l'ensemble des états finaux, Σ l'alphabet, Γ l'ensemble des symboles de pile, $\bot \in \Gamma$ est le symbole de fond de pile, δ l'ensemble des transitions, et $\mathcal{S}$ est l'ensemble semilinéaire d'acceptation.

Il n'est pas difficile de transformer $\mathcal{B}$ en un automate équivalent qui vide sa pile avant d'accepter un mot : il suffit d'ajouter deux états $q_\emptyset$ et q_f à Q, et de remplacer les états finaux de F par simplement $F' = \{q_f\}$. L'automate commence par remplacer $\bot$ au fond de la pile par $\$\bot$, avec $\$ \notin \Gamma$ un nouveau symbole, puis il simule $\mathcal{B}$. À chaque fois que l'automate se retrouve dans un état originellement final $q \in F$, il choisit de façon non déterministe d'aller dans l'état $q_\emptyset$ par une $(\varepsilon, \mathbf{0})$-transition. Dans l'état $q_\emptyset$, l'automate ne lit plus aucune lettre ; il vide sa pile avec des $(\varepsilon, \mathbf{0})$-transitions, jusqu'à dépiler le symbole $\$$; il va ensuite dans l'état q_f, qui n'a aucune transition

sortante. Le fait que l'automate ainsi transformé reconnaisse le même langage, et que la transformation préserve la faible non-ambiguïté, est immédiat.

Sans perte de généralité, nous pouvons aussi supposer que les transitions de $\mathcal{B}$ n'ajoutent sur la pile qu'au plus un symbole, quitte à ajouter des états intermédiaires et des ε-transitions.

En conclusion, nous supposons que $\mathcal{B}$ ajoute sur sa pile au plus un symbole, et accepte avec une pile vide dans un unique état final.

La transformation suivante est la méthode classique des triplets pour transformer un automate à pile en grammaire équivalente, adaptée dans le cas des grammaires pondérées. Nous définissons la grammaire contrainte $G_{\mathcal{B}} = (N, \Sigma, S, D, \mathcal{S})$ où $N \subseteq \{S\} \cup \{[q, q', z] : q, q' \in Q, z \in \Gamma \cup \{\varepsilon\}\}$, et D est défini par :

- $[q, q', z] \xrightarrow{\boldsymbol{d}} y$ si $(q, z \xrightarrow{(y,\boldsymbol{d})} q', \varepsilon) \in \delta$
- $[q, q', z] \xrightarrow{\boldsymbol{d}} y[q'', q', z_1]$ si $(q, z \xrightarrow{(y,\boldsymbol{d})} q'', z_1) \in \delta$
- $[q, q', z] \xrightarrow{\boldsymbol{d}} y[q_1, q_2, z_1][q_2, q', z_2]$, pour tout $q_2 \in Q$, si $(q, z \xrightarrow{(y,\boldsymbol{d})} q_1, z_1 z_2) \in \delta$
- $S \xrightarrow{\mathbf{0}} [q_I, q_F, \bot]$ pour tout $q_F \in F$

Il est classique de démontrer par induction que pour tout toute dérivation gauche à partir du symbole $[q, q', z]$, de la forme $[q, q', z] \xrightarrow{r_1} w_2 \ldots \xrightarrow{r_{n-1}} w_n = w$, on peut associer un calcul de $\mathcal{B}$ qui commence dans l'état q, avec comme symbole de sommet de pile z, et qui dépile pour la première fois ce symbole en arrivant dans l'état q', après avoir lu exactement le mot w, avec comme vecteur somme associé $r_1 + \ldots + r_n$. De plus, cette correspondance est bijective, dans le sens où les dérivations gauches simulent exactement les calculs de l'automate, et rien d'autre.

Ainsi, les dérivations gauches de la grammaire sont capables de dérouler les calculs de l'automate de Parikh à pile, et chaque dérivation gauche terminale de la grammaire simule exactement un calcul valide de l'automate de Parikh à pile, étiqueté par le même mot et le même poids. Par conséquent, la non-ambiguïté est préservée durant la transformation. ■

Par la suite, les $\mathbb{C}$FG non ambiguës $G = (N, \Sigma, S, D, \mathcal{S})$ considérées seront supposées issues de la construction précédente, et donc vérifier sans perte de généralité les conditions (H) :

a. tout symbole non terminal est productif (quitte à supprimer les symboles non productifs) ;

b. chaque règle de dérivation est de la forme $A \to w$ où $w \in \Sigma^i N^j$, avec $i \in \{0, 1\}$ et $j \in \{0, 1, 2\}$;

c. l'axiome S n'est jamais produit par une règle de dérivation de la grammaire.

Nous cherchons à modifier une grammaire de cette forme de manière à obtenir une grammaire non ambiguë équivalente, telle que toute règle de production commence par une lettre de Σ (c'est la forme normale de Greibach). Nous suivons les étapes classiques de la mise sous forme normale de Greibach dans le cadre des grammaires classiques.

6.5.3 Élimination des règles d'effacement

Un symbole $A \in N \cup \{\varepsilon\}$ d'une grammaire contrainte est dit effaçable si $A \neq S$ et si $A \to^* \varepsilon$.

Nullable en anglais.

Le but de cette section est de supprimer d'une grammaire contrainte les dérivations qui effacent un symbole effaçable.

Si A est effaçable, nous notons T_A l'ensemble des vecteurs qui peuvent étiqueter une dérivation $A \to^* \varepsilon$ qui l'efface.

Lemme 6.80.
▶ T_A est semilinéaire. ◀

Démonstration. Nous construisons une grammaire hors-contexte $G' = (N, \Sigma', A, D')$ à partir de G sur l'alphabet $\Sigma' = \{a_1, \ldots, a_d\}$, qui reconnaît un langage dont l'image de Parikh est T_A.

La grammaire G' se construit à partir de G en prenant comme axiome le symbole A, et en supprimant de D toute règle de dérivation qui produit une lettre de Σ. Enfin, les poids sont encodés dans la grammaire hors-contexte : chaque règle de dérivation de la forme $B \xrightarrow{\boldsymbol{v}} B_1 \ldots B_r$ est remplacée par la règle $B \to a_1^{v_1} \ldots a_d^{v_d} B_1 \ldots B_r$ où $r \geqslant 0$, $B_i \in N$ et $1 \leqslant i \leqslant r$. Une dérivation gauche terminale de G' simule exactement une dérivation gauche de G commençant en A et produisant le mot vide ε, et réciproquement toute dérivation gauche de G effaçant A se transforme en une seule dérivation gauche terminale de G'. De plus l'image de Parikh du mot reconnu par un arbre de dérivation terminal de G' est égale au vecteur étiquetant la dérivation gauche de ε correspondante dans G. Par le théorème de Parikh, l'image de Parikh de G' est semilinéaire, et donc par la correspondance, T_A est semilinéaire. ■

Proposition 6.81.
▶ Soit $G = (N, \Sigma, S, D, \mathcal{S})$ une $\mathbb{C}$FG non ambiguë, vérifiant les conditions (H). Alors nous pouvons construire une $\mathbb{C}$FG G' généralisée non ambiguë, sans symbole effaçable, qui vérifie toujours la condition (H), et telle que $L(G') = L(G) \backslash \{\varepsilon\}$. ◀

Démonstration. Nous adaptons les idées classiques issues des grammaires sans poids. Nous notons $G' = (N, \Sigma, S, D', \mathcal{S})$ la $\mathbb{C}$FG obtenue à partir de G en modifiant uniquement les règles de dérivation comme suit :

- Pour toute règle t_i de D de la forme $t_i = A \xrightarrow{\boldsymbol{d_i}} \alpha$ avec $\alpha \in \Sigma$ (*i.e.* $\alpha \neq \varepsilon$), nous ajoutons à D' la règle $t_i^1 = A \xrightarrow{\{\boldsymbol{d_i}\}} \alpha$.
- Pour toute règle t_i de D de la forme $t_i = A \xrightarrow{\boldsymbol{d_i}} \alpha B$ avec $\alpha \in \Sigma \cup \{\varepsilon\}$ et $B \in N$, nous ajoutons les règles suivantes à D' :
  - $t_i^1 = A \xrightarrow{\{\boldsymbol{d_i}\}} \alpha B$;
  - $t_i^2 = A \xrightarrow{\{\boldsymbol{d_i}+T_B\}} \alpha$, si B est effaçable dans G et $\alpha \neq \varepsilon$.
- Pour toute règle t_i de D de la forme $t_i = A \xrightarrow{\boldsymbol{d_i}} \alpha BC$, avec $B, C \in N$, et $\alpha \in \Sigma \cup \{\varepsilon\}$, nous ajoutons les règles suivantes à D' :
  - $t_i^1 = A \xrightarrow{\{\boldsymbol{d_i}\}} \alpha BC$;
  - $t_i^2 = A \xrightarrow{\{\boldsymbol{d_i}\}+T_B} \alpha C$, si B est effaçable ;
  - $t_i^3 = A \xrightarrow{\{\boldsymbol{d_i}\}+T_C} \alpha B$, si C est effaçable ;
  - $t_i^4 = A \xrightarrow{\{\boldsymbol{d_i}\}+T_B+T_C} \alpha$, si B et C sont effaçables et $\alpha \neq \varepsilon$ (autrement cela introduirait un symbole effaçable).

Par construction, G' n'a aucune symbole effaçable, et vérifie toujours la condition (H).

Tout arbre de dérivation terminal T pour un mot $w \in \Sigma^*$ de G se transforme facilement en un arbre de dérivation T' pour w dans G' : il est obtenu en effaçant tous les sous-arbres maximaux de T de frontière ε, en remplaçant les règles de dérivation utilisées au niveau des parents de ces sous-arbres maximaux par les règles de G' qui simulent leur suppression. Il est immédiat que si T est étiqueté par un vecteur $\boldsymbol{d}$, alors T' est étiqueté par un ensemble semilinéaire L tel que $\boldsymbol{d} \in L$, si bien que si $w \in L(G)$ alors $w \in L(G')$.

Réciproquement, à partir d'un arbre de dérivation terminale T' dans G' d'un mot $w \in \Sigma^*$, étiquetée par un ensemble semilinéaire L, nous pouvons recréer un arbre de dérivation terminal T de w dans G, étiqueté par un vecteur $\boldsymbol{d}$ tel que $\boldsymbol{d} \in L$: il suffit de remplacer les règles de la forme t_i^r par la règle t_i de G dont elles proviennent, et de dériver, à partir des symboles B effaçables qui apparaissent dans t_i mais pas dans t_i^r, des arbres de frontière ε (qui sont par définition étiquetées par un vecteur de T_B). Plus précisément, si nous supposons que T' est un arbre de dérivation acceptant le mot w, alors il existe un vecteur $\boldsymbol{d} \in L \cap \mathcal{S}$. Par définition, L est la somme des ensembles semilinéaires qui apparaissent dans les règles de dérivation de l'arbre. Donc nous pouvons retrouver, à partir d'une décomposition de $\boldsymbol{d}$ dans cette somme de semilinéaires, des valeurs pour les vecteurs étiquetant les arbres de frontière ε que l'on dérive lors de la construction de T à partir de T', de telle sorte que T est étiqueté par le vecteur somme $\boldsymbol{d} \in \mathcal{S}$. Ce qui prouve que $w \in L(G)$.

Enfin, G' est non ambiguë : par la construction ci-dessus, deux arbre de dérivation terminaux différents de G' acceptant un même mot impliqueraient l'existence de deux arbres de dérivation différents pour ce mot dans G, qui est non ambiguë. ■

6.5.4 Élimination des règles unité

Soit $G = (N, \Sigma, S, D, \mathcal{S})$ une $\mathbb{C}$FG non ambiguë de d, vérifiant les conditions (H), sans symbole effaçable.

Une règle unité est une règle de la forme $A \to B$, où A et B sont des symboles non terminaux.

Pour tout couple de symbole non terminal $(A, B) \in N^2$, nous notons $T_{(A,B)}$ l'union de tous les ensembles semilinéaires qui étiquettent une dérivation dans G de la forme $A \xrightarrow{unit^+} B$, où $\xrightarrow{unit^+}$ signifie que la dérivation n'utilise que des règles unité et n'est pas triviale.

Lemme 6.82 .
▶ L'ensemble $T_{A,B}$ est semilinéaire. ◀

Démonstration. L'ensemble $T_{A,B}$ est reconnu par l'automate à vecteurs de dimension d suivant : l'ensemble d'états est N, avec des états supplémentaires décrits ci-après, l'état initial est A, et l'état final est B. Pour toute règle de G de la forme $B \xrightarrow{L} C$, nous relions B par une $\mathbf{0}$-transition à un automate reconnaissant L, puis nous reconnectons l'état final de cet automate à l'état C par une $\mathbf{0}$-transition. L'ensemble des vecteurs reconnus par cet automate est $T_{A,B}$, qui est donc semilinéaire. ■

Proposition 6.83 .
▶ Soit $G = (N, \Sigma, S, D, \mathcal{S})$ une $\mathbb{C}$FG non ambiguë de dimension d , vérifiant

les conditions (H), sans symbole effaçable (en particulier $\varepsilon \notin L(G)$). Alors nous pouvons construire une grammaire contrainte généralisée G' non ambiguë, sans symbole effaçable ni règle unité. ◀

C'est finalement la même technique que celle utilisée pour l'élimination des ε-transitions dans un automate de Parikh.

Démonstration. Nous adaptons les idées des grammaires hors-contexte classiques. Nous posons $G' = (N, \Sigma, S, D', \mathcal{S})$, où cette fois encore la seule différence vient des règles de production D' définies par :

- D' contient toutes les règles de D qui ne sont pas des règles unité.
- Pour tout symbole non terminal, pour toute règle non unité $t_i = B \xrightarrow{L_i} \beta$ avec $\beta \in (N \cup \Sigma)^+$ et $\beta \notin N$, et pour tout symbole $A \in N$ tel que $T_{(A,B)} \neq \emptyset$, alors D' contient la règle $t_i^A = A \xrightarrow{L_i + T_{(A,B)}} \beta$.

Il est immédiat que tout arbre de dérivation T' de G' est obtenu à partir d'un arbre de dérivation T de G obtenu en contractant ses chaînes maximales de règles unité et en les fusionnant avec la première règle de dérivation non unité qui termine la chaîne. Par définition, le semilinéaire de la forme $T_{(A,B)}$ ajouté aux transitions de D' inclut toutes les sommes possibles de semilinéaires étiquetant ces chaînes supprimées de règles unité. Le raisonnement pour les symboles effaçables se transpose ensuite facilement pour prouver que $L(G) = L(G')$ et que G' est non ambigu. ■

6.5.5 Mise sous forme normale de Greibach

La mise sous forme normale de Greibach est plus compliquée, car les transformations en jeu réarrangent complètement les arbres de dérivation ; il ne s'agit plus simplement de contracter des chemins ou des arbres de frontière ε. La préservation de la non-ambiguïté est ainsi plus délicate à étudier. Pour simplifier l'étude de la preuve tout en restant rigoureux, nous introduisons une variante de la mise sous forme normale classique : nous choisissons notamment d'abord de considérer une transformation Φ qui transforme les arbres de dérivation terminaux de G en des arbres dont tous les noeuds internes ont une lettre Σ à profondeur 1 ou 2 sur leur gauche. Nous prouvons que Φ est bijective sur un sous-ensemble d'arbres de dérivation bien identifiés, appelés contextes, et préserve la frontière et le semilinéaire étiquetant les arbres de dérivation de G. Puis nous introduisons une grammaire contrainte G' dont les arbres de dérivation coïncident avec l'image par Φ des arbres de dérivation de G. La grammaire reconnaît ainsi le même langage que G et, à une petite modification près, est sous forme normale de Greibach.

La transformation classique élimine la récursivité gauche par itérations successives, symbole par symbole. Cela simplifie la construction de la grammaire associée, mais complexifie les preuves s'intéressant aux arbres de dérivation, car il faut trouver les bons invariants de boucle sur la forme des arbres de dérivation des grammaires intermédiaires.

Dans la suite de cette section, nous fixons une dimension $d \in \mathbb{N}^*$.

Définition 6.84 (V-contextes).

▶ Soit $G = (N, \Sigma, S, D, \mathcal{S})$ une grammaire hors-contexte contrainte généralisée.

- Un arbre de dérivation de $C \in Tree_G$ est appelé un V-contexte pour $V \in N \cup \Sigma$ si $left(t) = V$ et si $Fr(t) \in V\Sigma^*$. Autrement dit le symbole le plus en bas à gauche de l'arbre est V, et c'est le seul symbole possiblement non terminal de t.
- Si $C \in Tree_G$ est un V-contexte avec $V \in N$, et $t \in Tree_G(V)$ est un arbre de dérivation de racine V, nous écrivons $C[t] \in Tree_G$ l'arbre de dérivation défini par :
  - $C[t] = t$ si $C = V$
  - $C[t] = (v, L, t_1[t], t_2, \ldots, t_r)$ si $C = (v, L, t_1, \ldots, t_r)$

Dans ce cas, t_1 est aussi un V-contexte, donc $C[t]$ est bien définie.

Autrement dit, $C[t]$ représente l'arbre obtenu en remplaçant le symbole $V \in N$ dans C par l'arbre t. On remarque que $C[t]$ est un $left(t)$-contexte si C est un V-contexte avec $V \in N$.

— Nous notons $Context_G[V]$ l'ensemble de tous les V-contextes, pour $V \in N \cup \Sigma$, et plus généralement, pour tout ensemble de symboles A, $Context_G[A]$ désigne l'ensemble de tous les V-contextes tels que $V \in A$ (en pratique nous n'utiliserons que $A = N$, Σ ou $N \cup \Sigma$).

— Pour $P, V \in N \cup \Sigma$, la notation $Context_G^P[V]$ désigne l'ensemble des V-contextes de racine P, et $Context_G^P[\Sigma]$ l'ensemble des arbres de dérivations terminaux de racine P. ◀

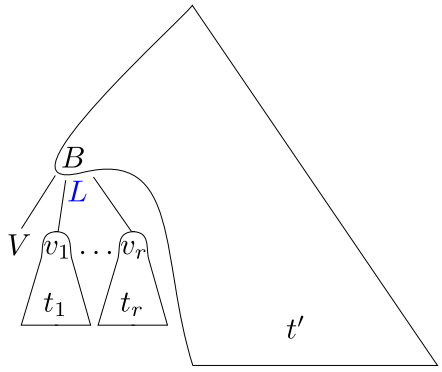


Figure 6.2. Décomposition d'un V-contexte sous la forme $t = t'[(B, L, V, t_1, \ldots, t_r)]$

Dans la suite, $G = (N, \Sigma, S, D, \mathcal{S})$ désigne une $\mathbb{C}$FG non ambiguë, sans symbole effaçable ni règle unité.

Proposition 6.85.
▶ Soit $V \in N \cup \Sigma$ et t un V-contexte de hauteur plus grande que 1. Alors il existe une règle de production de la forme $B \xrightarrow{L} V v_1 \ldots v_r$ avec $r \geqslant 0$, des arbres de dérivation $t_i \in Tree_G(v_i)$ pour $i \in [r]$, et un B-contexte t', tels que t se décompose sous la forme $t = t'[(B, L, V, t_1, \ldots, t_r)]$. Le choix de cette décomposition est par ailleurs unique. De plus :

— $r \geqslant 1$ si $V \in N$, puisque G est sans règle unité

— si $r \geqslant 1$, alors pour tout $1 \leqslant i \leqslant r$, $Fr(t_i) \in \Sigma^+$. En particulier $left(t_i) \in \Sigma$.

◀

Démonstration. La preuve est immédiate par induction. ■

Comme annoncé, nous définissons une application Φ qui transforme les contextes G de telle sorte que tout noeud interne possède une feuille $a \in \Sigma$ à profondeur 1 ou 2 sur sa gauche.

Nous invitons le lecteur à regarder les Figures 6.3 à 6.6 pour visualiser la définition de Φ.

Nous notons $N' := \Sigma \cup \{\langle V \rangle \mid V \in N\} \cup \{\langle V, B \rangle \mid V, B \in N\}$ un ensemble de nouveaux symboles construits à partir de $\Sigma \cup N$.

Définition 6.86 (Définition de l'application Φ).
▶ L'application $\Phi : Context_G[\Sigma \cup N] \to Tree_{N'}$ est définie par induction :

$\Phi(V) = \langle V \rangle$ sert uniquement à définir Φ sur les V-contextes réduits à $V \in N$. En effet la valeur de Φ sur N ne sert pas pour définir Φ sur les contextes de hauteur plus grande que 1.

- pour tout $a \in \Sigma$ et $V \in N$, $\Phi(a) = a$, et $\Phi(V) = \langle V \rangle$;
- si $ht(t) \geqslant 1$, par la Proposition 6.85, t se décompose de façon unique sous la forme $t = t'[(B, L, V, t_1, \ldots, t_r)]$, avec $r \geqslant 0$, $B \xrightarrow{L} V v_1 \ldots v_r \in D$, $t_i \in Tree_G(v_i)$, et t' qui est un B-contexte. Alors :
  - si $a := left(t) \in \Sigma$, $\Phi(t) = (\langle A \rangle, L, a, \Phi(t_1), \ldots, \Phi(t_r), \Phi(t'))$, où nous avons noté $A := root(t')$. Dans le cas où t' est réduit à $B \in N$, nous omettons le dernier fils $\Phi(t')$ dans $\Phi(t)$.
  - si $V := left(t) \in N$, $\Phi(t) = (\langle A, V \rangle, L, \Phi(t_1), \ldots, \Phi(t_r), \Phi(t'))$, où nous avons noté $A := root(t')$. Là encore, si t' est réduit à $B \in N$, nous omettons le dernier fils $\Phi(t')$ dans $\Phi(t)$. Nous rappelons que si $left(t) \in N$, alors $r \geqslant 1$.

◀

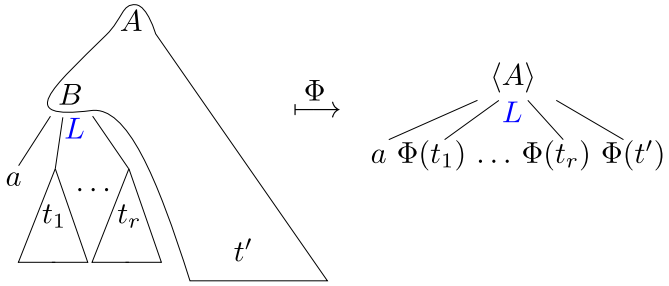


Figure 6.3. Action de Φ sur un a-contexte t, avec $left(t) = a \in \Sigma$, et t' n'est pas réduit à sa racine.

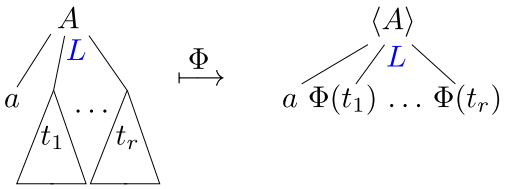


Figure 6.4. Action de Φ sur un a-contexte, avec $a \in \Sigma$, et t' est réduit à sa racine $A \in N$.

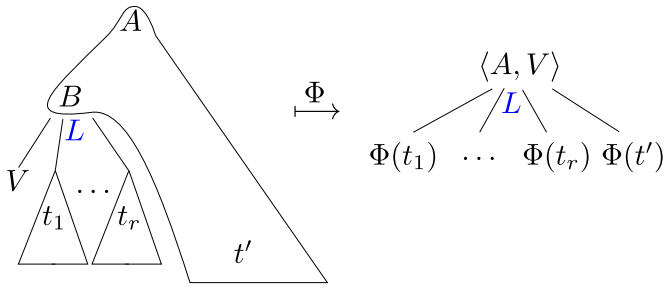


Figure 6.5. Action de Φ sur un V-contexte, avec $V \in N$, et t' n'est pas réduit à sa racine.

Nous vérifions simplement la proposition suivante par induction :

Proposition 6.87 .
▶ Si $t \in Context_G[\Sigma \cup N]$, $t_\Sigma \in Context_G[\Sigma]$, et $t_N \in Context_G[N]$, alors :

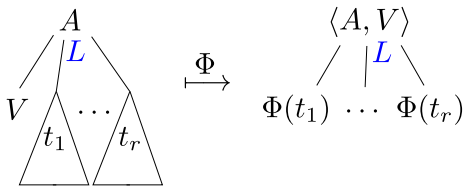


Figure 6.6. Action de Φ sur un V-contexte, avec $V \in N$, et t' est réduit à sa racine $A \in N$.

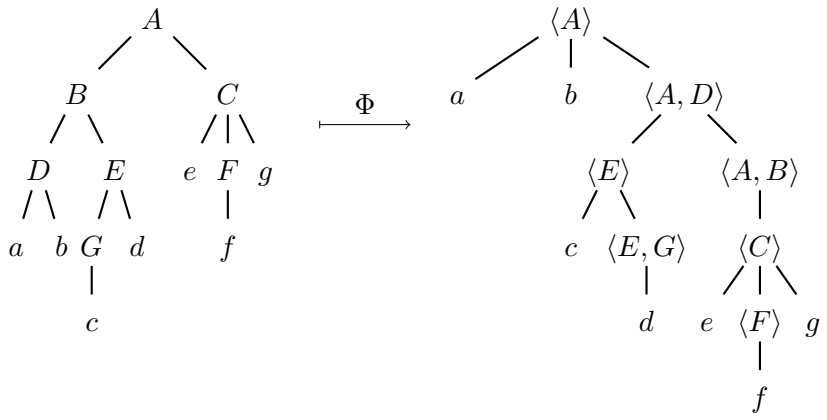


Figure 6.7. Exemple de transformation d'arbre par la fonction Φ. Pour plus de lisibilité, nous n'avons pas mis de poids sur les noeuds.

- $|\Phi(t_\Sigma)| = |t_\Sigma|$ et $|\Phi(t_N)| = |t_N| - 1$ où $|t|$ représente le nombre de noeuds dans l'arbre t.
- $\Phi(t)$ et t ont la même hauteur, et sont étiquetés par les mêmes ensembles semilinéaires. Ainsi $\mathcal{S}(\Phi(t)) = \mathcal{S}(t)$, et $ht(\Phi(t)) = ht(t)$.
- $Fr(\Phi(t_\Sigma)) = Fr(t_\Sigma)$, et $Fr(\Phi(t_N)) = left(t_N)^{-1} Fr(t_N)$ si $ht(t_N) \geqslant 1$.
- si t' est un sous-arbre de $\Phi(t)$, alors t' est l'image d'un sous-arbre ou d'un sous-contexte de t par Φ. ◀

Démonstration. Par induction immédiate d'après la définition de Φ. ■

Proposition 6.88.
▶ Φ est injective sur $Context_G[N \cup \Sigma]$. ◀

Démonstration. Nous le démontrons par induction sur la hauteur des arbres de $Context_G[N \cup \Sigma]$. Comme Φ préserve la hauteur d'un arbre, il suffit de vérifier l'injectivité sur des arbres de même hauteur.

- Par définition, Φ est trivialement injective sur les arbres réduits à un noeud.
- Soient $t_1, t_2 \in Context_G[N \cup \Sigma]$ de même hauteur plus grande que 1, tels que $\Phi(t_1) = \Phi(t_2)$. Par la Proposition 6.85, t_1 peut se décomposer sous la forme $t_1 = t'_1[(B_1, L_1, v_1, T_1, \ldots, T_r)]$, et de même $t_2 = t'_2[(B_2, L_2, v_2, T'_1, \ldots, T'_R)]$. Comme $\Phi(t_1) = \Phi(t_2)$, automatiquement $L_1 = L_2$, et t'_1 et t'_2 ont la même racine. Si cette racine est de la forme $\langle A \rangle$, alors t'_1 et t'_2 ont pour racine A, par définition de Φ . Par ailleurs, le premier fils de $\Phi(t_1)$ est une lettre $a \in \Sigma$. Par construction, cela signifie que $a = v_1 = v_2 \in \Sigma$. Sinon, la racine de $\phi(t_1) = \phi(t_2)$ est par

construction de la forme $\langle root(t_1), left(t_1)\rangle = \langle root(t_2), left(t_2)\rangle$. Dans tous les cas, nous pouvons affirmer que $left(t_1) = left(t_2)$ et $root(t_1) = root(t_2)$.

Montrons que $t'_1 = B_1 \in N$ si et seulement si $t'_2 = B_2 \in N$. Sans perte de généralité, supposons que $t'_1 = B_1$ mais $t'_2 \neq B_2$. Comme $t'_2 \neq B_2$, cela signifie que le fils le plus à droite de $\Phi(t_2)$ est égal à $\Phi(t'_2)$. Le fils le plus à droite de $\phi(t_1)$ est de la forme $\phi(t_\Sigma)$, où $t_\Sigma = T_r$ si $r \geqslant 1$, et $t_\Sigma = v_1 \in \Sigma$ si $r = 0$. Comme $\Phi(t_1) = \Phi(t_2)$, on en déduit que $\Phi(t_\Sigma) = \Phi(t'_2)$, et par hypothèse d'induction, $t_\Sigma = t'_2$. Or c'est impossible car $Fr(t_\Sigma) \in \Sigma^+$ tandis que $Fr(t'_2) \in N\Sigma^*$.

Donc $r = R$, et $\Phi(T_i) = \Phi(T'_i)$ pour tout $1 \leqslant i \leqslant r$ par hypothèse d'induction. De plus soit $t'_1 = B_1 = root(t_1)$ et $t'_2 = B_2 = root(t_2)$ et donc $t'_1 = t'_2$, soit $t'_1 \neq B_1$ et $t'_2 \neq B_2$, et donc $\Phi(t'_1) = \Phi(t'_2)$, et par conséquent par hypothèse d'induction $t'_1 = t'_2$.

Donc $t_1 = t_2$. ■

Remarque 6.89 .

▶ Φ étant injective sur $Context_G[\Sigma \cup N]$, elle réalise une bijection sur son image : $\Phi : Context_G[\Sigma \cup N] \to \Phi(Context_G[\Sigma \cup N])$ est une bijection. ◀

Nous cherchons maintenant à construire à partir de G une grammaire hors-contexte contrainte donc les arbres de dérivation coïncident avec $\Phi(Context_G[\Sigma \cup N])$.

Définition 6.90 .

▶ Nous définissons la $\mathbb{C}$FG $G_\Phi = (N', \Sigma, \langle S\rangle, D', \mathcal{S})$, où pour rappel $N' = \{\langle V\rangle \ : \ V \in N\} \cup \{\langle V, V'\rangle \ : \ V, V' \in N\}$, et D' est construit comme suit :

— Pour toute règle de dérivation de D de la forme $Q \xrightarrow{L} aP_1 \dots P_r$, telle que $a \in \Sigma$, et $P_i \in \Sigma \cup N$ pour $i \in [r]$, nous ajoutons dans D' la règle de dérivation

$$\langle Q\rangle \xrightarrow{L} a\langle P_1\rangle \dots \langle P_r\rangle,$$

où pour simplifier l'écriture, nous prenons comme convention que $\langle b\rangle = b$ lorsque $b \in \Sigma$. De plus pour tout $P \in N$, nous ajoutons à D' la règle de dérivation :

$$\langle P\rangle \xrightarrow{L} a\langle P_1\rangle \dots \langle P_r\rangle\langle P, Q\rangle\,.$$

— Pour tout règle de dérivation de D de la forme $Q \xrightarrow{L} RP_1 \dots P_r$, telle que $R \in N$, $r \geqslant 1$, et $P_i \in \Sigma \cup N$ pour $i \in [r]$, nous ajoutons à D' la règle de dérivation

$$\langle Q, R\rangle \xrightarrow{L} \langle P_1\rangle \dots \langle P_r\rangle,$$

et pour tout $P \in N$,nous ajoutons à D' la règle de dérivation :

$$\langle P, R\rangle \xrightarrow{L} \langle P_1\rangle \dots \langle P_r\rangle\langle P, Q\rangle\,.$$

◀

Exemple 6.91 .

▶ Considérons la grammaire $G = (N, \Sigma, S_a, D, \mathcal{S})$ définie par $N = \{S_a, S_b\}$, $a, b \in \Sigma$, et D est composé des deux règles $S_a \xrightarrow{(1,0)} a|S_b a$ et $S_b \xrightarrow{(0,1)} b|S_a b$. Alors $G_\Phi =$

$(N', \Sigma, \langle S_a\rangle, D', \mathcal{S})$ est définie par $N' = \{\langle S_a\rangle, \langle S_b\rangle, \langle S_a, S_b\rangle, \langle S_a, S_a\rangle, \langle S_b, S_b\rangle, \langle S_b, S_a\rangle\}$, et D' par :

$$\langle S_a\rangle \xrightarrow{(1,0)} a|a\langle S_a, S_a\rangle \qquad \langle S_a, S_b\rangle \xrightarrow{(1,0)} a|a\langle S_a, S_a\rangle$$
$$\langle S_a\rangle \xrightarrow{(0,1)} b\langle S_a, S_b\rangle \qquad \langle S_a, S_a\rangle \xrightarrow{(0,1)} b\langle S_a, S_b\rangle$$
$$\langle S_b\rangle \xrightarrow{(0,1)} b|b\langle S_b, S_b\rangle \qquad \langle S_b, S_a\rangle \xrightarrow{(0,1)} b|b\langle S_b, S_b\rangle$$
$$\langle S_b\rangle \xrightarrow{(1,0)} a\langle S_b, S_a\rangle \qquad \langle S_b, S_b\rangle \xrightarrow{(1,0)} a\langle S_b, S_a\rangle$$

Nous remarquons que la construction de la grammaire produit des symboles terminaux comme $\langle S_b\rangle$, $\langle S_b, S_a\rangle$ et $\langle S_b, S_b\rangle$ qui ne sont pas accessibles depuis l'axiome $\langle S_a\rangle$.

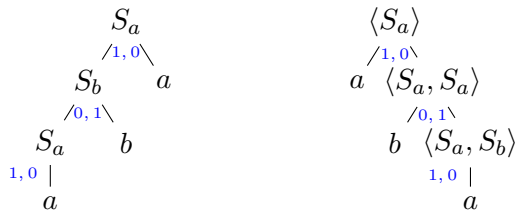


Figure 6.8. À gauche un arbre de dérivation t de G, et à droite, un arbre de dérivation t' de G_Φ, tel que $\Phi(t) = t'$.

◀

La proposition suivante établit précisément le lien entre les arbres de dérivation de G_Φ et l'image par Φ des arbres de dérivation de G :

Proposition 6.92.

▶ $Context_{G_\Phi}[N'] = \Phi(Context_G[N \cup \Sigma])$. Plus précisément, pour tout $P, Q \in N$:

a. L'ensemble des arbres de dérivation terminaux de G_Φ, de racine $\langle P\rangle$, est l'image par Φ de l'ensemble des arbres de dérivation terminaux de G ayant pour racine P. Avec nos notations :

$$Context_{G_\Phi}^{\langle P\rangle}[\Sigma] = \Phi(Context_G^P[\Sigma]).$$

b. L'ensemble des arbres de dérivation terminaux de G_Φ de racine $\langle P, Q\rangle$ est l'image par Φ de l'ensemble des Q-contextes de G de hauteur plus grande que 1, ayant pour racine P. Avec nos notations :

$$Context_{G_\Phi}^{\langle P,Q\rangle}[\Sigma] = \Phi(Context_G^P[Q] \setminus \{Q\}).$$

◀

Démonstration. Par double inclusion et induction ; nous devons prouver les deux affirmations simultanément.

Prouvons l'égalité dans les cas de base. Soit $t \in Context_{G_\Phi}^{\langle P\rangle}[\Sigma]$ de hauteur 1, qui est donc de la forme $t = (\langle P\rangle, L, a_1, \ldots, a_r)$ où $a_i \in \Sigma$ pour tout $1 \leqslant i \leqslant r$.

Comme t ne contient aucun symbole de la forme $\langle P, Q\rangle$, par construction de G_Φ, t correspond à l'application de la règle de dérivation $\langle P\rangle \xrightarrow{L} a_1 \dots a_r$, qui elle-même provient de la règle $P \xrightarrow{L} a_1 \dots a_r$ de G. Alors $t_G = (P, L, a_1, \dots, a_r)$ est bien dans $Context_G^P[\Sigma]$, et vérifie $\Phi(t_G) = t$.
Réciproquement, soit $t_G \in Context_G^P[\Sigma]$ de hauteur 1. Alors t_G est de la forme $t_G = (P, L, a_1, \dots, a_r)$, avec $a_i \in \Sigma$ pour tout $1 \leqslant i \leqslant r$. Ainsi $\Phi(t_G) = (\langle P\rangle, L, a_1, \dots, a_r)$ et comme $P \xrightarrow{L} a_1 \dots a_r$ est une règle de dérivation de G, par construction $\langle P\rangle \xrightarrow{L} a_1 \dots a_r$ est une règle de dérivation de G_Φ. Donc $\Phi(t_G) \in Context_{G_\Phi}^{\langle P\rangle}[\Sigma]$.

Soit $t \in Context_{G_\Phi}^{\langle P,Q\rangle}[\Sigma]$ de hauteur 1, de la forme $t = (\langle P, Q\rangle, L, a_1, \dots, a_r)$ où $a_i \in \Sigma$ pour tout $1 \leqslant i \leqslant r$. Comme t ne contient aucune symbole de la forme $\langle P, Q\rangle$ parmi ses fils, par construction t commence par l'application d'une règle $\langle P, Q\rangle \xrightarrow{L} a_1 \dots a_r$, qui vient de la règle de $P \xrightarrow{L} Qa_1 \dots a_r$ dans G. Alors $t_G = (P, L, Q, a_1, \dots, a_r)$ est bien dans $Context_G^P[Q]$, $t_G \neq Q$, et $\Phi(t_G) = t$.
Réciproquement, soit $t_G \in Context_G^P[Q]$ de hauteur 1 (donc $t_G \neq Q$). Alors il est de la forme $t_G = (P, L, Q, a_1, \dots, a_r)$, avec $a_i \in \Sigma$ pour tout $1 \leqslant i \leqslant r$. Ainsi $\Phi(t_G) = (\langle P, Q\rangle, L, a_1, \dots, a_r)$ et comme $P \xrightarrow{L} Qa_1 \dots a_r$ est une règle de dérivation de G, par construction $\langle P, Q\rangle \xrightarrow{L} a_1 \dots a_r$ est une règle de dérivation de G_Φ. Donc $\Phi(t_G) \in Context_{G_\Phi}^{\langle P\rangle}[Q]$.

Nous démontrons alors par induction l'inclusion suivante pour tous $P, Q \in N$:

$$Context_{G_\Phi}^{\langle P\rangle}[\Sigma] \subseteq \Phi(Context_G^P[\Sigma]) \text{ et } Context_{G_\Phi}^{\langle P,Q\rangle}[\Sigma] \subseteq \Phi(Context_G^P[Q]\backslash\{Q\}).$$

— Soit $t \in Context_{G_\Phi}^{\langle P\rangle}[\Sigma]$ de hauteur strictement plus grande que 1. Il y a deux cas de figure.

a. Premier cas : t est de la forme $t = (\langle P\rangle, L, a, t_1, \dots, t_r)$, et commence par une dérivation de la forme $\langle P\rangle \xrightarrow{L} a\langle P_1\rangle \dots \langle P_r\rangle$, avec $root(t_i) = \langle P_i\rangle$. Par construction de D', cela signifie que $P \xrightarrow{L} aP_1 \dots P_r$ est une règle de dérivation de G. De plus, pour tout $1 \leqslant i \leqslant r$, $t_i \in Context_{G_\Phi}^{\langle P_i\rangle}[\Sigma]$, donc par hypothèse d'induction, il existe des arbres de dérivation $t'_i \in Context_G^{P_i}[\Sigma]$ tels que $\Phi(t'_i) = t_i$.
Donc $t_G = (P, L, a, t'_1, \dots, t'_r) \in Context_G^P[\Sigma]$. Il est alors immédiat que $\Phi(t_G) = t$.

b. Second cas : t est de la forme $t = (\langle P\rangle, L, a, t_1, \dots, t_r, T)$ et commence par une dérivation de la forme $\langle P\rangle \xrightarrow{L} a\langle P_1\rangle \dots \langle P_r\rangle\langle P, Q\rangle$, avec $root(t_i) = \langle P_i\rangle$, et $root(T) = \langle P, Q\rangle$. Par construction de D', cela signifie que $Q \xrightarrow{L} aP_1 \dots P_r$ est une règle de dérivation de G. De plus, pour tout $1 \leqslant i \leqslant r$, $t_i \in Context_{G_\Phi}^{\langle P_i\rangle}[\Sigma]$, donc par hypothèse d'induction, il existe des arbres de dérivation $t'_i \in Context_G^{P_i}[\Sigma]$ tels que $\Phi(t'_i) = t_i$. Par hypothèse d'induction, comme $T \in Context_{G_\Phi}^{\langle P,Q\rangle}[\Sigma]$, il existe aussi un contexte $T' \in Context_G^P[Q]\backslash\{Q\}$ tel que $\Phi(T') = T$.
Donc $t_G = T'[(Q, L, a, t'_1, \dots, t'_r)] \in Context_G^P[\Sigma]$. Il est alors immédiat que $\Phi(t_G) = t$.

— Soit $t \in Context_{G_\Phi}^{\langle P,Q\rangle}[\Sigma]$ de hauteur > 1. Nous sommes aussi confrontés à deux cas de figure.

a. Premier cas : t est de la forme $t = (\langle P, Q\rangle, L, t_1, \dots, t_r)$, et commence par une dérivation de la forme $\langle P, Q\rangle \xrightarrow{L} \langle P_1\rangle \dots \langle P_r\rangle$, avec $root(t_i) = \langle P_i\rangle$. Par construction de D', $P \xrightarrow{L} QP_1 \dots P_r \in D$. De plus, pour tout $1 \leqslant i \leqslant r$, $t_i \in Context_{G_\Phi}^{\langle P_i\rangle}[\Sigma]$, donc par hypothèse d'induction, on peut trouver $t'_i \in Context_G^{P_i}[\Sigma]$ tel que $\Phi(t'_i) = t_i$.
Donc $t_G = (P, L, Q, t'_1, \dots, t'_r) \in Context_G^P[Q]$ et $\Phi(t_G) = t$.

b. Second cas : $t = (\langle P, Q\rangle, L, t_1, \dots, t_r, T)$, commence par une dérivation de la forme $\langle P, Q\rangle \xrightarrow{L} \langle P_1\rangle \dots \langle P_r\rangle\langle P, R\rangle$, avec $root(t_i) = \langle P_i\rangle$, et $root(T) = \langle P, R\rangle$. Donc $R \xrightarrow{L} QP_1 \dots P_r \in D$ par construction. Comme pour tout $1 \leqslant i \leqslant r$, $t_i \in Context_{G_\Phi}^{\langle P_i\rangle}[\Sigma]$, par hypothèse d'induction, il existe $t'_i \in Context_G^{P_i}[\Sigma]$ tel que $\Phi(t'_i) = t_i$. Et aussi par hypothèse d'induction, puisque $T \in Context_{G_\Phi}^{\langle P,R\rangle}[\Sigma]$, il existe $T' \in Context_G^P[R] \setminus \{R\}$ tel que $\Phi(T') = T$.
Donc $t_G = T'[(R, L, Q, t'_1, \dots, t'_r)] \in Context_G^P[Q]$, et enfin $\Phi(t_G) = t$.

L'inclusion dans l'autre sens utilise les mêmes idées, nous ne traitons qu'un seul cas, les autres étant similaires.

Soit $t \in Context_G^P[Q]$ de hauteur strictement plus grande que 1, avec $Q \in N$. Posons $R = root(t)$. Alors par la Proposition 6.85 il existe une règle de dérivation de la forme $B \xrightarrow{L} QP_1 \dots P_r$ avec $r \geqslant 0$, des arbres de dérivation $t_i \in Context_G^{P_i}[\Sigma]$, et $t' \in Tree_G(R)[B]$ un B-contexte tels que t se décompose en $t = t'[(B, L, Q, t_1, \dots, t_r)]$. Par définition, $\Phi(t) = (\langle R, Q\rangle, L, \Phi(t_1), \dots, \Phi(t_r), \Phi(t'))$ (t' n'est pas réduit à R car t serait alors de hauteur 1).

Par hypothèse d'induction, $\Phi(t_i) \in Context_{G_\Phi}^{\langle P_i\rangle}[\Sigma]$, et $\Phi(t') \in Context_{G_\Phi}^{\langle R,B\rangle}[\Sigma]$. De plus, par construction de D', $\langle R, Q\rangle \xrightarrow{L} \langle P_1\rangle \dots \langle P_r\rangle\langle R, B\rangle$ est une règle de dérivation de D'.

Donc $\Phi(t) \in Context_{G_\Phi}^{\langle R,Q\rangle}[\Sigma]$.

■

Corollaire 6.93 .
▶ $L(G) = L(G_\Phi)$ et G_Φ est non ambiguë. ◀

Démonstration. C'est une conséquence du fait que Φ est une bijection, par la Proposition 6.88, entre $Context_G^S[\Sigma]$ et $Context_{G_\Phi}^{\langle S\rangle}[\Sigma]$, qui préserve la frontière et la somme des semilinéaires. Comme G est non ambiguë, nous en déduisons que G_Φ est non ambiguë. ■

Proposition 6.94 .
▶ Il existe une $\mathbb{C}$FG $G' = (N'', \Sigma, \langle S\rangle, D'', \mathcal{S})$ non ambiguë, équivalente à G_Φ, dont toutes les règles de dérivation sont de la forme :

$$A \xrightarrow{L} aV_1 \dots V_r \quad \text{avec } a \in \Sigma,\ r \geqslant 0, \text{ et } V_i \in N' \text{ pour tout } i \in [r].$$

Autrement dit, toutes les règles de dérivation de G' commencent par la production d'une lettre de Σ, et toute production d'une lettre a lieu uniquement au début d'une règle de dérivation. ◀

Démonstration. Par construction, toutes les productions de G_Φ à partir d'un non terminal $\langle P\rangle$ commencent par produire une lettre de Σ. Toutes les productions à

partir d'un non terminal de la forme $\langle P, Q\rangle$ commencent par produire soit une lettre, soit un non terminal de la forme $\langle P_1\rangle$. Ainsi, G_Φ n'a aucune récursivité gauche.

Les dérivations de D'' sont construites à partir de D' en supprimant les règles de la forme $\langle Q, R\rangle \xrightarrow{L} \langle P_1\rangle \dots \langle P_r\rangle$ avec $\langle P_1\rangle$ un non terminal. Pour chaque règle supprimée de la forme $\langle Q, R\rangle \xrightarrow{L} \langle P_1\rangle \dots \langle P_r\rangle$, et pour chaque règle de D' de la forme $\langle P_1\rangle \xrightarrow{L_1} a\langle Q_1\rangle \dots \langle Q_s\rangle$, nous ajoutons à D'' la règle

$$\langle Q, R\rangle \xrightarrow{L+L_1} a\langle Q_1\rangle \dots \langle Q_s\rangle \dots \langle P_r\rangle\,.$$

Quitte à ajouter des non terminaux V_a à N' et des règles de dérivation de la forme $V_a \xrightarrow{\{0\}} a$, pour tout $a \in \Sigma$, il est possible de supprimer les terminaux qui ne sont pas en première position des membres droits des productions de G'.

La grammaire G'_Φ reconnaît alors facilement le même langage que G_Φ et est non ambiguë. ■

Nous avons ainsi démontré la proposition suivante :

Proposition 6.95 (Forme normale de Greibach)**.**
▶ Soit $L \subseteq \Sigma^*$ un langage algébrique de Parikh faiblement non ambigu. Alors $L \setminus \{\varepsilon\}$ est reconnaissable par une $\mathbb{C}$FG G' généralisée non ambiguë, telle que toute règle de dérivation de G' produise exactement une lettre de Σ, au début de la règle de dérivation. ◀

6.5.6 Élimination des ε-transitions

Proposition 6.96 .
▶ Soit $G = (N, \Sigma, S, D, \mathcal{S})$ une $\mathbb{C}$FG généralisée non ambiguë dont les productions sont dans $\Sigma \times N^*$. Alors $L(G)$ est reconnu par un automate de Parikh faiblement non ambigu sans ε-transitions. ◀

Démonstration. La construction classique s'adapte facilement en ajoutant des poids. Nous considérons ainsi l'automate de Parikh à pile, *qui accepte par pile vide*, $\mathcal{B} = (\{q_I\}, q_I, \{q_I\}, \Sigma, N, S, \Delta, \mathcal{S})$, avec Δ défini par :

$$\Delta = \{q_I, A \xrightarrow{a,L} q_I, \beta \;:\; (A \xrightarrow{L} a\beta) \in D, a \in \Sigma, \beta \in N^*\}\,.$$

Il est facile de voir que tout calcul final de l'automate à pile simule exactement une dérivation gauche de G, étiquetée par le même mot et de plus la bijection conserve les semilinéaires des calculs et des dérivations. On vérifie facilement que comme G est non ambiguë, $\mathcal{B}$ l'est aussi. On peut transformer $\mathcal{B}$ en automate de Parikh à vecteurs par la Proposition 6.57.

Comme nous avions défini les automates à pile qui acceptent par état final, il faut modifier légèrement l'automate à pile $\mathcal{B}$ qui accepte par pile vide et non par état final, en faisant attention à ne pas introduire d'ε-transition. Je propose de marquer le symbole de fin de pile par une barre pour que l'automate sache que le symbole qu'il dépile est en fin de pile. L'automate est construit pour ne pouvoir aller dans l'état final qu'en dépilant le dernier symbole de sa pile et en lisant une dernière lettre.

Mettre un symbole de fin de pile $\perp$ requiert un dépilement supplémentaire par rapport à la longueur du mot lu, et donc introduit une ε-transition.

Ainsi $\mathcal{B} = (\{q_I, q_f\}, q_I, \{q_f\}, \Sigma, N, \overline{S}, \Delta', \mathcal{S})$, avec $\Delta' = \Delta_1 \cup \Delta_2 \cup \Delta_3$ où :

$$\Delta_1 = \{q_I, A \xrightarrow{a,L} q_I, \beta \ : \ (A \xrightarrow{L} a\beta) \in D, a \in \Sigma, \beta \in N^*\}$$
$$\Delta_2 = \{q_I, \overline{A} \xrightarrow{a,L} q_I, V_1 \dots V_{r-1}\overline{V_r} \ : \ (A \xrightarrow{L} aV_1 \dots V_r) \in D, a \in \Sigma, r > 1\}$$
$$\Delta_3 = \{q_I, \overline{A} \xrightarrow{a,L} q_f, \varepsilon \ : \ (A \xrightarrow{L} a) \in D, a \in \Sigma\}\,.$$

Cette construction préserve le langage et la faible non-ambiguïté. ■

Proposition 6.97.
▶ Soit $\mathcal{B} = (Q, q_I, F, \Sigma, \Gamma, \bot, \delta, \mathcal{S})$ un automate de Parikh à pile faiblement non ambigu. Alors il existe un automate de Parikh à pile faiblement non ambigu sans ε-transitions, reconnaissant le même langage. ◀

Démonstration. En appliquant successivement les Propositions 6.79, 6.81, 6.83, 6.95 et 6.96, nous obtenons un automate de Parikh à pile généralisé

$$\mathcal{B}' = (Q', q_I', \{q_f'\}, \Sigma, \Gamma', \bot', \Delta', \mathcal{S})$$

tel que $L(\mathcal{B}') = L(\mathcal{B})\backslash\{\varepsilon\}$.

Si $\varepsilon \in L(\mathcal{B})$, nous voulons ajouter le mot vide à $L(\mathcal{B}')$, tout en conservant la faible non-ambiguïté. Nous considérons une copie

$$\mathcal{B}'' = (Q \cup \{q_0\}, \{q_0\}, \{q_f', q_0\}, \Sigma, \Gamma', \bot', \Delta'', \mathcal{S}')\,,$$

en ajoutant à l'automate $\mathcal{B}'$ un nouvel état q_0, qui est l'état initial, et est aussi final. L'ensemble des transitions Δ'' est une copie de Δ', et pour toute transition $q_I, A \xrightarrow{a,L} q \in \Delta'$, nous ajoutons aussi à Δ'' la transition $q_0, A \xrightarrow{a,\{\mathbf{1}\}+L} q$ où $\mathbf{1}$ est le vecteur avec des 1 à toutes les coordonnées. Ainsi $\mathcal{B}''$ ne peut se trouver dans l'état q_0 qu'au tout début du calcul, avant de lire la première lettre d'un mot, et q_0 est une copie de q_I qui est finale juste pour accepter le mot vide. De plus l'automate quitte l'état q_0 en ajoutant un 1 à toutes les coordonnées. Il suffit alors d'ajuster le semilinéaire $\mathcal{S}' = \{\mathbf{0}\} \cup (\mathcal{S} + \{\mathbf{1}\})$ pour accepter à la fois le calcul du mot vide, et les calculs en dehors de l'état q_0 qui sont les calculs acceptants de $\mathcal{B}'$ avec un 1 supplémentaire dans toutes les coordonnées. Il est immédiat que $L(\mathcal{B}'') = L(\mathcal{B}') \cup \{\varepsilon\} = L(\mathcal{B})$ et que $\mathcal{B}''$ est faiblement non ambigu.

Dans tous les cas, nous obtenons un automate généralisé qui reconnaît $\mathcal{L}(\mathcal{B})$, sans ε-transitions. Il suffit alors de transformer l'automate généralisé en automate à vecteurs par la même transformation que pour les automate de Parikh (Proposition 6.17). On remarque que cette transformation ne change pas les lettres des transitions (donc n'ajoute pas d'ε-transitions), et préserve la non-ambiguïté. ■

6.6 Conclusion et ouverture

Dans ce chapitre, nous avons formalisé différents modèles de machines à compteurs non ambiguës, et étudié leurs propriétés de clôture, ainsi que leurs liens avec d'autres modèles. Les exemples de langages de Parikh (faiblement) non ambigus de ce chapitre sont relativement simples, et sont tous reconnus par une RBCM déterministe bidirectionnelle. Il serait intéressant, pour étudier la propriété de clôture des langages

de Parikh non ambigus par complémentaire, de se demander si ces automates sont équivalents à un modèle sous-jacent déterministe.

C'est le cas pour les automates de Parikh non ambigus, qui comme nous l'avons dit, sont équivalents à au moins deux modèles d'automates déterministes [CFM13, FGM19]. Nous pouvons trouver une autre classe déterministe équivalente aux automates de Parikh non ambigus. Un automate de Parikh avec *lookahead* régulier est un automate de Parikh dont les transitions sont étiquetées par un tuple $(a, R, \boldsymbol{v})$ où a est une lettre, R un langage régulier, et $\boldsymbol{v}$ un vecteur : l'automate ne peut emprunter la transition $(a, R, \boldsymbol{v})$ que si la lettre en cours de lecture est a, et si la suite du mot qui n'a pas encore été lue appartient au langage régulier R : l'automate dispose d'un oracle qui lui dit si la suite du mot appartient bien au langage R. L'automate est déterministe si toutes les transitions sortantes d'un état étiquetées par la même lettre sont étiquetées par des langages réguliers deux à deux disjoints.

Il n'est pas difficile de montrer que la classe des automates de Parikh non ambigus et celle des automates de Parikh déterministes avec *lookahead* régulier sont équivalentes : l'automate de Parikh déterministe utilise son lookahead régulier pour guider de façon déterministe le bon choix des transitions qui mène à l'unique calcul final pour le mot en cours de lecture dans l'automate de Parikh non ambigu. Réciproquement, l'automate de Parikh non ambigu, simulant un automate déterministe avec *lookahead* régulier, choisit d'emprunter une transition de façon non déterministe, et vérifiera que son choix était correct à la fin du mot (sans utiliser le semilinéaire). Il peut le faire en lançant par exemple en parallèle des calculs dans des automates finis reconnaissant les langages réguliers présents des transitions. L'automate à la fin du mot vérifie que toutes les simulations des lookaheads réguliers sont dans un état acceptant : si c'est le cas et seulement si c'est le cas, l'automate va dans un état final. Comme l'automate avec lookahead était déterministe, l'automate non déterministe construit ainsi est non ambigu : tout mot possède un unique calcul final dans l'automate.

Comme il y a un nombre fini d'états, et que tous les calculs lisent les mêmes lettres en même temps, on peut simuler avec un automate fini le lancement d'un nombre arbitraire de calculs dans un automate fini.

Qu'en est-il de la classe des langages de Parikh faiblement non ambigus ? Existe-t-il un modèle équivalent déterministe ? Intuitivement, si nous cherchons un modèle déterministe équivalent :

— nous avons besoin d'un modèle bidirectionnel, sans quoi le modèle ne sera pas capable d'adapter son comportement en fonction de la fin du mot.

— nous avons besoin d'un modèle de type RBCM, qui peut adapter son calcul en fonction de la valeur de ses compteurs, sinon nous n'arriverons pas à trouver le milieu du mot.

Comme nous l'avons déjà évoqué, une première piste est la classe des RBCM déterministes bidirectionnelles. Nous pensons qu'elle est cependant plus faible, et que l'ajout d'un *lookahead* régulier augmente l'expressivité du modèle. Considérons le langage $d^*((a+b)c^+(a+b))^*$ des mots qui sont constitués d'un bloc initial de d, puis de blocs de c encadrés de chaque côté par un a ou un b. Nous nous intéressons au langage L des mots de $d^*((a+b)c^+(a+b))^*$ qui ont autant de d que de lettres c présentes dans un bloc entouré par la même lettre (il faut donc compter uniquement les c dans les patterns ac^+a et bc^+b). L'intuition nous pousse à penser que la RBCM a besoin d'un lookahead régulier pour savoir si elle doit compter le nombre de c dans le bloc qu'elle est en train de lire. Il faut cependant faire attention, car la machine est capable de calculer en fait plusieurs données : pour un mot de $w \in d^*((a+b)c^+(a+b))^*$, notons $|w|_c^{aa}$ (resp. $|w|_c^{bb}$, $|w|_c^{ab}$, $|w|_c^{ba}$)

le nombre de c présents dans des blocs encadrés par un a à gauche et un a à droite (resp. b et b, resp. a et b, resp. b et a). Avec un premier parcours du mot w la machine peut compter les valeurs $|w|_c^{a?} = |w|_c^{aa} + |w|_c^{ab}$, et $|w|_c^{b?} = |w|_c^{ba} + |w|_c^{bb}$, et en parcourant le mot dans l'autre sens, elle calcule les valeurs $|w|_c^{?a} = |w|_c^{aa} + |w|_c^{ba}$, et $|w|_c^{?b} = |w|_c^{ab} + |w|_c^{bb}$. Nous nous retrouvons avec un système de quatre équations à quatre inconnues, de rang 3 (car $|w|_c^{a?} + |w|_c^{b?} = |w|_c^{?a} + |w|_c^{?b} = |w|_c$). Cela ne suffit pas à calculer $|w|_c^{aa} + |w|_c^{bb}$, qui ne s'exprime pas comme combinaison linéaire des $|w|_c^{a?}, |w|_c^{b?}, |w|_c^{?a}, |w|_c^{?b}$, mais la machine peut par cette méthode déterminer $|w|_c^{aa} - |w|_c^{bb} = |w|_c^{a?} - |w|_c^{?b}$. Cela contredit déjà notre première intuition, et nous pousse à rester vigilant ; d'autant plus que la machine ne se restreint pas à calculer des combinaisons linéaires de cette forme, elle peut compter un bloc sur deux, compter un bloc uniquement si le précédent était bien encadré, voire compter seulement certains c dans des blocs, etc.

Si comme nous le pensons ce modèle déterministe est strictement inclus dans le celui des langages de Parikh faiblement non ambigus, il est toujours possible de lui ajouter un lookahead régulier pour augmenter son expressivité. Mais ce modèle a atteint pour le moment les limites de notre créativité pour concevoir des langages faiblement non ambigus : nous aurons besoin de nouvelles idées pour saisir l'origine de la non-ambiguïté, qui sorte du schéma déterministe bidirectionnel qui avait guidé inconsciemment nos exemples jusqu'alors. C'est sans doute une tâche difficile, mais qui serait très intéressante pour tester et mieux comprendre nos modèles.

Ce chapitre est en partie issu de l'article de conférence [BCKN20].

7 Intrinsèque ambiguïté et séries génératrices

7.1 Introduction : intrinsèque ambiguïté des langages algébriques

Le but de ce chapitre est d'étudier l'intrinsèque faible ambiguïté des langages de Parikh et des langages algébriques de Parikh introduits au chapitre précédent. Comme les langages algébriques de Parikh sont une généralisation des langages algébriques, il est important d'avoir une vision générale des techniques connues sur les langages algébriques, afin d'essayer de voir lesquelles leur sont spécifiques, et lesquelles ont une chance de s'étendre aux modèles à compteurs. Dans cette introduction détaillée, nous présentons donc un large panel des techniques utilisées pour démontrer l'intrinsèque ambiguïté des langages algébriques.

Soit Σ un alphabet fini. Une grammaire hors-contexte $G = (N, \Sigma, S, D)$ est dite *non ambiguë* si pour tout mot $w \in \mathcal{L}(G)$, il existe exactement un arbre de dérivation de G, de racine S, et de frontière w. Du côté des automates, un automate à pile $\mathcal{A}$ est non ambigu si pour tout mot de $w \in \mathcal{L}(\mathcal{A})$, il existe exactement un calcul acceptant étiqueté par w. Heureusement, ces notions coïncident pour les deux modèles définissant les langages algébriques : les constructions classiques d'équivalence entre automates à pile et grammaires hors-contexte préservent facilement la non ambiguïté, si bien qu'un langage algébrique est reconnu par une grammaire hors-contexte non ambiguë si et seulement s'il est reconnu par un automate à pile non ambigu.

Nous l'avons redémontré au chapitre précédent dans le cas plus général des automates de Parikh à pile.

Un langage algébrique est *intrinsèquement ambigu* s'il n'est reconnaissable par aucune grammaire non ambiguë (ou aucun automate à pile non ambigu). En 1961, Parikh démontre l'existence de langages algébriques intrinsèquement ambigus, en prouvant que le langage $L = \{a^n b^m a^p b^q \mid n = p \text{ ou } m = q, \text{ avec } n, m, p, q \in \mathbb{N}^*\}$ est intrinsèquement ambigu ([Par61], [Par66, Theorem 3]). La preuve de Parikh utilise un argument d'itération sur les arbres de dérivation d'une grammaire non ambiguë reconnaissant le langage. Les arguments par itération de ce type sont subtils et difficiles à mettre en place, car ils doivent s'appliquer à toute grammaire non

ambiguë reconnaissant le langage étudié. Plusieurs théorèmes ont été proposés dans le but de simplifier, dans les preuves d'intrinsèque ambiguïté, les arguments d'itération sur les arbres de dérivation. En 1966, Ginsburg et Ullian [GU66] arrivent ainsi à caractériser exactement l'intrinsèque ambiguïté des langages algébriques *bornés* par une propriété sur des ensembles semilinéaires : leur équivalence permet de prouver l'intrinsèque ambiguïté de certains langages bornés par des itérations plus simples portant sur des ensembles semilinéaires, et non plus sur des arbres de dérivation. Le célèbre lemme d'Ogden [Ogd68] quant à lui simplifie la mise en place d'itérations dans des grammaires, en facilitant l'identification de paires itérantes. Au début des années 80, pour démontrer l'intrinsèque ambiguïté d'un langage non borné, les principales techniques reposent donc sur des arguments d'itération, qui s'avèrent fastidieux voire inadaptés pour étudier certains langages algébriques complexes. En 1987, Flajolet [Fla87] propose une nouvelle approche, fondée sur l'étude des séries génératrices des langages. Partant du théorème de Chomsky-Schützenberger [CS63], qui démontre que la série génératrice d'un langage algébrique non ambigu est algébrique, Flajolet démontre l'intrinsèque ambiguïté de nombreux langages, en établissant des critères simples pour montrer que leur série génératrice n'est pas algébrique. Cette nouvelle approche complète très bien les techniques d'itérations vues précédemment : ces dernières sont plutôt efficaces et rapides pour cerner les langages qui ont une structure simple, et qui auront tendance à avoir une série génératrice algébrique (par exemple, tous les langages algébriques bornés ont une série génératrice rationnelle), tandis que l'approche par séries génératrices permet de traiter rapidement des langages suffisamment complexes pour avoir une série génératrice qui n'est pas algébrique.

Un langage L est borné s'il existe des mots $w_1, \dots, w_d$ tel que $L \subseteq w_1^* \dots w_d^*$.

Il n'est pas étonnant que les deux approches se complètent, car il n'existe pas de méthode universelle pour aborder tous les langages algébriques intrinsèquement ambigus : décider si un langage algébrique est intrinsèquement ambigu est indécidable [GU66], par réduction du problème de correspondance de Post (il s'agit d'un problème différent de celui de savoir si une grammaire donnée est ambiguë, qui est aussi indécidable [CS63]). Il est possible de démontrer l'indécidabilité de l'intrinsèque ambiguïté (et d'autres problèmes, comme l'intrinsèque infinie ambiguïté) à l'aide des critères généraux de Greibach [Gre68] d'indécidabilité de problèmes sur des familles de langages.

Nous utiliserons les techniques de Greibach pour démontrer l'indécidabilité de l'intrinsèque faible ambiguïté des automates de Parikh.

Dans la sous-section 7.1.1, nous donnons des exemples d'applications des techniques basées sur l'itération de Ginsburg et Ullian, et d'Ogden, et dans la sous-section 7.1.2, nous présentons les techniques analytiques de Flajolet. Enfin, dans la sous-section 7.1.3, nous proposons une nouvelle approche pour prouver l'intrinsèque ambiguïté des langages bornés à l'aide de séries génératrices (il s'agit de la seule partie de cette introduction qui est une véritable contribution personnelle).

7.1.1 Preuves d'intrinsèque ambiguïté par des arguments d'itération

Dans cette section, nous présentons deux outils utiles pour démontrer l'intrinsèque ambiguïté de certains langages algébriques : les critères de Ginsburg et Ullian [GU66], et le lemme d'Ogden [Ogd68]. Nous rappelons que ces deux outils ont pour objectif de simplifier les raisonnements par itération, soit en transposant le problème sur une propriété d'ensembles semilinéaires pour les langages bornés, soit en facilitant l'identification des paires itérantes dans une grammaire.

Critères de Ginsburg et Ullian pour les langages bornés.

Dans toute cette section, nous fixons $d \geqslant 1$, et Σ un alphabet de taille plus grande que 2. Nous fixons aussi un tuple de d mots $w_1, \dots, w_d \in \Sigma^*$, noté $\langle w \rangle := \langle w_1, \dots, w_d \rangle$.

Les langages algébriques sur un alphabet de taille 1 sont rationnels par le théorème de Parikh.

Définition 7.1 (Langages algébriques bornés, [GU66]).
▶ Un langage L est dit *borné* par rapport à $\langle w \rangle$ si $L \subseteq w_1^* \dots w_d^*$.
Définissons la fonction $f_{\langle w \rangle} : \mathbb{N}^d \to w_1^* \dots w_d^*$ par $f_{\langle w \rangle}(p_1, \dots, p_d) = w_1^{p_1} \dots w_d^{p_d}$ pour tout $\boldsymbol{p} \in \mathbb{N}^d$.
Un langage L borné par rapport à $\langle w \rangle$ est appelé *semilinéaire* si $f_{\langle w \rangle}^{-1}(L)$ est un ensemble semilinéaire. ◀

Si chaque w_i est une lettre distincte de Σ, alors l'application $f_{\langle w \rangle}$ est bijective, et sa réciproque est l'image de Parikh sur $w_1^* \dots w_d^*$.

Remarque 7.2 .
▶ Les langages bornés semilinéaires coïncident avec les langages de Parikh bornés (et sont même reconnaissables par un langage de Parikh déterministe) [CFM12b]. Les techniques présentées ici sont donc propres aux langages algébriques *sans compteurs*. ◀

La proposition suivante est une généralisation du théorème de Parikh dans le cas des langages bornés :

Proposition 7.3 ([GU66]).
▶ Tout langage algébrique borné est semilinéaire. ◀

La définition suivante définit une classe d'ensembles semilinéaires associés aux langages algébriques bornés. Dans un article précédent [GS66], les auteurs démontrent qu'un langage L borné dans $a_1^* \dots a_d^*$ avec $\langle a \rangle = \langle a_1, \dots, a_d \rangle$ des lettres distinctes, est algébrique si et seulement si $f_a^{-1}(L)$ est une union finie d'ensembles linéaires, dont les ensembles de périodes sont stratifiés.

Définition 7.4 (Ensemble stratifié, [GS66, GU66]).
▶ Un sous-ensemble $X \subseteq \mathbb{N}^d$ est stratifié si :

a. chaque élément de X possède au plus deux coordonnées non nulles ;

b. on ne peut trouver quatre entiers distincts $1 \leqslant i < j < k < m \leqslant d$ ni deux vecteurs $\boldsymbol{x}, \boldsymbol{x}' \in X$ tels que $x_i x'_j x_k x'_m \neq 0$. En d'autres mots, deux éléments distincts de X ne peuvent avoir des coordonnées non nulles qui "s'entrelacent".

◀

Nous dirons parfois par abus de langage qu'un ensemble linéaire est stratifié si son ensemble de périodes est stratifié. Le théorème suivant de Ginsburg and Ullian est fondamental et caractérise les ensembles semilinéaires associés aux langages bornés inambigus :

Théorème 7.5 (Caractérisation de l'intrinsèque ambiguïté des langages bornés, [GU66]).
▶ Soit L un langage algébrique borné, tel que $L \subseteq w_1^* \dots w_d^*$. Alors L est intrinsèquement ambigu si et seulement si $f_{\langle w \rangle}^{-1}(L)$ n'est pas une union finie d'ensembles linéaires *disjoints*, dont chaque ensemble de période est *stratifié* et constitué de vecteurs *linéairement indépendants*. ◀

Il n'est pas nécessaire que la décomposition d'un mot dans $w_1^* \dots w_d^*$ soit non ambiguë.

Un des sens de l'équivalence se conçoit bien dans le cas où les w_i sont des symboles distincts et l'ensemble semilinéaire associé à L est une union disjointe d'ensembles linéaires à périodes stratifiées linéairement indépendantes. En raisonnant sur un ensemble linéaire de cette forme, on construit facilement une grammaire non ambiguë reconnaissant le langage (la condition de non entrelacement permet d'ordonner les vecteurs des périodes selon leurs couples de coordonnées non nulles, dans un ordre qui les emboîte). C'est l'autre direction qui est le coeur du théorème de Ginsburg et Ullian, et qui est fondée sur des arguments profonds sur les arbres de dérivation. De l'aveu d'un des auteurs dans son livre [Gin66, p. 188], *"The proof of the necessity is extremely complicated"*.

L'avantage des critères de Ginsburg et Ullian est d'avoir réussi à quitter le monde des grammaires et des arbres de dérivation, en se ramenant à des ensembles semilinéaires, dont la structure est un peu plus simple : ils guident les arguments d'itérations en se focalisant sur le semilinéaire, et en s'abstrayant ainsi des lettres du langage. Cependant, comme il s'agit d'une reformulation exacte de l'intrinsèque ambiguïté en terme de semilinéaires, leur critère n'est pas applicable directement pour effectuer une itération. Un travail de préparation avant de trouver une itération est ainsi nécessaire. Nous donnons un exemple de preuve d'intrinsèque ambiguïté fondée sur cet argument :

Proposition 7.6 (Un langage intrinsèquement ambigu, [GU66]).
▶ Le langage $L = \{a^n b^m c^p \mid n = m \text{ ou } m = p, \text{ avec } n, m, p \in \mathbb{N}^*\}$ est intrinsèquement ambigu. ◀

La preuve peut être passée ou lue rapidement, l'idée est de se convaincre qu'en pratique il manque aux critères de Ginsburg et Ullian un outil pour guider l'itération dans les semilinéaires. En réalité, les arguments d'itération ne sont pas la meilleure utilisation des critères de Ginsburg et Ullian, cf. Remarque 7.11 et la sous-section 7.1.3.

Démonstration. Nous reportons ici la preuve de [GU66, Theorem 6.2], qui le démontrent pour une variante similaire ; la preuve est exactement la même.

Notons $\langle \boldsymbol{w} \rangle = \langle a, b, c \rangle$. Supposons que $S := f_{\boldsymbol{w}}^{-1}(L)$ s'écrit sous la forme $S = \biguplus_{j=1}^{k} L_j$, avec chaque $L_j = \boldsymbol{c_j} + P_j^*$ qui est un ensemble linéaire stratifié dont les périodes sont linéairement indépendantes. Fixons $L_j = \boldsymbol{c} + P^*$ un tel ensemble, avec $\boldsymbol{c}$ son vecteur constant et P son ensemble de périodes stratifiées et linéairement indépendantes.

Soit $\boldsymbol{z} = (z_1, z_2, z_3) \in \mathbb{N}^3$ une somme finie de périodes de P. Alors $z_1 = z_2$ ou $z_2 = z_3$. En effet, supposons que $z_1 \neq z_2$. Comme pour tout $n \geqslant 0$, $\boldsymbol{c} + n\boldsymbol{z} \in L_j$, alors pour tout $n \geqslant 0$, ou bien $c_1 + nz_1 = c_2 + nz_2$, ou bien $c_2 + nz_2 = c_3 + nz_3$. Or il n'y a qu'un nombre fini de n tels que $c_1 + nz_1 = c_2 + nz_2$: en effet s'il y en avait une infinité, l'égalité $(c_1 - c_2)/n = z_2 - z_1$ impliquerait $z_1 = z_2$. Donc il y a une infinité d'entiers n tels que $c_2 + nz_2 = c_3 + nz_3$, et donc $z_2 = z_3$.

En particulier, par ce qui précède et par stratification, toutes les périodes des linéaires L_j sont de la forme $(d, d, 0)$, $(0, 0, d)$, $(0, d, d)$ ou $(d, 0, 0)$ avec $d \in \mathbb{N}^*$.

Soit m la plus grande coordonnée apparaissant dans les constantes des L_j. On désigne par $H \subseteq [1, k]$ l'ensemble des indices j tels que P_j^* génère des vecteurs dont les trois coordonnées sont non nulles (autrement dit on peut trouver trois vecteurs $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in P_j$, pas forcément distincts, tels que $x_1 > 0, y_2 > 0, z_3 > 0$). Ainsi, si un vecteur $\boldsymbol{v} \in S$ vérifie $v_1 > m$, $v_2 > m$ et $v_3 > m$, alors $\boldsymbol{v} \in L_j$ pour un certain $j \in H$.

On peut partitionner $H = H_1 \uplus H_2$ avec H_1 qui contient les indices des linéaires dont les périodes contiennent un vecteur de la forme $(d, d, 0)$, avec $d > 0$, et H_2 ceux dont les périodes contiennent un vecteur de la forme $(0, e, e)$, avec $e > 0$. H_1 et H_2 sont bien disjoints, car sinon il existerait une somme de vecteurs d'un même

ensemble de périodes de la forme $(d, d+e, e)$ avec $d \neq d+e$ et $d+e \neq e$, ce qui contredit que toute somme $\boldsymbol{z}$ de périodes vérifie $z_1 = z_2$ ou $z_2 = z_3$.

Notons R l'ensemble des entiers non nuls $d > 0$ tels que $(d, d, 0) \in H_1$ ou $(0, d, d) \in H_2$. Soit $p = \prod_{d \in R} d$ le produit de tous les entiers de R. Le vecteur $\boldsymbol{x} := (m+p+1, m+p+1, m+p+1)$ appartient à S, et $m+1+p > m$ donc $\boldsymbol{x} \in L_r$ pour un $r \in H$.

Supposons que $r \in H_1$ (la preuve est analogue si $r \in H_2$). Alors comme le vecteur $\boldsymbol{u} := (m+p+1, m+1, m+1) \in S$, par le même argument, $\boldsymbol{u} \in L_s$ pour un $s \in H$. Si $s \in H_1$, alors L_s contient une période de la forme $(d, d, 0)$ avec $d > 0$, si bien que le vecteur $(m+p+1+d, m+1+d, m+1) \in L_s$, mais ce vecteur n'appartient pas à S, contradiction. Donc $s \in H_2$, et ainsi L_s contient une période de la forme $(0, e, e)$ avec $e > 0$ et $e \in R$, donc e divise p. Ainsi L_s contient le vecteur $\boldsymbol{x} = \boldsymbol{u} + \frac{p}{e}(0, e, e)$. Comme les ensembles linéaires sont disjoints, $r = s$. Mais $s \in H_1$ et $s \in H_2$, contradiction avec le fait que $H_1 \cap H_2 = \emptyset$. Donc r ne peut être dans H_1.

On prouve par un raisonnement similaire que r ne peut être dans H_2, ce qui est absurde. Donc S n'est pas une union disjointe de linéaires stratifiés dont les périodes sont indépendantes. Donc L est intrinsèquement ambigu. ■

Les critères de Ginsburg et Ullian sont utilisés par [HU66] pour montrer l'indépendance de l'ambiguïté des langages algébriques par passage au complémentaire vis-à-vis de l'algébricité :

Proposition 7.7 (Indépendance de l'intrinsèque ambiguïté par passage au complémentaire, [HU66]).
▶ Le langage $L_1 = \{a^p b^q c^r d^s e^t \mid (p = q \wedge r = s) \vee (q = r \wedge s = t)\}$ est intrinsèquement ambigu, et son complémentaire est un langage algébrique.

Le langage $L_2 = \{a^p b^q c^r d^s \mid ((10p < q < 12p \vee 10q < p < 12q) \wedge (10r < s < 12r \vee 10r < s < 12r)) \vee (10q < r < 12q \wedge 6p < s < 8p)\}$ est algébrique non ambigu, et son complémentaire n'est pas algébrique. ◀

Remarque 7.8 .
▶ Nous n'avons pas à notre connaissance d'exemple de langage algébrique inambigu *non borné* dont le complémentaire n'est pas algébrique. En trouver un serait une piste pour montrer que la classe des langages de Parikh algébriques faiblement non ambigus n'est pas close par passage au complémentaire. ◀

Lemme d'Ogden

En 1968, Ogden démontre son fameux lemme, et l'utilise pour démontrer l'intrinsèque ambiguïté d'un langage non borné.

Il s'agit du langage $abM^* \cup M^*a^*b$ où $M = \{a^i b a^{i+1} b \mid i \in \mathbb{N}\}$.

Théorème 7.9 (Lemme d'Ogden, [Ogd68, Car08]).
▶ Soit $G = (N, \Sigma, S, D)$ une grammaire hors-contexte. Alors il existe un entier k tel que pour tout mot $\xi \in (N \cup \Sigma)^*$ dérivé à partir de l'axiome S, ayant au moins k positions distinguées, il existe un symbole $A \in N$, et des mots $\alpha, u, \beta, v, \gamma \in (N \cup \Sigma)^*$ tels que

a. $S \Rightarrow^* \alpha A \gamma$, $A \Rightarrow^* uAv + \beta$, et $\xi = \alpha u \beta v \gamma$;

b. au moins une des affirmations suivantes est vraie :

— les mots α, u et β contiennent tous les trois des positions distinguées ;

Le lemme d'Ogden dans [Ogd68] est présenté avec une dérivation d'un mot de Σ^*, nous présentons ici la version de [Car08, Lemme 2.43] avec une dérivation d'un mot de $(N \cup \Sigma)^*$ (l'argument est exactement le même).

— les mots β, v et γ contiennent tous les trois des positions distinguées ;

c. le mot $u\beta v$ contient moins de k positions distinguées. ◀

Le lemme d'Ogden simplifie les arguments par itération dans des grammaires, et permet entre autres de prouver que certains langages ne sont pas algébriques, ou sont intrinsèquement ambigus. La preuve du lemme d'Ogden n'est pas très compliquée, bien qu'astucieuse, et contraste avec la mécanique de preuve très complexe mise en place par Ginsburg et Ullian pour les langages bornés. Ogden répond en quelque sorte au problème ouvert posé par Ginsburg dans son livre [Gin66, p. 211], de savoir s'il existe une preuve "simple" d'intrinsèque ambiguïté du langage $a^n b^m c^p$ avec $n = m$ ou $m = p$ qui n'utilise pas ses techniques de preuve.

Ogden [Ogd68] remarque, sans faire la preuve, que les arguments exposés dans son article s'adaptent pour redémontrer l'intrinsèque ambiguïté du langage des mots $a^n b^m c^p$ avec $n = m$ ou $m = p$. Nous présentons la preuve de cette affirmation présentée par [Car08] dans son livre :

Proposition 7.10 (Un langage intrinsèquement ambigu, [Car08, Ogd68]).
▶ Le langage $L = \{a^n b^m c^p \mid n = m \text{ ou } m = p, \text{ avec } n, m, p \in \mathbb{N}^*\}$ est intrinsèquement ambigu. ◀

Démonstration. Nous présentons la preuve de [Car08] qui utilise les arguments présentés dans [Ogd68]. Supposons que L est reconnu par une grammaire non ambiguë $G = (N, \Sigma, S, D)$. Soit k l'entier du lemme d'Ogden. Nous appliquons le lemme au mot $\xi = a^k b^k c^{k+k!}$, en distinguant les lettres b : il existe un symbole $A \in N$, et des mots $\alpha, u, \beta, v, \gamma \in (N \cup \Sigma)^*$ tels que $a^k b^k c^{k+k!} = \alpha u \beta v \gamma$, $S \Rightarrow^* \alpha A \gamma$ et $A \Rightarrow^* uAv + \beta$. De plus le mot $u\beta v$ contient moins de k positions distinguées, et au moins l'un des triplets (α, u, β) ou (β, v, γ) contient des positions distinguées dans chacun de ses mots.

Certains arguments ressemblent à ceux utilisés dans la preuve de Ginsburg et Ullian.

La règle $A \Rightarrow^* uAv + \beta$ permet d'itérer u et v tout en restant dans le langage, donc forcément u est de la forme a^i et v est de la forme b^i, pour $0 \leqslant i$. Comme $u\beta v$ contient moins de k positions distinguées, $i \leqslant k$. De plus au moins l'un des deux mots u ou v n'est pas vide, car u ou v doit contenir une position distinguée, donc $i, j > 0$. En itérant la règle $A \Rightarrow^* uAv$ exactement $k!/i$ fois dans la dérivation du mot ξ, nous obtenons un arbre de dérivation T_1 pour le mot $a^{k+k!} b^{k+k!} c^{k+k!}$. Cet arbre contient le sous-arbre associé à la dérivation $A \Rightarrow^* a^{k!} A b^{k!} \Rightarrow^* a^{k!} \beta b^{k!}$, dont la frontière ne contient que des a et des b, et au moins $k!$ lettres a et $k!$ lettres b.

Il ne faut pas oublier de préciser que le sous-arbre ne contient pas que des b dans sa frontière.

Le même raisonnement appliqué au mot $\xi' = a^{k+k!} b^k c^k$ en distinguant les lettres b permet d'obtenir un arbre de dérivation T_2 du même mot $a^{k+k!} b^{k+k!} c^{k+k!}$, qui contient un sous-arbre dont la frontière ne contient que des b et des c, et au moins $k!$ lettres b et $k!$ lettres c.

Enfin T_1 et T_2 ne peuvent pas être égaux, car il n'y a que $k + k!$ lettres b dans leur frontière, et il n'est pas possible d'avoir deux sous-arbres distincts qui contiennent chacun plus de $k!$ lettres b. Nous obtenons donc deux arbres de dérivation distincts pour un même mot du langage. Contradiction avec la non ambiguïté de G. Donc L est intrinsèquement ambigu. ■

Remarque 7.11 .
▶ Sur des exemples simples, comme le langage précédent, la preuve par le lemme d'Ogden est finalement plus simple que celle de Ginsburg et Ullian. De plus, le

champ d'application du lemme d'Ogden ne se restreint pas aux langages bornés. Cependant, contrairement au lemme d'Ogden qui ne fournit qu'un outil guidant les arguments par itération, le théorème de Ginsburg et Ullian est une caractérisation exacte des langages bornés algébriques intrinsèquement ambigus : ne l'utiliser que pour guider des arguments d'itération dans des semilinéaires, comme cela a été fait historiquement, est en quelque sorte un contre-emploi pour ce théorème bien plus profond et difficile à démontrer que le lemme d'Ogden. Nous montrerons à la sous-section 7.1.3 comment, avec les bons outils, les critères de Ginsburg et Ullian permettent de démontrer en quelques lignes l'intrinsèque ambiguïté de nombreux langages bornés. ◀

À l'aide de séries génératrices.

7.1.2 Preuves d'intrinsèque ambiguïté par argument analytique : la méthode de Flajolet

En 1987, Flajolet propose une nouvelle approche pour démontrer l'intrinsèque ambiguïté de certains langages, à partir de leurs séries génératrices [Fla87]. Son article contraste avec les techniques précédemment exposées par la facilité avec laquelle il démontre l'intrinsèque ambiguïté de nombreux langages algébriques (une quinzaine !), dont certains étaient jusqu'alors inaccessibles. Comme sa méthode échoue à cerner l'intrinsèque ambiguïté des langages bornés, elle complète ainsi parfaitement les techniques usuelles par itération.

En quelques mots, l'approche de Flajolet repose sur la contraposée du théorème de Chomsky-Schützenberger, qui démontre qu'un langage algébrique non ambigu possède une série génératrice algébrique. À partir de ce constat, [Fla87] a développé des critères pour montrer que certains langages algébriques sont intrinsèquement ambigus, en montrant que leur série génératrice n'est pas algébrique. Cette approche nous intéresse particulièrement, car c'est en s'inspirant des travaux de Flajolet qu'en 1993, [Mas93] a introduit sa classe LCL, puis en 2017 [CM17] la classe RCM, qui sont des classes de langages construites exprès pour avoir une série génératrice holonome. Certains critères développés par Flajolet se généralisent ainsi naturellement pour démontrer l'intrinsèque faible ambiguïté de certains langages algébriques de Parikh.

Les séries holonomes constituent une classe de séries plus large que les séries algébriques.

Nous rappelons la définition de la série génératrice d'un langage :

Définition 7.12 (Série génératrice d'un langage).
▶ Soit $\Sigma = \{a_1, \dots, a_d\}$ un alphabet fini, avec $d \geqslant 1$. Soit $L \subseteq \Sigma^*$ un langage. La *série génératrice multivariée* du langage L est la série formelle de variables $(x_1, \dots, x_d)$ définie par :

$$L(x_1, \dots, x_d) := \sum_{w \in L} x_1^{|w|_{a_1}} \dots x_d^{|w|_{a_d}} = \sum_{i_1, \dots, i_d \in \mathbb{N}} \ell_{i_1,\dots,i_d} x_1^{i_1} \dots x_d^{i_d} ,$$

où $\ell_{i_1,\dots,i_d}$ désigne le nombre de mots dans le langage L ayant i_k occurrences de la lettre a_k pour $k \in [d]$.

La *série génératrice univariée* de L est la série en x définie par :

$$L(x) := \sum_{w \in L} x^{|w|} = \sum_{n \in \mathbb{N}} \ell_n x^n ,$$

où ℓ_n désigne le nombre de mots dans le langage L de longueur n. Nous remarquons notamment que la série univariée $L(x)$ s'obtient à partir de la série multivariée en remplaçant chaque variable $x_1, \dots, x_d$ par x. ◀

Exemple 7.13 .
▶ La série génératrice multivariée du langage $(ab)^*$ est $L(x_1, x_2) = \sum_{n\in\mathbb{N}} x_1^n x_2^n = \frac{1}{1-x_1x_2}$. Sa série univariée est $L(x) = \frac{1}{1-x^2}$. ◀

Pour simplifier les notations dans les exemples, nous utiliserons souvent le même symbole pour les lettres de l'alphabet et les variables de la série multivariées : nous écrirons donc $L(a, b) = \frac{1}{1-ab}$.

Définition 7.14 (Série algébrique).
▶ Une série $L(x_1, \ldots, x_d)$ est dite algébrique sur $\mathbb{Q}$ s'il existe un polynôme non nul $P(x_1, \ldots, x_d, Y) \in \mathbb{Q}[x_1, \ldots, x_d, Y]$ tel que

$$P(x_1, \ldots, x_d, L(x_1, \ldots, x_d)) = 0\,.$$

Une série qui n'est pas algébrique est dite *transcendante*. ◀

Exemple 7.15 .
▶ Toute série rationnelle est algébrique : si on note L sous la forme $L = P/Q$, avec P et Q des polynômes dans $\mathbb{Q}[x_1, \ldots, x_d]$, alors L est annulée par le polynôme $Q(x_1, \ldots, x_d)Y - P(x_1, \ldots, x_d)$. ◀

Le théorème suivant établit le lien entre les séries algébriques et les langages algébriques :

Théorème 7.16 (Théorème de Chomsky-Schützenberger, [CS63]).
▶ La série génératrice (multivariée) d'un langage algébrique non ambigu est algébrique sur $\mathbb{Q}$. ◀

La contraposée de ce théorème s'énonce ainsi : si la série génératrice d'un langage algébrique n'est pas algébrique, alors ce langage est intrinsèquement ambigu. Dans son article, [Fla87] Flajolet rassemble de nombreux outils pour prouver qu'une série n'est pas algébrique. Nous en présentons quelques-uns dans la proposition suivante :

Proposition 7.17 (Outils pratiques pour montrer la transcendance d'une série, [Fla87]).
▶ Soit $L(z) = \sum_{n\in\mathbb{N}} \ell_n z^n$ une série entière.

a. Si la fonction $L(z)$ possède une infinité de singularités, alors $L(z)$ est transcendante.

b. Si $\ell_n \sim_{n\to\infty} \gamma\beta^n n^r$, avec $r \notin \mathbb{Q}\backslash\{-1, -2, -3, \ldots\}$, ou β transcendant, ou $\gamma\Gamma(r + 1)$ transcendant, alors $L(z)$ est transcendante.

c. Si ℓ_n ne vérifie pas à partir d'un certain rang une équation de récurrence linéaire à coefficients polynomiaux en n, alors $\ell(z)$ est transcendante.

◀

Corollaire 7.18 (Séries lacunaires, [Fla87]).
▶ Soit $L(z)$ une série. Nous l'écrivons de façon à omettre ses coefficients nuls, sous la forme $L(z) = \sum_{i\in\mathbb{N}} a_i x^{c_i}$ avec pour tout $i \in \mathbb{N}$, $a_i \neq 0$, et la suite (c_i) est une suite strictement croissante d'entiers. Alors la série $L(z)$ est appelée *lacunaire* si

La suite (c_i) est le support de $L(z)$.

$$\lim_{i\to\infty} (c_{i+1} - c_i) = +\infty\,,$$

autrement dit si ses coefficients non nuls sont de plus en plus espacés.

Une série lacunaire n'est pas algébrique, pour au moins deux raisons :

— les séries de cette forme ont été bien étudiées, et elles admettent un nombre infini de singularités, qui sont denses dans son cercle de convergence.

— plus simplement, on se convainc facilement que ℓ_n ne peut vérifier une équation de récurrence linéaire à coefficients polynomiaux en n de la forme

$$p_r(n)\ell_{n+r} + \ldots + p_0(n)\ell_n = 0$$

sans être nulle à partir d'un certain rang.

Par conséquent, les séries lacunaires $\sum_{n\in\mathbb{N}} x^{n^2}$ et $\sum_{n\in\mathbb{N}} x^{2^n}$ sont transcendantes. ◀

Comme les séries algébriques sont closes par de nombreuses opérations (addition, multiplication, composition, dérivation), les preuves analytiques d'intrinsèque ambiguïté consistent à supposer par l'absurde que la série génératrice du langage est algébrique, puis en utilisant ces opérations de clôture, la transformer en une série plus simple qui ne vérifie pas un des critères ci-dessus, et qui n'est donc pas algébrique, ce qui amène à une contradiction.

Nous reprenons maintenant deux exemples de preuves d'intrinsèque ambiguïté suivant ce schéma, présentés par Flajolet dans son article [Fla87]. En reprenant les notations de Flajolet, pour tout entier $n \in \mathbb{N}$, nous définissons la notation $\underline{n} := a^n b$ qui représente l'entier n encodé en unaire dans l'alphabet $\{a, b\}$.

Proposition 7.19 (Deux langages intrinsèquement ambigus, [Fla87]).
▶ Les langages $\Omega_3 = \{w \in \{a,b,c\}^* \mid |w|_a \neq |w|_b \text{ ou } |w|_b \neq |w|_c\}$ et $\Omega_4 = \{w \in \{a,b,c,d\}^* \mid |w|_a \neq |w|_b \text{ ou } |w|_c \neq |w|_d\}$ sont intrinsèquement ambigus. ◀

Nous verrons que ces deux langages sont des langages de Parikh déterministes, ce qui illustre que la notion d'intrinsèque ambiguïté dépend du modèle sous-jacent d'automate.

Démonstration. Notons $L_3(x)$ la série génératrice de Ω_3 et supposons qu'elle est algébrique. Alors par stabilité pas somme, la série génératrice $\overline{L_3}(x) = \frac{1}{1-3x} - L_3(x)$ du complémentaire de Ω_3 dans $\{a,b,c\}^*$ est algébrique. Or, les mots dans le complémentaire de Ω_3 sont les mots de $\{a,b,c\}^*$ qui ont le même nombre de a, de b et de c : ils sont donc de longueur un multiple de 3. Pour $n \in \mathbb{N}^*$, il y a donc $\ell_{3n} = \binom{3n}{n,n,n} := \frac{(3n!)}{n!n!n!}$ mots de taille $3n$ dans le complémentaire de Ω_3. Par la formule de Stirling, $\ell_{3n} \sim_{n\to\infty} \sqrt{3}\frac{3^{3n}}{2\pi n}$, et l'exposant -1 du terme n^{-1} est un critère de transcendance, par la Proposition 7.17. Donc $\overline{L_3}(x)$ est transcendant, contradiction. Donc $L_3(x)$ est transcendant, et donc Ω_3 est intrinsèquement ambigu.

Par la même idée, notons $\overline{L_4}(x)$ la série génératrice du complémentaire de Ω_4, qui compte les mots de $\{a,b,c,d\}^*$ qui ont autant de a que de b, et autant de c que de d. Ce sont donc des mots de longueur paire. Notons ℓ_{2n} le nombre de mots dans $\overline{\Omega_4}$ de longueur $2n$. Nous voyons facilement que $\ell_{2n} = \sum_{k=0}^{n} \binom{2n}{2k}\binom{2k}{k}\binom{2(n-k)}{n-k}$ (on partitionne selon le nombre de a dans le mot. Le premier binôme correspond au choix dans le mot des $2k$ positions qui accueilleront des a et des b, le second au choix, parmi ces $2k$ positions, des k positions qui accueilleront les a, et le dernier binôme correspond au choix des $n-k$ positions des lettres c parmi les $2(n-k)$ restantes). En développant les formules binomiales avec des factorielles, on obtient $\ell_{2n} = \binom{2n}{n}\sum_{k=0}^{n}\binom{n}{k}^2 = \binom{2n}{n}^2$ par la formule de Vandermonde. Or par la formule de Stirling, $\ell_{2n} \sim_{n\to\infty} \frac{2^{4n}}{\pi n}$, et le terme en n^{-1} est incompatible avec l'algébricité. Donc la série génératrice du langage Ω_4 est forcément transcendante, et donc Ω_4 est intrinsèquement ambigu. ■

Remarque 7.20.
▶ Le langage $\Omega_3 \cap a^*b^*c^* = \{a^n b^m c^p \,|\, n \neq m \text{ ou } m \neq p\}$ est borné et a une série génératrice rationnelle. Ce langage résiste à la fois aux techniques analytiques de Flajolet et aux arguments d'itérations, qui sont plutôt adaptés aux contraintes d'égalités. L'intrinsèque ambiguïté de ce langage était un problème ouvert au moment où Flajolet a publié son article en 1987. Nous verrons à la section suivante que les critères de Ginsburg et Ullian permettent en fait de démontrer son intrinsèque ambiguïté, ainsi que l'intrinsèque ambiguïté du langage $\Omega_4 \cap a^*c^*b^*d^*$. Remarquons que leur intrinsèque ambiguïté implique celle des langages Ω_3 et Ω_4, par stabilité des langages non ambigus par intersection avec un langage rationnel. ◀

Pour $k \in \mathbb{N}^*$, une grammaire est dite k-ambiguë si tout mot qu'elle génère admet au plus k arbres de dérivations différents. Un langage algébrique L est dit k-ambigu s'il est reconnu par une grammaire k-ambiguë, finiment ambigu s'il existe un entier k tel que L est k-ambigu, et infiniment ambigu si L n'est pas finiment ambigu. L'union de deux langages algébriques est donc toujours 2-ambiguë. Une question naturelle est de se demander si le nombre (fini ou infini) de singularités de la série génératrice d'un langage est relié à sa finie ou infinie ambiguïté. En fait, ces deux notions sont indépendantes : le langage $(L\#)^*$ avec $L = \{a^n b^m c^p \,|\, n = m \text{ ou } m = p\}$ est en effet facilement infiniment ambigu en adaptant la preuve par le lemme d'Ogden, mais sa série génératrice est rationnelle, et a donc un nombre fini de singularités. La proposition suivante fournit pour l'autre direction un exemple de langage finiment ambigu dont la série génératrice possède une infinité de singularités.

Proposition 7.21 (Un langage 2-ambigu dont la série génératrice a une infinité de singularités, [Fla87]).
▶ Le langage $P_2 := \{\underline{n}_1 \dots \underline{n}_k \mid (n_1 = 1 \text{ et } \forall j, n_{2j} = 2n_{2j-1}) \text{ ou } (\forall j, n_{2j+1} = 2n_{2j})\}$ est intrinsèquement ambigu. ◀

Démonstration. Remarquons que le langage P_2 est l'union de deux langages déterministes $L_1 = \{\underline{n}_1 \dots \underline{n}_k \,|\, n_1 = 1 \text{ et } \forall j, n_{2j} = 2n_{2j-1}\}$ et $L_2 = \{\underline{n}_1 \dots \underline{n}_k \,|\, \forall j, n_{2j+1} = 2n_{2j}\}$, qui sont donc non ambigus et ont une série génératrice algébrique.

Notons $I = L_1 \cap L_2$, et $P_2(a,b)$, $L_1(a,b)$, $L_2(a,b)$ et $I(a,b)$ les séries génératrices bivariées des langages P_2, L_1, L_2 et I.

Alors $P_2(a,b) = L_1(a,b) + L_2(a,b) - I(a,b)$ (il faut retirer $I(a,b)$ car sinon on compte deux fois les mots de l'intersection). Comme $L_1(a,b) + L_2(a,b)$ est algébrique, par propriétés de clôture, si $P_2(a,b)$ est algébrique, alors $I(a,b)$ l'est aussi.

En vérifiant que $I = \{aba^2ba^4b \dots a^{2^p}b \mid p \in \mathbb{N}\}$, on peut calculer la série génératrice $I(a,b) = \sum_{p \geqslant 1} b^p a^{2^{p-1}}$. Par clôture, si $I(a,b)$ était algébrique, alors $1 + xI(x,1) = \sum_{n \geqslant 0} x^{2^n}$ serait algébrique. Mais cette série est lacunaire, donc transcendante. Donc $P_2(a,b)$ est transcendante, et le langage P_2 est intrinsèquement ambigu. ■

Remarque 7.22.
▶ Cette introduction n'a pas vocation à être exhaustive ; j'ai seulement cherché à exposer certaines techniques de preuves d'intrinsèque ambiguïté de langages algébriques. J'ai trouvé par hasard un exemple d'autre technique de preuve dans les travaux de [Hon96, Theorem 5.1], portant sur certains langages algébriques non ambigus, appelés *Parikh slender* : un langage sur un alphabet à d lettres est dit *Parikh*

slender s'il existe une constante k telle que pour tout vecteur $\boldsymbol{v} \in \mathbb{N}^d$, le nombre de mots du langage ayant $\boldsymbol{v}$ pour image de Parikh est borné par k. En décrivant précisément la forme des langages algébriques *Parikh slender*, les auteurs démontrent alors que la série génératrice d'un tel langage non ambigu est rationnelle, et en déduisent un nouvel outil pour démontrer l'intrinsèque ambiguïté de certains langages. Cet outil peut en réalité être vu comme une conséquence des techniques de Flajolet, car toute série holonome dont les coefficients appartiennent à un ensemble fini de $\mathbb{N}$ est rationnelle (voir Lemme 7.90) : lorsque les auteurs prouvent une intrinsèque ambiguïté en montrant qu'une certaine série associée au langage n'est pas rationnelle, ils montrent en fait aussi que la série génératrice du langage n'est pas holonome. Il s'agit donc plus d'un raccourci pratique de preuve que d'un nouveau critère qui vienne étendre les techniques de Flajolet. ◀

7.1.3 Deux nouveaux critères d'intrinsèque ambiguïté portant sur des séries génératrices rationnelles.

Dans cette section nous proposons de dériver deux critères à partir de l'équivalence de Ginsburg et Ullian au Théorème 7.5, pour prouver facilement l'intrinsèque ambiguïté de certains langages algébriques bornés, dont la série est rationnelle. Les critères présentés ici ne sont pas exhaustifs, et ne sont que des conditions suffisantes pour prouver l'intrinsèque ambiguïté des langages bornés (à notre connaissance, le problème de décision de l'intrinsèque ambiguïté des langages algébriques bornés est encore ouvert). Néanmoins, ils permettent d'éviter grandement les preuves à la main d'itération dans des semilinéaires présentées dans [GU66, HU66] et de démontrer en quelques lignes seulement l'intrinsèque ambiguïté de certains langages bornés.

Dans la suite nous fixons un alphabet Σ de taille plus grande que 2, $d \geqslant 1$, ainsi qu'un tuple de d mots $\langle w \rangle = \langle w_1, \dots, w_d \rangle$. Nous rappelons la notation $f_{\langle w \rangle}$ qui désigne l'application de $\mathbb{N}^d$ dans $w_1^* \dots w_d^*$ définie par $f_{\langle w \rangle}(i_1, \dots, i_d) = w_1^{i_1} \dots w_n^{i_d}$. Nous rappelons que $f_{\langle w \rangle}$ n'a pas besoin d'être injective dans les critères de Ginsburg et Ullian. Nous rappelons aussi qu'un langage L est dit borné par rapport à $\langle w \rangle$ si $L \subseteq w_1^* \dots w_d^*$, et que si L est de plus algébrique, alors $f^{-1}_{\langle w \rangle}(L)$ est semilinéaire.

La proposition suivante propose un premier critère simple pour démontrer l'intrinsèque ambiguïté de certains langages bornés. La preuve s'appuie sur les critères de Ginsburg et Ullian, et un peu d'arithmétique dans $\mathbb{Q}[x_1, \dots, x_d]$, dont l'unicité de la décomposition en facteurs irréductibles.

À l'aide d'un logiciel de calcul formel, nous utiliserons ce critère pour démontrer en quelques lignes seulement que les langages $\{a^i b^j c^k$ avec $i = j$ ou $j = k\}$, $\{a^i b^j c^k$ avec $i \neq j$ ou $j \neq k\}$ et $\{a^i b^j c^k$ avec $i = j$ ou $j \neq k\}$ sont intrinsèquement ambigus.

Remarque 7.23 (Travaux relatifs).
▶ Le travail de cette section est la seule contribution nouvelle de cette thèse à cette introduction détaillée. Il n'a pas été publié, car il s'agit encore d'un travail en cours de réflexion.

Le premier critère de cette section (le Théorème 7.24) a été démontré indépendamment par [Mak21], dans le cas particulier des langages bornés sur des lettres distinctes, en utilisant des techniques différentes : l'auteur s'est intéressé aux grammaires $GF(2)$, une classe de grammaires hors-contextes pour lesquelles l'union est remplacée par la différence symétrique, et la concaténation de deux langages K et L est remplacée par une concaténation spéciale $K \odot L$ qui ne garde que les

mots w de $K \cdot L$ qui admettent un nombre impair de décompositions de la forme $w = w_k w_\ell$ avec $w_k \in K$ et $w_\ell \in L$. Ces conditions coïncident avec l'union usuelle et la concaténation usuelle pour les grammaires hors-contexte classiques non ambiguës : par non ambiguïté, tout mot du langage admet un unique arbre de dérivation, donc les unions dans les règles de la grammaire sont disjointes, et tout mot obtenu à partir d'une concaténation de symboles d'une règle de dérivation admet une unique décomposition sous cette forme (donc bien un nombre impair de décompositions). L'auteur étudie les séries génératrices associées aux langages reconnaissables par des $GF(2)$ qui sont inclus dans $a_1^* \dots a_d^*$, avec $a_1, \dots, a_d$ des symboles distincts, et démontre ainsi que les polynômes irréductibles à leur dénominateur ont au plus deux variables, ce qui aboutit au premier critère de cette section (dans le cas où $w_1, \dots, w_d$ sont des lettres distinctes). L'auteur démontre notamment avec ce critère l'intrinsèque ambiguïté du langage $L := \{a^i b^j c^k \text{ avec } i \neq j \text{ ou } j \neq k\}$.

En supposant qu'on a retiré les symboles non productifs, et les symboles non accessibles à partir de l'axiome S.

[Mak21] cite l'article de Ginsburg et Ullian [GU66], en disant qu'il serait possible d'utiliser leurs critères pour démontrer l'intrinsèque ambiguïté du langage L, mais explique que la preuve ne serait pas plus simple. Je ne suis pas sûr que l'auteur ait remarqué que l'équivalence de Ginsburg et Ullian se traduit en fait facilement en des critères analogues aux siens sur les séries génératrices associées aux langages algébriques bornés, et permet de plus de gérer les langages bornés sur des mots, et non uniquement des symboles distincts, ainsi que l'intrinsèque ambiguïté liée à l'entrelacement des lettres (ce que nous verrons au Théorème 7.32). ◀

Théorème 7.24 (Un premier critère pour montrer l'intrinsèque ambiguïté).
▶ Soit $L \subseteq w_1^* \dots w_d^*$ un langage algébrique borné par rapport à $\langle w \rangle$. Soit $S = f_{\langle w \rangle}^{-1}(L)$ son ensemble semilinéaire associé, et nous notons

$$S(x_1, \dots, x_d) = \frac{P(x_1, \dots, x_d)}{Q(x_1, \dots, x_d)} \in \mathbb{Q}(x_1, \dots, x_d)$$

la série génératrice de S, telle que P et Q sont des polynômes de $\mathbb{Q}[x_1, \dots, x_d]$ (pas forcément premiers entre eux). Supposons qu'il existe un polynôme irréductible $D \in \mathbb{Q}[x_1, \dots, x_d]$ qui divise Q, mais ne divise pas P, et tel que $D \notin \mathbb{Q}[x_i, x_j]$ pour tout $1 \leqslant i, j \leqslant d$ (autrement dit D est un polynôme qui fait intervenir au moins trois variables). Alors L est intrinsèquement ambigu. ◀

Démonstration. Supposons que L n'est pas ambigu. Par les critères de Ginsburg et Ullian du Théorème 7.5, le semilinéaire S peut s'écrire sous la forme

$$S = \biguplus_{i=1}^{r} (\boldsymbol{c}_i + P_i^*)$$

où l'union est disjointe, chaque P_i est stratifié, et les vecteurs dans chaque ensemble de périodes P_i sont linéairement indépendants.

L'union disjointe ainsi que l'indépendance des périodes signifie que cette représentation est une représentation non ambiguë de S. Par conséquent, sa série génératrice est donnée par :

$$S(\boldsymbol{x}) = \sum_{i=1}^{r} \frac{\boldsymbol{x}^{\boldsymbol{c}_i}}{\prod_{\boldsymbol{p} \in P_i} (1 - \boldsymbol{x}^{\boldsymbol{p}})} = \frac{P_2(\boldsymbol{x})}{Q_2(\boldsymbol{x})}$$

avec $Q_2(\boldsymbol{x}) = \prod_{i=1}^{r} \prod_{\boldsymbol{p} \in P_i} (1 - \boldsymbol{x}^{\boldsymbol{p}})$.

Ainsi $PQ_2 = P_2Q$. Le polynôme irréductible D divise Q, donc il divise PQ_2 ; comme $D \wedge P = 1$, il divise Q_2.

Cependant, comme S est stratifié, aucune période $\boldsymbol{p}$ dans les P_i n'a strictement plus de deux coordonnées non nulles. Cela signifie que Q_2 est un produit de polynômes de la forme $(1 - t)$ où t est un monôme à au plus deux variables. Chacun de ces monômes admet une factorisation unique en polynômes irréductibles, qui ont donc tous au plus deux variables. Par unicité de la factorisation en irréductibles dans $\mathbb{Q}[x_1, \ldots, x_d]$, D ne peut pas diviser Q_2. Contradiction. ■

Remarque 7.25 .
▶ Comme tout semilinéaire possède une représentation non ambiguë ([ES69] et [Ito69]), il est toujours possible de calculer la série génératrice d'un ensemble semi-linéaire. Comme pour les critères de Flajolet, c'est la non ambiguïté qui permet de faire le lien entre théorie des langages et séries génératrices. ◀

Ce premier critère permet de démontrer l'intrinsèque ambiguïté des langages suivants :

Proposition 7.26 .
▶ Les langages algébriques suivants sont intrinsèquement ambigus :

a. $\{a^i b^j c^k$ avec $i = j$ ou $j = k\}$

b. $\{a^i b^j c^k$ avec $i \neq j$ ou $j \neq k\}$

c. $\{a^i b^j c^k$ avec $i = j$ ou $j \neq k\}$

◀

Démonstration. Il suffit de calculer la série génératrice associée au semilinéaire des langages étudiés, de la mettre sous une forme irréductible à l'aide d'un logiciel de calcul formel, et de montrer la présence d'un facteur irréductible faisant intervenir au moins trois variables au dénominateur :

a. La série génératrice du semilinéaire est

$$\begin{aligned}&\tfrac{1}{(1-ab)(1-c)} + \tfrac{1}{(1-bc)(1-a)} - \tfrac{1}{1-abc}\\ =&\tfrac{1-3\,a^2b^2c^2+2\,a^2b^2c+2\,ab^2c^2+2\,a^2bc-ab^2c+2\,abc^2-a^2b+2\,abc-bc^2-ac}{(1-a)(1-bc)(1-c)(1-ab)(1-abc)}\end{aligned}$$

Le polynôme $1-abc$ au dénominateur est irréductible dans $\mathbb{Q}[a, b, c]$, et implique trois variables. De plus, $(1 - abc)$ ne divise pas le numérateur (on le vérifie avec Maple, ou à la main : dans $\mathbb{Q}[a, b][c]$, le numérateur est de degré 2 en c, donc si $(1 - abc)$ le divisait, il serait de la forme $(1 - abc)(\lambda c + \mu)$ avec $\lambda, \mu \in \mathbb{Q}[a, b]$, si bien que chaque terme en c^2 doit avoir ab en facteur, ce qui n'est pas le cas du terme $-bc^2$ au numérateur).

b. La série génératrice du semilinéaire est

$$\tfrac{1}{(1-a)(1-b)(1-c)} - \tfrac{1}{1-abc} = \tfrac{a+b+c-ab-ac-bc}{(1-a)(1-b)(1-c)(1-abc)}$$

Le polynôme irréductible $1 - abc$ implique trois variables, et ne divise pas le numérateur puisqu'il est de degré total 3, tandis que le numérateur est de degré total 2.

c. La série génératrice du semilinéaire est

$$\frac{1}{(1-a)(1-b)(1-c)} - \left(\frac{1}{(1-a)(1-bc)} - \frac{1}{1-abc}\right) = \frac{3\,ab^2c^2 - 2\,ab^2c - 2\,abc^2 - b^2c^2 + b^2c + bc^2 + ab + ac - 2\,bc - a + 1}{(1-a)(1-b)(1-c)(1-bc)(1-abc)}$$

Là encore, le terme irréductible $1 - abc$ implique trois variables et ne divise pas le numérateur à cause du terme bc^2. ■

Cette technique simplifie aussi la preuve d'intrinsèque ambiguïté du Théorème 6.2 de Ginsburg and Ullian [GU66], qui porte sur un langage borné dont les symboles ne sont pas distincts :

Proposition 7.27 ([GU66, Theorem 6.2]).
▶ Pour tout mot $w \in \{a, b\}^+$, le langage algébrique $L := \{a^i bwba^j ba^k \;:\; i = j \text{ ou } j = k,\ i, j, k \geqslant 1\}$ est intrinsèquement ambigu. ◀

Démonstration. Le langage est borné par rapport à $\langle a, bwb, a, b, a\rangle$. La série associée à l'ensemble semilinéaire est (avec x, y, z associés à chaque groupes de a, et α et β à bwb et b) :

$$\alpha\beta xyz\left(\frac{1}{(1-xy)(1-z)} + \frac{1}{(1-yz)(1-x)} - \frac{1}{1-xyz}\right)$$

La fraction entre parenthèses a déjà été étudiée dans la proposition précédente, son dénominateur a un facteur irréductible $(1 - xyz)$ qui ne divise pas son numérateur. Et $(1 - xyz)$ ne divise pas non plus $\alpha\beta xyz$. Donc L est intrinsèquement ambigu. ■

Le critère du Théorème 7.24 n'exploite pas la condition de non entrelacement. Il échoue sur le langage $\{a^n b^m a^p b^q \mid n = p \text{ ou } m = q\}$ dont la série génératrice du semilinéaire ne contient à son dénominateur que des polynômes irréductibles à au plus deux variables. Le but de la fin de cette section est d'établir dans la Théorème 7.32 un deuxième critère qui prend en compte l'entrelacement. Avant cela, nous avons besoin de plusieurs lemmes techniques : le Lemme 7.28 et le Lemme 7.29 sont deux lemmes classiques d'algèbre sur les polynômes ; le Lemme 7.30 et le Lemme 7.31 étudient plus précisément la forme des polynômes irréductibles qui divisent les dénominateurs des séries associées à des ensembles linéaires stratifiés.

Lemme 7.28 .
▶ Si $f \in \mathbb{Q}[x, y]$ est homogène, alors tout diviseur de f est homogène. ◀

Démonstration. Factorisons $f = gh$, avec $g, h \in \mathbb{Q}[x, y]$. Nous décomposons $g = \sum_{i=s}^{r} g_i$ et $h = \sum_{i=s'}^{r'} h_i$ sous la forme d'une somme de polynômes homogènes, où pour tout i, h_i et g_i sont soit nuls, soit de degré total i. De plus, nous supposons que $g_s, g_r, h_{s'}$ et $h_{r'}$ sont non nuls. Nous avons donc $f = (\sum_{i=s}^{r} g_i)(\sum_{i=s'}^{r'} h_i)$. Le terme de plus haut degré total de f est $g_r h_{r'}$, de degré total $r + r'$, et celui de plus bas degré est $g_s h_{s'}$, de degré total $s + s'$. Comme f est homogène, forcément $r + r' = s + s'$, et comme $s \leqslant r$ et $s' \leqslant r'$, $s = r$ et $s' = r'$, autrement dit g et h sont homogènes. ■

Lemme 7.29 (Irréductibilité de $1 - x^n y^m$).
▶ Soit $n, m \in \mathbb{N}$. Le polynôme $1 - x^n y^m$ est irréductible dans $\mathbb{Q}[x, y]$ si et seulement si $n \wedge m = 1$. ◀

Démonstration. Si n et m ne sont pas premiers entre eux, soit $\delta > 1$ un diviseur commun. Alors $1 - x^n y^m = 1 - (x^{n/\delta} y^{m/\delta})^\delta = (1 - x^{n/\delta} y^{m/\delta}) \sum_{k=1}^{\delta-1} x^{kn/\delta} y^{km/\delta}$ n'est pas irréductible.

Si n et m sont premiers entre eux, nous nous inspirons de la jolie preuve de [Ele], en la détériorant un peu (nous la rendons un peu plus élémentaire, mais moins élégante). Notons $f = 1 - x^n y^m$. Supposons que $f = gh$.

Sans perte de généralité, $g = (a_0 + \ldots + a_{r'} x^r y^{r'})$ avec $a_0 \neq 0$, $a_{r'} \neq 0$, r' le degré de g en y, et r le degré du polynôme en x qui est le coefficient de $y^{r'}$. De même, $h = (a_0^{-1} + \ldots + -a_{r'} x^s y^{s'})$ avec $a_0 \neq 0$, $a_{r'} \neq 0$, s' le degré de h en y, et s le degré du coefficient de $y^{s'}$. Nous avons ainsi $r' + s' = m$, et nous pouvons supposer sans perte de généralité que $r' \neq 0$.

Le polynôme

$$Y^{nm} - X^{nm} = Y^{nm} f(X^m, Y^{-n}) = Y^{nr'} g(X^m, Y^{-n}) Y^{ns'} h(X^m, Y^{-n})$$

est homogène. Comme $Y^{nr'} g(X^m, Y^{-n})$ et $Y^{ns'} h(X^m, Y^{-n})$ sont des polynômes de $\mathbb{Q}[X, Y]$, ils sont homogènes par le lemme précédent.

Donc $a_0 Y^{nr'} + \ldots + a_{r'} X^{mr}$ est homogène, donc $mr = nr'$. Comme m et n sont premiers entre eux, m divise r', et comme $r' \neq 0$, $m \leqslant r'$, donc $m = r'$ et $s' = 0$. Donc $h(x, y)$ est un polynôme $\tilde{h}(x)$ en x uniquement, mais $Y^{ns'} h(X^m, Y^{-n}) = Y^{ns'} \tilde{h}(X^m)$ est homogène, donc $\tilde{h}(X^m)$ est homogène. Comme $a_0 \neq 0$, h est une constante. Donc f est bien irréductible. ∎

Les deux lemmes suivants étudient la forme des polynômes irréductibles qui divisent un polynôme de la forme $\prod_{\boldsymbol{p} \in P} (1 - \boldsymbol{x}^{\boldsymbol{p}})$.

Lemme 7.30 .
▶ Soit $n, m \in \mathbb{N}^*$. Alors on peut écrire $1 - x^n y^m = (1 - x^\alpha y^\beta) P(x, y)$ où $\alpha \wedge \beta = 1$, et $P(x, y)$ est un polynôme non nul dont les coefficients sont dans $\{0, 1\}$. De plus $\alpha = n/(n \wedge m)$ et $\beta = m/(n \wedge m)$. ◀

Démonstration. Posons $\delta = n \wedge m$. Alors $1 - x^n y^m = (1 - x^{n/\delta} y^{m/\delta}) P(x, y)$, avec $P(x, y) = \sum_{k=1}^{\delta-1} x^{kn/\delta} y^{km/\delta}$ est un polynôme à coefficients dans $\{0, 1\}$. Par définition du pgcd, $(n/\delta) \wedge (m/\delta) = 1$. ∎

Lemme 7.31 .
▶ Soit $S = \boldsymbol{c} + P^*$ un ensemble linéaire stratifié à périodes linéairement indépendantes. Soient $k \geqslant 1$ un entier, $n, m \geqslant 1$ deux entiers tels que $n \wedge m = 1$, et $i \neq j$ deux indices de variables. Alors :

- si $(1 - x_i^n x_j^m)^k \mid \prod_{\boldsymbol{p} \in P} (1 - \boldsymbol{x}^{\boldsymbol{p}})$, alors $k = 1$;
- si $(1 - x_i^n x_j^m) \not\mid \prod_{\boldsymbol{p} \in P} (1 - \boldsymbol{x}^{\boldsymbol{p}})$, alors $\prod_{\boldsymbol{p} \in P} (1 - \boldsymbol{x}^{\boldsymbol{p}})|_{x_i = a^m, x_j = a^{-n}} \neq 0$ pour tout $a \in \mathbb{Q}_+^* \setminus \{1\}$.

◀

Démonstration. Comme les périodes sont linéairement indépendantes, il existe au plus deux périodes $\boldsymbol{p_1}, \boldsymbol{p_2} \in P$ telles que $(1 - \boldsymbol{x}^{\boldsymbol{p_1}})$ et $(1 - \boldsymbol{x}^{\boldsymbol{p_2}})$ sont dans $\mathbb{Q}[x_i, x_j]$.

Écrivons $(1 - \boldsymbol{x}^{\boldsymbol{p_1}}) = (1 - x_i^{n_1} x_j^{m_1}) = (1 - x_i^{n_1/d_1} x_j^{m_1/d_1}) P_1(x_i, x_j)$, avec $n_1, m_1 \geqslant 1$, $P_1(x_i, x_j)$ qui est non nul à coefficients dans $\{0, 1\}$, et $d_1 = n_1 \wedge m_1$. De même, notons $(1 - \boldsymbol{x}^{\boldsymbol{p_2}}) = (1 - x_i^{n_2} x_j^{m_2}) = (1 - x_i^{n_2/d_2} x_j^{m_2/d_2}) P_2(x_i, x_j)$ avec les mêmes conditions et notations.

Comme $P_1(1,1)$ ne peut pas être nul, P_1 n'est divisible par aucune polynôme de la forme $(1-\boldsymbol{x}^{\boldsymbol{p}})$ (et de même pour P_2). De plus, les autres facteurs au dénominateur $\prod_{\boldsymbol{p}\in P}(1-\boldsymbol{x}^{\boldsymbol{p}})$ ne sont pas divisibles par les polynômes irréductibles $(1-x_i^{n_1/d_1}x_j^{m_1/d_1})$ et $(1-x_i^{n_2/d_2}x_j^{m_2/d_2})$, car ils ne dépendent pas à la fois de x_i et de x_j.

Enfin, $(n_1/d_1, m_1/d_1) \neq (n_2/d_2, m_2/d_2)$, car sinon nous aurions $d_2\boldsymbol{p_1} = d_1\boldsymbol{p_2}$, et donc $\boldsymbol{p_1}$ et $\boldsymbol{p_2}$ seraient liés.

En conclusion, tout polynôme irréductible de la forme $(1-x_i^n x_j^m)$ avec $n\wedge m = 1$ a une multiplicité au plus 1 dans la décomposition en facteurs irréductibles de $\prod_{\boldsymbol{p}\in P}(1-\boldsymbol{x}^{\boldsymbol{p}})$. De plus, les autres facteurs irréductibles qui dépendent à la fois de x_i et x_j sont des diviseurs de polynômes à coefficients dans $\{0,1\}$. Le premier point est prouvé.

Le second point de la proposition vient de ce qui précède et des observations suivantes :

— un polynôme non nul à coefficients dans $\{0,1\}$, et donc a fortiori ses diviseurs, ne devient pas nul en remplaçant une de ses variables par un nombre strictement positif.

— si a n'est pas une racine de l'unité (en particulier si $a \in \mathbb{Q}_+^* \setminus \{1\}$), un polynôme univarié de la forme $(1-x_s^{p_s})$ ne s'annule pas en $x_s = a$;

— un polynôme de la forme $(1-x_t^{p_t}x_s^{p_s})$ avec $p_t, p_s \geqslant 1$ et $\{x_t, x_s\} \neq \{x_i, x_j\}$ ne devient pas le polynôme nul en remplaçant x_i et x_j par des rationnels non nuls.

Soit $a \in \mathbb{Q}_+^* \setminus \{1\}$. Les seuls facteurs irréductibles de $\prod_{\boldsymbol{p}\in P}(1-\boldsymbol{x}^{\boldsymbol{p}})$ susceptibles de s'annuler avec la substitution $x_i = a^m, x_j = a^{-n}$ sont donc de la forme $(1-x_i^{n_1}x_j^{m_1})$ avec $n_1, n_2 \geqslant 1$, $n_1 \wedge n_2 = 1$ et $(n,m) \neq (n_1, n_2)$. Alors, la substitution remplace ce polynôme par le rationnel $(1-a^{mn_1-nm_1})$, qui est nul si et seulement si $mn_1 - nm_1 = 0$, si et seulement si $n = n_1$ et $m = m_1$ (car $n \wedge m = 1$ et $n_1 \wedge n_2 = 1$). Donc aucun facteur irréductible de $\prod_{\boldsymbol{p}\in P}(1-\boldsymbol{x}^{\boldsymbol{p}})$ ne s'annule par la substitution, donc $\prod_{\boldsymbol{p}\in P}(1-\boldsymbol{x}^{\boldsymbol{p}})|_{x_i=a^m, x_j=a^{-n}} \neq 0$. ∎

La proposition suivante constitue notre second critère pour démontrer l'intrinsèque ambiguïté de certains langages bornés :

Théorème 7.32 (Second critère d'intrinsèque ambiguïté).
▶ Soit $L \subseteq w_1^* \dots w_d^*$ un langage algébrique borné par rapport à $\langle w\rangle$. Notons $S = f_{\langle w\rangle}^{-1}(L)$ l'ensemble semilinéaire associé, et $S(x_1,\dots,x_d) = \frac{P(x_1,\dots,x_d)}{Q(x_1,\dots,x_d)} \in \mathbb{Q}[x_1,\dots,x_d]$ sa série génératrice, avec P/Q qui n'est pas forcément complètement sous la forme d'une fraction irréductible. Supposons de plus que :

a. Q est divisible par deux polynômes irréductibles non univariés $D(x_j, x_\ell)$ et $\pi(x_i, x_k)$ avec $j < \ell$ et $i < k$ entrelacés (i.e. $i < j < k < \ell$ ou $j < i < \ell < k$).

b. $\pi(x_i, x_k)$ est de la forme $\pi(x_i, x_k) = (1-x_i^n x_k^m)$, avec $n, m \geqslant 1$ et $n \wedge m = 1$;

c. enfin, il existe un rationnel $\alpha \in \mathbb{Q}_+^* \setminus \{1\}$ tel que $D \not| P|_{x_i=\alpha^m, x_k=\alpha^{-n}}$.

Alors L est intrinsèquement ambigu. ◀

Démonstration. Supposons que L est non ambigu. Par le Théorème 7.5, S peut s'écrire sous la forme

$$S = \biguplus_{s=1}^{r} (\boldsymbol{c}_s + P_s^*)$$

où l'union est disjointe, les périodes P_i sont stratifiées, et les vecteurs dans chaque P_i sont linéairement indépendants. Sa série génératrice s'écrit alors :

$$\frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})} = S(\boldsymbol{x}) = \sum_{s=1}^{r} \frac{\boldsymbol{x}^{\boldsymbol{c}_s}}{\prod_{\boldsymbol{p}\in P_s}(1-\boldsymbol{x}^{\boldsymbol{p}})} = \frac{\tilde{P}(\boldsymbol{x})}{\tilde{Q}(\boldsymbol{x})}$$

où l'on a réduit la somme de fractions au même dénominateur, puis mis la fraction obtenue sous forme irréductible. Ainsi $\tilde{P} \wedge \tilde{Q} = 1$, et $\tilde{Q}$ divise le plus petit multiple commun des $\prod_{\boldsymbol{p}\in P_s}(1-\boldsymbol{x}^{\boldsymbol{p}})$.

Par le Lemme 7.31, au plus un facteur irréductible de $\tilde{Q}$ devient nul après la substitution $x_i = \alpha^m, x_k = \alpha^{-n}$, et $\tilde{Q}$ devient nul après cette substitution si et seulement si $\pi(x_i, x_k) := (1 - x_i^n x_k^m)$ divise $\tilde{Q}$.

Par hypothèse, $P|_{x_i=\alpha^m, x_k=\alpha^{-n}} \neq 0$ (car D divise 0), donc $\pi(x_i, x_k)$ ne divise pas P. Comme π est irréductible, et π divise Q, l'égalité $\tilde{P}Q = P\tilde{Q}$ implique que π divise $\tilde{Q}$, et c'est donc son seul facteur irréductible qui s'annule en appliquant la substitution $x_i = \alpha^m, x_k = \alpha^{-n}$.

Écrivons $Q = \pi R$ et $\tilde{Q} = \pi\tilde{R}$, avec notamment $\tilde{R}|_{x_i=\alpha^m, x_k=\alpha^{-n}} \neq 0$. Comme $\tilde{P}R = P\tilde{R}$, ni $\tilde{P}$ ni R ne deviennent nuls après la substitution.

Alors en multipliant par π, nous obtenons l'égalité suivante :

$$\sum_{s=1}^{r} \frac{(1 - x_i^n x_k^m)\, \boldsymbol{x}^{\boldsymbol{c}_s}}{\prod_{\boldsymbol{p}\in P_s}(1-\boldsymbol{x}^{\boldsymbol{p}})} = \frac{P(\boldsymbol{x})}{R(\boldsymbol{x})}\,.$$

Posons I_1 l'ensemble des indices s tels que $(1 - x_i^n x_k^m) \mid \prod_{\boldsymbol{p}\in P_s}(1-\boldsymbol{x}^{\boldsymbol{p}})$, et I_2 son complémentaire.

Pour tout $s \in I_1$, nous notons $\prod_{\boldsymbol{p}\in P_s}(1-\boldsymbol{x}^{\boldsymbol{p}}) = (1 - x_i^n x_k^m)R_s(\boldsymbol{x})$. Par le Lemme 7.31, $R_s|_{x_i=\alpha^m, x_k=\alpha^{-n}} \neq 0$, et par la condition de non entrelacement, aucun facteur irréductible de R_s n'est un polynôme en exactement les deux variables x_j, x_ℓ. Ainsi, aucun facteur irréductible de $R_s|_{x_i=\alpha^m, x_k=\alpha^{-n}}$ n'est un polynôme en exactement les deux variables x_j et x_ℓ.

Pour tout $s \in I_2$, $\prod_{\boldsymbol{p}\in P_s}(1-\boldsymbol{x}^{\boldsymbol{p}})|_{x_i=\alpha^m, x_k=\alpha^{-n}} \neq 0$ puisque $\pi \not| \prod_{\boldsymbol{p}\in P_s}(1-\boldsymbol{x}^{\boldsymbol{p}})$, par le Lemme 7.31. Par conséquent, pour tout $s \in I_2$, $\left.\frac{\boldsymbol{x}^{\boldsymbol{c}_s}}{\prod_{\boldsymbol{p}\in P_s}(1-\boldsymbol{x}^{\boldsymbol{p}})}\right|_{x_i=\alpha^m, x_k=\alpha^{-n}}$ est une fraction rationnelle bien définie à coefficients dans $\mathbb{Q}$.

Ainsi en évaluant en $x_i = \alpha^m, x_k = \alpha^{-n}$ l'égalité :

$$\sum_{s\in I_1} \frac{\boldsymbol{x}^{\boldsymbol{c}_s}}{R_s(\boldsymbol{x})} + \pi(x_i, x_k) \sum_{s\in I_2} \frac{\boldsymbol{x}^{\boldsymbol{c}_s}}{\prod_{\boldsymbol{p}\in P_s}(1-\boldsymbol{x}^{\boldsymbol{p}})} = \frac{P(\boldsymbol{x})}{R(\boldsymbol{x})}\,,$$

nous obtenons :

$$\sum_{s\in I_1} \frac{x^{\boldsymbol{c}_s}|_{x_i=\alpha^m, x_k=\alpha^{-n}}}{R_s|_{x_i=\alpha^m, x_k=\alpha^{-n}}} = \frac{P|_{x_i=\alpha^m, x_k=\alpha^{-n}}}{R|_{x_i=\alpha^m, x_k=\alpha^{-n}}}\,.$$

Comme $D(x_j, x_\ell)$ est un polynôme irréductible en exactement les deux variables x_j et x_ℓ, il reste inchangé par la substitution, et donc divise $R|_{x_i=\alpha^m, x_k=\alpha^{-n}}$. Comme $D \not| P|_{x_i=\alpha^m, x_k=\alpha^{-n}}$, D reste un facteur irréductible du dénominateur de la fraction $\sum_{s\in I_1} \frac{x^{\boldsymbol{c}_s}|_{x_i=\alpha^m, x_k=\alpha^{-n}}}{R_s|_{x_i=\alpha^m, x_k=\alpha^{-n}}}$ mise sous forme irréductible.

Mais en réduisant cette somme au même dénominateur, puis en la mettant sous forme irréductible, on voit que les les facteurs irréductibles du dénominateur sont des

diviseurs irréductibles des polynômes $R_s|_{x_i=\alpha^m, x_k=\alpha^{-n}}$ avec $s \in I_1$. Or nous avons démontré qu'aucun de ces polynômes n'admet de facteur irréductible dépendant exactement des deux variables x_j et x_ℓ. Nous obtenons donc une contradiction.

Donc L est intrinsèquement ambigu. ■

Remarque 7.33.

▶ Je n'ai pas encore eu le temps de bien mûrir ce critère. Je pense que la dernière hypothèse peut être remplacée par la condition que P n'appartienne pas à $\langle \pi, D \rangle$, l'idéal engendré par π et D, mais je ne suis pas arrivé à adapter la preuve, il me manque sans doute un peu de connaissances mathématiques sur les espaces quotients de polynômes en plusieurs variables. Intuitivement, le fait d'évaluer en $x_i = \alpha^m, x_k = \alpha^{-n}$ revient à se placer dans un espace quotient pour lequel $\pi(x_i, x_k) = 0$. On pourrait chercher à adapter la preuve en se plaçant ainsi dans l'espace des fractions rationnelles en les variables x_s pour $s \in [1, d] \setminus \{i, k\}$, à coefficients dans l'anneau $\mathbb{Q}[x_i, x_k]/(\pi)$, mais ce dernier n'est pas factoriel, donc les raisonnements sur les facteurs irréductibles ne s'adaptent pas a priori. Je suis donc resté sur une version avec un argument par évaluation, qui reste dans un anneau factoriel. ◀

Nous pouvons alors utiliser ce critère pour démontrer l'intrinsèque ambiguïté des langages suivants :

Proposition 7.34.

▶ Les langages algébriques suivants sont intrinsèquement ambigus :

a. $L_1 = \{a^i b^j c^k d^\ell \ : \ i = k \text{ ou } j = \ell\}$

b. $L_2 = \{a^i b^j c^k d^\ell \ : \ i \neq k \text{ ou } j \neq \ell\}$

c. $L_3 = \{a^i b^j c^k d^\ell \ : \ i = k \text{ ou } j \neq \ell\}$ (et de façon similaire $L_4 = \{a^i b^j c^k d^\ell \ : \ i \neq k \text{ ou } j = \ell\}$)

Pour changer un peu des semilinéaires avec des égalités simples.

d. $L_2' = \{a^i b^j c^k d^\ell \ : \ 3i \neq 5k \text{ ou } 2j \neq 3\ell\}$

◀

Démonstration. **a.** Calculons la série génératrice du semilinéaire :

$$\frac{1}{(1-ac)(1-b)(1-d)} + \frac{1}{(1-bd)(1-a)(1-c)} - \frac{1}{(1-ac)(1-bd)}$$
$$= \frac{1-ab-ac-ad-bc-bd-cd+2\,abc+2\,abd+2\,acd+2\,bcd-3\,abcd}{(1-ac)(1-bd)(1-a)(1-b)(1-c)(1-d)}$$

Nous posons $D(b, d) = (1 - bd)$ et $\pi(a, c) = (1 - ac)$ qui sont tous les deux irréductibles, et leurs variables sont entrelacées. Soit $P = 1 - ab - ac - ad - bc - bd - cd + 2\,abc + 2\,abd + 2\,acd + 2\,bcd - 3\,abcd$.

Nous pourrions même prendre $\alpha = -1$ dans ce cas, car aucun autre polynôme ne devient nul en -1.

Posons $\alpha = 2$. Alors $P|_{a=2,c=1/2} = bd - b/2 - d/2$ et comme $(1 - bd)$ et $(bd - b/2 - d/2)$ sont tous deux de degré 1 en b, $(1 - bd) \not| (bd - b/2 - d/2)$.

En appliquant le Théorème 7.32, le langage L_1 est intrinsèquement ambigu.

b. La série génératrice du semilinéaire associé au langage est :

$$\frac{1}{(1-a)(1-b)(1-c)(1-d)} - \frac{1}{(1-ac)(1-bd)} = \frac{abc+abd+acd+bcd-ab-2\,ac-ad-bc-2\,bd-cd+a+b+c+d}{(1-ac)(1-bd)(1-a)(1-b)(1-c)(1-d)}$$

Posons $\pi = (1 - ac)$, $D = (1 - bd)$, qui sont irréductibles, et P le numérateur. Comme $(1 - bd) \not| P|_{a=2,c=1/2} = \frac{1}{2}(bd - b - d - 1)$, L_2 est intrinsèquement ambigu par le Théorème 7.32.

c. La série génératrice du semilinéaire associé au langage est :

Car $i = k \vee j \neq \ell \equiv \neg(\neg(i = k) \wedge j = \ell$

$$\frac{1}{(1-a)(1-b)(1-c)(1-d)} - \frac{1}{1-bd}\left(\frac{1}{(1-a)(1-c)} - \frac{1}{(1-ac)}\right)$$
$$= \frac{3\,abcd - 2\,abc - abd - 2\,acd - bcd + ab + ac + ad + bc - bd + cd - a - c + 1}{(1-a)(1-b)(1-c)(1-d)(1-bd)(1-ac)}$$

Posons $\pi = (1 - ac)$ et $D = (1 - bd)$, qui sont irréductibles, et P le numérateur Comme $(1 - bd) \not| P|_{a=2,c=1/2} = \frac{1}{2}(-bd + b + d - 1)$, L_3 est intrinsèquement ambigu.

d. La série génératrice du semilinéaire associé au langage est :

$$\frac{1}{(1-a)(1-b)(1-c)(1-d)} - \frac{1}{(1-b^3d^2)(1-a^5c^3)}$$
$$= \frac{a^5b^3c^3d^2 - a^5c^3 - b^3d^2 - abcd + abc + abd + acd + bcd - ab - ac - ad - bc - bd - cd + a + b + c + d}{(1-a)(1-b)(1-c)(1-d)(1-b^3d^2)(1-a^5c^3)}$$

Posons $\pi = (1 - a^5c^3)$ et $D = (1 - b^3d^2)$, qui sont irréductibles, et P le numérateur.

Posons $\alpha = 2$. Comme $(1 - b^3d^2) \not| P|_{a=8,c=1/32} = \frac{217}{32}(bd - b - d + 1)$, L_2' est intrinsèquement ambigu.

■

Remarque 7.35 .

▶ Les preuves précédentes sont fondées sur la forme du semilinéaire, donc fonctionnent aussi pour des variantes comme $\{a^iba^jba^kba^\ell b \mid i \neq k \text{ ou } j \neq \ell\}$. ◀

Remarque 7.36 .

▶ La question de la décidabilité de l'intrinsèque ambiguïté des langages algébriques bornés semble être toujours ouverte. Il serait intéressant de trouver des semilinéaires qui ne vérifient pas les critères de Ginsburg et Ullian, mais dont la série génératrice vérifie nos deux critères. ◀

7.1.4 Plan du chapitre

Dans cette introduction détaillée, nous avons présenté de façon non exhaustive quelques techniques existantes, dont certaines sont récentes, pour démontrer l'intrinsèque ambiguïté des langages algébriques. Le but de ce chapitre est de s'inspirer des outils présentés ici pour aborder le problème de l'intrinsèque faible ambiguïté des langages (algébriques) de Parikh.

— Dans la section 7.2, nous démontrons que la série génératrice d'un langage (algébrique) de Parikh faiblement non ambigu est holonome.

— En section 7.3, nous nous intéressons au problème de l'intrinsèque faible ambiguïté des langages (algébriques) de Parikh. Nous utilisons notamment la section précédente pour généraliser les techniques analytiques de Flajolet, pour prouver l'intrinsèque faible ambiguïté de plusieurs langages. Nous montrons que le problème est indécidable en général, puis nous terminons par une preuve utilisant un argument d'itération, pour donner un exemple de langage algébrique déterministe qui est un langage de Parikh intrinsèquement faiblement ambigu.

— La section 7.4 ouvre le sujet, en présentant plusieurs pistes et idées que nous j'ai eues pendant la thèse, mais qui n'ont pas encore pleinement abouti.

7.2 Langages algébriques de Parikh faiblement non ambigus et séries holonomes

Le but de cette section est de démontrer le lien entre les automates de Parikh à pile faiblement non ambigus et les séries holonomes. Pour les automates de Parikh sans pile, ce résultat se déduit de l'équivalence démontrée au Théorème 6.41 entre les langages de Parikh faiblement non ambigus et la classe RCM : dans [CM17], les auteurs démontrent en effet que la série génératrice des langages de RCM est holonome (la classe a même été construite dans l'objectif d'avoir une série holonome).

7.2.1 Stabilité des séries holonomes

Dans cette section, nous présentons quelques opérations utiles qui laissent stable la classe des séries holonomes. Le lecteur est invité à se reporter au chapitre des préliminaires pour un rappel sur les définitions et premières propriétés des séries holonomes. Nous rappelons que la classe des séries multivariées rationnelles et algébriques est strictement incluse dans celle des séries holonomes, et que les séries holonomes sont stables par addition, multiplication, substitution algébrique.

Définition 7.37 (Produit d'Hadamard).
▶ Le produit d'Hadamard de deux séries ayant le même nombre de variables est le produit terme à terme : si $A(x_1, \dots, x_k) := \sum_{\boldsymbol{n}\in\mathbb{N}^k} a(\boldsymbol{n})\boldsymbol{x^n}$ et $B(x_1, \dots, x_k) := \sum_{\boldsymbol{n}\in\mathbb{N}^k} b(\boldsymbol{n})\boldsymbol{x^n}$ sont des séries à d variables, le *produit d'Hadamard* des deux séries, noté $A \odot B$, est la série formelle définie par :

$$A \odot B(x_1, \dots, x_k) = \sum_{n_1,\dots,n_k\in\mathbb{N}^k} a(n_1, \dots, n_k)b(n_1, \dots, n_k)x_1^{n_1} \dots x_k^{n_k}.$$

◀

Remarque 7.38.
▶ Le support de la série $A \odot B$ est l'intersection des supports de A et B. ◀

Remarque 7.39.
▶ À une variable, il existe quelques propriétés de clôtures supplémentaires : les séries rationnelles univariées sont closes par produit d'Hadamard, et le produit d'Hadamard d'une série rationnelle et d'une série algébrique est algébrique [Jun31]. Mais ces clôtures ne se vérifient généralement pas à plusieurs variables. En particulier, le produit d'Hadamard de deux séries algébriques n'est pas nécessairement algébrique : la série $\frac{1}{1-(a+b+c)} = \sum_n (a+b+c)^n$ est la série génératrice des mots sur l'alphabet $\{a, b, c\}^*$. Comme $\frac{1}{1-abc} = \sum_n a^n b^n c^n$, la série $\frac{1}{1-(a+b+c)} \odot \frac{1}{1-abc}$ est la série génératrice des mots de $\{a, b, c\}^*$ qui ont le même nombre de a, de b et de c. Nous avons vu à la Proposition 7.19 que cette série n'est pas algébrique. ◀

Le théorème suivant a été démontré par Lipshitz en 1988 :

Théorème 7.40 (Clôture des séries holonomes par produit d'Hadamard, [Lip88]).
▶ Les séries holonomes sont closes par produit d'Hadamard. ◀

La proposition suivante est une conséquence de la stabilité des séries holonomes par substitution algébrique. Elle peut néanmoins se démontrer facilement à la main :

Proposition 7.41 (Stabilité par spécialisation).
▶ Soit $A(x_1, \dots, x_k, y_1, \dots, y_\ell)$ une série holonome telle que pour tout indice $(i_1, \dots, i_k)$, $[x_1^{i_1} \dots x_k^{i_k}]A$ est un polynôme en $y_1, \dots, y_\ell$.
Alors la série $B(x_1, \dots, x_k) := A(x_1, \dots, x_k, 1, \dots, 1)$ est holonome. ◀

Démonstration. Écrivons $A(x_1, \dots, x_k, y_1, \dots, y_\ell)$ sous la forme :

$$A(x_1, \dots, x_k, y_1, \dots, y_\ell) = \sum_{i_1, \dots, i_k} P_{i_1, \dots, i_k}(y_1, \dots, y_\ell) x_1^{i_1} \dots x_k^{i_k},$$

où chaque $P_{i_1,\dots,i_k}$ est un polynôme en les variables y_1, …, y_ℓ. Tout d'abord, nous observons que $B(x_1, \dots, x_k)$ est bien définie, puisque $[x_1^{i_1} \dots x_k^{i_k}]B$ est simplement l'évaluation du polynôme $P_{i_1,\dots,i_k}$ en $(y_1, \dots, y_\ell) = (1, \dots, 1)$.

Nous prouvons le résultat en remplaçant chaque variable y_i par 1, l'une après l'autre. Nous nous intéressons donc à l'opération $y_\ell = 1$.

Posons $C(x_1, \dots, x_k, y_1, \dots, y_{\ell-1}) := A(x_1, \dots, x_k, y_1, \dots, y_{\ell-1}, 1)$.

Par une récurrence directe, nous pouvons affirmer que pour tout entier $s \in \mathbb{N}$ et tout $z \in \{x_1, \dots, x_n, y_1, \dots, y_{\ell-1}\}$:

$$(\partial_z^s A)(x_1, \dots, x_k, y_1, \dots, y_{\ell-1}, 1) = \partial_z^s C(x_1, \dots, x_k, y_1, \dots, y_{\ell-1}).$$

Fixons une variable $z \in \{x_1, \dots, x_n, y_1, \dots, y_{\ell-1}\}$. La série A est holonome, donc vérifie une équation différentielle partielle en z de la forme :

$$p_s(\boldsymbol{x}, \boldsymbol{y})\partial_z^s A(\boldsymbol{x}, \boldsymbol{y}) + \dots + p_1(\boldsymbol{x}, \boldsymbol{y})\partial_z A(\boldsymbol{x}, \boldsymbol{y}) + p_0(\boldsymbol{x}, \boldsymbol{y})A(\boldsymbol{x}, \boldsymbol{y}) = 0$$

avec $p_0(\boldsymbol{x}, \boldsymbol{y}), \dots, p_s(\boldsymbol{x}, \boldsymbol{y})$ des polynômes à coefficients rationnels. Posons $\tilde{\boldsymbol{y}} = (y_1, \dots, y_{\ell-1})$ pour limiter la taille des équations.

Par ce qui précède, en évaluant l'équation en $y_\ell = 1$, nous obtenons une équation pour C :

$$p_s(\boldsymbol{x}, y_1, \dots, y_{\ell-1}, 1)\partial_z^s C(\boldsymbol{x}, \tilde{\boldsymbol{y}}) + \dots + p_0(\boldsymbol{x}, y_1, \dots, y_{\ell-1}, 1)C(\boldsymbol{x}, \tilde{\boldsymbol{y}}) = 0\,.$$

Cette dernière équation a cependant un problème : il se peut qu'elle soit triviale, de la forme $0 = 0$, si tous les polynômes $p_s \dots, p_0$ s'annulent en $y_\ell = 1$. Nous devons éviter cette possibilité.

Regardons chaque polynôme $p_s, \dots, p_0$ comme un polynôme sur $\mathbb{Q}(\boldsymbol{x}, \tilde{\boldsymbol{y}})[y_\ell]$. Dans cette anneau, un polynôme s'annule en $y_\ell = 1$ si et seulement si il est divisible par $y_\ell - 1$.

Soit m_0 le plus grand entier tel que $(y_\ell - 1)^{m_0}$ divise tous les polynômes $p_s, \dots, p_0$. Notons alors pour tout $i \in \{0, \dots, s\}$, $p_i = (y_\ell - 1)^{m_0} q_i$. Comme $(y_\ell - 1)^{m_0}$ est unitaire, il est possible d'obtenir q_i par une division euclidienne dans $\mathbb{Q}[\boldsymbol{x}, \tilde{\boldsymbol{y}}][y_\ell]$, ainsi q_i est bien dans $\mathbb{Q}[\boldsymbol{x}, \boldsymbol{y}]$. Par ailleurs par définition de m_0, il existe un indice i tel que q_i ne s'annule pas en $y_\ell = 1$.

En factorisant l'équation différentielle de A par $(y_\ell - 1)^{m_0}$, puis en la simplifiant par ce facteur, nous obtenons une nouvelle équation pour A :

$$q_s(\boldsymbol{x}, \boldsymbol{y})\partial_z^s A(\boldsymbol{x}, \boldsymbol{y}) + \dots + q_0(\boldsymbol{x}, \boldsymbol{y})A(\boldsymbol{x}, \boldsymbol{y}) = 0$$

qui cette fois fournit une équation non triviale pour C en l'évaluant en $y_\ell = 1$. ■

Remarque 7.42 .
▶ Souvent, les preuves de stabilité des séries holonomes pour la substitution algébrique oublient la possibilité que l'équation devienne triviale après substitution, et remplacent en une seule fois toutes les variables par leur valeurs. Dans la preuve précédente, l'intérêt de procéder variable par variable est de pouvoir assurer que si les polynômes s'annulent après la substitution $y_\ell = 1$, alors ils sont divisibles par une puissance de $y_\ell - 1$. Ce n'est plus vrai si on effectue plusieurs substitutions en même temps : par exemple le polynôme $(y_1 - y_2)$ devient nul par la substitution simultanée $y_1 = 1, y_2 = 1$ sans être divisible par $(y_1 - 1)$ ni $(y_2 - 1)$. ◀

7.2.2 Analogue du théorème de Chomsky-Schützenberger pour les automates de Parikh (à pile) faiblement non ambigus.

Dans cette section, nous montrons l'analogue du théorème de Chomsky-Schützenberger pour les automates de Parikh (à pile) faiblement non ambigus : la série génératrice d'un langage algébrique de Parikh faiblement non ambigu est holonome. Nous le démontrons d'abord dans le cas des langages de Parikh, qui est un peu plus simple à exposer.

Comme nous l'avons dit dans l'introduction de la partie, l'idée de la preuve remonte à l'article [Lip88] de stabilité des séries holonomes par produit d'Hadamard : Lipshitz avait déjà proposé de contraindre le support d'une série holonome par un ensemble de contraintes linéaires sur les occurrences des variables. Une preuve similaire à celle que nous présentons a été proposée dans [Mas93, CM17] pour les langages LCL et RCM, en utilisant une série similaire (mais qui utilise la substitution algébrique plus compliquée $x_i = x$ plutôt que la spécialisation à 1).

Le fait d'utiliser la spécialisation à 1 plutôt que la substitution algébrique $x_i = x$ simplifiera l'étude algorithmique du chapitre suivant.

La preuve que la série génératrice d'un langage de Parikh faiblement non ambigu est holonome repose sur le produit d'Hadamard de deux séries rationnelles : la première compte les calculs de l'automate, tandis que la deuxième décrit l'ensemble semilinéaire ; le produit d'Hadamard entre les deux séries permet de filtrer les calculs acceptants. Ces deux séries sont introduites dans le Lemme 7.43 et le Lemme 7.44.

Lemme 7.43 (Série génératrice des calculs d'un automate de Parikh).
▶ Soit $\mathcal{A}$ un automate de Parikh faiblement non ambigu de dimension d. Nous définissons la série génératrice pondérée $\overline{A}(x, y_1, \dots, y_d)$ par :

$$\overline{A}(x, y_1, \dots, y_d) := \sum_{(n,i_1,\dots,i_d)\in\mathbb{N}^{d+1}} \ell_{n,i_1,\dots,i_d} x^n y_1^{i_1} \dots y_d^{i_d}$$

où pour tout $(n, i_1, \dots, i_d) \in \mathbb{N}^{d+1}$, $\ell_{n,i_1,\dots,i_d}$ compte le nombre de calculs de $\mathcal{A}$ joignant l'état initial à un état final, étiquetés par un mot de longueur n, et le vecteur $(i_1, \dots, i_d)$. Alors $\overline{A}(x, y_1, \dots, y_d)$ est rationnelle. ◀

Ébauche de preuve. La preuve est une simple application de [FS09, Proposition V.6] qui est un résultat plus général portant sur les séries génératrices comptant les chemins d'un digraphe pondéré. Précisons que la variable n compte bien la longueur des chemins dans $\mathcal{A}$, car $\mathcal{A}$ est sans ε-transitions. Notamment $\ell_{n,i_1,\dots,i_d}$ est bien fini pour tout $(n, i_1, \dots, i_d) \in \mathbb{N}^{d+1}$.

Le résultat est bien connu pour les automates finis classiques. Une autre façon de démontrer le lemme, en partant du résultat connu sur les automates finis, est de transformer l'automate de Parikh en automate fini, en introduisant d nouvelles

lettres $b_1, \ldots, b_d$, et en remplaçant toute transition étiquetée par le couple $(a, \boldsymbol{i})$ par une transition joignant les mêmes états, mais étiquetée par le mot $ab_1^{i_1} \ldots b_d^{i_d}$ (quitte à ajouter des états supplémentaires si on souhaite un automate fini sur des lettres plutôt que sur des mots).

Nous détaillerons la preuve au chapitre suivant, dans le Lemme 8.12, en décrivant en plus des bornes sur le degré et la taille des coefficients des polynômes intervenant dans la fraction rationnelle représentant $\overline{A}(x, y_1, \ldots, y_d)$. ■

Lemme 7.44 (Série génératrice d'un ensemble semilinéaire).
▶ Soit C un ensemble semilinéaire sur $\mathbb{N}^d$. Nous posons $C(y_1, \ldots, y_d)$ sa série génératrice définie par :

$$C(y_1, \ldots, y_d) = \sum_{\boldsymbol{i} \in C} y_1^{i_1} \ldots y_d^{i_d} .$$

Alors $C(y_1, \ldots, y_d)$ est rationnelle. ◀

Démonstration. Par la Proposition 1.34 vue dans les préliminaires, C est représentable sous une forme non ambiguë :

$$C = \biguplus_{i=1}^{r} (\boldsymbol{c}_i + P_i^*)$$

où l'union est disjointe et les vecteurs dans chaque ensemble de périodes P_i sont linéairement indépendants. Par conséquent, sa série génératrice est donnée par :

$$C(\boldsymbol{y}) = \sum_{i=1}^{r} \frac{\boldsymbol{y}^{\boldsymbol{c}_i}}{\prod_{\boldsymbol{p} \in P_i} (1 - \boldsymbol{y}^{\boldsymbol{p}})}$$

qui est une somme de fractions rationnelles, donc est bien rationnelle.

Une autre façon de démontrer ce théorème est de considérer, comme nous l'avons dit dans les préliminaires, que C est reconnaissable par un automate fini non ambigu, et d'utiliser le résultat de [FS09, Proposition V.6] portant sur les séries génératrices associées aux chemins dans un automate (voir par exemple : l'Exemple 1.35 et l'Exemple 1.75 des préliminaires). ■

Comme nous l'avons annoncé, le théorème suivant est une conséquence directe du Théorème 6.41 et du fait que la série génératrice d'un langage de RCM est holonome [CM17]. Nous en proposons une preuve directe :

Théorème 7.45 (Conséquence de [CM17]).
▶ La série génératrice d'un langage de Parikh faiblement non ambigu est holonome. ◀

Démonstration. Soit L un langage reconnu par un automate de Parikh faiblement non ambigu $\mathcal{A}$ de dimension d. Notons $L(x)$ la série génératrice de L, et $C \subseteq \mathbb{N}^d$ l'ensemble semilinéaire associé à l'automate.

Nous notons

$$\overline{A}(x, y_1, \ldots, y_d) := \sum_{(n, i_1, \ldots, i_d) \in \mathbb{N}^{d+1}} \ell_{n, i_1, \ldots, i_d} x^n y_1^{i_1} \ldots y_d^{i_d}$$

la série génératrice pondérée associée aux calculs de $\mathcal{A}$, définie au Lemme 7.43. Cette série est rationnelle, donc holonome.

Posons $C(y_1, \ldots, y_d) = \sum_{\boldsymbol{i} \in C} y_1^{i_1} \ldots y_d^{i_d}$ la série génératrice du semilinéaire C. Cette série est aussi rationnelle par le Lemme 7.44. Nous notons alors

$$\overline{C}(x, y_1, \ldots, y_d) := \sum_{n \in \mathbb{N}} \sum_{\boldsymbol{i} \in C} x^n y_1^{i_1} \ldots y_d^{i_d} = \frac{1}{1-x} C(y_1, \ldots, y_d)$$

qui est aussi rationnelle. Posons alors $G(x, \boldsymbol{y}) := \overline{A}(x, \boldsymbol{y}) \odot \overline{C}(x, \boldsymbol{y})$. Que se passe-t-il quand nous faisons le produit d'Hadamard de ces deux séries ?

Rappelons que le produit d'Hadamard de deux séries qui ont le même nombre de variables est le produit terme à terme. Autrement dit, pour tout $n, i_1, \ldots, i_d$:

$$[x^n y_1^{i_1} \ldots y_d^{i_d}] G(x, \boldsymbol{y}) = [x^n y_1^{i_1} \ldots y_d^{i_d}] \overline{A}(x, \boldsymbol{y}) \times [x^n y_1^{i_1} \ldots y_d^{i_d}] \overline{C}(x, \boldsymbol{y}) \,.$$

Or, par définition, $[x^n y_1^{i_1} \ldots y_d^{i_d}] \overline{C}(x, \boldsymbol{y})$ vaut 1 si $(i_1, \ldots, i_d) \in C$, et vaut 0 sinon.

Autrement dit : $[x^n y_1^{i_1} \ldots y_d^{i_d}] G(x, \boldsymbol{y}) = [x^n y_1^{i_1} \ldots y_d^{i_d}] \overline{A}(x, \boldsymbol{y})$ si $(i_1, \ldots, i_d) \in C$, et vaut 0 sinon. Le produit d'Hadamard a donc supprimé du support de $\overline{A}$ tous les calculs étiquetés par un vecteur qui n'était pas dans le semilinéaire C. Ainsi :

$$G(x, \boldsymbol{y}) := \overline{A}(x, \boldsymbol{y}) \odot \overline{C}(x, \boldsymbol{y}) = \sum_{n \in \mathbb{N}} \sum_{\boldsymbol{i} \in C} \ell_{n, i_1, \ldots, i_d} x^n y_1^{i_1} \ldots y_d^{i_d} \,.$$

Par stabilité des séries holonomes par produit d'Hadamard, $G(x, \boldsymbol{y})$ est holonome.

Pour $n \in \mathbb{N}$, posons $\ell_n = \sum_{\boldsymbol{i} \in C} \ell_{n, i_1, \ldots, i_d}$ le nombre de calculs acceptants de $\mathcal{A}$ étiquetés par un mot de longueur n. Ce nombre est fini car $\mathcal{A}$ ne possède pas d'ε-transitions. Par faible non ambiguïté, ℓ_n désigne aussi le nombre de mots de longueur n dans le langage L.

De plus, en remarquant que $G(x, 1, \ldots, 1) = \sum_{n \in \mathbb{N}} \ell_n x^n$, nous en déduisons que $L(x) = G(x, 1, \ldots, 1)$. Donc $L(x)$ est holonome, par la Proposition 7.41. ■

Remarque 7.46 .
▶ Nous avons montré que la série génératrice univariée d'un langage de Parikh faiblement non ambigu est holonome, mais la preuve s'adapte facilement pour montrer que sa série multivariée est aussi holonome (à la place de compter en x la longueur du mot, il faut introduire une variable x_a pour chaque lettre $a \in \Sigma$ qui compte le nombre d'occurrences de a dans les calculs de $\mathcal{A}$). ◀

Lemme 7.47 (Série génératrice des calculs d'un automate de Parikh à pile).
▶ Soit $\mathcal{A}$ un automate de Parikh à pile faiblement non ambigu de dimension d, sans ε-transitions. Nous définissons la série génératrice pondérée $\overline{A}(x, y_1, \ldots, y_d)$ par :

$$\overline{A}(x, y_1, \ldots, y_d) := \sum_{(n, i_1, \ldots, i_d) \in \mathbb{N}^{d+1}} \ell_{n, i_1, \ldots, i_d} x^n y_1^{i_1} \ldots y_d^{i_d}$$

où pour tout $(n, i_1, \ldots, i_d) \in \mathbb{N}^{d+1}$, $\ell_{n, i_1, \ldots, i_d}$ compte le nombre de calculs de $\mathcal{A}$ joignant l'état initial à un état final, étiquetés par un mot de longueur n, et le vecteur $(i_1, \ldots, i_d)$. Alors $\overline{A}(x, y_1, \ldots, y_d)$ est algébrique. ◀

Démonstration. Comme nous l'avons déjà mentionné au chapitre 6, le langage des calculs acceptants d'un automate à pile est algébrique déterministe, et ce même si l'automate à pile de départ est ambigu. Par le théorème de Chomsky-Schützenberger, la série génératrice qui compte les calculs acceptants d'un automate à pile est donc algébrique. Une simple adaptation de la preuve du théorème de Chomsky-Schützenberger permet de démontrer que $\overline{A}(x, y_1, \dots, y_d)$ est algébrique (ou peut sinon transformer $\mathcal{A}$ en automate à pile sur des lettres en encodant les vecteurs des transitions sous la forme de lettres, et le résultat est alors une conséquence directe de l'application du théorème de Chomsky-Schützenberger). ■

Théorème 7.48.
▶ La série génératrice d'un langage algébrique de Parikh faiblement non ambigu est holonome. ◀

Démonstration. Il s'agit de la même preuve que le Théorème 7.45, avec cette fois $\overline{A}(x, y_1, \dots, y_d)$ qui est algébrique et non plus rationnelle (mais toujours holonome). ■

Remarque 7.49 (Réciproque).
▶ Il est naturel de se demander si les séries associées aux langages algébriques de Parikh faiblement non ambigus recouvrent toutes les séries holonomes, et si ce n'est pas le cas, de caractériser les séries qui sont atteintes par cette correspondance.

Nous en discuterons plus précisément dans les ouvertures, à la sous-section 7.4.4, où nous montrerons que les séries associées aux langages de Parikh (resp. langages algébriques de Parikh) faiblement non ambigus sont exactement les diagonales de séries $\mathbb{N}$-rationnelles (resp. de séries $\mathbb{N}$-algébriques). ◀

7.3 Intrinsèque faible ambiguïté

7.3.1 Premiers exemples de langages intrinsèquement faiblement ambigus

La contraposée du Théorème 7.48 étend la méthode de Flajolet pour montrer l'intrinsèque ambiguïté des langages algébriques de Parikh :

Proposition 7.50.
▶ Si la série génératrice (multivariée) d'un langage algébrique de Parikh n'est pas holonome, alors le langage est intrinsèquement faiblement ambigu. ◀

Comme les séries algébriques sont incluses dans les séries holonomes, il se trouve que certains des critères utilisés par Flajolet dans [FS87] pour prouver qu'une série est transcendante montrent en fait aussi qu'elle n'est pas holonome :

Proposition 7.51 (Outils pratiques pour montrer qu'une série n'est pas holonome, [Fla87]).
▶ Soit $L(z) = \sum_{n\in\mathbb{N}} \ell_n z^n$ une série entière.

a. Si la fonction $L(z)$ possède une infinité de singularités, alors $L(z)$ n'est pas holonome.

b. Si ℓ_n ne vérifie pas à partir d'un certain rang une équation de récurrence linéaire à coefficients polynomiaux en n, alors $\ell(z)$ n'est pas holonome. ◀

En particulier, les séries lacunaires (voir définition p.222) ne sont pas holonomes. Comme les séries holonomes sont closes par les mêmes opérations usuelles que les séries algébriques, le principe de preuve pour démontrer l'intrinsèque faible ambiguïté d'un langage est le même que pour les langages algébriques : on suppose que la série génératrice du langage est holonome, puis on la transforme par des opérations usuelles qui préservent l'holonomie pour obtenir une série plus simple qu'on sait identifier comme non holonome, et obtenir une contradiction.

En fait, tous les langages algébriques démontrés intrinsèquement ambigus dans [Fla87] ou bien sont reconnaissables par un automate de Parikh déterministe, ou bien ont une série génératrice qui n'est pas holonome et sont donc aussi intrinsèquement faiblement ambigus en tant que langages algébriques de Parikh. Par exemple :

Proposition 7.52 (Un premier langage algébrique de Parikh intrinsèquement faiblement ambigu, [Fla87]).

Nous rappelons que la notation $\underline{n}$ désigne le mot $a^n b$ pour tout $n \in \mathbb{N}$.

▶ Le langage P_2, défini par

$$P_2 := \{\underline{n_1} \dots \underline{n_k} \mid k \in \mathbb{N}^*, (n_1 = 1 \text{ et } \forall j, n_{2j} = 2n_{2j-1}) \text{ ou } (\forall j, n_{2j+1} = 2n_{2j})\}$$

est un langage algébrique de Parikh intrinsèquement faiblement ambigu. ◀

Démonstration. Nous avons démontré à la Proposition 7.21 que la série génératrice de ce langage avait une infinité de singularités. Elle n'est donc pas holonome. Comme le langage est algébrique, c'est aussi un langage algébrique de Parikh. ■

La proposition suivante, fondée sur le même principe, donne un exemple de langage de Parikh qui est intrinsèquement ambigu :

Proposition 7.53 (Un langage de Parikh intrinsèquement faiblement ambigu).
▶ Le langage $\mathcal{D} = \{\underline{n_1}\ \underline{n_2} \dots \underline{n_k} \mid k \in \mathbb{N}^*,\ n_1 = 1 \text{ et } \exists j < k, n_{j+1} \neq 2n_j\}$ est un langage (algébrique) de Parikh intrinsèquement faiblement ambigu. ◀

Démonstration. Notons $\overline{\mathcal{D}} = ab(a^*b)^* \setminus \mathcal{D}$. Sa série génératrice est $\overline{D}(x_a, x_b) = \sum_{k \geqslant 1} x_a^{2^k - 1} x_b^k$ qui a été démontrée non holonome dans la preuve de la Proposition 7.21. ■

Remarque 7.54 .
▶ Le langage précédent est aussi intrinsèquement faiblement ambigu en tant que langage algébrique de Parikh. Il faut toujours préciser par rapport à quelle classe un langage est intrinsèquement (faiblement) ambigu. Nous verrons en sous-section 7.1.1 qu'il existe des langages intrinsèquement faiblement ambigus en tant que langages de Parikh, mais qui sont faiblement non ambigus en tant que langages algébriques de Parikh. ◀

Le langage suivant avait été annoncé à la Proposition 6.9 du chapitre 6 pour démontrer que les langages de Parikh faiblement ambigus ne sont pas clos par quotient à gauche par un langage régulier :

Proposition 7.55 (Le langage de Shamir est intrinsèquement faiblement ambigu).
▶ Le langage $\mathcal{S} = \{a^n b v_1 a^n v_2 : n \geqslant 1, v_1, v_2 \in \{a, b\}^*\}$ est un langage de Parikh intrinsèquement faiblement ambigu. ◀

Démonstration. On peut facilement construire un automate de Parikh ambigu de dimension 2 qui reconnaît $\mathcal{S}$: l'automate utilise sa première coordonnée pour stocker le nombre n de a présents au début du mot avant le premier b, puis de façon non déterministe, choisit de compter dans sa deuxième coordonnée un nombre m de lettres a qu'il lit dans une séquence de a consécutifs. La contrainte linéaire de l'automate pour que le calcul soit acceptant est $n \leqslant m$.

[Fla87] prouve que ce langage est algébrique intrinsèquement ambigu, car sa série génératrice $S(z) = \frac{z(1-z)}{1-2z} \sum_{n\geqslant 1} \frac{z^{2n}}{1-2z+z^{n+1}}$ a une infinité de singularités. Cela signifie aussi qu'elle n'est pas holonome, et que donc $\mathcal{S}$ est intrinsèquement faiblement ambigu en tant que langage de Parikh. ■

Il est par conséquent aussi intrinsèquement faiblement ambigu en tant que langage algébrique de Parikh, et la classe $\mathrm{wu}\mathbb{P}\mathrm{A}$ *n'est donc pas close non plus par quotient à gauche par un langage régulier.*

Nous pouvons désormais déduire les propriétés de non clôture des langages algébriques de Parikh, annoncées au chapitre précédent :

Proposition 7.56 (Non clôture des langages algébriques de Parikh faiblement non ambigus)**.**
▶ La classe des langages algébriques de Parikh faiblement non ambigus n'est *pas* close par intersection, union, concaténation, ni quotient à gauche. ◀

Démonstration. Les démonstrations de non clôture suivantes fonctionnent aussi pour les langages algébriques non ambigus.

Pour l'union, rappelons que P_2 est l'union de deux langages algébriques déterministes $L_1 = \{\underline{n}_1 \dots \underline{n}_k \mid n_1 = 1 \text{ et } \forall j, n_{2j} = 2n_{2j-1}\}$ et $L_2 = \{\underline{n}_1 \dots \underline{n}_k \mid \forall j, n_{2j+1} = 2n_{2j}\}$, qui sont donc aussi faiblement non ambigus en tant que langages algébriques de Parikh. Comme P_2 est intrinsèquement faiblement ambigu, ceci montre que la classe n'est pas close par union.

Par ailleurs, l'intrinsèque faible ambiguïté de P_2 est prouvée en montrant que la série génératrice de $I = L_1 \cap L_2$ n'est pas holonome. Donc $L_1 \cap L_2$ n'est pas un langage algébrique de Parikh faiblement non ambigu : ceci prouve la non clôture par intersection.

La non clôture par concaténation vient du fait que la série génératrice du langage $Pal^2 = \{uv \mid u, v \in Pal\}$ des concaténations de deux palindromes n'est pas holonome [Kem82, FS87].

Crestin prouve que c'est un langage algébrique infiniment ambigu [Cre72].

La non clôture par quotient à gauche vient du fait que la classe n'est pas close par quotient à gauche par un langage régulier. ■

Remarque 7.57 .
▶ Comme nous l'avions déjà dit au chapitre précédent, la question de la clôture par complémentaire reste ouverte. ◀

Les langages précédents sont directement ou indirectement issus de l'article de Flajolet [Fla87]. Nous proposons, pour diversifier un peu les exemples, un langage issu des marches à petits pas confinées dans le quart de plan. Considérons $\mathbb{N}^2$ le quart de plan plongé dans $\mathbb{Z}^2$, et $\Sigma = \{\nwarrow, \nearrow, \searrow\}$ un ensemble de petis pas. Nous interprétons le symbole $\nwarrow$ comme le vecteur $(-1, 1)$, le symbole $\nearrow$ comme le vecteur $(1, 1)$, et le symbole $\searrow$ comme le vecteur $(1, -1)$. Une suite de symboles de Σ représente alors une marche dans $\mathbb{Z}^2$, qui commence à l'origine du plan $(0, 0)$, et qui se déplace en suivant les vecteurs associés aux symboles de Σ (voir par exemple Figure 7.1). Une telle marche est alors dite *confinée dans le quart de plan* si elle reste dans $\mathbb{N}^2$ (elle ne traverse jamais les axes). Les marches confinées dans le quart de plan, associées aux pas décrits par $\Sigma = \{\nwarrow, \nearrow, \searrow\}$, ont été étudiées par [MR09].

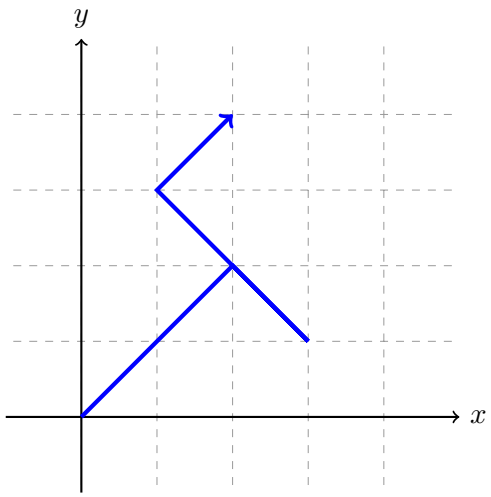


Figure 7.1. Exemple de marche confinée dans le quart de plan, associée au mot $\nearrow\nearrow\searrow\nwarrow\nwarrow\nearrow$.

Proposition 7.58 (Langage issu des marches du quart de plan, [MR09]).
▶ Soit $\Sigma = \{\nwarrow, \nearrow, \searrow\}$, et $L \subseteq \Sigma^*$ le langage des mots qui représentent une marche à déplacements dans Σ qui *n'est pas confinée dans le quart de plan* (autrement dit qui traverse les axes). Alors L est un langage (algébrique) de Parikh intrinsèquement faiblement ambigu. ◀

Démonstration. Il est facile de voir que si w représente une marche de pas dans Σ, alors la marche sort du quart de plan si et seulement s'il existe un préfixe σ de w tel que $|\sigma|_\nwarrow + |\sigma|_\nearrow < |\sigma|_\searrow$ ou $|\sigma|_\nearrow + |\sigma|_\searrow < |\sigma|_\nwarrow$ (autrement dit dans le préfixe σ, ou bien le nombre de flèches qui montent est strictement plus petit que le nombre de flèches descendantes, ou bien le nombre de flèches allant à droite est strictement plus petit que le nombre de flèches allant à gauche).

On construit alors facilement un automate de Parikh qui reconnaît L : l'automate commence à lire un préfixe de w en entrée, en stockant dans ses trois coordonnées le nombre d'occurrences de chaque lettre de Σ lue jusqu'ici ; puis il décide de façon non déterministe qu'il a lu un préfixe σ candidat, finit de lire le mot sans modifier ses coordonnées, puis vérifie par sa contrainte semilinéaire que σ vérifiait bien la condition de W sur les occurrences des lettres.

Le langage $\Sigma^* \setminus L$ désigne l'ensemble des mots $w \in \Sigma^*$ tels que pour tout préfixe σ de w on ait à la fois $|\sigma|_\nwarrow + |\sigma|_\nearrow \geqslant |\sigma|_\searrow$ et $|\sigma|_\nearrow + |\sigma|_\searrow \geqslant |\sigma|_\nwarrow$. Si on interprète les mots de Σ^* comme des marches dans le plan partant de la coordonnée $(0, 0)$, cette condition signifie que la marche doit rester dans le quart de plan $\{x \geqslant 0, y \geqslant 0\}$. Dans [MR09], les auteurs démontrent que la série génératrice $w(x)$ des marches dans le quart de plan qui utilisent l'ensemble de pas $\{\nwarrow, \nearrow, \searrow\}$ a une infinité de singularité et n'est pas holonome. Par conséquent la série génératrice du langage L, qui vaut $L(x) = \frac{1}{1-3x} - w(x)$ n'est pas holonome.

Donc le langage L est intrinsèquement faiblement ambigu. ■

Nous touchons avec l'exemple précédent les limites des critères analytiques utilisés dans cette thèse : étant donné un ensemble de pas élémentaires $\Sigma \subseteq \{\leftarrow, \swarrow, \downarrow, \searrow, \rightarrow, \nearrow, \uparrow, \nwarrow\}$, il me paraît difficile de construire un automate de Parikh à pile capable de reconnaître de façon faiblement non ambiguë les marches de Σ^* qui ne sont pas

confinées dans le quart de plan (sauf pour certaines marches, dites singulières, qui sont équivalentes à des marches confinées dans le demi-plan [Mis09]). Pour exhiber un langage intrinsèquement faiblement ambigu, nous avons été obligé de nous tourner vers une marche dont la série génératrice n'est pas holonome. Si nous considérons la marche à petits pas dans $\Sigma = \{\swarrow, \searrow, \nearrow, \nwarrow\}$, qui a une série génératrice holonome [BP03], son complémentaire n'a intuitivement pas plus de raison d'être dans la classe $\mathrm{wu}\mathbb{P}\mathrm{A}$ que la marche étudiée dans la proposition précédente, mais nous sommes incapables de démontrer son intrinsèque faible ambiguïté avec notre critère.

Avec $\Sigma = \{\nearrow, \searrow\}$ par exemple nous retrouvons les chemins de Dyck.

Il s'agit d'une intuition, peut-être fausse.

Une première piste pour s'attaquer à l'intrinsèque faible ambiguïté des langages qui ont une série holonome est de préciser un sous-ensemble strict de séries holonomes auquel appartiennent les séries génératrices des langages (algébriques) de Parikh faiblement non ambigus. Nous en discuterons dans la section 7.4. Une autre idée est de prouver à la main, par des techniques d'itération, qu'un langage n'est reconnaissable par aucun automate de Parikh à pile non ambigu. La section suivante est une première piste dans cette direction, même si nous nous sommes restreints au cas plus simple des automates de Parikh (donc sans pile).

Nous n'avons pas encore réussi à prouver par des techniques d'itération l'intrinsèque faible ambiguïté d'un langage algébrique de Parikh.

7.3.2 Vers un argument d'itération pour montrer l'intrinsèque faible ambiguïté d'un langage de Parikh

Comme pour les langages algébriques, la méthode analytique ne suffit pas toujours pour prouver l'intrinsèque faible ambiguïté d'un langage. Dans cette section, nous développons un exemple où le critère de non holonomie échoue. Nous considérons le langage $\mathcal{L}_{even}$ suivant :

$$\mathcal{L}_{even} = \left\{ \underline{n_1}\,\underline{n_2} \dots \underline{n_{2k}} \;:\; k \in \mathbb{N},\ \forall i \leqslant 2k, n_i > 0,\ \text{et } \exists j \leqslant k, n_{2j} = n_{2j-1} \right\}.$$

En d'autre mots, le langage $\mathcal{L}_{even}$ reconnaît les suites de paires d'entiers, dont une paire au moins contient les mêmes entiers (nous rappelons que la notation $\underline{n}$ est un raccourci d'écriture pour le mot $a^n b$).

Le langage $\mathcal{L}_{even}$ est reconnu par un automate de Parikh ambigu : en effet, il est reconnu par un automate de dimension 2 qui devine simplement la paire à vérifier : il stocke dans sa première coordonnée le nombre de a du premier bloc de la paire, et dans sa seconde coordonnée le nombre de a du deuxième bloc. Le semilinéaire vérifie que ces deux valeurs sont égales. L'automate est ambigu, car un mot a autant de calculs acceptants qu'il contient de paires d'entiers égaux : par exemple le mot $\underline{1111}$ a deux calculs acceptants.

Le langage $\mathcal{L}_{even}$ est aussi algébrique déterministe (en particulier il est reconnu par un automate de Parikh à pile faiblement non ambigu). Il est en effet reconnu par l'automate à pile déterministe décrit ci-après : en lisant une paire d'entiers $\underline{n_{2i}}\underline{n_{2i+1}}$, l'automate déterministe empile le nombre n_{2i} de a qu'il lit dans le premier bloc, puis dépile en lisant les a dans le deuxième bloc : s'il atteint son symbole de fond de pile juste avant de lire la lettre b qui clôt la paire, alors il accepte le mot, sinon il recommence la démarche avec la paire suivante, après avoir vidé sa pile si elle contenait encore des symboles. L'automate décrit est bien déterministe, et détecte la première paire d'entiers égaux d'un mot de $\mathcal{L}_{even}$.

Nous déduisons du théorème de Chomsky-Schützenberger que la série génératrice $L_{even}(z)$ du langage $\mathcal{L}_{even}$ est algébrique. En fait, $L_{even}(z)$ est même rationnelle : en effet, $\mathcal{L}_{even} = ((a^+b)^2)^* \setminus \{a^n b a^m b \,|\, n \neq m, \text{ et } n, m > 0\}^+$, et $\{a^n b a^m b \,|\, n \neq$

$m,$ et $n, m > 0\} = (a^+b)^2 \setminus \{a^n b a^n b \,|\, n > 0\}$. Ainsi :

$$L_{even}(z) = \frac{1}{1-\frac{z^4}{(1-z)^2}} - \frac{\frac{z^4}{(1-z)^2} - \frac{z^4}{1-z^2}}{1-\frac{z^4}{(1-z)^2}+\frac{z^4}{1-z^2}} = \frac{2\,z^9-4\,z^7+8\,z^6-3\,z^5-3\,z^4+2\,z^3+2\,z^2-3\,z+1}{(2\,z^5-z^3+z^2+z-1)(z^2-z+1)(z^2+z-1)}\,.$$

Ainsi les critères que nous avons développés à la section précédente échouent, pourtant $\mathcal{L}_{even}$ ne semble intuitivement pas être reconnaissable par un automate de Parikh faiblement non ambigu. Le but de cette section est de développer une technique de preuve pour le montrer.

Décomposition d'un calcul

Les arguments de cette section ne se limitent cependant pas à un alphabet à deux lettres.

Le but de cette sous-section est de poser les bases d'une décomposition des calculs acceptants d'un automate de Parikh faiblement non ambigu, qui nous sera utile pour prouver l'intrinsèque faible ambiguïté de certains langages de Parikh. Nous fixons l'alphabet $\Sigma = \{a, b\}$ dans cette section, et $\mathcal{A}$ un automate de Parikh faiblement non ambigu sur Σ, sans ε-transition (en particulier nous ne faisons aucune supposition sur le langage reconnu par $\mathcal{A}$). Nous nous concentrons dans un premier temps sur les calculs de $\mathcal{A}$ qui ne lisent que la même lettre a :

Définition 7.59 (a-calcul, cycle élémentaire).
▶ Un calcul π de $\mathcal{A}$ est appelé *a-calcul*, si π est de la forme $p_0 \xRightarrow[\pi]{w,\boldsymbol{v}} p_n$, avec $w \in a^*$: autrement dit les transitions de π ne lisent que la lettre a.

Un *a-cycle élémentaire* de $\mathcal{A}$ est un a-calcul ω non vide, commençant et finissant dans le même état, et qui ne passe pas deux fois par le même état (sauf en ses extrémités). Nous appelons *origine* d'un a-cycle élémentaire w son état de départ (qui est aussi son état d'arrivée).

L'ensemble des a-cycles de $\mathcal{A}$ est un ensemble fini, que nous notons $\Pi(\mathcal{A})$. ◀

Définition 7.60 (a-maxcalcul).
▶ Soit R un calcul de $\mathcal{A}$, et π un sous-a-calcul de R non vide. Nous disons que π est un a-maxcalcul de R si π n'est précédé ni suivi par une transition qui lit un a dans R. ◀

Soit $\mathcal{R}$ un calcul acceptant de $\mathcal{A}$, et π un a-maxcalcul de $\mathcal{R}$. Nous appelons *décomposition canonique* de π la décomposition de π sous la forme

$$\pi = w_1\sigma_1^{s_1}w_2\sigma_2^{s_2}\cdots w_f\sigma_f^{s_f}w_{f+1},$$

avec $f \in \mathbb{N}$ et $s_1, \ldots, s_f > 0$, obtenue comme suit :

- Nous lisons les premières transitions du calcul π jusqu'à la première fois où il passe pour la deuxième fois dans un état q (ou jusqu'à la fin du calcul sinon) : nous obtenons un sous-calcul non vide de la forme $w_1\sigma_1$, où w_1 ne passe jamais deux fois par le même état, et σ_1, s'il est non vide, est un a-cycle élémentaire, d'origine q. De plus w_1 s'il est non vide ne passe par aucun état en commun avec σ_1, à l'exception de son état d'arrivée, qui est l'origine de σ_1.
- Nous répétons la procédure décrite ci-dessus avec le suffixe $(w_1\sigma_1)^{-1}\pi$ de π.
- Nous obtenons donc à la fin de la procédure une décomposition de π de la forme $\pi = w_1\sigma_1w_2\sigma_2\cdots w_f\sigma_t w_{t+1}$, que nous factorisons en $\pi = w_1\sigma_1^{s_1}w_2\sigma_2\cdots w_f\sigma_f^{s_f}w_{f+1}$ en regroupant autant que possible les a-cycles élémentaires identiques consécutifs séparés par des w_i vides : par exemple $\sigma_4\varepsilon\sigma_4\varepsilon\sigma_4$ devient σ_4^3.

La procédure ci-dessus est déterministe, donc il n'y a pas d'ambiguïté pour parler de *la* décomposition canonique d'un a-maxcalcul. La proposition suivante établit des propriétés de la décompositions d'un a-maxcalcul apparaissant dans un calcul acceptant d'un automate de Parikh faiblement non ambigu.

Proposition 7.61 (Propriétés de la décomposition canonique d'un a-maxcalcul).
▶ Soit $\mathcal{R}$ un *calcul acceptant* de $\mathcal{A}$, et π un a-maxcalcul de $\mathcal{R}$. Alors la décomposition canonique $\pi = w_1\sigma_1^{s_1}w_2\sigma_2^{s_2}\cdots w_f\sigma_f^{s_f}w_{f+1}$ de π vérifie les propriétés suivantes :

Nous rappelons que $\mathcal{A}$ est faiblement non ambigu

a. $f \in \mathbb{N}$ et $s_1, \ldots, s_f > 0$;

b. chaque σ_i est un a-cycle élémentaire (non vide) ;

c. chaque w_i est un a-calcul qui ne passe jamais deux fois par le même état. De plus, w_i ne passe par aucun état en commun avec σ_i, sauf son état d'arrivée qui est l'origine de σ_i ;

d. les a-cycles élémentaires qui apparaissent dans la décomposition canonique ont tous une origine distincte ; en particulier pour tout $1 < i < f+1$, $w_i \neq \varepsilon$, et $f \leqslant |Q_{\mathcal{A}}|$.

◀

Démonstration. Les trois premiers points sont immédiats par construction de la forme canonique. Le dernier point vient du fait que $\mathcal{A}$ est un automate de Parikh faiblement non ambigu. Nous rappelons que par construction, si $w_i = \varepsilon$ pour un certain $1 < i < f+1$, alors $\sigma_{i-1} \neq \sigma_i$. Supposons par l'absurde que dans la décomposition canonique de π de la forme

$$\pi = w_1\sigma_1^{s_1}w_2\cdots w_f\sigma_1^{s_f}w_{f+1}\,,$$

il existe deux indices $i < j$ tels que les a-cycles élémentaires σ_i et σ_j ont la même origine q. Nous considérons alors le a-calcul π' obtenu en déplaçant $\sigma_j^{s_j}$ juste avant $\sigma_i^{s_i}$ (voir aussi Figure 7.2) :

$$\pi' = w_1\sigma_1^{s_1}w_2\cdots w_i\sigma_j^{s_j}\sigma_i^{s_i}w_{i+1}\cdots w_jw_{j+1}\cdots w_f\sigma_f^{s_f}w_{f+1}.$$

Les a-calculs π et π' sont distincts :

— si $\sigma_i = \sigma_j$ alors π et π' n'ont donc pas la même décomposition canonique (le s'_i de π' vaut $s_i + s_j > s_i$), ils sont donc distincts.

— si $\sigma_i \neq \sigma_j$, alors là encore π et π' n'ont pas la même décomposition canonique : si $i > 1$ et $\sigma_j = \sigma_{i-1}$, alors le s'_{i-1} de π' vaut $s_{i-1} + s_j > s_{i-1}$) ; et si $i = 1$ ou $\sigma_j \neq \sigma_{i-1}$, alors le i-ème cycle élémentaire dans la décomposition canonique de π' est σ_j, tandis que celui de π est $\sigma_i \neq \sigma_j$.

De plus, π et π' sont deux a-calculs qui ont la même longueur, et qui sont étiquetés par le même vecteur, par commutativité de l'addition sur les vecteurs. Nous obtenons ainsi à partir de $\mathcal{R}$ un calcul acceptant $\mathcal{R}'$ différent de $\mathcal{R}$, étiqueté par le même mot. C'est impossible par faible non ambiguïté de $\mathcal{A}$. Donc tous les cycles élémentaires de la décomposition de π ont des origines différentes. ■

Définition 7.62 (Signature d'un maxcalcul).
▶ Soit $\mathcal{R}$ un *calcul acceptant* de $\mathcal{A}$, et π un a-maxcalcul de $\mathcal{R}$, de décomposition canonique $\pi = w_1\sigma_1^{s_1}w_2\sigma_2^{s_2}\cdots w_f\sigma_f^{s_f}w_{f+1}$. La *signature* de π est le tuple :

$$(w_1, \sigma_1, w_2, \ldots, w_f, \sigma_f, w_{f+1})\,,$$

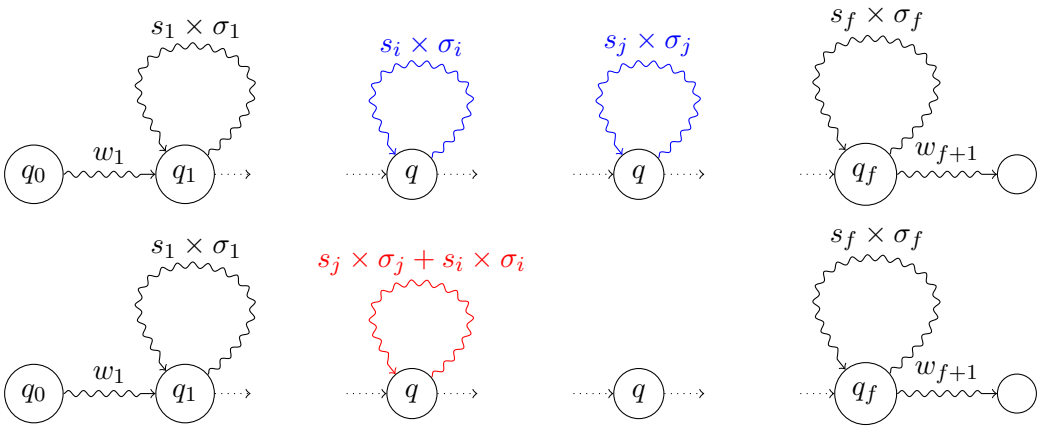


Figure 7.2. Si dans la décomposition canonique d'un a-maxcalcul, il existe deux a-cycles élémentaires σ_i et σ_j ayant la même origine, alors nous pouvons changer le calcul de façon à calculer le même mot et le même vecteur. C'est impossible si ce maxcalcul fait partie d'un calcul acceptant d'un automate de Parikh faiblement non ambigu.

obtenu à partir de sa décomposition canonique en omettant les exposants des cycles. ◀

Proposition 7.63 (Le nombre de signatures est fini).
▶ L'ensemble des signatures associées à des calculs acceptants de $\mathcal{A}$ est fini. ◀

Démonstration. Par la proposition précédente, une décomposition canonique d'un a-maxcalcul d'un calcul acceptant de $\mathcal{A}$ possède au plus $|Q_\mathcal{A}|$ cycles élémentaires. Une signature contient au plus $2|Q_\mathcal{A}| + 1$ éléments, qui sont soit des cycles élémentaires, soit des chemins simples de $\mathcal{A}$. Comme le nombre de cycles élémentaires dans $\mathcal{A}$ est fini, et le nombre de chemins simples dans $\mathcal{A}$ est aussi fini (comme un chemin simple ne passe pas deux fois par le même état, il est de longueur au plus $|Q_\mathcal{A}|$), le nombre de signatures associé à des calculs acceptants de $\mathcal{A}$ est fini. ■

Nous pouvons passer désormais à la preuve d'intrinsèque faible ambiguïté du langage $\mathcal{L}_{even}$.

Preuves d'intrinsèque faible ambiguïté.

Théorème 7.64 .
▶ Le langage $\mathcal{L}_{even}$ est intrinsèquement faiblement ambigu en tant que langage de Parikh. ◀

Démonstration. Par l'absurde, supposons que $\mathcal{L}_{even}$ est reconnu par un automate de Parikh faiblement non ambigu $\mathcal{A}$, sans ε-transition.

Soit c le nombre de signatures associées aux a-maxcalculs des calculs acceptants de $\mathcal{A}$. Ce nombre est fini par la Proposition 7.63.

Le Théorème de Ramsey [Ram29] garantit qu'il existe un entier r tel que tout graphe complet non dirigé possédant au moins r sommets, dont les arcs sont colorés par c^2 couleurs différentes, possède un triangle monochromatique.

Fixons deux entier n et k, que nous préciserons plus tard, et qui dépendent uniquement de $\mathcal{A}$.

Pour tout $\ell \in \{1, \dots, r\}$, notons w_ℓ le mot $w_\ell = \underline{n_1}\,\underline{n_2} \dots \underline{n_{2r}}$, où $n_{2i+1} = n$ pour tout i, et $n_{2i} = n + k$ si $i \neq \ell$, sinon $n_{2\ell} = n$ (autrement dit toutes les paires de w_ℓ sont $\underline{n}\,\underline{n+k}$, sauf la paire numéro ℓ qui vaut $\underline{n}\,\underline{n}$). Chaque w_ℓ contient ainsi une unique paire d'entiers égaux, donc appartient au langage $\mathcal{L}_{even}$. Par faible non ambiguïté, chaque w_ℓ est reconnu par un unique calcul acceptant $\mathcal{R}_\ell$ dans $\mathcal{A}$, qui par définition, possède exactement $2r$ a-maxcalculs.

Pour i, j dans $\{1, \dots, r\}$, nous notons λ_{ij} la signature du $2j$-ième a-maxcalcul de $\mathcal{R}_i$, qui par définition est un calcul de longueur $n + k$ si $i \neq j$. Nous construisons le graphe complet non dirigé $\mathcal{G}_\mathcal{A}$ de sommets $\{1, \dots, r\}$, et tel que pour tout $i < j$, l'arc (i, j) est coloré par le couple $(\lambda_{ij}, \lambda_{ji})$. Par le théorème de Ramsey, $\mathcal{G}_\mathcal{A}$ admet un triangle monochromatique de sommets $\alpha < \beta < \gamma$. En particulier, $\lambda_{\alpha\beta} = \lambda_{\alpha\gamma}$ et $\lambda_{\gamma\alpha} = \lambda_{\gamma\beta}$.

Autrement dit, le 2β-ième maxcalcul et le 2γ-ième maxcalcul de $\mathcal{R}_\alpha$ ont la même signature. Pour simplifier les notations, notons π^α_β (resp. π^α_γ) le 2β-ième (resp. 2γ-ième) maxcalcul de $\mathcal{R}_\alpha$.

Nous précisons alors comment nous choisissons k et n, en fonction de $\mathcal{A}$ uniquement : nous définissons $k = \mathrm{ppcm}(\{|\sigma| : \sigma \in \Pi(\mathcal{A})\})$, et n suffisamment grand pour que tout a-maxcalcul de longueur plus grande que n d'un calcul acceptant de $\mathcal{A}$ contient dans sa décomposition canonique un a-cycle élémentaire répété au moins $k + 1$ fois : c'est possible car nous rappelons que dans la décomposition canonique d'un a-maxcalcul, il y a un nombre fini de cycles élémentaires et de chemins simples w_i, et que chaque w_i est est de longueur bornée.

Ainsi, π^α_γ contient un a-cycle élémentaire σ qui est répété strictement plus de s fois, avec $s = k/|\sigma|$. Comme π^α_γ et π^α_β ont la même signature, ce cycle σ se trouve aussi dans π^α_β. Nous pouvons alors changer le calcul acceptant $\mathcal{R}_\alpha$ en un calcul $\mathcal{R}'_\alpha$ en déplaçant la boucle σ^s originellement présente dans π^α_γ pour la placer au niveau du cycle σ présent dans π^α_β. Nous obtenons ainsi le calcul $\mathcal{R}'_\alpha$ pour le mot :

$$w = \cdots \underbrace{\underline{n}\,\underline{n}}_{2\alpha} \cdots \underline{n}\,\underbrace{\underline{n+2k}}_{2\beta} \cdots \underbrace{\underline{n}\,\underline{n}}_{2\gamma} \cdots .$$

Ce calcul est acceptant, car il possède les même états de départ et d'arrivée que $\mathcal{R}_\alpha$, et est de plus étiqueté par le même vecteur, par commutativité. De plus, les signatures des a-maxcalculs de $\mathcal{R}'_\alpha$ sont les mêmes que ceux de $\mathcal{R}_\alpha$.

C'est pour cela que nous avons choisi n de façon à ce que le cycle σ soit répété strictement plus de s fois dans π^α_γ, afin que sa signature reste la même si on lui retire la boucle σ^s.

De façon similaire, à partir de l'égalité $\lambda_{\gamma\alpha} = \lambda_{\gamma\beta}$, nous pouvons modifier le calcul acceptant $\mathcal{R}_\gamma$ en un calcul acceptant $\mathcal{R}'_\gamma$ de même signatures que $\mathcal{R}_\gamma$ et étiqueté par le mot w, en déplaçant $s' = k/|\sigma'|$ itérations d'un cycle bien choisi σ' en position 2α vers la position 2β.

Nous avons donc construit deux calculs acceptants pour le mot w. Par faible non ambiguïté, ces deux calculs sont identiques. Mais comme les signatures des maxcalculs de $\mathcal{R}'_\alpha$ et $\mathcal{R}'_\gamma$ sont les mêmes que ceux de $\mathcal{R}_\alpha$ et $\mathcal{R}_\gamma$, cela signifie que les signatures des maxcalculs de $\mathcal{R}_\alpha$ et $\mathcal{R}_\gamma$ sont identiques.

En particulier $\lambda_{\alpha,\alpha} = \lambda_{\gamma,\alpha}$ et $\lambda_{\alpha,\beta} = \lambda_{\gamma,\beta}$. Or par monochromaticité, $\lambda_{\gamma,\beta} = \lambda_{\gamma,\alpha}$. Donc $\lambda_{\alpha,\alpha} = \lambda_{\alpha,\beta}$. Absurde, car nous pourrions retirer un a-cycle élémentaire en position 2α dans $\mathcal{R}_\alpha$ pour le déplacer en position 2β, et obtenir alors un calcul acceptant pour un mot qui n'a aucune paire de nombre égaux, et qui n'est donc pas dans $\mathcal{L}_{even}$.

Donc $\mathcal{L}_{even}$ est intrinsèquement faiblement non ambigu. ■

Corollaire 7.65 .

▶ Le langage $\mathcal{L}_{even}$ est intrinsèquement ambigu pour tout modèle d'automate de

Parikh étendu en remplaçant le semilinéaire d'acceptation C par un ensemble quelconque de $\mathbb{N}^d$. ◀

Démonstration. Ceci est dû au fait que la preuve du Théorème 7.64 ne repose que sur des manipulations de chemins dans un automate, et sur la commutativité de l'addition sur les vecteurs, mais est indépendante de la forme de l'ensemble qui accepte les vecteurs calculés dans les calculs de l'automate. Elle s'adapte donc directement à n'importe quelle extension des automates de Parikh ensemble C d'acceptation. ■

Remarque 7.66 (Piste future pour montrer que $\overline{Pal}$ est intrinsèquement faiblement ambigu)**.**
▶ Le langage suivant est une première piste de recherche pour exhiber un langage plus naturel qui soit intrinsèquement faiblement ambigu :

$$\mathcal{L}_{even}{}^{\neq} = \left\{ \underline{n_1}\ \underline{n_2} \dots \underline{n_{2k}} \ :\ k \in \mathbb{N},\ \forall i \leqslant 2k, n_i > 0,\ \text{et } \exists j \leqslant k, n_{2j} \neq n_{2j-1} \right\} .$$

En effet, s'il est possible de montrer qu'il est intrinsèquement faiblement ambigu, et si par chance la preuve ressemble à celle de $\mathcal{L}_{even}$, elle ne prendra pas en compte la position des paires. Dans ce cas elle s'adaptera directement pour prouver que le langage

$$P^{\neq} = \left\{ b\underline{n_{2k-1}} \dots \underline{n_3 n_1}\ \underline{n_2 n_4} \dots \underline{n_{2k}} \ :\ k \in \mathbb{N},\ \forall i \leqslant 2k, n_i > 0,\ \text{et } \exists j \leqslant k, n_{2j} \neq n_{2j-1} \right\}$$

est intrinsèquement faiblement ambigu, ce qui permettra de prouver que le langage $\overline{Pal}$ des mots qui ne sont pas des palindromes est intrinsèquement faiblement ambigu en tant que langage de Parikh (car $P^{\neq} = \overline{Pal} \cap b((a^+b)^2)^*$). ◀

7.3.3 Problèmes de décision liés à l'intrinsèque faible ambiguïté

Cette sous-section étudie brièvement deux problèmes de décision liés à l'intrinsèque faible ambiguïté des langages de Parikh. La proposition suivante contraste avec les grammaires hors-contextes, pour lesquelles la question de l'ambiguïté est indécidable.

Théorème 7.67 .
▶ Soit $\mathcal{A}$ un automate de Parikh. Alors on peut décider si $\mathcal{A}$ est faiblement ambigu. ◀

Démonstration. Notons d la dimension de $\mathcal{A}$, Q son ensemble d'états, q_I son état initial, Δ son ensemble de transitions, F son ensemble d'états finaux, et C son ensemble semilinéaire.

Nous utilisons la même construction que pour l'intersection des deux automates de Parikh faiblement ambigus : nous partons de la construction de l'automate de Parikh produit $\mathcal{A}_1 \times \mathcal{A}_2$ pour reconnaître l'intersection $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$, en prenant $\mathcal{A}_1 = \mathcal{A}_2 = \mathcal{A}$ deux copies de $\mathcal{A}$.

Ainsi $\mathcal{B} = (Q \times Q, (q_I, q_I), F \times F, \Delta', C')$, où C est l'ensemble semilinéaire des $(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{N}^{2d}$ tels que $\boldsymbol{u} \in C$ et $\boldsymbol{v} \in C$, et Δ' simule deux calculs simultanés dans chaque copie $\mathcal{A}$, lisant le même mot.

Nous modifions $\mathcal{B}$ en ajoutant une dimension supplémentaire. Nous remplaçons les transitions Δ' par l'ensemble Δ'' défini comme suit : pour toute paire de transitions $(t_1, t_2) \in \Delta \times \Delta$, étiquetées par la même lettre $a \in \Sigma$, avec $t_1 = (q_1, (a, \boldsymbol{v_1}), q_1')$ et

$t_2 = (q_2, (a, \boldsymbol{v_2}), q_2')$, alors $((q_1, q_2), (a, (\boldsymbol{v_1}, \boldsymbol{v_2}, x_{t_1,t_2})), (q_1', q_2')) \in \delta''$, où $x_{t_1,t_2} = 0$ si $t_1 = t_2$, 1 sinon. Ainsi, la dimension supplémentaire compte le nombre de fois que les deux calculs simultanés simulés par $\mathcal{B}$ empruntent des transitions différentes.

Le semilinéaire C' est modifié en conséquence : c'est l'ensemble des vecteurs $(\boldsymbol{u}, \boldsymbol{v}, x) \in \mathbb{N}^{2d+1}$ tels que $\boldsymbol{u} \in C$, $\boldsymbol{v} \in C$ et $x > 0$.

Comme par définition $\mathcal{A}$ est faiblement non ambigu si et seulement si aucun mot ne possède deux calculs acceptants, il est direct que $\mathcal{A}$ est faiblement non ambigu si et seulement si le langage reconnu par $\mathcal{C}$ est vide. Comme le vide d'un langage de Parikh est décidable [KR02] (car on peut effectivement calculer l'image de Parikh des langages de Parikh), on peut donc décider la faible ambiguïté de $\mathcal{A}$. ■

Remarque 7.68.
▶ La preuve précédente ne se généralise pas aux automates à pile faiblement non ambigus, car ces derniers n'admettent pas de construction produit, et ne sont pas stables par intersection. Comme le problème est déjà indécidable pour les grammaires algébriques, et par conséquent pour les automates à pile, il est aussi indécidable pour les automates de Parikh à pile. ◀

La question de l'intrinsèque ambiguïté d'un langage est différente de la détection de l'ambiguïté dans un automate, et est indécidable :

Proposition 7.69.
▶ Le problème de l'intrinsèque faible ambiguïté d'un langage de Parikh est indécidable. ◀

Démonstration. La classe des langages de Parikh faiblement non ambigus vérifie tous les critères du théorème général d'indécidabilité de Greibach [Gre68], et donc l'appartenance à cette classe est indécidable. Nous reprenons les arguments de la preuve du théorème général, spécialisés à notre problème, par soucis de complétude.

Nous réduisons le problème de l'universalité des langages de Parikh, qui est indécidable [KR02]. Étant donnée un instance L_1 du problème de l'universalité, nous voulons construire une instance L du problème de l'intrinsèque faible ambiguïté telle que :

C'est le même argument que pour l'universalité des langages algébriques : on peut construire un langage de Parikh à partir d'une machine de Turing M et d'un mot w, qui est universel si et seulement si M n'accepte pas w.

$$L_1 = \Sigma_1^* \Leftrightarrow L \text{ est intrinsèquement faiblement ambigu.}$$

Le langage L est construit comme suit. Notons Σ_1 l'alphabet de L_1, c un nouveau symbole qui n'est pas dans Σ_1, et finalement L_2 un langage intrinsèquement faiblement ambigu (par exemple le langage de Shamir S de la Proposition 7.55) sur un alphabet Σ_2 (par forcément distinct de Σ_1, nous pouvons avoir $\Sigma_1 \cap \Sigma_2 \neq \emptyset$).

Nous définissons alors $L = L_1 c \Sigma_2^* \cup \Sigma_1^* c L_2$, qui est bien un langage de Parikh. Ainsi : $L_1 = \Sigma_1^*$ si et seulement si L est intrinsèquement faiblement ambigu. En effet :

⇒ Si $L_1 = \Sigma_1^*$ alors $L = \Sigma_1^* c \Sigma_2^*$ est un langage régulier, donc est reconnaissable par une automate de Parikh faiblement non ambigu.

⇐ Si L est faiblement non ambigu, alors forcément $L_1 = \Sigma_1^*$. En effet, par l'absurde, si $L_1 \neq \Sigma_1^*$, il existe un mot $y \in \Sigma_1^*$ qui n'est pas dans L_1.
Comme $yc\Sigma_2^*$ est régulier, $L \cap yc\Sigma_2^*$ est un langage faiblement non ambigu, par clôture des langages faiblement non ambigus par intersection.
Comme $y \notin L_1$ et $c \notin \Sigma_1$, $L \cap yc\Sigma_2^* = ycL_2$.

Donc ycL_2 est faiblement non ambigu, et par stabilité par quotient à gauche *par un mot* (Proposition 6.24), $L_2 = (yc)^{-1}(ycL_2)$ est faiblement non ambigu. Contradiction, car nous avions choisi L_2 intrinsèquement faiblement ambigu.

■

Remarque 7.70.
▶ La preuve utilise au moins 3 symboles. On peut prouver que le problème reste indécidable avec 2 lettres, en encodant les symboles sur 2 lettres de façon non ambiguë. ◀

La question de l'intrinsèque faible ambiguïté des langages de Parikh, et a fortiori, des langages algébriques de Parikh, est donc une question difficile, et il est donc attendu que les différentes techniques présentées dans ce chapitre ne fonctionnent pas pour tous les langages.

7.4 Ouvertures

Nous clôturons le chapitre avec quelques ouvertures sur les questions d'ambiguïté des langages et de leur lien avec les séries holonomes.

7.4.1 Sur l'intrinsèque ambiguïté des langages algébriques bornés

Comme je l'ai déjà indiqué, il serait intéressant de généraliser le deuxième critère d'intrinsèque ambiguïté du Théorème 7.32 pour éviter l'évaluation des polynômes en une constante $a \in \mathbb{Q}$, en se plaçant dans un cadre algébrique plus solide.

Je suis aussi à la recherche de langages intrinsèquement ambigus bornés qui ne sont pas détectés par les deux critères des Théorème 7.24 et Théorème 7.32. Cela permettrait de tester leurs limites, et donner des pistes pour les généraliser.

Une autre piste dans ce sens consisterait à étudier la simplification d'une somme de fractions rationnelles de type *séries de semilinéaires* en fraction irréductible, et les conditions responsables de la disparition de certains facteurs irréductibles au dénominateur. Il est facile de trouver un exemple où la fraction se simplifie dans le cas non ambigu : nous savons que $L := \{a^n b^n c^p \mid n, p \in \mathbb{N}\}$ est un langage non ambigu ; la série du semilinéaire est simplement :

$$C(a,b,c) = \frac{1}{(1-ab)(1-c)}.$$

En distinguant selon que $p > n$ ou $p \leqslant n$, nous pouvons l'écrire comme l'union des $a^n b^n c^{n+r}$ avec $r > 0$ et des $a^{n+r} b^{n+r} c^r$ avec $r \geqslant 0$, ce qui donne une description non stratifiée du semilinéaire :

$$\frac{c}{(1-abc)(1-c)} + \frac{1}{(1-abc)(1-ab)}.$$

Dans ce cas particulier, en réduisant au même dénominateur, le terme $(1-abc)$ disparaît du dénominateur : $\frac{(1-ab)c+(1-c)}{(1-abc)(1-ab)(1-c)} = \frac{1}{(1-ab)(1-c)}$ et on retrouve comme prévu $C(a,b,c)$.

Nous voyons dans cet exemple que des simplifications peuvent apparaître lors de la réduction au même dénominateur, et éliminent certains polynômes irréductibles

qui auraient pu nous permettre de détecter l'intrinsèque ambiguïté. Dans l'exemple précédent, la simplification permet de retrouver une forme stratifiée, mais en général le polynôme est développé, ses coefficients sont de signe quelconque. Est-ce que ces simplifications sont tout de même possibles si le langage est intrinsèquement ambigu ? Si c'est le cas, est-il possible à partir de la fraction simplifiée de retrouver les facteurs simplifiés ? Il serait intéressant dans un premier temps d'étudier les simplifications de ce type sur des langages algébriques inclus dans $a^*b^*c^*$, car la condition de stratification en dimension 3 est plus simple (elle demande uniquement des périodes qui ont au plus deux coordonnées non nulles).

7.4.2 Séries $\mathbb{N}$-rationnelles, séries $\mathbb{N}$-algébriques, lien avec l'ambiguïté des langages algébriques

Nous avons vu dans le chapitre 1 des préliminaires que la série génératrice d'un langage régulier est rationnelle, et le théorème de Chomsky-Schützenberger exposé dans ce chapitre démontre que la série génératrice d'un langage algébrique non ambigu est algébrique. Il se trouve que la réciproque est fausse, toute série rationnelle n'est pas la série génératrice d'un langage régulier, et de même toute série algébrique n'est pas la série génératrice d'un langage algébrique non ambigu : plus exactement, les séries génératrices des langages réguliers sont appelées $\mathbb{N}$-rationnelles, et les séries génératrices des langages algébriques non ambigus sont appelées $\mathbb{N}$-algébriques, et elles constituent respectivement une sous classe stricte des séries rationnelles et algébriques. Nous reprenons dans un premier temps la présentation de ces classes de [Bou06], puis nous ouvrons sur les outils que cette caractérisation plus précise des séries des langages algébriques non ambigus est susceptible de nous apporter.

Nous aurons besoin des séries $\mathbb{N}$-rationnelles pour l'ouverture sur les séries des langages de Parikh faiblement non ambigus.

Nous commençons pas donner une définition des séries $\mathbb{N}$-rationnelles :

Définition 7.71 (Série $\mathbb{N}$-rationnelle, [Bou06]).
▶ Une série $L(x) = \sum_{n\in\mathbb{N}} \ell_n x^n$ à coefficients entiers est appelée $\mathbb{N}$-rationnelle s'il existe un langage régulier dont la série génératrice est $L(x) - \ell_0$ (ou de façon équivalente $L(x) - L(0)$). ◀

Remarque 7.72 .
▶ Il faut soustraire le premier terme ℓ_0 car il n'y a qu'un seul mot de longueur 0, le mot vide ε, peu importe la taille de l'alphabet. Ainsi si $L(x) = \sum_{n\in\mathbb{N}} \ell_n x^n$ est la série génératrice d'un langage, alors forcément $\ell_0 = 0$ ou $\ell_0 = 1$. Ne pas exiger $\ell_0 = 0$ ou $\ell_0 = 1$ dans la définition des séries $\mathbb{N}$-rationnelles permet plus de souplesse dans leur manipulation. ◀

Par définition, les séries $\mathbb{N}$-rationnelles sont associées aux séries génératrices des langages réguliers. Elles vérifient la propriété suivante (qui est parfois utilisée comme définition des séries $\mathbb{N}$-rationnelles, comme dans [GP14]) :

Proposition 7.73 (Définition alternative équivalente, [Bou06, GP14, SS78]).
▶ La classe des séries $\mathbb{N}$-rationnelles est le plus petit sous-ensemble de $\mathbb{N}[[x]]$ qui contienne 0, x, et qui soit clos par addition, multiplication, et quasi-inverse (*i.e.* si $L(x)$ est $\mathbb{N}$-rationnelle et $L(0) = 0$, alors $\frac{1}{1-L(x)}$ est $\mathbb{N}$-rationnelle). ◀

Remarque 7.74 .
▶ Dans [SS78] les auteurs font la distinction entre la série formelle à une variable et la fonction rationnelle associée (qu'ils appellent fonction génératrice de la série). ◀

Il existe des critères simples pour décider si une série rationnelle est $\mathbb{N}$-rationnelle. Je ne les détaille pas car ils ne vont pas nous servir ici, je réfère le lecteur à [SS78, Theorems 10.2 et 10.5], ou à leur version condensée dans [Bou06, Theorem 2.5]. Un exemple de série rationnelle qui n'est pas $\mathbb{N}$-rationnelle est donné dans [Bou06] : $A(x) = \frac{1-2x+225x^2}{(1-25x)(625x^2+14x+1)}$.

Nous présentons maintenant les séries $\mathbb{N}$-algébriques, l'analogue des séries $\mathbb{N}$-rationnelles pour les langages algébriques non ambigus. Comme pour les séries $\mathbb{N}$-rationnelles, on trouve dans la littérature plusieurs définitions de cette classe [BD15, Bou06, SS78]. Je présente celle de [Bou06] :

Définition 7.75 (Système propre positif, [Bou06]).
▶ Un système propre positif est un système de la forme :

$$\begin{cases} y_1 &= P_1(x; y_1, \dots, y_d) \\ &\vdots \\ y_d &= P_d(x; y_1, \dots, y_d) \end{cases},$$

Un système de cette forme est plutôt appelé well-defined *dans [BD15].*

où les P_i sont des polynômes à coefficients dans $\mathbb{N}$, sans terme constant ($P_i(x; \mathbf{0}) = 0$), ni terme linéaire en les variables $\boldsymbol{y}$. ◀

Proposition 7.76 (Série $\mathbb{N}$-algébrique, [Bou06]).
▶ Les solutions à coefficients entiers de système propres sont appelées $\mathbb{N}$-algébriques. Plus précisément :

- Tout système propre admet une unique solution $(y_1(x), \dots, y_d(x))$ de séries dans $\mathbb{N}[[x]]$.
- Une série $y_1(x)$ à coefficients entiers est appelée $\mathbb{N}$-algébrique si elle est la première coordonnée solution d'un système propre.
- Toute série $\mathbb{N}$-algébrique est la première coordonnée d'un système propre de la forme de la Définition 7.75, en supposant de plus que chaque P_i est divisible par x (autrement dit $P_i = xQ_i$ avec $Q_i \in \mathbb{N}[x; \boldsymbol{y}]$).

◀

Remarque 7.77 .
▶ Demander $P_i(x; \mathbf{0}) = 0$ s'interprète du côté grammaire par l'absence d'ε-production, l'absence de terme linéaire en l'absence de règle unit, et le fait de demander que les P_i soit divisibles par z ressemble à une forme normale de Greibach : notons que la "mise sous forme de Greibach" d'un système est beaucoup plus simple que pour les grammaires, car d'une part les lettres ne sont pas différenciées, et d'autre part les variables commutent entre elles (voir par exemple la preuve de [Bou06, Theorem 3.4]). ◀

Les séries $\mathbb{N}$-algébriques correspondent aux séries des langages algébriques non ambigus :

Proposition 7.78 ([Bou06, Proposition 3.7]).
▶ Une série $L(x) = \sum_{n\in\mathbb{N}} \ell_n x^n$ à coefficients entiers est $\mathbb{N}$-algébrique si et seulement s'il existe un langage algébrique non ambigu dont la série génératrice est $L(x) - \ell_0$. ◀

La preuve de [Bou06] utilise le fait que tout langage algébrique non ambigu, privé du mot vide ε, est reconnaissable par une grammaire sans règle unit ni ε-production (nous l'avons vérifié dans le chapitre précédent lors de la mise sous forme normale de Greibach). Dans [BD15], les auteurs caractérisent alors le comportement asymptotique des coefficients d'une série $\mathbb{N}$-algébrique :

Proposition 7.79 (Exposants critiques des coefficients d'une série $\mathbb{N}$-algébrique, [BD15]).
▶ Soit $F(z) = \sum_n f_n z^n$ une série $\mathbb{N}$-algébrique. Si F a une unique singularité sur son rayon de convergence $|z| = \rho$, alors

$$f_n \sim_{n\to\infty} \frac{C}{\Gamma(1+\alpha)} n^\alpha \rho^{-n}; \tag{7.1}$$

avec C, ρ des constantes positives algébriques, et α appartient à l'ensemble suivant :

$$\mathbb{D}_2 := \{-1 - 2^{-(k+1)} : k \geqslant 0\} \cup \left\{-1 + \frac{r}{2^k} : k \geqslant 0, r \geqslant 1\right\}.$$

Si $F(z)$ a plusieurs singularités, alors il existe un entier p tel que pour tout $\ell \in [0, p-1]$, ou bien $f_{\ell+np} = 0$ à partir d'un certain rang, ou bien $f_{\ell+np}$ a un équivalent asymptotique de la même forme que dans l'équation (7.1). ◀

Ainsi, certains exposants critiques sont compatibles avec l'algébricité (voir Proposition 7.17), mais pas la $\mathbb{N}$-algébricité. Il serait intéressant de pouvoir trouver un langage algébrique intrinsèquement ambigu, dont la série est algébrique mais pas $\mathbb{N}$-algébrique. Les exemples de séries de ce type cités par [BD15] viennent des cartes ou des marches dans le plan, pour lesquels il me semble à première vue difficile de trouver un langage algébrique qui possède la même série.

Par ailleurs, les séries $\mathbb{N}$-algébriques ont a priori moins de propriétés de clôture que les séries algébriques, ce qui rend plus difficile les preuves de non $\mathbb{N}$-algébricité. Par exemple, toute fonction algébrique à coefficients dans $\mathbb{N}$ peut s'écrire comme la différence de deux fonctions $\mathbb{N}$-algébriques [BD15, Proposition 1] : il n'est donc pas automatique que la différence de deux fonctions $\mathbb{N}$-algébriques reste $\mathbb{N}$-algébrique.

Un exemple de série algébrique qui n'est pas $\mathbb{N}$-algébrique est la série génératrice des marches de Gessel [BD15, BK10] $G(x) = \sum_{n\in\mathbb{N}} g_n x^n$, où la suite g_n compte le nombre de marches de longueur n, d'origine $(0,0)$ et confinées dans le quart de plan, et d'ensemble de pas $\{\leftarrow, \rightarrow, \swarrow, \nearrow\}$. L'asymptotique de g_n de la forme $c4^n n^{-2/3}$ est incompatible avec l'ensemble d'exposant $\mathbb{D}_2$ de la Proposition 7.79, donc $G(x)$ n'est pas $\mathbb{N}$-algébrique.

Les excursions de Gessel sont les marches de Gessel qui reviennent au point $(0,0)$. Leur série est aussi algébrique, mais pas $\mathbb{N}$-algébrique.

Ainsi la série $G(x)$ ne compte les arbres de dérivations d'aucune grammaire hors-contexte. En particulier le langage $\mathcal{G}$ des marches de Gessel n'est pas algébrique non ambigu. Cependant, il n'est pas algébrique tout court, puisque $\mathcal{G} \cap \nearrow^*\rightarrow^*\swarrow^*\leftarrow^* = \{\nearrow^n\rightarrow^m\swarrow^p\leftarrow^q \mid n \geqslant p \text{ et } n+m \geqslant p+q\}$ n'est pas un langage algébrique. Le langage complémentaire des marches sur $\{\leftarrow, \rightarrow, \swarrow, \nearrow\}$ qui ne sont pas des marches de Gessel est plus intéressant pour nous, car c'est un langage algébrique : l'automate qui le reconnaît devine (de façon ambiguë) un préfixe u tel que $|u|_\swarrow > |u|_\nearrow$ ou $|u|_\swarrow + |u|_\leftarrow > |u|_\nearrow + |u|_\rightarrow$ (autrement dit il devine un moment où la marche franchit l'axe des x ou des y). Sa série génératrice est par définition :

En s'inspirant par exemple de la preuve, à l'aide du lemme d'Ogden, de [Car08] pour le langage $a^n b^m c^n d^m$.

$$\overline{G}(x) = \frac{1}{1-4x} - G(x).$$

Je n'ai pas beaucoup réfléchi à la question, je passe peut-être à côté d'un argument simple.

Il serait intéressant de pouvoir prouver facilement que $\overline{G}(x)$ n'est pas non plus $\mathbb{N}$-algébrique pour prouver l'intrinsèque ambiguïté du complémentaire $\overline{\mathcal{G}}$, et de généraliser l'approche à d'autres complémentaires de marches reconnaissables par des grammaires ambiguës.

Remarque 7.80.

▶ Notons que nous pouvons démontrer que $\overline{\mathcal{G}}$ est intrinsèquement ambigu à l'aide de nos critères sur les langages bornés. En effet, $\overline{\mathcal{G}} \cap \nearrow^*\rightarrow^*\swarrow^*\leftarrow^*$ est intrinsèquement ambigu : le semilinéaire associé est $C = \{(n,m,p,q) \mid n < p \text{ ou } n+m < p+q\}$, dont la série génératrice est :

J'ai omis les calculs, mais ce n'est en fait pas toujours complètement immédiat de calculer la série génératrice d'un semilinéaire à la main.

$$\sum_{(n,m,p,q)\in C} a^n b^m c^p d^q = \frac{abcd^2 - acd - bd - cd + c + d}{(1-ac)\,(1-ad)\,(1-bd)\,(1-d)\,(1-c)\,(1-b)}\,.$$

Nous posons alors $D = 1 - bd$, $\pi = 1 - ac$, irréductibles aux variables entrelacées, et P le numérateur. En prenant $\alpha = 2$, $P|_{a=2,c=1/2} = bd^2 - bd - d/2 + 1/2$ n'est pas divisible par D (ils sont tous les deux de degré 1 en b), donc par le Théorème 7.32, $\overline{\mathcal{G}} \cap \nearrow^*\rightarrow^*\swarrow^*\leftarrow^*$ est intrinsèquement ambigu, et donc $\overline{\mathcal{G}}$ est aussi intrinsèquement ambigu. ◀

Remarque 7.81.

▶ Dans tous les exemples de cette thèse, pour prouver l'intrinsèque ambiguïté d'un langage, il était facile de démontrer que le langage était algébrique, mais difficile de montrer qu'il n'était jamais reconnu par une grammaire non ambiguë. Ce n'est pas toujours le cas : il arrive qu'on puisse savoir qu'un langage n'est pas reconnu par une grammaire non ambiguë, sans savoir si ce langage est algébrique ou non. C'est le cas du langage des mots primitifs, des mots qui ne sont la puissance d'aucun mot plus petit. Si Σ est un alphabet, le langage des mots primitifs $\mathcal{P}$ est défini par :

$$\mathcal{P} = \{w \in \Sigma^* \mid \exists u \in \Sigma^*, w \in u^* \Rightarrow u = w\}\,.$$

En 1994, [Pet94] a montré que la série génératrice de $\mathcal{P}$ n'est pas holonome, et ainsi que ce langage n'est reconnu par aucune grammaire non ambiguë. Cependant la question de savoir si $\mathcal{P}$ est algébrique ou non est toujours ouverte. ◀

7.4.3 Ouverture sur l'infinie ambiguïté des langages algébriques

Dans les ouvertures de son article [Fla87], Flajolet se demande s'il serait possible de capturer par des outils analytiques des propriétés plus fines liées à l'intrinsèque ambiguïté des langages algébriques, comme par exemple l'infinie ambiguïté. Nous présentons dans cette section un pas dans cette direction, en exhibant un langage pour lequel il est possible, par des arguments analytiques, de démontrer son intrinsèque infinie ambiguïté. Une grammaire hors-contexte est dite d'ambiguïté bornée s'il existe un entier $K \in \mathbb{N}^*$ tel que tout mot de son langage admet au plus K arbres de dérivation. Une grammaire qui n'est pas d'ambiguïté bornée est appelée infiniment ambiguë. Un langage algébrique est intrinsèquement infiniment ambigu si toutes les grammaires qui le reconnaissent sont infiniment ambiguës.

Supposons qu'un langage L est reconnu par une grammaire G qui est K-ambiguë. Notons a_n le nombre de mots de longueur n dans de langage L, et g_n le nombre de

dérivations gauches terminales de G étiquetées par un mot de longueur n. Alors par définition, pour tout $n \in \mathbb{N}$, $a_n \leqslant g_n \leqslant K a_n$, ou de façon équivalente :

$$\forall n \in \mathbb{N},\ \frac{g_n}{K} \leqslant a_n \leqslant g_n\,,$$

c'est-à-dire que a_n est encadré, à un facteur constant multiplicatif près, par le coefficient d'une série $\mathbb{N}$-algébrique, dont les asymptotiques sont de la forme $Cn^\alpha \rho^{-n}$ par la Proposition 7.79 de [BD15], avec $\alpha \in \mathbb{D}_2$. Si on arrive à prouver que $a_n = \Theta(f(n))$, où $f(n)$ est incompatible avec un encadrement de cette forme, alors L est intrinsèquement infiniment ambigu.

Illustrons ce principe sur le langage de Shamir. Soit $\Sigma = \{\#, a_1, \dots, a_k\}$ un alphabet de $k+1$ lettres, avec $k \geqslant 2$. Nous considérons le langage de Shamir étendu L_k définir par :

$$L_k = \{w \in \Sigma \,|\, w = s\#us^R v \text{ avec } s, u, v \in \{a_1, \dots, a_k\}^*\},$$

où la lettre # sert uniquement de séparateur, et s^R désigne le miroir de s. Ce langage est facilement reconnaissable par un automate à pile ambigu.

Si $s = s_1 s_2 \dots s_{n-1} s_n$, alors $s^R = s_n s_{n-1} \dots s_2 s_1$.

Pour $k = 2$, le langage L_2 fait partie des langages dont Shamir [Sha71] démontre l'intrinsèque infinie ambiguïté, en utilisant des raisonnements d'itérations sur les dérivations semblables au lemme d'Ogden (il montre en fait un résultat plus fin, à savoir que la plupart des mots du langage de la forme $s\#w$ ont autant d'arbres de dérivation que d'occurrence de s^R dans w).

Je propose ici une preuve analytique de l'intrinsèque infinie ambiguïté du langage L_k. Dans toute la suite, a_n désigne le nombre de mots de L_k de longueur n. Toute la preuve repose sur l'encadrement suivant :

Proposition 7.82 .
▶ Il existe des constantes $b_1, b_2 > 0$ telles que pour n assez grand,

$$b_1 \log_k n \leqslant \frac{a_n}{k^{n-1}} \leqslant b_2 \log_k n\,.$$ ◀

Autrement dit $a_n = \Theta(k^{n-1}\log_k(n))$.

Démonstration. Pour un préfixe s donné, nous notons a_n^s le nombre de mots de taille n dans L_k de la forme $s\#w$, et pour $a_n^\ell = \sum_{|s|=\ell-1} a_n^s$ le nombre de mots de L_k de la forme $s\#w$, de taille n, pour tous les mots s tels que $|s| = \ell - 1 \geqslant 1$.

Majoration. Fixons un mot s, de longueur $\ell-1$. Nous rappelons que a_n^s compte le nombre de mots dans le langage de la forme $s\#w$ de longueur n, tels que w contienne le facteur s^R. En retirant cette dernière contrainte sur les w, facilement $a_n^s \leqslant k^{n-\ell}$.

Par ailleurs, en partitionnant selon la position j dans le mot w où apparaît le facteur s^R, nous en déduisons aussi $a_n^s \leqslant \sum_{j=0}^{n-2\ell+1} k^j k^{n-2\ell+1-j} = (n-2\ell+2)k^{n-2\ell+1} \leqslant nk^{n-2\ell+1}$.

Ainsi, $a_n^s \leqslant \min(k^{n-\ell}, nk^{n-2\ell+1})$. De plus $\min(k^{n-\ell}, nk^{n-2\ell+1}) = k^{n-\ell}$ si $\ell \leqslant \log_k n + 1$.

En fin de compte, pour tout $\ell \geqslant 2$, $a_n^\ell = \sum_{|s|=\ell-1} a_n^s \leqslant k^{\ell-1}\min(k^{n-\ell}, nk^{n-2\ell+1})$, et donc on peut majorer :

$$a_n^\ell \leqslant \begin{cases} k^{n-1} & \text{si } \ell < \log_k n + 1 \\ nk^{n-\ell} & \text{si } \ell \geqslant \log_k n + 1 \end{cases}$$

En majorant a_n par la somme de tous les a_n^ℓ, et en partitionnant selon la position de ℓ par rapport à $\log_k n + 1$, nous obtenons :

$$\begin{aligned} a_n &\leqslant \sum_{2\leqslant \ell < \log_k n+1} a_n^\ell + \sum_{\ell \geqslant \log_k n} a_n^\ell \\ &\leqslant k^{n-1}(\log_k n - 1) + nk^n \sum_{\ell \geqslant \log_k n+1} k^{-\ell} \\ &\leqslant k^{n-1}(\log_k n - 1) + nk^n k^{-\log_k n - 1} \frac{k}{k-1} \\ &= k^{n-1}(\log_k n - 1) + k^{n-1}\frac{k}{k-1} \sim_{n\to\infty} k^{n-1}\log_k n \end{aligned}$$

Donc il existe une constante $b_2 > 0$ telle que pour n assez grand, $a_n \leqslant b_2 k^{n-1}\log_k n$.

Minoration. Nous regardons toujours un mot de taille n de L_k de la forme $s\#w$, en fixant s de taille $|s| = \ell - 1$. Nous découpons w en t blocs de taille $\ell - 1$, avec $t = \lfloor \frac{n-\ell}{\ell-1} \rfloor$. Le dernier bloc restant est de taille $r = (n-\ell) - (\ell-1)t$. L'idée est de minorer les mots de L_k associés à un s de taille $\ell - 1$ en ne regardant que les mots w tels que s^R apparaît dans un des t blocs. Ainsi :

$$\begin{aligned} a_n^s &\geqslant \operatorname{card}\{w : s^R \text{ apparaît dans une des } t \text{ cases de } w, \text{ avec } |w| = n-\ell\} \\ &= k^{n-\ell} - \operatorname{card}\{w : s^R \text{ n'apparaît dans aucune des } t \text{ cases de } w, \text{ avec } |w| = n-\ell\} \\ &= k^{n-\ell} - k^r(k^{\ell-1}-1)^t = k^{n-\ell}\left(1 - \left(1 - \frac{1}{k^{\ell-1}}\right)^t\right) \end{aligned}$$

Puisque $r - (n - \ell) = -t(\ell-1)$.

Comme cette majoration ne dépend que de $|s| = \ell$, nous en déduisons que

$$a_n^\ell \geqslant k^{n-1}\left(1 - \left(1 - \frac{1}{k^{\ell-1}}\right)^t\right).$$

Remarquons que $\left(1 - \frac{1}{k^{\ell-1}}\right)^t = \exp\left(\lfloor \frac{n-\ell}{\ell-1} \rfloor \ln\left(1 - \frac{1}{k^{\ell-1}}\right)\right) \leqslant \exp\left(-\lfloor \frac{n-\ell}{\ell-1} \rfloor \frac{1}{k^{\ell-1}}\right)$.

Ainsi, si $\ell - 1 \leqslant \frac{\log_k n}{2}\}$, alors $-\frac{1}{k^{\ell-1}} \leqslant -\frac{1}{\sqrt{n}}$ et pour $n \geqslant 4$, $n - \ell \geqslant \frac{n}{2}$ et $n - \log_k n \geqslant \frac{n}{2}$. Ainsi : $\left\lfloor \frac{n-\ell}{\ell-1} \right\rfloor \geqslant \frac{n-\ell}{\ell-1} - 1 \geqslant \frac{n}{\log_k n} - 1 \geqslant \frac{n}{2\log_k n}$. Et donc pour $\ell \leqslant \frac{\log_k n}{2} + 1$:

$$(1 - \tfrac{1}{k^{\ell-1}})^t \geqslant (1 - \exp(-\tfrac{\sqrt{n}}{2\log_k n}))\,,$$

où le terme droit ne dépend pas de ℓ, et tend vers 1 pour $n \to \infty$. Pour n assez grand, on peut le minorer par $1/2$: ainsi il existe un rang $n_0 > 0$ indépendant de ℓ tel que pour tout $n \geqslant n_0$ et $\ell \leqslant \frac{\log_k n}{2} + 1$, on ait la minoration $a_n^\ell \geqslant \frac{1}{2}k^{n-1}$.

Par conséquent, pour $n \geqslant n_0$, $a_n \geqslant \sum_{\ell-1 \leqslant \frac{1}{2}\log_k n} a_n^\ell \geqslant \frac{1}{4}k^{n-1}\log_k n$. ■

Corollaire 7.83 .

▶ Le comportement asymptotique de a_n est incompatible avec l'ambiguïté bornée : le langage de Shamir L_k est infiniment ambigu. ◀

Démonstration. Nous supposons que L_k est reconnu par une grammaire G qui est K-ambiguë, et nous rappelons que g_n désigne le nombre d'arbres de dérivation de G étiquetés par un mot de taille n, et que par définition $a_n \leqslant g_n \leqslant Ka_n$. Notons $G(z) = \sum_{n\geqslant 0} g_n z^n$ sa série génératrice, qui est $\mathbb{N}$-algébrique.

Par la proposition précédente, il existe des constantes $b_1, b_2 > 0$ telles que pour n assez grand,

$$b_1 \log_k n \leqslant \frac{a_n}{k^{n-1}} \leqslant b_2 \log_k n$$

Il ne reste alors plus qu'à formaliser l'incompatibilité de cet encadrement, qui contient un terme logarithmique, avec le comportement asymptotique de g_n. Comme G est ($\mathbb{N}$-)algébrique, il existe un entier p tel que pour tout j entre 0 et $p-1$, g_n admet des équivalents de la forme suivante [Fla87, BD15] :

$$g_{np+j} \sim_{n\to+\infty} \frac{C_j}{\Gamma(1+\alpha_j)}(np+j)^{\alpha_j}\beta_j^{np+j}$$

avec $C_j > 0, \beta_j > 0$. Fixons un j entre 0 et $p-1$. En utilisant la Proposition 7.82, la forme de l'équivalent de g_n et $a_n \leqslant g_n \leqslant Ka_n$, nous pouvons trouver des constantes strictement positives c_1 et c_2 telles que pour n assez grand :

$$c_1 k^{np+j} \log_k(np+j) \leqslant (np+j)^{\alpha_j}\beta_j^{np+j} \leqslant c_2 k^{np+j} \log_k(np+j)\,.$$

Et ainsi :

$$c_1 \leqslant \frac{(np+j)^{\alpha_j}}{\log_k(np+j)}\left(\frac{\beta_j}{k}\right)^{np+j} \leqslant c_2\,.$$

Ainsi, la quantité du milieu ne peut pas tendre vers l'infini, ni vers 0. Par croissances comparées, cela implique $\beta_j = k$. Mais alors le terme du milieu tend vers 0 si $\alpha_j \leqslant 0$ et vers l'infini si $\alpha_j > 0$.

Nous obtenons ainsi une contradiction : la grammaire G ne peut pas être d'ambiguïté bornée, et le langage de Shamir L_k est infiniment ambigu. ■

En conclusion de cette ouverture, nous avons réussi sur un exemple à utiliser les techniques analytiques pour établir l'infinie ambiguïté d'un langage algébrique. Il s'agit d'une piste pour de futurs travaux. Notre critère n'est pas suffisamment générique pour s'appliquer à beaucoup d'autres langages.

7.4.4 Diagonale de série $\mathbb{N}$-rationnelle ou $\mathbb{N}$-algébrique : lien avec les langages (algébriques) de Parikh faiblement non ambigus

Dans cette section, nous essayons de capturer plus finement les classes de séries holonomes qui sont reliées aux calculs d'automates de Parikh (à pile). Dans un premier temps, nous définissons les séries $\mathbb{N}$-rationnelles à plusieurs variables, puis la diagonale d'une série.

Nous rappelons que si $\Sigma = \{a_1, \ldots, a_k\}$ est un alphabet à k lettres, et $L \subseteq \Sigma^*$ un langage, la série génératrice multivariée est la série

$$L(x_1, \ldots, x_k) = \sum_{i_1,\ldots,i_k \in \mathbb{N}} \ell_{i_1,\ldots,i_k} x_1^{i_1} \ldots x_k^{i_k}$$

où $\ell_{i_1,\ldots,i_k} \in \mathbb{N}$ est le nombre de mots de L qui ont i_j occurrences de la lette a_j pour $j \in [k]$.

Définition 7.84 (Série $\mathbb{N}$-rationnelle, [Bou06, GP14]).
▶ Une série $L(x_1, \ldots, x_k) = \sum_{i_1,\ldots,i_k \in \mathbb{N}} \ell_{i_1,\ldots,i_k} x_1^{i_1} \ldots x_k^{i_k}$ à coefficients entiers est appelée $\mathbb{N}$-rationnelle s'il existe un langage régulier sur l'alphabet $\Sigma = \{a_1, \ldots, a_k\}$ dont la série génératrice *multivariée* est $L(\boldsymbol{x}) - L(0, \ldots, 0)$.

De façon équivalente, la classe des séries $\mathbb{N}$-rationnelles est le plus petit sous-ensemble de $\mathbb{N}[[x_1, \ldots, x_k]]$ qui contienne $0, x_1, \ldots, x_k$, et qui soit clos par addition, multiplication, et quasi-inverse (*i.e.* si $L(\boldsymbol{x})$ est $\mathbb{N}$-rationnelle, et $L(\mathbf{0}) = 0$, alors $\frac{1}{1-L(\boldsymbol{x})}$ est $\mathbb{N}$-rationnelle). ◀

Remarque 7.85.
▶ La généralisation dans [SS78] de la définition des séries $\mathbb{N}$-rationnelles (puis $\mathbb{N}$-algébriques) *à plusieurs variables* porte sur des séries non commutatives ($xy \neq yx$). ◀

Par définition, les séries $\mathbb{N}$-rationnelles sont ainsi associées aux séries génératrices multivariées des langages réguliers. Nous définissons maintenant la diagonale d'une série :

Définition 7.86 (Diagonale d'une série).
▶ Soit $F(x_1, x_2, \ldots, x_k) = \sum_{i_1,\ldots,i_k \in \mathbb{N}} \ell_{i_1,\ldots,i_k} x_1^{i_1} \ldots x_k^{i_k}$ une série formelle. La diagonale de F, notée ΔF, est la série définie par :

$$\Delta F(x) = \sum_{n \in \mathbb{N}} \ell_{n,n,\ldots,n} x^n$$

◀

Remarque 7.87.
▶ [Lip88] appelle ΔF *la* diagonale de F, ou la diagonale complète, par opposition à *une* diagonale, pour laquelle on n'identifie que deux variables ; par exemple $\Delta_{x_1,x_2} F$ désigne la série $\sum_{i_1,i_3,\ldots,i_k \in \mathbb{N}} \ell_{i_1,i_1,i_3\ldots,i_d} x_1^{i_1} x_3^{i_3} \ldots x_k^{i_k}$. On vérifie aisément que les diagonales sont reliées aux produits d'Hadamard :

$$\Delta F = (F(\boldsymbol{x}) \odot \frac{1}{1 - x_1 \ldots x_k})(x, 1, \ldots, 1)$$
$$F(\boldsymbol{x}) \odot G(\boldsymbol{x}) = \Delta_{x_1,x_1'} \ldots \Delta_{x_k,x_k'} F(\boldsymbol{x}) \cdot G(\boldsymbol{x}')$$

[Lip88] prouve que la diagonale d'une série holonome est holonome. ◀

Nous pouvons prouver que les séries associées aux langages de Parikh faiblement non ambigus constituent une sous-classe des séries holonomes :

Théorème 7.88.
▶ Une série $L(x) = \sum_{n \in \mathbb{N}} \ell_n x^n$ à coefficients entiers est la diagonale d'une série $\mathbb{N}$-rationnelle si et seulement s'il existe un langage de Parikh faiblement non ambigu dont la série génératrice est $L(x) - \ell_0$. ◀

Démonstration. Nous prouvons d'abord l'implication, puis la réciproque. Soit $L(x) = \sum_{n \in \mathbb{N}} \ell_n x^n$ une série à coefficients entiers telle que $L(x) = \Delta F$, avec $F(y_1, y_2, \ldots, y_k)$ qui est $\mathbb{N}$-rationnelle. Par définition des séries $\mathbb{N}$-rationnelles, nous pouvons construire un automate fini $\mathcal{A} = (Q, q_I, \{q_f\}, \delta)$ non ambigu sur l'alphabet $\Sigma = \{a_1, \ldots, a_k\}$ tel que :

— la série génératrice multivariée de $\mathcal{L}(\mathcal{A})$ est $F(y_1, \dots, y_k) - F(0, \dots, 0)$;

— $\mathcal{A}$ n'accepte pas le mot vide ;

— $\mathcal{A}$ a un unique état initial q_I, qui est sans transition entrante, et un unique état final q_f, dont aucune transition ne sort.

Ainsi $[y_1^{i_1} \dots y_k^{i_k}]F$ pour $i_1 + \dots + i_k > 0$ compte le nombre de calculs finaux de l'automate étiquetés par des mots d'image de Parikh $(i_1, \dots, i_k)$, et pour $n > 0$, $\ell_n := [y_1^n \dots y_k^n]F$ compte ceux étiquetés par des mots de longueur nk qui ont le même même nombre d'occurrences n pour chaque lettre.

La seule difficulté pour interpréter la diagonale en terme de mots vient du fait que $y_1, \dots, y_k$ comptent les occurrences de chaque lettre : ainsi $[y_1^n \dots y_k^n]F$ compte dans ce formalisme des mots de longueur nk, et non de longueur n.

L'automate de Parikh $\mathcal{B}$ est donc construit à partir de $\mathcal{A}$ en contractant les calculs de longueur de k en une seule transition. Plus précisément, l'ensemble d'états de $\mathcal{B}$ est Q, son état initial q_I, son état final q_f, son alphabet est $\Gamma \subseteq \delta^k$ l'ensemble des calculs de longueur k de $\mathcal{A}$. Enfin, pour toute paire d'états $p, q \in Q$, pour tout calcul $p \xRightarrow[\pi]{w} q$ de longueur k dans Γ, nous ajoutons la transition $(p, (\pi, \boldsymbol{\pi_{kh}}(w)), q)$ où $\boldsymbol{\pi_{kh}}(w)$ est le vecteur de Parikh de w. Remarquons que $\|\boldsymbol{\pi_{kh}}(w)\|_1 = |w|$.

Ainsi les calculs finaux de $\mathcal{B}$ sont exactement les couples $(\pi, \boldsymbol{v})$ où $\pi \in \Gamma^*$ de longueur $n > 0$ représente un calcul acceptant de $\mathcal{A}$ de longueur nk étiqueté par un mot dont l'image de Parikh est $\boldsymbol{v}$. Comme $\mathcal{A}$ est non ambigu, $\mathcal{B}$ est non ambigu, au sens de [CFM13], donc il restera faiblement non ambigu peu importe le semilinéaire de $\mathcal{B}$.

On associe à $\mathcal{B}$ le semilinéaire représentant la contrainte $y_1 = y_2 = \dots = y_k$. Les calculs acceptants de $\mathcal{B}$ de longueur $n > 0$ sont donc les calculs finaux de $\mathcal{B}$ étiquetés par un couple $(\pi, (n, \dots, n))$ où π est un calcul acceptant de $\mathcal{A}$ étiqueté par un mot de longueur nk de vecteur de Parikh $(n, \dots, n)$. Par non ambiguïté, il y a autant de calculs acceptants de $\mathcal{B}$ de longueur n que de mots dans $\mathcal{L}(A)$ ayant le même nombre n d'occurrences de chaque lettre, et la série génératrice de $\mathcal{L}(\mathcal{B})$ est donc $\sum_n \ell_n x^n - L(0)$.

Réciproquement, soit $\mathcal{A}$ un automate de Parikh faiblement non ambigu, qui ne reconnaît pas le mot vide, et normalisé comme dans la Proposition 6.20, de dimension $2d$, dont les vecteurs sont de la forme $(\boldsymbol{e_i}, \boldsymbol{e_j})$ et le semilinéaire est $\bigwedge_{i=1..d} x_i = x_{d+i}$.

Nous obtenons facilement un automate de Parikh faiblement non ambigu équivalent en remplaçant les vecteurs $(\boldsymbol{e_i}, \boldsymbol{e_j})$ par le vecteur $(\mathbf{1} + \boldsymbol{e_i} - \boldsymbol{e_j}, 1) \in \mathbb{N}^{d+1}$, où $\mathbf{1} \in \mathbb{N}^d$ désigne le vecteur avec des 1 à toutes ses coordonnées, en remplaçant le semilinéaire par $x_1 = x_2 = \dots = x_d = x_{d+1}$: un calcul de l'automate de longueur n étiqueté par un couple $(w, (n + i_1, n + i_2, \dots, n + i_d, n))$ où w est de longueur n simule un calcul de $\mathcal{A}$ étiqueté par le couple $(w, (\boldsymbol{u}, \boldsymbol{v}))$ avec $i_k = u_k - v_k \in [-n, +n]$, pour $k \in [d]$.

En voyant $\mathcal{B}$ comme un automate fini, en représentant ses vecteurs avec des lettres par exemple, nous obtenons que la série

$$F(x, y_1, \dots, y_d, y_{d+1}) = \sum_{n \in \mathbb{N}, i_1, \dots, i_d \in [-n, n]} a(n, i_1, \dots, i_d) x^n y_{d+1}^n y_1^{n+i_1} \dots y_d^{n+i_d}$$

est $\mathbb{N}$-algébrique, où $a(n, i_1, \dots, i_d)$ désigne le nombre de calculs finaux de $\mathcal{B}$ étiquetés par un couple de la forme $(w, (n + i_1, n + i_2, \dots, n + i_d, n))$ avec $|w| = n$. Ainsi pour $n \in \mathbb{N}$, $\Delta F(x) = \sum_n a(n, n, \dots, n) x^n$ compte le nombre de calculs acceptants

de $\mathcal{B}$ de longueur n, et par faible non ambiguïté le nombre de mots de longueur n dans le langage de $\mathcal{B}$. ■

Notons que ce résultat permet directement de dire qu'il s'agit d'une sous-classe stricte des séries holonomes : si ℓ_n désigne le coefficient de la diagonale d'une série $\mathbb{N}$-rationnelle, alors ℓ_n compte des mots d'un langage : si on note Σ l'alphabet de ce langage, alors $\ell_n \leqslant |\Sigma|^n$ pour tout $n \in \mathbb{N}$. Cela signifie que $n!$ qui croît plus vite que toute exponentielle $|\Sigma|^n$, n'est pas le coefficient d'une diagonale de série $\mathbb{N}$-rationnelle, mais $\sum_n n! x^n$ est holonome.

$u_n = n!$ vérifie $u_{n+1} - (n+1)u_n = 0$.

Les diagonales de séries $\mathbb{N}$-rationnelles ont été étudiées en détail par [GP14], à l'aide de pavages unidimensionnels de tuiles de longueurs irrationnelles. Il n'est pas étonnant que les calculs d'un automate de Parikh et les pavages par des tuiles soient comptés par les mêmes séries, car les deux objets sont assez proches : sans entrer dans les détails, car ce n'est pas le but de cette section, on peut transformer un automate de Parikh en système de tuiles, en encodant les états dans les frontières des tuiles, et les vecteurs dans les longueurs des tuiles (en se plaçant par exemple dans $\mathbb{Q}[\alpha_1, \dots, \alpha_d]$, où $\alpha_1, \dots, \alpha_d$ sont d nombres irrationnels indépendants dans $\mathbb{Q}$). Les auteurs [GP14] décrivent alors les formes asymptotiques de tels coefficients :

Proposition 7.89 (Forme des asymptotiques, [GP14, Theorem 4.1 et 4.2]).
▶ Si ℓ_n est le coefficient d'une diagonale de série $\mathbb{N}$-rationnelle, alors il existe un entier m tel que pour tout $j \in [m-1]$:

$$\ell_n \sim_{n\to\infty} C_j \rho_j^n n^{\alpha_j} \log(n)^{\beta_j} \quad \text{avec } n \equiv j \mod m\,,$$

où $\alpha_j \in \mathbb{Q}$, $\beta_j \in \mathbb{N}$, et enfin ρ_j est algébrique.

De plus, si $\rho_j = 1$, alors $f(nm+j)$ est à partir d'un certain rang n_0 égal à un polynôme en n. ◀

Il serait intéressant de trouver des langages de Parikh dont la série génératrice est holonome mais n'est pas une diagonale de série $\mathbb{N}$-rationnelle, pour prouver leur intrinsèque ambiguïté.

Par ailleurs, en définissant de façon analogue les séries $\mathbb{N}$-algébriques multivariées, qui sont associées aux séries génératrices multivariées des langages algébriques non ambigus, nous pouvons démontrer par la même technique de preuve qu'une série $L(x) = \sum_{n\in\mathbb{N}} \ell_n x^n$ à coefficients entiers est la diagonale d'une série $\mathbb{N}$-algébrique si et seulement s'il existe un langage algébrique de Parikh faiblement non ambigu dont la série génératrice est $L(x) - \ell_0$. Une autre ouverture possible serait de séparer les différentes classes de séries génératrices. Nous conjecturons que la classe des séries $\mathbb{N}$-algébriques et les diagonales de séries $\mathbb{N}$-rationnelles sont incomparables : une direction est facile, la série génératrice du langage Ω_3 de la Proposition 7.19 n'est pas $\mathbb{N}$-algébrique, mais le langage est reconnu par un automate de Parikh faiblement non ambigu, donc sa série est bien une diagonale de série $\mathbb{N}$-rationnelle. Pour l'autre direction, les auteurs de [GP14] conjecturent que les nombres de Catalan ne sont pas exprimables comme coefficients de la diagonale d'une série $\mathbb{N}$-rationnelle.

Par contre les nombres de Catalan s'expriment comme la diagonale d'une série rationnelle.

7.4.5 Tentatives échouées d'extensions

Comme nous l'avons remarqué ci-dessus dans le cas particulier des diagonales de séries $\mathbb{N}$-rationnelles, si $L(x) = \sum_n \ell_n x^n$ est la série génératrice d'un langage

quelconque $\mathcal{L} \subseteq \Sigma^*$, alors pour tout $n \in \mathbb{N}$, $\ell_n \leqslant |\Sigma|^n$: autrement dit la croissance des coefficients de $L(x)$ est bornée par une exponentielle. Les séries holonomes à coefficients entiers qui vérifient cette propriété appartiennent à la classe des *G-fonctions* qui est une sous-classe stricte des séries holonomes : nous ne pouvons donc pas atteindre toutes les séries holonomes avec des séries génératrices de langages.

Jusqu'ici, nous avons rencontré des séries $\mathbb{N}$-rationnelles, $\mathbb{N}$-algébriques, et des diagonales de séries $\mathbb{N}$-rationnelles ou $\mathbb{N}$-algébriques. Je présente dans cette partie des ouvertures deux extensions que j'ai regardées en cherchant à étendre (en vain) les classes de langages pour trouver d'autres classes de séries holonomes que celles-là.

Première idée : étendre les ensembles semilinéaires

Une première idée consiste à remarquer que pour les langages de Parikh, nous avons obtenu les séries génératrices en intersectant un langage régulier ou algébrique non ambigu avec des contraintes semilinéaires. Pouvons-nous généraliser cette idée avec des contraintes non semilinéaires, c'est-à-dire pouvons-nous trouver un ensemble $E \subseteq \mathbb{N}^d$ dont la série support

$$E(\boldsymbol{y}) = \sum_{\boldsymbol{i} \in E} y_1^{i_1} \dots y_d^{i_d}$$

soit holonome, mais tel que E ne soit pas semilinéaire ? Malheureusement ce n'est pas possible.

Lemme 7.90 ([BC17]).
▶ Si $F(\boldsymbol{x}) = \sum_{\boldsymbol{\nu} \in \mathbb{N}^d} a_{\boldsymbol{\nu}} \boldsymbol{x}^{\boldsymbol{\nu}}$ est holonome et ses coefficients $a_{\boldsymbol{\nu}}$ appartiennent à un ensemble fini $E \subseteq \mathbb{N}$, alors $F(\boldsymbol{x})$ est une fraction rationnelle. ◀

[BC17] montrent en fait une version plus générale, avec E un sous-ensemble fini d'un corps K de caractéristique nulle.

Proposition 7.91 ([Woo15, BC17]).
▶ Soit $E(\boldsymbol{x}) = \sum_{\boldsymbol{\nu} \in \mathbb{N}^d} a_{\boldsymbol{\nu}} \boldsymbol{x}^{\boldsymbol{\nu}}$ une série telle que pour tout $\boldsymbol{\nu} \in \mathbb{N}^d$, $a_{\boldsymbol{\nu}} \in \{0, 1\}$. Alors $E(\boldsymbol{x})$ est rationnelle si et seulement c'est la série support d'un ensemble semilinéaire (autrement dit l'ensemble $E = \{\boldsymbol{\nu} \in \mathbb{N}^d \mid a_{\boldsymbol{\nu}} = 1\}$ est semilinéaire). ◀

En combinant ces deux résultats, nous obtenons que si la série support d'un ensemble E est holonome, alors elle est forcément rationnelle, et en fait E est semilinéaire.

Deuxième idée : extension aux automates d'arbres de Parikh

Nous avons voulu savoir s'il était possible d'obtenir une classe de séries différente en regardant les séries associées aux automates d'arbres de Parikh non ambigus. Ce n'est pas non plus le cas.

Nous redéfinissons ici les arbres associés à un ensemble d'opérateurs : il s'agit de la même notion que la Définition 2.2 du chapitre 2, mais nous avions noté S l'ensemble d'opérateurs, et a l'arité, qui dans le monde des langages et des grammaires sont plutôt réservés à des symboles de pile ou de lettres.

Soit $\mathcal{E} = (E, ar)$ un tuple, où E est un ensemble de symboles, et $ar : E \to \mathbb{N}$ est une fonction qui associé à chaque symbole $f \in E$ un nombre $ar(f)$, appelé son arité. Nous rappelons que les symboles d'arité 0 sont appelées des feuilles. Nous supposons qu'il existe au moins une feuille dans $\mathcal{E}$.

Nous écrirons $\mathcal{E}$ sous la forme $\{f(ar(f)) : f \in E\}$. Par exemple, $\mathcal{E} = \{a(0), b(0), f(2)\}$ est un ensemble qui contient un noeud binaire, et deux feuilles.

Nous écrirons abusivement $f \in \mathcal{E}$ à la place de $f \in E$. Nous rappelons la définition formelle d'un arbre construit sur $\mathcal{E}$:

Définition 7.92 .

▶ L'ensemble des arbres construits sur $\mathcal{E}$, noté $\mathcal{T}(\mathcal{E})$, est défini par induction :

- pour toute feuille $a \in \mathcal{E}$, $a \in \mathcal{T}(\mathcal{E})$
- si $f \in E$ est d'arité $r = ar(f)$, et $t_1, \ldots, t_r$ sont dans $\mathcal{T}(\mathcal{E})$, alors $(f, t_1, \ldots, t_r) \in \mathcal{T}(\mathcal{E})$.

◀

Définition 7.93 .

▶ Un *automate d'arbre top-down de Parikh* de dimension d sur une famille d'arbres $Tree_{\mathcal{E}}$ est un tuple $(Q, \mathcal{E}, I, \Delta, C)$ où

- Q est l'ensemble d'états
- $I \subseteq Q$ est l'ensemble d'états initiaux
- Δ est l'ensemble de dérivations. Chaque transition est de la forme $(f, q) \xrightarrow{\boldsymbol{v}} q_1, \ldots, q_r$, où $f \in E$, $ar(f) = r \geqslant 0$, $\boldsymbol{v} \in \mathbb{N}^d$, $q, q_1, \ldots, q_r \in Q$.
  Remarquons que si $ar(f) = 0$, alors les transitions sont de la forme $(f, q) \xrightarrow{\boldsymbol{v}}$.
- $C \subseteq \mathbb{N}^d$ est l'ensemble semilinéaire d'acceptation

◀

Définition 7.94 .

▶ Soit $t \in Tree_{\mathcal{E}}$ et $\mathcal{A} = (Q, \mathcal{E}, I, \Delta, C)$ un automate d'arbre de Parikh. L'ensemble des calculs de $\mathcal{A}$ sur l'arbre t commençant en l'état $q \in Q$, noté $Runs_{\mathcal{A}}(t, q)$, est défini comme suite :

- si $ht(t) = 0$, *i.e.* t est une feuille, et $((t, q) \xrightarrow{\boldsymbol{v}}) \in \Delta$, alors $((t, q), \boldsymbol{v}) \in Runs_{\mathcal{A}}(t, q)$
- si $t = (f, t_1, \ldots, t_r)$ avec $ar(t) = r > 0$ et $(f, q) \xrightarrow{\boldsymbol{v}} q_1, \ldots, q_r \in \Delta$, et $\pi_i \in Runs_{\mathcal{A}}(t_i, q_i)$ pour $i = 1 \ldots r$, alors $\pi = ((f, q), \boldsymbol{v}, \pi_1, \ldots, \pi_r) \in Runs_{\mathcal{A}}(t, q)$

En d'autres termes, un calcul de $\mathcal{A}$ sur l'arbre t est un étiquetage des noeuds de t avec des états de Q et des vecteurs de $\mathbb{N}^d$ qui respecte les règles de transition de l'automate $\mathcal{A}$.

Un calcul de $\mathcal{A}$ sur t est *acceptant* si la racine est étiquetée par un état initial de I, et si la somme des vecteurs apparaissant dans l'étiquetage appartient au semilinéaire S. L'arbre t est dit accepté par l'automate.

Le langage d'un automate est l'ensemble *des arbres* acceptés par l'automate. ◀

Exemple 7.95 .

▶ Soit $\mathcal{E} = \{a(0), b(0), f(2)\}$, $\mathcal{A} = (Q, \mathcal{E}, I, \Delta, S)$ avec $Q = \{q_1, q_2\}$, $I = \{q_1\}$, $S = \{(n, n) : n \in \mathbb{N}\}$, et Δ l'ensemble de règles :

$$(f, q_1) \xrightarrow{(0,0)} q_2, q_2 \qquad (b, q_1) \xrightarrow{(0,1)} \qquad (a, q_2) \xrightarrow{(1,0)}$$
$$(f, q_2) \xrightarrow{(0,0)} q_1, q_1 \qquad (b, q_2) \xrightarrow{(0,1)}$$

L'automate reconnaît les arbres de $Tree_{\mathcal{E}}$ qui ont le même nombre de a et de b, et tels que les feuilles a sont à une profondeur impaire. ◀

Définition 7.96 .
▶ Un automate d'arbre de Parikh est dit *non ambigu* si tout arbre accepté a un unique calcul acceptant. ◀

Comme pour les automates de Parikh, on peut définir la classe des automates d'arbre de Parikh généralisés, en remplaçant les vecteurs par des ensembles semilinéaires de $\mathbb{N}^d$.

Proposition 7.97 .
▶ La classe des automates d'arbres de Parikh (non ambigus) généralisés est équivalente à celle des automates d'arbres de Parikh (non ambigus). ◀

Démonstration. Il s'agit de la même preuve que pour les automates de Parikh. ■

Les automates d'arbres de Parikh non ambigus ont une série génératrice holonome :

Proposition 7.98 .
▶ Soit $\mathcal{A}$ un automate d'arbre de Parikh non ambigu. Notons $f(x) = \sum_{n\in\mathbb{N}} a_n x^n$ la série génératrice de $\mathcal{A}$ comptant le nombre *d'arbres* acceptés par $\mathcal{A}$ à n noeuds. Alors f est holonome. ◀

Démonstration. Il s'agit de la même technique de preuve que pour les automates de Parikh faiblement non ambigus, (voir le Théorème 7.45 ou le Théorème 7.48). ■

En revanche, les séries génératrices issues d'automates d'arbres de Parikh non ambigus ne constituent pas une classe plus grande de séries génératrices que celles que nous avons vues jusqu'à présent.

Définition 7.99 (Description préfixe d'un calcul).
▶ Soit $\mathcal{A} = (Q, \mathcal{E}, I, \Delta, C)$ un automate d'arbre de Parikh. Notons $\Sigma = E \times Q \times V$, avec V qui est l'ensemble des vecteurs étiquetant les transitions de $\mathcal{A}$. La description préfixe d'un calcul π de $\mathcal{A}$ est le mot $Prefix(\pi) \in \Sigma^*$, défini par :

- si $\pi = ((t, q), \boldsymbol{v})$, avec t une feuille, alors $Prefix(\pi) := (t, q, \boldsymbol{v})$
- si $\pi = ((f, q), \boldsymbol{v}, \pi_1, \ldots, \pi_r)$, alors

$$Prefix(\pi) := (f, q, \boldsymbol{v})Prefix(\pi_1) \ldots Prefix(\pi_r)\,.$$

Il est facile de voir qu'à partir de $Prefix(\pi)$ il est possible de reconstruire π, car tout symbole de $\mathcal{E}$ a une arité fixée et $|\pi| = |Prefix(\pi)|$ (autrement dit la longueur de $Prefix(\pi)$ est égale au nombre de noeuds dans π).

Nous appelons $Prefix(\mathcal{A})$ le langage des préfixes des calculs acceptant de $\mathcal{A}$. ◀

Proposition 7.100 .
▶ Si $\mathcal{A}$ est un automate d'arbre de Parikh non ambigu, alors le langage $Prefix(\mathcal{A})$ est reconnu par une grammaire hors-contexte contrainte faiblement non ambiguë. ◀

Démonstration. À partir de l'automate $\mathcal{A} = (Q, \mathcal{E}, I, \Delta, C)$, nous construisons la grammaire hors-contexte contrainte $G = (N, \Sigma, S, D, C)$, où $\Sigma = \{(f, q, \boldsymbol{v}) : f \in \mathcal{E}, q \in Q, \boldsymbol{v} \in V\}$, $N = \{[f, q] : f \in \mathcal{E}, q \in Q\} \cup \{S\}$, S est l'axiome, et D est défini par :

— pour tout état initial $q \in I$, pour tout $f \in \mathcal{E}$, nous ajoutons à D la dérivation $S \xrightarrow{\mathbf{0}} [f, q]$,

— pour toute transition de la forme $(f, q) \xrightarrow{\boldsymbol{v}} q_1, \ldots, q_r \in \Delta$ avec $r > 0$ et pout tout $(f_1, \ldots, f_r) \in \mathcal{E}^r$, nous ajoutons à D la dérivation $[f, q] \xrightarrow{\boldsymbol{v}} (f, q, \boldsymbol{v})[f_1, q_1] \ldots [f_r, q_r]$,

— pour toute transition de la forme $((a, q) \xrightarrow{\boldsymbol{v}}) \in \Delta$ (dans ce cas a est une feuille de $\mathcal{E}$), nous ajoutons à D la dérivation $[a, q] \xrightarrow{\boldsymbol{v}} (a, q, \boldsymbol{v})$.

On montre facilement par induction que les arbres de dérivation terminaux à partir du symbole $[f, q]$ simulent exactement les calculs de $\mathcal{A}$ sur des arbres de racine f, commençant dans l'état q ; de plus, si un arbre de dérivation terminal simule un calcul π, alors il reconnaît le mot $Prefix(\pi)$, et la somme des vecteurs de la dérivation est la même que celle des vecteurs du calcul π.

La faible non ambiguïté de G vient directement de celle de $\mathcal{A}$ et de la bijection entre les calculs de $\mathcal{A}$ et les arbres de dérivation de G. ■

Ainsi, les séries génératrices des langages d'arbres de Parikh non ambigus n'étendent pas la classe des séries vues précédemment.

7.5 Conclusion

Dans ce chapitre, nous avons étendu le lien entre l'ambiguïté de certains langages et leurs séries génératrices.

Pour cela, nous avons d'abord réexploré le lien entre les séries et les langages algébriques non ambigus, initié par Flajolet dans [Fla87]. De façon surprenante, nous avons pu aborder deux questions ouvertes laissées par Flajolet à la fin de son article :

Aucun de ces deux résultats n'a encore été publié.

— nous avons réussi à capturer par des arguments sur les séries génératrices l'intrinsèque ambiguïté de certains langages bornés ;

— nous avons pu redémontrer par des arguments analytiques l'infinie ambiguïté d'un langage.

Puis nous nous sommes attachés à étendre les techniques de [Fla87] à une classe de langages strictement plus expressive que les langages algébriques non ambigus. Nous pouvons résumer notre approche, en tenant compte de ce qui a été dit dans les ouvertures, par le schéma de la Figure 7.3.

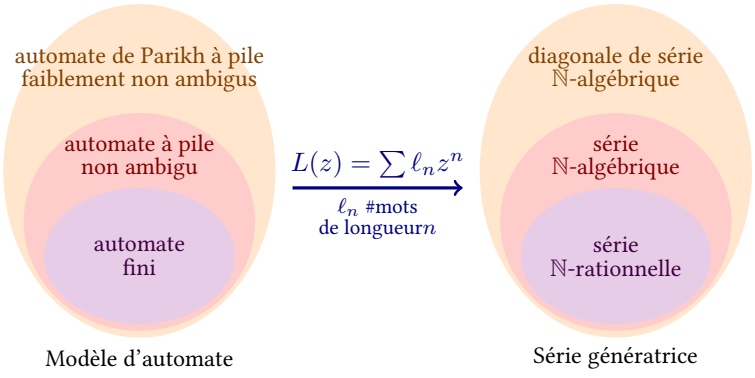


Figure 7.3. Schéma récapitulatif de la correspondance entre séries génératrices et langages non ambigus

Cette nouvelle correspondance n'a été utilisée pour l'instant que dans l'unique but de prouver l'intrinsèque ambiguïté de certains langages. Pourtant, elle a aussi de fortes conséquences sur l'algorithmique des automates de Parikh, que nous explorons dans le chapitre suivant.

Ce chapitre est en partie issu de l'article de conférence [BCKN20].

8

Le problème de l'inclusion des automates de Parikh faiblement non ambigus

8.1 Introduction

8.1.1 Introduction au problème de l'inclusion

Dans ce chapitre, nous étudions une conséquence algorithmique de la propriété d'holonomie des séries de comptage des automates de Parikh faiblement non ambigus. Le problème de l'inclusion est le problème de décision suivant :

`entrée` : deux automates $\mathcal{A}$ et $\mathcal{B}$

`sortie` : est-ce que $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$?

La décidabilité et la complexité de ce problème dépendent des classes d'automates dont sont issus $\mathcal{A}$ et $\mathcal{B}$. Si $\mathcal{A}$ et $\mathcal{B}$ sont des automates finis, le problème est classiquement décidable, et est PSPACE-complet [MS72]. Si les automates sont non ambigus, le problème est décidable en temps polynomial [SI85], par un argument de comptage que nous allons développer en détail dans cette introduction. Si $\mathcal{A}$ et $\mathcal{B}$ sont des grammaires hors-contextes, ce problème est indécidable, car l'universalité est déjà indécidable [BHPS61, Theorem 6.2]. Le problème reste indécidable si $\mathcal{A}$ et $\mathcal{B}$ sont des grammaires déterministes [GG66, Theorem 5.3], et donc *a fortiori* si les grammaires sont non ambiguës ([AN00] fournit une preuve directe dans le cas non ambigu, par réduction du problème de correspondance de Post). D'autres variantes ont été étudiées, en mélangeant les classes (par exemple si $\mathcal{A}$ est une grammaire hors-contexte quelconque, et $\mathcal{B}$ un automate fini, le problème est EXPTIME-complet [KI92]).

Une approche simple pour aborder le problème de l'inclusion, d'un point de vue purement automate, consiste à utiliser l'équivalence suivante :

$$\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B}) \Leftrightarrow \mathcal{L}(\mathcal{A}) \cap \overline{\mathcal{L}(\mathcal{B})} = \emptyset\,,$$

où $\overline{\mathcal{L}(\mathcal{B})}$ désigne le complémentaire de $\mathcal{L}(\mathcal{B})$. Il suffit pour décider l'inclusion de calculer ce complémentaire, lorsque c'est possible, puis de calculer l'intersection,

et enfin de tester le vide du langage obtenu. Cette approche souffre du recours à la complémentation : même dans le cas des automates finis qui sont clos par complémentaire, cette opération a un coût exponentiel ; par ailleurs pour des classes plus compliquées, le calcul du complémentaire n'est pas pas forcément faisable (les langages algébriques (non ambigus) ne sont pas clos par complémentaire [HU66], et nous ne savons pas si les automates de Parikh faiblement non ambigus sont clos par complémentaire). Pour contourner ce problème, dans le cadre des automates finis non ambigus, Stearns et Hunt [SI85] ont utilisé plutôt l'équivalence suivante :

$$\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B}) \Leftrightarrow \mathcal{L}(\mathcal{A}) \cap \overline{\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})} = \emptyset\,.$$

Mathématiquement, il s'agit exactement de la même égalité ; en effet en utilisant les lois de De Morgan, $\mathcal{L}(\mathcal{A}) \cap \overline{\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})} = \mathcal{L}(\mathcal{A}) \cap \overline{\mathcal{L}(\mathcal{B})}$. Cependant, considérer l'intersection $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$ est intéressant car l'inclusion $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B}) \subseteq \mathcal{L}(\mathcal{A})$ est toujours vraie, avec égalité si et seulement si $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. Par conséquent, pour décider le problème de l'inclusion, il suffit, pour tout $n \in \mathbb{N}$, de vérifier qu'il y a bien autant de mots de longueur n dans $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$ que de mots de longueur n dans $\mathcal{L}(\mathcal{A})$. Comme $\mathcal{A}$ et $\mathcal{B}$ sont des automates finis non ambigus, $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$ est reconnu par un automate fini non ambigu $\mathcal{C}$, obtenu en utilisant la construction produit. Notons $A(x) = \sum_n a_n x^n$ et $C(x) = \sum_n c_n x^n$ les séries génératrices de $\mathcal{L}(\mathcal{A})$ et $\mathcal{L}(\mathcal{C}) = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$. Le raisonnement par dénombrement de Stearns et Hunt s'écrit sous la forme :

$$\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B}) \Leftrightarrow \forall n \in \mathbb{N}, a_n - c_n = 0\,.$$

La non ambiguïté et l'astuce de l'intersection permettent ainsi de remplacer le calcul des mots d'un complémentaire, qui nécessite une construction exponentielle, par le dénombrement des mots de ce complémentaire, qui est bien plus simple. En effet, en utilisant la non ambiguïté des automates, il est possible de montrer que la suite (a_n) (resp. (c_n)) satisfait une récurrence linéaire à coefficients constants, d'ordre borné par $|Q_{\mathcal{A}}|$ (resp. $|Q_{\mathcal{A}}||Q_{\mathcal{B}}|$), où $|Q_{\mathcal{A}}|$ et $|Q_{\mathcal{B}}|$ désignent le nombre d'états de $\mathcal{A}$ et $\mathcal{B}$. Les suites de cette forme sont closes par soustraction, si bien que $d_n := a_n - c_n$ satisfait une équation de la forme :

$$\forall n \in \mathbb{N},\ d_{n+r} = \frac{1}{a_r}(a_{r-1}d_{n+r-1} + \ldots + a_0 d_n)\,,$$

avec $a_r \neq 0$, et $r \leqslant |Q_{\mathcal{A}}||Q_{\mathcal{B}}| + |Q_{\mathcal{A}}|$. Cette égalité implique que (d_n) est la suite nulle si et seulement si ses r premiers termes sont nuls. Stearns et Hunt ont donc démontré l'équivalence suivante :

$$\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B}) \Leftrightarrow \forall n \in \{0, \ldots, (|Q_{\mathcal{A}}|+1)|Q_{\mathcal{B}}| - 1\},\ a_n - c_n = 0\,.$$

Il suffit donc pour résoudre le problème de l'inclusion de comparer les nombres de mots acceptés par $\mathcal{A}$ et $\mathcal{C}$ pour toutes les tailles inférieures à $(|Q_{\mathcal{A}}|+1)|Q_{\mathcal{B}}| - 1$, ce qui se fait en temps polynomial en la taille de $\mathcal{A}$ et de $\mathcal{B}$. Stearns et Hunt déduisent aussi de cet argument que si $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{B})$, alors il existe un mot w de longueur plus petite que $|Q_{\mathcal{A}}||Q_{\mathcal{B}}| + |Q_{\mathcal{A}}|$, tel que $w \in \mathcal{L}(\mathcal{A})$ mais $w \notin \mathcal{L}(\mathcal{B})$.

Dans ce chapitre, nous cherchons à étendre la méthode de Stearns et Hunt, qui s'est avérée fructueuse dans le cas des automates finis, aux automates de Parikh non ambigus. Le problème est indécidable pour des automates de Parikh quelconques ; il

Rappelons que l'universalité des automates de Parikh est indécidable [KR02]

est co-NEXP-complet pour les automates de Parikh déterministes [FGM19], lorsque l'ensemble semilinéaire est représenté sous la forme d'une formule de Presburger existentielle. La décidabilité du problème de l'inclusion pour les automates de Parikh faiblement non ambigus peut se déduire des travaux de [CM17] sur la classe RCM, mais les auteurs ne fournissent pas de borne de complexité, et passent sous silence la complication due aux racines du polynôme de tête de la récurrence.

Notre but est donc de déterminer une borne $n_{\max}(\mathcal{A}, \mathcal{B})$ telle que :

$$\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B}) \Leftrightarrow \forall n \in \{0, \dots, n_{\max}(\mathcal{A}, \mathcal{B})\},\ a_n - c_n = 0\,.$$

Si la méthode de Stearns et Hunt s'adapte aux automates de Parikh faiblement non ambigus, elle se confronte cependant aux difficultés suivantes :

— les séries génératrices $A(x)$ et $C(x)$ ne s'obtiennent plus directement à partir des automates en résolvant un simple système linéaire. En effet, ces séries ne sont plus rationnelles, mais holonomes, et l'opération utilisée pour obtenir leurs équations différentielles, que ce soit par un produit d'Hadamard ou une diagonale de série, n'est pas triviale.

— la suite $d_n = a_n - c_n$ satisfait une équation de récurrence linéaire *à coefficients polynomiaux*, et non plus constants, de la forme :

$$p_r(n)d_{n+r} = \sum_{k=0}^{r-1} p_k(n)d_{n+k}$$

avec $p_r(n)$ qui n'est pas le polynôme nul. Les racines du polynôme $p_r(n)$ compliquent le comportement de la suite (d_n) : il ne suffit plus que les r premiers termes de d_n soient nuls pour que la suite soit nulle. Par exemple, le polynôme $D(x) = x^{1000}$ satisfait l'équation $1000D(x) - x\partial_x D(x) = 0$, et la récurrence $(n - 1000)d_n = 0$. Il est clair que vérifier que $d_0 = 0$ ne suffit pas à affirmer que (d_n) est la suite nulle. Pour pouvoir s'assurer que la suite (d_n) est nulle, il faut dépasser à la fois l'ordre de la récurrence et la plus grande racine entière du polynôme de tête. Pour borner cette racine, nous devons borner aussi la taille des coefficients des polynômes de la récurrence.

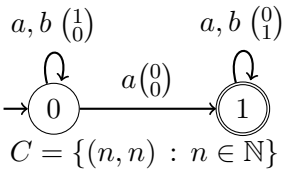
Ces deux difficultés compliquent considérablement l'analyse du problème de l'inclusion dans le cas des automates de Parikh faiblement non ambigus.

8.1.2 Introduction à l'algorithme de Lipshitz

Comme nous l'avons vu aux chapitres précédents, la propriété principale pour démontrer que la série génératrice d'un automate de Parikh non ambigu est holonome est la stabilité des séries holonomes par *diagonale* (ou produit d'Hadamard). Il s'agit d'une opération non triviale sur les fonctions holonomes, qui est toujours l'objet de recherche active en calcul formel. La clôture des fonctions holonomes par diagonale a été démontrée par Lipshitz en 1988 [Lip88] ; sa preuve repose sur des séries formelles, et consiste à interpréter la diagonale d'une série comme une extraction de coefficients. Cette extraction de coefficients, du point de vue analyse complexe, avec la formule de Cauchy, s'exprime sous la forme d'une intégrale (ou période), ce qui a donné lieu à de nouvelles méthodes de calcul de diagonale, plus récentes et efficaces, fondées sur le *télescopage créatif* ([Chy14, BCLS18]). Nous nous sommes limités dans cette

thèse à l'algorithme historique de Lipshitz [Lip88], qui a l'avantage d'être facile à comprendre et à étudier (mais s'avère peu efficace en pratique).

Pour donner une idée du principe de l'algorithme de Lipshitz, nous illustrons son fonctionnement dans cette introduction, sur un exemple simple. Soit $\mathcal{A}_{\text{ex}}$ l'automate de Parikh non ambigu suivant :



L'automate $\mathcal{A}_{\text{ex}}$ reconnaît les mots de longueur impaire qui ont un a au milieu. Ce langage est en fait algébrique non ambigu, et sa série génératrice $L(x) = \frac{x}{1-4x^2}$ est même rationnelle. Regardons comment l'algorithme de Lipshitz permet de trouver une équation différentielle satisfaite par $L(x)$.

Comme nous l'avons démontré au chapitre 7, en notant $A(x, y_1, y_2) = \frac{x}{(1-2xy_1)(1-2xy_2)}$ la série multivariée des calculs de l'automate, et $C(y_1, y_2) = \frac{1}{1-y_1y_2}$ la série du semilinéaire, la série $L(x)$ s'exprime comme le produit d'Hadamard $A(x, y_1, y_2) \odot \frac{1}{1-x}C(y_1, y_2)$, spécialisé en $y_1 = y_2 = 1$. L'idée de Lipshitz est d'exprimer le produit d'Hadamard sous la forme d'une extraction de coefficient :

$$A(x, y_1, y_2) \odot \frac{1}{1-x}C(y_1, y_2) = [u_1^{-1}u_2^{-1}]\frac{1}{u_1u_2}A(x, u_1, u_2)C(y_1/u_1, y_2/u_2)\,.$$

Même si nous sortons du monde des séries entières en écrivant $C(y_1/u_1, y_2/u_2)$, des arguments algébriques permettent d'étendre la notion d'holonomie à ce type de séries formelles. Plutôt que de calculer ce produit d'Hadamard en trois variables, pour évaluer deux variables à 1 ensuite, nous effectuons la substitution avant l'extraction de coefficient (nous justifierons dans la suite que nous avons le droit de le faire, et que les objets considérés sont bien définis). Ainsi :

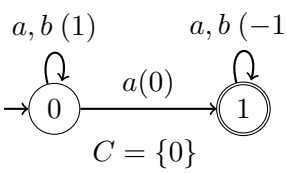
$$L(x) = [u_1^{-1}u_2^{-1}]\frac{1}{u_1u_2}A(x, u_1, u_2)C(u_1^{-1}, u_2^{-1})\,.$$

Pour que les calculs restent raisonnables dans cette introduction, nous simplifions encore cette formule. En remarquant que la contrainte du semilinéaire demande l'égalité des exposants des variables y_1 et y_2, qui s'exprime déjà par une extraction de coefficient bien choisie, nous pouvons vérifier simplement que :

$$L(x) = [y^{-1}]\tfrac{1}{y}A(x, y, y^{-1}) = [y^{-1}]\frac{x}{(1-2xy)(y-2x)}\,.$$

Remarque 8.1.

▶ Cette formule a une interprétation simple si on considère une généralisation équivalente aux automates de Parikh faiblement non ambigus, avec des vecteurs dans $\mathbb{Z}^d$ (sans contrainte de positivité sur les vecteurs étiquetant les calculs). Ainsi l'automate suivant reconnaît le même langage que $\mathcal{A}_{\text{ex}}$:



et la série des calculs finaux de cet automate est bien $A(x, y, y^{-1})$. ◀

Nous posons $F = \frac{P}{Q}$ avec $P = x$ et $Q = (1 - 2xy)(y - 2x)$. L'argument de Lipshitz consiste à remarquer que si on arrive à trouver une équation différentielle pour F de la forme :

$$\sum_{i=i_0}^{r} p_i(x, \partial_x)\partial_y^i F(x, y) = 0$$

avec les p_i des opérateurs différentiels, *qui ne dépendent pas de* y, et $p_{i_0}(x, \partial_x) \neq 0$, alors $p_{i_0}(x, \partial_x)L(x) = 0$.

La preuve de Lipshitz dit, par un argument de dimension, qu'il est toujours possible de trouver une telle équation pour F, en considérant la famille des polynômes $Q^{N+1}\partial_x^j\partial_y^i F$ avec $i + j < N$, pour des valeurs croissantes de N. Vérifions-le sur notre exemple : pour $N = 3$, nous demandons à Maple de résoudre le système suivant dans $\mathbb{Q}(x)$:

$$\lambda_1 Q^4 F + \lambda_2 Q^4 \partial_x F + \lambda_3 Q^4 \partial_y F + \lambda_4 Q^4 \partial_x^2 F + \lambda_5 Q^4 \partial_x\partial_y F + \lambda_6 Q^4 \partial_y^2 F = 0\,.$$

Nous trouvons alors la relation de dépendance suivante (le gras est juste utilisé pour rendre l'équation plus lisible) :

$$\begin{aligned} 0 = 16\ &\left(4\,x^2+3\right) x^4 \boldsymbol{Q^4\partial_x F} \\ &+ 2\,x^3\left(2\,x-1\right)\left(2\,x+1\right)\left(4\,x^2+1\right) \boldsymbol{Q^4\partial_x^2 F} \\ &+ \left(2\,x-1\right)\left(2\,x+1\right)\left(16\,x^4+16\,x^2+1\right) \boldsymbol{Q^4\partial_y F} \\ &+ x\left(4\,x^2+1\right)\left(2\,x-1\right)^2\left(2\,x+1\right)^2 \boldsymbol{Q^4\partial_x\partial_y F} \end{aligned}$$

En extrayant le coefficient de plus petit degré en ∂_y (les deux premières lignes), puis en divisant par $2x^3Q^4$, nous obtenons l'équation différentielle suivante satisfaite par $L(x)$:

$$\left(2\,x-1\right)\left(2\,x+1\right)\left(4\,x^2+1\right)\partial_x^2 L(x) + 8x\left(4\,x^2+3\right)\partial_x L(x) = 0\,.$$

On vérifie facilement que $L(x) = \frac{x}{1-4x^2}$ est bien solution de cette équation différentielle. On remarque que la méthode de Lipshitz ne donne pas une équation minimale satisfaite par $L(x)$; en effet, en tant que fraction rationnelle, $L(x)$ vérifie une équation du premier ordre :

$$x(1-2x)(1+2x)\partial_x L(x) - (1+4x^2)L(x) = 0\,.$$

Nous avons ainsi vu dans cet exemple introductif que la méthode de Lipshitz peut se résumer à trouver une relation de dépendance sur des polynômes bien choisis, obtenus en dérivant une fraction rationnelle issue d'un automate. Une partie centrale de ce chapitre consiste ainsi à analyser cet algorithme de Lipshitz, afin d'obtenir des bornes sur les degrés, l'ordre et surtout la taille des coefficients de l'équation différentielle satisfaite par la série génératrice d'un automate de Parikh non ambigu.

Nous rappelons enfin que l'idée de ce chapitre n'est pas d'implémenter l'algorithme de Lisphitz, ni d'autres algorithmes de calcul formel pour résoudre le problème de l'inclusion. Nous cherchons juste à obtenir des bornes sur les tailles des équations différentielles en jeu, pour répondre au problème de l'inclusion par une simple énumération de mots reconnus par les automates.

8.1.3 Cadre de travail, notations, et plan du chapitre

Dans la suite de ce chapitre, un automate de Parikh faiblement inambigu de dimension $d \geqslant 1$ est donné sous la forme d'un tuple $\mathcal{A} = (\Sigma, Q, q_I, F, C, \Delta)$ où on rappelle que Σ est l'alphabet, Q l'ensemble d'états, $q_I \in Q$ l'état initial, $F \subseteq Q$ l'ensemble des états finaux, $C \subseteq \mathbb{N}^d$ son ensemble semilinéaire d'acceptation, et $\Delta \subseteq Q \times (\Sigma \times \mathbb{N}^d) \times Q$ la relation de transition. On suppose que C est donné sous une forme inambiguë $C = \biguplus_{i=1}^{p} \boldsymbol{c_i} + P_i^*$.

En particulier il n'y a pas de ε-transitions

Définition 8.2 (Taille d'un automate de Parikh faiblement inambigu).
▶ Nous introduisons les notations suivantes pour quantifier la taille de $\mathcal{A}$:
- on désigne par $|\mathcal{A}| := |Q| + |\Delta| + p + \sum_i |P_i|$ la taille unaire de $\mathcal{A}$;
- pour $j \in [d]$, on note $\|\mathcal{A}\|_{j,\infty}$ la valeur maximale qui apparaît en coordonnée j dans les vecteurs de Δ, des P_i et des $\boldsymbol{c_i}$;
- on note $\|\mathcal{A}\|_\infty = \max_j \|\mathcal{A}\|_{j,\infty}$ la coordonnée maximale qui apparaît dans toutes les composantes des vecteurs de Δ, des P_i et des $\boldsymbol{c_i}$;
- on notera, pour $q \in Q$, $\deg_q^{out}$ le nombre de transitions sortantes de l'état q.

◀

Les sections 8.2 et 8.3 sont dédiées au calcul de la série génératrice d'un automate de Parikh non ambigu, en bornant notamment les coefficients de leur équation différentielle. Le résultat principal de ces deux sections est résumé dans la proposition suivante :

Proposition 8.3 (Taille de l'équation différentielle satisfaite par un langage de Parikh faiblement non ambigu).
▶ Soit $\mathcal{A}$ un automate de Parikh faiblement non ambigu. Alors la série $L(x)$ du langage de $\mathcal{A}$ satisfait une équation différentielle linéaire de la forme :

$$q_s(x)\partial_x^s L(x) + \cdots + q_0(x)L(x) = 0\,,$$

avec $s \leqslant (d|\mathcal{A}|\,\|\mathcal{A}\|_\infty)^{O(d)}$, et pour tout $i \in [0, s]$,

$$\deg(q_i) \leqslant (d|\mathcal{A}|\,\|\mathcal{A}\|_\infty)^{O(d^2)}\,,$$
$$\log \|q_i\|_1 \leqslant (d|\mathcal{A}|\,\|\mathcal{A}\|_\infty)^{O(d^2)}\,.$$

◀

Ensuite, la section 8.4 exploite les majorations obtenues précédemment pour borner la taille du mot minimal qui témoigne de la non inclusion des langages $\mathcal{L}(\mathcal{A})$ et $\mathcal{L}(\mathcal{B})$; les deux résultat principaux de cette section sont les suivants :

Théorème 8.4 (Taille du plus petit témoin de non inclusion).
▶ Si $\mathcal{L}(A)$ n'est pas inclus dans $\mathcal{L}(B)$, alors il existe un mot w qui soit dans $\mathcal{L}(\mathcal{A})$ mais pas dans $\mathcal{L}(\mathcal{B})$ tel que :

$$|w| \leqslant 2^{(dM)^{O(d^2)}}\,,$$

avec $d = d_\mathcal{A} + d_\mathcal{B}$, et $M = |\mathcal{A}||\mathcal{B}| \max(\|\mathcal{A}\|_\infty, \|\mathcal{B}\|_\infty)$. ◀

Corollaire 8.5 (Borne de complexité du problème de l'inclusion).
▶ Soient deux automates de Parikh faiblement non ambigus $\mathcal{A}$ and $\mathcal{B}$ de dimensions $d_\mathcal{A}$ et $d_\mathcal{B}$. On peut décider le problème de l'inclusion $L(\mathcal{A}) \subseteq L(\mathcal{B})$ en temps $2^{2^{O(d^2 \log(dM))}}$ où $d = d_\mathcal{A} + d_\mathcal{B}$ et $M = |\mathcal{A}|\,|\mathcal{B}|\,\max(\|\mathcal{A}\|_\infty, \|\mathcal{B}\|_\infty)$. ◀

8.2 Séries génératrices rationnelles associées à un automate de Parikh

8.2.1 Notations, bornes sur les déterminants, règle de Cramer

Dans ce chapitre, nous allons travailler avec des polynômes multivariés à coefficients rationnels, c'est-à-dire des éléments de $\mathbb{Q}[x_1, \ldots, x_n]$. On note $\boldsymbol{\alpha} \in \mathbb{N}^n$ le *multi-indice* $(\alpha_1, \ldots, \alpha_n)$ et $x^{\boldsymbol{\alpha}}$ le monôme $x^{\boldsymbol{\alpha}} = \prod_{i=1}^n x_i^{\alpha_i}$. Le *degré total* de $x^{\boldsymbol{\alpha}}$ est défini par $\sum_{i=1}^n \alpha_i$, mais dans la suite nous allons principalement travailler avec les degrés partiels donnés par le multi-indice, et le *degré maximal* $\deg_m(x^{\boldsymbol{\alpha}}) = \max_{i=1}^n \alpha_i$.

On rappelle qu'un polynôme non nul P de $\mathbb{Q}[x_1, \ldots, x_n]$ est une combinaison linéaire finie de monômes sur $\mathbb{Q}$

$$P = \sum_{\boldsymbol{\alpha} \in \mathrm{Supp}(P)} \lambda_{\boldsymbol{\alpha}} x^{\boldsymbol{\alpha}},$$

où $\mathrm{Supp}(P)$ est un ensemble fini de multi-indices, vérifiant $\lambda_{\boldsymbol{\alpha}} \neq 0$ pour tout $\boldsymbol{\alpha} \in \mathrm{Supp}(P)$. L'ensemble $\mathrm{Supp}(P)$ est appelé le support monomial de P. On note

$$\mathfrak{s}(P) = \mathrm{card}(\mathrm{Supp}(P))\,.$$

Le *degré total* de P, $\deg(P)$, est le degré maximal de ses monômes :

$$\deg(P) = \max\{\deg(x^{\boldsymbol{\alpha}}) : \ \boldsymbol{\alpha} \in \mathrm{Supp}(P)\}\,.$$

Le degré partiel de P par rapport à x_i, noté $\deg_{x_i}(P)$, est la plus grande i-ième composante des tuples de $\mathrm{Supp}(P)$: autrement dit il s'agit du degré de P vu comme un polynôme en x_i.

Le degré maximal d'un polynôme est le maximum des degrés maximaux de ses monômes :

$$\deg_m(P) = \max\{\deg_m(x^{\boldsymbol{\alpha}}) : \ \boldsymbol{\alpha} \in \mathrm{Supp}(P)\} = \max_{i \in [n]}(\deg_{x_i}(P)).$$

On utilisera les normes classiques suivantes sur les polynômes de $\mathbb{Q}[x_1, \ldots, x_n]$:

$$\|P\|_1 = \sum_{\boldsymbol{\alpha} \in \mathrm{Supp}(P)} |\lambda_{\boldsymbol{\alpha}}|, \ \text{ et } \ \|P\|_\infty = \max_{\boldsymbol{\alpha} \in \mathrm{Supp}(P)} |\lambda_{\boldsymbol{\alpha}}|.$$

On obtient facilement la relation suivante :

$$\|P\|_1 \leqslant \mathfrak{s}(P)\|P\|_\infty \leqslant \|P\|_\infty \prod_{i=1}^n (\deg_{x_i}(P)+1) \leqslant (\deg_m(P)+1)^n \|P\|_\infty\,, \quad (8.1)$$

et ces inégalités sont des égalités pour le polynôme $P = \sum_{\boldsymbol{\alpha} \in [d]^n} x^{\boldsymbol{\alpha}}$, car alors $\|P\|_1 = (d+1)^n$.

Lemme 8.6 (Produit).
▶ Si P et Q sont des polynômes de $\mathbb{Q}[x_1, \ldots, x_n]$, alors

$$\|PQ\|_\infty \leqslant \|P\|_\infty \|Q\|_1 \leqslant \|P\|_1 \|Q\|_1 \quad \text{et} \quad \|PQ\|_1 \leqslant \|P\|_1 \|Q\|_1\,. \qquad \blacktriangleleft$$

Démonstration. On note $P = \sum \lambda_{\boldsymbol{\alpha}} x^{\boldsymbol{\alpha}}$ et $Q = \sum \mu_{\boldsymbol{\beta}} x^{\boldsymbol{\beta}}$. Alors $PQ = \sum c_{\boldsymbol{\gamma}} x^{\boldsymbol{\gamma}}$ avec $|c_{\boldsymbol{\gamma}}| = |\sum_{\boldsymbol{\alpha}+\boldsymbol{\beta}=\boldsymbol{\gamma}} \lambda_{\boldsymbol{\alpha}}\mu_{\boldsymbol{\beta}}| \leqslant \|P\|_\infty \sum_{\boldsymbol{\alpha}+\boldsymbol{\beta}=\boldsymbol{\gamma}} |\mu_{\boldsymbol{\beta}}| \leqslant \|P\|_\infty \|Q\|_1$. L'autre inégalité vient du fait que $\|P\|_\infty \leqslant \|P\|_1$.

Remarquons que la première inégalité est atteinte pour $P = Q = \sum_{\boldsymbol{\alpha}\in[d]^n} x^{\boldsymbol{\alpha}}$, car on a alors $\|P^2\|_\infty = (d+1)^n = \|P\|_1$.
Enfin, $PQ = \sum_{\boldsymbol{\alpha}} \sum_{\boldsymbol{\beta}} \lambda_{\boldsymbol{\alpha}}\mu_{\boldsymbol{\beta}} x^{\boldsymbol{\alpha}+\boldsymbol{\beta}}$, d'où $\|PQ\|_1 \leqslant \sum_{\boldsymbol{\alpha}} |\lambda_{\boldsymbol{\alpha}}| \sum_{\boldsymbol{\beta}} |\mu_{\boldsymbol{\beta}}| = \|P\|_1 \|Q\|_1$. ■

Lemme 8.7 (Borne polynomiale simple).
▶ Soit A une matrice $p \times p$ dont les coefficients sont des polynômes de $\mathbb{Q}[x_1, \ldots, x_n]$. Alors

$$\|\det(A)\|_1 \leqslant \prod_{i=1}^{p} \sum_{j=1}^{p} \|A_{i,j}\|_1 .$$

◀

Démonstration. Par définition du déterminant :

La dernière égalité s'obtient par distributivité.

$$\|\det(A)\|_1 \leqslant \sum_{\sigma\in\mathfrak{S}_p} \prod_{i=1}^{p} \|A_{i,\sigma(i)}\|_1 \leqslant \sum_{\sigma:[1,p]\to[1,p]} \prod_{i=1}^{p} \|A_{i,\sigma(i)}\|_1 = \prod_{i=1}^{p} \sum_{j=1}^{p} \|A_{i,j}\|_1.$$

■

Remarque 8.8 (Borne d'Hadamard).
▶ Nous pouvons prouver une borne plus fine sur les déterminants, à partir des bornes d'Hadamard, en utilisant la norme 2 du déterminant :

$$\|\det(A)\|_2 \leqslant \prod_{i=1}^{p} \sqrt{\sum_{j=1}^{p} \|A_{i,j}\|_1^2} .$$

Cette formule se démontre en adaptant la preuve de [GG74], qui porte sur $\mathbb{C}[x]$, aux polynômes multivariés ; une borne similaire apparaît dans [EP05, Lemma 3.1] et, dans un contexte différent, dans [Bro71, Eq. (21)] (sans preuve). Cependant, la norme 1 intervient plus naturellement dans l'analyse du problème, et nous ne sommes pas arrivés à tirer pleinement parti de cette borne plus fine : nous finissions par borner la somme des carrés par le carré de la somme, ce qui revient à utiliser la borne du Lemme 8.7. ◀

Un *mineur* d'ordre m d'une matrice $m \times n$ avec $m < n$ est un sous-déterminant de la matrice, obtenu en sélectionnant m colonnes de A. La proposition suivante permet d'exprimer les solutions d'un système linéaire à l'aide de déterminants :

Proposition 8.9 (Formules de Cramer).
▶ Soit $\mathbb{K}$ un corps quelconque.

a. Soit A une matrice $n \times n$ inversible à coefficients dans $\mathbb{K}$, $\boldsymbol{b}$ et $\boldsymbol{v}$ deux vecteurs de $\mathbb{K}^n$ tels que $A\boldsymbol{v} = \boldsymbol{b}$. Alors pour tout $i \in [1, n]$, $v_i = \frac{\det(A_i)}{\det(A)}$ où A_i désigne la matrice obtenue à partir de A en remplaçant sa i-ième colonne par le vecteur $\boldsymbol{b}$.

b. Soit A une matrice $m \times n$ à coefficients dans $\mathbb{K}$ où $m < n$. Il existe un vecteur $\boldsymbol{v} \in \mathbb{K}^m \setminus \{\mathbf{0}\}$ dont les coordonnées sont des mineurs ou des opposés de mineurs de A d'ordre m, tel que $A\boldsymbol{v} = \mathbf{0}$.

◀

Démonstration. Voir [AADM17] et [Kau14]. ■

8.2.2 Séries génératrices des calculs de l'automate et du semilinéaire

Définition 8.10 (Séries génératrices associées à l'automate).

▶ On définit dans un premier temps deux séries associées à un automate de Parikh :

— La série (multivariée) des calculs de $\mathcal{A}$, notée $A(x, y_1, \dots, y_d)$, est définie par :

$$A(x, y_1, \dots, y_d) := \sum_{n\in\mathbb{N}, \boldsymbol{v}\in\mathbb{N}^d} a(n, \boldsymbol{v}) x^n y_1^{v_1} \cdots y_d^{v_d},$$

où $a(n, \boldsymbol{v})$ compte le nombre de calculs de $\mathcal{A}$ de longueur n, partant de l'état initial, finissant dans un état final, et étiquetés par le vecteur $\boldsymbol{v}$. Comme on suppose qu'aucune transition n'est étiquetée par ε, $a(n, \boldsymbol{v})$ est bien fini pour tout $n \in \mathbb{N}$ et $\boldsymbol{v} \in \mathbb{N}^d$.

— La série génératrice du semilinéaire C est définie par

$$C(y_1, \dots, y_d) = \sum_{\boldsymbol{v}\in C} y_1^{v_1} \cdots y_d^{v_d}.$$

◀

Remarque 8.11 (Notation condensée).

▶ Nous utiliserons les notations $\boldsymbol{y} := (y_1, \dots, y_d)$ et $\boldsymbol{y}^{\boldsymbol{v}} := y_1^{v_1} \cdots y_d^{v_d}$. ◀

Lemme 8.12 (Série génératrice des calculs).

▶ La série des calculs $A(x, y_1, \dots, y_d)$ est égale à une fraction $\frac{R}{S}$, où R et S sont deux polynômes de $\mathbb{Z}[x, y_1, \dots, y_d]$ vérifiant :

$$\begin{array}{cc} \deg_x(R) \leqslant |Q_{\mathcal{A}}| - 1 & \deg_x(S) \leqslant |Q_{\mathcal{A}}| \\ \forall i \in [d],\ \deg_{y_i}(R) \leqslant (|Q_{\mathcal{A}}| - 1)\|\mathcal{A}\|_{i,\infty} & \deg_{y_i}(S) \leqslant |Q_{\mathcal{A}}|\|\mathcal{A}\|_{i,\infty} \\ \|R\|_\infty \leqslant \|R\|_1 \leqslant |F| \prod_{q\in Q_{\mathcal{A}}} (1 + \deg_q^{out}) & \|S\|_\infty \leqslant \|S\|_1 \leqslant \prod_{q\in Q_{\mathcal{A}}} (1 + \deg_q^{out}) \end{array}$$

◀

Démonstration. Pour tout état $q \in Q_{\mathcal{A}}$, on considère la série

$$q(x, y_1, \dots, y_d) = \sum_{n\in\mathbb{N}, \boldsymbol{v}\in\mathbb{N}^d} a_q(n, \boldsymbol{v}) x^n y_1^{v_1} \dots y_d^{v_d},$$

où $a_q(n, \boldsymbol{v})$ décompte le nombre de calculs de longueur n depuis l'état q jusqu'à un état final, étiquetés par le vecteur $\boldsymbol{v}$. En particulier, la série génératrice $A(x, y_1, \dots, y_d)$ des calculs de $\mathcal{A}$ est égale à $q_I(x, y_1, \dots, y_d)$.

Pour tout état q, nous avons classiquement l'égalité :

$$q(x, y_1, \dots, y_d) = [\![q \in F]\!] + \sum_{(q,(a,\boldsymbol{v}),q')\in\Delta} x y_1^{v_1} \cdots y_d^{v_d} \cdot q'(x, y_1, \dots, y_d),$$

où $[\![P]\!] = 1$ si la propriété P est vraie et $[\![P]\!] = 0$ sinon.

Par conséquent, le vecteur de séries génératrices $\boldsymbol{q} = (q(x, y_1, \dots, y_d))_{q\in Q_{\mathcal{A}}}$ vérifie l'équation :

$$\boldsymbol{q} = xM\boldsymbol{q} + \boldsymbol{f},$$

où $\boldsymbol{f} = (\llbracket q \in F \rrbracket)_{q \in Q_\mathcal{A}}$, et M est une matrice $|Q_\mathcal{A}| \times |Q_\mathcal{A}|$ de polynômes, définie par $M_{q,q'}(y_1, \ldots, y_d) = \sum_{(q,(a,\boldsymbol{v}),q') \in \Delta} y_1^{\boldsymbol{v}_1} \cdots y_d^{\boldsymbol{v}_d}$ pour tout q et q' dans $Q_\mathcal{A}$. En particulier, pour tout $q \in Q_\mathcal{A}$,

$$\sum_{q' \in Q_\mathcal{A}} \|M_{q,q'}\|_1 = d_q^{out}.$$

Le vecteur $\boldsymbol{q}$ satisfait l'équation $(\mathtt{Id} - xM)\boldsymbol{q} = \boldsymbol{f}$. La matrice $\mathtt{Id} - xM$ est inversible dans le corps des fractions rationnelles $\mathbb{Q}(x, y_1, \ldots, y_d)$. En effet, $\det(\mathtt{Id} - M)(x, y_1, \ldots, y_d)$ est un polynôme non nul car $\det(\mathtt{Id} - xM)(0, \ldots, 0) = \det(I) = 1$. Par les formules de Cramer (Proposition 8.9) :

$$A(x, y_1, \ldots, y_d) = q_I(x, y_1, \ldots, y_d) = \frac{R(x, y_1, \ldots, y_d)}{S(x, y_1, \ldots, y_d)},$$

avec $R(x, y_1, \ldots, y_d) = \det(M')$, où M' est la matrice obtenue en remplaçant la colonne numéro q_I de $\mathtt{Id} - xM$ par le vecteur $\boldsymbol{f}$, et $S(x, y_1, \ldots, y_d) = \det(\mathtt{Id} - xM)$.

Remarquons alors que pour $q \in Q_\mathcal{A}$:

$$\sum_{q' \in Q_\mathcal{A}} \|(\mathtt{Id} - xM)_{q,q'}\|_1 = 1 + \deg_q^{out}$$

avec le 1 qui vient de l'identité. Ainsi en utilisant la borne du Lemme 8.7 à la matrice $\mathtt{Id} - xM$, on obtient

$$\|S\|_\infty \leqslant \|S\|_1 \leqslant \prod_{q \in Q_\mathcal{A}} (1 + \deg_q^{out}).$$

Par définition du déterminant, chaque terme de S est une combinaison linéaire de produits de $|Q_\mathcal{A}|$ polynômes, de degré maximal 1 en x, et de degré maximal $\|A\|_{i,\infty}$ en y_i, donc $\deg_x(S) \leqslant |Q_\mathcal{A}|$ et $\deg_{y_i}(S) \leqslant |Q_\mathcal{A}| \|\mathcal{A}\|_{i,\infty}$.

En développant le déterminant de M' selon la première colonne $\boldsymbol{f}$, et en appliquant la borne du Lemme 8.7 aux sous-déterminants, nous obtenons la formule suivante :

$$\|R\|_1 \leqslant \sum_{q_f \in F} \prod_{q \neq q_f} \sum_{q' \neq q_I} \|(\mathtt{Id} - xM)_{q,q'}\|_1 .$$

Trivialement, $\sum_{q' \neq q_I} \|(\mathtt{Id} - xM)_{q,q'}\|_1 \leqslant \sum_{q' \in Q_\mathcal{A}} \|(\mathtt{Id} - xM)_{q,q'}\|_1 = 1 + \deg_q^{out}$.

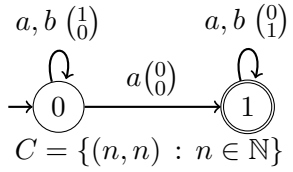
Et $\prod_{q \neq q_f} (1 + \deg_q^{out}) \leqslant \prod_{q \in Q_\mathcal{A}} (1 + \deg_q^{out})$, car $(1 + \deg_{q_f}^{out}) \geqslant 1$. Ainsi :

$$\|R\|_\infty \leqslant \|R\|_1 \leqslant |F| \prod_{q \in Q_\mathcal{A}} (1 + \deg_q^{out}).$$

■

Exemple 8.13 .

▶ Tout au long de cette section et de la suivante, nous illustrons les résultats des différentes propositions sur l'automate $\mathcal{A}_{\text{ex}}$ de l'introduction. Nous rappelons que $\mathcal{A}_{\text{ex}}$ est l'automate suivant, qui reconnaît les mots de longueur impaire qui ont un a au milieu :



Réécrivons le semilinéaire sous la forme voulue $C = \mathbf{0} + \{(1,1)\}^*$. Dans ces conditions :

- $d = 2$
- $|\mathcal{A}_{\text{ex}}| = 2 + 3 + 1 + 1 = 7$;
- $\|\mathcal{A}_{\text{ex}}\|_{1,\infty} = \|\mathcal{A}_{\text{ex}}\|_{2,\infty} = \|\mathcal{A}_{\text{ex}}\|_{\infty} = 1$;
- $\deg_{q_0}^{out} = 3$ et $\deg_{q_1}^{out} = 2$. ◀

Nous obtenons alors le système suivant pour les séries génératrices des calculs :

$$\begin{cases} q_0(x, y_1, y_2) = 2xy_1 q_0(x, y_1, y_2) + xq_1(x, y_1, y_2) \\ q_1(x, y_1, y_2) = 2xy_2 q_1(x, y_1, y_2) + 1 \end{cases}$$

qui s'écrit sous la forme matricielle :

$$\begin{pmatrix} 1 - 2xy_1 & -x \\ 0 & 1 - 2xy_2 \end{pmatrix} \cdot \begin{pmatrix} q_0(x, y_1, y_2) \\ q_1(x, y_1, y_2) \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

et avec les formules de Cramer nous avons $A(x, y_1, y_2) = \frac{R}{S}$ avec

$$R = \begin{vmatrix} 0 & -x \\ 1 & 1 - 2xy_2 \end{vmatrix} = x \quad \text{et} \quad S = \begin{vmatrix} 1 - 2xy_1 & -x \\ 0 & 1 - 2xy_2 \end{vmatrix} = (1-2xy_1)(1-2xy_2)\,.$$

Et nous vérifons bien les inégalités :

$$\begin{aligned} \deg_x(R) = 1 &\leqslant |Q_{\mathcal{A}_{\text{ex}}}| - 1 = 1 & \deg_x(S) = 2 &\leqslant |Q_{\mathcal{A}_{\text{ex}}}| = 2 \\ \deg_{y_1}(R) = 0 &\leqslant (|Q_{\mathcal{A}_{\text{ex}}}| - 1)\|\mathcal{A}_{\text{ex}}\|_{1,\infty} = 1 & \deg_{y_1}(S) = 1 &\leqslant |Q_{\mathcal{A}_{\text{ex}}}|\|\mathcal{A}_{\text{ex}}\|_{1,\infty} = 2 \\ \deg_{y_2}(R) = 0 &\leqslant (|Q_{\mathcal{A}_{\text{ex}}}| - 1)\|\mathcal{A}_{\text{ex}}\|_{2,\infty} = 1 & \deg_{y_2}(S) = 1 &\leqslant |Q_{\mathcal{A}_{\text{ex}}}|\|\mathcal{A}_{\text{ex}}\|_{2,\infty} = 2 \\ \|R\|_1 = 1 &\leqslant |F| \prod_{q \in Q_{\mathcal{A}_{\text{ex}}}} (1 + \deg_q^{out}) = 12 & \|S\|_1 = 9 &\leqslant \prod_{q \in Q_{\mathcal{A}_{\text{ex}}}} (1 + \deg_q^{out}) = 12 \end{aligned}$$

Remarque 8.14 .

▶ En utilisant la définition du déterminant $\det(\mathtt{Id} - xM)$ comme une somme portant sur toutes les permutations s de $Q_{\mathcal{A}}$, nous pouvons interpréter la décomposition de s en produits de cycles disjoints en termes de cycles dans l'automate, pour borner plus finement le degré de R et de S. Soit $i \in [d]$, pour tout cycle σ élémentaire de $\mathcal{A}$, notons, dans cette remarque uniquement, $|\sigma|_i$ la somme des i-ièmes composantes des vecteurs des transitions empruntées dans le cycle σ.

En notant Ω l'ensemble des cycles élémentaires de $\mathcal{A}$ et $\mathcal{P}_{\neq}(\Omega)$ l'ensemble des parties de Ω composées de cycles qui ne passent par aucun état en commun, alors $\deg_{y_i}(S) \leqslant \max_{X \in \mathcal{P}_{\neq}(\Omega)} \sum_{\sigma \in X} |\sigma|_i$.

Cette interprétation du déterminant avec des cycles dans un graphe est un cas particulier de la formule plus générale de Lindström-Gessel-Viennot [GV89]. Nous n'aurons pas besoin d'être aussi précis cependant. Cette borne implique celle que

nous avons trouvée plus haut : pour tout cycle élémentaire σ, $|\sigma|_i \leqslant \|\mathcal{A}\|_{i,\infty}|\sigma|$ où $|\sigma|$ est le nombre d'états par lequel passe le cycle élémentaire. Comme les cycles sont élémentaires et ne passent par aucun état en commun, $\sum_{\sigma\in X}|\sigma|_i \leqslant |Q_{\mathcal{A}}|\|\mathcal{A}\|_{i,\infty}$. ◀

Lemme 8.15 (Série génératrice du semilinéaire).
▶ On rappelle que le semilinéaire est donné sous une forme non ambiguë $C = \biguplus_{i=1}^{p} \boldsymbol{c_i} + P_i^*$. Alors la série génératrice de C s'écrit sous la forme d'une fraction $C(\boldsymbol{y}) = \frac{P_C}{Q_C}$ avec :

$$\forall i \in [d],\ \deg_{y_i}(P_C) \leqslant (1+\sum_{j=1}^{p}|P_j|)\|\mathcal{A}\|_{i,\infty} \qquad \deg_{y_i}(Q_C) \leqslant (\sum_{j=1}^{p}|P_j|)\|\mathcal{A}\|_{i,\infty}$$
$$\|P_C\|_1 \leqslant p2^{\sum_{i=1}^{p}|P_i|} \qquad \|Q_C\|_1 \leqslant 2^{\sum_{i=1}^{p}|P_i|}$$

◀

Démonstration. Par non-ambiguïté, $C(\boldsymbol{y}) = \sum_{i=1}^{p} \frac{\boldsymbol{y}^{\boldsymbol{c_i}}}{\prod_{\boldsymbol{v}\in P_i}(1-\boldsymbol{y}^{\boldsymbol{v}})}$. En mettant au même dénominateur, on obtient $C(\boldsymbol{y}) = \frac{P_C}{Q_C}$ avec :

$$P_C = \sum_{i=1}^{p} \boldsymbol{y}^{\boldsymbol{c_i}} \prod_{j\neq i}\prod_{\boldsymbol{v}\in P_j}(1-\boldsymbol{y}^{\boldsymbol{v}}) \quad \text{et} \quad Q_C = \prod_{i=1}^{p}\prod_{\boldsymbol{v}\in P_i}(1-\boldsymbol{y}^{\boldsymbol{v}}).$$

Ainsi $\deg_{y_i}(P_C) \leqslant (1+\sum_{j=1}^{p}|P_j|)\|\mathcal{A}\|_{i,\infty}$, et $\deg_{y_i}(Q_C) \leqslant (\sum_{j=1}^{p}|P_j|)\|\mathcal{A}\|_{i,\infty}$ pour tout $i \in [d]$.

De plus par non-ambiguïté, $\|\prod_{\boldsymbol{v}\in P_i}(1-\boldsymbol{y}^{\boldsymbol{v}})\|_\infty = 1$, et $\|\prod_{\boldsymbol{v}\in P_i}(1-\boldsymbol{y}^{\boldsymbol{v}})\|_1 = \mathfrak{s}\big(\prod_{\boldsymbol{v}\in P_i}(1-\boldsymbol{y}^{\boldsymbol{v}})\big) = 2^{|P_i|}$.

Ainsi $\|Q_C\|_1 \leqslant 2^{\sum_{i=1}^{p}|P_i|}$, et $\|P_C\|_1 \leqslant p2^{\sum_{i=1}^{p}|P_i|}$. ■

8.3 Série génératrice d'un automate de Parikh faiblement non ambigu

8.3.1 Préparation de la série génératrice à l'algorithme de Lipshitz

Un des objectifs de ce chapitre est de fournir des bornes sur les degrés et les normes des coefficients des polynômes qui apparaissent dans l'équation différentielle satisfaite par la série génératrice du langage de $\mathcal{A}$. Pour cela nous décrivons rapidement dans un premier temps l'approche de Lipshitz [Lip88], qui consiste à introduire la fonction :

$$F(x,\boldsymbol{y},\boldsymbol{u}) = \frac{1}{u_1\cdots u_d}A\left(x,u_1,\ldots,u_d\right)\cdot C\left(\frac{y_1}{u_1},\ldots,\frac{y_d}{u_d}\right).$$

Si cette opération s'interprète bien du côté des fractions rationnelles, il faut vérifier qu'elle ait bien un sens dans celui des séries, car nous quittons l'anneau des séries formelles $\mathbb{Q}[x,\boldsymbol{y},\boldsymbol{u}]$: la substitution $y_1 \leftarrow \frac{y_1}{u_1}$ dans la série de C introduit une infinité de monômes à exposants négatifs. Lipshitz se place en fait dans l'ensemble $\mathcal{M}$ des

sommes infinies formelles de la forme :

$$G = \sum_{\substack{n\in\mathbb{N},\boldsymbol{i}\in\mathbb{N}^d,\boldsymbol{j}\in\mathbb{Z}^d \\ i_1+j_1\geqslant -k \\ \cdots \\ i_d+j_d\geqslant -k}} a(n,\boldsymbol{i},\boldsymbol{j}) x^n \boldsymbol{y}^{\boldsymbol{i}} \boldsymbol{u}^{\boldsymbol{j}}$$

où k est une constante qui dépend de G. L'algorithme de Lipshitz repose sur les propriétés suivantes de l'ensemble $\mathcal{M}$:

- $\mathcal{M}$ est un $\mathbb{Q}[x,\boldsymbol{y},\boldsymbol{u}]$-module (au sens où pour tous $G, G' \in \mathcal{M}$, et $p \in \mathbb{Q}[x,\boldsymbol{y},\boldsymbol{u}]$, $G+G' \in \mathcal{M}$ et $pG \in \mathcal{M}$)
- $\mathcal{M}$ est stable par dérivation (si $t \in \{x, y_1, \dots, y_d, u_1, \dots, u_d\}$ et $G \in \mathcal{M}$, alors $\partial_t G \in M$, où $\partial_t G$ désigne la dérivation formelle terme à terme de G).
- enfin, si $p \in \mathbb{Q}[x,\boldsymbol{y},\boldsymbol{u}]$ et $G \in \mathcal{M}$ sont tels que $p \neq 0$ et $pG = 0$, alors $G = 0$.

L'étude de $F(x,\boldsymbol{y},\boldsymbol{u})$ est alors justifiée par l'égalité suivante :

$$A(x,y_1,\dots,y_d) \odot \frac{1}{1-x} C(y_1,\dots,y_d) = [u_1^{-1}\cdots u_d^{-1}]\, F(x,\boldsymbol{y},\boldsymbol{u})\,,$$

l'extraction de coefficients ayant bien un sens par la définition de $\mathcal{M}$. Multiplier par $\frac{1}{y_1\dots y_d}$ dans la définition de F, pour ensuite extraire $[(y_1\dots y_d)^{-1}]F$, peut sembler inutile au premier abord. En fait, l'argument de Lipshitz repose fortement sur le fait que le coefficient extrait est bien $[(u_1\dots u_d)^{-1}]F$: comme $(u_1\dots u_d)^{-1}$ ne peut pas s'obtenir à partir de la dérivation formelle d'un terme de la forme $u_1^{\pm i_1}\dots u_d^{\pm i_d}$, cela permettra d'intervertir dérivation et extraction de coefficient dans une équation différentielle de F bien choisie, pour la transformer en une équation pour $[(u_1\dots u_d)^{-1}]F$.

L'algorithme de Lipshitz permet ainsi de trouver une équation différentielle partielle en x satisfaite par $A(x,y_1,\dots,y_d) \odot \frac{1}{1-x} C(y_1,\dots,y_d)$. Nous en déduisons ensuite en spécialisant $y_i = 1$ une équation différentielle pour $L(x)$.

C'est ce que nous avons fait dans les annexes de notre article publié à ICALP en 2020. Cette approche a le gros inconvénient d'introduire des variables auxiliaires $u_1,\dots,u_d$, et d'ainsi doubler le nombre de variables par rapport à la dimension de l'automate de départ. Cette augmentation de la dimension vouait à l'échec toute tentative d'appliquer l'algorithme de Lipshitz sur des exemples, même élémentaires. Cette méthode implique de calculer l'équation du produit d'Hadamard de deux séries en $x, y_1, \dots, y_d$, pour ensuite remplacer toutes les variables y_i par 1. Nous calculons donc un objet plus complexe, la série multivariée, pour ensuite "jeter" les variables.

Il nous faudra donc faire les mêmes vérifications que Lipshitz, en définissant un module analogue à $\mathcal{M}$ adapté aux séries que nous rencontrerons.

Dans ce chapitre, j'opte donc pour une méthode légèrement différente de celle de notre article, en remplaçant toutes les variables y_i par 1 *avant* d'appliquer l'algorithme de Lipshitz. Nous nous sommes donc intéressés à la somme infinie suivante :

$$\frac{1}{y_1\cdots y_d} \sum_{n\in\mathbb{N},\boldsymbol{v}\in\mathbb{N}^d,\boldsymbol{v'}\in C} a(n,\boldsymbol{v}) x^n \boldsymbol{y}^{\boldsymbol{v}-\boldsymbol{v'}}$$

$$\text{" = "} \frac{1}{y_1\cdots y_d} A\left(x,y_1,\dots,y_d\right)\cdot C\left(\frac{1}{y_1},\dots,\frac{1}{y_d}\right)$$

Il faut cependant faire attention, car cette somme infinie n'est pas automatiquement bien définie. Nous le voyons sur un exemple simple : si $A(x,y) = x\sum_{n\in\mathbb{N}} y^n$, et $C(y) = \sum_{n\in\mathbb{N}} y^n$, alors $A(x,y)C(1/y) = "x\sum_{n,m\in\mathbb{N}} y^{n-m}"$ n'est pas bien définie

en tant que somme infinie, même si la fraction rationnelle $\frac{x}{1-y}\frac{1}{1-1/y}$ existe. Le problème s'observe aussi en se plongeant dans les séries entières : si les rayons de convergence de $A(x,y)$ et $C(x,y)$ sont 1 par exemple, la série de $A(x,y)$ converge pour $|y|<1$, mais celle de $C(x,1/y)$ converge pour $|y|>1$.

Cependant, dans notre cas particulier, nous allons voir que la somme est bien définie, car $A(x,\boldsymbol{y})$ a une forme particulière. Dans un premier temps nous introduisons un ensemble $\mathcal{M}'$ analogue à celui de Lipshitz, adapté à notre problème :

Proposition 8.16 .
▶ L'ensemble $\mathcal{M}'$ des sommes formelles de la forme

$$G = \sum_{\substack{n\in\mathbb{N},\boldsymbol{v}\in\mathbb{Z}^d \\ v_1\leqslant nk \\ \cdots \\ v_d\leqslant nk}} a(n,\boldsymbol{v})x^n\boldsymbol{y}^{\boldsymbol{v}},$$

où k est une constante qui dépend de G, vérifie toutes les propriétés nécessaires à l'algorithme de Lipshitz :

— $\mathcal{M}'$ est un $\mathbb{Q}[x,\boldsymbol{y}]$-module

— $\mathcal{M}'$ est stable par dérivation

— enfin, si $p\in\mathbb{Q}[x,\boldsymbol{y}]$ est non nul, $G\in\mathcal{M}'$, et de plus $pG=0$, alors $G=0$.

◀

Démonstration. Seul le dernier point n'est pas trivial. Il se démontre par une adaptation de la preuve pour le module $\mathcal{M}$ de Lipshitz [Lip88] : si $p(x,\boldsymbol{y})G(x,\boldsymbol{y})=0$ avec $p(x,\boldsymbol{y})\neq 0$ et $G(x,\boldsymbol{y})\in\mathcal{M}'$, notons k la constante associée à G, et $r := \max(k,\deg_m(p))$. Alors $G(xy_1^r\dots y_d^r, y_1^{-1},\dots,y_d^{-1})\in\mathbb{Q}[[x,y_1,\dots,y_d]]$ est une série formelle, et $p(xy_1^r\dots y_d^r, y_1^{-1},\dots,y_d^{-1})$ est toujours un polynôme non nul. Ainsi $p(xy_1^r\dots y_d^r, y_1^{-1},\dots,y_d^{-1})G(xy_1^r\dots y_d^r, y_1^{-1},\dots,y_d^{-1})=0$, et comme l'anneau des séries formelles est intègre, $G(xy_1^r\dots y_d^r, y_1^{-1},\dots,y_d^{-1})=0$. En remarquant que la substitution qui remplace $(x,\boldsymbol{y})$ par $(xy_1^r\dots y_d^r, y_1^{-1},\dots,y_d^{-1})$ ne crée pas de collision, au sens où les deux séries ont exactement le même ensemble de coefficients, nous en déduisons que $G=0$. ■

La proposition suivante démontre que F est bien définie :

Proposition 8.17 (Définition de F).
▶ La somme infinie suivante :

$$F(x,\boldsymbol{y}) := \frac{1}{y_1\cdots y_d}\sum_{n\in\mathbb{N},\boldsymbol{v}\in\mathbb{N}^d,\boldsymbol{v'}\in C} a(n,\boldsymbol{v})x^n\boldsymbol{y}^{\boldsymbol{v}-\boldsymbol{v'}}$$

est bien définie et appartient à $\mathcal{M}'$. ◀

Démonstration. Rappelons que l'automate $\mathcal{A}$ ne possède pas d'ε-transitions, et qu'ainsi la longueur d'un calcul de $\mathcal{A}$ coïncide avec la longueur du mot qui l'étiquette. Comme tout vecteur $\boldsymbol{v}$ qui étiquette un calcul de longueur n vérifie $\|\boldsymbol{v}\|_\infty\leqslant n\|\mathcal{A}\|_\infty$, pour tout $n\in\mathbb{N}$, $[x^n]A(x,\boldsymbol{y}) = \sum_{\|\boldsymbol{v}\|_\infty\leqslant n\|\mathcal{A}\|_\infty} a(n,\boldsymbol{v})\boldsymbol{y}^{\boldsymbol{v}}$ est donc un polynôme, dont le degré maximal est borné par $n\|\mathcal{A}\|_\infty$.

Ainsi la somme infinie $\sum_{n\in\mathbb{N},\boldsymbol{v}\in\mathbb{N}^d,\boldsymbol{v'}\in C} a(n,\boldsymbol{v})x^n\boldsymbol{y}^{\boldsymbol{v}-\boldsymbol{v'}}$ est bien définie, et F appartient à l'ensemble $\mathcal{M}'$. ■

Remarque 8.18 .
▶ Avec un peu de recul, nous aurions sans doute dû tourner les choses un tout petit peu différemment. Si je suis parti de l'expression $A(x, \boldsymbol{y})C(\boldsymbol{y}^{-\mathbf{1}})$, c'est parce qu'elle s'interprète bien en imaginant un automate qui calcule d'abord un vecteur positif en incrémentant des compteurs, puis les décrémente avec une machine qui calcule le semilinéaire. D'un point de vue séries formelles cependant, il est plus naturel d'inverser l'endroit où nous inversons les exposants, en considérant plutôt :

$$F(x, \boldsymbol{y}) = \frac{1}{y_1 \cdots y_d} A\left(x, \frac{1}{y_1}, \ldots, \frac{1}{y_d}\right) \cdot C\left(y_1, \ldots, y_d\right).$$

Avec cette expression, pour tout $n \in \mathbb{N}$, $[x^n]F(x, \boldsymbol{y})$ n'a qu'un nombre fini de termes qui ont un exposant négatif, autrement dit $[x^n]F(x, \boldsymbol{y})$ est une série de Laurent formelle. Les deux approches sont analogues, mais celle de cette remarque a l'avantage d'introduire des classes de séries formelles plus standard.

Si les propositions qui suivent sont en fait valides pour les deux définitions de F, je n'ai pas changé la définition de F faute de temps, car il aurait fallu refaire tous les exemples ; ce n'était pas prioritaire dans le processus de rédaction de la thèse. ◀

Nous vérifions alors facilement que formellement, en notant m un entier plus grand que $\deg_m(P_C)$ et $\deg_m(Q_C)$, nous avons l'égalité :

$$S(x, \boldsymbol{y})(y_1 \ldots y_d)^{m+1} Q_C(\tfrac{1}{y_1}, \ldots, \tfrac{1}{y_d}) \cdot F(x, \boldsymbol{y}) = R(x, \boldsymbol{y})(y_1 \ldots y_d)^m P_C(\tfrac{1}{y_1}, \ldots, \tfrac{1}{y_d}).$$

Autrement dit :

$$F(x, \boldsymbol{y}) = \frac{1}{y_1 \cdots y_d} A\left(x, y_1, \ldots, y_d\right) \cdot C\left(\frac{1}{y_1}, \ldots, \frac{1}{y_d}\right)$$

où le membre droit est interprété comme une fraction rationnelle P/Q avec $P, Q \in \mathbb{Z}[x, \boldsymbol{y}]$. Enfin, une extraction de coefficient de $F(x, \boldsymbol{y})$ fait apparaître $L(x)$:

$$\begin{aligned}[y_1^{-1} \cdots y_d^{-1}]F(x, \boldsymbol{y}) &= [y_1^{-1} \cdots y_d^{-1}] \frac{1}{y_1 \cdots y_d} \sum_{n \in \mathbb{N}, \boldsymbol{v} \in \mathbb{N}^d, \boldsymbol{v'} \in C} a(n, \boldsymbol{v}) x^n \boldsymbol{y}^{\boldsymbol{v} - \boldsymbol{v'}} \\ &= \sum_{n \in \mathbb{N}, \boldsymbol{v} \in C} a(n, \boldsymbol{v}) x^n = L(x)\end{aligned} \tag{8.2}$$

où nous rappelons que la série génératrice des calculs acceptants de $\mathcal{A}$ vaut $L(x)$ par faible non ambiguïté.

Tout le terrain théorique préparatoire à la mise en place de l'algorithme de Lipshitz est donc prêt, et nous pouvons désormais dériver les premières propriétés de la fraction rationnelle $F(x, \boldsymbol{y})$:

Lemme 8.19 (Propriétés de la fraction rationnelle).
▶ La fraction rationnelle $F(x, \boldsymbol{y})$ peut s'écrire sous la forme $\frac{P}{Q}$, avec P et Q des polynômes de $\mathbb{Z}[x, \boldsymbol{y}]$ qui vérifient :

$$\begin{array}{cc} \deg_x(P) \leqslant |Q_{\mathcal{A}}| - 1 & \deg_x(Q) \leqslant |Q_{\mathcal{A}}| \\ \forall i \in [d],\ \deg_{y_i}(P) < |\mathcal{A}|\|\mathcal{A}\|_{i,\infty} & \deg_{y_i}(Q) < |\mathcal{A}|\|\mathcal{A}\|_{i,\infty} \\ \|P\|_1 \leqslant 2^{|\mathcal{A}|} & \|Q\|_1 \leqslant 2^{|\mathcal{A}|} \end{array}$$

◀

Démonstration. Nous utilisons les bornes des Lemmes 8.12 et 8.15.

Posons $m_i = \max(1, \sum_{j=1}^{p} |P_j|)\|\mathcal{A}\|_{i,\infty}$ pour tout $i \in [d]$. Alors $\frac{1}{y_1 \dots y_d} \frac{\boldsymbol{y}^{\boldsymbol{m}} P_C(\boldsymbol{y}^{-\mathbf{1}})}{\boldsymbol{y}^{\boldsymbol{m}} Q_C(\boldsymbol{y}^{-\mathbf{1}})}$ s'écrit sous la forme $\frac{\tilde{P}_C}{\tilde{Q}_C}$, avec $\tilde{P}_C$ et $\tilde{Q}_C$ deux polynômes de $\mathbb{Z}[x, \boldsymbol{y}]$ qui vérifient $\|\tilde{P}_C\|_1 = \|P_C\|_1$ et $\|\tilde{Q}_C\|_1 = \|Q_C\|_1$. De plus $\deg_{y_i}(\tilde{P}_C) \leqslant m_i$ et $\deg_{y_i}(\tilde{Q}_C) \leqslant m_i + 1$, pour tout $i \in [d]$.

Ainsi $P = R\tilde{P}_C$ et $Q = S\tilde{Q}_C$, si bien que :

$$\begin{aligned}
\log(\|P\|_1) &\leqslant \log(\|R\|_1 \|\tilde{P}_C\|_1) \\
&\leqslant \log(|F|) + \sum_{q \in Q_\mathcal{A}} \log(1 + \deg_q^{out}) + \log(p) + \sum_{i=1}^{p} |P_i| \\
&\leqslant \log(|F|) + \sum_{q \in Q_\mathcal{A}} \deg_q^{out} + \log(p) + \sum_{i=1}^{p} |P_i| \\
&\leqslant \log(|F|) + |\Delta| + \log(p) + \sum_{i=1}^{p} |P_i| \leqslant |\mathcal{A}|
\end{aligned}$$

Nous obtenons la même majoration pour Q : $\|Q\|_1 \leqslant 2^{|\mathcal{A}|}$.

En ce qui concerne la variable x, $\deg_x(P) = \deg_x(R)$ et $\deg_x(Q) = \deg_x(S)$.

Enfin, $\deg_{y_i}(P) \leqslant \deg_{y_i}(R) + m_i \leqslant (|Q_\mathcal{A}| - 1 + 1 + \sum_{j=1}^{p} |P_j|)\|\mathcal{A}\|_{i,\infty} < |\mathcal{A}|\|\mathcal{A}\|_{i,\infty}$.

Nous supposons que $\|\mathcal{A}\|_{i,\infty} \neq 0$, sinon cette coordonnée ne sert à rien.

De même, $\deg_{y_i}(Q) \leqslant \deg_{y_i}(S) + m_i + 1 \leqslant (|Q_\mathcal{A}| + 1 + \sum_{j=1}^{p} |P_j|)\|\mathcal{A}\|_{i,\infty} + 1 < |\mathcal{A}|\|\mathcal{A}\|_{i,\infty}$. ■

Exemple 8.20.

▶ Si nous reprenons l'exemple de l'automate $\mathcal{A}_{\text{ex}}$, nous avons

$$A(x, y_1, y_2) = \frac{x}{(1 - 2xy_1)(1 - 2xy_2)} \quad \text{et} \quad C(y_1, y_2) = \frac{1}{1 - y_1 y_2},$$

si bien que :

$$\begin{aligned}
F(x, y_1, y_2) &= \frac{1}{y_1 y_2} \frac{x}{(1 - 2xy_1)(1 - 2xy_2)} \frac{1}{1 - y_1^{-1} y_2^{-1}} \\
&= \frac{x}{(1 - 2xy_1)(1 - 2xy_2)(y_1 y_2 - 1)}
\end{aligned}$$

et ainsi $L(x) = [y_1^{-1} y_2^{-1}] F(x, y_1, y_2)$.

On vérifie bien que les bornes annoncées sont vérifiées sur cet exemple, même si sur cet exemple nous surestimons un peu, avec notamment $1 = \|P\|_1 \leqslant 2^{|\mathcal{A}_{\text{ex}}|} = 128$). ◀

Remarque 8.21.

▶ Nous n'utilisons plus l'astuce, propre à cet exemple particulier, utilisée dans l'introduction pour réduire le nombre de variables. En effet le but de cet exemple est désormais d'illustrer l'approche générale, adaptée à tout type de semilinéaire. ◀

8.3.2 Équation différentielle satisfaite par F

Dans cette section nous analysons le procédé utilisé par Lipshitz pour dériver l'équation différentielle satisfaite par F.

Nous nous intéressons à la fraction $F = \frac{P}{Q}$ en les variables $x, y_1, \ldots, y_d$.

Pour simplifier les notations dans un premier temps, supposons qu'il existe des constantes M_i telles que $\deg_{y_i}(P), \deg_{y_i}(Q) \leqslant M_i - 1$ pour tout $i \in [d]$, et M_0 telle que $\deg_x(P), \deg_x(Q) \leqslant M_0 - 1$. Par le Lemme 8.19, $M_0 = |Q_{\mathcal{A}}| + 1 \leqslant |\mathcal{A}|$, et $M_i = |\mathcal{A}|\|\mathcal{A}\|_{i,\infty}$.

Nous noterons $y_0 = x$ pour simplifier le traitement des différentes variables, lorsque x ne joue pas de rôle particulier par rapport aux variables $\boldsymbol{y}$.

Lemme 8.22.

▶ Soit $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_d) \in \mathbb{N}^{d+1}$. On note $\partial_{\boldsymbol{\alpha}}$ l'opérateur défini par $\partial_{\boldsymbol{\alpha}} = \partial_{y_1}^{\alpha_1} \ldots \partial_{y_d}^{\alpha_d} \partial_x^{\alpha_0}$. Alors il existe un polynôme $P_{\boldsymbol{\alpha}}$ de $\mathbb{Z}[x, \boldsymbol{y}]$ tel que $Q^{s+1}\partial_{\boldsymbol{\alpha}} F = P_{\boldsymbol{\alpha}}$, avec $s = \|\boldsymbol{\alpha}\|_1 = \sum_i \alpha_i$ et :

$$\deg_{y_i}(P_{\boldsymbol{\alpha}}) \leqslant (s+1)(M_i - 1)$$
$$\|P_{\boldsymbol{\alpha}}\|_1 \leqslant 2^s s! \|Q\|_1^s \|P\|_1 \prod_{i=0}^{d} (M_i - 1)^{\alpha_i}$$ ◀

Démonstration. Par récurrence sur $s = \|\boldsymbol{\alpha}\|_1$. Si $s = 0$, alors $QF = P$ et la proposition est vraie.

Soit $s > 0$ tel que la proposition soit vraie au rang $s - 1$. Soit $\boldsymbol{\alpha}$ tel que $\|\boldsymbol{\alpha}\|_1 = s$. Écrivons $\boldsymbol{\alpha} = \boldsymbol{\beta} + \boldsymbol{e_i}$, pour une coordonnée i telle que $\alpha_i > 0$; ainsi $\|\boldsymbol{\beta}\|_1 = s - 1$.

Par hypothèse de récurrence $Q^s \partial_{\boldsymbol{\beta}} F = P_{\boldsymbol{\beta}}$, où $P_{\boldsymbol{\beta}}$ satisfait les conclusions de la proposition au rang $s - 1$. Dérivons cette relation par rapport à y_i (en utilisant le temps de cette preuve la notation $y_0 = x$) :

$$(\partial_{y_i} Q) s Q^{s-1} \partial_{\boldsymbol{\beta}} F + Q^s \partial_{\boldsymbol{\alpha}} F = \partial_{y_i} P_{\boldsymbol{\beta}}.$$

Multiplions par Q :

$$Q^{s+1} \partial_{\boldsymbol{\alpha}} F = Q \partial_{y_i} P_{\boldsymbol{\beta}} - (\partial_{y_i} Q) s Q^s \partial_{\boldsymbol{\beta}} F = Q \partial_{y_i} P_{\boldsymbol{\beta}} - (\partial_{y_i} Q) s P_{\boldsymbol{\beta}}.$$

Posons $P_{\boldsymbol{\alpha}} = Q \partial_{y_i} P_{\boldsymbol{\beta}} - (\partial_{y_i} Q) s P_{\boldsymbol{\beta}}$. Le polynôme $P_{\boldsymbol{\alpha}}$ est bien à coefficients entiers.

De plus, pour $j \neq i$, $\deg_{y_j}(P_{\boldsymbol{\alpha}}) \leqslant \deg_{y_j}(Q) + \deg_{y_j}(P_{\boldsymbol{\beta}}) \leqslant_{H.R.} \deg_{y_j}(Q) + s(M_j - 1) \leqslant (s+1)(M_j - 1)$.

Et de même $\deg_{y_i}(P_{\boldsymbol{\alpha}}) \leqslant M_i - 1 + s(M_i - 1) - 1 \leqslant (s+1)(M_i - 1)$.

De plus :

$$\begin{aligned}
\|P_{\boldsymbol{\alpha}}\|_1 &\leqslant \|Q\|_1 \|\partial_{y_i} P_{\boldsymbol{\beta}}\|_1 + s \|\partial_{y_i} Q\|_1 \|P_{\boldsymbol{\beta}}\|_1 \\
&\leqslant \|Q\|_1 \deg_{y_i}(P_{\boldsymbol{\beta}}) \|P_{\boldsymbol{\beta}}\|_1 + s \deg_{y_i}(Q) \|Q\|_1 \|P_{\boldsymbol{\beta}}\|_1 \\
&\leqslant_{H.R.} 2s(M_i - 1) \|Q\|_1 \|P_{\boldsymbol{\beta}}\|_1 \\
&\leqslant_{H.R.} 2^s s! \|Q\|_1^s \|P\|_1 \prod_{i=0}^{d} (M_i - 1)^{\alpha_i}
\end{aligned}$$

■

Lemme 8.23.
▶ Soit $N > 0$ un entier. Soit $\boldsymbol{\alpha} \in \mathbb{N}^{d+1}$ tel que $\sum_i \alpha_i < N$. Il existe un polynôme $R_{\boldsymbol{\alpha}}$ en les variables $x, y_1, \ldots, y_d$, à coefficients entiers, tel que

$$Q^N \partial_{y_1}^{\alpha_1} \ldots \partial_{y_d}^{\alpha_d} \partial_x^{\alpha_0} F = R_{\boldsymbol{\alpha}}.$$

De plus

$$\deg_{y_i}(R_{\boldsymbol{\alpha}}) \leqslant N(M_i - 1)$$
$$\|R_{\boldsymbol{\alpha}}\|_1 \leqslant 2^s s! \|Q\|_1^{N-1} \|P\|_1 \prod_{i=0}^{d} (M_i - 1)^{\alpha_i}$$ ◀

Démonstration. Par le Lemme 8.22, nous savons que $Q^{s+1} \partial_{\boldsymbol{\alpha}} F = P_{\boldsymbol{\alpha}}$, avec $s = \|\boldsymbol{\alpha}\|_1$. On a donc $R_{\boldsymbol{\alpha}} = Q^{N-s-1} P_{\boldsymbol{\alpha}}$.
Ainsi $\deg_{y_i} R_{\boldsymbol{\alpha}} \leqslant (N - s - 1) \deg_{y_i}(Q) + \deg_{y_i}(P_{\boldsymbol{\alpha}}) \leqslant N(M_i - 1)$.
Et $\|R_{\boldsymbol{\alpha}}\|_1 \leqslant 2^s s! \|Q\|_1^{N-1} \|P\|_1 \prod_{i=0}^{d} (M_i - 1)^{\alpha_i}$.
■

Lemme 8.24 (Comparaison des dimensions).
▶ Soit $N > 1$ un entier quelconque.

- On note $\mathcal{D}$ l'ensemble des opérateurs $\partial_{y_1}^{\alpha_1} \ldots \partial_{y_d}^{\alpha_d} \partial_x^{\alpha_0}$, tels que $\sum_i \alpha_i < N$. Alors $|\mathcal{D}| = \binom{N+d}{d+1}$.
- On note $\mathcal{M}$ l'ensemble des monômes de la forme $y_1^{i_1} \ldots y_d^{i_d}$, avec $i_1 \leqslant N(M_1 - 1)$, …, $i_d \leqslant N(M_d - 1)$. Alors $|\mathcal{M}| = \prod_{i=1}^{d} \big((N(M_i - 1) + 1\big) < N^d \prod_{i=1}^{d} M_i$.
- Pour $N = (d+1)! \prod_{i=1}^{d} M_i - d$, on a l'inégalité stricte $|\mathcal{D}| > |\mathcal{M}|$.

◀

Démonstration. Les deux premiers points sont classiques. Il reste juste à vérifier que pour $N = (d+1)! \prod_{i=1}^{d} M_i - d$, on ait bien $\binom{N+d}{d+1} > N^d \prod_{i=1}^{d} M_i$:

$$\begin{aligned}
\frac{\binom{N+d}{d+1}}{N^d \prod_{i=1}^{d} M_i} &= \frac{(N+d)(N+d-1)\cdots(N+1)N}{(d+1)! N^d \prod_{i=1}^{d} M_i} \\
&= \frac{N+d}{(d+1)! \prod_{i=1}^{d} M_i} \prod_{i=0}^{d-1} (1 + \frac{i}{N}) \\
&\geqslant \frac{N+d}{(d+1)! \prod_{i=1}^{d} M_i} = 1\,.
\end{aligned}$$ ■

Exemple 8.25 (Sanity check).
▶ Dans l'exemple de l'automate $\mathcal{A}_{\text{ex}}$, nous avons $d = 2$ et $M_1 = M_2 = 3$. Ainsi

$$|\mathcal{D}| = \binom{N+2}{3} = N(N+1)(N+2)/6 \quad \text{et} \quad |\mathcal{M}| = (2N+1)^2\,.$$

$N = 22$ le rate de peu : $|\mathcal{D}| = 2024$ tandis que $|\mathcal{M}| = 2025$.

On peut calculer à la main que $N = 23$ est le plus petit N tel que $|\mathcal{D}| > |\mathcal{M}|$. On a alors $|\mathcal{D}| = 2300 > 2209 = |\mathcal{M}|$).

La valeur que nous annonçons dans la proposition précédente est $N = 6 \times 9 - 2 = 52$, qui est bien une borne supérieure de 23. ◀

Proposition 8.26 (Argument de dimension de Lipshitz).
▶ Soit $N = (d+1)!|\mathcal{A}|^d \prod_{i=1}^{d} \|\mathcal{A}\|_{i,\infty} - d$. La famille $\{\partial_{y_1}^{\alpha_1} \dots \partial_{y_d}^{\alpha_d} \partial_x^{\alpha_0} F \ : \ \sum_i \alpha_i < N\}$ est liée sur $\mathbb{Z}[x]$. Plus précisément, il existe une relation de dépendance non triviale de la forme

$$\sum_{\|\boldsymbol{\alpha}\|_1 < N} v_{\boldsymbol{\alpha}}(x)\ \partial_{y_1}^{\alpha_1} \dots \partial_{y_d}^{\alpha_d} \partial_x^{\alpha_0} F = 0,$$

où chaque $v_{\boldsymbol{\alpha}}(x)$ est un polynôme dans $\mathbb{Z}[x]$, de degré en x inférieur à $(d|\mathcal{A}|\|\mathcal{A}\|_\infty)^{O(d^2)}$, et $\log \|v_{\boldsymbol{\alpha}}\|_1 \leqslant (d|\mathcal{A}|\|\mathcal{A}\|_\infty)^{O(d^2)}$. ◀

Démonstration. Nous rappelons que $M_0 = |Q_{\mathcal{A}}| + 1 \leqslant |\mathcal{A}|$, et $M_i = |\mathcal{A}|\|\mathcal{A}\|_{i,\infty}$. Ainsi nous bornons les M_i par $|\mathcal{A}|\|\mathcal{A}\|_\infty$. Avec ces notations,

$$N \leqslant ((d+1)|\mathcal{A}|\|\mathcal{A}\|_\infty)^d \text{ et } |\mathcal{M}| \leqslant (N|\mathcal{A}|\|\mathcal{A}\|_\infty)^d \leqslant (d+1)^{d^2}(|\mathcal{A}|\|\mathcal{A}\|_\infty)^{d(d+1)}\,.$$

Par les Lemmes 8.23 et 8.24, pour tout $\partial_{\boldsymbol{\alpha}} \in \mathcal{D}$, $Q^N \partial_{\boldsymbol{\alpha}} F$ peut s'écrire comme une combinaison linéaire sur $\mathbb{Z}[x]$ d'éléments de $\mathcal{P}$. On peut donc écrire leur décomposition dans une matrice $\mathcal{R}$, qui est la matrice des vecteurs R_α, pour $\|\alpha\|_1 < N$, dans la base $\mathcal{M}$. Ainsi, $\mathcal{R}$ est une matrice de taille $|\mathcal{M}| \times |\mathcal{D}|$ à coefficients dans $\mathbb{Z}[x]$, qui de plus, par le lemme 8.24, a plus de colonnes que de lignes. Les colonnes de cette matrice sont donc liées.

Nous pouvons alors appliquer les formules de Cramer (cf Proposition 8.9) : il existe une solution non nulle $\boldsymbol{v}$ à l'équation $\mathcal{R}\boldsymbol{v} = \mathbf{0}$, telle que chaque coordonnée de $\boldsymbol{v}$ est un mineur de $\mathcal{R}$ ou l'opposé d'un mineur de $\mathcal{R}$ d'ordre $|\mathcal{M}|$. Comme $\mathcal{R}$ est une matrice à coefficients dans $\mathbb{Z}[x]$, les coordonnées de $\boldsymbol{v}$ sont aussi dans $\mathbb{Z}[x]$.

Ainsi, nous pouvons avons trouvé une relation de dépendance de la forme :

$$Q^N \sum_{\|\boldsymbol{\alpha}\|_1 < N} v_{\boldsymbol{\alpha}}(x)\ \partial_{y_1}^{\alpha_1} \dots \partial_{y_d}^{\alpha_d} \partial_x^{\alpha_0} F = 0.$$

Comme Q^N est un polynôme non nul, et la somme $\sum_{\|\boldsymbol{\alpha}\|_1 < N} v_{\boldsymbol{\alpha}}(x)\ \partial_{y_1}^{\alpha_1} \dots \partial_{y_d}^{\alpha_d} \partial_x^{\alpha_0} F$ est un élément du module $\mathcal{M}'$, par la Proposition 8.16, nous en déduisons que

$$\sum_{\|\boldsymbol{\alpha}\|_1 < N} v_{\boldsymbol{\alpha}}(x)\ \partial_{y_1}^{\alpha_1} \dots \partial_{y_d}^{\alpha_d} \partial_x^{\alpha_0} F = 0.$$

De plus pour tout α,

$$\deg_x(v_{\boldsymbol{\alpha}}) \leqslant |\mathcal{M}|N(M_0 - 1) \leqslant (d+1)^{d(d+1)}|\mathcal{A}|^{d(d+2)+1}\|\mathcal{A}\|_\infty^{d(d+2)}\,.$$

Nous pouvons alors appliquer les bornes du Lemme 8.7 aux mineurs de la matrice (en les appliquant à la transposée pour faire intervenir les normes des colonnes) : chaque mineur est borné par le produit des sommes des normes 1 de chacune de ses colonnes. Or chaque colonne du mineur est extrait d'une colonne de $\mathcal{R}$, qui est simplement la décomposition d'un $R_{\boldsymbol{\beta}}$ dans la base canonique de $\mathcal{M}$. Ainsi la somme des normes 1 dans $\mathbb{Z}[x]$ des composantes d'une colonne $\boldsymbol{\beta} \in \mathcal{D}$ de $\mathcal{R}$ est égale à $\|R_{\boldsymbol{\beta}}\|_1$ dans $\mathbb{Z}[x, y_1, \dots, y_d]$.

Pour tout β tel que $\|\beta\|_1 < N$, par les bornes du Lemme 8.23, nous avons

$$\|R_{\boldsymbol{\beta}}\|_1 \leqslant 2^{N-1}(N-1)!\|Q\|_1^{N-1}\|P\|_1(|\mathcal{A}|\|\mathcal{A}\|_\infty)^N\,.$$

Ainsi en utilisant la borne du Lemme 8.7 sur les déterminants :

$$\|v_{\boldsymbol{\alpha}}\|_1 \leqslant \left(2^{N-1}(N-1)!\|Q\|_1^{N-1}\|P\|_1(|\mathcal{A}|\|\mathcal{A}\|_\infty)^N\right)^{|\mathcal{M}|}. \tag{8.3}$$

En passant au log, et en utilisant les bornes sur N et $\mathcal{M}$, et le fait que $\|P\|_1, \|Q\|_1 < 2^{|\mathcal{A}|}$, nous obtenons finalement

$$\begin{aligned}\log(\|v_{\boldsymbol{\alpha}}\|_1) &\leqslant |\mathcal{M}|\big(N-1+(N-1)\log(N)+N|\mathcal{A}|+N\log(|\mathcal{A}|\|\mathcal{A}\|_\infty)\big)\\ &\leqslant (d|\mathcal{A}|\|\mathcal{A}\|_\infty)^{O(d^2)}\,(1+|\mathcal{A}|+d\log(d+1)+(d+1)\log(|\mathcal{A}|\|\mathcal{A}\|_\infty))\\ &= (d|\mathcal{A}|\|\mathcal{A}\|_\infty)^{O(d^2)}\end{aligned}$$

■

Exemple 8.27 .

Cela explique pourquoi j'ai simplifié F dans l'introduction.

▶ Pour l'exemple de l'automate $\mathcal{A}_{\text{ex}}$, je trouve en fait une relation de dépendance pour $N = 8$. L'équation différentielle satisfaite par F est trop grande pour être écrite ici. Elle vérifie $\deg_x(v_{\boldsymbol{\alpha}}) \leqslant 45$ et $\|v_{\boldsymbol{\alpha}}\|_1 \leqslant 348\,086\,586\,267\,256\,320$ pour tout $\|\alpha\|_1 < 8$.

Si on applique les bornes annoncées dans la preuve, avec $N = 8$, $\|Q\|_1 = 18$, $|\mathcal{M}| = 289$, $M_0 = M_1 = M_2 = 3$ (en prenant les degrés exacts de Q), nous obtenons $\deg_x(v_{\boldsymbol{\alpha}}) \leqslant 4624$ et par l'équation (8.3), $\|v_{\boldsymbol{\alpha}}\|_1 \leqslant (101\,108\,579\,083\,223\,040)^{289}$. Même s'il s'agit d'un exemple, qui n'est en aucun cas représentatif, il est probable que nos bornes surestiment beaucoup la taille des coefficients. ◀

8.3.3 Équation différentielle satisfaite par un langage de Parikh faiblement non ambigu

Nous pouvons désormais démontrer la proposition principale de cette section :

Proposition 8.3 (Taille de l'équation différentielle satisfaite par un langage de Parikh faiblement non ambigu).
▶ Soit $\mathcal{A}$ un automate de Parikh faiblement non ambigu. Alors la série $L(x)$ du langage de $\mathcal{A}$ satisfait une équation différentielle linéaire de la forme :

$$q_s(x)\partial_x^s L(x) + \cdots + q_0(x)L(x) = 0\,,$$

avec $s \leqslant (d|\mathcal{A}|\,\|\mathcal{A}\|_\infty)^{O(d)}$, et pour tout $i \in [0, s]$,

$$\begin{aligned}\deg(q_i) &\leqslant (d|\mathcal{A}|\,\|\mathcal{A}\|_\infty)^{O(d^2)}\,,\\ \log\|q_i\|_1 &\leqslant (d|\mathcal{A}|\,\|\mathcal{A}\|_\infty)^{O(d^2)}\,.\end{aligned}$$

◀

Démonstration. D'après la Proposition 8.26, il existe une relation de dépendance non triviale de la forme

$$\sum_{\|\boldsymbol{\alpha}\|_1<N} v_{\boldsymbol{\alpha}}(x)\;\partial_{y_1}^{\alpha_1}\ldots\partial_{y_d}^{\alpha_d}\partial_x^{\alpha_0}F = 0,$$

où $N \leqslant (d|A|\,\|A\|_\infty)^{O(d)}$, chaque $v_{\boldsymbol{\alpha}}(x)$ est un polynôme dans $\mathbb{Z}[x]$, de degré en x inférieur à $(d|\mathcal{A}|\|\mathcal{A}\|_\infty)^{O(d^2)}$, et $\log \|v_{\boldsymbol{\alpha}}\|_1 \leqslant (d|\mathcal{A}|\|\mathcal{A}\|_\infty)^{O(d^2)}$. Nous réécrivons cette équation différentielle sous la forme suivante :

$$\sum_{\|\boldsymbol{\beta}\|_1 < N} p_{\boldsymbol{\beta}}(x, \partial_x)\ \partial_{y_1}^{\beta_1} \dots \partial_{y_d}^{\beta_d} F = 0, \tag{8.4}$$

où pour $\boldsymbol{\beta} \in \mathbb{N}^d$, $p_{\boldsymbol{\beta}}(x, \partial_x) = \sum_{\alpha_0=0}^{N-\|\boldsymbol{\beta}\|_1} v_{\alpha_0,\boldsymbol{\beta}}(x)\partial_x^{\alpha_0}$.

On rappelle que F a été choisie telle que $[y_1^{-1}\cdots y_d^{-1}]F(x,\boldsymbol{y}) = L(x)$. Soit $\mathcal{B}$ l'ensemble des vecteurs qui interviennent dans la relation de dépendance, c'est-à-dire l'ensemble des $\boldsymbol{\beta} \in \mathbb{N}^d$ tels que $\|\boldsymbol{\beta}\|_1 < N$ et $p_{\boldsymbol{\beta}}(x, \partial_x) \neq 0$. Soit $\boldsymbol{\beta^{\min}}$ le plus petit vecteur de $\mathcal{B}$, pour l'ordre lexicographique.

Nous montrons alors que $p_{\boldsymbol{\beta^{\min}}}(x, \partial x)(L) = 0$. Nous rappelons les notations : $\boldsymbol{y}^{-\boldsymbol{\beta^{\min}}} = y_1^{-\beta_1^{\min}} \cdots y_d^{-\beta_d^{\min}}$, et $\boldsymbol{y^{-1}} = y_1^{-1}\cdots y_d^{-1}$.

Pour cela, nous admettons un instant l'égalité suivante : pour tout $\boldsymbol{\beta} \in \mathbb{N}^d$,

$$[\boldsymbol{y}^{-\boldsymbol{\beta^{\min}}-\mathbf{1}}]\partial_{y_1}^{\beta_1}\cdots\partial_{y_d}^{\beta_d}F = \begin{cases} 0 & \text{si } \boldsymbol{\beta} \neq \boldsymbol{\beta^{\min}} \\ (-1)^{\beta_1+\dots\beta_d}\beta_1!\dots\beta_d![\boldsymbol{y^{-1}}]F & \text{si } \boldsymbol{\beta} = \boldsymbol{\beta^{\min}} \end{cases} \tag{8.5}$$

On remarque alors que comme les $p_{\boldsymbol{\beta}}(\boldsymbol{x}, \partial x)$ ne dépendent pas des variables $\boldsymbol{y}$, en extrayant dans l'Équation (8.4) les coefficients en $\boldsymbol{y}^{-\boldsymbol{\beta^{\min}}-\mathbf{1}}$, on obtient l'égalité :

$$(-1)^{\beta_1+\dots\beta_d}\beta_1!\dots\beta_d!\cdot p_{\boldsymbol{\beta^{\min}}}(x,\partial)[\boldsymbol{y^{-1}}]F = 0.$$

Comme $L(x) = [\boldsymbol{y^{-1}}]F$, nous avons établi que $p_{\boldsymbol{\beta^{\min}}}(x, \partial x)(L) = 0$, ce qui conclut la preuve – les bornes sur les degrés, ordre et coefficients venant directement des bornes sur l'équation de F.

Il reste donc uniquement à prouver l'Équation (8.5). Rappelons que $\boldsymbol{\beta^{\min}}$ est le plus petit vecteur, pour l'ordre lexicographique, qui apparaît dans l'Équation (8.4). Rappelons que F appartient à $\mathbb{Q}[[x, y_1, y_1^{-1}, \dots, y_d^{-1}, y_d]]$. En regroupant les termes en x, on peut écrire F sous la forme :

$$F = \sum_{\boldsymbol{\alpha}\in\mathbb{Z}^d} H_{\boldsymbol{\alpha}}(x) y_1^{\alpha_1}\cdots y_d^{\alpha_d}.$$

Ainsi :

$$[\boldsymbol{y}^{-\boldsymbol{\beta^{\min}}-\mathbf{1}}]\partial_{y_1}^{\beta_1}\cdots\partial_{y_n}^{\beta_n}F = [\boldsymbol{y}^{-\boldsymbol{\beta^{\min}}-\mathbf{1}}]\sum_{\boldsymbol{\alpha}\in\mathbb{Z}^d} H_{\boldsymbol{\alpha}}(\boldsymbol{x})\partial_{y_1}^{\beta_1}\cdots\partial_{y_n}^{\beta_n} y_1^{\alpha_1}\cdots y_n^{\alpha_n}.$$

On peut alors remarquer que dès qu'un α_i est positif, $\partial_{y_1}^{\beta_1}\cdots\partial_{y_d}^{\beta_d} y_1^{\alpha_1}\cdots y_d^{\alpha_d}$ ne peut contribuer au coefficient de $\boldsymbol{y}^{-\boldsymbol{\beta^{\min}}-\mathbf{1}}$. On peut donc limiter la somme précédente aux exposants $\boldsymbol{\alpha}$ qui n'ont que des composantes strictement négatives (donc inférieures ou égales à -1). De plus, si $\boldsymbol{\beta} \in \mathcal{B}$ est différent de $\boldsymbol{\beta^{\min}}$, alors il existe une coordonnée i telle que $\beta_i > \beta_i^{\min}$, par définition de $\boldsymbol{\beta^{\min}}$. Alors l'exposant en y_i de $\partial_{y_1}^{\beta_1}\cdots\partial_{y_d}^{\beta_d} y_1^{\alpha_1}\cdots y_d^{\alpha_d}$ est $\alpha_i - \beta_i < -1 - \beta_i^{\min}$: donc ce terme ne contribue pas non plus au coefficient de $\boldsymbol{y}^{-\boldsymbol{\beta^{\min}}-\mathbf{1}}$. De même, par minimalité de $\boldsymbol{\beta^{\min}}$, $\boldsymbol{\alpha}$ doit être égal à $-\mathbf{1}$ pour pouvoir contribuer au terme $\boldsymbol{y}^{-\boldsymbol{\beta^{\min}}-\mathbf{1}}$. Finalement :

$$\begin{aligned}
[\boldsymbol{y}^{-\boldsymbol{\beta^{\min}}-\mathbf{1}}]\partial_{y_1}^{\beta_1}\cdots\partial_{y_n}^{\beta_n}F &= [\boldsymbol{y}^{-\boldsymbol{\beta^{\min}}-\mathbf{1}}]H_{-\mathbf{1}}(\boldsymbol{x})\partial_{y_1}^{\beta_1^{\min}}\cdots\partial_{y_n}^{\beta_n^{\min}}\boldsymbol{y}^{-1} \\
&= H_{-\mathbf{1}}(\boldsymbol{x})(-1)^{\beta_1^{\min}+\dots+\beta_n^{\min}}\beta_1^{\min}!\cdots\beta_n^{\min}! \\
&= (-1)^{\beta_1^{\min}+\dots+\beta_n^{\min}}\beta_1^{\min}!\cdots\beta_n^{\min}!\ [\boldsymbol{y^{-1}}]F\,,
\end{aligned}$$

qui est bien l'égalité annoncée à l'Équation (8.5). ■

Remarque 8.28 (Comparaison avec la borne de notre article ICALP).
▶ Dans notre article publié à ICALP, nous avions une meilleure borne sur le degré : $\deg(q_i) \leqslant (d|\mathcal{A}|\,\|\mathcal{A}\|_\infty)^{O(d)}$. Cette différence vient du fait que nous avions appliqué l'algorithme de Lipshitz dans $\mathbb{Q}$, et non dans $\mathbb{Q}[x]$. Avec cette méthode, la matrice associée à F pour le calcul de l'équation différentielle est plus grosse, mais les degrés des polynômes sont bornés en x, comme pour l'ordre, par N et non par la taille de la matrice. J'ai décidé d'utiliser des matrices dans $\mathbb{Q}[x]$ dans ce chapitre, car cela permet d'essayer la méthode sur des exemples simples, pas trop gros (la taille de la matrice est un facteur très limitant en pratique). Les tailles des coefficients des polynômes ont dans les deux versions la même borne, qui est au moins exponentiellement plus grande que celle du degré, si bien que cette différence n'a pas de conséquence dans la suite du chapitre . ◀

8.4 Le problème de l'inclusion des automates de Parikh faiblement non ambigus

Nous pouvons enfin attaquer le problème principal de ce chapitre, à savoir le problème de l'inclusion. Nous rappelons le contexte : soient deux automates de Parikh $\mathcal{A}$ et $\mathcal{B}$ faiblement non ambigus, donnés en entrée sous la forme précisée au début de chapitre. Nous notons $d_\mathcal{A}$ et $d_\mathcal{B}$ leurs degrés. Nous cherchons à décider si $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. Nous rappelons la méthode inspirée de celle de [SI85] pour résoudre le problème :

a. Nous bornons les ordres, degrés, et coefficients des équations différentielles de la série génératrice $A(x) = \sum_n a_n x^n$ de $\mathcal{L}(\mathcal{A})$, et de la série génératrice $C(x) = \sum_n c_n x^n$ de $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$ en sous-section 8.4.1.

b. En sous-section 8.4.1, nous bornons l'ordre, le degré, et les coefficients de l'équation différentielle satisfaite par $D(x) := A(x) - C(x)$, la série de comptage des mots qui sont dans $\mathcal{L}(\mathcal{A})$ mais pas dans $\mathcal{L}(\mathcal{B})$.

c. Nous bornons l'ordre et les racines du polynôme de tête de la récurrence linéaire satisfaite par la suite $d_n := a_n - c_n$, en sous-section 8.4.2. Nous en déduisons la borne sur la taille du plus petit témoin de non inclusion annoncée dans le Théorème 8.4.

d. Nous développons un algorithme élémentaire (qui n'utilise pas d'algorithmes de calcul formel) d'énumération pour résoudre le problème d'inclusion, à l'aide de la borne sur la taille du mot témoin. Cette partie est présentée en sous-section 8.4.3.

8.4.1 Bornes sur les équations différentielles

Dans un premier temps, nous calculons les bornes sur l'équation différentielle satisfaite par la série génératrice $C(x)$ du langage $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$:

Proposition 8.29 (Taille de l'intersection).
▶ La série $C(x)$ du langage $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$ satisfait une équation différentielle linéaire de la forme :

$$q_s(x)\partial_x^s C(x) + \cdots + q_0(x)C(x) = 0\,,$$

avec $s \leqslant ((d_{\mathcal{A}} + d_{\mathcal{B}})|\mathcal{A}||\mathcal{B}| \max(\|\mathcal{A}\|_\infty, \|\mathcal{B}\|_\infty))^{O(d_{\mathcal{A}}+d_{\mathcal{B}})}$, et pour tout $i \in [0, s]$,

$$\deg(q_i) \leqslant ((d_{\mathcal{A}} + d_{\mathcal{B}})|\mathcal{A}||\mathcal{B}| \max(\|\mathcal{A}\|_\infty, \|\mathcal{B}\|_\infty))^{O((d_{\mathcal{A}}+d_{\mathcal{B}})^2)},$$
$$\log \|q_i\|_1 \leqslant ((d_{\mathcal{A}} + d_{\mathcal{B}})|\mathcal{A}||\mathcal{B}| \max(\|\mathcal{A}\|_\infty, \|\mathcal{B}\|_\infty))^{O((d_{\mathcal{A}}+d_{\mathcal{B}})^2)}.$$

◀

Démonstration. Il suffit de spécialiser les bornes de la Proposition 8.3 pour l'automate produit $\mathcal{C} = \mathcal{A} \times \mathcal{B}$ qui calcule l'intersection : on a alors $|\mathcal{C}| \leqslant |\mathcal{A}| \times |\mathcal{B}|$, $d_{\mathcal{C}} = d_{\mathcal{A}} + d_{\mathcal{B}}$ et $\|\mathcal{C}\|_\infty = \max(\|\mathcal{A}\|_\infty, \|\mathcal{B}\|_\infty)$. ■

Pour borner l'ordre, le degré, et les coefficients de l'équation différentielle satisfaite par $D(x) := A(x) - C(x)$, nous avons besoin du lemme suivant :

Proposition 8.30 (Somme/soustraction de séries holonomes [Kau14]).
▶ Soient deux séries holonomes $A(x)$ et $C(x)$ satisfaisant deux équations différentielles linéaires non triviales de la forme :

$$p_r^A(x)\partial^r A(x) + \cdots + p_0^A(x)A(x) = 0 \quad \text{et} \quad p_r^C(x)\partial^r C(x) + \cdots + p_0^C(x)C(x) = 0.$$

Soit $D_{\max} = \max_{i\in[r]}(\deg(p_i^A), \deg(p_i^C))$ et $S_\infty = \max_{i\in[r]}(\|p_i^A)\|_\infty, \|p_i^C\|_\infty)$. Alors, $D(x) = A(x) - C(x)$ vérifie une équation différentielle linéaire non triviale de la forme

$$q_{2r}(x)\partial^{2r} D(x) + \cdots + q_0(x)D(x) = 0,$$

où pour tout $i \in [2r]$, $\log(\|q_i\|_\infty) \leqslant O(r^2(1 + \log(r) + \log(D_{\max}) + \log(S_\infty))$
et $\deg(q_i) \leqslant 2(r+1)D_{\max}$.

◀

Démonstration. Nous appliquons le résultat de [Kau14, Theorem 2], qui nous permet d'affirmer que $D(x)$ satisfait une équation différentielle non triviale de la forme :

$$q_{2r}\partial^{2r} D(x) + \cdots + q_0 D(x) = 0$$

où pour chaque $i \in [2r]$, $\deg(q_i) \leqslant 2(r+1)D_{\max}$. De plus, pour tout $i \in [2r]$,

$$\begin{aligned}\mathrm{ht}(\|q_i\|_\infty) \leqslant\ & \mathrm{ht}(2r) + \mathrm{ht}((2r+1)!) + (2r+1)\mathrm{ht}(D_{\max}) \\ & + (2r+2)((2r)(\mathrm{ht}(1) + \mathrm{ht}(D_{\max})) + \mathrm{ht}(S_\infty))\end{aligned}$$

avec $\mathrm{ht}(x) = \log(1 + |x|)$. Comme pour tout $x \geqslant 1$, $\log(1 + x) \leqslant 1 + \log(x)$, nous pouvons en déduire que :

$$\log(\|q_i\|_\infty)) \leqslant 8(r+1)^2 + (2r+2)\log(r) + (4r^2+6r+1)\log(D_{\max}) + (2r+2)\log(S_\infty).$$

■

En appliquant ce lemme à $A(x)$ et $C(x)$, nous déduisons directement la proposition suivante :

Proposition 8.31 (Bornes sur l'équation satisfaite par $D(x)$).
▶ La série $D(x) := A(x) - C(x)$ satisfait une équation différentielle non triviale de la forme

$$q_s(x)\partial_x^s D(x) + \cdots + q_0(x)D(x) = 0\,,$$

avec $s \leqslant ((d_{\mathcal{A}} + d_{\mathcal{B}})|\mathcal{A}||\mathcal{B}|\ \max(\|\mathcal{A}\|_\infty, \|\mathcal{B}\|_\infty))^{O(d_{\mathcal{A}}+d_{\mathcal{B}})}$, et pour tout $i \in [0, s]$,

$$\deg(q_i) \leqslant ((d_{\mathcal{A}} + d_{\mathcal{B}})|\mathcal{A}||\mathcal{B}|\ \max(\|\mathcal{A}\|_\infty, \|\mathcal{B}\|_\infty))^{O((d_{\mathcal{A}}+d_{\mathcal{B}})^2)}\,,$$
$$\log \|q_i\|_\infty \leqslant ((d_{\mathcal{A}} + d_{\mathcal{B}})|\mathcal{A}||\mathcal{B}|\ \max(\|\mathcal{A}\|_\infty, \|\mathcal{B}\|_\infty))^{O((d_{\mathcal{A}}+d_{\mathcal{B}})^2)}\,.$$

◀

Démonstration. On applique la Proposition 8.30, associée aux Propositions 8.3 et 8.29. Les grands O en exposants absorbent beaucoup les détails plus fins des expressions de la Proposition 8.30, et on obtient alors les mêmes types de bornes pour $D(x)$ que pour $C(x)$. ■

8.4.2 Bornes sur la récurrence et témoin de non inclusion

En une variable, la suite des coefficients d'une série holonome $D(x)$ satisfait une équation de récurrence linéaire à coefficients polynomiaux. Nous voulons dans cette section établir des bornes sur le degré et la taille des coefficients du polynôme de tête de la récurrence satisfaite par la suite $d_n = a_n - c_n$, qui compte le nombre de mots qui sont dans $\mathcal{L}(A)$ mais pas dans $\mathcal{L}(B)$. Pour cela, nous avons besoin de transformer l'équation différentielle satisfaite par $D(x)$ en récurrence sur d_n. Nous commençons par une proposition technique :

Proposition 8.32 (Conversion d'une équation différentielle en récurrence linéaire).
▶ Soit $H(x) = \sum u_n x^n$ une série satisfaisant l'équation différentielle suivante :

$$q_r(x)\partial_r H(x) + \ldots + q_0(x)H(x) = 0\,.$$

Alors la suite (u_n) satisfait une récurrence de la forme :

$$\sum_{k=-s}^{S} t_k(n)u_{n+k} = 0, \text{ pour } n \geqslant s, \text{ avec } t_S \neq 0,$$

et de plus $0 \leqslant s \leqslant \max_i(\deg(q_i)), \qquad 0 \leqslant S \leqslant r,$

$$\|t_S\|_\infty \leqslant \max_i(\|q_i\|_\infty) \cdot r^{r+1}, \qquad \deg(t_S) \leqslant r\,.$$

◀

Démonstration. Sans perte de généralité, nous pouvons supposer que pour un certain indice $k_0 \in [0, r]$, $q_{k0}(0) \neq 0$. Sinon, nous pouvons nous ramener à ce cas en divisant l'équation différentielle par le plus grand monôme x^m qui divise tous les polynômes q_k. Remarquons que cette opération de division par x^m ne change pas la norme infinie des coefficients polynomiaux, et diminue leur degré.

Soit $D = \max_i \deg(q_i)$ le plus grand degré des coefficients polynomiaux de l'équation différentielle. Nous pouvons réécrire cette équation satisfaite par H sous la forme suivante :

$$\sum_{k=0}^{r}\sum_{k'=0}^{D} a_{k,k'}x^{k'}\partial_x^k H(x) = 0.$$

Cette équation se traduit alors en la relation de récurrence suivante, valable pour $n \geqslant D$, sur les coefficients de $H(x)$

$$\sum_{k=0}^{r}\sum_{k'=0}^{D} a_{k,k'}(n-k'+1)(n-k'+2)\dots(n-k'+k)u_{n-k'+k} = 0.$$

En posant $j = k - k'$, nous pouvons réécrire cette égalité sous la forme :

$$\sum_{j=-D}^{r} t_j(n)u_{n+j} = 0 \text{ avec } t_j(n) = \sum_{k=\max(0,j)}^{\min(r,D+j)} a_{k,k-j}\prod_{\ell=1}^{k}(n+j-\ell+1)$$

Cependant, certains polynômes t_j peuvent être nuls ; nous devons préciser le terme de tête de la récurrence.

Posons $S := \max\{k-k' \mid k \in [0,r], k' \in [0,D] \text{ et } a_{k,k'} \neq 0\}$. L'hypothèse $q_{k_0}(0) \neq 0$ implique que $a_{k_0,0} \neq 0$, si bien que $S \geqslant k_0 \geqslant 0$.

Montrons que $t_S(n)$ est bien le polynôme de tête dans la récurrence. Par maximalité de S, $t_j(n) = 0$ pour $S < j \leqslant r$. De plus par définition, $t_S(n) = \sum_{k=S}^{r} r_k(n)$ où $r_k(n) = a_{k,k-S}\prod_{\ell=1}^{k}(n+S-\ell+1)$ pour tout $k \in [S,r]$. Par définition de S, au moins un des r_k est non nul. Comme pour tout $k \in [S,r]$, $\deg(r_k) = k$ dès que $a_{k,k-S} \neq 0$, nous savons que $t_S(n) \neq 0$ et $t_S(n)$ est de degré au plus r.

Il ne reste plus qu'à borner la taille des coefficients de t_S. Nous remarquons qu'en développant le produit $\prod_{\ell=1}^{k}(n+S-\ell+1) = \sum_{m=0}^{k} b_m n^m$, nous avons, pour tout $m \in [k]$, $b_m = \sum_{I\subseteq[k],|I|=k-m}\prod_{\ell\in I}(S-\ell+1)$, si bien que $|b_m| \leqslant \binom{k}{m} r^{k-m} \leqslant k^m r^{k-m} \leqslant r^r$. Ainsi :

$$\begin{array}{rcl} \|t_S\|_\infty & \leqslant & \sum_{k=S}^{r}|a_{k,k-S}| \cdot \|\prod_{\ell=1}^{k}(n+S-\ell+1)\|_\infty \\ & \leqslant & \sum_{k=S}^{r}|a_{k,k-S}| \cdot r^r \\ & \leqslant & \max_i(\|q_i\|_\infty) \cdot r^{r+1}. \end{array}$$

■

Pour pouvoir localiser les racines du polynôme de tête, nous avons besoin du lemme suivant :

Lemme 8.33 (Localisation des racines).
▶ Soit $P \neq 0$ un polynôme de $\mathbb{Z}[x]$. Alors $P(n) \neq 0$ pour tout $n \geqslant \|P\|_\infty + 1$. ◀

Démonstration. Soit $n \geqslant \|P\|_\infty + 1$. On écrit $P(x) = \sum_{k=0}^{d} a_k x^k$. Supposons par l'absurde que $P(n) = 0$. Alors :

$$n^d \leqslant \left|a_d n^d\right| = \left|\sum_{k=0}^{d-1} a_k n^k\right| \leqslant \|P\|_\infty \frac{n^d-1}{n-1} \leqslant n^d - 1$$

ce qui mène à une contradiction. ■

Nous pouvons désormais borner la récurrence satisfaite par d_n, et en déduire un majorant de la taille du plus petit mot témoin de non inclusion :

Théorème 8.4 (Taille du plus petit témoin de non inclusion).
▶ Si $\mathcal{L}(A)$ n'est pas inclus dans $\mathcal{L}(B)$, alors il existe un mot w qui soit dans $\mathcal{L}(\mathcal{A})$ mais pas dans $\mathcal{L}(\mathcal{B})$ tel que :

$$|w| \leqslant 2^{(dM)^{O(d^2)}},$$

avec $d = d_{\mathcal{A}} + d_{\mathcal{B}}$, et $M = |\mathcal{A}||\mathcal{B}|\max(\|\mathcal{A}\|_\infty, \|\mathcal{B}\|_\infty)$. ◀

Démonstration. Pour $n \in \mathbb{N}$, on note a_n (resp c_n) le nombre de mots de taille n dans $\mathcal{L}(\mathcal{A})$ (resp. $\mathcal{L}(\mathcal{B})$). On pose $d_n = a_n - c_n$ le nombre de mots de taille n qui sont dans $\mathcal{L}(\mathcal{A})$ mais pas dans $\mathcal{L}(B)$. Alors par les Proposition 8.31 et Proposition 8.32, la suite (d_n) satisfait une récurrence linéaire de la forme

$$\sum_{k=-s}^{S} t_k(n) d_{n+k} = 0, \quad \text{pour } n \geqslant s, \text{ avec } t_S \neq 0$$

que l'on réécrit sous la forme :

$$t_S(n-S)d_n = -\sum_{k=1}^{S+s} t_{S-k}(n-S)d_{n-k} \quad \text{pour } n \geqslant S+s,$$

avec $S + s \leqslant (dM)^{O(d)}$, $\deg(t_S) \leqslant (dM)^{O(d)}$ et $\log(\|t_S\|_\infty) \leqslant (dM)^{O(d^2)}$.

Soit $n_0 \in \mathbb{N}$. Si $t_S(n_0 - S)$ n'est pas nul, et si $u_{n_0-1} = \cdots = u_{n_0-S-s} = 0$ alors $u_{n_0} = 0$. Pour être sûr que $n_0 - S$ n'est pas une racine de t_S, il suffit de choisir $n_0 - S \geqslant \|t_S\|_\infty + 1$ par le Lemme 8.33). En posant $W = s + S + \|t_S\|_\infty + 1$, nous avons donc l'équivalence :

$$D(x) = 0 \quad \text{si et seulement si} \quad \forall n \leqslant W,\ d_n = 0\,.$$

Autrement dit, du point de vue des langages, $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$ si et seulement si $L(\mathcal{A}) \setminus L(\mathcal{B})$ contient un mot de longueur au plus $W \leqslant 2^{(dM)^{O(d^2)}}$. ■

8.4.3 Borne de complexité du problème de l'inclusion

Dans cette section, nous développons un algorithme de décision du problème de l'inclusion. Cet algorithme est élémentaire dans le sens où il ne demande aucun outil de calcul formel. Le but est de prouver le corollaire suivante :

Corollaire 8.5 (Borne de complexité du problème de l'inclusion).
▶ Soient deux automates de Parikh faiblement non ambigus $\mathcal{A}$ and $\mathcal{B}$ de dimensions $d_\mathcal{A}$ et $d_\mathcal{B}$. On peut décider le problème de l'inclusion $L(\mathcal{A}) \subseteq L(\mathcal{B})$ en temps $2^{2^{O(d^2 \log(dM))}}$ où $d = d_\mathcal{A} + d_\mathcal{B}$ et $M = |\mathcal{A}|\,|\mathcal{B}|\,\max(\|\mathcal{A}\|_\infty, \|\mathcal{B}\|_\infty)$. ◀

D'après le Théorème 8.4, si $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{B})$ alors on peut trouver un mot w de taille $|w| \leqslant W = 2^{(dM)^{O(d^2)}}$, avec $d = d_\mathcal{A} + d_\mathcal{B}$ et $M = |\mathcal{A}||\mathcal{B}| \max(\|\mathcal{A}\|_\infty, \|\mathcal{B}\|_\infty)$, qui soit dans $\mathcal{L}(\mathcal{A})$ mais pas dans $\mathcal{L}(\mathcal{B})$. L'algorithme de résolution du problème de l'inclusion est simple : nous allons compter sur les automates le nombre de mots de taille n dans $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$ et dans $\mathcal{L}(\mathcal{A})$, jusqu'à la taille $n = W$. Autrement dit nous calculons sur les automates les W premiers termes des séries génératrices des langages $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$ et $\mathcal{L}(\mathcal{A})$. Si les deux valeurs calculées correspondent jusqu'à la taille W, alors il y a bien inclusion $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$, sinon on a trouvé la taille du plus petit mot dans $\mathcal{L}(\mathcal{B}) \setminus \mathcal{L}(\mathcal{A})$.

Une approche naïve consiste à énumérer tous les calculs possibles de longueur n des automates, et de vérifier si le calcul est acceptant ; cependant cette énumération a un surcoût exponentiel en n, et comme n va aller jusqu'à W, qui est déjà doublement exponentiel, on aimerait éviter une exponentielle supplémentaire.

La proposition suivante explique comment compter simplement les calculs de longueur n de l'automate :

Proposition 8.34 (Dénombrement élémentaire des calculs de l'automate).
▶ Soit $\mathcal{A} = (\Sigma, Q, q_I, F, C, \Delta)$ un automate de Parikh faiblement non ambigu de dimension $d \geqslant 1$, donné sous la forme précisée au début du chapitre. Soit $W \in \mathbb{N}$ tel que $W \geqslant d|\mathcal{A}|\|\mathcal{A}\|_\infty$. Alors on peut compter le nombre total de calculs de $\mathcal{A}$ de longueur inférieure à W qui joignent q_I à un état final, groupés par valeur du vecteur qui les étiquette, en temps $W^{O(d)}$. ◀

Démonstration. Dans un premier temps, nous remarquons que tout calcul de longueur n dans $\mathcal{A}$ est étiqueté par un vecteur $\boldsymbol{v}$ de norme infinie inférieure à $n\|\mathcal{A}\|_\infty$.

Pour tout $q \in Q$, tout entier $n \leqslant W$, tout vecteur $\boldsymbol{v} \in \mathbb{N}^d$, nous notons $a_q(n, \boldsymbol{v})$ le nombre de calculs de $\mathcal{A}$ de longueur n, partant de q, finissant dans un état final, et étiquetés par le vecteur $\boldsymbol{v}$. On a facilement $a_q(n, \boldsymbol{v}) \leqslant |\Delta|^n$.

Pour $\boldsymbol{v}, \boldsymbol{u}$ deux vecteurs de $\mathbb{N}^d$, nous notons $\boldsymbol{u} \preceq \boldsymbol{v}$ si pour tout $i \in [d]$, $u_i \leqslant v_i$.

On vérifie simplement que les $a_q(n, \boldsymbol{v})$ vérifient la récurrence suivante, pour tout $q \in Q$, $n \geqslant 1$, et $\|\boldsymbol{v}\|_\infty \leqslant n\|\mathcal{A}\|_\infty$:

Il s'agit simplement du système linéaire satisfait par les séries génératrices, exprimé du point de vue des coefficients.

$$a_q(n, \boldsymbol{v}) = \sum_{\substack{(q,(a,\boldsymbol{u}),q') \in \Delta \\ \text{avec } \boldsymbol{u} \preceq \boldsymbol{v}}} a_{q'}(n-1, \boldsymbol{v} - \boldsymbol{u}). \tag{8.6}$$

avec comme conditions initiales $a_q(0, \mathbf{0}) = 1$ si $q \in F$, 0 sinon.

L'énumération des vecteurs $\boldsymbol{v}$ de norme infinie inférieure à $W\|\mathcal{A}\|_\infty$, dans l'ordre lexicographique, se fait en $(W\|\mathcal{A}\|_\infty + 1)^d \leqslant W^{O(d)}$ itérations. Pour chaque vecteur $\boldsymbol{v}$, et pour tout n allant de 1 à W, on utilise l'équation 8.6 pour calculer $a_q(n, \boldsymbol{v})$ (l'ordre lexicographique assure que toutes les valeurs de la somme de l'équation 8.6 ont bien été calculées avant) : on effectue une itération sur les transitions sortantes de q, qui sont au plus $|\Delta|$; dans chacune de ces transitions, on calcule la valeur du vecteur $\boldsymbol{v} - \boldsymbol{u}$ en temps $O\big(d \log(n\|\mathcal{A}\|_\infty)\big) \leqslant O(W^2)$, puis on ajoute la valeur de $a_{q'}(n, \boldsymbol{u})$ à $a_q(n, \boldsymbol{v})$, en temps $O\big(n \log(|\Delta|)\big) \leqslant O(W^2)$.

On peut donc calculer tous les $a_q(n, \boldsymbol{v})$, pour tout $n \leqslant W$, tout $q \in Q$ et $\|\boldsymbol{v}\|_\infty \leqslant W\|\mathcal{A}\|_\infty$ en $W^{O(d)}$ opérations.

■

Proposition 8.35 (Énumération des vecteurs du semilinéaire).
▶ Soit $C = \cup_{i=1}^{p} C_i$ un semilinéaire de dimension $d \geqslant 1$, représenté sous une forme non ambiguë, avec $C_i = \boldsymbol{c_i} + P_i^*$. Soit $r \in \mathbb{N}$ tel que $r \geqslant dp$. Alors on peut énumérer tous les vecteurs de C de norme infinie plus petite que r en temps $r^{O(\sum_i |P_i|)}$. ◀

Démonstration. Soit $1 \leqslant i \leqslant p$ tel que $r \geqslant \|\boldsymbol{c_i}\|_\infty$. On note $P_i = \{\boldsymbol{p_1}, \ldots, \boldsymbol{p_{|P_i|}}\}$ (on omet les indices i pour plus de lisibilité). Quitte à retirer certains de vecteurs P_i, on peut supposer que $r \geqslant \max_{\boldsymbol{p} \in P_i}(\|\boldsymbol{p}\|_\infty)$.

Pour tout vecteur $\boldsymbol{u}$ dans $[0, r]^{|P_i|}$, on note $b_i(\boldsymbol{u})$ le vecteur $\boldsymbol{c_i} + \sum_{k=1}^{|P_i|} u_k \boldsymbol{p_k}$. On énumère ainsi tous les vecteurs de C_i par une simple boucle à $(r+1)^{|P_i|}$ itérations. Si $\boldsymbol{u}$ est tel que $u_k \geqslant 1$ alors $b_i(\boldsymbol{u}) = b_i(\boldsymbol{u} - \boldsymbol{e_k}) + \boldsymbol{p_k}$ se calcule en temps $O(d \log(r))$.

On peut donc énumérer tous les vecteurs de C_i en temps $O((r+1)^{|P_i|} d \log(r)) \leqslant r^{O(|P_i|)}$, et donc tous les vecteurs de C en $r^{O(\sum_i |P_i|)}$ opérations. ■

Nous pouvons désormais démontrer le Corollaire 8.5 :

Preuve du Corollaire 8.5. On pose $W = 2^{(dM)^{O(d^2)}}$, avec $d = d_{\mathcal{A}} + d_{\mathcal{B}}$ et $M = |\mathcal{A}||\mathcal{B}| \max(\|\mathcal{A}\|_\infty, \|\mathcal{B}\|_\infty)$. On rappelle que $\mathcal{C}$ désigne l'automate produit qui calcule $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$.

Nous calculons dans un premier temps les vecteurs du semilinéaire de $\mathcal{A}$ et de $\mathcal{C}$ de norme infinie inférieure à $W \max(\|\mathcal{A}\|_\infty, \|\mathcal{B}\|_\infty)$. D'après la Proposition 8.35, cela se fait en $(W \max(\|\mathcal{A}\|_\infty, \|\mathcal{B}\|_\infty)^{O(|\mathcal{A}|+|\mathcal{B}|)}$ opérations.

Nous comptons ensuite tous les calculs de longueur inférieure à W joignant l'état initial à un état final, groupés par le vecteur qui les étiquette, de l'automate $\mathcal{A}$ et de l'automate produit $\mathcal{C}$. Cela se fait d'après la Proposition 8.34 en $W^{O(d)}$ opérations. On modifie légèrement la procédure de la Proposition 8.34, sans changer sa complexité, pour additionner lors de l'énumération des vecteurs, tous les $a_{q_I}(n, \boldsymbol{v})$ calculés tels que $\boldsymbol{v} \in C$ afin d'avoir à la fin de la procédure, pour chaque $n \leqslant W$, le nombre de calculs acceptants de longueur n.

On peut ainsi calculer en $2^{(dM)^{O(d^2)}}$ opérations le nombre de mots de longueur inférieure à W dans $\mathcal{L}(\mathcal{A})$ et dans $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$, et ainsi décider le problème de l'inclusion en $2^{(dM)^{O(d^2)}}$ opérations . ■

8.5 Conclusion, et ouverture

Dans ce chapitre, nous avons utilisé le caractère holonome de la série génératrice des automates de Parikh faiblement non ambigus pour en déduire un algorithme élémentaire de résolution du problème de l'inclusion, avec une première borne sur sa complexité. Comme nous l'avons expliqué, nous sommes allés au chemin le plus direct pour obtenir ces bornes, qui s'avèrent sur des exemples simples surestimer un peu trop la taille des coefficients.

Je pense qu'il est possible, en suivant la même démarche que nous avons utilisée, d'obtenir des majorations plus petites, à l'aide de bornes plus précises lors des calculs, et notamment pour l'étude de l'algorithme de Lipshitz. Il arrive aussi en calcul formel que les coefficients des calculs intermédiaires d'un algorithme grossissent artificiellement pour s'effondrer à la toute fin ; c'est le cas par exemple pour le calcul du pgcd par l'algorithme d'Euclide [VZGG13]. Il faudrait s'assurer que nous ne sommes pas confrontés au même problème dans notre analyse de l'algorithme de Lipshitz.

Une autre possibilité, pour continuer le travail dans une autre direction, serait d'étudier des algorithmes plus efficaces de calcul de diagonales de fractions rationnelles, comme ceux à base de *télescopage créatif* ([Chy14, BCLS18]) : peut-être permettront-ils d'obtenir des bornes plus fines sur les équations différentielles. C'est un travail que j'aimerais bien continuer, mais qui demandera un investissement mathématique important. Une autre idée serait d'essayer de prendre en compte le caractère $\mathbb{N}$-rationnel des séries étudiées dans les algorithmes de calcul de diagonales.

Dans ce chapitre, nous avons utilisé les algorithmes sur les séries holonomes pour borner uniquement la taille du plus petit témoin de non inclusion : l'algorithme final que nous fournissons pour résoudre l'inclusion est élémentaire et repose uniquement sur cette borne. Il serait intéressant de résoudre le problème de l'inclusion en calculant en pratique les équations différentielles avec des outils de calcul formel, et d'étudier sa complexité.

Pour ce qui concerne la complexité du problème de l'inclusion, il nous manque des familles de langages pour tester nos bornes, qui vérifieraient que la taille du plus petit témoin de non inclusion est "grande" : si possible exponentielle ou doublement exponentielle en la taille des automates. Avant d'étudier la complexité théorique du problème de l'inclusion des automates de Parikh faiblement non ambigus, il est

plus intéressant d'étudier celle de l'universalité : étant donné un automate de Parikh faiblement non ambigu $\mathcal{A}$, est-ce que $\mathcal{L}(\mathcal{A}) = \Sigma^*$? En effet, dans [Cle20], l'auteur explique comment le problème de l'inclusion $L \subseteq M$ est souvent équivalent au problème de l'universalité de $(M \cap L) \cup (\Sigma^* \setminus L)$, où L peut être supposé sans perte de généralité déterministe. Son approche présente ainsi le problème de l'universalité comme central dans l'étude du problème de l'inclusion pour de nombreuses classes d'automates, dont les automates de Parikh faiblement non ambigus : comme les automates de Parikh déterministes sont clos par complémentaire, et les automates de Parikh faiblement non ambigus sont clos par union disjointe et intersection, le langage $(M \cap L) \cup (\Sigma^* \setminus L)$ est bien reconnu par un automate de Parikh faiblement non ambigu.

Enfin, il est naturel d'étendre la démarche de ce chapitre avec les automates de Parikh à pile. Le problème de l'inclusion étant indécidable pour les langages algébriques non ambigus, la méthode ne se généralise pas, mais elle peut s'appliquer au problème de l'universalité $\mathcal{L}(\mathcal{A}) = \Sigma^*$. Deux directions sont possibles pour aborder le problème :

— reprendre l'algorithme de Lipshitz depuis le début, et l'analyser dans le cas où la série holonome dont on calcule une diagonale est algébrique ;

— sinon réutiliser l'étude du cas rationnel, en utilisant le fait que toute série algébrique est une diagonale de fraction rationnelle [DL87].

La seconde me semble dans un premier temps plus envisageable que la première.

Conclusion

J'ai fait le choix pour cette thèse de conclure la plupart des chapitres par quelques ouvertures, qu'il serait redondant de recopier ici. Je profite plutôt de cet espace de discussion pour récapituler les principales contributions de cette thèse et conclure par des réflexions générales sur les sujets que nous avons abordés.

La première partie a été en très grande partie dédiée à démontrer qu'utiliser des arbres d'expressions suivant la distribution uniforme, que ce soit pour une analyse théorique en moyenne ou pour produire des benchmarks, était *a priori* une mauvaise idée ; nous avons également montré que la situation paraissait moins tranchée pour la distribution ABR. Plus précisément :

— Dans le Théorème 2.32 nous avons montré qu'en présence d'un motif absorbant pour au moins un des opérateurs, les arbres d'expressions décrits par des systèmes se réduisent drastiquement à une taille bornée en moyenne. Les conditions sur le système sont techniques mais suffisamment naturelles pour s'appliquer à de nombreux exemples de la littérature.

— Notre étude permet aussi de borner les moments d'ordres supérieurs associés à cette réduction (Théorème 2.60) ; ainsi nous avons montré dans la Proposition 2.61 que tout problème sur les expressions pour lequel on connaît un algorithme de complexité en temps polynomial admet une complexité au plus linéaire en moyenne – tout le temps de calcul servant en réalité à calculer la simplification.

— Dans le Théorème 3.1, nous avons revisité le Théorème 2.32 pour pouvoir modéliser des arités non bornées, au prix d'une restriction sur la forme des systèmes décrivant les expressions (système à une dimension). La Proposition 3.14 fournit dans ce contexte un analogue du Théorème 2.60 pour les moments d'ordres supérieurs.

— Afin d'évaluer les conséquences pratiques des résultats précédents, nous avons étudiée une réduction *ad-hoc* sur les expressions régulières. Pour cette réduction, nous avons calculé dans le Théorème 4.19 et le Théorème 4.26 précisément la constante qui borne la taille moyenne des expressions régulières après réduction. Pour un alphabet à deux lettres, nous avons calculé que cette taille est d'environ 77.8, ce qui est confirmé par des simulations (Figure 4.3).

— Au passage, nous avons encadré la proportion asymptotique d'expressions universelles dans la Proposition 4.24. Par exemple, il y a asymptotiquement entre 31% et 46% d'expressions universelles sur un alphabet à deux lettres.

— Enfin, inspirés par les algorithmes de génération aléatoire utilisés en pratique pour la création de benchmarks, nous avons considéré la distribution ABR. Dans le Théorème 5.17 et le Théorème 5.39, nous montrons un comportement plus riche que pour les expressions uniformes, avec l'existence de deux seuils dépendant des probabilités des opérateurs. Nous retrouvons des cas dégénérés, mais aussi quatre autres régimes que nous avons pu observer en pratique sur des simulations (Figure 5.3).

Dans la seconde partie, nous nous sommes intéressés au lien entre classes de fonctions et classes de langages. Ce lien était bien connu entre langages rationnels et séries rationnelles, et entre langages algébriques non ambigus et séries algébriques. Nous l'avons étendu aux séries holonomes en considérant des langages acceptés par des automates de Parikh à pile. Plus précisément :

— Nous avons montré que les langages de Parikh faiblement non ambigus coïncident avec les langages reconnus par des RBCM non ambiguës (Théorème 6.29), et la classe RCM (Théorème 6.41). Nous avons montré que les séries génératrices de ces langages sont holonomes (Théorème 7.45).

— De la même manière que les langages rationnels correspondent précisément aux séries $\mathbb{N}$-rationnelles, et les langages algébriques non ambigus correspondent précisément aux séries $\mathbb{N}$-algébriques, nous avons montré que les séries des langages de Parikh non ambigus correspondent exactement aux diagonales de séries $\mathbb{N}$-rationnelles (Théorème 7.88).

— Ces résultats ont été étendus aux langages algébriques de Parikh faiblement non ambigus, qui correspondent aux version à pile des automates de Parikh faiblement non ambigus, aux RBCM à pile non ambiguës (Proposition 6.60), et à la classe LCM (Théorème 6.69). Les séries de ces langages sont holonomes Théorème 7.45 ; et plus précisément, elles correspondent exactement aux diagonales de séries $\mathbb{N}$-algébriques.

— À l'aide de ce nouveau lien entre séries et langages de Parikh, nous avons développé un outil permettant de prouver l'intrinsèque ambiguïté de certains langages pour ces modèles d'automates (Proposition 7.50 et Proposition 7.51). Nous avons donné un exemple intéressant en soi de langage intrinsèquement faiblement ambigu qui échappe à ces outils (Théorème 7.64).

— Nous avons ensuite analysé une conséquence algorithmique de ce lien entre non-ambiguïté et holonomie des séries génératrices. Dans le Théorème 8.4 et son Corollaire 8.5, nous avons ainsi prouvé que le problème de l'inclusion entre langages de Parikh faiblement non ambigus est décidable avec une complexité doublement exponentielle.

— Au passage, nous avons apporté des éléments de réponse à des questions posées par Philippe Flajolet, quand il a introduit sa méthode appliquant des outils de la théorie de la combinatoire analytique à l'étude de l'intrinsèque ambiguïté. Plus précisément, nous avons utilisé un critère analytique pour établir l'infinie ambiguïté d'un langage algébrique (Corollaire 7.83). Nous avons également proposé aux Théorème 7.24 et Théorème 7.32 deux critères pour montrer l'intrinsèque ambiguïté de certains langages bornés, dont les séries génératrices sont rationnelles.

La prochaine question qui se pose après la première partie de cette thèse portant sur les arbres d'expression aléatoires est la suivante : quelle serait une *bonne* distribution

sur les expressions ? Pour certaines valeurs des paramètres, nos résultats laissent un espoir du côté de la distribution ABR. Afin de préciser quand cette dernière est adaptée à la génération aléatoire, il devient nécessaire de prendre en compte toute la sémantique des expressions représentées. Le travail du chapitre 4 a déjà un peu initié cette approche sur les expressions régulières dans le cas uniforme. La prochaine étape est naturellement d'étudier les expressions LTL suivant la distribution ABR.

Dans la deuxième partie, nous avons vu le très fort lien entre la non-ambiguïté des langages et les séries génératrices. Cette partie constitue une première actualisation des travaux de [Fla87], à la lumière de recherches récentes en théorie des automates et en calcul formel : en effet, depuis l'article de [Fla87], de nombreuses classes de langages ont été introduites en théorie des langages et en vérification ; de même, plusieurs outils efficaces de calcul formel ont aussi été développés du côté des séries, notamment sur les séries holonomes. Toutes ces avancées ouvrent la voie à d'intéressants travaux futurs, visant à exploiter davantage la richesse du lien entre séries génératrices et langages formels.

Bibliographie

[AADM17] Azamat Akhtyamov, Meirav Amram, Miriam Dagan & Artour Mouftahkov (2017) : *Cramer's rule for nonsingular $m \times n$ matrices.* *The Teach. Math.* 20(1), pp. 13–19.

[AM06] Cyril Allauzen & Mehryar Mohri (2006) : *A Unified Construction of the Glushkov, Follow, and Antimirov Automata.* In Rastislav Královič & Paweł Urzyczyn, editors : *Mathematical Foundations of Computer Science 2006*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 110–121.

[AMP99] Steven P. Abney, David A. McAllester & Fernando Pereira (1999) : *Relating Probabilistic Grammars and Automata.* In : *27th Annual Meeting of the Association for Computational Linguistics, ACL 1999*, ACL, pp. 542–549, doi :10.3115/1034678.1034759.

[AN00] Peter R.J. Asveld & Anton Nijholt (2000) : *The inclusion problem for some subclasses of context-free languages.* *Theoretical Computer Science* 230(1), pp. 247–256, doi :https://doi.org/10.1016/S0304-3975(99)00113-9. Available at https://www.sciencedirect.com/science/article/pii/S0304397599001139.

[Ant95] Valentin Antimirov (1995) : *Partial derivatives of regular expressions and finite automata constructions.* In Ernst W. Mayr & Claude Puech, editors : *STACS 95*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 455–466.

[Apo69] T.M. Apostol (1969) : *Calculus. Vol. II : Multi-variable Calculus and Linear Algebra, with Applications to Differential Equations and Probability.* Blaisdell international textbook series, Xerox College Publ.

[AU72] Alfred V. Aho & Jeffrey D. Ullman (1972) : *The Theory of Parsing, Translation, and Compiling.* Prentice-Hall, Inc., USA.

[BB74] Brenda S. Baker & Ronald V. Book (1974) : *Reversal-Bounded Multipushdown Machines.* *J. Comput. Syst. Sci.* 8(3), pp. 315–332, doi :10.1016/S0022-0000(74)80027-9.

[BBLL98] François Bergeron, F Bergeron, Gilbert Labelle & Pierre Leroux (1998) : *Combinatorial species and tree-like structures.* 67, Cambridge University Press.

[BBY10] Jason P. Bell, Stanley Burris & Karen A. Yeats (2010) : *Characteristic Points of Recursive Systems.* *Electr. J. Comb.* 17(1).

[BC17] Jason P. Bell & Shaoshi Chen (2017) : *Power series with coefficients from a finite set. J. Comb. Theory, Ser. A* 151, pp. 241 – 253, doi :10.1016/j.jcta.2017.05.002.

[BCKN20] Alin Bostan, Arnaud Carayol, Florent Koechlin & Cyril Nicaud (2020) : *Weakly-Unambiguous Parikh Automata and Their Link to Holonomic Series.* In Artur Czumaj, Anuj Dawar & Emanuela Merelli, editors : *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, *LIPIcs* 168, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 114 :1–114 :16, doi :10.4230/LIPIcs.ICALP.2020.114. Available at https://doi.org/10.4230/LIPIcs.ICALP.2020.114.

[BCLS18] Alin Bostan, Frédéric Chyzak, Pierre Lairez & Bruno Salvy (2018) : *Generalized Hermite reduction, creative telescoping and definite integration of D-finite functions.* In : *Proceedings of the 2018 ACM International Symposium on Symbolic and Algebraic Computation*, pp. 95–102.

[BD15] Cyril Banderier & Michael Drmota (2015) : *Formulae and Asymptotics for Coefficients of Algebraic Functions. Combinatorics, Probability & Computing* 24(1), pp. 1–53.

[Ber72] Joseph N. Bernstein (1972) : *Analytic continuation of generalized functions with respect to a parameter. Funct. Anal. Appl.* 6(4), pp. 273–28, doi :10.1007/BF01077645.

[BHPS61] Yehoshua Bar-Hillel, M. Perles & E. Shamir (1961) : *On Formal Properties of Simple Phrase Structure Grammars. Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung* 14, pp. 143–172. Reprinted in Y. Bar-Hillel. (1964). *Language and Information : Selected Essays on their Theory and Application*, Addison-Wesley 1964, 116–150.

[BK93] Anne Brüggemann-Klein (1993) : *Regular expressions into finite automata. Theoretical Computer Science* 120(2), pp. 197–213, doi :https://doi.org/10.1016/0304-3975(93)90287-4. Available at https://www.sciencedirect.com/science/article/pii/0304397593902874.

[BK10] Alin Bostan & Manuel Kauers (2010) : *The complete generating function for Gessel walks is algebraic. Proc. Amer. Math. Soc.* 138(9), pp. 3063–3078, doi :10.1090/S0002-9939-2010-10398-2. With an Appendix by Mark van Hoeij.

[BMMR11] Sabine Broda, António Machiavelo, Nelma Moreira & Rogério Reis (2011) : *On The Average State Complexity Of Partial Derivative Automata : An Analytic Combinatorics Approach. International Journal of Foundations of Computer Science* 22(07), pp. 1593–1606, doi :10.1142/S0129054111008908. Available at https://doi.org/10.1142/S0129054111008908.

[BMMR12] Sabine Broda, António Machiavelo, Nelma Moreira & Rogério Reis (2012) : *On the Average Size of Glushkov and Partial derivative Automata. Int. J. Found. Comput. Sci.* 23(5), pp. 969–984, doi :10.1142/S0129054112400400. Available at https://doi.org/10.1142/S0129054112400400.

[BMMR15] Sabine Broda, António Machiavelo, Nelma Moreira & Rogério Reis (2015) : *Average Size of Automata Constructions from Regular Expressions. Bulletin of the EATCS* 116.

[BMMR21] Sabine Broda, António Machiavelo, Nelma Moreira & Rogério Reis (2021) : *On the Uniform Distribution of Regular Expressions. arXiv preprint arXiv :2103.13175.*

[BMS92] Alberto Bertoni, Paolo Massazza & Nicoletta Sabadini (1992) : *Holonomic Generating Functions and Context Free Languages. Int. J. Found. Comput. Sci.* 3(2), pp. 181–191, doi :10.1142/S0129054192000127. Available at https://doi.org/10.1142/S0129054192000127.

[Bou06] Mireille Bousquet-Mélou (2006) : *Rational and algebraic series in combinatorial enumeration.* In : *International Congress of Mathematicians (ICM 2006)*, 3, Eur. Math. Soc., Zürich, pp. 789–826, doi :10.4171/022-3/40.

[BP03] Mireille Bousquet-Mélou & Marko Petkovsek (2003) : *Walks confined in a quadrant are not always D-finite. Theor. Comput. Sci.* 307(2), pp. 257–276, doi :10.1016/S0304-3975(03)00219-6.

[Bro71] William S. Brown (1971) : *On Euclid's algorithm and the computation of polynomial greatest common divisors. J. Assoc. Comput. Mach.* 18, pp. 478–504, doi :10.1145/321662.321664.

[BRW12] Marcello M. Bonsangue, Jan J. M. M. Rutten & Joost Winter (2012) : *Defining Context-Free Power Series Coalgebraically.* In : *Coalgebraic Methods in Computer Science - 11th International Workshop, CMCS 2012, Lecture Notes in Computer Science* 7399, Springer, pp. 20–39, doi :10.1007/978-3-642-32784-1_2.

[BYCDM92] Ricardo Baeza-Yates, Rafael Casas, Josep Díaz & Conrado Martinez (1992) : *On the average size of the intersection of binary trees. SIAM Journal on Computing* 21(1), pp. 24–32.

[Cad13] Michaël Cadilhac (2013) : *Automata with a semilinear constraint.* Ph.D. thesis, Université de Montréal.

[Car08] Olivier Carton (2008) : *Langages formels, calculabilité et complexité.* 28, Vuibert.

[CDG+07] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison & M. Tommasi (2007) : *Tree Automata Techniques and Applications.* Available on : http://www.grappa.univ-lille3.fr/tata. Release October, 12th 2007.

[CDM91] R. Casas, J. Díaz & C. Martínez (1991) : *Statistics on random trees.* In J. Leach, B. Monien & M. Rodríguez-Artalejo, editors : *Proc. of the 18th Int. Col. on Automata, Languages and Programming (ICALP), Lecture Notes in Computer Science* 510, Springer-Verlag, pp. 186–203. Invited lecture.

[CDM93] R. Casas, J. Díaz & C. Martínez (1993) : *Average-case analysis on simple families of trees using a balanced probability model. Theoret. Comput. Sci.* 117, pp. 99–112.

[CFCS90] Rafael Casas, María-Inés Fernández-Camacho & Jean-Marc Steyaert (1990) : *Algebraic simplification in computer algebra : an analysis of*

bottom-up algorithms. Theoretical Computer Science 74(3), pp. 273–298, doi :https://doi.org/10.1016/0304-3975(90)90078-V. Available at https://www.sciencedirect.com/science/article/pii/030439759090078V.

[CFM12a] Michaël Cadilhac, Alain Finkel & Pierre McKenzie (2012) : *Affine Parikh automata. RAIRO - Theor. Inf. and Applic.* 46(4), pp. 511–545, doi :10.1051/ita/2012013.

[CFM12b] Michaël Cadilhac, Alain Finkel & Pierre McKenzie (2012) : *Bounded parikh automata. International Journal of Foundations of Computer Science* 23(08), pp. 1691–1709.

[CFM13] Michaël Cadilhac, Alain Finkel & Pierre McKenzie (2013) : *Unambiguous constrained automata. Int. J. Found. Comput. Sci.* 24(7), pp. 1099–1116, doi :10.1142/S0129054113400339.

[CGM11] Brigitte Chauvin, Danièle Gardy & Cécile Mailler (2011) : *The growing tree distribution on Boolean functions. Proceedings of the Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pp. 45–56, doi :10.1137/1.9781611973013.5. Available at https://epubs.siam.org/doi/abs/10.1137/1.9781611973013.5.

[Che94] Siu-nang Bruce Cheung (1994) : *A theory of automatic language acquisition.* Ph.D. thesis, University of Hong Kong, Pokfulam, Hong Kong SAR. Available at http://hdl.handle.net/10722/34911.

[Cho56] Noam Chomsky (1956) : *Three models for the description of language. IRE Transactions on information theory* 2(3), pp. 113–124.

[Chy94] Frédéric Chyzak (1994) : *Holonomic systems and automatic proofs of identities.* Technical Report RR-2371, INRIA.

[Chy14] Frédéric Chyzak (2014) : *The ABC of Creative Telescoping—Algorithms, Bounds, Complexity.* Ph.D. thesis, Ecole Polytechnique X.

[Cle20] Lorenzo Clemente (2020) : *On the complexity of the universality and inclusion problems for unambiguous context-free grammars. arXiv preprint arXiv :2008.04667.*

[CLO07] David A. Cox, John Little & Donal O'Shea (2007) : *Ideals, Varieties, and Algorithms : An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3/e (Undergraduate Texts in Mathematics).* Springer-Verlag, Berlin, Heidelberg.

[CM17] Giusi Castiglione & Paolo Massazza (2017) : *On a class of languages with holonomic generating functions. Theor. Comput. Sci.* 658, pp. 74–84, doi :10.1016/j.tcs.2016.07.022.

[Col15] Thomas Colcombet (2015) : *Unambiguity in Automata Theory.* In Jeffrey Shallit & Alexander Okhotin, editors : *Descriptional Complexity of Formal Systems - 17th International Workshop, DCFS 2015, Waterloo, ON, Canada, June 25-27, 2015. Proceedings, Lecture Notes in Computer Science* 9118, Springer, pp. 3–18, doi :10.1007/978-3-319-19225-3_1. Available at https://doi.org/10.1007/978-3-319-19225-3_1.

[Com64] Louis Comtet (1964) : *Calcul pratique des coefficients de Taylor d'une fonction algébrique. Enseignement Math. (2)* 10, pp. 267–270.

[Coo72] David C Cooper (1972) : *Theorem proving in arithmetic without multiplication. Mach. Intell.* 7(91-99).

[Cre72] JP Crestin (1972) : *Un langage non ambigu dont le carré est d'ambiguité non bornée.* In : *ICALP*, pp. 377–390.

[CS63] Noam Chomsky & Marcel-Paul Schützenberger (1963) : *The Algebraic Theory of Context-Free Languages. Studies in Logic and the Foundations of Mathematics* 35, Elsevier, doi :10.1016/S0049-237X(08)72023-8.

[Dev86] Luc Devroye (1986) : *A Note on the Height of Binary Search Trees. J. ACM* 33(3), pp. 489–498, doi :10.1145/5925.5930. Available at https://doi.org/10.1145/5925.5930.

[Dev12] Luc Devroye (2012) : *Simulating Size-constrained Galton-Watson Trees. SIAM J. Comput.* 41(1), pp. 1–11, doi :10.1137/090766632. Available at https://doi.org/10.1137/090766632.

[DGV99] Marco Daniele, Fausto Giunchiglia & Moshe Y. Vardi (1999) : *Improved Automata Generation for Linear Temporal Logic.* In Nicolas Halbwachs & Doron Peled, editors : *Computer Aided Verification*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 249–260.

[DL87] J Denef & L Lipshitz (1987) : *Algebraic power series and diagonals. Journal of Number Theory* 26(1), pp. 46–67, doi :https://doi.org/10.1016/0022-314X(87)90095-3. Available at https://www.sciencedirect.com/science/article/pii/0022314X87900953.

[DLLF+16] Alexandre Duret-Lutz, Alexandre Lewkowicz, Amaury Fauchille, Thibaud Michaud, Etienne Renault & Laurent Xu (2016) : *Spot 2.0 — a framework for LTL and ω-automata manipulation.* In : *Proceedings of the 14th International Symposium on Automated Technology for Verification and Analysis (ATVA'16)*, *Lecture Notes in Computer Science* 9938, Springer, pp. 122–129, doi :10.1007/978-3-319-46520-3_8.

[Drm97] Michael Drmota (1997) : *Systems of functional equations. Random Struct. Algorithms* 10(1-2), pp. 103–124.

[Drm09] Michael Drmota (2009) : *Random Trees : An Interplay Between Combinatorics and Probability*, 1st edition. Springer Publishing Company, Incorporated.

[Ele] Georges Elencwajg : *Necessary and sufficient condition for $x^n - y^m$ to be irreducible in $\mathbb{C}[x, y]$.* Mathematics Stack Exchange. Available at https://math.stackexchange.com/q/489703. URL :https ://math.stackexchange.com/q/489703 (version : 2013-09-10).

[EP05] Ioannis Z. Emiris & Victor Y. Pan (2005) : *Improved algorithms for computing determinants and resultants. J. Complexity* 21(1), pp. 43–71, doi :10.1016/j.jco.2004.03.003.

[ES69] Samuel Eilenberg & Marcel-Paul Schützenberger (1969) : *Rational sets in commutative monoids. J. Algebra* 13(2), pp. 173 – 191, doi :10.1016/0021-8693(69)90070-2.

[FB09] Eberhard Freitag & Rolf Busam (2009) : *Complex analysis*, second edition. Universitext, Springer-Verlag, Berlin, doi :10.1007/978-3-540-93983-2. Available at https://doi.org/10.1007/978-3-540-93983-2.

[FGM97] Ph. Flajolet, X. Gourdon & C. Martínez (1997) : *Patterns in random binary search trees. Random Structures & Algorithms* 11(3), pp. 223–245. Available at http://www.lsi.upc.edu/~conrado/research/papers/rsa-fgm97.pdf.

[FGM19] Emmanuel Filiot, Shibashis Guha & Nicolas Mazzocchi (2019) : *Two-way Parikh Automata. CoRR* abs/1907.09362.

[Fla87] Philippe Flajolet (1987) : *Analytic models and ambiguity of context-free languages. Theor. Comput. Sci.* 49(2), pp. 283 – 309, doi :10.1016/0304-3975(87)90011-9.

[FO82] Philippe Flajolet & Andrew M. Odlyzko (1982) : *The Average Height of Binary Trees and Other Simple Trees. J. Comput. Syst. Sci.* 25(2), pp. 171–213, doi :10.1016/0022-0000(82)90004-6.

[FO90] Philippe Flajolet & Andrew M. Odlyzko (1990) : *Singularity Analysis of Generating Functions. SIAM J. Discrete Math.* 3(2), pp. 216–240, doi :10.1137/0403019. Available at https://doi.org/10.1137/0403019.

[FS87] Philippe Flajolet & Jean-Marc Steyaert (1987) : *A Complexity Calculus for Recursive Tree Algorithms. Mathematical Systems Theory* 19(4), pp. 301–331.

[FS09] Philippe Flajolet & Robert Sedgewick (2009) : *Analytic Combinatorics*. Cambridge University Press.

[FSS90] Philippe Flajolet, Paolo Sipala & Jean-Marc Steyaert (1990) : *Analytic Variations on the Common Subexpression Problem*. In : *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, UK, July 16-20, 1990, Proceedings*, *Lecture Notes in Computer Science* 443, Springer, pp. 220–234.

[Gar05] Daniele Gardy (2005) : *Random boolean expressions. Discrete Mathematics & Theoretical Computer Science* DMTCS Proceedings vol. AF, Computational Logic and Applications (CLA '05), pp. 1–36.

[GG66] Seymour Ginsburg & Sheila Greibach (1966) : *Deterministic context free languages. Information and Control* 9(6), pp. 620–648, doi :https://doi.org/10.1016/S0019-9958(66)80019-0. Available at https://www.sciencedirect.com/science/article/pii/S0019995866800190.

[GG74] A. J. Goldstein & R. L. Graham (1974) : *A Hadamard-type bound on the coefficients of a determinant of polynomials. SIAM Rev.* 16(3), pp. 394–395, doi :10.1137/1016065.

[GG10] Hermann Gruber & Stefan Gulan (2010) : *Simplifying Regular Expressions*. In Adrian-Horia Dediu, Henning Fernau & Carlos Martín-Vide, editors : *Language and Automata Theory and Applications*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 285–296.

[Gin66] Seymour Ginsburg (1966) : *The Mathematical Theory of Context-Free Languages*. McGraw-Hill, Inc., USA.

[GP14] Scott Garrabrant & Igor Pak (2014) : *Counting with irrational tiles. arXiv preprint arXiv :1407.8222.*

[GR01] Christopher D. Godsil & Gordon F. Royle (2001) : *Algebraic Graph Theory*. Graduate texts in mathematics, Springer.

[Gre68] Sheila Greibach (1968) : *A note on undecidable properties of formal languages. Mathematical Systems Theory* 2(1), pp. 1–6, doi :10.1007/BF01691341.

[GS64] Seymour Ginsburg & Edwin H Spanier (1964) : *Bounded ALGOL-like languages. Transactions of the American Mathematical Society* 113(2), pp. 333–368.

[GS66] Seymour Ginsburg & Edwin Spanier (1966) : *Semigroups, Presburger formulas, and languages. Pac. J. Math.* 16(2), pp. 285–296.

[GU66] Seymour Ginsburg & Joseph Ullian (1966) : *Ambiguity in Context Free Languages. J. ACM* 13(1), pp. 62–89, doi :10.1145/321312.321318.

[GV89] Ira M Gessel & Xavier Viennot (1989) : *Determinants, paths, and plane partitions. preprint* 132(197.15).

[Haa18] Christoph Haase (2018) : *A survival guide to Presburger arithmetic. ACM SIGLOG News* 5(3), pp. 67–82.

[HMU01] John E Hopcroft, Rajeev Motwani & Jeffrey D Ullman (2001) : *Introduction to automata theory, languages, and computation. Acm Sigact News* 32(1), pp. 60–65.

[Hon96] Juha Honkala (1996) : *On Parikh Slender Languages and Power Series. Journal of Computer and System Sciences* 52(1), pp. 185–190, doi :https://doi.org/10.1006/jcss.1996.0014. Available at https://www.sciencedirect.com/science/article/pii/S0022000096900148.

[HU66] Thomas N. Hibbard & Joseph Ullian (1966) : *The Independence of Inherent Ambiguity From Complementedness Among Context-Free Languages. J. ACM* 13(4), pp. 588–593, doi :10.1145/321356.321366. Available at https://doi.org/10.1145/321356.321366.

[Iba78] Oscar H. Ibarra (1978) : *Reversal-Bounded Multicounter Machines and Their Decision Problems. J. ACM* 25(1), pp. 116–133, doi :10.1145/322047.322058.

[Iba16] Oscar H. Ibarra (2016) : *Visibly Pushdown Automata and Transducers with Counters. Fundam. Inform.* 148(3-4), pp. 291–308, doi :10.3233/FI-2016-1436.

[Ito69] Ryuichi Ito (1969) : *Every Semilinear Set is a Finite Union of Disjoint Linear Sets. J. Comput. Syst. Sci.* 3(2), pp. 221–231, doi :10.1016/S0022-0000(69)80014-0.

[IY03] Lucian Ilie & Sheng Yu (2003) : *Follow automata. Information and Computation* 186(1), pp. 140–162, doi :https://doi.org/10.1016/S0890-5401(03)00090-7. Available at https://www.sciencedirect.com/science/article/pii/S0890540103000907.

[Jun31] Reinwald Jungen (1931) : *Sur les séries de Taylor n'ayant que des singularités algébrico-logarithmiques sur leur cercle de convergence. Comment. Math. Helv.* 3, pp. 266–306, doi :10.1007/BF01601817.

[Kar04] Wong Karianto (2004) : *Parikh automata with pushdown stack* Diplomarbeit Diplomarbeit im Studiengang Informatik.

[Kas78] Masaki Kashiwara (1978) : *On the holonomic systems of linear differential equations. II. Invent. Math.* 49(2), pp. 121–135, doi :10.1007/BF01403082.

[Kau14] Manuel Kauers (2014) : *Bounds for D-finite closure properties.* In : *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation, (ISSAC 2014)*, ACM, New York, pp. 288–295, doi :10.1145/2608628.2608634.

[Kem82] R. Kemp (1982) : *On the number of words in the language* $\{w \in \Sigma^* \mid w = w^R\}^2$. *Discret. Math.* 40(2-3), pp. 225–234, doi :10.1016/0012-365X(82)90123-6. Available at https://doi.org/10.1016/0012-365X(82)90123-6.

[KI92] Takumi Kasai & Shigeki Iwata (1992) : *Some Problems in Formal Language Theory Known as Decidable are Proved EXPTIME Complete* 796, pp. 8–21.

[KNR19] Florent Koechlin, Cyril Nicaud & Pablo Rotondo (2019) : *Uniform Random Expressions Lack Expressivity*. In Peter Rossmanith, Pinar Heggernes & Joost-Pieter Katoen, editors : *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, *LIPIcs* 138, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 51 :1–51 :14.

[KNR20] Florent Koechlin, Cyril Nicaud & Pablo Rotondo (2020) : *On the Degeneracy of Random Expressions Specified by Systems of Combinatorial Equations*. In Natasa Jonoska & Dmytro Savchuk, editors : *Developments in Language Theory - 24th International Conference, DLT 2020, Tampa, FL, USA, May 11-15, 2020, Proceedings*, *Lecture Notes in Computer Science* 12086, Springer, pp. 164–177.

[KNR21] Florent Koechlin, Cyril Nicaud & Pablo Rotondo (2021) : *Simplifications of Uniform Expressions Specified by Systems. International Journal of Foundations of Computer Science*, pp. 1–28.

[KR02] Felix Klaedtke & Harald Rueß (2002) : *Parikh automata and Monadic Second-Order logics with linear cardinality constraints*. Technical Report 177, Freiburg University.

[KR03] Felix Klaedtke & Harald Rueß (2003) : *Monadic Second-Order Logics with Cardinalities*. In : *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003*, *Lecture Notes in Computer Science* 2719, Springer, pp. 681–696, doi :10.1007/3-540-45061-0_54.

[KR21a] Florent Koechlin & Pablo Rotondo (2021) : *Absorbing Patterns in BST-Like Expression-Trees*. In Markus Bläser & Benjamin Monmege, editors : *38th International Symposium on Theoretical Aspects of Computer Science (STACS 2021)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 187, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 48 :1–48 :15, doi :10.4230/LIPIcs.STACS.2021.48. Available at https://drops.dagstuhl.de/opus/volltexte/2021/13693.

[KR21b] Florent Koechlin & Pablo Rotondo (2021) : *Analysis of an Efficient Reduction Algorithm for Random Regular Expressions Based on Universality Detection*. In Rahul Santhanam & Daniil Musatov, editors : *Computer Science - Theory and Applications - 16th International Computer Science Symposium in Russia, CSR 2021, Sochi, Russia, June 28 - July 2, 2021, Proceedings*, *Lecture Notes in Computer Science* 12730, Springer, pp. 206–222, doi :10.1007/978-3-030-79416-3_12. Available at https://doi.org/10.1007/978-3-030-79416-3_12.

[Lip88] Leonard Lipshitz (1988) : *The diagonal of a D-finite power series is D-finite*. *J. Algebra* 113(2), pp. 373 – 378, doi :10.1016/0021-8693(88)90166-4.

[Lip89] Leonard Lipshitz (1989) : *D-finite power series. J. Algebra* 122(2), pp. 353 – 373, doi :10.1016/0021-8693(89)90222-6.

[LS05] Jonathan Lee & Jeffrey Shallit (2005) : *Enumerating Regular Expressions and Their Languages.* In Michael Domaratzki, Alexander Okhotin, Kai Salomaa & Sheng Yu, editors : *Implementation and Application of Automata*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 2–22.

[Mak21] Vladislav Makarov (2021) : *Bounded Languages Described by GF(2)-grammars.* In Nelma Moreira & Rogério Reis, editors : *Developments in Language Theory - 25th International Conference, DLT 2021, Porto, Portugal, August 16-20, 2021, Proceedings*, *Lecture Notes in Computer Science* 12811, Springer, pp. 279–290, doi :10.1007/978-3-030-81508-0_23. Available at https://doi.org/10.1007/978-3-030-81508-0_23.

[Mar91] C. Martínez (1991) : *Average-case analysis of equality of binary trees.* In L. Budach, editor : *Proc. of the 8th Int. Conf. on Fundamentals of Computation Theory (FCT)*, *Lecture Notes in Computer Science* 529, Springer-Verlag, pp. 350–359.

[Mar92] Conrado Martínez (1992) : *Statistics under the BST model.* Ph.D. thesis, University Barcelona.

[Mas93] Paolo Massazza (1993) : *Holonomic Functions and Their Relation to Linearly Constrained Languages. ITA* 27(2), pp. 149–161, doi :10.1051/ita/1993270201491.

[Mas17] Paolo Massazza (2017) : *On the Conjecture* $\mathcal{L}_{\mathrm{DFCM}} \subsetneq \mathrm{RCM}$. In : *Implementation and Application of Automata - 22nd International Conference, CIAA 2017*, *Lecture Notes in Computer Science* 10329, Springer, pp. 175–187, doi :10.1007/978-3-319-60134-2_15.

[Mas18] Paolo Massazza (2018) : *On the Generating Functions of Languages Accepted by Deterministic One-reversal Counter Machines.* In : *Proceedings of the 19th Italian Conference on Theoretical Computer Science, (ICTCS 2018)*, *CEUR Workshop Proceedings* 2243, CEUR-WS.org, pp. 191–202.

[Mis09] Marni Mishna (2009) : *Classifying lattice walks restricted to the quarter plane. Journal of Combinatorial Theory, Series A* 116(2), pp. 460–477.

[Mis19] Marni Mishna (2019) : *Analytic Combinatorics : A Multidimensional Approach.* Chapman and Hall/CRC.

[MM89] A Meir & J.W Moon (1989) : *On an asymptotic method in enumeration. Journal of Combinatorial Theory, Series A* 51(1), pp. 77 – 89.

[MR09] Marni Mishna & Andrew Rechnitzer (2009) : *Two non-holonomic lattice walks in the quarter plane. Theoretical Computer Science* 410(38-40), pp. 3616–3630.

[MS72] A. R. Meyer & L. J. Stockmeyer (1972) : *The Equivalence Problem for Regular Expressions with Squaring Requires Exponential Space.* In : *Proceedings of the 13th Annual Symposium on Switching and Automata Theory (Swat 1972)*, SWAT '72, IEEE Computer Society, USA, pp. 125–129, doi :10.1109/SWAT.1972.29. Available at https://doi.org/10.1109/SWAT.1972.29.

[Nic09] Cyril Nicaud (2009) : *On the Average Size of Glushkov's Automata.* In Adrian-Horia Dediu, Armand-Mihai Ionescu & Carlos Martín-Vide,

editors : *Language and Automata Theory and Applications, Third International Conference, LATA 2009, Tarragona, Spain, April 2-8, 2009. Proceedings*, *Lecture Notes in Computer Science* 5457, Springer, pp. 626–637.

[NPR10] Cyril Nicaud, Carine Pivoteau & Benoît Razet (2010) : *Average Analysis of Glushkov Automata under a BST-Like Model.* In Kamal Lodaya & Meena Mahajan, editors : *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 8, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 388–399, doi :10.4230/LIPIcs.FSTTCS.2010.388. Available at http://drops.dagstuhl.de/opus/volltexte/2010/2880.

[NT04] Michel Nguyên-Thê (2004) : *Distribution of Valuations on Trees.* Theses, Ecole Polytechnique X.

[Ogd68] William Ogden (1968) : *A helpful result for proving inherent ambiguity. Mathematical systems theory* 2(3), pp. 191–194, doi :10.1007/BF01694004. Available at https://doi.org/10.1007/BF01694004.

[Par61] Rohit J Parikh (1961) : *Language generating devices. Quarterly Progress Report* 60, pp. 199–212.

[Par66] Rohit J. Parikh (1966) : *On Context-Free Languages. J. ACM* 13(4), pp. 570–581, doi :10.1145/321356.321364.

[Pet94] Holger Petersen (1994) : *The ambiguity of primitive words.* In : *Annual Symposium on Theoretical Aspects of Computer Science*, Springer, pp. 679–690.

[Pon12] Yann Ponty (2012) : *Rule-weighted and terminal-weighted context-free grammars have identical expressivity. CoRR* abs/1205.0627.

[Pre29] Mojżesz Presburger (1929) : *Uber die Volstandigkeit eines gewissen Systems der Arithmetik ganzer zahlen, in welchem die Addition als einzige Operation hervortritt. C.R. 1er congres des Mathematiciens des pays slaves*, pp. 92–101.

[PSS12] Carine Pivoteau, Bruno Salvy & Michele Soria (2012) : *Algorithms for combinatorial structures : Well-founded systems and Newton iterations. Journal of Combinatorial Theory, Series A* 119(8), pp. 1711–1773, doi :10.1016/j.jcta.2012.05.007. Available at https://hal.inria.fr/inria-00622853.

[Ram29] Frank P. Ramsey (1929) : *On a Problem of Formal Logic. Proc. London Math. Soc. (2)* 30(4), pp. 264–286, doi :10.1112/plms/s2-30.1.264.

[Rom21] Jean-Étienne Rombaldi (2021) : *Analyse matricielle-Cours et exercices résolus.* EDP sciences.

[SCFC94] José-Ramón Sánchez-Couso & María-Inés Fernández-Camacho (1994) : *Average-case analysis of pattern-matching in trees under the BST probability model.* In : *International Colloquium on Automata, Languages, and Programming*, Springer, pp. 178–190.

[SCFC06] José-Ramón Sánchez-Couso & María-Inés Fernández-Camacho (2006) : *Reductions in binary search trees. Theoretical computer science* 355(3), pp. 327–353.

[Sha71] Eliahu Shamir (1971) : *Some inherently ambiguous context-free languages. Inf. Cont.* 18(4), pp. 355 – 363, doi :10.1016/S0019-9958(71)90455-4.

[SI85] Richard Edwin Stearns & Harry B. Hunt III (1985) : *On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. SIAM J. Comput.* 14(3), pp. 598–611, doi :10.1137/0214044.

[SS78] Arto Salomaa & M Soittola (1978) : *Automata : Theoretic Aspects of Formal Power Series.*

[Sta72] Donald F. Stanat (1972) : *A homomorphism theorem for weighted context-free grammars. J. Comput. Syst. Sci.* 6(3), pp. 217–232, doi :10.1016/S0022-0000(72)80003-5.

[Sta80] Richard P. Stanley (1980) : *Differentiably Finite Power Series. Eur. J. Comb.* 1(2), pp. 175 –188, doi :10.1016/S0195-6698(80)80051-5.

[Sta99] Richard P. Stanley (1999) : *Enumerative combinatorics. Vol. 2. Cambridge Studies in Advanced Mathematics* 62, Cambridge University Press, Cambridge, doi :10.1017/CBO9780511609589.

[SZ94] Bruno Salvy & Paul Zimmermann (1994) : *GFUN : A Maple Package for the Manipulation of Generating and Holonomic Functions in One Variable. ACM Trans. Math. Softw.* 20(2), pp. 163–177, doi :10.1145/178365.178368. Available at https://doi.org/10.1145/178365.178368.

[Tak92] Nobuki Takayama (1992) : *An approach to the zero recognition problem by Buchberger algorithm. J. Symbolic Comput.* 14(2-3), pp. 265–282, doi :10.1016/0747-7171(92)90039-7.

[Tau00] Heikki Tauriainen (2000) : *Automated Testing of Büchi Automata Translators for Linear Temporal Logic.* Research Report A66, Helsinki University of Technology, Laboratory for Theoretical Computer Science, Espoo, Finland.

[Tes12] Gerald Teschl (2012) : *Ordinary differential equations and dynamical systems.* 140, American Mathematical Soc.

[VZGG13] Joachim Von Zur Gathen & Jürgen Gerhard (2013) : *Modern computer algebra.* Cambridge university press.

[Was18] W. Wasow (2018) : *Asymptotic Expansions for Ordinary Differential Equations.* Dover Books on Mathematics, Dover Publications. Available at https://books.google.fr/books?id=NQNKDwAAQBAJ.

[Woo15] Kevin Woods (2015) : *Presburger arithmetic, rational generating functions, and quasi-polynomials. The Journal of Symbolic Logic* 80(2), pp. 433–449.

[Yam14] Hiroaki Yamamoto (2014) : *A new finite automaton construction for regular expressions.* In Suna Bensch, Rudolf Freund & Friedrich Otto, editors : *Sixth Workshop on Non-Classical Models for Automata and Applications - NCMA 2014, Kassel, Germany, July 28-29, 2014. Proceedings, books@ocg.at* 304, Österreichische Computer Gesellschaft, pp. 249–264.

A

Annexes

A.1 Théorèmes d'intégration singulière

Lemme 5.21 ([FS09, Theorem VI.9]).
▶ Soit $f(z)$ une fonction analytique sur Ω telle que dans un voisinage de $z = 1$, $f(z) = O((1-z)^s)$, avec $s > -1$. Alors

$$\int_0^z f(\zeta)d\zeta = \int_0^1 f(t)dt + O((1-z)^{s+1})\,. \quad \blacktriangleleft$$

Démonstration. Comme $f(z) = O(|1-z|^s)$ au voisinage de $z = 1$, l'intégrale $\int_0^1 f(t)dt$ sur la ligne $[0,1)$ existe bien et est finie.

Soit $r > 0$ tel que pour tout $\zeta \in \Omega$ vérifiant $|1-\zeta| < r$, on ait l'inégalité $|f(\zeta)| \leqslant M|\zeta - 1|^s$, avec M un constante.

Soit $z \in \Omega$ vérifiant $|z-1| < r$. On peut décomposer $\int_0^z f(\zeta)d\zeta = \int_0^{1-|z-1|} f(t)dt + \int_{1-|z-1|}^z f(\zeta)d\zeta$.

La première partie de la somme se décompose à son tour en $\int_0^{1-|z-1|} f(t)dt = \int_0^1 f(t)dt - \int_{1-|z-1|}^1 f(t)dt$. Comme chaque réel $t \in [1-|z-1|, 1]$ satisfait $|1-t| \leqslant r$, on peut borner $|\int_{1-|z-1|}^1 f(t)dt| \leqslant M \int_{1-|z-1|}^1 (1-t)^s dt = \frac{M}{s+1}|z-1|^{s+1}$.

La seconde intégrale se borne en intégrant le long du chemin γ qui relie $1-|z-1|$ à z en suivant une portion de cercle de centre 1 et de rayon $|z-1|$. En effet, pour chaque $\zeta \in \gamma$, $|1-\zeta| = |1-z|$, si bien que $|f(\zeta)| \leqslant M|z-1|^s$ et $|\gamma| \leqslant 2\pi|z-1|$. Ainsi $|\int_{1-|z-1|}^z f(\zeta)d\zeta| \leqslant 2\pi M|z-1|^{s+1}$. ■

Le Lemme suivant est une adaptation du précédent dans le cas où la fonction est bornée par un terme en log :

Lemme 5.22 ([FS09, Theorem VI.9]).
▶ Soit $f(z)$ une fonction analytique sur Ω telle que dans un voisinage de $z = 1$, $f(z) = O((1-z)^{-1}(-\log(1-z))^{-\beta})$, avec $\beta > 1$. Soit $a_0 > 0$. Alors

$$\int_{a_0}^z f(\zeta)d\zeta = \int_{a_0}^1 f(t)dt + O((-\log(1-z))^{1-\beta})\,. \quad \blacktriangleleft$$

Démonstration. Comme $f(z) = O((1-z)^{-1}(-\log(1-z))^{-\beta})$ au voisinage de $z=1$, avec $\beta > 1$, l'intégrale $\int_0^1 f(t)dt$ sur la ligne $[0,1)$ existe et est finie.

Soit $r > 0$ tel que pour tout $\zeta \in \Omega$ vérifiant $|1-\zeta| < r$, on ait l'inégalité $|f(\zeta)| \leqslant M|1-\zeta|^{-1}| - \log(1-\zeta)|^{-\beta}$, avec M une constante.

Soit $z \in \Omega$ tel que $|z-1| < r$. On peut décomposer $\int_{a_0}^{z} f(\zeta)d\zeta = \int_{a_0}^{1-|z-1|} f(t)dt + \int_{1-|z-1|}^{z} f(\zeta)d\zeta$.

La première partie se décompose à son tour $\int_{a_0}^{1-|z-1|} f(t)dt = \int_{a_0}^{1} f(t)dt - \int_{1-|z-1|}^{1} f(t)dt$. Comme tout réel $t \in [1-|z-1|, 1]$ vérifie $|1-t| \leqslant r$, on peut borner l'intégrale $|\int_{1-|z-1|}^{1} f(t)dt| \leqslant M \int_{1-|z-1|}^{1} M(1-t)^{-1}(-\log(1-t))^{-\beta})dt = \frac{M}{\beta-1}\left(\log \frac{1}{|1-z|}\right)^{1-\beta}$.

La seconde intégrale se borne en intégrant le long du chemin γ reliant $1-|z-1|$ à z en suivant une portion de cercle de centre 1 et de rayon $|z-1|$. Comme pour tout $\zeta \in \gamma$, on a l'égalité $|1-\zeta| = |1-z|$, alors $|f(\zeta)| \leqslant M|z-1|^{-1}\frac{1}{|\log(\frac{1}{1-\zeta})|^{\beta}}$. Comme de plus $|\log(\frac{1}{1-\zeta})| \geqslant |\log(\frac{1}{|1-\zeta|})| = |\log(\frac{1}{|1-z|})|$ et $|\gamma| \leqslant 2\pi|z-1|$, on en déduit $|\int_{1-|z-1|}^{z} f(\zeta)d\zeta| = O(\frac{1}{|\log|1-z||^{\beta}}) = O\left(\left(\log \frac{1}{|1-z|}\right)^{1-\beta}\right)$.

On conclut en utilisant le fait que $O(\log(|1-z|)) = O(\log(1-z))$. ∎

Lemme 5.23 (Intégration des équivalents).
▶ Soit $f(z)$ une fonction analytique sur Ω telle qu'au voisinage de $z=1$, $f(z) \sim_{z=1} (1-z)^s$. Alors :

a. si $s > -1$, l'intégrale converge et $\int_0^z f(\zeta)d\zeta \to_{z=1} \int_0^1 f(\zeta)d\zeta$;

b. si $s < -1$, $\int_0^z f(\zeta)d\zeta \sim_{z=1} \frac{1}{-s-1}(1-z)^{s+1}$;

c. si $s = 1$, $\int_0^z f(\zeta)d\zeta \sim_{z=1} -\log(1-z)$. ◀

Démonstration. Le premier cas (cas convergent) a déjà été démontré au-dessus. On s'intéresse donc au cas des intégrales divergentes. On rappelle que dans le cas réel, si $f(x) \sim_{x=1} g(x)$ et si $g(x)$ est positive au voisinage de 1 et g n'est pas intégrable en $x=1$, alors les intégrales partielles sont équivalentes : $\int_0^x f(t)dt \sim \int_0^x g(t)dt$.

b. Soit $\varepsilon > 0$. Soit η tel que si $|z-1| < \eta$, alors on ait à la fois $|f(z) - (1-z)^s| \leqslant \varepsilon|1-z|^s$ et $\left|\int_0^{1-|z-1|} f(t) - (1-t)^s dt\right| \leqslant \varepsilon\frac{|1-z|^{s+1}}{-s-1}$. Alors par le même découpage que dans les lemmes précédents, on obtient la majoration :

$$\left|\int_0^z (f(w) - (1-w)^s)dw\right| \leqslant \varepsilon\frac{|1-z|^{s+1}}{-s-1} + 2\varepsilon\pi|1-z|^{s+1}$$

À découpage des epsilons près, on a montré que $f(z) - \frac{(1-z)^{s+1}}{-s-1} = o(\frac{(1-z)^{s+1}}{-s-1})$, autrement dit que $f(z) \sim_{z=1} \frac{(1-z)^{s+1}}{-s-1}$.

c. Soit $\varepsilon > 0$. Soit η tel que si $|z-1| < \eta$, alors on ait à la fois $|f(z) - (1-z)^{-1}| \leqslant \varepsilon|1-z|^{-1}$ et $\left|\int_0^{1-|z-1|} f(t) - (1-t)^{-1}dt\right| \leqslant \varepsilon|-\log(|1-z|)|$. Alors par le même découpage que dans les lemmes précédents, on obtient la majoration :

$$\left|\int_0^z (f(w) - (1-w)^{-1})dw\right| \leqslant \varepsilon|-\log(|1-z|)| + 2\pi|1-z|\varepsilon|1-z|^{-1}$$

$\Re(\log(z)) = \log(|z|)$

Comme $|-\log(|1-z|)| \leqslant |-\log(1-z)|$, à découpage des epsilons près, on a montré que $f(z) + \log(1-z) = o(-\log(1-z))$, autrement dit que $f(z) \sim_{z=1} -\log(1-z)$.

■

A.2 Équations différentielles linéaires

On rappelle dans cette section les propositions utiles pour résoudre et étudier les équations différentielles linéaires.

Proposition A.1 (Existence, unicité et analycité [Was18, Thm. 2.2]).
▶ L'équation différentielle linéaire :

$$u^{(n)}(z) + a_1(z)u^{(n-1)}(z) + \ldots + a_n(z)u(z) = g(z)$$

avec $a_1(z), \ldots, a_n(z), g(z)$ analytiques sur un ouvert simplement connexe Ω, admet exactement une solution $u(z)$ telle que

En pratique Ω sera un ouvert étoilé.

$$u^{(j)}(a) = \alpha_j \quad \text{pour } j = 0, 1, \ldots, n-1$$

où a est un point quelconque de Ω, et les α_j sont des nombres complexes. De plus $v(z)$ est analytique sur Ω tout entier. ◀

Proposition A.2 ([FB09, Theorem II.2.7]).
▶ Soit $g(z)$ une fonction analytique sur un Ω ouvert étoilé centré en $z = 0$. Alors l'intégrale $h(z) = \int_0^z g(\zeta)d\zeta$ définit une fonction analytique sur Ω, qui est une primitive de $g(z)$ et vérifie $h(0) = 0$. ◀

Par exemple, $\Omega = \mathbb{C} \setminus [1, +\infty)$ convient.

Proposition A.3 (Équation linéaire du premier ordre [Was18, Thm. 2.1],[Apo69, Th. 6.1]).
▶ On considère l'équation différentielle linéaire : $U'(z) = f(z) + g(z)U(z)$ où f, g sont des fonctions analytiques sur un ouvert étoilé Ω centré en 0. Alors il existe une unique solution $U(z)$ satisfaisant $U(0) = u_0$. De plus $U(z)$ est analytique sur Ω, et vérifie :

$$U(z) = \exp\left(\int_0^z g(\zeta)d\zeta\right)\left(u_0 + \int_0^z f(\zeta)\exp\left(-\int_0^\zeta g(w)dw\right)d\zeta\right),$$

pour tout $z \in \Omega$. ◀

La proposition suivante permet de résoudre certaines équations dont les coefficients $f(z)$ et $g(z)$ ont une singularité en $z = 0$, alors que les conditions initiales pour la solution sont justement données en $z = 0$:

Corollaire A.4.
▶ On considère l'équation différentielle linéaire suivante :

$$U'(z) = -u_1 + u_1 cz + z^2 f(z) + \left(\tfrac{2}{z} - c + zg(z)\right)U(z)$$

où f, g sont des fonctions analytiques sur un ouvert étoilé Ω centré en 0, et $c, u_1, u_2 \in \mathbb{R}$. Alors il existe une unique solution $U(z)$ vérifiant $U(0) = 0, U'(0) = u_1, U''(0) = 2u_2$. De plus $U(z)$ est analytique sur Ω, et est donnée par la formule :

$$U(z) = u_1 z + z^2 I(z) \cdot \left(u_2 + \int_0^z \left(f(\zeta) + u_1 g(\zeta)\right) I(\zeta)^{-1} d\zeta\right),$$

où $I(z) := \exp\left(-cz + \int_0^z \zeta g(\zeta)d\zeta\right)$, pour tout $z \in \Omega$. ◀

Démonstration. La fonction $\tilde{U}(z) := \frac{U(z)-u_1 z}{z^2}$ satisfait $\tilde{U}'(z) = f(z) + u_1 g(z) + (-c + zg(z))\tilde{U}(z)$, avec $\tilde{U}(0) = u_2$. Il suffit alors d'appliquer la Proposition A.3. ■

Le corollaire suivant est simplement le cas particulier $c = u_1 = u_2 = 0$.

Corollaire A.5 .
▶ On considère l'équation $U'(z) = F(z) + \left(\frac{2}{z} + G(z)\right) U(z)$, avec $G(z) = zg(z)$ et $F(z) = z^2 f(z)$, où $f(z)$ et $g(z)$ sont analytiques sur un ouvert Ω, étoilé, centré en 0.

Alors l'unique solution $U(z)$ vérifiant $U(0) = U'(0) = U''(0) = 0$ est :

$$U(z) = z^2 \exp\left(\int_0^z G(\zeta)d\zeta\right) \cdot \int_0^z F(\zeta) \cdot \zeta^{-2} \exp\left(-\int_0^\zeta G(w)dw\right) d\zeta \,.$$

◀

Remarque A.6 .
▶ Nous avons besoin de deux conditions initiales supplémentaires par rapport à la Proposition A.3. ◀

B

Code Maple

Cette annexe rassemble le code Maple utilisé dans le chapitre 4.

```
> restart;
> ######################################################################
> ##                    Parametrizable section                        ##
> ######################################################################
> ## k = number of letters
> k:=2
```

$$k := 2$$

```
> ## Digits = precision for numerical values
> Digits:=10
```

$$Digits := 10$$

```
> ## p = size of the tree representing the universal language
> p:=2*k
```

$$p := 4$$

```
> ### to solve exactly the triangular system for very small k (exact
solutions are big and long to compute when k increases)
> exact:= false:


> ######################################################################
> ##                       custom functions                           ##
> ######################################################################
> myevalf:= proc(x) if exact then return x; else return evalf(x,Digits);
fi;end:
> ######################################################################
> ##                    creation of the system                        ##
> ######################################################################

> indic:=proc(x):  if evalb(x) then return 1 else return 0 end
if;end:
> T[0]:=add(binomial(k,i)*T[i,0],i=0..k)
```

$$T_0 := T_{0,0} + 2\,T_{1,0} + T_{2,0}$$

```
> T[1]:=add(binomial(k,i)*T[i,1],i=0..k)
```

$$T_1 := T_{0,1} + 2\,T_{1,1} + T_{2,1}$$

```
> system_of_eq_no_eps:=i->z*indic(i=1)+z*indic(i=0)*(T[0])^2
+z*add(add(binomial(i,a)*binomial(a,j)*T[a,0]*T[i+j-a,0], j=0..a),a=0..i)
+2*z*T[1]*T[i,0]:
> system_of_eq_eps:=i->z*indic(i=0)+z*T[i,0]+z*T[i,1]
+2*z*add(add(binomial(i,a)*binomial(a,j)*T[a,1]*T[i+j-a,1], j=0..a),a=0..i)
+2*z*add(add(binomial(i,a)*binomial(a,j)*T[a,1]*T[i+j-a,0], j=0..a),a=0..i):
```

```
> for i from 0 to k do:
> phi[i,0]:=(system_of_eq_no_eps(i));
> phi[i,1]:=(system_of_eq_eps(i));
> od;
```

$$\phi_{0,0} := z\left(T_{0,0}+2\,T_{1,0}+T_{2,0}\right)^2+z{T_{0,0}}^2+2\,z\left(T_{0,1}+2\,T_{1,1}+T_{2,1}\right)T_{0,0}$$

$$\phi_{0,1} := 2\,z T_{0,0}T_{0,1}+2\,z{T_{0,1}}^2+z T_{0,0}+z T_{0,1}+z$$

$$\phi_{1,0} := z+z\left(2\,T_{0,0}T_{1,0}+{T_{1,0}}^2\right)+2\,z\left(T_{0,1}+2\,T_{1,1}+T_{2,1}\right)T_{1,0}$$

$$\phi_{1,1} := 2\,z\left(2\,T_{0,1}T_{1,1}+{T_{1,1}}^2\right)+2\,z\left(T_{0,0}T_{1,1}+T_{0,1}T_{1,0}+T_{1,0}T_{1,1}\right)+z T_{1,0}+z T_{1,1}$$

$$\phi_{2,0} := z\left(2\,T_{0,0}T_{2,0}+2\,{T_{1,0}}^2+4\,T_{1,0}T_{2,0}+{T_{2,0}}^2\right)+2\,z\left(T_{0,1}+2\,T_{1,1}+T_{2,1}\right)T_{2,0}$$

$$\begin{aligned} t\phi_{2,1} :=& 2\,z\left(2\,T_{0,1}T_{2,1}+2\,{T_{1,1}}^2+4\,T_{1,1}T_{2,1}+{T_{2,1}}^2\right)\\ &+2\,z\left(T_{0,0}T_{2,1}+T_{0,1}T_{2,0}+2\,T_{1,0}T_{1,1}+2\,T_{1,0}T_{2,1}+2\,T_{1,1}T_{2,0}+T_{2,0}T_{2,1}\right)\\ &+z T_{2,0}+z T_{2,1}\end{aligned}$$

```
> T[L]:=add(binomial(k,i)*(T[i,1]+T[i,0]),i=0..k-1)+R+T[G]+T[k,0]
```

$$T_L := T_{0,1}+T_{0,0}+2\,T_{1,1}+2\,T_{1,0}+R+T_G+T_{2,0}$$

```
> T[1]:=add(binomial(k,i)*T[i,1],i=0..k-1)+R+T[G]
```

$$T_1 := T_{0,1}+2\,T_{1,1}+R+T_G$$

```
> Phi[G]:=subs(T[k,1]=T[G],phi[k,1]-(z*T[k,0]+z*T[k,1]))
```

$$\Phi_G := 2\,z\left({T_G}^2+2\,T_G T_{0,1}+4\,T_G T_{1,1}+2\,{T_{1,1}}^2\right)+2\,z\left(T_G T_{0,0}+2\,T_G T_{1,0}+T_G T_{2,0}+T_{0,1}T_{2,0}+2\,T_{1,0}T_{1,1}+2\,T_{1,1}T_{2,0}\right)$$

```
> Phi[R]:=z*(T[k,0]+T[G]+R+R*T[L]+(T[L]-R)*R+R*T[1] + (T[1]-R)*R)
```

$$\begin{aligned}\Phi_R :=& z\Big(T_{2,0}+T_G+R+R\left(T_{0,1}+T_{0,0}+2\,T_{1,1}+2\,T_{1,0}+R+T_G+T_{2,0}\right)\\ &+\left(T_{0,1}+T_{0,0}+2\,T_{1,1}+2\,T_{1,0}+T_G+T_{2,0}\right)R+R\left(T_{0,1}+2\,T_{1,1}+R+T_G\right)+\left(T_{0,1}+2\,T_{1,1}+T_G\right)R\Big)\end{aligned}$$

```
> for i from 0 to k do:
> Phi[i,0]:=subs(T[k,1]=R+T[G],phi[i,0]);
> Phi[i,1]:=subs(T[k,1]=R+T[G],phi[i,1]);
> od;
```

$$\Phi_{0,0} := z\left(T_{0,0}+2\,T_{1,0}+T_{2,0}\right)^2+z{T_{0,0}}^2+2\,z\left(T_{0,1}+2\,T_{1,1}+R+T_G\right)T_{0,0}$$

$$\Phi_{0,1} := 2\,z T_{0,0}T_{0,1}+2\,z{T_{0,1}}^2+z T_{0,0}+z T_{0,1}+z$$

$$\Phi_{1,0} := z+z\left(2\,T_{0,0}T_{1,0}+{T_{1,0}}^2\right)+2\,z\left(T_{0,1}+2\,T_{1,1}+R+T_G\right)T_{1,0}$$

$$\Phi_{1,1} := 2\,z\left(2\,T_{0,1}T_{1,1}+{T_{1,1}}^2\right)+2\,z\left(T_{0,0}T_{1,1}+T_{0,1}T_{1,0}+T_{1,0}T_{1,1}\right)+z T_{1,0}+z T_{1,1}$$

$$\Phi_{2,0} := z\left(2\,T_{0,0}T_{2,0}+2\,{T_{1,0}}^2+4\,T_{1,0}T_{2,0}+{T_{2,0}}^2\right)+2\,z\left(T_{0,1}+2\,T_{1,1}+R+T_G\right)T_{2,0}$$

$$\begin{aligned}\Phi_{2,1} :=& 2\,z\left(2\,T_{0,1}\left(R+T_G\right)+2\,{T_{1,1}}^2+4\,T_{1,1}\left(R+T_G\right)+\left(R+T_G\right)^2\right)\\ &+2\,z\left(\left(R+T_G\right)T_{0,0}+T_{0,1}T_{2,0}+2\,T_{1,0}T_{1,1}+2\,\left(R+T_G\right)T_{1,0}+2\,T_{1,1}T_{2,0}+\left(R+T_G\right)T_{2,0}\right)\\ &+z T_{2,0}+z\left(R+T_G\right)\end{aligned}$$

```
> ########################################################################
> ##                    values at rho of OGFs                           ##
```

```
> ######################################################################
> hval:='hval'
```

$$hval := hval$$

```
> #### generating series of every words
> eq[L] := -L + (k+1)*z + z*L + 2*z*L^2
```

$$eq_L := 2\,L^2 z + Lz - L + 3\,z$$

```
> sol[L]:= solve(eq[L],L)[2]
```

$$sol_L := -1/4\,\frac{z - 1 + \sqrt{-23\,z^2 - 2\,z + 1}}{z}$$

```
> rho:=min(map(abs,[solve(-8*k*z^2-7*z^2-2*z+1,z)]))
```

$$\rho := -1/23 + 2/23\,\sqrt{6}$$

```
> hval[L]:= myevalf(simplify(subs(z=rho,sol[L])))
```

$$hval_L := 1.224744871$$

```
> #### generating series of trees recognizing epsilon
> eq[1]:= T=z+z*L+2*z*T*L-z*T^2+z*T^2
```

$$eq_1 := T = 2\,LTz + Lz + z$$

```
> sol[1]:=simplify(subs(L=sol[L],solve(eq[1],T)))
```

$$sol_1 := \frac{3\,z + 1 - \sqrt{-23\,z^2 - 2\,z + 1}}{2\,z + 2 + 2\,\sqrt{-23\,z^2 - 2\,z + 1}}$$

```
> hval[1]:=myevalf(simplify(subs(z=rho, sol[1])))
```

$$hval_1 := 0.6449489742$$

```
> #### value at rho of the generating series of trees not recognizing
epsilon
> hval[0]:=simplify(hval[L]-hval[1])
```

$$hval_0 := 0.5797958968$$

```
> ##### auxilary function to write the system in triangular form
> is_positive:=x->evalb(evalf(x)>0):
> system_of_eq_no_eps_rho:=i->rho*indic(i=1)+rho*indic(i=0)*(hval[0])^2
+rho*add(add(binomial(i,a)*binomial(a,j)*hval[a,0]*hval[i+j-a,0],
j=0..a),a=0..i)+2*rho*hval[1]*hval[i,0]:
> system_of_eq_eps_rho:=i->rho*indic(i=0)
+2*rho*add(add(binomial(i,a)*binomial(a,j)*hval[a,1]*hval[i+j-a,1],
j=0..a),a=0..i)+2*rho*add(add(binomial(i,a)*binomial(a,j)*hval[a,1]*hval[i+j-a,0],
j=0..a),a=0..i)+rho*hval[i,0]+rho*hval[i,1]:
> #### resolution of the triangular system
```

```
> if assigned(hval[1,0]) then print("values already computed");
else:
> for i from 0 to k do:
> hval[i,0]:=simplify(min(select(is_positive,
[solve(-hval[i,0]+system_of_eq_no_eps_rho(i))])));
> od:
> for i from 0 to k do:
> hval[i,1]:=simplify(min(select(is_positive,
[solve(-hval[i,1]+system_of_eq_eps_rho(i))])));
> od:end;
> #### Value at rho of the OGFs of every class with no epsilon:
> evalf(seq(hval[i,0],i=0..k))
```

$$0.07412715846,\ 0.2367373896,\ 0.03219395926$$

```
> #### Value at rho of the OGFs of every class with no epsilon:
(the last value is the sum for R and TG)
> evalf(seq(hval[i,1],i=0..k))
```

$$0.2530566229,\ 0.1176880417,\ 0.1565162679$$

```
> #### Value at rho of the OGF of R
> hval[R]:=(min(select(is_positive,[solve(-R+rho*(hval[k,0]+hval[k,1]+R*hval[L]
+ R*(hval[L]-R) + R*hval[1] + R*(hval[1]-R))])));
> evalf(hval[R])
```

$$0.08126620721$$

```
> #### Value at rho of the OGF of TG
> hval[G]:=hval[k,1]-hval[R]:
> evalf(hval[G])
```

$$0.07525006069$$

```
> ######################################################################
> ##                      probas of each class                        ##
> ######################################################################

> #          ***  we solve the linear system by hand ***           #

> gval:='gval':
> gval[L]:=simplify(1/(4*rho)*sqrt((8*k+7)*rho*(rho+(1+2*sqrt(2+2*k))/(8*k+7)))):
evalf(%)
```

$$1.900624222$$

```
> gval[1]:=simplify(gval[L]*4*rho*(1+2*rho)/(1+rho)^2):evalf(%)
```

$$1.261704146$$

```
> gval[0]:=gval[L]-gval[1]:
```

```
> system_of_eq_differentiated_noeps:=i->diff(TT[i,0](z)=rho*indic(i=1)
+rho*indic(i=0)*(TT[0](z))^2
+rho*add(add(binomial(i,a)*binomial(a,j)*TT[a,0](z)*TT[i+j-a,0](z),
j=0..a),a=0..i)+2*rho*TT[1](z)*TT[i,0](z),z):
> if assigned(gval[0,0]) then print("values already computed");
else:  for i from 0 to k do:
> diffeq:=system_of_eq_differentiated_noeps(i):
> substitution:={diff(TT[0](z), z)=2*rho*gval[0], diff(TT[1](z),
z)=2*rho*gval[1]} union {seq(diff(TT[j, 0](z), z)=2*rho*gval[j,0],j=0..i)}:
> diffeq:=subs(substitution,diffeq):
> substitution:={TT[0](z)=hval[0], TT[1](z)=hval[1]} union {seq(TT[j,
0](z)=hval[j,0],j=0..i)}:
> diffeq:=subs(substitution,diffeq):
> gval[i,0]:=solve(diffeq):
> od:end:
> ### Asymptotics probabilities for classes with no epsilon
> evalf(seq(gval[j,0]/gval[L],j=0..k))
```

$$0.1094480504,\ 0.09182304304,\ 0.04306914559$$

```
> system_of_eq_differentiated_eps:=i->diff(TT[i,1](z)=rho*indic(i=0)
+2*rho*add(add(binomial(i,a)*binomial(a,j)*TT[a,1](z)*TT[i+j-a,1](z),
j=0..a),a=0..i)+2*rho*add(add(binomial(i,a)*binomial(a,j)*TT[a,1](z)*TT[i+j-a,0](z),
j=0..a),a=0..i)+rho*TT[i,0](z)+rho*TT[i,1](z),z):
> if assigned(gval[0,1]) then print("values already computed");
else:for i from 0 to k do:
> diffeq:=system_of_eq_differentiated_eps(i):
> substitution:={diff(TT[0](z), z)=2*rho*gval[0], diff(TT[1](z),
z)=2*rho*gval[1]} union {seq(diff(TT[j, 0](z), z)=2*rho*gval[j,0],j=0..i)}
union {seq(diff(TT[j, 1](z), z)=2*rho*gval[j,1],j=0..i)}:
> diffeq:=subs(substitution,diffeq):
> substitution:={TT[0](z)=hval[0], TT[1](z)=hval[1]} union {seq(TT[j,
0](z)=hval[j,0],j=0..i)} union {seq(TT[j, 1](z)=hval[j,1],j=0..i)}:
> diffeq:=subs(substitution,diffeq):
> gval[i,1]:=solve(diffeq):
> od:end:
> ### Asymptotics probabilities for classes with epsilon (the last
being the sum for R and TG)
> evalf(seq(gval[j,1]/gval[L],j=0..k))
```

$$0.04409287595,\ 0.08134641384,\ 0.4570510135$$

```
> #### Asymptotic probability of being fully reducible
> gval[R]:=solve(X=rho*(gval[k,0]+gval[k,1]+ X*hval[L] + hval[R]*gval[L]
+ hval[R]*(gval[L]-X) + X*(hval[L]-hval[R]) + X*hval[1] + hval[R]*gval[1]
+ X*(hval[1]-hval[R]) + hval[R]*(gval[1]-X))):
> evalf(gval[R]/gval[L])
```

$$0.3101223526$$

```
> #### Asymptotic probability of being in TG
> gval[G]:=gval[k,1]-gval[R]:
> evalf(gval[G]/gval[L])
```

$$0.1469286607$$

```
> ######################################################################
> ##           Resolution of the sytem for Qytilde(rho)               ##
> ######################################################################

> indetermined_tilde:=[seq(op([T[i,0],T[i,1]]),i=0..k-1),T[k,0],T[G]]
```

$$indetermined\_tilde := [T_{0,0}, T_{0,1}, T_{1,0}, T_{1,1}, T_{2,0}, T_G]$$

```
> Phitilde:=<seq(Phi[op(x)], x in indetermined_tilde)>
```

$$Phitilde := rtable\Big(1\ldots 6, [z\left(T_{0,0}+2\,T_{1,0}+T_{2,0}\right)^2 + z{T_{0,0}}^2 + 2\,z\left(T_{0,1}+2\,T_{1,1}+R+T_G\right)T_{0,0},$$
$$2\,zT_{0,0}T_{0,1} + 2\,z{T_{0,1}}^2 + zT_{0,0} + zT_{0,1} + z,$$
$$z + z\left(2\,T_{0,0}T_{1,0} + {T_{1,0}}^2\right) + 2\,z\left(T_{0,1}+2\,T_{1,1}+R+T_G\right)T_{1,0},$$
$$2\,z\left(2\,T_{0,1}T_{1,1} + {T_{1,1}}^2\right) + 2\,z\left(T_{0,0}T_{1,1} + T_{0,1}T_{1,0} + T_{1,0}T_{1,1}\right) + zT_{1,0} + zT_{1,1},$$
$$z\left(2\,T_{0,0}T_{2,0} + 2\,{T_{1,0}}^2 + 4\,T_{1,0}T_{2,0} + {T_{2,0}}^2\right) + 2\,z\left(T_{0,1}+2\,T_{1,1}+R+T_G\right)T_{2,0},$$
$$2\,z\left({T_G}^2 + 2\,T_GT_{0,1} + 4\,T_GT_{1,1} + 2\,{T_{1,1}}^2\right) + 2\,z\left(T_GT_{0,0} + 2\,T_GT_{1,0} + T_GT_{2,0} + T_{0,1}T_{2,0} + 2\,T_{1,0}T_{1,1} + 2\,T_{1,1}T_{2,0}\right)],$$
$$subtype = Vector_{column}\Big)$$

```
> Jacytilde:=VectorCalculus:-Jacobian(Phitilde,(indetermined_tilde))
```

$$Jacytilde := rtable\left(1\ldots 6, 1\ldots 6, [[2\,z\left(T_{0,0}+2\,T_{1,0}+T_{2,0}\right), subtype = Matrix\right)$$

```
> DRphi:=<diff(convert(Phitilde,list),R)>:
> Identity:=LinearAlgebra:-IdentityMatrix(2*(k+1)):
> substitution:={z=rho, R=hval[R], seq(x=hval[op(x)], x in indetermined_tilde)}:
> vect:=evalf(subs(substitution,Phitilde+p*R*DRphi))
```

$$vect := rtable(1\ldots 6,$$
$$[0.08229675198, 0.2530566230, 0.2628283428, 0.1176880417, 0.03574207258, 0.07525006068],$$
$$subtype = Vector_{column})$$

```
> M:=evalf(subs(substitution,Identity-Jacytilde ))
```

$$M := rtable(1\ldots 6, 1\ldots 6, [$$
$$[0.5596282265, -0.02513219740, -0.3931499668, -0.05026439481, -0.1965749833, -0.02513219740],$$
$$[-0.2553175934, 0.6337534627, 0.0, 0.0, 0.0, 0.0],$$
$$[-0.08026384574, -0.08026384574, 0.6759393644, -0.1605276915, 0.0, -0.08026384574],$$
$$[-0.03990115309, -0.1600661519, -0.2952187464, 0.4736873115, 0.0, 0.0],$$
$$[-0.01091509450, -0.01091509450, -0.1823578804, -0.02183018899, 0.5847604237, -0.01091509450],$$
$$[-0.02551290809, -0.06194091066, -0.1308281223, -0.4440141251, -0.1911119604, 0.4212010969]],$$
$$subtype = Matrix)$$

```
> Qytilde_vect:=LinearAlgebra:-LinearSolve(M,vect):
> ### Values of Qytilde at z=rho
```

```
> for i from 1 to nops(indetermined_tilde) do:
> hvaltilde[op(indetermined_tilde[i])]:= Qytilde_vect[i];
> od;
```

$$hvaltilde_{0,0} := 1.79358085271177$$
$$hvaltilde_{0,1} := 1.12187058805744$$
$$hvaltilde_{1,0} := 1.50475089533306$$
$$hvaltilde_{1,1} := 1.71644489322149$$
$$hvaltilde_{2,0} := 0.705795987963761$$
$$hvaltilde_G := 3.04931489034816$$

```
> ######################################################################
> ##                 Computation of g_Qytilde(rho)                    ##
> ######################################################################
> indetermined_Phi:=[seq(op([T[i,0],T[i,1]]),i=0..k-1),T[k,0],T[G],R]
```

$$indetermined\_Phi := [T_{0,0}, T_{0,1}, T_{1,0}, T_{1,1}, T_{2,0}, T_G, R]$$

```
> substitution:={z=rho, seq(x=hval[op(x)], x in indetermined_Phi)}
```

$$substitution := \{R = 0.08126620721, z = -1/23 + 2/23\sqrt{6}, T_G = 0.07525006069, T_{0,0} = 0.07412715846, T_{0,1} = 0.2530566229, T_{1,0} = 0.2367373896, T_{1,1} = 0.1176880417, T_{2,0} = 0.03219395926\}$$

```
> substitution_g:={z=rho, seq(x=gval[op(x)], x in indetermined_Phi)}
```

$$substitution\_g := \{R = 0.5894260557, z = -1/23 + 2/23\sqrt{6}, T_G = 0.2792561716, T_{0,0} = 0.2080196158, T_{0,1} = 0.08380398812, T_{1,0} = 0.1745210998, T_{1,1} = 0.1546089646, T_{2,0} = 0.08185826134\}$$

```
> Jacy:=VectorCalculus:-Jacobian(Phitilde,(indetermined_Phi))
```

$$Jacy := rtable(1\ldots6, 1\ldots7, [...], subtype = Matrix)$$

```
> DRJactilde:=LinearAlgebra:-Copy(Jacy):
> DRJactilde:=map(diff,DRJactilde,R):
> gy:=<seq(gval[op(x)],x in indetermined_Phi)>
```

$$gy := rtable(1\ldots7, [0.2080196158, 0.08380398812, 0.1745210998, 0.1546089646, 0.08185826134, 0.2792561716, 0.5894260557], subtype = Vector_{column})$$

```
> vect:=evalf(subs(substitution,Jacy.gy+ p*gval[R]*DRphi +
p*hval[R]*DRJactilde.gy)+subs(substitution_g,(Jacytilde-subs({seq(x=0,
x in indetermined_Phi)},Jacytilde)).Qytilde_vect))
```

$$vect := rtable(1\ldots6, [2.91356831746550, 0.277639134512868, 1.77790203517211, 1.05435910824861, 1.09383532295684, 4.65515838962638], subtype = Vector_{column})$$

```
> gvaltilde_vect:=LinearAlgebra:-LinearSolve(M,vect):
```

```
> ### Values of g_Qytilde at z=rho
> for i from 1 to nops(indetermined_tilde) do:
> gvaltilde[op(indetermined_tilde[i])]:= gvaltilde_vect[i];
> od;
```

$$gvaltilde_{0,0} := 22.7105001312604$$

$$gvaltilde_{0,1} := 9.58737068993794$$

$$gvaltilde_{1,0} := 15.2609199684249$$

$$gvaltilde_{1,1} := 16.8897440393078$$

$$gvaltilde_{2,0} := 8.61516558962385$$

$$gvaltilde_{G} := 40.2912611107740$$

```
> ######################################################################
> ##              Expected size after reduction                       ##
> ######################################################################

> Norm1:=add((gvaltilde[i,0]+gvaltilde[i,1])* binomial(k,i), i=0..k-1)+
gvaltilde[G]+gvaltilde[k,0]
```

$$Norm1 := 145.505625537062$$

```
> ## The expected size after reduction is:
> S:= evalf(1/gval[L]*(p*gval[R] + Norm1))
```

$$S := 77.7972457523802$$