The limits of Nečiporuk's method and the power of programs over monoids taken from small varieties of finite monoids

Nathan Grosshans

université PARIS-SACLAY

école normale supérieure paris-saclay

Université de Montréal

The limits of Nečiporuk's method and the power of programs over monoids taken from small varieties of finite monoids

Thèse de doctorat de l'Université Paris-Saclay, préparée à l'École Normale Supérieure de Cachan et de l'Université de Montréal

École doctorale n°580 : Sciences et technologies de l'information et de la communication

Faculté des arts et des sciences, Département d'Informatique et de Recherche Opérationnelle

Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Cachan, le 25 septembre 2018, par

Grosshans Nathan

Composition du jury :

Diekert Volker	Professeur, Universität Stuttgart	Rapporteur / Examinateur externe
Függer Matthias	Chargé de recherche, CNRS	Examinateur / Membre du jury
Hamel Sylvie	Professeur, Université de Montréal	Présidente / Présidente-rapporteuse
McKenzie Pierre	Professeur, Université de Montréal	Directeur de thèse
Segoufin Luc	Directeur de recherche, INRIA	Directeur de thèse
Zeitoun Marc	Professeur, Université de Bordeaux	Rapporteur / Examinateur externe

NNT: 2018SACLN028

Résumé

Cette thèse porte sur des minorants pour des mesures de complexité liées à des sous-classes de la classe P de langages pouvant être décidés en temps polynomial par des machines de Turing. Nous considérons des modèles de calcul non uniformes tels que les programmes sur monoïdes et les programmes de branchement.

Notre première contribution est un traitement abstrait de la méthode de Nečiporuk pour prouver des minorants, indépendamment de toute mesure de complexité spécifique. Cette méthode donne toujours les meilleurs minorants connus pour des mesures telles que la taille des programmes de branchements déterministes et non déterministes ou des formules avec des opérateurs booléens binaires arbitraires ; nous donnons une formulation abstraite de la méthode et utilisons ce cadre pour démontrer des limites au meilleur minorant obtenable en utilisant cette méthode pour plusieurs mesures de complexité. Par là, nous confirmons, dans ce cadre légèrement plus général, des résultats de limitation précédemment connus et exhibons de nouveaux résultats de limitation pour des mesures de complexité auxquelles la méthode de Nečiporuk n'avait jamais été appliquée.

Notre seconde contribution est une meilleure compréhension de la puissance calculatoire des programmes sur monoïdes issus de petites variétés de monoïdes finis. Les programmes sur monoïdes furent introduits à la fin des années 1980 par Barrington et Thérien pour généraliser la reconnaissance par morphismes et ainsi obtenir une caractérisation en termes de semi-groupes finis de NC^1 et de ses sous-classes. Étant donné une variété $\mathbf{V}$ de monoïdes finis, on considère la classe $\mathcal{P}(\mathbf{V})$ de langages reconnus par une suite de programmes de longueur polynomiale sur un monoïde de $\mathbf{V}$: lorsque l'on fait varier $\mathbf{V}$ parmi toutes les variétés de monoïdes finis, on obtient différentes sous-classes de NC^1, par exemple AC^0, ACC^0 et NC^1 quand $\mathbf{V}$ est respectivement la variété de tous les monoïdes apériodiques finis, résolubles finis et finis. Nous introduisons une nouvelle notion de docilité pour les variétés de monoïdes finis, renforçant une notion de Péladeau. L'intérêt principal de cette notion est que quand une variété $\mathbf{V}$ de monoïdes finis est docile, nous avons que $\mathcal{P}(\mathbf{V})$ contient seulement des langages réguliers qui sont quasi reconnus par morphisme par des monoïdes de $\mathbf{V}$. De nombreuses questions ouvertes à propos de la structure interne de NC^1 seraient réglées en montrant qu'une variété de monoïdes finis appropriée est docile, et, dans cette thèse, nous débutons modestement une étude exhaustive de quelles variétés de monoïdes finis sont dociles. Plus précisément, nous portons notre attention sur deux petites variétés de monoïdes apériodiques finis bien connues : $\mathbf{DA}$ et $\mathbf{J}$. D'une part, nous montrons que $\mathbf{DA}$ est docile en utilisant des arguments de théorie des semi-groupes finis. Cela nous permet de dériver une caractérisation algébrique exacte de la classe des langages réguliers dans $\mathcal{P}(\mathbf{DA})$. D'autre part, nous montrons que $\mathbf{J}$ n'est pas docile. Pour faire cela, nous présentons une astuce par laquelle des programmes sur monoïdes de $\mathbf{J}$ peuvent reconnaître beaucoup plus de langages réguliers que seulement ceux qui sont quasi reconnus par morphisme par des monoïdes de $\mathbf{J}$. Cela nous amène à conjecturer une caractérisation algébrique exacte de la classe de langages réguliers dans $\mathcal{P}(\mathbf{J})$, et nous exposons quelques résultats partiels appuyant cette conjecture. Pour chacune des variétés $\mathbf{DA}$ et $\mathbf{J}$, nous exhibons également une hiérarchie basée sur la longueur des programmes à l'intérieur de la classe des langages reconnus par programmes sur monoïdes de la variété, améliorant par là les résultats de Tesson et Thérien sur la propriété de longueur polynomiale pour les monoïdes de ces variétés.

Mots-clés Complexité algorithmique, minorants, Nečiporuk, programmes sur monoïdes, DA, J

Abstract

This thesis deals with lower bounds for complexity measures related to subclasses of the class P of languages that can be decided by Turing machines in polynomial time. We consider non-uniform computational models like programs over monoids and branching programs.

Our first contribution is an abstract, measure-independent treatment of Nečiporuk's method for proving lower bounds. This method still gives the best lower bounds known on measures such as the size of deterministic and non-deterministic branching programs or formulæ with arbitrary binary Boolean operators; we give an abstract formulation of the method and use this framework to prove limits on the best lower bounds obtainable using this method for several complexity measures. We thereby confirm previously known limitation results in this slightly more general framework and showcase new limitation results for complexity measures to which Nečiporuk's method had never been applied.

Our second contribution is a better understanding of the computational power of programs over monoids taken from small varieties of finite monoids. Programs over monoids were introduced in the late 1980s by Barrington and Thérien as a way to generalise recognition by morphisms so as to obtain a finite-semigroup-theoretic characterisation of NC^1 and its subclasses. Given a variety $\mathbf{V}$ of finite monoids, one considers the class $\mathcal{P}(\mathbf{V})$ of languages recognised by a sequence of polynomial-length programs over a monoid from $\mathbf{V}$: as $\mathbf{V}$ ranges over all varieties of finite monoids, one obtains different subclasses of NC^1, for instance AC^0, ACC^0 and NC^1 when $\mathbf{V}$ respectively is the variety of all finite aperiodic, finite solvable and finite monoids. We introduce a new notion of tameness for varieties of finite monoids, strengthening a notion of Péladeau. The main interest of this notion is that when a variety $\mathbf{V}$ of finite monoids is tame, we have that $\mathcal{P}(\mathbf{V})$ does only contain regular languages that are quasi morphism-recognised by monoids from $\mathbf{V}$. Many open questions about the internal structure of NC^1 would be settled by showing that some appropriate variety of finite monoids is tame, and, in this thesis, we modestly start an exhaustive study of which varieties of finite monoids are tame. More precisely, we focus on two well-known small varieties of finite aperiodic monoids: $\mathbf{DA}$ and $\mathbf{J}$. On the one hand, we show that $\mathbf{DA}$ is tame using finite-semigroup-theoretic arguments. This allows us to derive an exact algebraic characterisation of the class of regular languages in $\mathcal{P}(\mathbf{DA})$. On the other hand, we show that $\mathbf{J}$ is not tame. To do this, we present a trick by which programs over monoids from $\mathbf{J}$ can recognise much more regular languages than only those that are quasi morphism-recognised by monoids from $\mathbf{J}$. This brings us to conjecture an exact algebraic characterisation of the class of regular languages in $\mathcal{P}(\mathbf{J})$, and we lay out some partial results that support this conjecture. For each of the varieties $\mathbf{DA}$ and $\mathbf{J}$, we also exhibit a program-length-based hierarchy within the class of languages recognised by programs over monoids from the variety, refining Tesson and Thérien's results on the polynomial-length property for monoids from those varieties.

Keywords Computational complexity, lower bounds, Nečiporuk, programs over monoids, DA, J

Contents

List of Tables

List of Figures

List of Acronyms and Abbreviations

e.g.	For example, from Latin *exemplī grātiā*
i.e.	That is to say, from Latin *id est*
$\oplus$BP	Parity Branching Program
BF	Binary Formula
BP	Branching Program
DTM	Deterministic Turing Machine
LBP	Levelled Branching Program
LNBF	Limited Non-deterministic Binary Formula
LNBP	Limited Non-deterministic Branching Program
NBP	Non-deterministic Branching Program
NTM	Non-deterministic Turing Machine
SUM	Strongly Unambiguous Monomial
SUMT	Strongly Unambiguous Monomial Tree

Dédicaces

À Jésus-Christ, mon Seigneur et mon Sauveur.

« En lui se trouvent cachés tous les trésors de la sagesse et de la connaissance. »

La Bible (traduction « Semeur »), épître aux Colossiens, chapitre 2, verset 3

Remerciements

Ça y est, vient enfin le temps d'écrire les remerciements en prélude de mon manuscrit de thèse, quasiment quatre ans jour pour jour après avoir débuté mon doctorat ! Je me livre donc ici à l'exercice de remercier toutes les personnes qui ont, de près ou de loin, fait partie de l'aventure de mon doctorat sachant que, bien souvent, ma reconnaissance à leur égard ne se limite de loin pas à celle que j'exprime succinctement ici en rapport avec ma thèse. Ma finitude humaine faisant que j'en oublierai très certainement certaines d'entre elles, je les prie d'ors et déjà de m'excuser.

J'aimerais commencer par remercier mes deux directeurs de thèse, Pierre McKenzie et Luc Segoufin, sans qui — et c'est indéniable — le manuscrit que voici n'aurait pas pu voir le jour : il n'aurait pas existé, ou tout du moins, aurait été fondamentalement différent. C'est à eux que je dois l'orientation des recherches effectuées à la fois durant mon stage de M2 et mon doctorat, c'est eux qui ont été à la genèse de nombre des questions abordées dans ce manuscrit et eux encore qui ont toujours été présents sur le chemin parfois tortueux vers leur résolution — chemin au bout duquel je ne suis pas encore arrivé pour certaines d'entre elles. Je les remercie tout particulièrement de m'avoir fait confiance en me prenant sous leur tutelle en stage de recherche de M2 puis en thèse, alors que je n'étais, du point de vue de mes résultats en Master, pas non plus l'étudiant le plus brillant qui soit !

Pierre, alors que nous avons peut-être moins travaillé ensemble que ce que j'ai pu faire avec Luc, j'ai toujours grandement apprécié les moments passés dans ton bureau quand j'étais à Montréal, afin d'y discuter de l'avancement de mes recherches (plutôt de celui de la rédaction de ma thèse au cours de l'année qui vient de passer, d'ailleurs), de ton cours l'été où j'en étais l'assistant d'enseignement, de problèmes intéressants liés à tes cours ou aux travaux de tes autres étudiants, ou alors tout simplement de choses et d'autres non académiques. Les pistes que tu m'as données pour mes recherches se sont révélées être précieuses voire fondamentales au cours des quatre ans et demi qui viennent de passer : sans elles, ma thèse n'aurait assurément pas le même visage ! Ton érudition en théorie de la complexité algorithmique ainsi que ta capacité à formuler des questions de recherche

intéressantes, parfois à la croisée de différents sujets, sont un exemple pour moi. Je te suis également reconnaissant pour ton important travail de relecture, que ce soit pour mon manuscrit de thèse, les articles que nous avons coécrits bien évidemment, diverses présentations ou encore des documents plus administratifs ; leur qualité linguistique, surtout pour ceux rédigés en anglais, s'est vue améliorée et tu as pu y contenir quelque peu mon style littéraire alambiqué. Par-dessus tout, je te remercie tout particulièrement d'avoir eu un souci discret mais constant pour moi, bien au-delà des aspects académiques de ma vie, et pas uniquement lors de mes séjours à Montréal.

Luc, bien que j'aie pour l'instant encore du mal à estimer son étendue, je pense que tu as exercé une influence durable et profonde sur ma manière de faire de la recherche, et je pense que c'est une très bonne chose. Nos longues et régulières sessions de réflexion communes sont non seulement à l'origine de nombre des résultats les plus importants de cette thèse, en plus d'avoir été un véritable moteur pour l'engrangement de connaissances en informatique théorique et le développement de mes compétences mathématiques, mais auront très certainement, à l'avenir, une portée plus importante encore pour ce qui est de ma manière d'aborder les problèmes de recherche : ta persévérance à toute épreuve face à un problème affreusement résistant, ta préoccupation marquée pour l'élégance et la simplicité que tu crois intrinsèque aux mathématiques sont autant de qualités de chercheur que j'admire et auxquelles j'aspire. Je te remercie pour les heures que tu as passées à relire mon manuscrit de thèse ou à remanier les articles que nous avons coécrits, souvent beaucoup trop longs, difficiles à lire et parfois inutilement techniques sous ma plume, et ce toujours avec le souci de me faire progresser dans ce domaine. Si cette thèse est un tant soit peu claire et lisible sans indigestion, tu n'y es justement pas pour rien. Merci, enfin, pour ta disponibilité exemplaire en tant que directeur de thèse, tes encouragements constants vis-à-vis de mon travail et tous tes bons conseils vis-à-vis de l'organisation du travail et de la carrière d'un chercheur.

J'aimerais à présent remercier les personnes qui ont accepté de faire partie de mon jury de thèse, sans qui je ne pourrais pas obtenir le titre de docteur.

Ma reconnaissance va d'abord à Volker Diekert et Marc Zeitoun, qui ont tous les deux accepté d'être rapporteurs de ma thèse, se prêtant par là, malgré toutes leurs contraintes et obligations de professeurs, à la lecture du pavé non négligeable que représente mon manuscrit de thèse. J'avoue être quelque peu gêné d'avoir dû leur infliger la lecture et l'évaluation d'une thèse aussi longue, aussi je leur suis doublement reconnaissant non seulement de l'avoir fait, mais aussi pour la qualité de leur travail. Avoir de tels chercheurs de premier plan en informatique théorique pour rapporteurs est un honneur pour moi

et j'ai été véritablement encouragé par leurs rapports très élogieux. Danke Volker, merci Marc !

Je tiens ensuite à exprimer ma gratitude à Stefan Göller, Matthias Függer qui l'a remplacé in extremis, et Sylvie Hamel pour avoir accepté d'être examinateurs dans mon jury de thèse. Danke Stefan für dein Interesse an meiner Arbeit, es hat mich gefreut mit dir ein bisschen über Forschung diskutieren zu können und es hätte mich gefreut, dich wie geplant als Jurymitglied meiner Dissertation Verteidigung zu haben. Danke Matthias, dass du noch im letzten Moment akzeptiert hast, Stefan zu ersetzen — das hast du sehr ernst genommen und toll gemacht, in sehr kurzer Zeit ! Merci Sylvie, toi qui fus déjà membre du jury de mon examen pré-doctoral à l'Université de Montréal, d'avoir en plus accepté d'être membre de mon jury de thèse. Les considérations de mes recherches ne sont certes pas des plus proches des tiennes, mais je sais que cela ne te pose pas de problème !

Je souhaiterais maintenant passer à toutes les personnes aux côtés desquelles j'ai pu travailler à Cachan, au LSV, ainsi qu'à Montréal, au DIRO.

Pour ce qui est du LSV, je commence par remercier tous les confrères doctorants que j'ai pu y côtoyer. Merci en particulier à toi Jérémy D. pour nos discussions de science et d'autre (même si parfois mes questions d'algèbre devaient paraître enfantines pour un matheux comme toi), pour le soutien mutuel dans notre travail et les galères administratives ; merci à toi Simon Ha. pour les nombreuses discussions que nous avons pu avoir (et pas seulement de théorie de la complexité algorithmique), mais surtout pour ta gracieuse et précieuse aide logistique extra-académique à plusieurs reprises ; thank you Juraj for our numerous discussions about so many subjects, you're an incredibly cultivated person and I was particularly glad to spend so much time with you talking about God and reading the Bible ; merci à toi Marie v.d.B. pour le temps que nous avons passé à discuter de beaucoup de choses en tant qu'occupants du quatrième étage, ainsi que pour tes conseils et ton soutien d'« aînée » ; merci à tous mes autres « aînés » que j'ai pu côtoyer, à savoir Guillaume S., Julien, Nadime, Rémy, Simon T., David M., Daniel St., Lucca, Patrick ; merci à tous les autres doctorants résidents du quatrième étage, dont Samy, Hugues M. et en particulier Pierre C. et Mengqi qui ont partagé mon bureau à différentes périodes. Je remercie également les autres occupants du quatrième étage que j'ai pu côtoyer assez régulièrement, dont Laurent D., Dietmar, Serge A., Étienne, Cristina, Stefan G., Thomas C. et spécialement Jakub, Thomas P.-T. et Fabian qui ont été, à un moment ou un autre, installés dans le même bureau que moi. Je ne veux pas oublier de remercier tous ceux avec qui j'ai pu enseigner dans divers cours à l'ENS Cachan, à savoir David B., Claudine et Stefan S., qui m'ont permis de le faire dans de très bonnes conditions tout en appre-

nant et en progressant ; à ce sujet, merci aussi à Paul G. et Serge H., directeurs successifs du département informatique de l'ENS Cachan. Ma reconnaissance va enfin à l'efficace et sympathique personnel administratif et technique du LSV, à savoir Catherine, Imane, Virginie, Thida, Francis et Hugues M.-V. : merci car j'ai toujours pu trouver chez vous une oreille attentive quand j'avais besoin de quelque chose !

Du côté du DIRO, je commence également par remercier tous les confrères doctorants du LITQ que j'y ai côtoyés : Charles B., Paul R.-R., Michael B., Sara et Dendi, sans oublier Philippe L. avec qui j'ai partagé mon bureau lors de mes trois séjours à Montréal. Je remercie aussi Martin et Hugo, étudiants de Pierre au niveau maîtrise, avec qui j'ai pu m'entretenir et travailler un peu, ainsi que Billel, doctorant d'une autre équipe, cooccupant du bureau qu'on m'avait affecté durant mes premiers mois à l'UdeM. Je souhaite exprimer ma gratitude à Louis pour son cours sur la cryptographie que j'ai pu suivre ainsi que pour nos nombreuses conversations touchant à la cryptographie, la théorie de la complexité algorithmique et bien d'autres sujets encore, scientifiques ou non ; merci aussi à lui d'avoir été membre du jury de mon examen pré-doctoral. Yuval, I am grateful for the moments we spent together during my first stay in Montreal and for the contact we maintained since then ; thanks also for your particular concern and encouragement about my research. Pour terminer, comme pour le côté français, ma reconnaissance va à la partie du personnel administratif et technique du DIRO à laquelle j'ai eu affaire, comparablement efficace et sympathique : Véronique, Céline, Violette, Linda et Mohamed, merci pour votre disponibilité et votre aide !

Il est un certain nombre d'amis non liés directement au LSV ou au DIRO, ni aux catégories qui suivent, que je me dois de remercier maintenant.

Wei Guo, thank you for all this time that we spent together eating Southeast Asian, Chinese, Alsatian and French food, discussing about a great variety of topics, encouraging each other and even celebrating Christmas together with my family. I'm glad we could have such a close relationship and I hope we will find some way to keep on with it.

Dorian, je t'exprime, toi aussi, ma gratitude pour tous les moments que nous avons passés ensemble à faire des mathématiques, manger, parfois jouer, mais aussi lire la Bible et parler de Dieu. Je te suis particulièrement reconnaissant d'avoir pris l'initiative, durant l'époque où nous étions encore en M2, de travailler avec moi les cours que nous avions en commun, et ce du début de l'année jusqu'aux examens finaux, alors que, tandis que toi tu survolais les cours, moi je frôlais dangereusement le sol — je pense avoir, mathématiquement parlant, véritablement progressé avec toi !

Je remercie également l'équipe des « Cachanais » avec qui j'ai fait mon M2, à savoir

Jérémy D., Simon Ha., Samy, Laurent, Romain, Étienne et Stéphane H. (dont certains furent déjà cités plus haut), qui n'ont pas hésité à m'intégrer dans leur groupe, alors même que je n'étais pas de l'ENS Cachan. Avec eux, j'ai non seulement découvert le raffinement de l'alimentation cachanaise mais aussi, et surtout, passé d'excellents moments entre informaticiens théoriciens en devenir (ne se prenant souvent pas trop au sérieux).

Merci beaucoup à toi, Robert, pour tous ces repas partagés au restaurant universitaire de l'UdeM, de même que les moments passés ensemble en dehors. Je me réjouis d'avoir pu t'aider et j'ai confiance que tu parviendras à obtenir ton diplôme !

Enfin, merci à Jordi, mon colocataire lors de mon premier séjour à Montréal (et à Gilles, notre génial propriétaire), à mes co-résidents et aux responsables de la résidence étudiante du Chemin Neuf où je logeais lors de mon deuxième séjour à Montréal et à mes co-résidents de la résidence étudiante de la Maison de l'Amitié qui fut mon domicile durant mon troisième et dernier séjour à Montréal.

Vient à présent un groupe de personnes pour lequel je ne peux vraiment pas remercier chacun nommément, et pour cause, il est beaucoup trop grand : il s'agit de mes frères et sœurs chrétiens, dont certains sont de très proches amis. À tous ceux-là : merci pour vos prières, vos paroles réconfortantes et encourageantes, les moments passés à discuter, lire la Bible ou encore louer Dieu ensemble ; sans vous, je ne serais sans doute pas arrivé au bout de mon doctorat, tout du moins en bien plus mauvais état ! Ma reconnaissance va spécialement (et de manière plus ou moins arbitraire) à Amen et Ingrid, David et Anne-Laure A., Clara A., François, Jérémy A., Josh, Dalmace, Yannick et Julie, Clémence, Ivan et Maïlys, Laetitia, Nicolas et Nolwenn Be., David B., Nicolas et Marie-Aude Bl., Joël B., Marie-Josée, Susan, Jacques et Nelly, Valérie, Kenny et Lisa, Salomon, Joël et Cécile C., Marianne, Philippe et Karina C., Ian-Alexis, Clara D., Marion D., Lucie, Nikki, Alain D.K., Eunice, Elyeser, Olivier F., Joyce, Tommy, Eduardo, Adrien, Joël G., Jean-Paul, Vincent et Micaëlla, Dylan, Lionel, Séverine, Billy, Paul H., Alix, Thomas H., Eudes, Philippe H., Simon Hu., Jonathan H., Marc, Boris, Gaston, Olivier K., Sarah, Nicolas K., Joël et Florine L., Libby, Tatiana, Marguerite, Stéphane L., Marjorie, Jonathan L., Caillou et Nathalie, Cécile M., Olivier M., Aloysia, Emmanuelle, Clifford, Nia, Laura Ma., Jean-Marc, Arianne, David M., Christelle, Laura Mi., Alain M.B., Laurie M., Emmanuel N., Ruben, Philippe N., Line, Jean-Baptiste O., Jonathan et Marie P., Marion P., Sophie P., Aina, Aro, Michaël Ran., Micaël Raz., Carlina, Théophile, Xavier, Sophia, Jérémie S., Daniel et Anna Sc., Guy-Pierre, Joël et Valérie S., Lauricia, Paul S., Michel et Pamina, Ulisses et Maryllya, Denis et Francine, Yohann, Génovah, Emmanuel V., Charles et Anneline V., Isabelle, Gaëtan, Guillaume Y., Po et Mblia, Situo, Lyès — j'oublie mal-

heureusement très certainement de citer quelques personnes qui auraient dû figurer dans cette liste, et je m'en excuse.

Je ne veux ceci dit surtout pas oublier de remercier ma famille « biologique » : mes parents, Philippe et Andrea, qui n'ont eu de cesse de me soutenir (en prières, paroles et actes), et pas seulement depuis le moment où j'ai commencé mon doctorat (même si à partir de ce moment, ce que je faisais dans mes études est devenu irrémédiablement obscur) ; ma sœur Rebecca et mon beau-frère Sebastian, pour le même type de soutien et la proximité malgré la distance qui nous sépare ; mon frère Matthias et ma sœur Roxanne pour leur amour fraternel ; ma grand-mère Doris, ma tante Betty, mon oncle Bernard, ma cousine Sophie et mon cousin Pierrick qui prennent de mes nouvelles régulièrement, m'encouragent et me soutiennent dans la prière ; mein Großvater Heinz, für dein Interesse an meiner Forschungsarbeit.

Dans la dernière ligne droite, j'ai aussi eu la joie de recevoir un soutien similaire de la part de mes futurs beaux-parents, Olivier et Sylvie, et de mes futures belles-sœurs, Laurie et Aimie, ainsi que de ma future belle-famille, au sens plus large du terme — cela m'a touché et je leur en suis vraiment reconnaissant !

Enfin, Marie, j'ai en toi une personne exceptionnelle que Dieu a mise à mes côtés et je t'exprime ici ma profonde gratitude pour ton immense soutien vis-à-vis de ma thèse, tout spécialement au cours des longs mois de rédaction passés à des milliers de kilomètres de toi. Alors que ce que je faisais au jour le jour t'était plutôt incompréhensible, tu as su, avec beaucoup de patience, en grande partie à distance mais avec beaucoup de constance, passer du temps avec moi, m'écouter, me réconforter, m'encourager, prier avec et pour moi et simplement être là. Merci.

Je souhaiterais terminer en remerciant Jésus-Christ, mon Seigneur, mon Dieu et mon Sauveur, l'alpha et l'oméga en et de qui j'ai tout, *absolument* tout. Si mon doctorat a pu se dérouler de la manière dont il s'est déroulé et s'achever comme il est en train de l'être, c'est grâce à toi. Si chaque personne citée ici a joué le rôle qu'elle a joué, c'est encore grâce à toi. Je te suis éternellement reconnaissant, toi, à l'amour et la fidélité sans faille, qui prends soin de moi comme personne. À toi seul la gloire !

> Je veux te louer, Eternel, au milieu des peuples,
> et te célébrer en musique parmi les nations.
> Ton amour s'élève plus haut que les cieux,
> ta fidélité jusqu'aux nues.
>
> La Bible (traduction « Semeur »), psaume 108, versets 4 à 5

Introduction

The general framework. The theory of computation is the branch of mathematics[1] that seeks to understand the foundations of computation. That branch was born during and as a consequence of the early 20th century foundational crisis of mathematics, when the need arose for mathematicians to formalise and understand precisely the notion of an *algorithm*, informally an effective procedure defining a sequence of operations carrying out a *computation*. Formalising computation hinges on *models of computation*, classes of mathematical objects that each implement a *computational strategy* built from elementary *computational steps*. A *computation* is then a particular rolling out of such a strategy given a certain mathematical object as *input*, which it transforms into a certain mathematical object called *output*. A given computational strategy therefore computes some *transformation* of a set of mathematical objects into another such set. It should be clear that the notion of computation depends on the model of computation. To each model of computation, we can also associate one or more *complexity measures*: each of them relies on some measure of *computational cost* associated to any computational strategy of this model, and gives for any transformation (often restricted to be of a certain type), the minimum cost of a computational strategy of this model computing that transformation.

A central example of a model of computation is the *Turing machine*: such a machine has a read/write head and a register holding one of finitely many possible states. The head points to a cell on an infinite one-dimensional discrete tape made of cells, each containing a letter taken from a finite set (called an alphabet). At each step, the head is positioned on a certain cell, reading some letter, and the register holds a certain state; following a finite set of rules, depending on this letter and this state, the machine then carries out the step by changing the state in its internal register, writing a letter in the cell the head is currently on and moving its head one cell to the left or one cell to the right. A computation is then the sequence of steps the machine does starting at an initial position, holding an initial state in its internal register and with the input tape initialised to some

[1] If we consider theoretical computer science as being a branch of mathematics.

input and stopping in some final state. Two essential examples of associated complexity measures are those based, respectively, on the notion of time (the number of steps done by the machine for a given computation) and the notion of space (the number of cells used by the machine for a given computation). Decision problems, in which given some input, one shall give a yes/no answer as to whether this input verifies some property, are central in the theory of computation. This is why the transformations we are mostly interested in are those that correspond to the indicator function of some language (a set of words over some given alphabet), that tells, for each word over the associated alphabet, whether the word belongs to the language or not. When some computational strategy of a given model of computation computes the indicator function of a language, we say it decides that language.

Two important branches in the theory of computation are:

- *computability theory*, that seeks to understand, given a certain model of computation, *what* transformations can be computed by a computational strategy from this model;

- *computational complexity theory*, that seeks to understand, given a certain model of computation and an associated complexity measure, *how efficiently*, in terms of computational cost (given by this measure), a transformation can be computed by a computational strategy from this model.

This Ph.D. thesis deals with a subject rooted at the heart of the second aforementioned branch. More precisely, a central question in computational complexity theory is to understand the relationship between the class L of languages that can be decided by Turing machines using an amount of space that is upper bounded logarithmically in the length of the input word and the class P of languages that can be decided by Turing machines using an amount of time that is upper bounded by a polynomial in the length of the input. It is well known that $\mathsf{L} \subseteq \mathsf{P}$, but almost nothing is known about the reverse inclusion, though it is widely believed not to hold.

A way to prove that $\mathsf{P} \nsubseteq \mathsf{L}$ would be to show strong enough lower bounds on the size of *branching programs* deciding some explicit language in P. A branching program on binary words of length n is a directed acyclic graph with one source and two sink vertices, one labelled 0, the other labelled 1, each internal node labelled by some position in the input and having exactly two outgoing arcs, one labelled 0 and the other one labelled 1. A certain input word then yields a unique path in that branching program that either goes to the sink vertex labelled 0 (rejection) or to the sink vertex labelled 1 (acceptance).

The fundamental measure of computational cost of a branching program is its size, the number of its non-sink vertices. All this can easily be generalised to an arbitrary alphabet. To decide a language of words of arbitrary length, we then provide a sequence of branching programs, each deciding the restriction of that language to words of a different length; this is why this model of computation is said to be *non-uniform*, as opposed to *uniform* models of computation like Turing machines, verifying that we provide a sole computational strategy for deciding a given language of words of arbitrary length. The measure of size associated with that model is, for a given language, the function giving for each possible input length the minimum size of any branching program deciding the restriction of that language to words with that given length. The model of branching programs is another well-studied model of computation and the complexity measure of size in this model captures precisely the measure of space in the model of Turing machines, in that if a language is decided by a Turing machine using space at most $s(n)$ on any input word of length n ($s\colon \mathbb{N} \to \mathbb{N}$ being a function satisfying some mild conditions), then it can be decided by a sequence of branching programs, the one for length $n \in \mathbb{N}$ input words being of size at most $2^{\alpha \cdot s(n)}$ for some constant $\alpha \in \mathbb{R}_{>0}$. Thus, proving a super-polynomial lower bound on the size of branching programs deciding some explicit language in P would separate the latter from L, because it would imply a super-logarithmic lower bound on the amount of space used by any Turing machine deciding that language.

Although the branching program model is combinatorially simpler to handle than the Turing machine model, such a lower bound is currently far out of reach, and a lower bound in $\Theta(n^2/\log^2 n)$ is still the best we have been knowing for more than 50 years. This state of affairs is similar to the case of the most common non-uniform models of computation based on *circuits*. The latter have been the most studied non-uniform models of computation because the measure of size of circuits with NOT and OR, AND gates of fan-in 2 captures precisely the measure of time in the model of Turing machines, thus giving a way to attack the most famous (and, perhaps, most fundamental) open question in computational complexity theory: understanding the relationship between P and the class NP of languages that can be decided by *non-deterministic Turing machines* using an amount of time that is upper bounded by a polynomial in the length of the input. Unfortunately, not even a super-linear lower bound on the size of such circuits is known, while a super-polynomial lower bound on the size of such circuits deciding some explicit language in NP would be needed.

This is, especially since the 1980s, why computational complexity theorists have mostly focused on size lower bounds for circuits, branching programs and other related non-

uniform models of computation on which some *restrictions* have been placed, in the hope that proving such lower bounds would give new ideas to prove size lower bounds for the unrestricted models. This approach has been rather successful, leading to some fundamental results in computational complexity theory, even if the aforementioned ultimate hope failed to materialise for the moment (and still seems well out of reach). The present Ph.D. thesis is a modest contribution to the advancement of that strand of research in computational complexity theory.

Nečiporuk's method. As pointed out earlier, the best lower bound known for the size of branching programs deciding a language in NP is still, at the time of writing of this thesis, the one proved by Nečiporuk more than 50 years ago Nečiporuk [1966], that is in $\Theta(n^2/\log^2 n)$. Nečiporuk implicitly used a technique that has later on been explicitly identified, defined and applied to several other complexity measures for different models of computation, and also been exposed in several classical textbooks Savage [1976], Wegener [1987, 2000], Jukna [2012]. The first contribution of this thesis is a formulation of Nečiporuk's lower bound method upstream from any specific complexity measure (which had never been done systematically before) and the analysis of the limitations of the method (in terms of the best lower bound obtainable) induced by upper bounds on the complexity of deciding one specific language. This framework is then used to apply the method and show its limitations for both classical computational models and associated complexity measures as well as variants of those never studied before. In the end, in this slightly more general framework, well-known lower bounds and limitation results are reproved and new ones are shown.

Programs over monoids. The remainder of this Ph.D. thesis then concentrates on one specific restricted variant of branching programs. Among the variants studied by computational complexity theorists, *bounded-width branching programs* appeared to be of particular interest, especially starting from the moment at which Barrington Barrington [1989] proved the unexpected result that bounded-width polynomial-size branching programs decide all languages in NC^1. NC^1 is the class of languages decided by polynomial-size logarithmic-depth circuits with NOT and OR, AND gates of fan-in 2, a central and well-studied complexity class based on circuits whose depth has been restricted. Its inclusion into L when considering an appropriate uniform variant is probably strict, but it is still not excluded that $\mathsf{NP} \subseteq \mathsf{NC}^1$. A wealth of the major lower bound results on the size of restricted variants of circuits were obtained for variants in which circuits are required to be of constant depth, variants which all correspond to important subclasses of NC^1.

As a direct follow-up of that result and its proof, Barrington and Thérien Barrington and Thérien [1988] introduced the *program-over-monoid* model. Given some finite monoid $(M, *)$ and an alphabet Σ, an $(M, *)$-program P over Σ for input length n is simply a sequence of instructions

$$(i_1, f_1)(i_2, f_2) \cdots (i_l, f_l) \ ,$$

where for each j, i_j indicates some position in the input and f_j associates an element of M to each element of Σ. That way, P associates to each input word w over Σ of length n a unique element of M,

$$P(w) = f_1(w_{i_1}) * f_2(w_{i_2}) * \cdots * f_l(w_{i_l}) \ .$$

A language of words in Σ^n is then recognised by P if and only if it is equal to the set of words in Σ^n to which the program P associates an element in some subset F of M. A language of words of arbitrary length is for its part recognised by a sequence of $(M, *)$-programs.

This model of computation and notion of recognition can be seen as a generalisation of the notion of recognition through morphisms into finite monoids, a notion at the very basis of algebraic automata theory. As for the case of classical recognition through such morphisms, we are usually interested in the class of languages recognised by a sequence of $(M, *)$-programs where $(M, *)$ is a finite monoid drawn from some variety of finite monoids $\mathbf{V}$ (such a variety being a class of finite monoids closed under direct product and division, two basic operations on monoids). The striking result proved by Barrington and Thérien (and in some following papers) is that NC^1 and almost all of its well-known subclasses (with the notable exception of TC^0) can each be characterised as some class $\mathcal{P}(\mathbf{V})$ of languages recognised by sequences of polynomial-length programs over monoids taken from such a variety $\mathbf{V}$. For instance, let us consider AC^0, the class of languages decided by polynomial-size constant-depth circuits with NOT and OR, AND gates of unbounded fan-in, and ACC^0 the same class with the addition of modular counting gates — two of the most important subclasses of NC^1. Then, we have that NC^1, ACC^0 and AC^0 contain, respectively, exactly those languages recognised by polynomial-length programs over monoids from the variety of all finite monoids, of finite solvable monoids and of finite aperiodic monoids. In algebraic automata theory, lots of techniques have been developed since the 1960s to characterise algebraically classes of regular languages by the varieties of finite monoids recognising them through morphisms. The hope was, and still is, that those would be generalisable to recognition through programs and help to tackle open

computational-complexity-theoretic questions related to NC^1 and its inner structure; or, at least, lead to new semigroup-theoretic proofs of well-known results in that realm. The most prominent and fundamental of these results is the one stating that for all $m \geq 2$, the language MOD_m of words over $\{0,1\}$ containing a number of 1s not divisible by m is not in AC^0.

But none of these hopes materialised for the moment, at the time of writing of this thesis. As already explained for the case of lower bounds on the sizes of (unrestricted) circuits, branching programs and other related non-uniform models of computation, one general approach one can follow in face of such difficulties is to focus on more restricted models and try to accumulate knowledge and techniques that could help in the unrestricted setting. The program-over-monoid formalism offers a straightforward way to restrict the power of the model, simply by restricting the "algebraic power" at hand by considering programs over monoids taken from some "small" (in the sense of inclusion) varieties; the hope is then to reuse what we learn by studying the power of programs over monoids taken from these "small" varieties when conducting that study with bigger varieties like those given just above. The study of the expressiveness of programs over "small" varieties of finite monoids also leads to questions interesting in their own right, often related to algebraic automata theory and logic. Several previous works have followed that line of research (e.g. Barrington et al. [1990], Gavaldà and Thérien [2003], Lautemann et al. [2006], Maciel et al. [2000], McKenzie et al. [1991], Tesson and Thérien [2001]).

The second contribution of this Ph.D. thesis is twofold: at a general level, the investigation of a general property of the class of regular languages contained in $\mathcal{P}(\mathbf{V})$ for any variety of finite monoids $\mathbf{V}$; at a more specific level, the study of each of the case $\mathbf{V} = \mathbf{DA}$ and $\mathbf{V} = \mathbf{J}$, two well-known subvarieties of that of finite aperiodic monoids, important in algebraic automata theory and related fields. Exactly characterising the class of regular languages in $\mathcal{P}(\mathbf{V})$ as $\mathbf{V}$ ranges over all possible varieties of finite monoids is a fundamental task because, as shown in McKenzie et al. [1991], two classes $\mathcal{P}(\mathbf{V})$ and $\mathcal{P}(\mathbf{W})$ are equal if and only if they contain exactly the same regular languages. Drawing inspiration from the work of Péladeau, Straubing and Thérien Péladeau et al. [1997] for programs over semigroups taken from finite semigroup varieties of a certain form, a new notion of *tameness* of a variety of finite monoids $\mathbf{V}$ is introduced, that basically captures the property that, for monoids stemming from this variety, recognition by polynomial-length programs does not allow to recognise unexpectedly more regular languages than classical morphism recognition.

The rest of the thesis then first shows that $\mathbf{DA}$ is tame, thus deriving an exact algebraic

characterisation of the class of regular languages belonging to $\mathcal{P}(\mathbf{DA})$. Additionally, other properties of independent interest concerning $\mathcal{P}(\mathbf{DA})$ are shown. The thesis at last turns to $\mathbf{J}$, the variety of so-called finite $\mathfrak{J}$-trivial monoids, a notable example of a variety of finite monoids which is proven not to be tame. This means, most importantly, that polynomial-length programs over monoids in $\mathbf{J}$ can recognise "much more" regular languages than when only considering classical morphism recognition through monoids in $\mathbf{J}$. The sizeable last chapter in this thesis develops (yet unpublished) partial results towards an algebraic characterisation of the class of regular languages belonging to $\mathcal{P}(\mathbf{J})$ (along with some distinct small results about $\mathcal{P}(\mathbf{J})$).

Outline of the thesis. This manuscript is organised as follows. This introduction is followed by Chapter 1, whose aim is to introduce the basic notions used throughout the thesis, give a brief history of computational complexity theory and precisely set the scientific context of the thesis. Chapter 2 is dedicated to the work on Nečiporuk's lower bound method. Chapter 3 then moves on to give the necessary background in algebraic automata theory before introducing the program-over-monoid formalism, giving some general properties about it and examining its behaviour with respect to regular languages, notably introducing the notion of tameness of a variety of finite monoids. Finally, building on this, Chapter 4 and Chapter 5 deal with the computational power of polynomial-length programs over monoids taken from $\mathbf{DA}$ and $\mathbf{J}$, respectively. The thesis concludes with a summary of the findings it presents and a list of ideas for future directions along the lines of the work it reports.

Publications. The present Ph.D. thesis is partly based on two scientific publications:

- Beame et al. [2016], a journal article published in ACM ToCT the thesis' author co-wrote with Paul Beame and his two Ph.D. advisors, Pierre McKenzie and Luc Segoufin, that in overwhelming majority corresponds to the contents of Chapter 2 as is;
- Grosshans et al. [2017], a conference article presented at the MFCS 2017 conference the thesis' author co-wrote with his two Ph.D. advisors, that forms the basis of Subsection 3.4.2 in Chapter 3 and of Chapter 4.

Chapter 1

Languages and computation

In this first chapter, we set the necessary mathematical background on which this Ph.D. thesis relies as a whole, formalising most of the concepts mentioned in the introduction. We then give a short history of the field of computational complexity theory, laying out the context in which this thesis fits.

1.1 Some mathematical conventions

We start reviewing some conventions and notations that we will use throughout this thesis.

Sets. We will denote by $\mathbb{N}$ the set of all natural numbers (including 0), by $\mathbb{Z}$ the set of all integers and $\mathbb{R}$ the set of all real numbers. For E any of these sets, $E_{>0}$ shall denote the set of positive numbers in E and $E_{\geq 0}$ the set of non-negative numbers in E. Given some positive natural number d, we shall denote by $\mathbb{Z}/d\mathbb{Z} = \{0, \dots, d-1\}$ the set of integers modulo d. For $n, m \in \mathbb{N}$, we denote by $[\![n, m]\!]$ the set of natural numbers from n to m, which is empty if $m < n$. We also denote by $[n]$ the set of integers from 1 to n, i.e. $[\![1, n]\!]$, using the convention that $[0] = \emptyset$. For some set E, we will denote by $\mathfrak{P}(E)$ the powerset of E, i.e. the set of all subsets of E. For two sets E_1 and E_2, $E_1 \subset E_2$ shall denote strict inclusion, and $E_1 \subseteq E_2$ inclusion or equality.

Indexed families. Let V be a subset of $\mathbb{N}$. We view $u \in E^V$ as a sequence $(u_i)_{i \in V}$ of elements $u_i \in E$ for all $i \in V$. For all $n \in \mathbb{N}$, we won't also make the difference between E^n and $E^{[n]}$.

Functions. For X some set, we shall denote by $\mathrm{id}_X \colon X \to X$ the identity function on X, i.e. defined by $\mathrm{id}_X(x) = x$ for all $x \in X$. Given a function $f \colon X \to Y$ from a set X to a set Y and a subset $S \subseteq X$ of X, we will denote by $f|_S \colon S \to Y$ the restriction of f to S, defined by $f|_S(x) = f(x)$ for all $x \in S$, and by $f(S) = \{f(x) \mid x \in S\}$ the image of S by f. We shall also denote by $\mathfrak{Im}(f)$ the image of E by the function f, i.e. $f(X)$. Given $T \subset Y$, we will denote by $f^{-1}(T) = \{x \in X \mid f(x) \in T\}$ the inverse image of T by f and shall write $f^{-1}(y)$ when T is the singleton $\{y\}$.

Given two functions $f \colon X \to Y$ and $g \colon Y \to Z$ where X, Y and Z are sets, we will denote by $g \circ f \colon X \to Z$ the function obtained by composition of f and g, defined by $(g \circ f)(x) = g(f(x))$ for all $x \in X$. When $X = Y$, for each $n \in \mathbb{N}$, we shall denote by $f^n \colon X \to X$ the composition of n copies of f, defined by

$$f^n = \begin{cases} f^{n-1} \circ f & \text{if } n > 0 \\ \mathrm{id}_E & \text{if } n = 0 \end{cases}.$$

For all $b \in \mathbb{R}_{>0}$, the base b logarithm function will be denoted by $\log_b$.

Equivalence relations. Given an equivalence relation $\mathcal{E} \subseteq E \times E$ on a set E, we will denote by $[e]_{\mathcal{E}} = \{e' \in E \mid e \ \mathcal{E} \ e'\}$ the equivalence class of $e \in E$ relative to $\mathcal{E}$ and $E/\mathcal{E} = \{[e]_{\mathcal{E}} \mid e \in E\}$ the quotient of E by $\mathcal{E}$ (the set of equivalence classes in E relative to $\mathcal{E}$).

Landau symbols. We will also use the following common Landau symbols. Let $f \colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ and $g \colon \mathbb{N} \to \mathbb{R}_{\geq 0}$, we will write:

- $f(n) \in \mathrm{O}(g(n))$ if and only if there exist $\alpha \in \mathbb{R}_{>0}$ and $n_0 \in \mathbb{N}$ such that for all $n \in \mathbb{N}, n \geq n_0$ we have $f(n) \leq \alpha \cdot g(n)$;
- $f(n) \in \mathrm{o}(g(n))$ if and only if for all $\alpha \in \mathbb{R}_{>0}$, there exists $n_0 \in \mathbb{N}$ such that for all $n \in \mathbb{N}, n \geq n_0$ we have $f(n) \leq \alpha \cdot g(n)$;
- $f(n) \in \Omega(g(n))$ if and only if there exist $\alpha \in \mathbb{R}_{>0}$ and $n_0 \in \mathbb{N}$ such that for all $n \in \mathbb{N}, n \geq n_0$ we have $f(n) \geq \alpha \cdot g(n)$;
- $f(n) \in \omega(g(n))$ if and only if for all $\alpha \in \mathbb{R}_{>0}$, there exists $n_0 \in \mathbb{N}$ such that for all $n \in \mathbb{N}, n \geq n_0$ we have $f(n) \geq \alpha \cdot g(n)$.

Given an additional function $h \colon \mathbb{N} \to \mathbb{R}_{\geq 0}$, we shall abuse notation by writing $f(n) \in g(n)^{\mathrm{O}(h(n))}$, $f(n) \in g(n)^{\mathrm{o}(h(n))}$, $f(n) \in g(n)^{\Omega(h(n))}$ or $f(n) \in g(n)^{\omega(h(n))}$ to express that

there exists a function $h' \colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ respectly verifying $h'(n) \in \mathrm{O}(h(n))$, $h'(n) \in \mathrm{o}(h(n))$, $h'(n) \in \Omega(h(n))$ or $h'(n) \in \omega(h(n))$ such that $f(n) = g(n)^{h'(n)}$ for all $n \in \mathbb{N}$.

1.2 Alphabets, words and languages

Let us start with the basics about alphabets, words and languages.

Alphabets and words The basic building block of languages is that of a letter, taken from some alphabet.

Definition 1.2.1. An *alphabet* is a finite set of symbols, called *letters*.

Given an alphabet, we can define words by assembling letters from this one in a sequence.

Definition 1.2.2. Let Σ be an alphabet.

A *word w over (or on) Σ* is a finite sequence of elements of Σ, i.e. an element of Σ^k for some $k \in \mathbb{N}$, and we say k is the *length of the word w*, written $|w|$. When $k = 0$, w is the *empty word*, denoted by ε, the only element in Σ^0, i.e. the empty sequence. Otherwise, when $k \in \mathbb{N}_{>0}$, instead of $(w_1, \ldots, w_k)$, we usually write $w_1 \cdots w_k$.

We denote by Σ^* the *set of all words over* Σ, i.e. $\Sigma^* = \bigcup_{k \in \mathbb{N}} \Sigma^k$, and by Σ^+ the *set of all non-empty words over* Σ, i.e. $\Sigma^+ = \bigcup_{k \in \mathbb{N}_{>0}} \Sigma^k$. For each $l \in \mathbb{N}$, we also denote by $\Sigma^{\leq l}$ the *set of all words over Σ of length at most l*, i.e. $\Sigma^{\leq l} = \bigcup_{k \in \mathbb{N}, k \leq l} \Sigma^k$.

Words can be combined by the operation of concatenation, that glues them together, to obtain new words.

Definition 1.2.3. Let Σ be an alphabet. For words $u \in \Sigma^k$ and $v \in \Sigma^l$ $(k, l \in \mathbb{N})$, we denote by $u \cdot v$ or uv the word $u \cdot v = u_1 \cdots u_k v_1 \cdots v_l$ in Σ^{k+l} which is the *concatenation of u and v*.

We might from time to time need to talk about the letters that appear specifically in a given word over some alphabet, and how many times each of those appears.

Definition 1.2.4. Let Σ be an alphabet. For a word $w \in \Sigma^*$, we will denote by $\mathrm{alph}(w)$ the *alphabet of w*, i.e. the set of letters of Σ that appear in w. Given additionally some specific letter $a \in \Sigma$, we will denote by $|w|_a$ the *number of occurrences of the letter a in w*.

Finally, another set of important notions is those talking about the different ways some word can appear in another one, formalised below.

Definition 1.2.5. Let Σ be an alphabet and $u, v \in \Sigma^*$ two words over Σ.

- We say that u *is a subword of* v if and only if v can be written as $v = v_0 a_1 v_1 \cdots a_l v_l$ where $l \in \mathbb{N}$, $v_i \in \Sigma^*$ for all $i \in [\![0, l]\!]$ and $u = a_1 \cdots a_l$ with $a_i \in \Sigma$ for all $i \in [l]$. For all $k \in \mathbb{N}, k \geq l$, we also say that u *is a* k*-subword of* v.
- We say that u *is a factor of* v if and only if v can be written as $v = v_0 u v_1$ where $v_0, v_1 \in \Sigma^*$. Additionally, we say that u *is a prefix of* v if and only if $v_0 = \varepsilon$, and similarly we say that u *is a suffix of* v if and only if $v_1 = \varepsilon$.

Languages Now that we have a notion of words, we can create collections of them to form languages.

Definition 1.2.6. Let Σ be an alphabet. A *language* L *over (or on)* Σ is a set of words over Σ, i.e. $L \subseteq \Sigma^*$. For any $n \in \mathbb{N}$ we define $L^{=n} = \{w \in L \mid |w| = n\}$ the set of words (or finite language of words) of L of length n. Finally, we denote by $\chi_L \colon \Sigma^* \to \{0, 1\}$ the *indicator function of* L, such that for all $w \in \Sigma^*$, $\chi_L(w) = 1$ if and only if $w \in L$.

Remark 1.2.7. For a given alphabet Σ, we shall call the empty language $\emptyset$ and the full language Σ^* over Σ the *trivial languages over* Σ. Furthermore, given $u \in \Sigma^*$, we shall sometimes, when the context is clear, write u for $\{u\}$ (the language composed solely of the word u).

Remark 1.2.8. We also could have chosen to have infinite alphabets or infinite words (or both), but in this work, both will always be finite. In contrast, languages can be both finite or infinite — and, in fact, we are usually interested in infinite languages.

A set of basic operations on languages to form new ones is the set of so-called *Boolean operations*.

Definition 1.2.9. Let Σ be an alphabet.

- Let $L \subseteq \Sigma^*$ be a language over Σ. The *complement of* L (in Σ, but this is usually clear from the context), denoted by $L^{\complement}$, is the language $L^{\complement} = \Sigma^* \setminus L$ over Σ.
- Let $L_1, L_2 \subseteq \Sigma^*$ be two languages over Σ. The *union (or sum) of* L_1 *and* L_2, denoted by $L_1 \cup L_2$ or $L_1 + L_2$, is simply the union of the two sets L_1 and L_2. Similarly the *intersection of* L_1 *and* L_2, denoted by $L_1 \cap L_2$, is simply the intersection of the two sets L_1 and L_2.

- Let $L_1, \ldots, L_k \in \Sigma^*$ be $k \in \mathbb{N}_{>0}$ languages over Σ. A *Boolean combination of the languages* $L_1, \cdots, L_k$ is a language over Σ obtained by a finite combination of unions, intersections or complements of these languages.

Another two basic operations operating on languages is the concatenation product and the associated star operation.

Definition 1.2.10. Let Σ be an alphabet.

- Let $L_1, L_2 \subseteq \Sigma^*$ be two languages over Σ. The *concatenation product (or product) of* L_1 *and* L_2, denoted by $L_1 \cdot L_2$ or $L_1 L_2$, is the language $L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1, w_2 \in L_2\}$ over Σ.

- Let $L \subseteq \Sigma^*$ be a language over Σ. For all $k \in \mathbb{N}$, the *k-th power of L*, denoted by L^k, is the language $L^k = \underbrace{L \cdot L \cdot \cdots \cdot L}_{k \text{ times}}$ over Σ, using the convention that $L^0 = \{\varepsilon\}$.

- Let $L \subseteq \Sigma^*$ be a language over Σ. The *star of L*, denoted by L^*, is the language $L^* = \bigcup_{k \in \mathbb{N}} L^k$ over Σ. We also set $L^+ = \bigcup_{k \in \mathbb{N}_{>0}} L^k$.

The last basic operation we may introduce is the quotient.

Definition 1.2.11. Let Σ be an alphabet. Let $L \subseteq \Sigma^*$ be a language over Σ. Given some word $u \in \Sigma^*$, the *left (right) quotient of L by u*, denoted by $u^{-1}L$ (Lu^{-1}), is the language $u^{-1}L = \{v \in \Sigma^* \mid uv \in L\}$ ($Lu^{-1} = \{v \in \Sigma^* \mid vu \in L\}$) over Σ. For any subset $K \subseteq \Sigma^*$ of words on Σ, the *left (right) quotient of L by K*, denoted by $K^{-1}L$ (LK^{-1}), is the language $K^{-1}L = \bigcup_{u \in K} u^{-1}L$ ($LK^{-1} = \bigcup_{u \in K} Lu^{-1}$) over Σ.

1.3 Models of computation and complexity classes

We now proceed to the definition of the main models of computation and associated complexity classes used throughout this manuscript. The main references used here are the book by Arora and Barak Arora and Barak [2009] and its French twin by Perifel Perifel [2014].

1.3.1 Turing machines

For a formal definition of Turing machines, of computations performed on such machines and of the notions of time and space taken by such computations, we refer the reader

to Perifel [2014]. We shall abbreviate "deterministic Turing machine" by DTM and "non-deterministic Turing machine" by NTM.

We define the usual Turing machine time and space complexity classes as follows. Let $f\colon \mathbb{N} \to \mathbb{N}$, we define

- $\mathsf{DTIME}(f(n))$ ($\mathsf{NTIME}(f(n))$) as the class of all languages over any alphabet Σ that are decided by a DTM (an NTM) running in time at most $\alpha \cdot f(|w|)$ (for all possible executions) on any input $w \in \Sigma^*$, $\alpha \in \mathbb{R}_{>0}$ being some constant depending only on the DTM (the NTM);
- $\mathsf{DSPACE}(f(n))$ ($\mathsf{NSPACE}(f(n))$) as the class of all languages over any alphabet Σ that are decided by a DTM (an NTM) halting and running in space at most $\alpha \cdot f(|w|)$ (for all possible executions) on any input $w \in \Sigma^*$, $\alpha \in \mathbb{R}_{>0}$ being some constant depending only on the DTM (the NTM).

We can now define the following well-known Turing machine complexity classes.

- The class of *deterministic polynomial time* $\mathsf{P} = \bigcup_{k\in\mathbb{N}} \mathsf{DTIME}(n^k)$.
- The class of *non-deterministic polynomial time* $\mathsf{NP} = \bigcup_{k\in\mathbb{N}} \mathsf{NTIME}(n^k)$.
- The class of *deterministic logarithmic space* $\mathsf{L} = \mathsf{DSPACE}(\log_2(n))$.
- The class of *non-deterministic logarithmic space* $\mathsf{NL} = \mathsf{NSPACE}(\log_2(n))$.

The models of DTMs and NTMs are *uniform* models of computation, because each Turing machine works on input words of any length and to decide a language, one should give a TM that works for all input lengths. As we shall see in the next subsection, there also exist *non-uniform* models of computation where each associated computational strategy works only on input words of a fixed length and to decide a language, one should give a sequence of such computational strategies, one working for each possible input length. To make the models based on Turing machines non-uniform, we can define the notion of classes with advice.

Definition 1.3.1. Let $f\colon \mathbb{N} \to \mathbb{N}$ be a function and C a complexity class. Then we denote $\mathsf{C}/f(n)$ the class of languages L over any alphabet Σ such that there exists a language K over the alphabet $\Sigma \cup \{\#\}$ (where $\#$ is a new letter) and a sequence of words $(u_n)_{n\in\mathbb{N}}$ verifying that:

- $u_n \in \{0,1\}^{f(n)}$ for all $n \in \mathbb{N}$;

- for all $w \in \Sigma^*$, $w \in L$ if and only if $w\#u_{|w|} \in K$.

Of particular importance in our case will be the classes with polynomial length advice.

Definition 1.3.2. Let C be a complexity class. Then $\mathsf{C}/\mathsf{poly} = \bigcup_{c,k\in\mathbb{N}} \mathsf{C}/(c + c \cdot n^k)$.

1.3.2 Circuits

The most important and common non-uniform models of computation are those based on circuits.

Definition 1.3.3. Let Σ be an alphabet and Ω be a set of Boolean functions of the form $g\colon \{0,1\}^m \to \{0,1\}$ with $m \in \mathbb{N}_{>0}$. A *circuit over basis* Ω *(or* Ω*-circuit) on* Σ^V, for some finite set $V \subseteq \mathbb{N}$, is a directed acyclic graph (whose vertices are also called *gates*) where

- there is a single sink vertex (also called *output gate*);
- every source vertex (also called *input gate*) is labelled by "$x_i \in \Gamma$" for some $i \in V$ and $\Gamma \subseteq \Sigma$, or by one of the constants 0 and 1;
- every non-source vertex (also called *internal gate*) is labelled by some Boolean function $g\colon \{0,1\}^m \to \{0,1\}$ ($m \in \mathbb{N}_{>0}$) from Ω and has exactly m predecessors, given in a certain order.

The *size of* C, denoted by $|C|$, is the number of its non-source vertices, while the *depth of* C, denoted by $\mathrm{depth}(C)$, is the length of the longest path from an input gate to the output gate.

Given a non-output gate γ in C, we will call *induced subcircuit of* C *at gate* γ the circuit over Ω on Σ^V obtained from C by setting γ as the output gate and keeping only the gates of C from which there exists a path to γ.

For each gate γ in C, we define $f_\gamma\colon \Sigma^V \to \{0,1\}$ the function computed at gate γ in the following way. For each $w \in \Sigma^V$:

- if γ is an input gate labelled by "$x_i \in \Gamma$" ($i \in V$ and $\Gamma \subseteq \Sigma$), then $f_\gamma(w) = 1$ if and only if $w_i \in \Gamma$, otherwise if γ is labelled by $b \in \{0,1\}$, then $f_\gamma(w) = b$;
- if γ is an internal gate labelled by $g\colon \{0,1\}^m \to \{0,1\}$ ($m \in \mathbb{N}_{>0}$) and having $\gamma_1, \ldots, \gamma_m$ as predecessors, then $f_\gamma(w) = g(f_{\gamma_1}(w), \ldots, f_{\gamma_m}(w))$.

C computes the function $f_o\colon \Sigma^V \to \{0,1\}$ where o is the output gate of C.

Below we define how circuits decide languages and define complexity classes based on the size and depth computational costs.

Definition 1.3.4. Let Σ be an alphabet. Let $L \subseteq \Sigma^*$ be a language on Σ. We say that a sequence $(C_n)_{n\in\mathbb{N}}$ of Ω-circuits (for some basis Ω) such that each C_n is on Σ^n for $n \in \mathbb{N}$ decides L if and only if C_n computes $\chi_L|_{\Sigma^n}$ for all $n \in \mathbb{N}$.

Let Ω be some basis, as well as $s\colon \mathbb{N} \to \mathbb{N}$ and $d\colon \mathbb{N} \to \mathbb{N}$ two functions. We define $\mathsf{SIZE}\text{-}\mathsf{DEPTH}_\Omega(s(n), d(n))$ ($\mathsf{SIZE}_\Omega(s(n))$, $\mathsf{DEPTH}_\Omega(d(n))$) as the class of languages L over any alphabet Σ such that there exist a constant $\alpha \in \mathbb{R}_{>0}$ and a sequence of Ω-circuits $(C_n)_{n\in\mathbb{N}}$ deciding L, where for all $n \in \mathbb{N}$, $|C_n| \leq \alpha \cdot s(n)$ and $\text{depth}(C_n) \leq \alpha \cdot d(n)$ ($|C_n| \leq \alpha \cdot s(n)$, $\text{depth}(C_n) \leq \alpha \cdot d(n)$).

Let us now define some specific circuit complexity classes that will be referred to in this manuscript. But first, we need to define some useful bases of Boolean functions.

Definition 1.3.5. We will denote by $\neg$ the Boolean NOT function and for all $m \in \mathbb{N}, m \geq 2$, $\wedge_m$ will denote the Boolean AND function on m inputs, and $\vee_m$ the Boolean OR function on m inputs. Moreover, for all $m, q \in \mathbb{N}, m, q \geq 2$, $\equiv_m^q$ will denote the boolean MOD-q function on m inputs, that is equal to 1 if and only if the number of inputs that are 1 is not divisible by m.

We shall use three quite common bases:

- the basis $\mathcal{B}_2$ containing only NOT and the fan-in 2 OR and AND, i.e. $\mathcal{B}_2 = \{\neg, \wedge_2, \vee_2\}$;
- the basis $\mathcal{B}_\infty$ containing NOT and arbitrary fan-in ORs and ANDs, i.e. $\mathcal{B}_\infty = \{\neg\} \cup \{\wedge_m \mid m \in \mathbb{N}, m \geq 2\} \cup \{\vee_m \mid m \in \mathbb{N}, m \geq 2\}$;
- for any $q \in \mathbb{N}, q \geq 2$, the basis $\mathcal{C}_{\infty,q}$ containing NOT and arbitrary fan-in ORs, ANDs and MOD-qs, i.e. $\mathcal{C}_{\infty,q} = \mathcal{B}_\infty \cup \{\equiv_m^q \mid m \in \mathbb{N}, m \geq 2\}$.

Definition 1.3.6. We define the following circuit complexity classes.

- For all $i \in \mathbb{N}$,
$$\mathsf{NC}^i = \bigcup_{k\in\mathbb{N}} \mathsf{SIZE}\text{-}\mathsf{DEPTH}_{\mathcal{B}_2}(n^k, \log_2{}^i(n))$$
and $\mathsf{NC} = \bigcup_{i\in\mathbb{N}} \mathsf{NC}^i$.

- For all $i \in \mathbb{N}$,
$$\mathsf{AC}^{\mathsf{i}} = \bigcup_{k \in \mathbb{N}} \mathsf{SIZE}\text{-}\mathsf{DEPTH}_{\mathcal{B}_\infty}(n^k, \log_2{}^i(n))$$
and $\mathsf{AC} = \bigcup_{i \in \mathbb{N}} \mathsf{AC}^{\mathsf{i}}$.

- For all $i \in \mathbb{N}$ and $q \in \mathbb{N}, q \geq 2$,
$$\mathsf{ACC}^{\mathsf{i}}[q] = \bigcup_{k \in \mathbb{N}} \mathsf{SIZE}\text{-}\mathsf{DEPTH}_{\mathcal{C}_{\infty,q}}(n^k, \log_2{}^i(n)) \ ,$$
$\mathsf{ACC}^{\mathsf{i}} = \bigcup_{q \in \mathbb{N}, q \geq 2} \mathsf{ACC}^{\mathsf{i}}[q]$ and $\mathsf{ACC} = \bigcup_{i \in \mathbb{N}} \mathsf{ACC}^{\mathsf{i}}$.

It is a classical result that

$$\mathsf{NC}^0 \subseteq \mathsf{AC}^0 \subseteq \mathsf{ACC}^0 \subseteq \mathsf{NC}^1 \subseteq \cdots \subseteq \mathsf{NC}^{\mathsf{i}} \subseteq \mathsf{AC}^{\mathsf{i}} \subseteq \mathsf{ACC}^{\mathsf{i}} \subseteq \mathsf{NC}^{\mathsf{i}+1} \subseteq \cdots \subseteq \mathsf{NC} = \mathsf{AC} = \mathsf{ACC}$$

for all $i \in \mathbb{N}$. Nevertheless, while all the inclusions in this infinite chain are widely conjectured to be strict, the only which are actually proven to be so are the first two, as well as the third one in the specific case of $\mathsf{ACC}^0[p]$ for $p \in \mathbb{N}$ some prime, using the following results.

For all $m \in \mathbb{N}, m \geq 2$, let

$$\mathrm{MOD}_{\mathrm{m}} = \{w \in \{0,1\}^* \mid |w|_1 \neq 0 \mod m\}$$

be the language on the alphabet $\{0,1\}$ of words containing a number of 1's not congruent to 0 modulo m.

Theorem 1.3.7 (Furst et al. [1984], Ajtai [1983])**.** *For all $m \in \mathbb{N}, m \geq 2$, $\mathrm{MOD}_{\mathrm{m}} \notin \mathsf{AC}^0$.*

Theorem 1.3.8 (Razborov [1987], Smolensky [1987])**.** *Let $p \in \mathbb{N}$ and $q \in \mathbb{N}$ be two distinct primes. Then $\mathrm{MOD}_{\mathrm{p}} \notin \mathsf{ACC}^0[q]$.*

The size of circuits over the basis $\mathcal{B}_2$ (or even any other basis including $\mathcal{B}_2$) is an important computational cost measure, as it captures the measure of time on DTMs, as made precise below.

Proposition 1.3.9 ([Perifel, 2014, Proposition 5-Y])**.** *For all $t\colon \mathbb{N} \to \mathbb{N}$,*

$$\mathsf{DTIME}(t(n)) \subseteq \mathsf{SIZE}_{\mathcal{B}_2}(t(n)^2) \ .$$

Proposition 1.3.10 ([Arora and Barak, 2009, Theorem 6.18], [Perifel, 2014, Proposition 5-AC]). *We have*

$$\mathsf{P/poly} = \bigcup_{k\in\mathbb{N}} \mathsf{SIZE}_{\mathcal{B}_2}(n^k) \ .$$

1.3.3 Branching programs

Other common non-uniform models of computation are those based on branching programs.

Definition 1.3.11. Let Σ be a finite alphabet. A *non-deterministic branching program (NBP)* on Σ^V, for a finite set V, is a tuple $P = (X, \delta, \tau, s, t_0, t_1)$ where

- X is a finite set of vertices (or states);
- $s \in X$ is the start (or source) vertex;
- $t_0, t_1 \in X$, $t_0 \neq t_1$ are two distinct sink vertices;
- $\delta\colon X \setminus \{t_0, t_1\} \times \Sigma \to \mathfrak{P}(X \setminus \{s\})$ gives the transition rules (for each non-sink vertex and letter of the alphabet, the set of successors it is linked to thanks to arcs labelled with this letter);
- $\tau\colon X \setminus \{t_0, t_1\} \to V$ labels each non-sink vertex.

The *size of* P, denoted by $|P|$, is the number of its non-sink vertices, that is to say $|P| = |X \setminus \{t_0, t_1\}|$.

Each assignment $w \in \Sigma^V$ defines a set of arcs

$$A[w] = \{(u, v) \in X \setminus \{t_0, t_1\} \times X \setminus \{s\} \mid v \in \delta(u, w_{\tau(u)})\}$$

and thus a directed graph $P[w] = (X, A[w])$. P computes a function $f\colon \Sigma^V \to \{0, 1\}$ given by $f(w) = 1$ if and only if there exists a path (*computation*) in $P[w]$ from state s to state t_1.

P is a *deterministic branching program (BP)* if and only if the underlying directed graph

$$(X, \{(u, v) \in X \setminus \{t_0, t_1\} \times X \setminus \{s\} \mid \exists a \in \Sigma, v \in \delta(u, a)\})$$

is acyclic and each non-sink vertex of X has precisely one out-arc labelled by each possible letter of Σ, i.e. $|\delta(u, a)| = 1$ for all $u \in X \setminus \{t_0, t_1\}$ and $a \in \Sigma$.

As for the case of circuits, we can now define how BPs and NBPs decide languages and define complexity classes based on the size computational cost measure.

Definition 1.3.12. For all finite alphabet Σ and function $f\colon \Sigma^V \to \{0,1\}$ (for $V \subseteq \mathbb{N}$ finite) we denote by $\mathbf{BP}(f)$ and $\mathbf{NBP}(f)$ the minimum size of, respectively, a BP and an NBP over Σ^V computing f.

Let $L \subseteq \Sigma^*$ be a language on Σ. We say that a sequence $(P_n)_{n\in\mathbb{N}}$ of BPs (NBPs) such that each P_n is on Σ^n for $n \in \mathbb{N}$ decides L if and only if P_n computes $\chi_L|_{\Sigma^n}$ for all $n \in \mathbb{N}$.

For all $s\colon \mathbb{N} \to \mathbb{N}$, we define $\mathsf{BPSIZE}(s(n))$ ($\mathsf{NBPSIZE}(s(n))$) as the class of languages L over any alphabet Σ such that exist a constant $\alpha \in \mathbb{R}_{>0}$ and a sequence of BPs (NBPs) $(P_n)_{n\in\mathbb{N}}$ deciding L, where for all $n \in \mathbb{N}$, $|P_n| \leq \alpha \cdot s(n)$. We set $\mathsf{PBP} = \bigcup_{k\in\mathbb{N}} \mathsf{BPSIZE}(n^k)$ and $\mathsf{PNBP} = \bigcup_{k\in\mathbb{N}} \mathsf{NBPSIZE}(n^k)$.

Similarly to the case of the size of circuits over the basis $\mathcal{B}_2$ that captures the measure of time on DTMs, the size of BPs (NBPs) captures the measure of space on DTMs (NTMs).

Proposition 1.3.13 ([Arora and Barak, 2009, Theorem 14.13])**.** *For all $s\colon \mathbb{N} \to \mathbb{N}$ such that $s(n) \geq \log_2(n)$ for all $n \in \mathbb{N}_{>0}$,*

- $\mathsf{DSPACE}(s(n)) \subseteq \bigcup_{c\in\mathbb{N}} \mathsf{BPSIZE}(2^{c\cdot s(n)})$*;*
- $\mathsf{NSPACE}(s(n)) \subseteq \bigcup_{c\in\mathbb{N}} \mathsf{NBPSIZE}(2^{c\cdot s(n)})$*.*

Proposition 1.3.14 (Razborov [1991])**.** *We have the following equalities.*

- $\mathsf{L/poly} = \mathsf{PBP}$*.*
- $\mathsf{NL/poly} = \mathsf{PNBP}$*.*

1.4 The great quest: on the relationship between time and space and the power of non-determinism

1.4.1 Facing walls, or the embarrassing state of affairs in computational complexity theory

As we know since the very beginning of the field of computational complexity theory, which really started in 1965 (see Fortnow and Homer [2003], Perifel [2014]), under some conditions, the more time we have on a deterministic Turing machine, the more languages we can decide, and similarly, under some conditions, the more space we have on such a machine, the more languages we can decide (this is an informal presentation — but

sufficient for introductory purposes — of what is called, respectively, the deterministic time hierarchy theorem [Perifel, 2014, Theorem 2-J] and the deterministic space hierarchy theorem [Perifel, 2014, Theorem 4-N]). It is also well known that a deterministic Turing machine running in time $t(n)$ runs in space at most $t(n)$ and, conversely, that such a machine running in space $s(n) \geq \log_2(n)$ can be made to run in time at most $2^{\alpha \cdot s(n)}$ for some fixed constant $\alpha \in \mathbb{R}_{>0}$ depending only on the machine.

That being said, it is absolutely remarkable that after more than 50 years of continuous developments in computational complexity, we are not able to say anything more precise about the relationship between time and space complexities on deterministic Turing machines. In particular, while it is intuitive to think that there exist languages decidable in polynomial time in the size of the input on a deterministic Turing machine that cannot be decided using only a logarithmic amount of space in the size of the input (i.e. $\mathsf{L} \subset \mathsf{P}$), there exists no proof of that belief, which might as well be false.

The other huge wall faced by complexity theorists is trying to understand the influence of non-determinism on time and space complexities. Indeed, it seems natural to think that allowing a Turing machine to do non-deterministic transitions would allow it to save some time or space, in comparison to the case in which only deterministic transitions are allowed. Back in 1970, Savitch showed that the space a TM can save by making non-deterministic choices cannot be more than quadratic, i.e. $\mathsf{NSPACE}(s(n)) \subseteq \mathsf{DSPACE}(s(n)^2)$ for any $s\colon \mathbb{N} \to \mathbb{N}$ satisfying some mild conditions [Perifel, 2014, Theorem 4-AS]. This proves that non-determinism does not make such a huge difference for space complexity, and that we have for instance $\mathsf{PSPACE} = \bigcup_{k\in\mathbb{N}} \mathsf{DSPACE}(n^k) = \bigcup_{k\in\mathbb{N}} \mathsf{NSPACE}(n^k) = \mathsf{NPSPACE}$. However, we do not know whether this inclusion is optimal, that is to say, it is not excluded that for some $s\colon \mathbb{N} \to \mathbb{N}$, we have $\mathsf{NSPACE}(s(n)) \subseteq \mathsf{DSPACE}(s'(n))$ for some $s'\colon \mathbb{N} \to \mathbb{N}$ verifying $s'(n) \in \mathrm{o}(s(n)^2)$: this means that even 48 years after the publication of Savitch's result, we still have some interesting and difficult open questions about the influence of non-determinism on space complexity, especially in the case of logarithmic space bounded TMs (i.e. L vs NL). In contrast, when it comes to time complexity, we know almost nothing about the influence of non-determinism, excepting the elementary fact that $\mathsf{NTIME}(t(n)) \subseteq \bigcup_{c\in\mathbb{N}} \mathsf{DTIME}(2^{c\cdot t(n)})$ for any $t\colon \mathbb{N} \to \mathbb{N}$ [Perifel, 2014, Proposition 2-AG]. The state of affairs in the case of the relationship between deterministic and non-deterministic time complexities on Turing machines is similar to the one in the case of deterministic time and space, and the most famous open question in computational complexity theory, the relationship between polynomial deterministic and non-deterministic time on Turing machines (or the P versus NP

question) seems to be, for all we already know about P and NP, a deep mathematical question Wigderson [2006], Allender [2009].

1.4.2 Finding small breaches in the walls, or how the field did evolve

While the fundamental questions above still remain unanswered, the field of computational complexity evolved much during its slightly more than 50 years of existence.

The focus in the beginning, until the end of the 1970s, was essentially directly on time and space complexities on deterministic and non-deterministic Turing machines; this led to results such as the aforementioned hierarchy theorems and, for lack of other separation results between complexity classes, to the first results of the immensely fruitful theory of resource bounded reducibility among languages and completeness within complexity classes (the most important and significant case being that of NP-completeness: see, e.g., Wigderson [2006], Allender [2009]). It soon became obvious, supported by a seminal 1975 result by Baker, Gill and Solovay [Perifel, 2014, Theorem 7-AA] and subsequent results such as the one by Ladner and Lynch in 1976 Ladner and Lynch [1976], that the diagonalisation techniques used so far, inherited from computability theory, and that handle TMs merely as black boxes, would be of no use to solve some of the central questions in computational complexity theory, among them the P versus NP and L versus P questions. It was therefore necessary to develop techniques that would analyse the precise internal working of Turing machines, but this is, as Perifel Perifel [2014] states it, rather tricky due to the complicated definition of a Turing machine.

This is where the close relationship between time on DTMs and size of circuits, established in 1972 by Savage Savage [1972], as well as the close relationship between space on Turing machines and size of various types of branching programs, first observed in 1976 by Masek Masek [1976] for the deterministic case, step in. Indeed, those relationships show that proving lower bounds for the circuit-type and branching program-type complexity measures would imply lower bounds for time and space Turing machine complexities, and the fact is that these models of computation are somewhat easier to handle structurally than TMs and are hence better suited for the precise internal analysis mentioned earlier. But, as Boppana and Sipser state it in their 1989 survey Boppana and Sipser [1990], this approach through non-uniform models of computation to proving lower bounds for Turing machine complexity measures first did not help much, because proving size lower bounds for general circuit-type or branching program-type models of computation appears to be a difficult task, and the best lower bounds known at the end of the 1970s (which did, by

the way, not evolve much until the time of redaction of this manuscript) were quite weak.

This approach started to really bear fruit in the 1980s when computational complexity theorists started to consider restricted types of circuit or branching program models of computation. Researchers were able to develop specific methods to prove significant lower bounds on the sizes of sequences of circuits on which they placed restrictions such as (see this survey Boppana and Sipser [1990] by Boppana and Sipser):

- requiring the depth of the circuits in the sequence to stay constant (those correspond to the classes AC^0 and $\mathsf{ACC}^0[q]$ for $q \in \mathbb{N}, q \geq 2$);
- in the case of Boolean circuits (i.e. operating on $\{0,1\}$), requiring that the basis at use does not contain NOT (those are called monotone Boolean circuits);
- requiring that the underlying graphs of the circuits in the sequence are all trees (those are called formulæ).

Similarly, researchers also developed specific methods to prove significant lower bounds on the sizes of sequences of branching programs on which they placed restrictions such as (see this survey Razborov [1991] by Razborov):

- requiring that each branching program in the sequence can have its set of vertices partitioned into levels whose size does not exceed a constant (those are called bounded-width branching programs);
- requiring that each branching program in the sequence can have its set of vertices partitioned into levels, all vertices in one level being labelled with the same input index (those are called oblivious branching programs);
- requiring that each branching program in the sequence verifies that any consistent computation path from the start to a source vertex queries each input variable at most once (those are called read-once branching programs).

In parallel, complexity theorists also started to introduce and study more and more new types of computational models and associated complexity measures: this led to the development of several new threads in computational complexity theory, sometimes deeply linked with (or even rooted in) other domains in mathematics, that brought forth plenty of new questions and further shaped the complicated and rich landscape of complexity classes, but also gave novel insights into central complexity classes. Some of these threads include the following — a much more precise account of them, as well as many other

threads, can be found in classical references about computational complexity theory Arora and Barak [2009], Perifel [2014], Jukna [2012] or historical surveys about the field Fortnow and Homer [2003], Impagliazzo [2005].

- The study of the power of randomness as a computational resource, just like for the case of non-determinism. Or, does allowing a Turing machine to do probabilistic transitions allow it to save some time or space, in comparison to the case in which only deterministic transitions are allowed? How efficiently can we remove the use of randomness in computation?

- The study of the complexity of communication protocols, models of computation where two or more players, each having a distinct input, communicate following a given protocol to compute a function of all of the players' inputs, the complexity of a protocol being measured as the total amount of communication.

- The study of the power of adding quantum computation abilities to basically all models of computation presented so far. The central question, in each case and for each complexity measure, is how the quantum variant compares to the "classical" variants (deterministic, non-deterministic, randomised, etc.) in terms of complexity.

- Descriptive complexity, that seeks to find out how expressive a logic must be so as to express membership in a given language thanks to a formula in that logic.

All these threads in computational complexity theory are still very active at the time of redaction of this manuscript and are also all largely of independent interest.

At the beginning of the second millennium, computational complexity theory had definitely become a mathematically important research field, as witnessed by its numerous connections to other domains in mathematics (Aaronson even says that "today (2011, editor's note) it draws on probability, number theory, combinatorics, representation theory, Fourier analysis, and nearly every other subject about which yellow books are written" Aaronson [2011]), its active research community as shown above, as well as the mathematical challenging and deep questions it raises Wigderson [2006], Allender [2009]. Moreover, as Impagliazzo claims, computational complexity theory, though made up of many threads, also exhibits a remarkable unity and connection between those, and "is best tackled as a single, united field, not splintered into specialized subareas" Impagliazzo [2005]. Some authors also point to the importance of computational complexity theory in a wealth of other sciences, as well as in philosophy Wigderson [2006], Allender [2009], Aaronson [2011].

Unfortunately, while all this flurry of activity and new developments since the 1980s brought several new techniques and a variety of new approaches and viewpoints for proving complexity measure lower bounds, significant progress on central complexity measures kept on facing new formal barriers, as explained in [Arora and Barak, 2009, Chapter 23]. In the 2000s and up to the date of writing of this manuscript, computational complexity continued to be developed along the same lines as described above, trying to circumvent these barriers. "Nevertheless", as Perifel puts it Perifel [2014], "the big questions remain without answer. The quest is not finished."

1.4.3 This thesis

The big framework of this Ph.D. thesis is the approach to the L versus P question based on lower bounds on the size of branching programs, its ultimate goal being to prove that there exists a language in P that does not belong to $\mathsf{PBP} = \mathsf{L/poly}$, thus settling this dauntingly difficult open question. If we zoom in into the lower levels of the NC hierarchy, we actually have the following:

$$\mathsf{AC}^0 \subseteq \mathsf{ACC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{PBP} \subseteq \mathsf{PNBP} \subseteq \mathsf{AC}^1$$

(see Mahajan [2007] and [Vollmer, 1999, Figure 4.6], though in these references the inclusion structure is presented for *uniform versions* of those classes, using a notion of uniformity that we won't elaborate — we refer the interested reader to Vollmer's book Vollmer [1999]), showing that PBP is well within the NC hierarchy, which is believed to be strict and even such that $\mathsf{P} \not\subseteq \mathsf{NC}$. However, as we explained in the previous section, strictness of the NC hierarchy is very far from being proven (we only know that AC^0 is stricly included in ACC^0) and concerning its relationship to P, the best result currently known is that $\mathsf{P} \not\subseteq \mathsf{ACC}^0[p]$ for $p \in \mathbb{N}$ some prime (see Theorem 1.3.8). In fact, though considered extremely unlikely by computational complexity theorists, it still isn't ruled out that actually $\mathsf{NP} \subseteq \mathsf{ACC}^0$ (the best result indicating this is wrong known at the time of writing of this thesis is William's result that $\mathsf{NTIME}(2^n) \not\subseteq \mathsf{ACC}^0$ Williams [2014]). This may be why AC^1 and its subclasses, that already yield a wealth of strikingly difficult and deep questions (also because of fundamental links with formal language theory and logic — see, e.g., Mahajan [2007] and [Vollmer, 1999, Chapter 4]), of whom most are still open, has been one of the main playgrounds for research in the domain of lower bounds for non-uniform models of computation since the 1980s, as witnessed by Jukna's recent (at the time of writing ot this thesis) book on the subject Jukna [2012]. (For instance, all lower

bound results for restricted types of circuit or branching program models of computation mentioned in the previous subsection are linked to classes inside AC^1, except for those holding for monotone Boolean circuits.)

This thesis precisely falls within this context of searching for lower bounds for complexity measures on non-uniform models of computation corresponding to subclasses of AC^1: we first study Nečiporuk's method to prove lower bounds for these complexity measures, as well as for some complexity measures on variants of these non-uniform models of computation that may be more powerful (i.e. correspond to classes believed to be bigger than AC^1), in Chapter 2, before we move on to a restricted variant of branching programs based on algebraic automata theory concepts capturing NC^1 and its subclasses in Chapter 3 and prove new results about mostly unstudied strict subclasses of AC^0 captured with this variant in Chapter 4 and Chapter 5.

Chapter 2

The Nečiporuk method and its limitations

As explained in the previous chapter, computational complexity theorists do not know how to prove "strong" complexity lower bounds in general non-uniform models of computation and, in fact, relatively few methods exist to prove such bounds at all. Fifty-two years ago, Nečiporuk wrote his famous two-page note entitled "A Boolean function" Nečiporuk [1966]. That note contained the first super-linear lower bounds on the size of Boolean formulæ over arbitrary bases and the size of contact schemes needed to compute some explicit Boolean function, both corresponding to subclasses of AC^1.

Nečiporuk's Nečiporuk [1966] method still yields the best lower bounds known today for explicit functions in a number of complexity measures. In particular, there are explicit functions for which Nečiporuk's method yields lower bounds of $\Omega(n^2/\log n)$ on formula size over an arbitrary basis, $\Omega(n^2/\log^2 n)$ on deterministic branching program size and on contact scheme size, and $\Omega(n^{3/2}/\log n)$ on non-deterministic branching program size, switching-and-rectifier network size, parity branching program size and span program size. (All of these complexity measures, except probably for the two last ones, related to subclasses of ACC^1, are related to subclasses of AC^1. Those not defined in the previous chapter are specifically defined in Subsection 2.1.3.) All of these are the best known lower bounds for these complexity measures for any explicit function. The first two of these lower bounds are contained in Nečiporuk's original paper Nečiporuk [1966]. Pudlák Pudlák [1987] points out that Nečiporuk's method yields the third lower bound for *non-deterministic* branching program size, as well as for switching-and-rectifier network size Razborov [1991]; Karchmer and Wigderson Karchmer and Wigderson [1993] point out that Pudlák's observation extends to parity branching program size and hence

also applies to span program size.

Two simple explicit functions that yield the lower bounds mentioned above are the Element Distinctness function and the Indirect Storage Access function (see Subsection 2.1.2).

Nečiporuk's method relies on counting subfunctions induced on blocks in a partition of the input variables, a concept introduced in Subsection 2.1.1. It is natural to try to optimise the use of the method, both in terms of how the bound depends on the numbers of subfunctions for each block in the partition and whether there are functions other than Element Distinctness and Indirect Storage Access for which one can prove stronger lower bounds.

For formula size over arbitrary bases, it is well known (see, e.g., Wegener [1987], Savage [1976]) that $\Theta(n^2/\log n)$ is indeed the best lower bound obtainable by Nečiporuk's method. Savage Savage [1976] also cites Paterson (unpublished) as improving the constant factor in the bound. Similarly, $\Theta(n^2/\log^2 n)$ is the best lower bound obtainable by Nečiporuk's method for deterministic branching program size, as noted by Wegener [Wegener, 1987, p. 422], who states the claim with a hint at its proof. Moreover, Alon and Zwick Alon and Zwick [1989] derived the optimal multiplicative constant in this lower bound as a function of the number of subfunctions of f in each block. Beame and McKenzie, in an unpublished note written in 2011 and now integrated in Section 2.2, formally showed that the third lower bound also uses Nečiporuk's method in an optimal way, namely that the best bound on non-deterministic branching program size obtainable by Nečiporuk's method is indeed $\Theta(n^{3/2}/\log n)$, as was already implicitly admitted. We observe that exactly the same arguments allow to show that this is also the best bound on parity branching program size obtainable by Nečiporuk's method. This automatically applies to span program size and switching-and-rectifier network size since these measures are upper-bounded by parity and non-deterministic branching program size, respectively.

The main contribution of this chapter is to be found in Section 2.3, where we define precisely *what* the Nečiporuk method actually is, independently from any specific complexity measure, so as to provide a unifying framework in which it makes sense to speak about *the* Nečiporuk method. Indeed, although Nečiporuk published his original result 50 years ago and his "technique" has been treated in several classical references (see Savage [1976], Wegener [1987, 2000], Jukna [2012]), to the best of our knowledge, there did not exist any abstract, measure-independent, unifying definition of the "method" that would encompass all previous applications of the "method" and allow new ones to be carried out easily, or at least in a clear way. The definition we suggest is in fact an abstract version of the general definition that was considered by Alon and Zwick in Alon and Zwick [1989]

for the case of deterministic branching programs. In this abstract framework we can then show, in a generic way, that for any complexity measure, an upper bound on the complexity, for this measure, of computing the Indirect Storage Access function with suitable parameters yields an upper bound on the best lower bound obtainable using Nečiporuk's method (defined that way).

In Section 2.4, we then prove an upper bound on the complexity of all Indirect Storage Access functions for each of the complexity measures introduced in Subsection 2.1.3. We finally use those to prove that the more general and abstract definition of Nečiporuk's lower bound method introduced in this chapter yields the same lower bounds and hits the same limits as those already known in the case of the size measure of non-determinitic and parity branching programs (Section 2.5), deterministic branching programs (Section 2.6) and formulæ (Section 2.7), as well as to prove new lower bounds and limitations results, in this framework, for the size measure of limited non-deterministic variants of branching programs (Section 2.6) and formulæ (Section 2.7). In each case, the Indirect Storage Access function family with suitable parameters yields the asymptotically best-possible lower bound.

The content of this chapter is almost taken as is from the article Beame et al. [2016].

2.1 Specific preliminaries

For $k \in \mathbb{N}_{>0}$ and $a \in \{0,1\}^k$ we denote by $\mathrm{bin}_k(a)$ its associated natural number with big-endian representation, i.e. $\sum_{i=1}^{k} a_i 2^{k-i}$. Throughout this chapter, the binary representation of a natural number will refer to its big-endian representation.

2.1.1 Boolean functions and subfunctions

In this chapter, we will essentially consider computation of Boolean functions in the following sense: for $n \in \mathbb{N}$, an n-ary *Boolean function over* V is a function $f\colon \{0,1\}^V \to \{0,1\}$, where $V \subseteq \mathbb{N}$ and $|V| = n$. When V is not specified we assume that $V = [n]$. A *family of Boolean functions* is an indexed family $F = \{f_i\}_{i\in I}$ where $I \subseteq \mathbb{N}$ and such that for all $i \in I$, f_i is a Boolean function of arity i.

Subfunctions will play a key role in the lower bound method studied in this chapter, the intuition being that the more different subfunctions a given Boolean function has, the more difficult it is to compute it.

Let f be an n-ary Boolean function. For any $V \subseteq [n]$ and any $\rho \in \{0,1\}^{[n]\setminus V}$, we will denote by $f|_\rho$ the *subfunction of f on V induced by the partial assignment ρ on* $[n] \setminus V$, that is the function $f|_\rho \colon \{0,1\}^V \to \{0,1\}$ such that for all $y \in \{0,1\}^V$, we have $f|_\rho(y) = f(x)$ where $x_i = y_i$ for all $i \in V$ and $x_i = \rho(i)$ for all $i \in [n] \setminus V$. We will also denote by $r_V(f)$ the *total number of subfunctions of f on V*, i.e. the cardinality of the set $s_V(f) = \{f|_\rho \mid \rho \in \{0,1\}^{[n]\setminus V}\}$.

The following easy lemma gives an upper bound on the total number of subfunctions of a given Boolean function on a certain subset of input variable indices.

Lemma 2.1.1. *Let $f \colon \{0,1\}^n \to \{0,1\}$ be a Boolean function. For any $V \subseteq [n]$, $r_V(f) \leq \min\{2^{2^{|V|}}, 2^{n-|V|}\}$.*

Proof. Since $r_V(f)$ counts the total number of subfunctions $f|_\rho \colon \{0,1\}^V \to \{0,1\}$ of f on V induced by a partial assignment $\rho \in \{0,1\}^{[n]\setminus V}$, it is at most the total number of Boolean functions on $|V|$ variables (i.e. $2^{2^{|V|}}$) and the total number of assignments to $n - |V|$ variables (i.e. $2^{n-|V|}$). □

Let $f \colon \{0,1\}^V \to \{0,1\}$ be a Boolean function and $i \in V$. We say that f *depends on its i-th variable* if there exist $a, a' \in \{0,1\}^V$ that differ only in the bit corresponding to i such that $f(a) \neq f(a')$ (definition based on Jukna [2012]). In particular, if f does not depend on a set W of variables and $a, a' \in \{0,1\}^V$ differ only on bits whose positions are in W, then $f(a) = f(a')$. The following proposition shows that variables on which f does not depend do not affect its number of subfunctions.

Proposition 2.1.2. *Let $f \colon \{0,1\}^n \to \{0,1\}$ be a Boolean function. Let $V \subseteq [n]$ and $W \subseteq V$ such that for all $i \in W$, f does not depend on x_i. Then for $V' = V \setminus W$, $r_V(f) = r_{V'}(f)$.*

Proof. We give a bijection from $s_V(f)$ to $s_{V'}(f)$. For $g \in s_V(f)$, say $g = f|_\rho$ for some $\rho \in \{0,1\}^{[n]\setminus V}$, define $\psi(g) \in s_{V'}(f)$ by $\psi(g) = g|_\zeta = f|_{\rho\zeta}$ where $\zeta = 0^W$ assigns 0 to all elements of W. By assumption, for all $i \in W$, f does not depend on x_i so for all $\zeta' \in \{0,1\}^W$, $f|_{\rho\zeta'} = f|_{\rho\zeta}$; moreover, it is also easy to see that $f|_{\rho\zeta} = f|_{\rho'\zeta}$ for any $\rho' \in \{0,1\}^{[n]\setminus V}$ such that $g = f|_{\rho'}$. Now let $h' \in s_{V'}(f)$. By definition, there is some $\rho' \in \{0,1\}^{[n]\setminus V}$ and $\zeta' \in \{0,1\}^W$ such that $h' = f|_{\rho'\zeta'}$ and by assumption, the latter equals $f|_{\rho'\zeta} = \psi(g')$ for $g' = f|_{\rho'}$ and hence ψ is surjective.

Similarly, for $g, g' \in s_V(f)$ such that $g \neq g'$, we have that there exists $a \in \{0,1\}^V$ verifying $g(a) \neq g'(a)$. Let $\rho, \rho' \in \{0,1\}^V$ such that $g = f|_\rho$ and $g' = f|_{\rho'}$, and let $\zeta' \in \{0,1\}^W$ such that $\zeta'(i) = a_i$ for all $i \in W$. Then $\psi(g) = f|_{\rho\zeta} = f|_{\rho\zeta'} \neq f|_{\rho'\zeta'} = f|_{\rho'\zeta} = \psi(g')$, so ψ is 1-1 and hence $r_V(f) = |s_V(f)| = |s_{V'}(f)| = r_{V'}(f)$. □

It will often also be useful to enlarge the size of the domain of a Boolean function by adding additional input variables on which the function does not depend in order to obtain complete families of Boolean functions even if we cannot build a specific Boolean function for each possible input size.

Lemma 2.1.3. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function. Let $n' \in \mathbb{N}$ such that $n' > n$ and let $f'\colon \{0,1\}^{n'} \to \{0,1\}$ be the Boolean function defined by $f'(a) = f(a_1, \ldots, a_n)$ for all $a \in \{0,1\}^{n'}$. Then, for any $V \subseteq [n]$, $r_V(f') = r_V(f)$.*

Proof. It is immediate by definition that for $V \subseteq [n]$, $s_V(f') = s_V(f)$. □

2.1.2 Hard functions: Indirect Storage Access functions and Element Distinctness

In this section we define two natural families of functions for which Nečiporuk's method is known to produce asymptotically optimal lower bounds for some complexity measures. The first is the Element Distinctness function.

Definition 2.1.4. The *Element Distinctness* function $\mathrm{ED}_{N,m}$ for $m \geq N$ is the Boolean function that takes as input $n = N \cdot \lceil \log_2 m \rceil$ bits representing N integers in $[m]$ (outputting 0 on illegal inputs) and outputs 1 iff all the N integers have distinct values. When $n = 2k \cdot 2^k$, we write ED_n for the function ED_{N,N^2} where $N = 2^k$.

The second is the family of Indirect Storage Access functions. These will turn out to be useful in a broader range of applications than the Element Distinctness function and we will see that we can characterise Nečiporuk's method in terms of the bounds it achieves for these functions.

Indirect Storage Access functions seem to have been originally defined by Paul in Paul [1977] to give an example of a family of Boolean functions for which we have a trade-off between the minimum sizes of Boolean binary formulæ computing them and the minimum sizes of Boolean binary circuits computing them.

Definition 2.1.5. The *Indirect Storage Access* function for $k, \ell \in \mathbb{N}_{>0}$, denoted

$$\mathrm{ISA}_{k,\ell}\colon \{0,1\}^{k+2^k\ell+2^\ell} \to \{0,1\}$$

is such that for all $a \in \{0,1\}^{k+2^k\ell+2^\ell}$, $\mathrm{ISA}_{k,\ell}(a) = a_{\gamma(a)}$ where γ is computed from a as follows.

Let $\alpha(a)$ be the number represented in binary by the first k bits of a. Let $\beta(a)$ be the number represented in binary by the sequence of ℓ bits of a starting at position $k+1+\ell\cdot\alpha(a)$. Then $\gamma(a)$ is the position $k+2^k\ell+1+\beta(a)$. Informally speaking, $\mathrm{ISA}_{k,\ell}$ is just a function reading a bit using two levels of addressing: a k-bit pointer selects an ℓ-bit pointer (among 2^k such pointers) that picks one bit from a 2^ℓ-bit data string.

It is known that both these families of Boolean functions yield the asymptotically strongest lower bounds obtainable using Nečiporuk's method for Boolean formula size over arbitrary binary bases, and deterministic branching program complexity Wegener [1987], Boppana and Sipser [1990], Wegener [2000]. The essence of the argument in each case is the existence of a good partition with a large count of the number of subfunctions on the variables of the partition.

Lemma 2.1.6. *Let $n=2k\cdot 2^k>0$ for $k\in\mathbb{N}$. There is a partition of $[n]$ into blocks $V_1,\ldots,V_N$ for $N=2^k$ such that for all $i\in[N]$, $|V_i|=2k$ and $r_{V_i}(\mathrm{ED}_n)=\binom{N^2}{N-1}+1\geq N^{N-1}=2^{k(2^k-1)}$.*

Proof. Each block in the partition $V_1,\ldots,V_N$ corresponds to the bits of one of the N numbers for the ED_{N,N^2} function. Observe that for each assignment of distinct values to the $N-1$ other blocks, the subfunction induced on the i-th block must be different, since precisely those $N-1$ values must be avoided for the function to have value 1. There are $\binom{N^2}{N-1}$ possible choices of those $N-1$ distinct values; for other assignments, we get the constant 0 function. □

We now see that for different choices of k and ℓ, the function family $\mathrm{ISA}_{k,\ell}$ provides similar bounds but a more flexible range of parameters to obtain partitions of different sizes.

Definition 2.1.7. In the definition of $\mathrm{ISA}_{k,\ell}$, we will refer to $\alpha(a)$ and $\beta(a)$ as to the *primary* and *secondary* pointers of the $\mathrm{ISA}_{k,\ell}$ instance a. The bits of the secondary pointer will be denoted $\mathrm{sec}_1,\ldots,\mathrm{sec}_\ell$, and more generally the bits of the p-th secondary pointer among the 2^k such pointers in the instance at hand will be denoted $\mathrm{sec}[p]_1,\ldots,\mathrm{sec}[p]_\ell$ for $p\in[2^k]$. The 2^ℓ data bits will be referred to as *Data* and $\mathit{Data}[b_1,\ldots,b_\ell]$ will stand for the data bit at position $\mathrm{bin}_\ell(b_1,\ldots,b_\ell)+1$. When the context is clear, bits of a will also be viewed as input variable indices.

We now see that we can partition the set of input variables of $\mathrm{ISA}_{k,\ell}$ in such a way that the number of induced subfunctions is identical and maximal for all elements of the partition but one: this is formalised in the following lemma.

Lemma 2.1.8. *For every $k, \ell \in \mathbb{N}_{>0}$, there exists a partition $V_1, \ldots, V_{2^k}, U$ of $[k+2^k\ell+2^\ell]$ such that $|V_i| = \ell$ and $r_{V_i}(\mathrm{ISA}_{k,\ell}) = 2^{2^\ell}$ for all $i \in [2^k]$.*

Proof. Let $k, \ell \in \mathbb{N}_{>0}$. Consider the partition $[k+2^k\ell+2^\ell] = V_1 \uplus \cdots \uplus V_{2^k} \uplus U$ where V_i is the set $\{\mathrm{sec}[i]_1, \ldots, \mathrm{sec}[i]_\ell\}$ of indices of the ℓ variables forming the i-th secondary pointer in the $\mathrm{ISA}_{k,\ell}$ instance a. Then for each setting of the first k variables $a_1, \ldots, a_k$ of a, i.e, for each value $i = \mathrm{bin}_k(a_1, \ldots, a_k)$ of the primary pointer, every possible fixing of the 2^ℓ-bit data string induces a different subfunction on V_{i+1}, hence $r_{V_{i+1}}(\mathrm{ISA}_{k,\ell}) = 2^{2^\ell}$. □

From $\mathrm{ISA}_{k,\ell}$ we define the Indirect Storage Access functions family $\mathrm{ISA} = \{\mathrm{ISA}_n\}_{n\in\mathbb{N}}$, such that for all $n \in \mathbb{N}$

- if $n < 5$, $\mathrm{ISA}_n(a) = 0$ for all $a \in \{0,1\}^n$;
- if there exists $k \in \mathbb{N}_{>0}$ such that $n = h_{\mathrm{ISA}}(k)$, then

$$\mathrm{ISA}_n = \mathrm{ISA}_{k,k+\lceil \log_2 k \rceil} \ ;$$

- otherwise,

$$\mathrm{ISA}_n(a) = \mathrm{ISA}_{k',k'+\lceil \log_2 k' \rceil}(a_1, \ldots, a_{n'})$$

  for all $a \in \{0,1\}^n$ where $k' = \max\{k \in \mathbb{N}_{>0} \mid h_{\mathrm{ISA}}(k) < n\}$ and $n' = h_{\mathrm{ISA}}(k')$.

where

$$\begin{aligned} h_{\mathrm{ISA}} \colon \mathbb{N}_{>0} &\to \mathbb{N}_{>0} \\ m &\mapsto m + 2^m(m + \lceil \log_2 m \rceil) + 2^{m+\lceil \log_2 m \rceil} \ . \end{aligned}$$

ISA will be used to give, for each complexity measure we study in this chapter (this notion will be precisely defined in the next subsection), an actual family of Boolean functions that achieves the best lower bound obtainable using Nečiporuk's lower bound method (to be defined later). The setting of k and ℓ in its definition is crucial, because if we would for example set $\mathrm{ISA}_n = \mathrm{ISA}_{k,k}$ for all $n \in \mathbb{N}$ such that there exists $k \in \mathbb{N}_{>0}$ verifying $n = k + 2^k k + 2^k$, we would not reach the desired bounds.

The next lemma is a simple useful adaptation of Lemma 2.1.8.

Lemma 2.1.9. *For all $n \in \mathbb{N}, n \geq 5$, there exist $p, q \in \mathbb{N}_{>0}$ verifying $p \geq \frac{1}{32} \cdot \frac{n}{\log_2 n}$ and $q \geq \frac{n}{16}$ such that there exists a partition $V_1, \ldots, V_p, U$ of $[n]$ such that $r_{V_i}(\mathrm{ISA}_n) = 2^q$ for all $i \in [p]$.*

Proof. Let $n \in \mathbb{N}, n \geq 5$. Let $k \in \mathbb{N}_{>0}$ be the unique positive integer verifying $h_{\mathrm{ISA}}(k) \leq n < h_{\mathrm{ISA}}(k+1)$. Set $n' = h_{\mathrm{ISA}}(k)$. By definition we have $\mathrm{ISA}_{n'} = \mathrm{ISA}_{k,k+\lceil \log_2 k \rceil}$. Let

$V_1, \ldots, V_{2^k}, U$ be a partition of $[n']$ such that $r_{V_i}(\mathrm{ISA}_{n'}) = 2^{2^{k+\lceil \log_2 k \rceil}}$ for all $i \in [2^k]$ as given by Lemma 2.1.8. Moreover, by definition of ISA_n and by Lemma 2.1.3 (for the case in which $n' < n$), we have that $r_{V_i}(\mathrm{ISA}_n) = r_{V_i}(\mathrm{ISA}_{n'}) = 2^{2^{k+\lceil \log_2 k \rceil}}$ for all $i \in [2^k]$.

Set $q = 2^{k+\lceil \log_2 k \rceil}$ and $p = 2^k$.

A bit of elementary algebra shows:

$$n \leq h_{\mathrm{ISA}}(k+1) \leq 16 \cdot 2^{k+\lceil \log_2 k \rceil} = 16q \leq 16 \cdot 2^{k+\log_2 k+1} \leq 32kp \ .$$

Hence $q \geq \frac{n}{16}$ and $p \geq \frac{1}{32} \cdot \frac{n}{k} \geq \frac{1}{32} \cdot \frac{n}{\log_2 n}$ (as $\log_2 n \geq \log_2 2^k = k$). □

2.1.3 Computational models

Note that in this chapter, all the models of computation that we consider are Boolean, in the sense that they all work only on input words over the alphabet $\{0,1\}$.

First, in view of defining a model-independent notion of Nečiporuk's method, we define a complexity measure merely as a function that associates a non-negative integer to each Boolean function, as follows.

Definition 2.1.10. A *complexity measure on Boolean functions* is a function

$$\mathbf{M}\colon \bigcup_{n\in\mathbb{N}} \{0,1\}^{\{0,1\}^n} \to \mathbb{N}\,. \tag{2.1}$$

Nečiporuk originally applied his technique to prove lower bounds for the size of BPs and the size of binary formulæ, that we are going to define soon; his technique was later used for size of NBPs: all these measures will be considered here. We will also apply it to the size of a variant of NBPs, called parity branching programs, where acceptance depends on the parity of the number of accepting paths. Parity branching programs of polynomial size capture $\oplus\mathsf{L}$, the class of languages such that there exists an NTM running in logarithmic space that has an odd number of paths leading to acceptance for a word if and only if this word belongs to the language.

Definition 2.1.11. An NBP $P = (X, \delta, \tau, s, t_0, t_1)$ on $\{0,1\}^V$ (for $V \subseteq \mathbb{N}$ finite) can also be interpreted as a *parity branching program ($\oplus$BP)* on $\{0,1\}^V$ that computes a Boolean function $f^{\oplus}\colon \{0,1\}^V \to \{0,1\}$ where for all $a \in \{0,1\}^V$, $f^{\oplus}(a) = 1$ if and only if there is an odd number of paths in $P[a]$ from start vertex s to accepting sink vertex t_1. For any Boolean function $f\colon \{0,1\}^V \to \{0,1\}$ (for $V \subseteq \mathbb{N}$ finite) we also denote by $\oplus\mathbf{BP}(f)$ the minimum size of a $\oplus$BP computing f.

Further, we will apply Nečiporuk's method to the size of BPs and binary formulæ that have access to a limited amount of non-deterministic bits, measures to which the method doesn't seem to have been applied before. These two models are motivated by the well-known observation that unrestricted non-deterministic Boolean formulæ capture NP (see Goldsmith et al. [1996]) and further by Klauck's analysis of restricted non-deterministic formulæ Klauck [2007]. We now define the first of these models, and the associated complexity measure.

Definition 2.1.12. A branching program P is a *δ-limited non-deterministic branching program (δ-LNBP)* for $f\colon \{0,1\}^V \to \{0,1\}$ if and only if P is a deterministic branching program computing a function $f'\colon \{0,1\}^{V'} \to \{0,1\}$ with $V \subseteq V'$ and $|V' \setminus V| = \delta$ such that $f(a) = \bigvee_{b\in\{0,1\}^{V'\setminus V}} f'(a,b)$. The *size of the δ-LNBP P*, denoted by $|P|$, is its size as a BP. For any Boolean function $f\colon \{0,1\}^V \to \{0,1\}$ (for $V \subseteq \mathbb{N}$ finite) we also denote by $\mathbf{LNBP}_\delta(f)$ the minimum size of a δ-LNBP computing f.

This definition of δ-limited non-deterministic BPs, which does not appear to have been studied previously, is inspired by notions of limited non-determinism for other models Goldsmith et al. [1996], Hromkovic and Schnitger [2003], Klauck [2007].

Remark 2.1.13. The limited non-determinism of the δ-LNBP model is formulated in a framework of verification of explicitly represented guesses that is typical for time-bounded non-determinism. In contrast, the NBP model only represents non-deterministic guesses implicitly, which allows them to be used without being stored, as is typical for space-bounded computation. In particular, even if δ is unbounded (say $\delta = \infty$), the smallest ∞-LNBP could be somewhat larger than the smallest equivalent NBP and vice-versa. It is not difficult to see that an NBP of size s can be simulated by such an ∞-LNBP of size at most $2s^2$. Indeed, simulating the k-way branch at a given state in this NBP in an ∞-LNBP can be made by accessing $\lceil \log_2 k \rceil$ fresh non-deterministic bits in a decision tree of size at most k; so since each of the s states of our original NBP branches to at most s different states for each of the possible values 0 or 1, we get that we can simulate it by an ∞-LNBP of size at most $2s^2$. Conversely, however, it is unclear by how much the size would increase when simulating an ∞-LNBP by an NBP, but it is widely conjectured to grow exponentially, since one can prove that polynomial size ∞-LNBPs capture (non-uniform) NP, while polynomial size NBPs capture (non-uniform) NL. Hence, the reader should keep in mind that LNBPs are not a restricted variant of NBPs.

Remark 2.1.14. Two other models comparable to the NBP are *switching networks* (also called *contact schemes*), and the more general *switching-and-rectifier (RS) networks* (see

Razborov [1991], Jukna [2012]). The graph of an RS network is almost the same as that of an NBP except that its vertices remain unlabelled and each of its arcs is labelled by some literal or remains unlabelled, with the acceptance condition that of the NBP. (Switching networks are a special case of RS networks whose graphs are undirected.) The size of an RS network is the number of its labelled arcs. Note that Jukna Jukna [2012] calls RS networks "non-deterministic branching programs", but we observe, as did Razborov Razborov [1991], that the size measure of RS networks and the size measure of NBPs used in this thesis are the same to within a constant factor. Indeed, one can simulate NBPs by RS networks of at most twice the size – each NBP vertex becomes an RS vertex with two new successors (one reached by an arc labelled $\neg x_i$, the other reached by an arc labelled x_i, where i is the variable index labelling that vertex in the NBP) which have unlabelled arcs pointing to the corresponding successor vertices in the NBP. Conversely, RS networks can be simulated by NBPs of the same size. It can first be assumed w.l.o.g. that any vertex in the RS network only has unlabelled arcs and arcs labelled by literals from the same variable (any vertex having arcs labelled by literals from several different variables can be replaced at no extra cost by a set of vertices connected by unlabelled arcs having the property). Then, for each vertex u in the RS network that has out-arcs labelled x_i or $\neg x_i$, we have a vertex u' in the NBP labelled i and for each arc e from u to v in the RS network,

- if e is unlabelled, we have two arcs (u', w') in the NBP, one labelled 0, the other labelled 1, for each vertex w in the RS network that has labelled out-arcs and is reachable from u via an unlabelled path starting with e (that uses unlabelled arcs and whose intermediate vertices do not have labelled out-arcs);
- if e is labelled ℓ
  - and v has labelled out-arcs, we have an arc (u', v') in the NBP labelled accordingly (0 if $\ell = \neg x_i$ and 1 if $\ell = x_i$),
  - otherwise v only has unlabelled out-arcs, and we have an arc (u', w') in the NBP labelled accordingly (0 if $\ell = \neg x_i$ and 1 if $\ell = x_i$) for each vertex w in the RS network that has labelled out-arcs and is reachable from v via an unlabelled path.

Span programs Karchmer and Wigderson [1993] can be simulated by parity branching programs of at most twice the size – their size is also at most polynomial in parity branching program size.

As stated before, we will also apply Nečiporuk's lower bound method to a certain type of restricted circuits, namely binary formulæ which are circuits whose underlying graph is a binary tree, whose internal gates may be labelled by any of the 16 possible Boolean functions on 2 variables and whose input gates might also be labelled by a negation of a given variable. The size of such circuits is then given by the number of non-constant leaves they have.

Definition 2.1.15 (Deterministic and δ-limited non-deterministic formulæ, inspired by Klauck [2007] and Jukna [2012])**.** A (deterministic) *Boolean binary formula (BF)* φ on $\{0,1\}^n$ ($n \in \mathbb{N}$) is a binary tree with

- a single root,
- every internal node of arity 2,
- every internal node labelled by a function $g\colon \{0,1\}^2 \to \{0,1\}$,
- every leaf labelled by one of $0, 1, x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n$.

The *size of* φ, denoted by $|\varphi|$, is the number of its non-constant leaves. φ computes a Boolean function f_φ on $\{0,1\}^n$ in the natural way by function composition. For any Boolean function $f\colon \{0,1\}^V \to \{0,1\}$ (for $V \subseteq \mathbb{N}$ finite) we denote by $\mathbf{L}(f)$ the minimum size of a BF computing f.

Let $\delta \in \mathbb{N}$. A *δ-limited non-deterministic binary formula (δ-LNBF)* on $\{0,1\}^n$ φ is a deterministic binary formula φ' on $\{0,1\}^{n+\delta}$. It computes a Boolean function f_φ such that for $a \in \{0,1\}^n$, $f_\varphi(a) = \bigvee_{b\in\{0,1\}^\delta} f_{\varphi'}(a,b)$. The *size of the δ-LNBF* φ, denoted by $|\varphi|$, is its size as a BF. For any Boolean function $f\colon \{0,1\}^V \to \{0,1\}$ (for $V \subseteq \mathbb{N}$ finite) we also denote by $\mathbf{LL}_\delta(f)$ the minimum size of a δ-LNBF computing f.

For f an n-ary Boolean function ($n \in \mathbb{N}$), we call *proof-checker function for f* any $(n+\delta)$-ary function g for some $\delta \in \mathbb{N}$ verifying that $f(a) = \bigvee_{b\in\{0,1\}^\delta} g(a,b)$ for all $a \in \{0,1\}^n$.

Lemma 2.1.16. *Let $\delta \in \mathbb{N}$, let $g\colon \{0,1\}^{n+\delta} \to \{0,1\}$ and let $f\colon \{0,1\}^n \to \{0,1\}$ ($n \in \mathbb{N}$) be given by $f(a) = \bigvee_{b\in\{0,1\}^\delta} g(a,b)$ for all $a \in \{0,1\}^n$. Then, for all $V \subseteq [n]$, we have $r_V(f) \leq B(r_V(g), 2^\delta) - 1 < r_V(g)^{2^\delta}$ where $B(m,r) = \sum_{i=0}^{r} \binom{m}{i}$ is the volume of the Hamming ball of radius r in $\{0,1\}^m$.*

Proof. Let $V \subseteq [n]$. For $\rho \in \{0,1\}^{[n]\setminus V}$, by definition, $f|_\rho = \bigvee_{b\in\{0,1\}^\delta} g|_{\rho b}$. Since ρb assigns all variables in $[n+\delta]\setminus V$, each function $g|_{\rho b}$ is in $s_V(g)$. Therefore, each $f|_\rho \in s_V(f)$ is the

$\bigvee$ of 2^δ functions in $s_V(g)$ (not necessarily distinct). Therefore over all choices of ρ, the function $f|_\rho$ only depends on the set of between 1 and 2^δ among these subfunctions of g that are distinct (and not what values b with which each such subfunction is associated). Therefore there are at most $B(r_V(g), 2^\delta)-1$ possible distinct subfunctions $f|_\rho$ in $s_V(f)$. $\square$

2.2 Non-deterministic Branching Program Lower Bounds via Shannon Bounds

In this section we describe the simplest form of Nečiporuk's technique and its applications in order to give some intuition about the technique. Readers may prefer to skip to the generalised abstract definition of Nečiporuk's method in Section 2.3. The simple version here is based on the so-called "Shannon bounds" for a complexity measure. The Shannon function for a complexity measure maps n to the maximum complexity of any Boolean function over $\{0,1\}^n$ in that measure. Lower bounds on the Shannon function typically follow by a simple enumeration of the number of distinct functions of bounded measure.

For all $n, s \in \mathbb{N}$, and $\mathbf{M}$ a complexity measure let us denote by $\mathbf{M}_{sem}(n,s)$ the number of distinct n-ary Boolean functions of complexity measure at most s. In particular, define N_{sem} to be the function $\mathbf{M}_{sem}$ for NBPs and $\oplus_{sem}$ be that for $\oplus$BPs. The next lemma is the core of the simple version of Nečiporuk's technique.

Lemma 2.2.1. *For any $n \in \mathbb{N}$, for any n-ary Boolean function f on V that depends on all its inputs and any partition $V_1, \ldots, V_p$ of V we have*

$$\mathbf{NBP}(f) \geq \sum_{i=1}^{p} \max\left\{|V_i|, \min\{s \in \mathbb{N} \mid N_{sem}(|V_i|, s) \geq r_{V_i}(f)\}\right\},$$

$$\oplus\mathbf{BP}(f) \geq \sum_{i=1}^{p} \max\left\{|V_i|, \min\{s \in \mathbb{N} \mid \oplus_{sem}(|V_i|, s) \geq r_{V_i}(f)\}\right\}.$$

Proof. Let $n \in \mathbb{N}$, f be an n-ary Boolean function on V depending on all its inputs and $V_1, \ldots, V_p$ a partition of V. Let P be a Boolean NBP computing f. For all $i \in [p]$ we will denote by $s_i \in \mathbb{N}$ the number of vertices in P labelled by elements in V_i. It is clear that P is of size at least $\sum_{i=1}^{p} s_i$.

Let $i \in [p]$. Observe that for every subfunction g of f on V_i, there is by definition a partial assignment ρ on $V \setminus V_i$ such that $f|_\rho = g$, so it is not too difficult to see that g is computed by the Boolean NBP of size s_i obtained from P by:

1. removing all non-sink vertices labelled by elements not in V_i;

2. defining the new start vertex as one of the vertices whose label is in V_i and reachable from the start vertex of P by a path of nodes labelled by elements outside of V_i and arcs labelled consistently with ρ, then adding both an arc labelled 0 and an arc labelled 1 from this new start vertex to each other such reachable vertex (except for the extreme case of a constant function, in which we just set the start vertex as the appropriate sink vertex);

3. connecting a vertex u to a vertex v by an arc labelled by $a \in \{0,1\}$ if and only if there exists a path from u to v in P verifying that any intermediate vertex of the path is labelled by an element outside of V_i, the first arc is labelled by a and each arc (but the first one) is labelled consistently with ρ.

Thus, $r_{V_i}(f)$ is necessarily upper-bounded by the number of semantically distinct such NBPs we can build from P that way, which is in turn at most $N_{sem}(|V_i|, s_i)$. Moreover, since, by construction, f depends on all variables whose indices are in V_i, we have that for each element $\ell \in V_i$, P contains at least one vertex labelled by ℓ, so $s_i \geq |V_i|$. Hence, for each i, $s_i \geq \max\{|V_i|, \min\{s \in \mathbb{N} \mid N_{sem}(|V_i|, s) \geq r_{V_i}(f)\}\}$. Since the NBP has size at least $\sum_{i=1}^{p} s_i$ and the NBP is arbitrary, the bound of the lemma follows.

The same argument also applies directly to yield the bound for $\oplus$BPs, with $\oplus_{sem}(|V_i|, s)$ replacing $N_{sem}(|V_i|, s)$. □

Proposition 2.2.2. *Let $s \geq n$. Then $N_{sem}(n,s), \oplus_{sem}(n,s) < 2^{2(s+1)^2}$.*

Proof. We simply count the number of distinct branching programs. Subject to renaming and reorganising, any n-ary Boolean function computable by an NBP or $\oplus$BP of size at most s, can be computed by one of size exactly s, having $\{v_j\}_{j=1}^{s+2}$ as vertices, v_1 as start vertex, v_{s+1} as 0-vertex and v_{s+2} as 1-vertex, where no arc goes to the 0-vertex (that is, since what matters for the computation of both an NBP and a $\oplus$BP is the number of paths from the start vertex to the 1-vertex, arcs ending in the 0-vertex are unnecessary). The out-arcs at each node v_i can be described by the subset of vertices v_j, $j \neq i$ and $j \neq v_{s+1}$, reached on each of values 0 and 1. There are $(s-1)!$ different ways of reordering the names of vertices $v_2, \ldots, v_s$ that keep identical connectivity of the branching program and hence the function it computes, both as an NBP and a $\oplus$BP. Hence, it directly follows that $N_{sem}(n,s), \oplus_{sem}(n,s) \leq (2^{2s}n)^s/(s-1)! \leq 2^{2s^2}s^s/(s-1)!$, since $s \geq n$, therefore, since $s! \geq (s/e)^s$, $N_{sem}(n,s), \oplus_{sem}(n,s) \leq 2^{2s^2}se^s < 2^{2(s+1)^2}$. □

Definition 2.2.3. For the complexity measures $\mathbf{M} = \mathbf{NBP}, \oplus\mathbf{BP}$, the *simple Nečiporuk lower bound method* consists of the following.

1. Giving explicitly a non-decreasing function $b\colon \mathbb{N}_{>0} \to \mathbb{N}$ such that for any $n \in \mathbb{N}$, for any n-ary Boolean function f on V that depends on all its inputs and any partition $V_1, \ldots, V_p$ of V, we have $\sum_{i=1}^{p} \max\{|V_i|, \min\{s \in \mathbb{N} \mid \mathbf{M}_{sem}(|V_i|, s) \geq r_{V_i}(f)\}\} \geq \sum_{i=1}^{p} b(r_{V_i}(f))$.

2. For a given n-ary Boolean function g on V that depends on all the variables whose indices are in V, explicitly choosing a partition $V_1, \ldots, V_p$ of V, computing $r_{V_i}(g)$ for all $i \in [p]$ and concluding that $\mathbf{M}(g) \geq \sum_{i=1}^{p} b(r_{V_i}(g))$.

A function b satisfying the condition of Step 1 in the definition above is called a *simple Nečiporuk function for* $\mathbf{M}$.

We now give an explicit simple Nečiporuk function for **NBP** and $\oplus$**BP**.

Proposition 2.2.4. *The function on* $\mathbb{N}_{>0} \to \mathbb{N}$ *given by* $m \mapsto \left\lceil \sqrt{\frac{1}{2}\log_2 m} - 1 \right\rceil$ *is a simple Nečiporuk function for* **NBP** *and for* $\oplus$**BP**.

Proof. We start by observing that the function on $\mathbb{N}_{>0} \to \mathbb{N}$ given by

$$m \mapsto \left\lceil \sqrt{\frac{1}{2}\log_2 m} - 1 \right\rceil$$

is obviously non-decreasing. Let $n \in \mathbb{N}$, f be an n-ary Boolean function f on V depending on all its variables and $V_1, \ldots, V_p$ be a partition of V. Let $i \in [p]$. Let $s_i = \max\{|V_i|, \min\{s \in \mathbb{N} \mid N_{sem}(|V_i|, s) \geq r_{V_i}(f)\}\}$ for all $i \in [p]$. We claim that $s_i \geq \left\lceil \sqrt{\frac{1}{2}\log_2 r_{V_i}(f)} - 1 \right\rceil$ for all $i \in [p]$.

By definition, $N_{sem}(|V_i|, s_i) \geq r_{V_i}(f)$, and since $s_i \geq |V_i|$, Proposition 2.2.2 implies that $N_{sem}(|V_i|, s_i) < 2^{2(s_i+1)^2}$ and hence $2^{2(s_i+1)^2} > r_{V_i}(f)$, that is to say, $s_i > \sqrt{\frac{1}{2}\log_2 r_{V_i}(f)} - 1$. Since s_i is integral, we deduce that $s_i \geq \left\lceil \sqrt{\frac{1}{2}\log_2 r_{V_i}(f)} - 1 \right\rceil$. The lemma follows for **NBP**; the argument for $\oplus$**BP** is identical replacing N_{sem} by $\oplus_{sem}$. □

This directly gives us the following lower bounds.

Proposition 2.2.5 (Pudlák [1987], Karchmer and Wigderson [1993])**.**

$$\mathbf{NBP}(\mathrm{ED}_n), \mathbf{NBP}(\mathrm{ISA}_n), \oplus\mathbf{BP}(\mathrm{ED}_n), \oplus\mathbf{BP}(\mathrm{ISA}_n) \in \Omega\Big(\frac{n^{3/2}}{\log n}\Big) .$$

Proof. We first consider ED_n for $n = 2k2^k$ and $k \geq 2$. By Lemma 2.1.6 there is a partition $V_1, \ldots, V_N$ of $[n]$ for $N = 2^k$ such that $r_{V_i}(\mathrm{ED}_n) \geq N^{N-1}$ and $|V_i| = 2k$ for all $i \in [N]$. Applying Proposition 2.2.4, since ED_n depends on all its variables, we have

$$\begin{aligned}\mathbf{NBP}(\mathrm{ED}_n), \oplus\mathbf{BP}(\mathrm{ED}_n) &\geq \sum_{i=1}^{N} \left\lceil \sqrt{\frac{1}{2}\log_2 r_{V_i}(\mathrm{ED}_n)} - 1 \right\rceil \geq N \cdot \left\lceil \sqrt{\frac{1}{2}\log_2 N^{N-1}} \right\rceil - N \\ &\geq N \cdot \sqrt{\frac{N-1}{2}\log_2 N} - N\end{aligned}$$

which is in $\Omega\big(N^{3/2}(\log_2 N)^{1/2}\big)$ and hence $\Omega\Big(\frac{n^{3/2}}{\log n}\Big)$ since n is $\mathrm{O}(N \log_2 N)$.

We now consider ISA. Let $n \in \mathbb{N}, n \geq 32$. Let $V_1, \ldots, V_p, U$ be a partition of $[n]$ such that $r_{V_i}(\mathrm{ISA}_n) = 2^q$ for all $i \in [p]$ where $p, q \in \mathbb{N}_{>0}$ verify $p \geq \frac{1}{32} \cdot \frac{n}{\log_2 n}$ and $q \geq \frac{n}{16}$ as given by Lemma 2.1.9. Applying Proposition 2.2.4, since ISA_n depends on all its variables, we have

$$\begin{aligned}\mathbf{NBP}(\mathrm{ISA}_n), \oplus\mathbf{BP}(\mathrm{ISA}_n) &\geq \sum_{i=1}^{p} \left\lceil \sqrt{\frac{1}{2}\log_2 r_{V_i}(\mathrm{ISA}_n)} - 1 \right\rceil \\ &\geq \sum_{i=1}^{p} \Big(\sqrt{\frac{1}{2}\log_2(2^q)} - 1 \Big) = p \cdot \Big(\sqrt{\frac{q}{2}} - 1 \Big) \\ &\geq \frac{1}{32} \cdot \frac{n}{\log_2 n} \cdot \Big(\sqrt{\frac{n}{32}} - 1 \Big) \ .\end{aligned}$$

So $\mathbf{NBP}(\mathrm{ISA}_n), \oplus\mathbf{BP}(\mathrm{ISA}_n) \in \Omega\Big(\frac{n^{3/2}}{\log n}\Big)$. □

To understand the best lower bounds we can prove with the simple Nečiporuk lower bound method, we first give a lower bound on $N_{sem}(n, s)$ and $\oplus_{sem}(n, s)$ (valid for suitable values of $n, s \in \mathbb{N}$) that will allow us to give an upper bound on all simple Nečiporuk functions for **NBP**, $\oplus$**BP**. We do this by giving an easy upper bound on the size needed by NBPs and $\oplus$BPs to compute any n-ary Boolean function[1]; i.e., simple upper bounds on the Shannon function for **NBP** and $\oplus$**BP**.

Lemma 2.2.6. *For any n-ary Boolean function f on $\{0,1\}^n$ ($n \in \mathbb{N}$),*

$$\mathbf{NBP}(f), \oplus\mathbf{BP}(f) \leq 3 \cdot 2^{\lceil \frac{n}{2} \rceil} \ .$$

[1]Note that there are somewhat tighter but more complicated upper bounds of $2^{n/2+1}$ for **NBP** due to Lupanov Lupanov [1958] and a tight asymptotic upper bound of $2^{(n+1)/2}$ for $\oplus$**BP** due to Nečiporuk Nečiporuk [1962], respectively; see Jukna [2012].

Proof. Assume that $n = 2t$ is even. The constructed NBPs will have only one non-deterministic level, will be the same for all functions for the other levels 1 to $t-1$ and $t+1$ to $2t$, and every vertex at each level i will query variable x_i.

The first $t-1$ levels form a complete decision tree of height $t-1$ on variables $x_1, \ldots, x_{t-1}$ with a vertex at level t for each assignment $a_1 \cdots a_{t-1}$ to these variables. The last t levels of the NBP consist of a complete fan-in tree of height t on variables $x_{t+1}, \ldots, x_{2t}$ as follows: there is a vertex at level $t' > t$ for every assignment $a_{t'} \cdots a_{2t}$ to $x_{t'}, \ldots, x_t$ and there is an out-arc labelled $a_{t'}$ from this vertex to the vertex at level $t'+1$ corresponding to $a_{t'+1} \cdots a_{2t}$. The 1-output vertex has two in-arcs, one labelled a_{2t} from each vertex corresponding to an assignment a_{2t} at level $2t$.

Finally, we define the non-deterministic level t of the NBP for function f. For each assignment $a_1 \cdots a_{2t}$ on which f evaluates to 1, there is an out-arc labelled a_t from the vertex corresponding to $a_1 \cdots a_{t-1}$ at level t (which queries x_t) to the vertex corresponding to $a_{t+1} \cdots a_{2t}$ at level $t+1$.

The constructed NBP has at most $3 \cdot 2^t = 3 \cdot 2^{\frac{n}{2}}$ vertices. By observing that there is precisely one accepting path on any accepted input, we see that it is also a $\oplus$BP. □

Corollary 2.2.7. *For all $n, s \in \mathbb{N}$, $n \geq 2\left\lfloor \log_2(\frac{s}{3}) \right\rfloor$, $N_{sem}(n,s), \oplus_{sem}(n,s) > 2^{\frac{s^2}{36}}$.*

Proof. Clearly $N_{sem}(n,s)$ is non-decreasing in n, so it suffices to prove the corollary for $n = 2\left\lfloor \log_2(\frac{s}{3}) \right\rfloor$. Then $3 \cdot 2^{\frac{n}{2}} \leq s < 6 \cdot 2^{\frac{n}{2}}$. There are precisely $2^{2^n} > 2^{\frac{s^2}{36}}$ different Boolean functions on n inputs and, by Lemma 2.2.6, each may be computed by an NBP or $\oplus$BP of size at most s. □

Theorem 2.2.8. *Let $F = \{f_n\}_{n \in \mathbb{N}}$ be a family of Boolean functions. Let $L\colon \mathbb{N} \to \mathbb{N}$ be such that for each $n \in \mathbb{N}$, the lower bound $L(n)$ for $\mathbf{NBP}(f_n)$ or $\oplus\mathbf{BP}(f_n)$ has been obtained using the simple Nečiporuk lower bound method. Then, $L(n) \in \mathrm{O}\left(\frac{n^{3/2}}{\log n}\right)$.*

Proof. Let $F = \{f_n\}_{n \in \mathbb{N}}$ be a family of Boolean functions, where for each $n \in \mathbb{N}$, f_n depends on all the variables in $D_n \subseteq [n]$. Let $L\colon \mathbb{N} \to \mathbb{N}$ be such that

$$\begin{aligned} &L(n) \\ &= \max\Big\{ \sum_{i=1}^{p} \max\big\{ |V_i|, \min\{ s \in \mathbb{N} \mid N_{sem}(|V_i|, s) \geq r_{V_i}(f_n) \} \big\} \;\Big|\; V_1, \ldots, V_p \text{ partition } D_n \Big\} \end{aligned}$$

for all $n \in \mathbb{N}$.

Let $n \in \mathbb{N}$ and $V_1, \ldots, V_p$ be a partition of D_n. Let $i \in [p]$ and set

$$s_i = \max\big\{ |V_i|, \min\{ s \in \mathbb{N} \;\big|\; N_{sem}(|V_i|, s) \geq r_{V_i}(f_n) \} \big\} .$$

Suppose that $s_i = \min\{s \in \mathbb{N} \mid N_{sem}(|V_i|, s) \geq r_{V_i}(f_n)\} > |V_i|$. Since V_i is non-empty and $s_i > |V_i|$, we have $s_i > 1$. But as all four Boolean functions on one variable can be computed by an NBP of size 1, we have $N_{sem}(|V_i|, 1) \geq 4$, so that we must have $r_{V_i}(f) > 4$. We now have two cases depending on $|V_i|$. If $|V_i| < \log_2 \log_2 r_{V_i}(f) + 3$ then, by Lemma 2.2.6, since NBPs of size $3 \cdot 2^{\lceil \frac{|V_i|}{2} \rceil}$ suffice to compute all functions on V_i, which include those counted in $r_{V_i}(f_n)$,

$$s_i \leq 3 \cdot 2^{\lceil \frac{|V_i|}{2} \rceil} \leq 3 \cdot 2^{(\log_2 \log_2 r_{V_i}(f))/2+2} = 12\sqrt{\log_2 r_{V_i}(f)} \, .$$

On the other hand, if $|V_i| \geq \log_2 \log_2 r_{V_i}(f) + 3$ then, setting $s = \left\lceil 6\sqrt{\log_2 r_{V_i}(f)} \right\rceil$, we have

$$\begin{aligned} 2\log_2\Big(\frac{s}{3}\Big) \leq 2\log_2\Big(\frac{6\sqrt{\log_2 r_{V_i}(f)}+1}{3}\Big) \leq 2\log_2\Big(\frac{7}{3}\sqrt{\log_2 r_{V_i}(f)}\Big) &\leq \log_2 \log_2 r_{V_i}(f) + 3 \\ &\leq |V_i| \end{aligned}$$

so, by Corollary 2.2.7, we have that $N_{sem}(|V_i|, s) > 2^{\frac{s^2}{36}} \geq r_{V_i}(f)$, which means that $s_i \leq \left\lceil 6\sqrt{\log_2 r_{V_i}(f)} \right\rceil$. Therefore, for all $n \in \mathbb{N}$,

$$L(n) \leq \max\Big\{ \sum_{i=1}^{p} \max\Big\{ |V_i|, 12\sqrt{\log_2 r_{V_i}(f)} \Big\} \;\Big|\; V_1, \ldots, V_p \text{ partition } D_n \Big\} \, .$$

Let $n \in \mathbb{N}, n \geq 4$. By Lemma 2.1.1, it follows that

$$\begin{aligned} L(n) &\leq \max\Big\{ \sum_{i=1}^{p} \max\Big\{ |V_i|, 12\sqrt{\log_2(\min\{2^{2^{|V_i|}}, 2^{n-|V_i|}\})} \Big\} \;\Big|\; V_1, \ldots, V_p \text{ partition } D_n \Big\} \\ &= \max\Big\{ \sum_{i=1}^{p} \max\Big\{ v_i, 12\sqrt{\log_2(\min\{2^{2^{v_i}}, 2^{n-v_i}\})} \Big\} \;\Big|\; \sum_{i=1}^{p} v_i \leq n \text{ and } \forall i \in [p], v_i > 0 \Big\} \\ &\leq \max\Big\{ \sum_{i=1}^{p} \max\Big\{ v_i, 12\sqrt{\min\{2^{v_i}, n-v_i\}} \Big\} \;\Big|\; \sum_{i=1}^{p} v_i = n \text{ and } \forall i \in [p], v_i > 0 \Big\} \\ &= \max\Big\{ \sum_{i=1}^{p} h(v_i) \;\Big|\; \sum_{i=1}^{p} v_i = n \text{ and } \forall i \in [p], v_i > 0 \Big\} \qquad (\star\star) \end{aligned}$$

where $h(v) = \max\{v, 12\min\{2^{v/2}, \sqrt{n-v}\}\}$ for all $v \in \mathbb{N}$.

Let now $v_1, \ldots, v_p$ realise the maximum in $(\star\star)$. Clearly, $h(v) = 12 \cdot 2^{v/2}$ for all $v \in \mathbb{N}, v \leq \log_2 n - 1$ and hence $h(v) + h(v') \leq h(v + v')$ if $v + v' \leq \log_2 n - 1$. It follows that without loss of generality we can assume that there exists at most one $j \in [p]$ such

that v_j is smaller than $\frac{\log_2 n-1}{2}$. Such a small v_j has $h(v_j) = 12 \cdot 2^{v_j/2} < 12 \cdot n^{1/4}$. Let now $I \subseteq [p]$ such that $i \in I$ if and only if $h(v_i) = v_i$. We have that $\sum_{i \in I} h(v_i) \leq n$ by definition of $v_1, \ldots, v_p$. Moreover, in $[p] \setminus I$, there are at most $\frac{2n}{\log_2 n-1}$ elements, since for all $i \in [p] \setminus \{j\}$, $v_i \geq \frac{\log_2 n-1}{2}$, and each such i verifies $h(v_i) \leq 12\sqrt{n - v_i} \leq 12\sqrt{n}$. Hence,

$$L(n) \leq \sum_{i=1}^{p} h(v_i) \leq 24 \cdot \frac{n^{3/2}}{\log_2 n - 1} + 12 \cdot n^{1/4} + n \leq 74 \cdot \frac{n^{3/2}}{\log_2 n}$$

which completes the proof, as the case of $\oplus$BPs is treated identically. □

Limitations of this Formulation

Simply using some adaptation of Definition 2.2.3 would not allow us to recover the well-known $\Omega\left(\frac{n^2}{\log n}\right)$ lower bound on the size of binary formulæ contained in Nečiporuk's original article Nečiporuk [1966]. Indeed, for all $n, s \in \mathbb{N}$, let us denote by $F_{sem}(n, s)$ the number of n-ary Boolean functions on some fixed V computable by BFs of size at most s. We can prove a Lemma analogous to Lemma 2.2.1 where **NBP** is replaced by **L** and N_{sem} by F_{sem}. Similarly, we can define the *simple Nečiporuk lower bound method* for **L** as in Definition 2.2.3, as well as *simple Nečiporuk functions for* **L** accordingly. However, Lupanov showed (Lupanov [1960], see [Jukna, 2012, p.32]) that for all $n \in \mathbb{N}$, any n-ary Boolean function on some V can be computed by a BF of size at most $\alpha \cdot \frac{2^n}{\log_2 n}$ for some constant $\alpha \in \mathbb{R}_{>0}$ (a result which is analogous to Lemma 2.2.6). Following the same strategy as for the proofs of Corollary 2.2.7 and Theorem 2.2.8, we can show that this implies there exists a constant $\beta \in \mathbb{R}_{>0}$ such that any simple Nečiporuk function $b\colon \mathbb{N}_{>0} \to \mathbb{N}$ for **L** verifies $b(m) \leq \beta \cdot \frac{\log_2 m}{\log_2 \log_2 \log_2 m}$ for any sufficiently large $m \in \mathbb{N}_{>0}$. This means that this does not allow us to recover the well-known Nečiporuk bounding function of $m \mapsto \frac{1}{4} \log_2 m$ (see e.g. [Jukna, 2012, Theorem 6.16]), and therefore also not Nečiporuk's original lower bound.

Even if we managed to adapt Definition 2.2.3 to the case of binary formulæ, we cannot really do it in a clean way for all complexity measures we would like to study. If we were to try to adapt Lemma 2.2.1 to the case of the size of limited non-deterministic branching programs (LNBPs), we would define, as usual, for all $n, s, \delta \in \mathbb{N}$, the number $LN_{sem}(n, s, \delta)$ of n-ary Boolean functions on some fixed V computable by LNBPs of size at most s and using δ non-deterministic bits. But then, it would be false to say that for any $\delta, n \in \mathbb{N}$, for any n-ary Boolean function f depending on all of V and any partition $V_1, \ldots, V_p$ of V we have $\mathbf{LNBP}_\delta(f) \geq \sum_{i=1}^{p} \max\left\{|V_i|, \min\{s \in \mathbb{N} \mid LN_{sem}(|V_i|, s, \delta) \geq\right.$

$r_{V_i}(f)\}\}$ (this would induce an overcount, as we would most certainly count vertices corresponding to non-deterministic variables several times).

These considerations led us to the more general formulation of the Nečiporuk method described in the next section.

2.3 An abstract formulation of Nečiporuk's method

In this section we define an abstract version of Nečiporuk's lower bound method and provide some model-independent meta-results on the limitations of this method.

The main idea of the general version of the method is, for a given Boolean function, to partition its set of input variables and to lower bound its complexity by a sum over each element of the partition of a partial cost that depends only on the number of subfunctions of the function on the variables in this element. More formally, we state the method in the following way.

Definition 2.3.1. For a given complexity measure $\mathbf{M}$ on Boolean functions, *Nečiporuk's lower bound method* consists of the following.

1. Giving explicitly a non-decreasing function $b\colon \mathbb{N}_{>0} \to \mathbb{N}$ such that for any $n \in \mathbb{N}$, for any n-ary Boolean function f and any partition $V_1, \ldots, V_p$ of $[n]$, we have
$$\mathbf{M}(f) \geq \sum_{i=1}^{p} b(r_{V_i}(f)) \ .$$
2. For a given n-ary Boolean function g, explicitly choosing a partition $V_1, \ldots, V_p$ of $[n]$, computing $r_{V_i}(g)$ for all $i \in [p]$ and concluding that $\mathbf{M}(g) \geq \sum_{i=1}^{p} b(r_{V_i}(g))$.

A function b satisfying the condition of Step 1 in Nečiporuk's method is called a *Nečiporuk function for* $\mathbf{M}$ and we denote by $\mathcal{N}_{\mathbf{M}}$ the *set of all Nečiporuk functions for* $\mathbf{M}$.

The first step of Definition 2.3.1 is usually not included in the Nečiporuk method. For instance in Wegener [1987], Jukna [2012], an explicit Nečiporuk function b is given for a complexity measure $\mathbf{M}$ and therefore the result concerning the limitation of the method is relative to this function b. In the case of deterministic branching programs, the best possible b was given by Alon and Zwick Alon and Zwick [1989], who use a similar definition but we are not aware of any result of this kind for other complexity measures.

It follows from Definition 2.3.1 that the best lower bound achievable by the Nečiporuk method for a family $F = \{f_n\}_{n\in\mathbb{N}}$ of Boolean functions and a complexity measure $\mathbf{M}$ is

the function $\mathrm{N}_F^{\mathbf{M}}$:

$$n \mapsto \max\Big\{\sum_{i=1}^{p} b(r_{V_i}(f_n)) \;\Big|\; b \in \mathcal{N}_{\mathbf{M}} \wedge V_1, \ldots, V_p \text{ partition of } [n]\Big\} \,. \tag{2.2}$$

2.3.1 Meta-results on Nečiporuk's method

We now give two results concerning Nečiporuk's method depending on hypotheses on the complexity measure $\mathbf{M}$. We will apply these results in the next section with the appropriate constants and functions for each of the concrete computational models we consider in this chapter.

The first meta-result is that an upper bound on the complexity of the functions $\mathrm{ISA}_{k,k}$ implies an upper bound on every $b \in \mathcal{N}_{\mathbf{M}}$. Intuitively this is possible because by definition, b entails a lower bound on $\mathbf{M}(f)$ for every function f.

Lemma 2.3.2. *Let $\mathbf{M}$ be a given complexity measure on Boolean functions and assume that we have a non-decreasing function $g_{\mathbf{M}} \colon [1, +\infty[\to \mathbb{R}_{\geq 0}$ such that $\mathbf{M}(\mathrm{ISA}_{k,k}) \leq g_{\mathbf{M}}(k)$ for all $k \in \mathbb{N}_{>0}$ and there exists a constant $\alpha \in \mathbb{R}_{>0}$ such that $\frac{g_{\mathbf{M}}(k+1)}{g_{\mathbf{M}}(k)} \leq \alpha$ for all $k \in \mathbb{N}_{>0}$. Then, any $b \in \mathcal{N}_{\mathbf{M}}$ is such that*

$$b(m) \leq \alpha \cdot \frac{g_{\mathbf{M}}(\log_2 \log_2 m)}{\log_2 m}$$

for all $m \in \mathbb{N}, m \geq 4$.

Proof. Let $b \in \mathcal{N}_{\mathbf{M}}$. Let $m \in \mathbb{N}, m \geq 4$ and $k \in \mathbb{N}_{>0}$ be such that $2^{2^k} \leq m \leq 2^{2^{k+1}}$. Hence $2^k \leq \log_2 m \leq 2^{k+1}$ and of course $k \leq \log_2 \log_2 m \leq k+1$. Consider now $\mathrm{ISA}_{k+1,k+1}$. By Lemma 2.1.8 we have a partition $V_1, \ldots, V_{2^{k+1}}, U$ of the set of indices $[(k+1) + 2^{k+1}(k+1) + 2^{k+1}]$ of the input variables of $\mathrm{ISA}_{k+1,k+1}$ such that $r_{V_i}(\mathrm{ISA}_{k+1,k+1}) = 2^{2^{k+1}}$ for all $i \in [2^{k+1}]$. By hypothesis, it therefore follows that:

$$\begin{aligned}
g_{\mathbf{M}}(k+1) &\geq \mathbf{M}(\mathrm{ISA}_{k+1,k+1}) \\
&\geq \sum_{i=1}^{2^{k+1}} b(r_{V_i}(\mathrm{ISA}_{k+1,k+1})) + b(r_U(\mathrm{ISA}_{k+1,k+1})) \\
&\geq \sum_{i=1}^{2^{k+1}} b\big(2^{2^{k+1}}\big) \\
&= 2^{k+1} b\big(2^{2^{k+1}}\big) \,,
\end{aligned}$$

therefore $b(2^{2^{k+1}}) \leq \frac{g_{\mathbf{M}}(k+1)}{2^{k+1}}$. But since $\frac{g_{\mathbf{M}}(k+1)}{g_{\mathbf{M}}(k)} \leq \alpha$, $g_{\mathbf{M}}(k) \leq g_{\mathbf{M}}(\log_2 \log_2 m)$ (because $g_{\mathbf{M}}$ is non-decreasing and $1 \leq k \leq \log_2 \log_2 m$), b is non-decreasing and $m \leq 2^{2^{k+1}}$ we have

$$b(m) \leq b\big(2^{2^{k+1}}\big) \leq \frac{g_{\mathbf{M}}(k+1)}{2^{k+1}} \leq \alpha \cdot \frac{g_{\mathbf{M}}(\log_2 \log_2 m)}{\log_2 m} \ .$$

The lemma follows. □

Assuming an upper bound on every $b \in \mathcal{N}_{\mathbf{M}}$, as given for instance by the previous lemma, we can derive an upper bound on $\mathrm{N}_F^{\mathbf{M}}$ independently of the family of Boolean functions F. That is to say that we can give an overall (asymptotic) upper bound on the best complexity lower bounds we may obtain using Nečiporuk's lower bound method for the complexity measure $\mathbf{M}$, exhibiting the limitation of the method.

Lemma 2.3.3. *Let $\mathbf{M}$ be a given complexity measure on Boolean functions and assume that we have a function $h_{\mathbf{M}} \colon [4, +\infty[\to \mathbb{R}_{\geq 0}$ such that there exist $x_0 \in [2^8, +\infty[$ and a constant $\alpha \in \mathbb{R}_{>0}$ verifying that:*

(i) $h_{\mathbf{M}}$ is non-decreasing on $[x_0, +\infty[$;

(ii) $h_{\mathbf{M}}(2^x) \geq \log_2 x$ for all $x \in [\log_2 x_0, +\infty[$;

(iii) $h_{\mathbf{M}}(2^{2^v}) + h_{\mathbf{M}}(2^{2^{v'}}) \leq h_{\mathbf{M}}(2^{2^{v+v'}})$ for all $v, v' \in \mathbb{N}$ verifying $2^{2^v} \geq x_0$ and $2^{2^{v'}} \geq x_0$;

(iv) for all $b \in \mathcal{N}_{\mathbf{M}}$ and $m \in \mathbb{N}, m \geq 4$, we have $b(m) \leq \alpha \cdot h_{\mathbf{M}}(m)$.

Then, for any family of Boolean functions $F = \{f_n\}_{n \in \mathbb{N}}$, we have

$$\mathrm{N}_F^{\mathbf{M}}(n) \leq \alpha \cdot \Big(4 + h_{\mathbf{M}}(\lfloor x_0 \rfloor)\Big) \cdot \frac{n}{\log_2 n} \cdot h_{\mathbf{M}}(2^n)$$

for all $n \in \mathbb{N}, n \geq \log_2 x_0$.

Proof. The condition $n \geq \log_2 x_0$ ensures that $h_{\mathbf{M}}(2^n)$ is always well defined and satisfies (ii). Let $F = \{f_n\}_{n \in \mathbb{N}}$ be a family of Boolean functions. For all $n \in \mathbb{N}_{>0}$, let $h'_n \colon [n] \to \mathbb{R}$ be the function defined on $[n]$ by

$$h'_n(v) = \begin{cases} \alpha \cdot \min\{h_{\mathbf{M}}(2^{2^v}), h_{\mathbf{M}}(2^{n-v})\} & \text{if } \min\{2^{2^v}, 2^{n-v}\} \geq x_0 \\ \alpha \cdot h_{\mathbf{M}}(\lfloor x_0 \rfloor) & \text{otherwise} \end{cases}$$

for all $v \in [n]$.

Claim 2.3.4. *If $n \in \mathbb{N}_{>0}$ and $v \in [n]$ are such that $v \leq \log_2 n - 1$ and $2^{2^v} \geq x_0$ then $h'_n(v) = \alpha \cdot h_{\mathbf{M}}(2^{2^v})$.*

Proof. With the hypothesis of the claim we have

$$2^{2^v} \leq 2^{2^{\log_2 n - 1}} = 2^{\frac{n}{2}} \leq 2^{n - \log_2 n + 1} \leq 2^{n-v} \ ,$$

the middle inequality being a consequence of $n \geq \log_2 x_0 \geq 8$. Hence in this case $\min\{2^{2^v}, 2^{n-v}\} = 2^{2^v}$ which is greater than x_0. As by (i) $h_{\mathbf{M}}$ is non-decreasing we have $h'_n(v) = \alpha \cdot h_{\mathbf{M}}(2^{2^v})$. □

Let $n \in \mathbb{N}_{>0}$, $b \in \mathcal{N}_{\mathbf{M}}$ and $V \subseteq [n], V \neq \emptyset$. According to (iv), we have $b(m) \leq \alpha \cdot h_{\mathbf{M}}(m)$ for all $m \in \mathbb{N}, m \geq 4$. Moreover, by Lemma 2.1.1, we have $r_V(f_n) \leq \min\{2^{2^{|V|}}, 2^{n-|V|}\}$, so since b is non-decreasing, it follows that $b(r_V(f_n)) \leq b(\min\{2^{2^{|V|}}, 2^{n-|V|}\})$. Now, if $\min\{2^{2^{|V|}}, 2^{n-|V|}\} \geq x_0$, we get that

$$\begin{aligned} b(\min\{2^{2^{|V|}}, 2^{n-|V|}\}) &\leq \alpha \cdot h_{\mathbf{M}}(\min\{2^{2^{|V|}}, 2^{n-|V|}\}) && \text{by (iv)} \\ &= \alpha \cdot \min\{h_{\mathbf{M}}(2^{2^{|V|}}), h_{\mathbf{M}}(2^{n-|V|})\} && \text{by (i)} \\ &= h'_n(|V|) \ ; \end{aligned}$$

otherwise (i.e. $\min\{2^{2^{|V|}}, 2^{n-|V|}\} < x_0$), we get that

$$b(\min\{2^{2^{|V|}}, 2^{n-|V|}\}) \leq b(\lfloor x_0 \rfloor) \leq \alpha \cdot h_{\mathbf{M}}(\lfloor x_0 \rfloor) = h'_n(|V|)$$

since b is non-decreasing and $\lfloor x_0 \rfloor \geq 4$. Hence, $b(r_V(f_n)) \leq h'_n(|V|)$ for all $n \in \mathbb{N}_{>0}$, $b \in \mathcal{N}_{\mathbf{M}}$ and $V \subseteq [n], V \neq \emptyset$. Therefore, by definition, it follows that for all $n \in \mathbb{N}_{>0}$, we have

$$\begin{aligned} \mathrm{N}_F^{\mathbf{M}}(n) &= \max\Big\{ \sum_{i=1}^{p} b(r_{V_i}(f_n)) \ \Big| \ b \in \mathcal{N}_{\mathbf{M}} \text{ and } V_1, \ldots, V_p \text{ partition of } [n] \Big\} \\ &\leq \max\Big\{ \sum_{i=1}^{p} h'_n(|V_i|) \ \Big| \ V_1, \ldots, V_p \text{ partition of } [n] \Big\} \\ &= \max\Big\{ \sum_{i=1}^{p} h'_n(v_i) \ \Big| \ \sum_{i=1}^{p} v_i = n \text{ and } \forall i \in [p], v_i > 0 \Big\} \ . \qquad (\star) \end{aligned}$$

Let $n \in \mathbb{N}, n \geq \log_2 x_0$ and $v_1, \ldots, v_p \in \mathbb{N}_{>0}$ such that $\sum_{i=1}^{p} v_i = n$ that realises the maximum ($\star$). We first show that without loss of generality we can assume that there exists at most one $j \in [p]$ such that $\min\{2^{2^{v_j}}, 2^{n-v_j}\} \geq x_0$ and $v_j \leq \frac{\log_2 n - 1}{2}$.

If this is not the case then we have $v, v' \in [n]$ such that $\min\{2^{2^v}, 2^{n-v}, 2^{2^{v'}}, 2^{n-v'}\} \geq x_0$ and $v + v' \leq \log_2 n - 1$. It follows from (iii) and Claim 2.3.4 that

$$h'_n(v) + h'_n(v') = \alpha \cdot (h_{\mathbf{M}}(2^{2^v}) + h_{\mathbf{M}}(2^{2^{v'}})) \leq \alpha \cdot h_{\mathbf{M}}(2^{2^{v+v'}}).$$

But as $v \leq v + v' \leq \log_2 n - 1$, we have

$$\min\{2^{2^{v+v'}}, 2^{n-(v+v')}\} = 2^{2^{v+v'}} \geq 2^{2^v} = \min\{2^{2^v}, 2^{n-v}\} \geq x_0 \ ;$$

hence by Claim 2.3.4, $\alpha \cdot h_{\mathbf{M}}(2^{2^{v+v'}}) = h'_n(v + v')$ and $h'_n(v) + h'_n(v') \leq h'_n(v + v')$ and the partition that unifies the corresponding elements would yield a bound at least as big in $(\star)$.

If it exists this j is such that

$$h'_n(v_j) = \alpha \cdot h_{\mathbf{M}}(2^{2^{v_j}}) \leq \alpha \cdot h_{\mathbf{M}}\Big(2^{2^{\frac{\log_2 n - 1}{2}}}\Big) \leq \alpha \cdot h_{\mathbf{M}}(2^n).$$

Consider now the remaining elements of the partition, i.e. those $i \in [p] \setminus \{j\}$. If moreover we have $\min\{2^{2^{v_i}}, 2^{n-v_i}\} \geq x_0$ then by definition of h'_n we have

$$h'_n(v_i) \leq \alpha \cdot h_{\mathbf{M}}(2^{n-v_i}) \leq \alpha \cdot h_{\mathbf{M}}(2^n) \ .$$

As for this case we have $v_i > \frac{\log_2 n - 1}{2}$ there are at most $\frac{2n}{\log_2 n - 1}$ such i. Notice that $\frac{n}{\log_2 n - 1} \leq \frac{3}{2} \cdot \frac{n}{\log_2 n}$ for $n \geq 8$.

If otherwise $\min\{2^{2^{v_i}}, 2^{n-v_i}\} < x_0$, then we have

$$h'_n(v_i) = \alpha \cdot h_{\mathbf{M}}(\lfloor x_0 \rfloor)$$

and there are at most n such i.

Putting all together, we get that

$$\begin{aligned}
\mathrm{N}_F^{\mathbf{M}}(n) &\leq \max\Big\{\sum_{i=1}^{p} h'_n(v_i) \;\Big|\; \sum_{i=1}^{p} v_i = n \text{ and } \forall i \in [p], v_i > 0\Big\} \\
&\leq \alpha \cdot h_{\mathbf{M}}(2^n) + \frac{3n}{\log_2 n} \cdot \alpha \cdot h_{\mathbf{M}}(2^n) + n \cdot \alpha \cdot h_{\mathbf{M}}(\lfloor x_0 \rfloor) \\
&\leq \alpha \cdot h_{\mathbf{M}}(2^n) + \frac{3n}{\log_2 n} \cdot \alpha \cdot h_{\mathbf{M}}(2^n) + \frac{h_{\mathbf{M}}(2^n)}{\log_2 n} \cdot n \cdot \alpha \cdot h_{\mathbf{M}}(\lfloor x_0 \rfloor) \qquad \text{by (ii)} \\
&\leq \alpha \cdot \Big(4 + h_{\mathbf{M}}(\lfloor x_0 \rfloor)\Big) \cdot \frac{n}{\log_2 n} \cdot h_{\mathbf{M}}(2^n).
\end{aligned}$$

□

2.4 Upper Bounds for the Computation of $\mathrm{ISA}_{k,\ell}$

The $\mathrm{ISA}_{k,\ell}$ functions play a critical role in our approach to studying Nečiporuk's method. This section collects size upper bounds for computing $\mathrm{ISA}_{k,\ell}$ on every model considered in this chapter. These bounds will be required when limits to the Nečiporuk method for these models are investigated.

Theorem 2.4.1. *Let $\delta \in \mathbb{N}$. For all $k, \ell \in \mathbb{N}_{>0}$,*

$$\mathbf{NBP}(\mathrm{ISA}_{k,\ell}), \oplus\mathbf{BP}(\mathrm{ISA}_{k,\ell}) \leq 3 \cdot 2^{k+\frac{\ell}{2}} + 2^{\ell} \tag{2.3}$$

$$\mathbf{LNBP}_{\delta}(\mathrm{ISA}_{k,\ell}) \leq \begin{cases} 12 \cdot 2^{k} \max\left\{\frac{2^{\ell-\delta}}{\ell-\delta}, \ell\right\} + \frac{2^{2\ell-\delta}}{\ell-\delta} & \text{if } \ell > \delta \\ 2^{k}(3\ell+1) + 2 \cdot 2^{\ell} & \text{if } \ell \leq \delta \end{cases} \tag{2.4}$$

$$\mathbf{BP}(\mathrm{ISA}_{k,\ell}) \leq 9 \cdot \frac{2^{k+\ell}}{\ell} + \frac{2^{2\ell}}{\ell} \tag{2.5}$$

$$\mathbf{LL}_{\delta}(\mathrm{ISA}_{k,\ell}) \leq 12 \cdot 2^{k} \cdot \max\{2^{\ell-\delta}, \ell\} + 3 \cdot 2^{\ell} \tag{2.6}$$

$$\mathbf{L}(\mathrm{ISA}_{k,\ell}) \leq 7 \cdot 2^{k} \cdot 2^{\ell} \ . \tag{2.7}$$

Proof. Recall the notation used to refer to the bits of an $\mathrm{ISA}_{k,\ell}$ instance a. Here we further use $a_1, \ldots, a_k$ for the bits of the primary pointer p and (when relevant) $x_{n+1}, \ldots, x_{n+\delta}$ for the non-deterministic variables.

We begin with simple constructions:

Lemma 2.4.2. *Let $v_1, \ldots, v_k, y_1, \ldots, y_k, z_1, \ldots, z_{2^k}$ be Boolean variables, $k \geq 1$.*

1. *A size $2^k - 1$ deterministic branching program can "read" $v_1, \ldots, v_k$ and route the 2^k possible outcomes to 2^k distinct arcs;*

2. *A size $3k$ deterministic branching program can test whether $v_i = y_i$ holds for every $i \in [k]$;*

3. *A size $2^{k+1} - 2$ deterministic branching program with 2^k distinguished states s_w for $w \in \{0,1\}^k$ can ascertain that $(v_1, \ldots, v_k) = w$, i.e., has the property that for each w, a computation started at s_w accepts iff $(v_1, \ldots, v_k) = w$;*

4. *A size $4k$ formula can test whether $v_i = y_i$ holds for every $i \in [k]$;*

5. *A formula with leaves $z_1, \ldots, z_{2^k}$ and, for every $i \in [k]$, with 2^i leaves v_i or $\neg v_i$ can compute $z_{\mathrm{bin}_k(v_1,\ldots,v_k)+1}$.*

Proof. For (1), a full binary tree suffices.

For (2), a size-3 program can test whether $v_i = y_i$ for a fixed i, so a cascade of k such programs can check equality for every i.

For (3), an inverted binary tree first queries v_1 at each of 2^k leaves s_w, $w \in \{0,1\}^k$; each answer $a \in \{0,1\}$ branches from s_w to the unique state $s_{w'}$, among 2^{k-1} states at the next level, for which $w = aw'$; each state at this next level queries v_2 and branches to one of 2^{k-2} states at the next level, and so on, down to level k with two states querying v_k, for a total of $\sum_{1 \le i \le k} 2^i$ states; every missing arc in the above description rejects.

For (4), the formula $\bigwedge_{1 \le i \le k}[(v_i \wedge y_i) \vee (\neg v_i \wedge \neg y_i)]$ expanded into a binary tree has $4k$ leaves.

For (5), we note that $(\neg v_1 \wedge z_1) \vee (v_1 \wedge z_2)$ computes $z_{\mathrm{bin}_1(v_1)+1}$ and use induction, having computed $z_{\mathrm{bin}_k(0,v_2,\ldots,v_k)+1}$ from the leaves $z_1, \ldots, z_{2^{k-1}}$ and the 2^i leaves v_{i+1} or $\neg v_{i+1}$ for $i \in [k-1]$, and having computed $z_{\mathrm{bin}_k(1,v_2,\ldots,v_k)+1}$ similarly from the leaves $z_{2^{k-1}+1}, \ldots, z_{2^k}$ and 2^i further leaves v_{i+1} or $\neg v_{i+1}$ for $i \in [k-1]$. □

The **NBP** *case.* If $\ell = 1$ then, by Lemma 2.4.2.1, a (deterministic) BP of size $2^k - 1 + 2^k + 2 < 3 \cdot 2^{k+\ell/2} + 2^\ell$ computes $\mathrm{ISA}_{k,\ell}$. So let $\ell > 1$. For every $w \in \{0,1\}^{\lceil \ell/2 \rceil}$, $w' \in \{0,1\}^{\lfloor \ell/2 \rfloor}$ and $p' \in [2^k]$, the NBP will have states $s_{(w,w')}$, $(p', s_{w'})$ and p'. Together with further states, the NBP implements the following:

- Read bits $a_1, \ldots, a_k, \mathrm{sec}_1, \ldots, \mathrm{sec}_{\lceil \ell/2 \rceil}$.
- Guess $w' \in \{0,1\}^{\lfloor \ell/2 \rfloor}$ and branch to $s_{(\mathrm{sec}_1,\ldots,\mathrm{sec}_{\lceil \ell/2 \rceil},w')}$, forgetting $a_1, \ldots, a_k$.
  (For every $w' \in \{0,1\}^{\lfloor \ell/2 \rfloor}$ and for every $a \in \{0,1\}$, every state querying $\mathrm{sec}_{\lceil \ell/2 \rceil}$, i.e., every bottom node in the binary tree formed by the first stage, is connected to the state $s_{(\mathrm{sec}_1,\ldots,\mathrm{sec}_{\lceil \ell/2 \rceil - 1},a,w')}$ with an arc labelled a.)
- If $\mathit{Data}[\mathrm{sec}_1, \ldots, \mathrm{sec}_{\lceil \ell/2 \rceil}, w'] = 1$ then guess the bits $a'_1, \ldots, a'_k$ of the primary pointer $p \in [2^k]$ and branch to the state $(\mathrm{bin}_k(a'_1, \ldots, a'_k) + 1, s_{w'})$.
  (For every $w \in \{0,1\}^{\lceil \ell/2 \rceil}$ and $w' \in \{0,1\}^{\lfloor \ell/2 \rfloor}$, the state $s_{(w,w')}$ queries $\mathit{Data}[w, w']$ and connects via an arc labelled 1 to every state $(p', s_{w'})$, $p' \in [2^k]$.)
- Ascertain that w' was guessed correctly.
  (For each $p' \in [2^k]$ separately, apply Lemma 2.4.2.3 to the distinguished states $(p', s_{w'})$, $w' \in \{0,1\}^{\lfloor \ell/2 \rfloor}$, to ascertain that a computation from $(p', s_{w'})$ reaches the state p' iff $(\mathrm{sec}[p']_{\lceil \ell/2 \rceil + 1}, \ldots, \mathrm{sec}[p']_\ell) = w'$.)

- Ascertain that p was guessed correctly.
  (Apply Lemma 2.4.2.3, to the 2^k distinguished states p', to ascertain that $(a_1, \ldots, a_k) = (a'_1, \ldots, a'_k)$.)

The first stage uses $2^{k+\lceil \ell/2 \rceil} - 1$ states by Lemma 2.4.2.1. The second stage needs the 2^ℓ states $s_{(w,w')}$. The fourth stage uses 2^k times $2^{\lfloor \ell/2 \rfloor + 1} - 2$ states by Lemma 2.4.2.3 (and also includes the $2^{k+\lfloor \ell/2 \rfloor}$ states $(p', s_{w'})$). The last stage uses $2^{k+1} - 2$ states by Lemma 2.4.2.3 for a total $< 2^k(2^{\lceil \ell/2 \rceil} + 2 \cdot 2^{\lfloor \ell/2 \rfloor} - 2) + 2^{k+1} + 2^\ell$, which equals $2^k(3 \cdot 2^{\ell/2}) + 2^\ell$ when ℓ is even and $2^k(\frac{4}{\sqrt{2}} \cdot 2^{\ell/2}) + 2^\ell < 2^k(3 \cdot 2^{\ell/2}) + 2^\ell$ when ℓ is odd.

The $\oplus$**BP** *case.* It is easy to check that the above NBP has a unique accepting path for any input for which $\mathrm{ISA}_{k,\ell}$ is 1 and hence as a $\oplus$BP it also computes $\mathrm{ISA}_{k,\ell}$.

The $\mathbf{LNBP}_\delta$ *case.* If $\ell \leq \delta$ then the secondary $\mathrm{ISA}_{k,\ell}$ pointer is no wider than δ, i.e., contains no more than δ bits. So a δ-LNBP can "store" the secondary pointer within its first ℓ non-deterministic variables $x_{n+1}, \ldots, x_{n+\ell}$ and solve $\mathrm{ISA}_{k,\ell}$ as follows:

- Read the primary pointer.
- Check that $(\mathrm{sec}_1, \ldots, \mathrm{sec}_\ell) = (x_{n+1}, \ldots, x_{n+\ell})$.
- Forget everything.
- Read $x_{n+1}, \ldots, x_{n+\ell}$.
- Check that $\mathit{Data}[x_{n+1}, \ldots, x_{n+\ell}] = 1$.

The first and second steps use $2^k - 1$ and $2^k 3\ell$ states respectively, appealing to Lemma 2.4.2.1 and Lemma 2.4.2.2. Note that across the second step, neither the secondary pointer nor $x_{n+1}, \ldots, x_{n+\ell}$ are remembered. The third step merges every arc that survived the second step and thus requires no state. The fourth and fifth steps require $2^\ell - 1$ and 2^ℓ states, for a total $< 2^k + 2^k 3\ell + 2 \cdot 2^\ell$.

Now suppose that $\ell > \delta$, i.e., the secondary pointer is strictly wider than δ. Let $m \in [\ell - \delta - 1]$, to be set optimally later. A δ-LNBP can implement the following strategy, where grey-shaded regions in the diagrams indicate the portion of the $\mathrm{ISA}_{k,\ell}$ variables that are remembered, at exponential cost in numbers of states, at any given time.

1. Read the primary pointer:

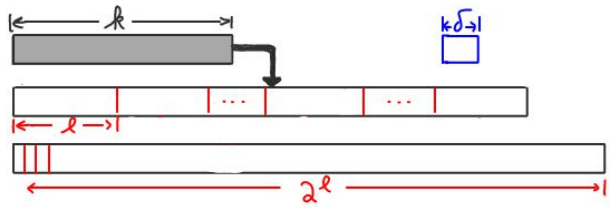


Figure 2.1 – $\mathbf{LNBP}_\delta(\mathrm{ISA}_{k,\ell})$ upper bound, step 1

Uses $2^k - 1$ states as per Lemma 2.4.2.1.

2. Check δ contiguous secondary pointer bits for equality with $x_{n+1}, \ldots, x_{n+\delta}$:

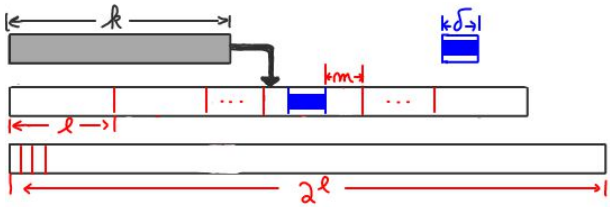


Figure 2.2 – $\mathbf{LNBP}_\delta(\mathrm{ISA}_{k,\ell})$ upper bound, step 2

Uses 2^k times 3δ states, again by Lemma 2.4.2.2. None of the checked bits are remembered.

3. Read $\ell - m - \delta$ other contiguous bits from the secondary pointer:

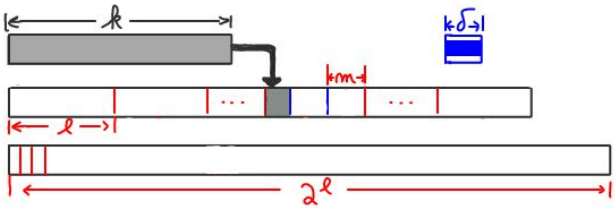


Figure 2.3 – $\mathbf{LNBP}_\delta(\mathrm{ISA}_{k,\ell})$ upper bound, step 3

Uses 2^k times $(2^{\ell-m-\delta} - 1)$ states.

4. Forget the primary pointer:

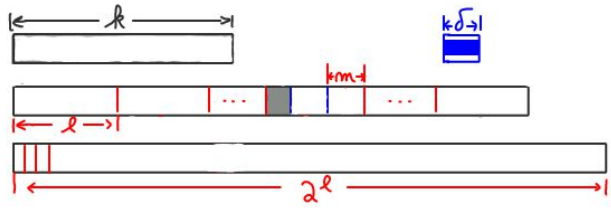


Figure 2.4 – $\mathbf{LNBP}_\delta(\mathrm{ISA}_{k,\ell})$ upper bound, step 4

No state required.

5. Read and remember the non-deterministic bits:

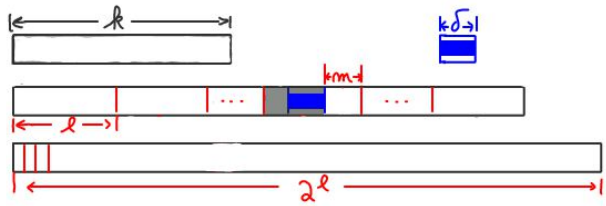


Figure 2.5 – $\mathbf{LNBP}_\delta(\mathrm{ISA}_{k,\ell})$ upper bound, step 5

Uses $2^{\ell-m-\delta}(2^\delta - 1) < 2^{\ell-m}$ states.

6. Read the data bits that remain candidates:

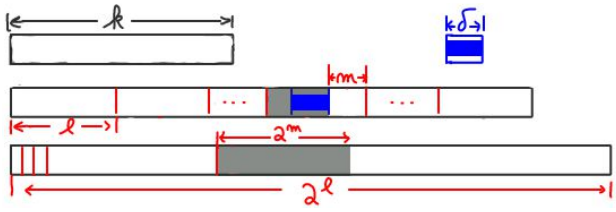


Figure 2.6 – $\mathbf{LNBP}_\delta(\mathrm{ISA}_{k,\ell})$ upper bound, step 6

Uses $2^{\ell-m}(2^{2^m} - 1) < 2^{\ell-m}2^{2^m}$ states.

7. Forget the part of the secondary pointer that was read:

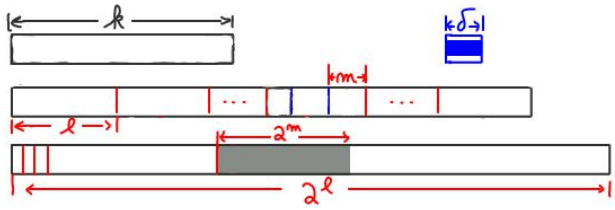


Figure 2.7 – $\mathbf{LNBP}_\delta(\mathrm{ISA}_{k,\ell})$ upper bound, step 7

No state required.

8. Read the primary pointer:

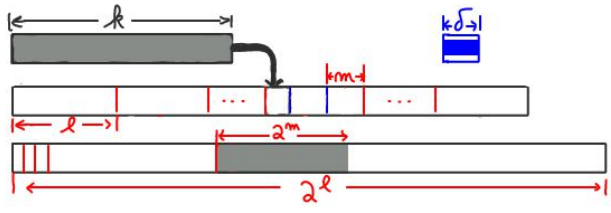


Figure 2.8 – $\mathbf{LNBP}_\delta(\mathrm{ISA}_{k,\ell})$ upper bound, step 8

Uses $2^{2^m}(2^k - 1) < 2^k 2^{2^m}$ states.

9. Read the secondary pointer bits that were never yet accessed:

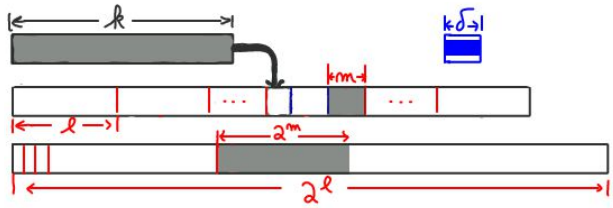


Figure 2.9 – $\mathbf{LNBP}_\delta(\mathrm{ISA}_{k,\ell})$ upper bound, step 9

Uses $2^{2^m}2^k(2^m-1) < 2^k 2^m 2^{2^m}$ states.

10. Output the appropriate data bit from memory: no state required.

The resulting δ-LNBP has fewer than

$$2^k + 2^k 3\delta + 2^k 2^{\ell-\delta-m} + 2^{\ell-m} + 2^{\ell-m}2^{2^m} + 2^k 2^{2^m} + 2^k 2^m 2^{2^m}$$

states, which is less than

$$12 \cdot 2^k \max\left\{\frac{2^{\ell-\delta}}{\ell-\delta}, \ell\right\} + \frac{2^{2\ell-\delta}}{\ell-\delta}$$

when m is set to $\left\lfloor \log_2\big(\ell - \delta - \log_2((\ell-\delta)^2)\big)\right\rfloor$ and $\ell - \delta \geq 8$ (and the degenerate case in which $1 \leq \ell - \delta < 8$ is treated separately by using a simpler method to compute $\mathrm{ISA}_{k,\ell}$).

The **BP** *case.* Follows from the $\mathbf{LNBP}_\delta$ case by setting $\delta = 0$. More specifically, stages 2 and 5 in the construction of the δ-LNBP are skipped.

The δ-**LL** *case.* We will not exploit more than ℓ non-deterministic variables amongst $x_{n+1}, \ldots, x_{n+\delta}$ so we suppose that $\delta \leq \ell$. Let $m = \ell - \delta$. The non-deterministic formula $V \wedge D$ solves $\mathrm{ISA}_{k,\ell}$ provided that V and D fulfil

$$V = 1 \text{ iff } (\mathrm{sec}_{m+1}, \ldots, \mathrm{sec}_\ell) = (x_{n+1}, \ldots, x_{n+\delta}),$$
$$D = 1 \text{ iff } \mathit{Data}[F_1, \ldots, F_m, x_{n+1}, \ldots, x_{n+\delta}] = 1,$$

and for $1 \leq j \leq m$, F_j evaluates to sec_j. By Lemma 2.4.2.5, D exists such that

$$|D| = 2^\ell + \Sigma_{j=1}^m 2^j \cdot |F_j| + \Sigma_{j=m+1}^\ell 2^j < 3 \cdot 2^\ell + \Sigma_{j=1}^m 2^j \cdot |F_j| \ . \tag{2.8}$$

By Lemma 2.4.2.5, each formula F_j, $1 \leq j \leq m$, can be constructed of size

$$|F_j| = 2^k + \Sigma_{j=1}^k 2^j < 3 \cdot 2^k \ . \tag{2.9}$$

By Lemma 2.4.2.4, for every $p \in [2^k]$, a formula V_p of size 4δ can be constructed that evaluates to 1 iff $(\mathrm{sec}[p]_{m+1}, \ldots, \mathrm{sec}[p]_\ell) = (x_{n+1}, \ldots, x_{n+\delta})$. The formula V can then be

constructed using Lemma 2.4.2.5, taking $z_1, \ldots, z_{2^k}$ as $V_1, \ldots, V_{2^k}$. The size of V is then

$$|V| = \sum_{j=1}^{2^k} |V_j| \ + \ \sum_{j=1}^{k} 2^j < 2^k \cdot 4\delta + 2^{k+1} \leq 2^k \cdot 6\ell \ . \tag{2.10}$$

Substituting (2.9) into (2.8) and using (2.10), the size of $V \wedge D$ is at most

$$2^k \cdot 6\ell + 3 \cdot 2^\ell + 2^{m+1} 3 \cdot 2^k \leq 6 \cdot 2^k(\ell + 2^m) + 3 \cdot 2^\ell \leq 12 \cdot 2^k \cdot \max\{2^m, \ell\} + 3 \cdot 2^\ell.$$

The **L** *case.* Follows from the δ-**LL** case by setting $\delta = 0$. More sharply, V from that construction is not needed, and $|D| = 2^\ell + \sum_{j=1}^{\ell} 2^j \, |F_j| < 2^\ell + 3 \cdot 2^k \cdot 2^{\ell+1} < 7 \cdot 2^k \cdot 2^\ell$. □

2.5 Non-deterministic and Parity Branching Programs revisited

We note in this section that, in the case of **NBP** and $\oplus$**BP**, the flexibility added by Definition 2.3.1 over Definition 2.2.3 yields no better lower bounds.

We first define the function $b_{\mathbf{NBP},\oplus\mathbf{BP}} \colon \mathbb{N}_{>0} \to \mathbb{N}$ by

$$b_{\mathbf{NBP},\oplus\mathbf{BP}}(m) = \begin{cases} \left\lceil \sqrt{\frac{1}{2}\log_2 m} - 1 \right\rceil & \text{if } m \geq 4 \\ 0 & \text{otherwise} \end{cases}$$

for all $m \in \mathbb{N}_{>0}$. Using the same strategy as in the proofs of Lemma 2.2.1 and Proposition 2.2.4, we can prove the following.

Proposition 2.5.1. *$b_{\mathbf{NBP},\oplus\mathbf{BP}}$ is a Nečiporuk bounding function for the* **NBP** *(respectively, $\oplus$***BP***) size complexity measure; i.e., $b_{\mathbf{NBP},\oplus\mathbf{BP}} \in \mathcal{N}_{\mathbf{NBP}}, \mathcal{N}_{\oplus\mathbf{BP}}$.*

Combining this with Lemmata 2.1.6 and 2.1.9, we can immediately derive asymptotic lower bounds on $\mathbf{NBP}(\mathrm{ED}_n)$, $\mathbf{NBP}(\mathrm{ISA}_n)$, $\oplus\mathbf{BP}(\mathrm{ED}_n)$ and $\oplus\mathbf{BP}(\mathrm{ISA}_n)$ using Nečiporuk's method and hence on $\mathrm{N}_{\mathrm{ED}}^{\mathbf{NBP}}$, $\mathrm{N}_{\mathrm{ISA}}^{\mathbf{NBP}}$, $\mathrm{N}_{\mathrm{ED}}^{\oplus\mathbf{BP}}$ and $\mathrm{N}_{\mathrm{ISA}}^{\oplus\mathbf{BP}}$.

Proposition 2.5.2. $\mathbf{NBP}(\mathrm{ED}_n)$, $\mathbf{NBP}(\mathrm{ISA}_n)$, $\oplus\mathbf{BP}(\mathrm{ED}_n)$, $\oplus\mathbf{BP}(\mathrm{ISA}_n) \in \Omega\left(\frac{n^{3/2}}{\log n}\right)$ *and hence we have that* $\mathrm{N}_{\mathrm{ED}}^{\mathbf{NBP}}(n), \mathrm{N}_{\mathrm{ISA}}^{\mathbf{NBP}}(n), \mathrm{N}_{\mathrm{ED}}^{\oplus\mathbf{BP}}(n), \mathrm{N}_{\mathrm{ISA}}^{\oplus\mathbf{BP}}(n)$ *are all* $\Omega\left(\frac{n^{3/2}}{\log n}\right)$.

Then, we can show that $b_{\mathbf{NBP},\oplus\mathbf{BP}}$ is in fact the asymptotically largest function in $\mathcal{N}_{\mathbf{NBP}} \cup \mathcal{N}_{\oplus\mathbf{BP}}$ and that the previous lower bound is in fact also the asymptotically

largest we may obtain. To do this, we appeal to our upper bound from Theorem 2.4.1 on the size of NBPs and $\oplus$BPs computing $\mathrm{ISA}_{k,\ell}$ and apply Lemma 2.3.2.

Proposition 2.5.3. *There exists a constant $c \in \mathbb{R}_{>0}$ verifying that any $b \in \mathcal{N}_{\mathbf{NBP}} \cup \mathcal{N}_{\oplus\mathbf{BP}}$ is such that $b(m) \leq c \cdot b_{\mathbf{NBP},\oplus\mathbf{BP}}(m)$ for all $m \in \mathbb{N}, m \geq 4$.*

Proof. Let $g\colon [1,+\infty[\to \mathbb{R}_{\geq 0}$ be the non-decreasing function defined by $g(x) = 4 \cdot 2^{\frac{3}{2}x}$ for all $x \in [1,+\infty[$. Theorem 2.4.1 tells us that for all $k \in \mathbb{N}_{>0}$, we have

$$\mathbf{NBP}(\mathrm{ISA}_{k,k}), \oplus\mathbf{BP}(\mathrm{ISA}_{k,k}) \leq 3 \cdot 2^{\frac{3}{2}k} + 2^k \leq 4 \cdot 2^{\frac{3}{2}k} = g(k)$$

and moreover, $\frac{g(k+1)}{g(k)} = \frac{4 \cdot 2^{\frac{3}{2}(k+1)}}{4 \cdot 2^{\frac{3}{2}k}} = 2\sqrt{2}$ for all $k \in \mathbb{N}_{>0}$. Therefore, by Lemma 2.3.2, any $b \in \mathcal{N}_{\mathbf{NBP}} \cup \mathcal{N}_{\oplus\mathbf{BP}}$ verifies

$$b(m) \leq 2\sqrt{2} \cdot \frac{g(\log_2 \log_2 m)}{\log_2 m} = 2\sqrt{2} \cdot \frac{4 \cdot 2^{\frac{3}{2}\log_2 \log_2 m}}{\log_2 m} = 8\sqrt{2} \cdot \sqrt{\log_2 m}$$

for all $m \in \mathbb{N}, m \geq 4$. □

Finally, using this and Lemma 2.3.3, we get the following result, showing that the asymptotically greatest lower bound we may expect using Nečiporuk's method for **NBP** and $\oplus$**BP** is (asymptotically) equivalent to the lower bound for ISA given in Proposition 2.5.2.

Theorem 2.5.4. *For any family of Boolean functions $F = \{f_n\}_{n\in\mathbb{N}}$, $\mathrm{N}_F^{\mathbf{NBP}}(n), \mathrm{N}_F^{\oplus\mathbf{BP}}(n) \in \mathrm{O}\left(\frac{n^{3/2}}{\log n}\right)$.*

Proof. We aim at applying Lemma 2.3.3 which requires four hypotheses, (i) to (iv).

For (i), let $h\colon [4,+\infty[\to \mathbb{R}_{\geq 0}$ be the function defined by $h(x) = \sqrt{\log_2 x}$ for all $x \in [4,+\infty[$ and $x_0 = 2^8$; as required, h is non-decreasing on $[2^8,+\infty[$.

For (ii), notice that $h(2^x) = \sqrt{x} \geq \log_2 x$ for all $x \in [2^8,+\infty[$.

For (iii), for all $v, v' \in \mathbb{N}$ verifying $2^{2^v} \geq 2^8$ and $2^{2^{v'}} \geq 2^8$, we have $h(2^{2^v}) + h(2^{2^{v'}}) = \sqrt{2^v} + \sqrt{2^{v'}} \leq \sqrt{2^{v+v'}} = h(2^{2^{v+v'}})$ because $x + y \leq xy$ when $x, y \geq 2$.

For (iv), by Proposition 2.5.3, we know that there exists a constant $\alpha \in \mathbb{R}_{>0}$ verifying that any $b \in \mathcal{N}_{\mathbf{NBP}} \cup \mathcal{N}_{\oplus\mathbf{BP}}$ is such that $b(m) \leq \alpha \cdot \sqrt{\log_2 m} = \alpha \cdot h(m)$ for all $m \in \mathbb{N}, m \geq 4$.

We can therefore apply Lemma 2.3.3 with $x_0 = 2^8$ and get that for any family of Boolean functions $F = \{f_n\}_{n\in\mathbb{N}}$ and all $n \in \mathbb{N}, n \geq 8$,

$$\mathrm{N}_F^{\mathbf{NBP}}(n), \mathrm{N}_F^{\oplus\mathbf{BP}}(n) \leq \alpha \cdot \left(4 + h(\lfloor 2^8 \rfloor)\right) \cdot \frac{n}{\log_2 n} \cdot h(2^n) = c \cdot \frac{n}{\log_2 n} \cdot \sqrt{n} = c \cdot \frac{n^{3/2}}{\log_2 n}$$

for some suitable constant $c \in \mathbb{R}_{>0}$, which implies that $\mathrm{N}_F^{\mathbf{NBP}}(n), \mathrm{N}_F^{\oplus\mathbf{BP}}(n) \in \mathrm{O}\left(\frac{n^{3/2}}{\log n}\right)$. □

2.6 Deterministic and Limited Non-deterministic Branching Programs

In this section, we focus on the model of Boolean deterministic branching programs, as well as its limited non-deterministic counterpart. In the case of **BP**, results related to the Nečiporuk method have been well-known for a long time (see for instance [Wegener, 1987, Chapter 14, Section 3] or Alon and Zwick [1989]). Reproving these results using what we presented in Section 2.3 is an opportunity to confirm the usability and validity of our approach.

Concerning limited non-deterministic branching programs, the definition of the model itself, as well as the results presented in this section concerning Nečiporuk's method for the associated measure seem to be novel.

For all $\delta \in \mathbb{N}$, let us define the functions $b_{\mathbf{LNBP}_\delta} \colon \mathbb{N}_{>0} \to \mathbb{N}$ and $b_{\mathbf{BP}} \colon \mathbb{N}_{>0} \to \mathbb{N}$ given by

$$b_{\mathbf{LNBP}_\delta}(m) = \begin{cases} \left\lceil \frac{1}{6} h_{\mathbf{LNBP}_\delta}(m) \right\rceil & \text{if } m \geq 4 \\ 0 & \text{otherwise} \end{cases}$$

and

$$b_{\mathbf{BP}}(m) = \begin{cases} \left\lceil \frac{1}{6} \frac{\log_2 m}{\log_2 \log_2 m} \right\rceil & \text{if } m \geq 4 \\ 0 & \text{otherwise} \end{cases}$$

for all $m \in \mathbb{N}_{>0}$, where $h_{\mathbf{LNBP}_\delta} \colon [4, +\infty[\to \mathbb{R}$ is defined as

$$h_{\mathbf{LNBP}_\delta}(x) = \begin{cases} \max\left\{ \frac{\log_2 x}{2^\delta (\log_2 \log_2 x - \delta)}, \log_2 \log_2 x \right\} & \text{if } 2^{2^{\delta+1}} \leq x \\ \log_2 \log_2 x & \text{otherwise .} \end{cases}$$

It is straightforward to see that $b_{\mathbf{BP}}(m) \leq b_{\mathbf{LNBP}_0}(m)$ for all $m \in \mathbb{N}_{>0}$ and that equality holds as soon as $b_{\mathbf{BP}}(m) \geq \log_2 \log_2 m$.

To prove that $b_{\mathbf{BP}} \in \mathcal{N}_{\mathbf{BP}}$, we reuse the well-known idea that is classically followed (see for instance Wegener [1987], Alon and Zwick [1989] or Jukna [2012]) to derive a specific function $b \in \mathcal{N}_{\mathbf{BP}}$, which is the fact that, given a Boolean function f and a Boolean BP P that computes it, we can compute any subfunction $f|_\rho$ of f with a Boolean BP obtained from P by "fixing" the values of the variables to which a value is affected by ρ (removing the associated vertices and directly linking their predecessors to their successors through

the arcs labelled accordingly). Therefore, if we denote by s the number of vertices in P labelled by elements from V, we get that an upper bound on the maximum number of subfunctions computed by BPs with s vertices obtained by "fixing" the values of a given set of variables in a given BP implies a lower bound on s depending on $r_V(f)$, as this number must be at least as big as $r_V(f)$. For the case of limited non-determinism, it suffices to observe that a Boolean δ-LNBP (for $\delta \in \mathbb{N}$) computing some Boolean function f does in fact deterministically compute a proof-checker function g for f. We can then combine the aforementioned technique with Lemma 2.1.16 binding the number of subfunctions of f on V and the number of subfunctions of g on V.

Proposition 2.6.1. *$b_{\mathbf{LNBP}_\delta} \in \mathcal{N}_{\mathbf{LNBP}_\delta}$ for all $\delta \in \mathbb{N}$. In particular, $b_{\mathbf{BP}} \in \mathcal{N}_{\mathbf{BP}}$.*

Proof. Let $\delta \in \mathbb{N}$. It is not too difficult to show that $b_{\mathbf{LNBP}_\delta}$ and $b_{\mathbf{BP}}$ are non-decreasing, we leave this to the reader.

Let f be an n-ary Boolean function on V, that verifies without loss of generality $V \cap [\delta] = \emptyset$, and $V_1, \ldots, V_p$ a partition of V. Let P be a Boolean δ-LNBP computing f and let g be the $(n+\delta)$-ary Boolean function computed by P when considering the δ non-deterministic bits as regular input variables (that is, g is such that, for all $a \in \{0,1\}^{V \cup [\delta]}$, $g(a) = 1$ if, and only if, $P[a]$ contains a path from s to t_1). g is a proof-checker function for f.

For all $i \in [p]$ we will denote by $s_i \in \mathbb{N}$ the number of vertices in P labelled by elements in V_i, as well as $q \in \mathbb{N}$ the number of vertices labelled by elements in $U = [\delta]$. It is clear that P is of size $\sum_{i=1}^{p} s_i + q \geq \sum_{i=1}^{p} s_i$.

We now claim that $s_i \geq b_{\mathbf{LNBP}_\delta}(r_{V_i}(f))$ for all $i \in [p]$.

Let $i \in [p]$. Let V_i' be the subset of V_i containing all indices of variables on which f depends. Then, by Lemma 2.1.2, $r_{V_i}(f) = r_{V_i'}(f)$. Moreover for each element $l \in V_i'$, P contains at least one vertex labelled by l. By Lemma 2.1.1, it follows that $r_{V_i}(f) = r_{V_i'}(f) \leq 2^{2^{|V_i'|}} \leq 2^{2^{s_i}}$ and $s_i \geq \log_2 \log_2(r_{V_i}(f))$.

If $r_{V_i}(f) \leq 3$ the claim is obvious from the definition of $b_{\mathbf{LNBP}_\delta}$.

In the case where $4 \leq r_{V_i}(f) < 4^{2^\delta}$ we have $\left\lceil \frac{1}{6} \log_2 \log_2(r_{V_i}(f)) \right\rceil = b_{\mathbf{LNBP}_\delta}(r_{V_i}(f))$ and we are also done as s_i is an integer and $s_i \geq \log_2 \log_2(r_{V_i}(f))$.

We now assume $r_{V_i}(f) \geq 4^{2^\delta}$. In particular this implies that $s_i \geq 1$.

Observe that for all $h \colon \{0,1\}^{V_i} \to \{0,1\}$ a subfunction of g on V_i, by definition, there exists a partial assignment $\rho \in \{0,1\}^{V \setminus V_i \cup [\delta]}$ such that $g|_\rho = h$, so it is not too difficult to see that h is computed by the Boolean BP of size s_i obtained from P by:

1. removing all non-sink vertices labelled by elements not in V_i;

2. defining the new start vertex as the only vertex whose label is in V_i and reachable from the start vertex of P by a path of nodes labelled by elements outside of V_i and arcs labelled consistently with ρ;

3. connecting a vertex u to a vertex v by an arc labelled by $a \in \{0,1\}$ if and only if there exists a path from u to v in P verifying that any intermediate vertex of the path is labelled by an element outside of V_i, the first arc is labelled by a and each arc (but the first one) is labelled consistently with ρ.

Thus, $r_{V_i}(g)$ is necessarily upper-bounded by the number of syntactically distinct such BPs we can build from P that way. Since, for such a BP, there are at most s_i+2 possible choices for the start vertex and by functionality of the set of arcs labelled 0 and the set of arcs labelled 1 seen as successor relations, there are at most $(s_i+1)^{s_i}$ possible choices for the set of arcs labelled 0, as well as at most $(s_i+1)^{s_i}$ possible choices for the set of arcs labelled 1, $r_{V_i}(g)$ is at most $(s_i+2)(s_i+1)^{2s_i}$. Assuming $2 \leq s_i$ we get:

$$\begin{aligned} r_{V_i}(g) \leq (s_i+2)(s_i+1)^{2s_i} &= 2^{\log_2(s_i+2)+2s_i\log_2(s_i+1)} \\ &\leq 2^{3s_i\log_2(s_i+2)} \\ &\leq 2^{6s_i\log_2(s_i)} \qquad \text{as } 2 \leq s_i \ . \end{aligned}$$

It follows that $s_i \geq \frac{1}{6} \cdot \frac{\log_2(r_{V_i}(g))}{\log_2\log_2(r_{V_i}(g))}$. If $s_i = 1$ it is clear as $r_{V_i}(g)$ is then at most 4, and if $s_i \geq 2$ we would otherwise have

$$\begin{aligned} s_i\log_2(s_i) &< \frac{1}{6} \cdot \frac{\log_2(r_{V_i}(g))}{\log_2\log_2(r_{V_i}(g))} \log_2\Big(\frac{1}{6} \cdot \frac{\log_2(r_{V_i}(g))}{\log_2\log_2(r_{V_i}(g))}\Big) \\ &= \frac{\log_2(r_{V_i}(g))}{6} - \frac{\log_2(r_{V_i}(g))\log_2\big(6\log_2\log_2(r_{V_i}(g))\big)}{6\log_2\log_2(r_{V_i}(g))} \\ &< \frac{\log_2(r_{V_i}(g))}{6} \end{aligned}$$

(observe that the last inequality follows from the fact that the subtracted member must necessarily be positive since $r_{V_i}(g) \geq 4$). From Lemma 2.1.16 we have $r_{V_i}(g) \geq r_{V_i}(f)^{\frac{1}{2^\delta}}$. The function $\frac{\log_2(x)}{\log_2\log_2(x)}$ being non-decreasing on $\big[e^{e\ln(2)}, +\infty\big[$ and as $\frac{\log_2(x)}{\log_2\log_2(x)} \leq 2$ for $x \in \big[4, e^{e\ln(2)}\big]$, we get for $r_{V_i}(f)^{\frac{1}{2^\delta}} \geq 4$:

$$s_i \geq \frac{1}{6} \cdot \frac{\log_2(r_{V_i}(f))}{2^\delta\big(\log_2\log_2(r_{V_i}(f)) - \delta\big)} \ .$$

In conclusion, for all $\delta, n \in \mathbb{N}$, and any n-ary Boolean function f on V and any par-

tition $V_1, \ldots, V_p$ of V, it holds that $\mathbf{LNBP}_\delta(f) \geq \sum_{i=1}^{p} b_{\mathbf{LNBP}_\delta}(r_{V_i}(f))$, hence $b_{\mathbf{LNBP}_\delta} \in \mathcal{N}_{\mathbf{LNBP}_\delta}$. It also directly follows that $b_{\mathbf{BP}} \in \mathcal{N}_{\mathbf{BP}}$ because $b_{\mathbf{BP}}$ is non-decreasing and $b_{\mathbf{BP}}(m) \leq b_{\mathbf{LNBP}_0}(m)$ for all $m \in \mathbb{N}_{>0}$. □

Let us define $\Gamma_{\mathbf{LNBP}} \colon [2, +\infty[\times \mathbb{N}_{>0} \to \mathbb{R}$ by

$$\Gamma(x, \delta) = \begin{cases} \max\left\{ \frac{x^2}{2^\delta (\log_2 x - \delta) \log_2 x}, x \right\} & \text{if } 2^{\delta+1} \leq x \\ x & \text{otherwise} \end{cases}$$

for all $(x, \delta) \in [2, +\infty[\times \mathbb{N}_{>0}$. Using the previous proposition and Lemma 2.1.9, we can immediately derive the following asymptotic lower bound on $\mathrm{N}_{\mathrm{ISA}}^{\mathbf{LNBP}_{\Delta(n)}}$ for any $\Delta \colon \mathbb{N} \to \mathbb{N}$.

Proposition 2.6.2. $\mathrm{N}_{\mathrm{ISA}}^{\mathbf{LNBP}_{\Delta(n)}}(n) \in \Omega\big(\Gamma_{\mathbf{LNBP}}(n, \Delta(n))\big)$ *for any* $\Delta \colon \mathbb{N} \to \mathbb{N}$. *In particular,* $\mathrm{N}_{\mathrm{ISA}}^{\mathbf{BP}}(n) \in \Omega\big(\frac{n^2}{\log^2 n}\big)$.

Proof. Let $\Delta \colon \mathbb{N} \to \mathbb{N}$. Let $n \in \mathbb{N}, n \geq 32$. Let $V_1, \ldots, V_p, U$ be a partition of $[n]$ such that $r_{V_i}(\mathrm{ISA}_n) = 2^q$ for all $i \in [p]$ where $p, q \in \mathbb{N}_{>0}$ verify $p \geq \frac{1}{32} \cdot \frac{n}{\log_2 n}$ and $q \geq \frac{n}{16}$ as given by Lemma 2.1.9. We have, as $\log_2\left(\frac{n}{16}\right) \geq \frac{\log_2 n}{16}$ and $x \mapsto \left\lceil \frac{1}{6} h_{\mathbf{LNBP}_{\Delta(n)}}(x) \right\rceil$ is non-decreasing on $[4, +\infty[$ (facts which are not too difficult to prove), that

$$\begin{aligned}
&\mathrm{N}_{\mathrm{ISA}}^{\mathbf{LNBP}_{\Delta(n)}}(n) \\
\geq &\sum_{i=1}^{p} b_{\mathbf{LNBP}_{\Delta(n)}}(r_{V_i}(\mathrm{ISA}_n)) + b_{\mathbf{LNBP}_{\Delta(n)}}(r_U(\mathrm{ISA}_n)) \\
\geq &\sum_{i=1}^{p} b_{\mathbf{LNBP}_{\Delta(n)}}(2^q) \\
\geq &\frac{1}{6} \cdot \frac{1}{32} \cdot \frac{n}{\log_2 n} \cdot \begin{cases} \max\left\{ \frac{\frac{n}{16}}{2^{\Delta(n)}(\log_2(\frac{n}{16}) - \Delta(n))}, \log_2(\frac{n}{16}) \right\} & \text{if } 4^{2^{\Delta(n)}} \leq 2^{\frac{n}{16}} \\ \log_2(\frac{n}{16}) & \text{otherwise} \end{cases} \\
\geq &\frac{1}{192} \cdot \frac{n}{\log_2 n} \cdot \frac{1}{16} \cdot \begin{cases} \max\left\{ \frac{n}{2^{\Delta(n)}(\log_2(n) - \Delta(n))}, \log_2 n \right\} & \text{if } 2^{\Delta(n)+5} \leq n \\ \log_2 n & \text{otherwise} \end{cases} \\
\geq &\frac{c}{3072} \cdot \frac{n}{\log_2 n} \cdot \begin{cases} \max\left\{ \frac{n}{2^{\Delta(n)}(\log_2(n) - \Delta(n))}, \log_2 n \right\} & \text{if } 2^{\Delta(n)+1} \leq n \\ \log_2 n & \text{otherwise} \end{cases} \qquad \text{for some } c \text{ below} \\
= &\frac{c}{3072} \cdot \begin{cases} \max\left\{ \frac{n^2}{2^{\Delta(n)}(\log_2(n) - \Delta(n)) \log_2 n}, n \right\} & \text{if } 2^{\Delta(n)+1} \leq n \\ n & \text{otherwise} \end{cases}
\end{aligned}$$

$$= \frac{c}{3072} \cdot \Gamma_{\mathbf{LNBP}}(n, \Delta(n)) \qquad \text{as desired .}$$

In order to show the inequality above it suffices to show that $\log_2 x \geq \frac{cx}{2^\alpha(\log_2(x)-\alpha)}$ for $x \in I = [2^{\alpha+1}, 2^{\alpha+5}]$, $\alpha \geq 0$ and some constant c.

It suffices to show that the function $f(x) = 2^\alpha \log_2(x) \log_2(\frac{x}{2^\alpha}) - cx$ is non-decreasing on I. This concludes the claim as $f(2^{\alpha+1}) = 2^\alpha(\alpha+1) - c2^{\alpha+1} \geq 0$ when $\alpha \geq 0$ and $c \leq \frac{1}{2}$. To see this notice that the derivative of f is $2^\alpha\big(\frac{\log_2 x}{x \ln 2} + \frac{\log_2(\frac{x}{2^\alpha})}{x \ln 2}\big) - c$ that has the same sign as $g(x) = 2^\alpha \log_2\big(\frac{x^2}{2^\alpha}\big) - xc \ln 2$ for $x \in I$.

The derivative of g is $\frac{2^{\alpha+1}}{x \ln 2} - c \ln 2$ that vanishes for a value $x_0 = \frac{2^{\alpha+1}}{c(\ln 2)^2}$. Assuming $c \leq \frac{1}{2^4(\ln 2)^2}$ we have $x_0 \geq 2^{\alpha+5}$ and the derivative of g is always non-negative on I.

We have $g(2^{\alpha+1}) = 2^\alpha(\alpha + 2 - 2c \ln 2)$ which is non-negative as soon as $c \leq \frac{1}{\ln 2}$. Hence g is non-negative on I.

Hence taking $c = \frac{1}{2^4(\ln 2)^2}$ yields the desired result. □

Now we show that for all $\delta \in \mathbb{N}$, $b_{\mathbf{LNBP}_\delta}$ is in fact an asymptotically largest function in $\mathcal{N}_{\mathbf{LNBP}_\delta}$ (as well as for $b_{\mathbf{BP}}$ and $\mathcal{N}_{\mathbf{BP}}$) and that the previous bound is in fact also the asymptotically largest we may obtain, using the meta-results of Section 2.3. To do this, we appeal to our upper bound from Theorem 2.4.1 on the size of a δ-LNBP (or a deterministic BP) computing $\mathrm{ISA}_{k,\ell}$ and apply Lemma 2.3.2.

Proposition 2.6.3. *There exists a constant $c \in \mathbb{R}_{>0}$ verifying that for each $\delta \in \mathbb{N}$, any $b \in \mathcal{N}_{\mathbf{LNBP}_\delta}$ is such that $b(m) \leq c \cdot b_{\mathbf{LNBP}_\delta}(m)$ for all $m \in \mathbb{N}, m \geq 4$. In particular, there exists a constant $c' \in \mathbb{R}_{>0}$ verifying that any $b \in \mathcal{N}_{\mathbf{BP}}$ is such that $b(m) \leq c' \cdot b_{\mathbf{BP}}(m)$ for all $m \in \mathbb{N}, m \geq 4$.*

Proof. Let $\delta \in \mathbb{N}$. Let $g\colon [1, +\infty[\to \mathbb{R}_{\geq 0}$ be the non-decreasing function defined by

$$g(x) = 13 \cdot 2^x \cdot \begin{cases} \max\{\frac{2^{x-\delta}}{x-\delta}, x\} & \text{if } \delta + 1 \leq x \\ x & \text{otherwise} \end{cases}$$

for all $x \in [1, +\infty[$.

Theorem 2.4.1 tells us that for all $k \in \mathbb{N}_{>0}$, we have

$$\begin{aligned} \mathbf{LNBP}_\delta(\mathrm{ISA}_{k,k}) &\leq \begin{cases} 12 \cdot 2^k \max\{\frac{2^{k-\delta}}{k-\delta}, k\} + \frac{2^{2k-\delta}}{k-\delta} & \text{if } \delta + 1 \leq k \\ 2^k(3k+1) + 2 \cdot 2^k & \text{otherwise} \end{cases} \\ &\leq \begin{cases} 2^k\big(12 \max\{\frac{2^{k-\delta}}{k-\delta}, k\} + \frac{2^{k-\delta}}{k-\delta}\big) & \text{if } \delta + 1 \leq k \\ 6 \cdot 2^k k & \text{otherwise} \end{cases} \qquad \text{as } k \geq 1 \end{aligned}$$

$$\leq g(k)$$

and moreover, $\frac{g(k+1)}{g(k)} \leq 4$ for all $k \in \mathbb{N}_{>0}$. Indeed, let $k \in \mathbb{N}_{>0}$, there are two cases to consider:

- if $g(k+1) = 13 \cdot 2^{k+1}(k+1)$ then notice that we always have $g(k) \geq 13 \cdot 2^k \cdot k$. Therefore we get $\frac{g(k+1)}{g(k)} \leq \frac{13 \cdot 2^{k+1}(k+1)}{13 \cdot 2^k k} = 2(1 + \frac{1}{k}) \leq 4$;
- otherwise $g(k+1) = 13 \cdot 2^{k+1} \frac{2^{k+1-\delta}}{k+1-\delta}$ and notice that either $g(k) \geq 13 \cdot 2^k \frac{2^{k-\delta}}{k-\delta}$ or $k = \delta$. If $k = \delta$ it is simple to check that $\frac{g(k+1)}{g(k)} \leq 4$, otherwise we have $\frac{g(k+1)}{g(k)} \leq \frac{13 \cdot 2^{k+1} \frac{2^{k+1-\delta}}{k+1-\delta}}{13 \cdot 2^k \frac{2^{k-\delta}}{k-\delta}} = 4 \cdot \frac{k-\delta}{k+1-\delta} \leq 4$.

Therefore, by Lemma 2.3.2, any $b \in \mathcal{N}_{\mathbf{LNBP}_\delta}$ verifies

$$\begin{aligned}
b(m) &\leq 4 \cdot \frac{g(\log_2 \log_2 m)}{\log_2 m} \\
&= 4 \cdot \frac{13 \cdot 2^{\log_2 \log_2 m} \begin{cases} \max\left\{ \frac{2^{\log_2 \log_2 (m) - \delta}}{\log_2 \log_2 (m) - \delta}, \log_2 \log_2 m \right\} & \text{if } \delta + 1 \leq \log_2 \log_2 m \\ \log_2 \log_2 m & \text{otherwise} \end{cases}}{\log_2 m} \\
&= 52 \cdot \begin{cases} \max\left\{ \frac{\log_2 m}{2^\delta (\log_2 \log_2 (m) - \delta)}, \log_2 \log_2 m \right\} & \text{if } 2^{2^{\delta+1}} \leq m \\ \log_2 \log_2 m & \text{otherwise} \end{cases} \\
&\leq c \cdot b_{\mathbf{LNBP}_\delta}(m)
\end{aligned}$$

for all $m \in \mathbb{N}, m \geq 4$, where $c \in \mathbb{R}_{>0}$ is a sufficiently large constant.

In the case where $\delta = 0$ notice that for $m \geq 4$ we have $\log_2 \log_2 m \leq d \cdot \frac{\log_2 x}{\log_2 \log_2 x}$ for some suitable constant d.

So we can also conclude that for any $b \in \mathcal{N}_{\mathbf{BP}} = \mathcal{N}_{\mathbf{LNBP}_0}$, we have

$$b(m) \leq 52 \cdot \max\left\{ \frac{\log_2 m}{\log_2 \log_2 m}, \log_2 \log_2 m \right\} \leq c' \cdot \frac{\log_2 m}{\log_2 \log_2 m}$$

for all $m \in \mathbb{N}, m \geq 4$, where $c' \in \mathbb{R}_{>0}$ is a sufficiently large constant. □

Finally, using this and Lemma 2.3.3, we get the following result, showing that the asymptotically greatest lower bound we may expect using Nečiporuk's method for $\mathbf{LNBP}_\delta$ for any $\delta \in \mathbb{N}$ is (asymptotically) equivalent to the lower bound for ISA given in Proposition 2.6.2.

Theorem 2.6.4. *For any family of Boolean functions $F = \{f_n\}_{n\in\mathbb{N}}$ and any $\Delta\colon \mathbb{N}\to\mathbb{N}$, $\mathrm{N}_F^{\mathbf{LNBP}_{\Delta(n)}}(n) \in \mathrm{O}\big(\Gamma_{\mathbf{LNBP}}(n,\Delta(n))\big)$.*

In particular, $\mathrm{N}_F^{\mathbf{BP}}(n) \in \mathrm{O}\big(\frac{n^2}{\log^2 n}\big)$.

Proof. Let $\delta\in\mathbb{N}$. We aim at applying Lemma 2.3.3 which requires four hypotheses, (i) to (iv).

For (i), we set h as $h(x) = h_{\mathbf{LNBP}_\delta}(x)$ for all $x\in[4,+\infty[$ and $x_0 = 2^8$. One can verify that h is non-decreasing on $[2^8,+\infty[$.

For (ii), for all $x\in[4,+\infty[$, we have $h(2^x)\geq \log_2\log_2(2^x) = \log_2 x$.

For (iii), for all $v,v'\in\mathbb{N}$ verifying $2^{2^v}\geq 2^8$ and $2^{2^{v'}}\geq 2^8$, we need to show that $h(2^{2^v})+h(2^{2^{v'}})\leq h(2^{2^{v+v'}})$. There are three cases to consider.

- If $h(2^{2^v}) = \log_2\log_2(2^{2^v}) = v$ and $h(2^{2^{v'}}) = \log_2\log_2(2^{2^{v'}}) = v'$, then $h(2^{2^v}) + h(2^{2^{v'}}) = v+v' = \log_2\log_2(2^{2^{v+v'}}) \leq h(2^{2^{v+v'}})$.

- If $h(2^{2^v}) = \frac{\log_2(2^{2^v})}{2^\delta(\log_2\log_2(2^{2^v})-\delta)} = \frac{2^{v-\delta}}{v-\delta} > v$ and $h(2^{2^{v'}}) = \log_2\log_2(2^{2^{v'}}) = v'$, then we necessarily have $v = \delta+\eta$ for some $\eta>0$.

  Notice that $\eta\geq 2$ because if $\eta = 1$ then $v<2$, a contradiction.

  We conclude by showing that $h(2^{2^v})+h(2^{2^{v'}}) = \frac{2^{v-\delta}}{v-\delta} + v' \leq \frac{2^{\eta+v'}}{\eta+v'} \leq h(2^{2^{v+v'}})$.

  Only the first inequality is non immediate. To see it, consider the function $f(x) = 2^{\eta+x} - (\frac{2^\eta}{\eta}+x)(\eta+x)$. A simple calculation shows that it is non-decreasing for $x\geq 2$ and $\eta\geq 2$. The inequality follows as $f(2)$ is non-negative when $\eta\geq 2$.

- In the remaining case $h(2^{2^v}) = \frac{2^{v-\delta}}{v-\delta} > v$ and $h(2^{2^{v'}}) = \frac{2^{v'-\delta}}{v'-\delta} > v'$. It implies that $v\geq\delta+1$ and $v'\geq\delta+1$. Arguing as above we actually have $v\geq\delta+2$ and $v'\geq\delta+2$, otherwise v or v' would be smaller than 2. We then have: $h(2^{2^v})+h(2^{2^{v'}}) = \frac{2^{v-\delta}}{v-\delta} + \frac{2^{v'-\delta}}{v'-\delta} \leq \frac{2^{v+v'-2\delta}}{(v-\delta)(v'-\delta)} \leq \frac{2^{v+v'-2\delta}}{v+v'-2\delta} \leq h(2^{2^{v+v'-\delta}}) \leq h(2^{2^{v+v'}})$.

  The first and second inequality are because $x+y\leq xy$ when both x and y are greater than 2 (in the second case we use that $v-\delta\geq 2$ and $v'-\delta\geq 2$). The third one is by definition of h and the last one by monotonicity of h.

For (iv), by Proposition 2.6.3, we know that there exists a constant $\alpha\in\mathbb{R}_{>0}$ verifying that any $b\in\mathcal{N}_{\mathbf{LNBP}_\delta}$ is such that $b(m)\leq\alpha\cdot h(m)$ for all $m\in\mathbb{N}, m\geq 4$.

We can therefore apply Lemma 2.3.3 with $x_0 = 2^8$ and get that for any family of Boolean functions $F=\{f_n\}_{n\in\mathbb{N}}$ and all $n\in\mathbb{N}, n\geq 8$,

$$\mathrm{N}_F^{\mathbf{LNBP}_\delta}(n) \leq \alpha\cdot\big(4+h(2^8)\big)\cdot\frac{n}{\log_2 n}\cdot h(2^n)$$

$$
\begin{aligned}
&= c \cdot \frac{n}{\log_2 n} \cdot \begin{cases} \max\{\frac{n}{2^\delta(\log_2(n)-\delta)}, \log_2 n\} & \text{if } 2^{2^{\delta+1}} \leq 2^n \\ \log_2 n & \text{otherwise} \end{cases} \\
&= c \cdot \begin{cases} \max\{\frac{n^2}{2^\delta(\log_2(n)-\delta)\log_2 n}, n\} & \text{if } 2^{\delta+1} \leq n \\ n & \text{otherwise} \end{cases} \\
&= c \cdot \Gamma_{\mathbf{LNBP}}(n, \delta)
\end{aligned}
$$

for some suitable constant $c \in \mathbb{R}_{>0}$.

Thus, since this holds for all $\delta \in \mathbb{N}$, we get the desired result. □

2.7 Deterministic and Limited Non-deterministic Formulæ

In this section, we focus on the model of Boolean binary formulæ and its limited non-deterministic variant. **L** is one of the two measures that were considered in Nečiporuk's original article Nečiporuk [1966] who gave an $\Omega(\frac{n^2}{\log n})$ lower bound for this complexity measure. If the model is restricted to the case of binary formulæ where only 2-ary AND and OR gates can be used, stronger lower bounds can be proven, the best known for instance being almost cubic and due to Håstad (see Håstad [1998]). Just as in Section 2.6, results for the Nečiporuk method for binary formulæ are known (see for instance [Wegener, 1987, Chapter 8, Section 7]), but we do not know about any attempt to consider the method in its full generality: an approach that would explicitly try to find the best Nečiporuk function as done in Alon and Zwick [1989] for the case of BPs rather than just giving one.

Concerning limited non-deterministic binary formulæ, Nečiporuk's lower bound method never seems to have been applied to the associated complexity measure, at least in a direct combinatorial sense that excludes Klauck's communication complexity formulation of the method Klauck [2007].

For all $\delta \in \mathbb{N}$, let us define the function $b_{\mathbf{LL}_\delta} \colon \mathbb{N}_{>0} \to \mathbb{N}$ given by

$$
b_{\mathbf{LL}_\delta}(m) = \begin{cases} \left\lceil \frac{1}{4} \max\{\frac{\log_2 m}{2^\delta}, \log_2 \log_2 m\} \right\rceil & \text{if } m \geq 4 \\ 0 & \text{otherwise} \end{cases}
$$

for all $m \in \mathbb{N}_{>0}$. We denote by $b_{\mathbf{L}}$ the function $b_{\mathbf{LL}_0}$.

We first prove that $b_{\mathbf{L}} \in \mathcal{N}_{\mathbf{L}}$ and $b_{\mathbf{LL}_\delta} \in \mathcal{N}_{\mathbf{LL}_\delta}$. This is similar to the limited non-

deterministic branching program case.

Proposition 2.7.1. *$b_{\mathbf{LL}_\delta} \in \mathcal{N}_{\mathbf{LL}_\delta}$ for all $\delta \in \mathbb{N}$. In particular, $b_{\mathbf{L}} \in \mathcal{N}_{\mathbf{L}}$.*

Proof. Let $\delta \in \mathbb{N}$. It is fairly obvious that $b_{\mathbf{LL}_\delta}$ is non-decreasing.

Let f be an n-ary Boolean function on V, that verifies without loss of generality $V \cap [\delta] = \emptyset$, and $V_1, \dots, V_p$ a partition of V. Let ϕ be a Boolean δ-LNBF computing f and let g be the $(n+\delta)$-ary Boolean function computed by ϕ when considering the δ non-deterministic bits as regular input variables.

For all $i \in [p]$ we will denote by $s_i \in \mathbb{N}$ the number of leaves in ϕ labelled by literals whose variable indices are in V_i, as well as $q \in \mathbb{N}$ the number of leaves in ϕ labelled by literals whose variable indices are not in V. It is clear that $|\phi| = \sum_{i=1}^{p} s_i + q \geq \sum_{i=1}^{p} s_i$. To conclude it remains to show that $s_i \geq b_{\mathbf{LL}_\delta}(r_{V_i}(f))$ for all $i \in [p]$.

Fix $i \in [p]$. The claim is obvious if $r_{V_i}(f) \leq 3$ hence we assume $r_{V_i}(f) \geq 4$. Let V_i' be the subset of V_i containing all indices of variables on which f depends. Then, by Lemma 2.1.2, $r_{V_i}(f) = r_{V_i'}(f)$. Moreover for each $l \in V_i'$, ϕ contains at least one leaf labelled by a literal whose variable index is l. By Lemma 2.1.1, it follows that $r_{V_i}(f) = r_{V_i'}(f) \leq 2^{2^{|V_i'|}} \leq 2^{2^{s_i}}$. So we can conclude that $s_i \geq \log_2 \log_2(r_{V_i}(f))$.

If $r_{V_i}(f) \leq 2^{2^{\delta+1}}$, we have

$$\left\lceil \frac{1}{4} \cdot \frac{\log_2(r_{V_i}(f))}{2^\delta} \right\rceil \leq \left\lceil \frac{1}{4} \cdot \frac{\log_2\left(2^{2^{\delta+1}}\right)}{2^\delta} \right\rceil = 1 \leq \left\lceil \frac{1}{4} \cdot \log_2 \log_2(r_{V_i}(f)) \right\rceil .$$

and therefore $s_i \geq b_{\mathbf{LL}_\delta}(r_{V_i}(f))$.

It remains to consider the case where $r_{V_i}(f) > 2^{2^{\delta+1}}$. Notice that this implies $s_i > 0$, as $2^{2^{s_i}} \geq r_{V_i}(f)$.

This part of the proof is taken from classical references, e.g. [Wegener, 1987, Proof of Theorem 7.1] or [Jukna, 2012, Proof of Theorem 6.16]. We denote by T_i the subtree of ϕ consisting of all paths from a leaf with a label in V_i to the root of ϕ. This tree has nodes of fan-in 0, 1 or 2 and is non-empty since $s_i > 0$. Let W_i be the set of nodes of T_i that have fan-in 2 and notice that $|W_i| \leq s_i - 1$. Let P_i be the set of paths in T_i starting from a leaf or a node in W_i and ending in a node in W_i or in the root of T_i and containing no node in W_i as inner node. Notice that $|P_i| \leq 2\,|W_i| + 1 \leq 2s_i$.

For any partial assignment $\rho \in \{0,1\}^{V \setminus V_i \cup [\delta]}$, we obtain a formula $\phi|_\rho$ of size s_i computing $g|_\rho$ by replacing each variable in $V \setminus V_i \cup [\delta]$ by the appropriate constant given by ρ. This assignment induces that any part of $\phi|_\rho$ corresponding to a path p in P_i, either computes a constant function, or is the identity or negates its input. Reciprocally any of

these four choices on p induces a subfunction of g. Hence we have $r_{V_i}(g) \leq 4^{|P_i|} \leq 2^{4s_i}$.

As g is a proof-checker function for f, from Lemma 2.1.16 it follows that $r_{V_i}(g) \geq r_{V_i}(f)^{\frac{1}{2^\delta}}$, therefore

$$s_i \geq \frac{1}{4}\log_2(r_{V_i}(g)) \geq \frac{1}{4}\log_2\Big(r_{V_i}(f)^{\frac{1}{2^\delta}}\Big) = \frac{1}{4}\cdot\frac{\log_2(r_{V_i}(f))}{2^\delta} \ .$$

Altogether, we have

$$s_i \geq \frac{1}{4}\max\Big\{\frac{\log_2(r_{V_i}(f))}{2^\delta}, \log_2\log_2(r_{V_i}(f))\Big\} \ ,$$

which implies that $s_i \geq b_{\mathbf{LL}_\delta}(r_{V_i}(f))$ as s_i is integral.

In conclusion for any n-ary Boolean function f on V and any partition $V_1, \ldots, V_p$ of V, it holds that $\mathbf{LL}_\delta(f) \geq \sum_{i=1}^p b_{\mathbf{LL}_\delta}(r_{V_i}(f))$, hence $b_{\mathbf{LL}_\delta} \in \mathcal{N}_{\mathbf{LL}_\delta}$. □

Using this and Lemma 2.1.9, we can immediately derive the following asymptotic lower bound on $\mathrm{N}_{\mathrm{ISA}}^{\mathbf{LL}_{\Delta(n)}}$.

Proposition 2.7.2. $\mathrm{N}_{\mathrm{ISA}}^{\mathbf{LL}_{\Delta(n)}}(n) \in \Omega\big(\max\big\{\frac{n^2}{2^{\Delta(n)}\log_2 n}, n\big\}\big)$ *for any* $\Delta\colon \mathbb{N}\to\mathbb{N}$. *In particular,* $\mathrm{N}_{\mathrm{ISA}}^{\mathbf{L}}(n) \in \Omega\big(\frac{n^2}{\log n}\big)$.

Proof. Let $\Delta\colon \mathbb{N}\to\mathbb{N}$. Let $n \in \mathbb{N}, n \geq 32$. Let $V_1, \ldots, V_p, U$ be a partition of V such that $r_{V_i}(\mathrm{ISA}_n) = 2^q$ for all $i \in [p]$ where $p, q \in \mathbb{N}_{>0}$ verify $p \geq \frac{1}{32}\cdot\frac{n}{\log_2 n}$ and $q \geq \frac{n}{16}$ as given by Lemma 2.1.9. We have

$$\begin{aligned}
\mathrm{N}_{\mathrm{ISA}}^{\mathbf{LL}_{\Delta(n)}}(n) &\geq \sum_{i=1}^{p} b_{\mathbf{LL}_{\Delta(n)}}(r_{V_i}(\mathrm{ISA}_n)) + b_{\mathbf{LL}_{\Delta(n)}}(r_U(\mathrm{ISA}_n)) \\
&\geq \sum_{i=1}^{p} \frac{1}{4}\cdot\max\Big\{\frac{\log_2(2^q)}{2^{\Delta(n)}}, \log_2\log_2(2^q)\Big\} \\
&= p\cdot\frac{1}{4}\cdot\max\Big\{\frac{q}{2^{\Delta(n)}}, \log_2 q\Big\} \\
&\geq c_1\cdot\frac{n}{\log_2 n}\cdot\max\Big\{\frac{n}{16\cdot 2^{\Delta(n)}}, \log_2\Big(\frac{n}{16}\Big)\Big\} \\
&\geq c_2\cdot\max\Big\{\frac{n^2}{2^{\Delta(n)}\log_2 n}, n\Big\} \quad \text{because } n \geq 32 \text{ implies } \log_2\Big(\frac{n}{16}\Big) \geq \frac{\log_2 n}{16}
\end{aligned}$$

for some suitable constants $c_1, c_2 \in \mathbb{R}_{>0}$. □

We now show that for all $\delta \in \mathbb{N}$, $b_{\mathbf{LL}_\delta}$ is in fact an asymptotically largest function in $\mathcal{N}_{\mathbf{LL}_\delta}$. To this end, we appeal to the upper bound on $\mathbf{LL}_\delta(\mathrm{ISA}_{k,\ell})$ from Theorem 2.4.1 and apply Lemma 2.3.2.

Proposition 2.7.3. *There exists a constant $c \in \mathbb{R}_{>0}$ verifying that for each $\delta \in \mathbb{N}$, any $b \in \mathcal{N}_{\mathbf{LL}_\delta}$ is such that $b(m) \leq c \cdot b_{\mathbf{LL}_\delta}(m)$ for all $m \in \mathbb{N}, m \geq 4$.*

Proof. Fix $\delta \in \mathbb{N}$. Let $g\colon [1, +\infty[\to \mathbb{R}_{\geq 0}$ be the non-decreasing function defined by $g(x) = 15 \cdot 2^x \cdot \max\{2^{x-\delta}, x\}$ for all $x \in [1, +\infty[$.

Notice that $\frac{g(k+1)}{g(k)} \leq 4$ for all $k \in \mathbb{N}_{>0}$.

Moreover, from Theorem 2.4.1 we have

$$\mathbf{LL}_\delta(\mathrm{ISA}_{k,k}) \leq 12 \cdot 2^k \cdot \max\{2^{k-\delta}, k\} + 3 \cdot 2^k \leq g(k)$$

for all $k \in \mathbb{N}_{>0}$. Therefore, by Lemma 2.3.2, any $b \in \mathcal{N}_{\mathbf{LL}_\delta}$ verifies

$$\begin{aligned} b(m) &\leq 4 \cdot \frac{g(\log_2 \log_2 m)}{\log_2 m} \\ &= 4 \cdot \frac{15 \cdot 2^{\log_2 \log_2 m} \cdot \max\{2^{\log_2 \log_2(m) - \delta}, \log_2 \log_2 m\}}{\log_2 m} \\ &= 60 \cdot \max\left\{\frac{\log_2 m}{2^\delta}, \log_2 \log_2 m\right\} \\ &\leq c \cdot b_{\mathbf{LL}_\delta}(m) \end{aligned}$$

for all $m \in \mathbb{N}, m \geq 4$, where $c \in \mathbb{R}_{>0}$ is a sufficiently large constant. □

Finally, using this and Lemma 2.3.3, we show that the asymptotically greatest lower bound we may expect using Nečiporuk's method for $\mathbf{LL}_\delta$ for any $\delta \in \mathbb{N}$ is (asymptotically) equivalent to the lower bound for ISA given in Proposition 2.7.2.

Theorem 2.7.4. *For any family of Boolean functions $F = \{f_n\}_{n \in \mathbb{N}}$ and any $\Delta\colon \mathbb{N} \to \mathbb{N}$, $\mathrm{N}_F^{\mathbf{LL}_{\Delta(n)}}(n) \in \mathrm{O}\big(\max\{\frac{n^2}{2^{\Delta(n)} \log_2 n}, n\}\big)$.*

In particular, $\mathrm{N}_F^{\mathbf{L}}(n) \in \mathrm{O}\big(\frac{n^2}{\log n}\big)$.

Proof. Fix $\delta \in \mathbb{N}$. We aim at applying Lemma 2.3.3 which requires four hypotheses, (i) to (iv).

For (i), let $h\colon [4, +\infty[\to \mathbb{R}_{\geq 0}$ be the function defined by

$$h(x) = \max\left\{\frac{\log_2 x}{2^\delta}, \log_2 \log_2 x\right\}$$

for all $x \in [4, +\infty[$ and $x_0 = 2^8$; as required, h is non-decreasing on $[2^8, +\infty[$.

For (ii), for all $x \in [4, +\infty[$, we have $h(2^x) \geq \log_2 \log_2(2^x) = \log_2 x$.

For (iii), for all $v, v' \in \mathbb{N}$ verifying $2^{2^v} \geq 2^8$ and $2^{2^{v'}} \geq 2^8$, we have $h(2^{2^v}) + h(2^{2^{v'}}) \leq h(2^{2^{v+v'}})$. Indeed, let $v, v' \in \mathbb{N}$ such that $2^{2^v} \geq 2^8$ and $2^{2^{v'}} \geq 2^8$, there are two cases to consider.

- If $h(2^{2^v}) = \log_2 \log_2(2^{2^v}) = v$ and $h(2^{2^{v'}}) = \log_2 \log_2(2^{2^{v'}}) = v'$, then $h(2^{2^v}) + h(2^{2^{v'}}) = v + v' = \log_2 \log_2(2^{2^{v+v'}}) \leq h(2^{2^{v+v'}})$.
- Otherwise, there is at least one $w \in \{v, v'\}$ such that $h(2^{2^w}) = \frac{\log_2(2^{2^w})}{2^\delta} = 2^{w-\delta}$: assume without loss of generality that it is v. Then, since $2^{2^v} \geq 16$, we have $v \geq 2$, so by hypothesis, it follows that $2^{v-\delta} > v \geq 2$. Moreover, $h(2^{2^{v'}}) = \max\{2^{v'-\delta}, v'\} \leq 2^{v'}$, so using our usual observation about the relationship between the sum and the product of two real numbers greater than or equal to 2, we get $h(2^{2^v}) + h(2^{2^{v'}}) \leq 2^{v-\delta} + 2^{v'} \leq 2^{v+v'-\delta} = \frac{\log_2(2^{2^{v+v'}})}{2^\delta} \leq h(2^{2^{v+v'}})$.

For (iv), by Proposition 2.7.3, we know that there exists a constant $\alpha \in \mathbb{R}_{>0}$ verifying that any $b \in \mathcal{N}_{\mathbf{LL}_\delta}$ is such that $b(m) \leq \alpha \cdot h(m)$ for all $m \in \mathbb{N}, \geq 4$.

Therefore, by Lemma 2.3.3, for any family of Boolean functions $F = \{f_n\}_{n \in \mathbb{N}}$ and all $n \geq 8$, we have

$$\begin{aligned} \mathrm{N}_F^{\mathbf{LL}_\delta}(n) &\leq \alpha \cdot \big(4 + h(\lfloor 2^8 \rfloor)\big) \cdot \frac{n}{\log_2 n} \cdot h(2^n) \\ &\leq c \cdot \frac{n}{\log_2 n} \cdot \max\Big\{\frac{n}{2^\delta}, \log_2 n\Big\} \\ &= c \cdot \max\Big\{\frac{n^2}{2^\delta \log_2 n}, n\Big\} \end{aligned}$$

for some suitable constant $c \in \mathbb{R}_{>0}$.

Thus, since this holds for all $\delta \in \mathbb{N}$, we get the desired result. □

2.8 Final insights

In this chapter, we have proposed a general interpretation of what it means to say "the method of Nečiporuk". We have applied the method to several complexity measures, as reported in Table 2.I, and shown in particular that the limitations of the method are very much determined by the complexity of the Indirect Storage Access function under each measure, at least for those we studied in this chapter. We observe incidentally that this also applies to the size measure of $\mathcal{B}_2$-circuits: while one can show that $m \to \lceil \log_2 \log_2 m \rceil - 1$ is a Nečiporuk function for that measure, one sees that $\mathrm{ISA}_{k,\ell}$ can be

computed by a Boolean circuit of size at most $2^k(k + 2\ell) + 3 \cdot 2^\ell$ (so in linear size). This implies asymptotic optimality of that Nečiporuk function and further implies that the best lower bound one might expect using Nečiporuk's method for the size of Boolean circuits is linear. Thus, our framework also applies to Boolean circuits, even though it only allows to re-derive the well-known result that Nečiporuk's method is unable prove super-linear circuit size lower bounds. Indeed, as shown by Uhlig in 1991, there exist Boolean functions having exponentially many different subfunctions that can be computed by linear-size Boolean circuits (see [Jukna, 2012, Remark 6.19]). Note that our focus was not on optimising the constant factors in the bounds obtained, most of which can certainly be improved.

Our abstract definition of a Nečiporuk function was inspired by Alon and Zwick Alon and Zwick [1989]. Our definition has the benefit of not specifying the way in which such a function is obtained, be it some "semantic count" of the number of different Boolean functions computable with a given cost, some "syntactic count" of the number of different devices of that cost as done usually, or any other technique. While in the literature, "Nečiporuk-style theorems" refer to giving an explicit Nečiporuk function as defined in step 1 in Definition 2.3.1 Wegener [1987], Jukna [2012], Alon and Zwick [1989], it is natural to ask whether we could even further twist the definition of a Nečiporuk function to get more out of the method.

Looking at our meta-results and how we draw the limitation results for Nečiporuk's method applied to a specific measure $\mathbf{M}$, namely using an upper bound on the $\mathrm{ISA}_{k,k}$ function for all $k \in \mathbb{N}_{>0}$, we observe that the main weakness of the method arises from the requirement that a Nečiporuk function for $\mathbf{M}$ must verify the conditions presented in step 1 of Definition 2.3.1 for *every* Boolean function. A natural question is therefore whether restricting the class of Boolean functions for which these conditions should be verified by a Nečiporuk function for $\mathbf{M}$ would allow to get stronger Nečiporuk functions (and thus, lower bounds) for $\mathbf{M}$ for this specific class of Boolean functions. This seems to be an interesting question to us, but is not treated in this chapter. Another interesting similar question that was suggested to us by one of the anonymous referees of Beame et al. [2016] is whether relaxing the condition "for any partition" in step 1 of Definition 2.3.1 to "there exists a partition" would allow to get stronger Nečiporuk functions.

Complexity measure	**Best lower bound obtainable**
Size of NBPs	$\Theta\left(\frac{n^{3/2}}{\log n}\right)$
Size of $\oplus$BPs	$\Theta\left(\frac{n^{3/2}}{\log n}\right)$
Size of LNBPs using $\Delta(n)$ non-deterministic bits	$\Theta\left(\frac{n^2}{2^{\Delta(n)}(\log_2(n)-\Delta(n))\log_2 n}\right)$ $(\star)$
Size of BPs	$\Theta\left(\frac{n^2}{\log^2 n}\right)$
Size of LNBFs using $\Delta(n)$ non-deterministic bits	$\Theta\left(\frac{n^2}{2^{\Delta(n)}\log_2 n}\right)$ $(\star)$
Size of BFs	$\Theta\left(\frac{n^2}{\log n}\right)$

Table 2.I – Bounds for the Indirect Storage Access function, which this chapter shows to be the best lower bounds obtainable by Nečiporuk's method for any function. The star indicates that the true function is more complicated; however, this current formulation holds for all $\Delta\colon \mathbb{N} \to \mathbb{N}$ verifying $\Delta(n) \leq \log_2 n - 1$ for all $n \in \mathbb{N}$.

Chapter 3

Algebraic automata theory and computational complexity theory

As we have seen in Section 1.4 of Chapter 1, while AC^1 is believed to be well within the (presumed) NC hierarchy, its internal structure and its relationship to classes like P or even NP are far from being elucidated and are at the heart of some of the most challenging open questions in computational complexity theory at the time of writing of thesis. And this is still true if one restricts to NC^1.

The main objectives of this chapter are to introduce the program-over-monoid formalism, that gives an algebraic-automata-theoretic viewpoint on NC^1 and its subclasses, and to prove some modest new general results about this formalism. But first, we provide the necessary background in finite semigroup theory (Section 3.1) and its connections to finite automata theory (Section 3.2). We then move on in Section 3.3 to define programs over monoids, explain their importance as well as the challenges that arise in their study, and give their general properties. We also lay out how our contributions fit within the body of research aiming to understand their power in some restricted settings. In Section 3.4, we study the link between regular languages and programs over monoids, explain its importance and introduce a notion of tameness as a tool to understand it better. This prepares the way for the last two chapter of the thesis, each dedicated to analysing the computational power of programs over monoids in a particular restricted setting.

3.1 Finite semigroup theory

Finite semigroup theory has had, and still has, an immense impact in the study of finite automata and is at the core of what is called algebraic automata theory.

We present the necessary background in this section and refer the reader to the classical references by Eilenberg Eilenberg [1974, 1976] and Pin Pin [1986], as well as Pin [2016], for a more thorough introduction to the theory of finite semigroups.

3.1.1 Basic definitions

We start with the definition of a semigroup and that of a monoid.

Definition 3.1.1. An *internal composition law on a set* E is a mapping from $E \times E$ to E.

Definition 3.1.2. A *semigroup* is a pair $(S, *)$ where S is a non-empty set and $*$ is an internal composition law on S such that $*$ is *associative*: for all $x, y, z \in S$, $(x * y) * z = x * (y * z)$.

The *order of* $(S, *)$ is the number of elements of S.

Definition 3.1.3. A *monoid* is a pair $(M, *)$ where M is a set and $*$ is an internal composition law on M such that:

- $*$ is *associative*: for all $x, y, z \in M$, $(x * y) * z = x * (y * z)$;
- $*$ has a *neutral element* (also called *identity element*, or simply *identity*): there exists $e \in M$ such that for all $m \in M$, $e * m = m * e = m$.

The *order of* $(M, *)$ is the number of elements of M.

Remark 3.1.4. A semigroup (monoid) with one unique element is called a *trivial semigroup* (*monoid*).

Remark 3.1.5. For any semigroup or monoid $(S, *)$, we might sometimes write xy instead of $x * y$ when it is clear from the context and call $*$ *multiplication* or *product* of $(S, *)$.

For X and Y two subsets of S, we shall denote by $X * Y$, or XY when the context is clear, the subset $\{xy \mid x \in X, y \in Y\}$ of S. When $X = \{x\}$ for some $x \in S$ or $Y = \{y\}$ for some $y \in S$, we shall abuse notation and respectively write xY instead of XY or Xy instead of XY.

Formally, for any $p \in \mathbb{N}_{>0}$ and $x \in S$, we shall denote by $x^{*,p}$ the p-th power of x, that is to say, the unique element of S obtained by multiplying x (through $*$) iteratively by itself $p - 1$ times; we shall write x^p when the context is clear.

Remark 3.1.6. For each monoid $(M, *)$, it is easy to see that it has a unique neutral element, called *the neutral element* (or *the identity element* or *the identity*) of $(M, *)$, unambiguously denoted by $1_{(M,*)}$.

An example of a semigroup that isn't a monoid is $(\mathbb{N}_{>0}, +)$ where $+$ denotes the canonical addition on the set of positive natural numbers. However, $(\mathbb{N}, +)$ where $+$ denotes the canonical addition on the set of natural numbers is a monoid with identity 0, and in the same way $(\mathbb{Z}, +)$ is also a monoid with identity 0. Given $d \in \mathbb{N}, d \geq 2$, an example of a finite monoid is $(\mathbb{Z}/d\,\mathbb{Z}, +)$ where $+$ denotes the canonical addition modulo d, which has identity 0; but also $(\mathbb{Z}/d\,\mathbb{Z}, \times)$ where $\times$ denotes the canonical product modulo d, which is a monoid with identity 1.

On rare occasions, we might also appeal to the notion of groups, monoids in which each element can be inverted, defined next.

Definition 3.1.7. A *group* is a pair $(G, *)$ that is a monoid verifying additionally that any element of G is *invertible*: for each $g \in G$, there exists $g' \in G$ such that $g * g' = g' * g = 1_{(G,*)}$, called an *inverse of* g.

Looking at the previous examples, $(\mathbb{N}, +)$ isn't a group, but $(\mathbb{Z}, +)$ is. Similarly, given $d \in \mathbb{N}, d \geq 2$, $(\mathbb{Z}/d\,\mathbb{Z}, +)$ is a group while $(\mathbb{Z}/d\,\mathbb{Z}, \times)$ isn't.

It is sometimes useful to talk about the monoid obtained from a semigroup by adding an identity element to it if it does not already contain one, as defined below.

Definition 3.1.8. Let $(S, *)$ be a semigroup. We will denote by $(S, *)^1 = (S^1, *^1)$ the monoid equal to $(S, *)$ if the latter is already a monoid and otherwise such that $S^1 = S \cup \{1\}$ where 1 is a new element and $*^1$ is the internal composition law on S^1 extending $*$ such that $s *^1 1 = 1 *^1 s = s$ for all $s \in S^1$ and $s_1 *^1 s_2 = s_1 * s_2$ for all $s_1, s_2 \in S$.

A very important notion is the notion of semigroup and monoid congruences.

Definition 3.1.9. Let $(S, *)$ be a semigroup. A *congruence* $\sim$ *on the semigroup* $(S, *)$ is a stable equivalence relation on S, i.e. such that for all $s, t \in S$ and $u, v \in S^1$, we have $s \sim t$ implies $u *^1 s *^1 v \sim u *^1 t *^1 v$. We will denote by $(S, *)/\!\sim = (S/\!\sim, */\!\sim)$ the *quotient semigroup of* $(S, *)$ *by* $\sim$ where $*/\!\sim$ is the internal composition law on $S/\!\sim$ defined by $[s]_\sim \,*/\!\sim\, [t]_\sim = [s * t]_\sim$ for all $[s]_\sim, [t]_\sim \in S/\!\sim$.

We define *congruence on monoids* and *quotient monoid of another monoid by a congruence* in the same way, by replacing "semigroup" with "monoid".

The most fundamental transformation in algebra is that of a (homo)morphism, defined below in the case of semigroups and monoids.

Definition 3.1.10. Let $(S, *)$ and $(T, \perp)$ be two semigroups. A mapping $\varphi\colon S \to T$ is a *semigroup morphism from* $(S, *)$ *to* $(T, \perp)$ if and only if for all $s_1, s_2 \in S$, $\varphi(s_1) \perp \varphi(s_2) =$

$\varphi(s_1 * s_2)$. The semigroup morphism φ is called *injective*, *surjective* and *bijective* if and only if the associated mapping has these properties. Moreover, we call φ a:

- *semigroup isomorphism* if and only if it is bijective (in that case, $(S, *)$ and $(T, \perp)$ are called *isomorphic*);
- *semigroup endomorphism* if and only if $(S, *) = (T, \perp)$;
- *semigroup automorphism* if and only if φ is both a semigroup isomorphism and endomorphism.

Definition 3.1.11. Let $(M, *)$ and $(N, \perp)$ be two monoids. A mapping $\varphi\colon M \to N$ is a *monoid morphism from* $(M, *)$ *to* $(N, \perp)$ if and only if for all $m_1, m_2 \in M$, $\varphi(m_1) \perp \varphi(m_2) = \varphi(m_1 * m_2)$, and $\varphi(1_{(M,*)}) = 1_{(N,\perp)}$. We use the terms *injective*, *surjective*, *bijective*, *isomorphism* (*isomorphic*), *endomorphism* and *automorphism* for monoid morphisms as for the case of semigroup morphisms.

For example, for all $d \in \mathbb{N}, d \geq 2$, the mapping

$$\begin{aligned} \varphi\colon \mathbb{N} &\to \mathbb{Z}/d\,\mathbb{Z} \\ n &\mapsto n \mod d \end{aligned}$$

is a monoid morphism from $(\mathbb{N}, +)$ to $(\mathbb{Z}/d\,\mathbb{Z}, +)$. However, the mapping

$$\begin{aligned} \psi\colon \mathbb{Z} &\to \mathbb{N} \\ n &\mapsto |n| \end{aligned}$$

is not a monoid morphism from $(\mathbb{Z}, +)$ to $(\mathbb{N}, +)$ because $\psi(3) + \psi(-2) = 5 \neq 1 = \psi(3 + (-2))$.

Another fundamental notion is that of a subsemigroup (submonoid) of a semigroup (monoid).

Definition 3.1.12. Let $(S, *)$ and $(T, \perp)$ be two semigroups. If $T \subseteq S$ and $\perp = *|_T$, we say $(T, \perp)$ *is a subsemigroup of* $(S, *)$.

Definition 3.1.13. Let $(M, *)$ and $(N, \perp)$ be two monoids. If $N \subseteq M$, $\perp = *|_N$ and $1_{(M,*)} \in N$, we say $(N, \perp)$ *is a submonoid of* $(M, *)$.

For example, $(\mathbb{N}_{>0}, +)$ is a subsemigroup of $(\mathbb{N}, +)$ and $(\mathbb{N}, +)$ is a submonoid of $(\mathbb{Z}, +)$.

For a given semigroup (monoid) $(S, *)$ and a subset E of S we call *the subsemigroup (submonoid) of* $(S, *)$ *generated by* E the inclusion-wise smallest semigroup (monoid) of

$(S, *)$ containing all of E. If that semigroup (monoid) is equal to $(S, *)$, we say $(S, *)$ is *generated by* E. For instance, $(\mathbb{N}, +)$ is generated by $\{1\}$.

We now define very common operations on semigroups and monoids.

Definition 3.1.14. Let $(S, *)$ and $(T, \perp)$ be two semigroups.

- We denote by $(S, *) \times (T, \perp) = (S \times T, * \times \perp)$ the *direct semigroup product of* $(S, *)$ *and* $(T, \perp)$, the semigroup whose internal composition law is defined as follows:

$$\begin{aligned} * \times \perp \colon (S \times T) \times (S \times T) &\to S \times T \\ \big((s_1, t_1), (s_2, t_2)\big) &\mapsto (s_1 * s_2, t_1 \perp t_2) \ . \end{aligned}$$

- If there exists a surjective semigroup morphism from $(S, *)$ to $(T, \perp)$, we say $(T, \perp)$ *is a semigroup quotient of* $(S, *)$.

- If there exists a subsemigroup $(\tilde{S}, \tilde{*})$ of $(S, *)$ such that $(T, \perp)$ is a semigroup quotient of $(\tilde{S}, \tilde{*})$, we say $(T, \perp)$ *semigroup divides* $(S, *)$.

We define *direct product of monoids*, *quotient of monoids* and *monoid division* in the same way, by replacing "semigroup" with "monoid".

3.1.2 Varieties

We very often will consider classes of finite semigroups or monoids that share some properties and are "nice" in the sense that they are stable (also said closed) under the common operations defined in the previous subsection. This leads to the notion of a variety defined below.

Definition 3.1.15. A *variety of finite semigroups* is a class $\mathbf{V}$ of finite semigroups such that:

- it contains all trivial semigroups (*trivial semigroups containment*);

- if $(S, *)$ and $(T, \perp)$ are two finite semigroups in $\mathbf{V}$, then $(S, *) \times (T, \perp)$ is also in $\mathbf{V}$ (*closure under finite direct semigroup product*);

- if $(S, *) \in \mathbf{V}$ and $(T, \perp)$ is a finite semigroup that semigroup divides $(S, *)$, then $(T, \perp) \in \mathbf{V}$ (*closure under semigroup division*).

We define a *variety of finite monoids* in the same way, by replacing "semigroup" with "monoid".

Remark 3.1.16. There is one unique variety of finite semigroups (monoids) that contains exactly all trivial semigroups (monoids), that we will call the *trivial variety of finite semigroups* (*monoids*).

For example, we will denote by $\mathbf{S}$ the variety of all finite semigroups, and by $\mathbf{M}$ the variety of all finite monoids. We also denote by $\mathbf{M}_{\text{sol}}$ the variety of all finite monoids which do only contain solvable groups.[1] Another variety we will use is the variety $\mathbf{D}$ of all definite semigroups, containing all finite semigroups $(S, *)$ verifying that there exists $k \in \mathbb{N}_{>0}$ such that $x * y_1 * \cdots * y_k = y_1 * \cdots * y_k$ for all $x, y_1, \ldots, y_k \in S$.

As we just saw for the case of $\mathbf{D}$, a variety of finite semigroups or monoids might be characterised equationally, that is to say, in an informal way, as the class of all finite semigroups or monoids satisfying certain equalities. In fact, Reiterman showed Reiterman [1982] that for a precise formal notion of (pseudo)identities and satisfaction of those by finite semigroups or monoids (to be found in, e.g., Almeida and Weil [1995], Pin [1997]), a class of finite semigroups (monoids) is a variety if and only if it can be defined by some set of (pseudo)identities, i.e. it is equal to the class of all finite semigroups (monoids) satisfying all identities in that set. Stating Reiterman's result formally requires topological notions and tools; what we will do in this thesis is only to use these equational characterisations in a "self-contained manner" (as we did to define $\mathbf{D}$), without referring to a formal notion of an identity.

We can also build new finite semigroup (monoid) varieties from any class of finite semigroups (monoids) as explained below.

Definition 3.1.17. Let $\mathcal{C}$ be a class of finite semigroups (monoids). We will denote by $\langle \mathcal{C} \rangle_{\mathbf{S}}$ ($\langle \mathcal{C} \rangle_{\mathbf{M}}$) the intersection of all varieties of finite semigroups (monoids) containing $\mathcal{C}$, which is a variety of finite semigroups (monoids), called *the variety of finite semigroups (monoids) generated by* $\mathcal{C}$.

We now define the notion of the wreath product between two semigroups, that we will encounter subsequently.

Definition 3.1.18. Let $(S, *)$ and $(T, \perp)$ be two semigroups. The *wreath product of* $(S, *)$ *and* $(T, \perp)$, denoted by $(S, *) \wr (T, \perp)$, is the semigroup $(S^{T^1} \times T, \diamond)$ where the internal composition law $\diamond$ is such that for all $(f, t), (f', t') \in S^{T^1} \times T$, $(f, t) \diamond (f', t') = (g, t \perp t')$ where $g \in S^{T^1}$ is defined by $g(x) = f(x) * f'(x \perp t)$ for all $x \in T^1$.

[1] The notion of a solvable group will only be mentioned once more in this thesis, further in this chapter. The definition can be found in standard group theory books.

A fundamental result in finite semigroup theory, showing the importance of this notion, is the Krohn-Rhodes decomposition theorem Krohn and Rhodes [1965], stating that any finite semigroup divides a (finite) wreath product of certain specific finite groups and aperiodic semigroups (a finite semigroup $(S, *)$ being aperiodic if and only if there exists $n \in \mathbb{N}_{>0}$ verifying that $s^n = s^{n+1}$ for all $s \in S$).

For $\mathbf{V}$ and $\mathbf{W}$ two varieties of finite semigroups or monoids, at least one of which is a variety of finite semigroups, the *wreath product of* $\mathbf{V}$ *and* $\mathbf{W}$, denoted by $\mathbf{V} * \mathbf{W}$, is the variety of all finite semigroups that are semigroup divisors of wreath products of the form $(S, *) \wr (T, \perp)$ with $(S, *) \in \mathbf{V}$ and $(T, \perp) \ \in \mathbf{W}$. When $\mathbf{V}$ and $\mathbf{W}$ are varieties of finite monoids, we define $\mathbf{V} * \mathbf{W}$ in the same way, but replacing "semigroup" with "monoid".

3.1.3 Idempotents

Let $(S, *)$ be a finite semigroup (monoid). An element $e \in S$ is called an *idempotent of* S if and only if $e * e = e$. It is a classical basic result that there is a positive number (the minimum such number), *the idempotent power of* $(S, *)$, often denoted ω, such that for any element $s \in S$, s^ω is idempotent.

Idempotents actually play a very important role when it comes to studying the structural properties of finite semigroups, as we will see in the next subsection. Here we introduce one notion that crucially involves idempotents.

For $\mathbf{V}$ a variety of finite monoids, we say that a finite semigroup $(S, *)$ is *locally* $\mathbf{V}$ if, for every idempotent e of $(S, *)$, the monoid $(eSe, *|_{eSe})$ belongs to $\mathbf{V}$; we denote by $\mathbf{LV}$ the class of locally-$\mathbf{V}$ finite semigroups, which happens to be a variety of finite semigroups. Equivalently, $\mathbf{LV}$ is the variety of all finite semigroups whose only finite monoids semigroup-dividing them belong to $\mathbf{V}$. Yet equivalently, $\mathbf{LV}$ is the inclusion-wise maximal variety of finite semigroups that contains all finite monoids of $\mathbf{V}$ and only those. A variety $\mathbf{V}$ is said to be *local* if $\mathbf{V} * \mathbf{D} = \mathbf{LV}$. This is not the usual definition of locality, defined using categories, but it is equivalent to it [Tilson, 1987, Theorem 17.3].

3.1.4 Green's relations

An essential tool to study the structure of semigroups (and monoids) is that of the so-called Green's preorder and equivalence relations.

Definition 3.1.19. Let $(S, *)$ be a semigroup. We define *Green's preorder relations* $\leq_\mathfrak{J}$, $\leq_\mathfrak{R}$, $\leq_\mathfrak{L}$ *and* $\leq_\mathfrak{H}$ such that for all $s, t \in S$

- $s \leq_\mathfrak{J} t$ if and only if there exist $x, y \in S^1$ such that $s = x *^1 t *^1 y$;

- $s \leq_{\mathfrak{R}} t$ if and only if there exists $x \in S^1$ such that $s = t *^1 x$;
- $s \leq_{\mathfrak{L}} t$ if and only if there exists $x \in S^1$ such that $s = x *^1 t$;
- $s \leq_{\mathfrak{H}} t$ if and only if $s \leq_{\mathfrak{R}} t$ and $s \leq_{\mathfrak{L}} t$.

The equivalence relations associated with these preoders are called *Green's equivalence relations* $\mathfrak{J}$, $\mathfrak{R}$, $\mathfrak{L}$ *and* $\mathfrak{H}$ respectively, and are such that for all $s, t \in S$

- $s \mathfrak{J} t$ if and only if $s \leq_{\mathfrak{J}} t$ and $t \leq_{\mathfrak{J}} s$;
- $s \mathfrak{R} t$ if and only if $s \leq_{\mathfrak{R}} t$ and $t \leq_{\mathfrak{R}} s$;
- $s \mathfrak{L} t$ if and only if $s \leq_{\mathfrak{L}} t$ and $t \leq_{\mathfrak{L}} s$;
- $s \mathfrak{H} t$ if and only if $s \leq_{\mathfrak{H}} t$ and $t \leq_{\mathfrak{H}} s$.

For any semigroup $(S, *)$, for all $s, t \in S$, we shall write

- $s <_{\mathfrak{J}} t$ if and only if $s \leq_{\mathfrak{J}} t$ and $s \not\mathfrak{J} t$;
- $s <_{\mathfrak{R}} t$ if and only if $s \leq_{\mathfrak{R}} t$ and $s \not\mathfrak{R} t$;
- $s <_{\mathfrak{L}} t$ if and only if $s \leq_{\mathfrak{L}} t$ and $s \not\mathfrak{L} t$;
- $s <_{\mathfrak{H}} t$ if and only if $s \leq_{\mathfrak{H}} t$ and $s \not\mathfrak{H} t$.

For any semigroup $(S, *)$, we also define a fifth Green's equivalence relation $\mathfrak{D}$ that is such that for all $s, t \in S$, $s \mathfrak{D} t$ if and only if there exists $u \in S$ such that $s \mathfrak{R} u$ and $u \mathfrak{L} t$ if and only if there exists $v \in S$ such that $s \mathfrak{L} v$ and $v \mathfrak{R} t$. In fact, $\mathfrak{D}$ is equal to $\mathfrak{J}$ when $(S, *)$ is finite, but it isn't the case in general.

The following is a well-known fact (see [Pin, 1986, Chapter 3, Proposition 1.4]).

Lemma 3.1.20. *For all elements u and v of M, if $u \leq_{\mathfrak{R}} v$ and $u \mathfrak{J} v$, then $u \mathfrak{R} v$. Similarly, if $u \leq_{\mathfrak{L}} v$ and $u \mathfrak{J} v$, then $u \mathfrak{L} v$.*

Green's relations are important, among other reasons, because they are instrumental in characterising many fundamental varieties of finite monoids or semigroups.

Let $(S, *)$ be a semigroup and $\mathcal{E}$ a relation on S. For each $s \in S$ we will denote by $\mathcal{E}(s) = \{t \in S \mid s \mathcal{E} t\}$ the set of elements of S that are $\mathcal{E}$-equivalent to s (the $\mathcal{E}$-equivalence class containing s). $(S, *)$ is said to be *$\mathcal{E}$-trivial* if and only if all $\mathcal{E}$-classes in S (i.e. $\mathcal{E}$-equivalence classes in S) contain a sole element, i.e. $\mathcal{E}(s) = \{s\}$ for all $s \in S$.

The classes of all finite $\mathfrak{J}$-trivial, $\mathfrak{R}$-trivial, $\mathfrak{L}$-trivial and $\mathfrak{H}$-trivial monoids each form varieties of finite monoids denoted respectively by $\mathbf{J}$, $\mathbf{R}$, $\mathbf{L}$ and $\mathbf{A}$. In fact, we can show that a finite monoid $(M, *)$ is *aperiodic*, i.e. such that there exists $n \in \mathbb{N}_{>0}$ verifying that $m^n = m^{n+1}$ for all $m \in M$, if and only if it is $\mathfrak{H}$-trivial. This is why we call the variety of finite $\mathfrak{H}$-trivial monoids the variety of finite aperiodic monoids and denote it by $\mathbf{A}$. The following holds: $\mathbf{J} \subset \mathbf{R} \subset \mathbf{A}$, $\mathbf{J} \subset \mathbf{L} \subset \mathbf{A}$ and $\mathbf{R} \neq \mathbf{L}$.

Concerning equational characterisations, we have that for any finite monoid $(M, *)$ of idempotent power ω,

- $(M, *) \in \mathbf{A}$ if and only if $x^\omega = x^{\omega+1}$ for all $x \in M$;
- $(M, *) \in \mathbf{L}$ if and only if $(xy)^\omega = y(xy)^\omega$ for all $x, y \in M$;
- $(M, *) \in \mathbf{R}$ if and only if $(xy)^\omega = (xy)^\omega x$ for all $x, y \in M$;
- $(M, *) \in \mathbf{J}$ if and only if $(xy)^\omega = (xy)^\omega x = y(xy)^\omega$ for all $x, y \in M$.

Those are well-studied natural varieties of finite monoids. Another well-studied variety of finite monoids is $\mathbf{DA}$, the variety of all finite monoids such that each of their $\mathfrak{D}$-classes containing at least an idempotent forms an aperiodic subsemigroup of the monoid (i.e. all the $\mathfrak{H}$-classes contained in this $\mathfrak{D}$-class are trivial and contain an idempotent as unique element). The following holds: $\mathbf{J} \subset \mathbf{R} \subset \mathbf{DA} \subset \mathbf{A}$, $\mathbf{J} \subset \mathbf{L} \subset \mathbf{DA} \subset \mathbf{A}$. Equationally, we have that any finite monoid $(M, *)$ of idempotent power ω belongs to $\mathbf{DA}$ if and only if $(xy)^\omega = (xy)^\omega x (xy)^\omega$ for all $x, y \in M$.

3.1.5 Some first links with formal language theory

The set of all words on some alphabet coupled with the concatenation operation forms a monoid, as defined below.

Definition 3.1.21. Let Σ be an alphabet. The *free monoid generated by* Σ *(or on* Σ*)* is the monoid $(\Sigma^*, \cdot)$. The *free semigroup generated by* Σ *(or on* Σ*)* is the semigroup $(\Sigma^+, \cdot)$.

It is easy to see that for any alphabet Σ and any monoid $(M, *)$, any monoid morphism $\varphi \colon \Sigma^* \to M$ from $(\Sigma^*, \cdot)$ to $(M, *)$ is uniquely determined by the images $\varphi(a)$ of all letters $a \in \Sigma$ (and the same holds when replacing $(\Sigma^*, \cdot)$ with $(\Sigma^+, \cdot)$ and "monoid" with "semigroup").

We call a *stamp* a surjective monoid morphism $\varphi \colon \Sigma^* \to M$ from the free monoid $(\Sigma^*, \cdot)$ generated by some alphabet Σ to a finite monoid $(M, *)$. When $(M, *)$ is trivial,

we also call φ *trivial*. We will sometimes encounter the important type of stamps defined next.

Definition 3.1.22. Let $(S, *)$ be a finite semigroup (monoid). The *evaluation morphism of* $(S, *)$ is the unique monoid morphism $\eta_{(S,*)} \colon S^* \to S^1$ from $(S^*, \cdot)$ to $(S, *)^1$ such that $\eta_{(S,*)}(s) = s$ for all $s \in S$.

3.1.6 $\mathcal{C}$-varieties of stamps

We shall see that when studying the link between finite automata theory and computational complexity theory, it will actually be necessary to use a more general notion of varieties whose elements are stamps rather than finite semigroups or monoids. The exposition of this notion is based on Straubing [2002], Pin and Straubing [2005] and Chaubard et al. [2006].

Let Σ and Γ be two alphabets, and let $\varphi \colon \Sigma^* \to \Gamma^*$ be a monoid morphism from $(\Sigma^*, \cdot)$ to $(\Gamma^*, \cdot)$. We say that φ is

- *length-preserving (an* lp*-morphism)* if and only if the image by φ of any letter in Σ is a letter of Γ, i.e. $\varphi(\Sigma) \subseteq \Gamma$;
- *non-erasing (an* ne*-morphism)* if and only if the image by φ of any letter in Σ is a non-empty word on Γ, i.e. $\varphi(\Sigma) \subseteq \Gamma^+$;
- *length-multiplying (an* lm*-morphism)* if and only if there exists $k \in \mathbb{N}$ such that the image by φ of any letter in Σ is a word on Γ of length k, i.e. $\varphi(\Sigma) \subseteq \Gamma^k$.

We consider a class $\mathcal{C}$ of monoid morphisms between two free monoids generated by alphabets that satisfies the following properties:

- if $f \colon \Sigma^* \to \Gamma^*$ from $(\Sigma^*, \cdot)$ to $(\Gamma^*, \cdot)$ and $g \colon \Gamma^* \to \mathrm{T}^*$ from $(\Gamma^*, \cdot)$ to $(\mathrm{T}^*, \cdot)$ where Σ, Γ and T are alphabets both are monoid morphisms in $\mathcal{C}$, then the monoid morphism $g \circ f \colon \Sigma^* \to \mathrm{T}^*$ does also belong to $\mathcal{C}$ (*closure under composition*);
- $\mathcal{C}$ contains all *lp*-morphisms;
- if $f \colon \Sigma^* \to \Gamma^*$ from $(\Sigma^*, \cdot)$ to $(\Gamma^*, \cdot)$ where Σ and Γ are alphabets is a monoid morphism in $\mathcal{C}$ and $g \colon \mathrm{T}^* \to \Lambda^*$ from $(\mathrm{T}^*, \cdot)$ to $(\Lambda^*, \cdot)$ where T and Λ are alphabets is a monoid morphism verifying $\{|g(b)| \mid b \in \mathrm{T}\} = \{|f(a)| \mid a \in \Sigma\}$, then g does also belong to $\mathcal{C}$.

For example, the classes of all *lp*-morphisms, all *ne*-morphisms, all *lm*-morphisms and simply all morphisms (*all*-morphisms) all verify these properties. When $\mathcal{C}$ is equal to one of these classes, we shall replace $\mathcal{C}$ by, respectively, *lp*, *ne*, *lm* or *all* in the vocabulary prefixed by $\mathcal{C}$ and in symbols using $\mathcal{C}$ in the following.

By defining the appropriate notions of products of stamps and $\mathcal{C}$-division of stamps, a notion of $\mathcal{C}$-varieties of stamps can be defined.

Definition 3.1.23. Let $\varphi\colon \Sigma^* \to M$ and $\psi\colon \Gamma^* \to N$ be two stamps such that Σ and Γ are alphabets and $(M, *)$ and $(N, \perp)$ are, respectively, the associated finite monoids.

- If $\Sigma = \Gamma$, we denote by $\varphi \times \psi$ the *product of φ and ψ*, the stamp $\eta\colon \Sigma^* \to M'$ such that $(M', *')$ is the submonoid of $(M, *)\times(N, \perp)$ generated by $\{(\varphi(a), \psi(a)) \mid a \in \Sigma\}$ and $\eta(a) = (\varphi(a), \psi(a))$ for all $a \in \Sigma$.

- If there exist a monoid morphism $f\colon \Sigma^* \to \Gamma^*$ in $\mathcal{C}$ from $(\Sigma^*, \cdot)$ to $(\Gamma^*, \cdot)$ and a surjective monoid morphism $\alpha\colon \mathfrak{Im}(\psi \circ f) \to M$ from $(\mathfrak{Im}(\psi \circ f), \perp|_{\mathfrak{Im}(\psi\circ f)})$ to $(M, *)$ verifying $\varphi = \alpha \circ \psi \circ f$, we say *$\varphi$ $\mathcal{C}$-divides ψ*.

Definition 3.1.24. A *$\mathcal{C}$-variety of stamps* is a class $\mathbf{V}$ of stamps such that:

- it contains all trivial stamps (*trivial stamps containment*);

- if $\varphi\colon \Sigma^* \to M$ and $\psi\colon \Sigma^* \to N$ are two stamps in $\mathbf{V}$, then $\varphi\times\psi$ is also in $\mathbf{V}$ (*closure under finite stamp product*);

- if $\varphi\colon \Sigma^* \to M$ is a stamp in $\mathbf{V}$ and $\psi\colon \Gamma^* \to N$ is a stamp that $\mathcal{C}$-divides φ, then $\psi \in \mathbf{V}$ (*stability under $\mathcal{C}$-division*).

As for varieties of finite semigroups or monoids, we can also build up new $\mathcal{C}$-varieties of stamps from any class of stamps, as well as, actually, from any class of finite semigroups or monoids.

Definition 3.1.25. Let $\mathcal{D}$ be a class of stamps. We will denote by $\langle\mathcal{D}\rangle_{\mathcal{C}}$ the intersection of all $\mathcal{C}$-varieties of stamps containing $\mathcal{D}$, which is a $\mathcal{C}$-variety of stamps, called *the $\mathcal{C}$-variety of stamps generated by $\mathcal{D}$*. When $\mathcal{D}$ is a class of finite semigroups (monoids), we shall denote by $\langle\mathcal{D}\rangle_{\mathcal{C}}$ the $\mathcal{C}$-variety of stamps generated by the class of all stamps $\varphi\colon \Sigma^* \to M$ from a free monoid $(\Sigma^*, \cdot)$ to a finite monoid $(M, *)$ such that the finite semigroup $(\varphi(\Sigma^+), *|_{\varphi(\Sigma^+)})$ belongs to $\mathcal{D}$ ($(M, *)$ belongs to $\mathcal{D}$).

Observe that to each variety of finite semigroups (monoids) $\mathbf{V}$, we can associate the *ne*-variety (*all*-variety) of all stamps $\varphi\colon \Sigma^* \to M$ from a free monoid $(\Sigma^*, \cdot)$ to a finite monoid $(M, *)$ such that the finite semigroup $(\varphi(\Sigma^+), *|_{\varphi(\Sigma^+)})$ belongs to $\mathbf{V}$ ($(M, *)$ belongs to $\mathbf{V}$), that is exactly $\langle \mathbf{V} \rangle_{ne}$ ($\langle \mathbf{V} \rangle_{all}$). This gives a bijective correspondence between varieties of finite semigroups (monoids) and *ne*-varieties (*all*-varieties) of stamps.

Any *all*- or *ne*-variety of stamps is also an *lm*-variety of stamps. (This is because when a stamp φ *lm*-divides a stamp ψ, we have that $\varphi = \alpha \circ \psi \circ f$ where α is an adequate monoid morphism and f an adequate *lm*-morphism that is necessarily also an *all*-morphism and an *ne*-morphism, except when f sends everything to the empty word, in which case φ must be a trivial stamp.) A concrete example of an *lm*-variety of stamps (that is also an *lp*-variety of stamps) is the class **MOD** of all stamps $\varphi\colon \Sigma^* \to G$ from a free monoid $(\Sigma^*, \cdot)$ to a finite cyclic group $(G, *)$ such that $\varphi(a) = \varphi(b)$ for all $a, b \in \Sigma$ (a *cyclic group* being just a group generated by only one of its elements).

We end this subsection as well as this section by defining the wreath product of two stamps.

Definition 3.1.26. Let $\varphi\colon (\Sigma \times N)^* \to M$ and $\psi\colon \Sigma^* \to N$ be two stamps such that Σ is an alphabet and $(M, *)$ and $(N, \perp)$ both are finite monoids. The *wreath product of* φ *and* ψ, denoted by $\varphi \wr \psi$, is the stamp $\mu\colon \Sigma^* \to K$ from $(\Sigma^*, \cdot)$ to the finite monoid $(K, \diamond)$ such that if for each $a \in \Sigma$, $f_a \in M^N$ is the function defined by $f_a(x) = \varphi(a, x)$ for all $x \in N$, we have that $(K, \diamond)$ is the submonoid of $(M, *) \wr (N, \perp)$ generated by $\{(f_a, \psi(a)) \mid a \in \Sigma\}$ and μ is the unique monoid morphism from $(\Sigma^*, \cdot)$ to $(K, \diamond)$ verifying that $\mu(a) = (f_a, \psi(a))$ for all $a \in \Sigma$.

For $\mathbf{V}$ and $\mathbf{W}$ two $\mathcal{C}$-varieties of stamps, the *wreath product of* $\mathbf{V}$ *and* $\mathbf{W}$, denoted by $\mathbf{V} * \mathbf{W}$, is the $\mathcal{C}$-variety of all stamps that are $\mathcal{C}$-divisors of wreath products of the form $\varphi \wr \psi$ with φ a stamp of $\mathbf{V}$ and ψ a stamp of $\mathbf{W}$.

3.2 Recognition by morphisms

In fact, there is a very strong connection between finite semigroups (monoids) and finite automata, the latter being known for defining exactly the class of regular languages by Kleene's seminal theorem. The algebraic theory of automata, that studies this connection, has been a very active and fruitful research domain for over 60 years. Here, we present some of the central results needed in the work presented in this thesis. As for the algebra of finite semigroups and monoids, we refer the reader to Eilenberg [1974, 1976], Pin [1986], as well as Pin [2016] for a more thorough introduction to the theory of regular languages,

finite automata and the links with the algebra of finite semigroups and monoids. For the case of $\mathcal{C}$-varieties of languages, we refer the reader to Straubing [2002], Pin and Straubing [2005] and Chaubard et al. [2006].

3.2.1 Regular languages and morphisms

Monoids can be used to compute, in the sense that they can be used to recognise languages through the use of morphisms, as formalised below.

Definition 3.2.1. Let Σ be an alphabet and $(M, *)$ a monoid. A language $L \subseteq \Sigma^*$ over Σ is *recognised by a monoid morphism* $\varphi\colon \Sigma^* \to M$ *from* $(\Sigma^*, \cdot)$ *to* $(M, *)$ if and only if there exists $P \subseteq M$ such that $L = \varphi^{-1}(P)$. We will say that $(M, *)$ *recognises* L if and only if such a monoid morphism exists.

Example 3.2.2. The language a^* over $\{a, b\}$ is recognised by the unique monoid morphism from $(\{a, b\}^*, \cdot)$ to $(\mathbb{Z}/2\,\mathbb{Z}, \times)$ verifying $\varphi(a) = 1$ and $\varphi(b) = 0$, because $a^* = \varphi^{-1}(1)$.

Example 3.2.3. The language $a^*(ba^*ba^*)^*$ over $\{a, b\}$ is recognised by the unique monoid morphism from $(\{a, b\}^*, \cdot)$ to $(\mathbb{Z}/2\,\mathbb{Z}, +)$ verifying $\varphi(a) = 0$ and $\varphi(b) = 1$, because $a^*(ba^*ba^*)^* = \varphi^{-1}(0)$.

Example 3.2.4. The language $\left\{w \in \{a, b\}^* \;\middle|\; |w|_a \neq |w|_b\right\}$ over $\{a, b\}$ is recognised by the unique monoid morphism from $(\{a, b\}^*, \cdot)$ to $(\mathbb{Z}, +)$ verifying $\varphi(a) = 1$ and $\varphi(b) = -1$, because $\left\{w \in \{a, b\}^* \;\middle|\; |w|_a \neq |w|_b\right\} = \varphi^{-1}(\mathbb{Z} \setminus \{0\})$.

One of the fundamental objects that comes into play when we talk about morphism-recognition of some given language is the syntactic monoid, defined using the syntactic congruence on the language's alphabet, and to which we associate the syntactic morphism.

Definition 3.2.5. Let Σ be alphabet and $L \subseteq \Sigma^*$ a language over Σ. The *syntactic congruence of* L is the congruence $\sim_L$ on $(\Sigma^*, \cdot)$ such that for all $u, v \in \Sigma^*$, we have $u \sim_L v$ if and only if, for all $x, y \in \Sigma^*$, $xuy \in L \Leftrightarrow xvy \in L$. The *syntactic monoid of* L is $\mathrm{M}(L) = (\Sigma^*, \cdot)/{\sim_L}$, the quotient monoid of $(\Sigma^*, \cdot)$ by the syntactic congruence $\sim_L$. Finally, the *syntactic morphism of* L is the unique monoid morphism $\eta_L\colon \Sigma^* \to \Sigma^*/{\sim_L}$ from $(\Sigma^*, \cdot)$ to $\mathrm{M}(L)$ such that $\eta_L(a) = [a]_{\sim_L}$ for all $a \in \Sigma$.

The reason why the syntactic monoid of a language is a fundamental object is because of the following property stating that, in fact, it is the smallest monoid, with respect to division, that recognises that language.

Proposition 3.2.6. *Let $L \subseteq \Sigma^*$ be a language over some alphabet Σ, $\mathrm{M}(L) = (\Sigma^*, \cdot)/\sim_L$ its syntactic monoid and $\eta_L \colon \Sigma^* \to \Sigma^*/\sim_L$ its syntactic morphism, where $\sim_L$ denotes the syntactic congruence of L. Then, a monoid $(M, *)$ recognises L if and only if $\mathrm{M}(L)$ divides $(M, *)$. Further, a stamp $\varphi \colon \Sigma^* \to M$ from $(\Sigma^*, \cdot)$ to a finite monoid $(M, *)$ recognises L if and only if η_L lp-divides φ.*

Let us now recall the definition of the regular languages.

Definition 3.2.7. Let Σ be an alphabet. The *set of regular languages over* Σ, denoted by $\mathcal{R}\mathsf{eg}(\Sigma^*)$, is the smallest set of languages $\mathcal{F}$ over Σ satisfying the following conditions:

1. $\mathcal{F}$ contains the languages $\emptyset$ and $\{a\}$ for each letter $a \in \Sigma$;
2. $\mathcal{F}$ is closed under finite union, i.e. if $L_1, L_2 \in \mathcal{F}$, then $L_1 \cup L_2 \in \mathcal{F}$;
3. $\mathcal{F}$ is closed under finite concatenation, i.e. if $L_1, L_2 \in \mathcal{F}$, then $L_1 \cdot L_2 \in \mathcal{F}$;
4. $\mathcal{F}$ is closed under star, i.e. if $L \in \mathcal{F}$, then $L^* \in \mathcal{F}$.

We will denote by $\mathcal{R}\mathsf{eg}$ the class of all regular languages, the union over all alphabets Σ of $\mathcal{R}\mathsf{eg}(\Sigma^*)$.

It is well known that the class of regular languages is exactly the class of languages decided by deterministic finite automata (DFA). The fundamental theorem of algebraic automata theory, at the very basis of it, is the following one. It states that regularity of a language is equivalent to finiteness of its syntactic monoid, or, equivalently, to the fact of being recognised by some finite monoid.

Theorem 3.2.8 (See [Eilenberg, 1974, Chapter III, Proposition 10.1 and Chapter VII, Theorem 5.1] and [Eilenberg, 1976, Chapter VII, Section 6], as well as [Pin, 1986, Chapter 1, Proposition 2.1, Kleene's theorem and Proposition 2.7])**.** *Let $L \subseteq \Sigma^*$ be a language over some alphabet Σ. The following statements are equivalent.*

1. *L is regular.*
2. *L is recognised by a finite monoid.*
3. *L's syntactic congruence has finite index, which is equivalent to saying that its syntactic monoid is finite.*
4. *L is decided by a deterministic finite automaton (DFA).*

3.2.2 Varieties of languages

As for finite semigroups and monoids, but also stamps, we very often consider classes of regular languages that share some properties, that are closed under common operations on languages. The set of operations under which those classes are closed may vary, but what we want at least for any such class is that for any language it contains, one may change the symbols of the alphabet over which this language is defined and still obtain a language belonging to the class. More formally, we say a *class of regular languages* is a correspondence $\mathcal{F}$ which associates with each alphabet Σ a set $\mathcal{F}(\Sigma^*)$ of regular languages over Σ in such a way that, if $\sigma\colon \Sigma \to \Gamma$ is a bijection from alphabet Σ into alphabet Γ, a language L belongs to $\mathcal{F}(\Sigma^*)$ if and only if $\sigma(L)$ belongs to $\mathcal{F}(\Gamma^*)$. For a given alphabet Σ, a *lattice of languages over* Σ is a set of languages over Σ containing the empty language $\emptyset$, the full language Σ^* and that is closed under finite union and finite intersection. It is moreover a *Boolean algebra of languages over* Σ if it is additionally closed under complement.

Definition 3.2.9. A *variety of languages* is a class of regular languages $\mathcal{V}$ such that for every alphabets Σ and Γ:

- $\mathcal{V}(\Sigma^*)$ is a Boolean algebra of languages over Σ (*closure under Boolean operations and trivial languages containment*);
- for every monoid morphism $\varphi\colon \Sigma^* \to \Gamma^*$ from $(\Sigma^*, \cdot)$ to $(\Gamma^*, \cdot)$, $L \in \mathcal{V}(\Gamma^*)$ implies $\varphi^{-1}(L) \in \mathcal{V}(\Sigma^*)$ (*closure under inverses of monoid morphisms*);
- if $L \in \mathcal{V}(\Sigma^*)$ and $u \in \Sigma^*$, $u^{-1}L \in \mathcal{V}(\Sigma^*)$ and $Lu^{-1} \in \mathcal{V}(\Sigma^*)$ (*closure under quotients*).

It is a classical result that the class of all regular languages $\mathcal{R}\mathsf{eg}$ is a variety of languages. Another well-known variety of languages is the class of *star-free* regular languages $\mathcal{SF}$, such that for each alphabet Σ, the set $\mathcal{SF}(\Sigma^*)$ of *star-free languages over* Σ is the smallest set of languages over Σ containing the languages $\emptyset$, $\{\varepsilon\}$ and $\{a\}$ for each letter $a \in \Sigma$ and that is closed under finite union, finite concatenation and complement.

One of the intentions behind this definition of a variety of languages is to have a bijective correspondence between varieties of languages and varieties of finite monoids, which gives an equivalent algebraic characterisation of such classes of languages. This is made formal in the following.

Let the correspondence $\mathbf{V} \to \mathcal{V}$ associate to each variety of finite monoids $\mathbf{V}$ the variety of languages $\mathcal{V}$, also denoted by $\mathcal{L}(\mathbf{V})$, such that for every alphabet Σ, $\mathcal{V}(\Sigma^*)$

is the set of all languages over Σ whose syntactic monoid belongs to $\mathbf{V}$. Conversely, let the correspondence $\mathcal{V} \to \mathbf{V}$ associate to each variety of languages $\mathcal{V}$ the variety of finite monoids $\mathbf{V}$, also denoted by $\mathbf{M}(\mathcal{V})$, generated by the syntactic monoids of the languages in $\mathcal{V}$.

Theorem 3.2.10 (Eilenberg — see [Eilenberg, 1976, Chapter VII, Theorem 3.2] and [Pin, 1986, Chapter 2, Theorem 2.5 and Theorem 2.7])**.** *The correspondences $\mathbf{V} \to \mathcal{V}$ and $\mathcal{V} \to \mathbf{V}$ define mutually bijective correspondences between varieties of finite monoids and varieties of languages.*

By virtue of Theorem 3.2.8, it is straightforward to see that $\mathcal{R}\mathsf{eg}$ is associated to $\mathbf{M}$, the variety of all finite monoids, through this correspondence. Concerning star-free languages, one of the seminal results of algebraic automata theory is Schützenberger's theorem Schützenberger [1965], stating that the variety of finite monoids associated to $\mathcal{SF}$ is exactly $\mathbf{A}$, the variety of finite aperiodic monoids.

Let $\mathcal{C}$ be a class of monoid morphisms between free monoids generated by alphabets satisfying the properties as required in Subsection 3.1.6. To obtain the same bijective correspondence for $\mathcal{C}$-varieties of stamps, we need to change the definition of a variety of languages by restricting the class of monoid morphisms under inverses of which it is closed to $\mathcal{C}$-morphisms.

Definition 3.2.11. A *$\mathcal{C}$-variety of languages* is a class of regular languages $\mathcal{V}$ such that for every alphabets Σ and Γ:

- $\mathcal{V}(\Sigma^*)$ is a Boolean algebra of languages over Σ (*closure under Boolean operations and trivial languages containment*);
- for every morphism $\varphi\colon \Sigma^* \to \Gamma^*$ in $\mathcal{C}$ from $(\Sigma^*, \cdot)$ to $(\Gamma^*, \cdot)$, $L \in \mathcal{V}(\Gamma^*)$ implies $\varphi^{-1}(L) \in \mathcal{V}(\Sigma^*)$ (*closure under inverses of $\mathcal{C}$-morphisms*);
- if $L \in \mathcal{V}(\Sigma^*)$ and $u \in \Sigma^*$, $u^{-1}L \in \mathcal{V}(\Sigma^*)$ and $Lu^{-1} \in \mathcal{V}(\Sigma^*)$ (*closure under quotients*).

Let the correspondence $\mathbf{V} \to \mathcal{V}$ associate to each $\mathcal{C}$-variety of stamps $\mathbf{V}$ the variety of languages $\mathcal{V}$, also denoted by $\mathcal{L}(\mathbf{V})$, such that for every alphabet Σ, $\mathcal{V}(\Sigma^*)$ is the set of all languages over Σ whose syntactic morphism belongs to $\mathbf{V}$. Conversely, let the correspondence $\mathcal{V} \to \mathbf{V}$ associate to each variety of languages $\mathcal{V}$ the $\mathcal{C}$-variety of stamps $\mathbf{V}$ generated by the syntactic morphisms of the languages in $\mathcal{V}$.

Theorem 3.2.12 (See [Straubing, 2002, Theorem 1 and Theorem 2] and [Chaubard et al., 2006, Subsection 1.3]). *The correspondences* $\mathbf{V} \to \mathcal{V}$ *and* $\mathcal{V} \to \mathbf{V}$ *define mutually bijective correspondences between* $\mathcal{C}$*-varieties of stamps and* $\mathcal{C}$*-varieties of languages.*

Observe that, for any variety of finite monoids $\mathbf{V}$, we have $\mathcal{L}(\mathbf{V}) = \mathcal{L}(\langle \mathbf{V} \rangle_{all})$.

Eilenberg's theorem, Theorem 3.2.10, establishes a one-to-one correspondence between varieties of finite monoids and varieties of languages, and Theorem 3.2.12 does it between $\mathcal{C}$-varieties of stamps and $\mathcal{C}$-varieties of languages. But what about varieties of finite semigroups? In fact, Eilenberg [Eilenberg, 1976, Chapter VII, Theorem 3.2s] did give a one-to-one correspondence between varieties of finite semigroups and so-called +-varieties of languages, however we won't present it in this thesis. Indeed, those +-varieties of languages have the annoying property of dealing with languages as subsets of Σ^+, and not Σ^*, for any alphabet Σ. Nevertheless, we can afford not presenting that correspondence in our case, since, as we have seen, to each variety of finite semigroups $\mathbf{V}$ we can bijectively associate the *ne*-variety $\langle \mathbf{V} \rangle_{ne}$ of all stamps $\varphi\colon \Sigma^* \to M$ from a free monoid $(\Sigma^*, \cdot)$ to a finite monoid $(M, *)$ such that the finite semigroup $(\varphi(\Sigma^+), *|_{\varphi(\Sigma^+)})$ belongs to $\mathbf{V}$. We thus define $\mathcal{L}(\mathbf{V})$ as being $\mathcal{L}(\langle \mathbf{V} \rangle_{ne})$ for any variety of finite semigroups $\mathbf{V}$.

3.2.3 Wreath product principle

The so-called "wreath product principle", originally due to Straubing Straubing [1979], allows to characterise the languages belonging to $\mathcal{L}(\mathbf{V} * \mathbf{W})$ for $\mathbf{V}$ and $\mathbf{W}$ two varieties of finite semigroups or monoids. We shall state the results and prove them in the more general framework of $\mathcal{C}$-varieties of stamps, inspired by Chaubard et al. [2006].[2]

Let $\psi\colon \Sigma^* \to N$ be some stamp from a free monoid $(\Sigma^*, \cdot)$ to a finite monoid $(N, \bot)$. We define the function $\sigma_\psi\colon \Sigma^* \to (\Sigma \times N)^*$ inductively as follows: for all $w \in \Sigma^*$,

$$\sigma_\psi(w) = \begin{cases} \varepsilon & \text{if } w = \varepsilon \\ \sigma_\psi(w')(a, \psi(w')) & \text{otherwise, } w = w'a \text{ for } w' \in \Sigma^* \text{ and } a \in \Sigma \ . \end{cases}$$

This means that for all $n \in \mathbb{N}_{>0}$, $w \in \Sigma^n$, we have

$$\sigma_\psi(w_1 w_2 w_3 \cdots w_n) = (w_1, 1_{(N,\bot)})(w_2, \psi(w_1))(w_3, \psi(w_1 w_2)) \cdots (w_n, \psi(w_1 w_2 \cdots w_{n-1})) \ .$$

Note that σ_ψ is what is called in the literature (see e.g. Pin [2016]) a *sequential function*,

[2] We provide full proofs in this subsection, because our statements are different from those in Chaubard et al. [2006], where this more general principle is stated, but without proofs.

realised by a *sequential transducer*.

Let $\mathcal{C}$ be a class of monoid morphisms between free monoids generated by alphabets satisfying the properties as required in Subsection 3.1.6.

We can now state the wreath product principle.

Proposition 3.2.13 (Wreath product principle)**.** *Let $\varphi\colon (\Sigma \times N)^* \to M$ and $\psi\colon \Sigma^* \to N$ be two stamps such that Σ is an alphabet and $(M, *)$ and $(N, \bot)$ both are finite monoids. Then, there exists a stamp $\eta\colon (\Sigma \times N)^* \to M'$ in $\langle\{\varphi\}\rangle_{\mathcal{C}}$ from $((\Sigma \times N)^*, \cdot)$ to a finite monoid $(M', *')$ such that each language over Σ recognised by the wreath product $\varphi \wr \psi$ is a finite union of languages of the form $W \cap \sigma_\psi^{-1}(V)$, where $W \subseteq \Sigma^*$ is recognised by ψ and $V \subseteq (\Sigma \times N)^*$ is recognised by η.*

Proof. Let $\mu = \varphi \wr \psi$. Then μ is the stamp $\mu\colon \Sigma^* \to K$ from $(\Sigma^*, \cdot)$ to the finite monoid $(K, \diamond)$ such that if for each $a \in \Sigma$, $f_a \in M^N$ is the function defined by $f_a(x) = \varphi(a, x)$ for all $x \in N$, we have that $(K, \diamond)$ is the submonoid of $(M, *) \wr (N, \bot)$ generated by $\{(f_a, \psi(a)) \mid a \in \Sigma\}$ and μ is the unique monoid morphism from $(\Sigma^*, \cdot)$ to $(K, \diamond)$ verifying that $\mu(a) = (f_a, \psi(a))$ for all $a \in \Sigma$.

Consider now the finite monoid $(M^N, \bullet)$ such that for all $f, f' \in M^N$, $f \bullet f' = g$ where $g \in M^N$ is defined by $g(x) = f(x) * f'(x)$ for all $x \in N$. For all $a \in \Sigma$ and $y \in N$, let us define $g_{a,y}$ by $g_{a,y}(x) = f_a(x \bot y)$ for all $x \in N$. We now let $(K', \bullet|_{K'})$ be the submonoid of $(M^N, \bullet)$ generated by $\{g_{a,y} \mid a \in \Sigma, y \in N\}$. Let us now define the stamp $\eta\colon (\Sigma \times N)^* \to K'$ as the unique monoid morphism from $((\Sigma \times N)^*, \cdot)$ to $(K', \bullet|_{K'})$ verifying that $\eta(a, y) = g_{a,y}$ for all $(a, y) \in \Sigma \times N$.

Let $w \in \Sigma^+$ and define $n = |w|$. By definition of μ, we have

$$\mu(w) = (f_{w_1}, \psi(w_1)) \diamond (f_{w_2}, \psi(w_2)) \diamond \cdots \diamond (f_{w_n}, \psi(w_n)) = (f, \psi(w))$$

where $f \in M^N$ verifies that for all $x \in N$,

$$\begin{aligned}
f(x) &= f_{w_1}(x) * f_{w_2}(x \bot \psi(w_1)) * f_{w_3}(x \bot \psi(w_1 w_2)) * \cdots * f_{w_n}(x \bot \psi(w_1 w_2 \cdots w_{n-1})) \\
&= \eta(w_1, 1_{(N,\bot)})(x) * \eta(w_2, \psi(w_1))(x) * \eta(w_3, \psi(w_1 w_2)) * \cdots * \\
&\quad \eta(w_n, \psi(w_1 w_2 \cdots w_{n-1}))(x) \\
&= \big(\eta(w_1, 1_{(N,\bot)}) \bullet \eta(w_2, \psi(w_1)) \bullet \eta(w_3, \psi(w_1 w_2)) \bullet \cdots \bullet \eta(w_n, \psi(w_1 w_2 \cdots w_{n-1}))\big)(x) \\
&= \eta(\sigma_\psi(w))(x) \ ,
\end{aligned}$$

so that $f = \eta(\sigma_\psi(w))$. Moreover, it holds that

$$\mu(\varepsilon) = (1_{(M^N,\bullet)}, 1_{(N,\perp)}) = \big(\eta(\varepsilon), \psi(\varepsilon)\big) = \big(\eta(\sigma_\psi(\varepsilon)), \psi(\varepsilon)\big) \ .$$

Hence, it follows that $\mu(w) = \big(\eta(\sigma_\psi(w)), \psi(w)\big)$ for all $w \in \Sigma^*$.

Consider now some $F \subseteq K$. It is straightforward to see that

$$\mu^{-1}(F) = \bigcup_{(f,x)\in F} \psi^{-1}(x) \cap \sigma_\psi^{-1}(\eta^{-1}(f))$$

where for each $(f, x) \in F$, $\psi^{-1}(x)$ is a language over Σ recognised by ψ and $\eta^{-1}(f)$ is a language over $\Sigma \times N$ recognised by η.

It now remains to show that $\eta \in \langle\{\varphi\}\rangle_{\mathcal{C}}$. For each $x \in N$, let us define $(M_x, *|_{M_x})$ as the submonoid of $(M, *)$ generated by $\{\varphi(a, x\perp y) \mid a \in \Sigma, y \in N\}$ and the stamp $\varphi_x \colon (\Sigma \times N)^* \to M_x$ as the unique monoid morphism from $(\Sigma \times N)^*$ to $(M_x, *|_{M_x})$ verifying that $\varphi_x(a, y) = \varphi(a, x\perp y) = f_a(x\perp y) = g_{a,y}(x)$ for all $(a, y) \in \Sigma \times N$. The idea is that the direct product $\prod_{x\in N}(M_x, *|_{M_x})$ is isomorphic to $(K', \bullet|_{K'})$, so that η *lp*-divides (and hence $\mathcal{C}$-divides) the product stamp $\prod_{x\in N} \varphi_x$. Moreover, it is straightforward to see that for each $x \in N$, φ_x *lp*-divides (and hence $\mathcal{C}$-divides) φ, so that $\eta \in \langle\{\varphi\}\rangle_{\mathcal{C}}$. □

Conversely, the following holds.

Proposition 3.2.14. *Let $\varphi \colon (\Sigma \times N)^* \to M$ and $\psi \colon \Sigma^* \to N$ be two stamps such that Σ is an alphabet and $(M, *)$ and $(N, \perp)$ both are finite monoids. Then, each language over Σ that is a finite union of languages of the form $W \cap \sigma_\psi^{-1}(V)$, where $W \subseteq \Sigma^*$ is recognised by ψ and $V \subseteq (\Sigma \times N)^*$ is recognised by φ, is recognised by the wreath product $\varphi \wr \psi$.*

Proof. Let $\mu = \varphi \wr \psi$. Then μ is the stamp $\mu \colon \Sigma^* \to K$ from $(\Sigma^*, \cdot)$ to the finite monoid $(K, \diamond)$ such that if for each $a \in \Sigma$, $f_a \in M^N$ is the function defined by $f_a(x) = \varphi(a, x)$ for all $x \in N$, we have that $(K, \diamond)$ is the submonoid of $(M, *) \wr (N, \perp)$ generated by $\{(f_a, \psi(a)) \mid a \in \Sigma\}$ and μ is the unique monoid morphism from $(\Sigma^*, \cdot)$ to $(K, \diamond)$ verifying that $\mu(a) = (f_a, \psi(a))$ for all $a \in \Sigma$.

Let $P_1, \ldots, P_l \subseteq M$ and $Q_1, \ldots, Q_l \subseteq N$, with $l \in \mathbb{N}_{>0}$. It is straightforward to see there exists some $F \subseteq M \times N$ verifying

$$\bigcup_{i=1}^{l} \psi^{-1}(Q_i) \cap \sigma_\psi^{-1}(\varphi^{-1}(P_i)) = \bigcup_{(m,x)\in F} \psi^{-1}(x) \cap \sigma_\psi^{-1}(\varphi^{-1}(m)) \ .$$

Let us define $F' = \{(f, x) \in M^N \times N \mid (f(1_{(N,\perp)}), x) \in F\}$.

Let $w \in \Sigma^+$ and define $n = |w|$. By definition of μ, we have

$$\mu(w) = (f_{w_1}, \psi(w_1)) \diamond (f_{w_2}, \psi(w_2)) \diamond \cdots \diamond (f_{w_n}, \psi(w_n)) = (f, \psi(w))$$

where $f \in M^N$ verifies that for all $x \in N$,

$$\begin{aligned}
&f(x)\\
=&f_{w_1}(x) * f_{w_2}(x \perp \psi(w_1)) * f_{w_3}(x \perp \psi(w_1 w_2)) * \cdots * f_{w_n}(x \perp \psi(w_1 w_2 \cdots w_{n-1}))\\
=&\varphi(w_1, x \perp 1_{(N,\perp)}) * \varphi(w_2, x \perp \psi(w_1)) * \varphi(w_3, x \ \perp \psi(w_1 w_2)) * \cdots *\\
&\varphi(w_n, x \perp \psi(w_1 w_2 \cdots w_{n-1}))\\
=&\varphi\big((w_1, x \perp 1_{(N,\perp)})(w_2, x \perp \psi(w_1))(w_3, x \ \perp \psi(w_1 w_2)) \cdots (w_n, x \perp \psi(w_1 w_2 \cdots w_{n-1}))\big) \ ,
\end{aligned}$$

so that $f(1_{(N,\perp)}) = \varphi(\sigma_\psi(w))$. Moreover, it holds that $\mu(\varepsilon) = (e, 1_{(N,\perp)}) = (e, \psi(\varepsilon))$ where $e \in M^N$ is defined by $e(x) = 1_{(M,*)}$ for all $x \in N$, so that $e(1_{(N,\perp)}) = 1_{(M,*)} = \varphi(\varepsilon) = \varphi(\sigma_\psi(w))$. Hence, it follows that for all $w \in \Sigma^*$, $\mu(w) = (f, \psi(w))$ where $f \in M^N$ verifies $f(1_{(N,\perp)}) = \varphi(\sigma_\psi(w))$.

Given this, it is straightforward to see that

$$\bigcup_{(m,x) \in F} \psi^{-1}(x) \cap \sigma_\psi^{-1}(\varphi^{-1}(m)) = \mu^{-1}(F') \ ,$$

which concludes the proof, since $\mu^{-1}(F')$ is a language recognised by μ. □

We can now derive the following theorem.

Theorem 3.2.15. *Let* $\mathbf{V}$ *and* $\mathbf{W}$ *be two* $\mathcal{C}$*-varieties of stamps. Then for each alphabet* Σ, $\mathcal{L}(\mathbf{V} * \mathbf{W})(\Sigma^*)$ *is the inclusion-wise smallest lattice of languages over* Σ *containing* $\mathcal{L}(\mathbf{W})(\Sigma^*)$ *and the languages of the form* $\sigma_\psi^{-1}(V)$*, where* $\psi \colon \Sigma^* \to N$ *is some stamp in* $\mathbf{W}$ *from* $(\Sigma^*, \cdot)$ *to a finite monoid* $(N, \perp)$ *and* $V \in \mathcal{L}(\mathbf{V})((\Sigma \times N)^*)$.

Proof. Fix Σ some alphabet. Let us denote by $\mathcal{E}$ the inclusion-wise smallest lattice of languages over Σ containing $\mathcal{L}(\mathbf{W})(\Sigma^*)$ and the languages of the form $\sigma_\psi^{-1}(V)$, where $\psi \colon \Sigma^* \to N$ is some stamp in $\mathbf{W}$ from $(\Sigma^*, \cdot)$ to a finite monoid $(N, \perp)$ and $V \in \mathcal{L}(\mathbf{V})((\Sigma \times N)^*)$.

Let L be some language in $\mathcal{L}(\mathbf{V} * \mathbf{W})(\Sigma^*)$ and let $\eta_L \colon \Sigma^* \to J$ its syntactic morphism, where $(J, \bullet)$ is its syntactic monoid. By definition, η_L is a stamp of $\mathbf{V} * \mathbf{W}$, so it $\mathcal{C}$-divides some wreath product of the form $\varphi \wr \psi \colon \Gamma^* \to K$ from $(\Gamma^*, \cdot)$ to a finite monoid $(K, \diamond)$ where $\varphi \colon (\Gamma \times N)^* \to M$ is a stamp in $\mathbf{V}$ from $((\Gamma \times N)^*, \cdot)$ to a finite monoid $(M, *)$ and

$\psi\colon \Gamma^* \to N$ is a stamp in $\mathbf{W}$ from $(\Gamma^*, \cdot)$ to a finite monoid $(N, \bot)$, Γ being an alphabet. This means there exist a monoid morphism $f\colon \Sigma^* \to \Gamma^*$ in $\mathcal{C}$ from $(\Sigma^*, \cdot)$ to $(\Gamma^*, \cdot)$ and a surjective monoid morphism $\alpha\colon \mathfrak{Im}((\varphi \wr \psi) \circ f) \to J$ from $(\mathfrak{Im}((\varphi \wr \psi) \circ f), \diamond|_{\mathfrak{Im}((\varphi \wr \psi) \circ f)})$ to $(J, \bullet)$ verifying $\eta_L = \alpha \circ (\varphi \wr \psi) \circ f$. We know there exists $F \subseteq J$ such that $L = \eta_L^{-1}(F)$, but by Proposition 3.2.13 and Proposition 3.2.6, since $(\varphi \wr \psi)^{-1}(\alpha^{-1}(F))$ is obviously recognised by $\varphi \wr \psi$, $(\varphi \wr \psi)^{-1}(\alpha^{-1}(F))$ is a finite union of languages of the form $W \cap \sigma_\psi^{-1}(V)$ where $W \in \mathcal{L}(\mathbf{W})(\Gamma^*)$ and $V \in \mathcal{L}(\mathbf{V})((\Gamma \times N)^*)$. This means that $L = f^{-1}((\varphi \wr \psi)^{-1}(\alpha^{-1}(F)))$ is a finite union of languages of the form $f^{-1}(W) \cap f^{-1}(\sigma_\psi^{-1}(V))$ where $W \in \mathcal{L}(\mathbf{W})(\Gamma^*)$ and $V \in \mathcal{L}(\mathbf{V})((\Gamma \times N)^*)$, as inverses of monoid morphisms commute with union and intersection. But since $\mathcal{L}(\mathbf{W})$ is a $\mathcal{C}$-variety of languages and f belongs to $\mathcal{C}$, $f^{-1}(W)$ belongs to $\mathcal{L}(\mathbf{W})(\Sigma^*)$ for all $W \in \mathcal{L}(\mathbf{W})(\Gamma^*)$. Moreover, according to Claim 3.2.16 at the end of the current proof, there exists a stamp $\psi'\colon \Sigma^* \to N'$ in $\mathbf{W}$ from $(\Sigma^*, \cdot)$ to a submonoid $(N', \bot|_{N'})$ of $(N, \bot)$ such that for any $V \in \mathcal{L}(\mathbf{V})((\Gamma \times N)^*)$ there exists $V' \in \mathcal{L}(\mathbf{V})((\Sigma \times N')^*)$ verifying $f^{-1}(\sigma_\psi^{-1}(V)) = \sigma_{\psi'}^{-1}(V')$. So L belongs to $\mathcal{E}$ and as it is true for any L, this shows $\mathcal{L}(\mathbf{V} * \mathbf{W})(\Sigma^*) \subseteq \mathcal{E}$.

Conversely, Proposition 3.2.14 combined with Proposition 3.2.6 tells us that any language in $\mathcal{L}(\mathbf{W})(\Sigma^*)$ is necessarily also in $\mathcal{L}(\mathbf{V} * \mathbf{W})(\Sigma^*)$ (because the full language $(\Sigma \times N)^*$ for any finite monoid $(N, \bot)$, whose inverse by σ_ψ for any stamp $\psi\colon \Sigma^* \to N$ from $(\Sigma^*, \cdot)$ to $(N, \bot)$ is the full language Σ^*, can be recognised by any trivial stamp from $((\Sigma \times N)^*, \cdot)$ to some trivial monoid, that necessarily belongs to $\mathbf{V}$) as well as all languages of the form $\sigma_\psi^{-1}(V)$, where $\psi\colon \Sigma^* \to N$ is some stamp in $\mathbf{W}$ from $(\Sigma^*, \cdot)$ to a finite monoid $(N, \bot)$ and $V \in \mathcal{L}(\mathbf{V})((\Sigma \times N)^*)$ (because the full language Σ^* can be recognised by the stamp ψ). Since $\mathcal{L}(\mathbf{V} * \mathbf{W})(\Sigma^*)$ is closed under union and intersection, it follows that $\mathcal{E} \subseteq \mathcal{L}(\mathbf{V} * \mathbf{W})(\Sigma^*)$.

Therefore, $\mathcal{L}(\mathbf{V} * \mathbf{W})(\Sigma^*) = \mathcal{E}$.

Claim 3.2.16. *Let Γ be an alphabet. Let $\psi\colon \Gamma^* \to N$ be some stamp in $\mathbf{W}$ from $(\Gamma^*, \cdot)$ to a finite monoid $(N, \bot)$ and $f\colon \Sigma^* \to \Gamma^*$ a monoid morphism in $\mathcal{C}$ from $(\Sigma^*, \cdot)$ to $(\Gamma^*, \cdot)$. Then there exists a stamp $\psi'\colon \Sigma^* \to N'$ in $\mathbf{W}$ from $(\Sigma^*, \cdot)$ to a submonoid $(N', \bot|_{N'})$ of $(N, \bot)$ such that for any $V \in \mathcal{L}(\mathbf{V})((\Gamma \times N)^*)$ there exists $V' \in \mathcal{L}(\mathbf{V})((\Sigma \times N')^*)$ verifying $f^{-1}(\sigma_\psi^{-1}(V)) = \sigma_{\psi'}^{-1}(V')$.*

Proof. Let us consider the stamp $\psi'\colon \Sigma^* \to N'$ from $(\Sigma^*, \cdot)$ to the submonoid $(N', \bot|_{N'})$ of $(N, \bot)$ equal to $\psi \circ f$, where $N' = \mathfrak{Im}(\psi \circ f)$. This means that ψ' $\mathcal{C}$-divides ψ, so that $\psi' \in \mathbf{W}$.

Let $V \in \mathcal{L}(\mathbf{V})((\Gamma \times N)^*)$; there exist $\varphi\colon (\Gamma \times N)^* \to M$ a stamp in $\mathbf{V}$ from $((\Gamma \times N)^*, \cdot)$ to a finite monoid $(M, *)$ and $F \subseteq M$ verifying $V = \varphi^{-1}(F)$. We define the monoid

morphism $g\colon (\Sigma \times N')^* \to (\Gamma \times N)^*$ as the unique monoid morphism from $((\Sigma \times N')^*, \cdot)$ to $((\Gamma \times N)^*, \cdot)$ verifying

$$g((a,x)) = \begin{cases} \varepsilon & \text{if } |f_a| = 0 \\ (u_1, x)(u_2, x \perp \psi(u_1)) \cdots (u_{|u|}, x \perp \psi(u_1 u_2 \cdots u_{|u|-1})) & \text{otherwise } (u = f(a)) \end{cases}$$

for all $(a,x) \in \Sigma \times N'$. We have $\{|g((a,x))| \mid (a,x) \in \Sigma \times N'\} = \{|f(a)| \mid a \in \Sigma\}$, so that g does also belong to $\mathcal{C}$, since it is a class of monoid morphisms between free monoids generated by alphabets satisfying the properties as required in Subsection 3.1.6. We can now consider the stamp $\varphi'\colon (\Sigma \times N')^* \to M'$ from $((\Sigma \times N')^*, \cdot)$ to the submonoid $(M', *|_{M'})$ of $(M, *)$ equal to $\varphi \circ g$, where $M' = \mathfrak{Im}(\varphi \circ g)$. This means that φ' $\mathcal{C}$-divides φ, so that $\varphi' \in \mathbf{V}$.

It is quite straightforward to prove that for all $w \in \Sigma^*$, $g(\sigma_{\psi'}(w)) = \sigma_\psi(f(w))$, by induction on the length of w, so that $g \circ \sigma_{\psi'} = \sigma_\psi \circ f$. Hence, we get that

$$\begin{aligned} f^{-1}(\sigma_\psi^{-1}(\varphi^{-1}(F))) = f^{-1}(\sigma_\psi^{-1}(\varphi^{-1}(F \cap M'))) &= \sigma_{\psi'}^{-1}(g^{-1}(\varphi^{-1}(F \cap M'))) \\ &= \sigma_{\psi'}^{-1}({\varphi'}^{-1}(F \cap M')) \end{aligned}$$

because $\varphi(\sigma_\psi(f(\Sigma^*))) = \varphi(g(\sigma_{\psi'}(\Sigma^*))) \subseteq \mathfrak{Im}(\varphi \circ g) = M'$. In conclusion, $f^{-1}(\sigma_\psi^{-1}(V)) = \sigma_{\psi'}^{-1}(V')$ where $V' = {\varphi'}^{-1}(F \cap M')$ is a language in $\mathcal{L}(\mathbf{V})((\Sigma \times N')^*)$, by Proposition 3.2.6 since φ' is a stamp in $\mathbf{V}$ recognising V'. □

This ends the proof of the theorem as all this is true for any alphabet Σ. □

We only proved the wreath product principle in the more general framework of $\mathcal{C}$-varieties of stamps, but the following lemma shows us it can also be applied in the case of varieties of finite semigroups or monoids. Indeed, for $\mathbf{V}$ and $\mathbf{W}$ two varieties of finite semigroups or monoids, at least one of which is a variety of finite semigroups, we have $\mathcal{L}(\mathbf{V} * \mathbf{W}) = \mathcal{L}(\langle \mathbf{V} * \mathbf{W} \rangle_{ne})$; and when $\mathbf{V}$ and $\mathbf{W}$ are varieties of finite monoids, we have $\mathcal{L}(\mathbf{V} * \mathbf{W}) = \mathcal{L}(\langle \mathbf{V} * \mathbf{W} \rangle_{all})$.

Lemma 3.2.17. *Let $\mathbf{V}$ and $\mathbf{W}$ be two varieties of finite semigroups or monoids, where $\mathbf{V}$ or $\mathbf{W}$ is a variety of finite semigroups; then $\langle \mathbf{V} * \mathbf{W} \rangle_{\mathrm{ne}} = \langle \mathbf{V} \rangle_{\mathrm{ne}} * \langle \mathbf{W} \rangle_{\mathrm{ne}}$. When $\mathbf{V}$ and $\mathbf{W}$ are varieties of finite monoids, we have $\langle \mathbf{V} * \mathbf{W} \rangle_{\mathrm{all}} = \langle \mathbf{V} \rangle_{\mathrm{all}} * \langle \mathbf{W} \rangle_{\mathrm{all}}$.*

Proof. Assume at least one of $\mathbf{V}$ and $\mathbf{W}$ is a variety of finite semigroups (the case when both are varieties of finite monoids is treated similarly).

Left-to-right inclusion. Let $(S,*) \in \mathbf{V}$ and $(T,\perp) \in \mathbf{W}$ be two finite semigroups. Let $(K,\diamond)$ be the submonoid of $(S^1,*^1) \wr (T^1,\perp^1)$ generated by $S^{T^1} \times T$. Let $\eta\colon (S^{T^1} \times T)^* \to K$ be the unique monoid morphism from $((S^{T^1} \times T)^*,\cdot)$ to $(K,\diamond)$ such that $\eta(f,t) = (f,t)$ for all $(f,t) \in S^{T^1} \times T$: it is obvious that it is a stamp of $\langle \mathbf{V} * \mathbf{W} \rangle_{ne}$, as $(K,\diamond)$ is isomorphic to $((S,*) \wr (T,\perp))^1$.

Let $\varphi\colon (S^{T^1} \times T \times T^1)^* \to S^1$ be the unique stamp from $((S^{T^1} \times T \times T^1)^*,\cdot)$ to $(S^1,*^1)$ such that for all $(f,t,t') \in S^{T^1} \times T \times T^1$, $\varphi(f,t,t') = f(t') \in S$. Observe that $\varphi((S^{T^1} \times T \times T^1)^+) = S$, so that $\varphi \in \langle \mathbf{V} \rangle_{ne}$.

Let $\psi\colon (S^{T^1} \times T)^* \to T^1$ be the unique stamp from $((S^{T^1} \times T)^*,\cdot)$ to $(T^1,\perp^1)$ such that for all $(f,t) \in S^{T^1} \times T$, $\psi(f,t) = t$. Observe that $\psi((S^{T^1} \times T)^+) = T$, so that $\psi \in \langle \mathbf{W} \rangle_{ne}$.

Now, note that for each $(f,t) \in S^{T^1} \times T$, f is the function of $S^{1^{T^1}}$ such that $f(t') = \varphi(f,t,t')$ for all $t' \in T^1$. But $(K,\diamond)$ is the submonoid of $(S^1,*^1) \wr (T^1,\perp^1)$ generated by $S^{T^1} \times T = \{(f,\psi(f,t)) \mid (f,t) \in S^{T^1} \times T\}$ and η is the unique monoid morphism from $((S^{T^1} \times T)^*,\cdot)$ to $(K,\diamond)$ verifying that $\eta(f,t) = (f,t) = (f,\psi(f,t))$ for all $(f,t) \in (S^{T^1} \times T)$. Hence, η is the wreath product $\varphi \wr \psi$ of φ and ψ, so that $\eta \in \langle \mathbf{V} \rangle_{ne} * \langle \mathbf{W} \rangle_{ne}$.

Let now more generally $\mu\colon \Sigma^* \to K'$ be a stamp from the free monoid $(\Sigma^*,\cdot)$ generated by the alphabet Σ to the finite monoid $(K',\diamond')$ such that $\mu(\Sigma^+) = U$ where $(U,\bullet)$ is some finite semigroup dividing the wreath product $(S,*) \wr (T,\perp)$. It is not too difficult to see that μ *lp*-divides η, so that we also have $\mu \in \langle \mathbf{V} \rangle_{ne} * \langle \mathbf{W} \rangle_{ne}$.

As this is true for any two finite semigroups $(S,*) \in \mathbf{V}$ and $(T,\perp) \in \mathbf{W}$, we have that $\langle \mathbf{V} * \mathbf{W} \rangle_{ne} \subseteq \langle \mathbf{V} \rangle_{ne} * \langle \mathbf{W} \rangle_{ne}$.

Right-to-left inclusion. Let Σ be an alphabet. Let $\varphi\colon (\Sigma \times N)^* \to M$ be a stamp in $\langle \mathbf{V} \rangle_{ne}$ from $((\Sigma \times N)^*,\cdot)$ to a finite monoid $(M,*)$ and $\psi\colon \Sigma^* \to N$ a stamp in $\langle \mathbf{W} \rangle_{ne}$ from $(\Sigma^*,\cdot)$ to a finite monoid $(N,\perp)$. This means that $\varphi((\Sigma \times N)^+) = S$ where $(S,*|_S)$ is a finite semigroup in $\mathbf{V}$ and $\psi(\Sigma^+) = T$ where $(T,\perp|_T)$ is a finite semigroup in $\mathbf{W}$. For each $a \in \Sigma$, let $f_a \in M^N$ be the function defined by $f_a(x) = \varphi(a,x)$ for all $x \in N$. Let finally $\mu\colon \Sigma^* \to K$ be the stamp, from $(\Sigma^*,\cdot)$ to the finite monoid $(K,\diamond)$, that is the wreath product $\varphi \wr \psi$, so that we have that $(K,\diamond)$ is the submonoid of $(M,*) \wr (N,\perp)$ generated by $\{(f_a,\psi(a)) \mid a \in \Sigma\}$ and μ is the unique monoid morphism from $(\Sigma^*,\cdot)$ to $(K,\diamond)$ verifying that $\mu(a) = (f_a,\psi(a))$ for all $a \in \Sigma$.

For each $a \in \Sigma$, we know that, in fact, $f_a \in S^N$, because $\varphi(a,x) \in S$ for all $x \in N$, and $\psi(a) \in T$. So, since $(T,\perp|_T)^1$ is isomorphic to $(N,\perp)$, we get that the subsemigroup of $(M,*) \wr (N,\perp)$ generated by $\{\mu(a) \mid a \in \Sigma\} = \{(f_a,\psi(a)) \mid a \in \Sigma\}$ is isomorphic to a subsemigroup of $(S,*|_S) \wr (T,\perp|_T)$, so that $(\mu(\Sigma^+),\diamond|_{\mu(\Sigma^+)})$ semigroup-divides $(S,*|_S) \wr (T,\perp|_T)$, which means $\mu \in \langle \mathbf{V} * \mathbf{W} \rangle_{ne}$.

More generally, we have that any stamp that *ne*-divides the wreath product $\mu = \varphi \wr \psi$ does necessarily also belong to $\langle \mathbf{V} * \mathbf{W} \rangle_{ne}$.

As this is true for any two stamps $\varphi \in \langle \mathbf{V} \rangle_{ne}$ and $\psi \in \langle \mathbf{W} \rangle_{ne}$ of which we can take the wreath product, we have that $\langle \mathbf{V} * \mathbf{W} \rangle_{ne} \supseteq \langle \mathbf{V} \rangle_{ne} * \langle \mathbf{W} \rangle_{ne}$. □

For a given variety of finite monoids $\mathbf{V}$, the two most important varieties obtained by wreath product in this thesis are $\langle \mathbf{V} \rangle_{all} * \mathbf{MOD}$ and $\mathbf{V} * \mathbf{D}$. The latter is important because it is central in the notion of locality of $\mathbf{V}$, and the first is because of its intimate link with the quasi-$\mathbf{V}$ *lm*-variety of stamps — yet to be defined — that will appear to be crucial in the last section of the present chapter as well as in the next two chapters. We shall only describe informally what the corresponding varieties of languages are (see Paperman [2014]).

For each alphabet Σ, $\mathcal{L}(\langle \mathbf{V} \rangle_{all} * \mathbf{MOD})(\Sigma^*)$ is the inclusion-wise smallest lattice of languages over Σ containing each language over Σ for which membership of each word over Σ only depends on its length modulo some integer $d \in \mathbb{N}_{>0}$, as well as each language L over Σ for which there is an integer $d \in \mathbb{N}_{>0}$ and a language L' in $\mathcal{L}(\mathbf{V})((\Sigma \times \mathbb{Z}/d\,\mathbb{Z})^*)$ such that L is the set of words $w \in \Sigma^*$ that belong to L' after appending to each letter of w its position minus 1 modulo d.

Similarly, for each alphabet Σ, $\mathcal{L}(\mathbf{V} * \mathbf{D})(\Sigma^*)$ is the inclusion-wise smallest lattice of languages over Σ containing each language over Σ for which membership of each word over Σ only depends on its $k \in \mathbb{N}$ last letters, as well as each language L over Σ for which there is an integer $k \in \mathbb{N}$ and a language L' in $\mathcal{L}(\mathbf{V})(\Sigma \times \Sigma^{\leq k})$ such that L is the set of words $w \in \Sigma^*$ that belong to L' after appending to each letter of w the word composed of the k (or fewer when near the beginning of w) letters preceding that letter.

3.3 Recognition by programs

Morphisms into finite monoids only allow recognising regular languages. To recognise bigger classes of languages, such as the well-known complexity classes presented in Chapter 1, there are two non-necessarily exclusive options:

- generalise the notion of finite monoid;
- generalise the notion of morphism.

One example mixing both is given by Krebs, Lange and Reifferscheid, who introduced the notion of a finitely typed monoid (that may be infinite) and an associated notion of a typed morphism in Krebs et al. [2007].

Choosing option two alone, Barrington and Thérien introduced in Barrington and Thérien [1988] the notion of recognition by finite monoids through programs, a generalisation of morphisms, that has strong links with the notion of decision by sequences of "small-depth" circuits. We are now going to present this notion and these links.

3.3.1 Definition and challenges

In Borodin et al. [1986], Borodin, Dolev, Fich and Paul introduced a special type of BPs, *levelled branching programs* (LBPs). A branching program $P = (X, \delta, \tau, s, t_0, t_1)$ on Σ^V, for some alphabet Σ and finite set $V \subseteq \mathbb{N}$, is said to be levelled if and only if its set of vertices X can be partitioned into subsets $L_0, L_1, \ldots, L_l$ ($l \in \mathbb{N}$) such that $s \in L_0$, $L_l = \{t_0, t_1\}$ and for each $i \in [\![0, l-1]\!]$, any arc going out of some vertex $u \in L_i$ goes into a vertex $v \in L_{i+1}$, i.e. $\delta(u, a) \subseteq L_{i+1}$ for any $a \in \Sigma$. The *length* of P is l and its *width* is defined as the size of its biggest level, i.e. $\max_{i \in [\![0,l]\!]} |L_i|$. It then makes sense to consider *bounded-width levelled branching programs*, that have a fixed maximum width (we will denote by w-LBP a levelled branching program of width at most $w \in \mathbb{N}$). In Borodin et al. [1986], the authors proved that the language MAJORITY $= \{w \in \{0,1\}^* \mid |w|_1 \geq \left\lceil \frac{|w|}{2} \right\rceil\}$ on the alphabet $\{0, 1\}$ of words containing a majority of 1's cannot be decided by a sequence of 2-LBPs whose length is a function in $\mathrm{O}(n^2/\log n)$. In fact, they conjectured that bounded-width LBPs of polynomial length cannot decide MAJORITY, i.e. for any $w \in \mathbb{N}$, MAJORITY cannot be decided by a sequence of w-LBPs whose length is a function in $\mathrm{O}(n^k)$ for some $k \in \mathbb{N}$.

However, it was a surprise when Barrington showed in Barrington [1989] that the language MAJORITY is in fact decidable by bounded-width LBPs of polynomial length and that moreover any language in NC^1 has this property (the converse being more easily seen to be true). To do this, he considered some even more specific type of BPs: those are kind of LBPs such that, for some fixed width $w \in \mathbb{N}$, each level is an instruction querying the input at a given index and outputting a possibly different function $f \colon [w] \to [w]$ for each possible queried value. Acceptance or rejection for a specific input word then depends on the function obtained by composition of the individual functions output at each level. Barrington showed that width 5 suffices, and more precisely that any language in NC^1 can be decided by a sequence of such specific LBPs of width 5 and polynomial length that use only functions from the group $(S_5, \circ)$, the group of permutations over [5]. Barrington and Thérien soon discovered (see Barrington and Thérien [1988]) that this specific definition of an LBP gives a way to meaningfully generalise finite automata to the non-uniform setting by restricting the provenance of the functions used to a certain finite

monoid. They introduced the notion of *programs over monoids*, that happens to give an algebraic, finite semigroup-theoretical, point of view on NC^1 and its subclasses.

Definition 3.3.1 (Programs over monoids)**.** Let Σ be an alphabet and $(M, *)$ be a finite monoid. A *program over the monoid* $(M, *)$ *on* Σ^n (or $(M, *)$*-program on* Σ^n), for some $n \in \mathbb{N}$, is a word $P \in ([n] \times M^\Sigma)^*$ over the alphabet $[n] \times M^\Sigma$, each letter of P being an *instruction*. The *length of* P, denoted by $|P|$ is the length of P as a word. If we let $l = |P|$, we have $P = (p_1, f_1)(p_2, f_2) \cdots (p_l, f_l)$ with $p_i \in [n]$ and $f_i \colon \Sigma \to M$ for all $i \in [l]$. We denote by $\xi_P \colon \Sigma^n \to M^*$ the *evaluation function of* P that transforms each word $w \in \Sigma^n$ into a word over M of length l according to the instructions of P, i.e.

$$\xi_P(w) = f_1(w_{p_1}) f_2(w_{p_2}) \cdots f_l(w_{p_l}) \ .$$

Then, P uniquely defines a function from Σ^n to M obtained by multiplying out (in $(M, *)$) the elements of the word $\xi_P(w)$ over M, i.e.

$$P(w) = f_1(w_{p_1}) * f_2(w_{p_2}) * \cdots * f_l(w_{p_l}) = \eta_{(M,*)}(\xi_P(w))$$

for all $w \in \Sigma^n$, where $\eta_{(M,*)}$ is the evaluation morphism of $(M, *)$.

Remark 3.3.2. In the above definition, when $P = \varepsilon$, then $\xi_P(w) = \varepsilon$ and $P(w) = 1_{(M,*)}$ for all $w \in \Sigma^n$.

Programs over monoids allow defining a new notion of recognition, generalising that by morphisms in a non-uniform way, as follows.

Definition 3.3.3. Let Σ be an alphabet and $(M, *)$ be a finite monoid.

For each $n \in \mathbb{N}$, a language $L \subseteq \Sigma^n$ is *recognised* by an $(M, *)$-program P on Σ^n if and only if there exists $F \subseteq M$ such that $L = P^{-1}(F)$.

Example 3.3.4. Consider the finite monoid $(\mathbb{Z}/2\,\mathbb{Z}, \times)$ where $\times$ denotes the canonical product modulo 2. Let $f_a, f_b \in (\mathbb{Z}/2\,\mathbb{Z})^{\{a,b\}}$ such that $f_a(a) = f_b(b) = 1$ and $f_a(b) = f_b(a) = 0$. For all $n \in \mathbb{N}$, the $(\mathbb{Z}/2\,\mathbb{Z}, \times)$-program

$$P_{2n} = (1, f_a) \cdots (n, f_a)(n+1, f_b) \cdots (2n, f_b)$$

on $\{a, b\}^{2n}$ recognises the language $\{a^n b^n\}$, as $\{a^n b^n\} = P^{-1}(1)$.

Definition 3.3.5. Let Σ be an alphabet and $(M, *)$ be a finite monoid. Let $L \subseteq \Sigma^*$ be a language on Σ. We say that a sequence $(P_n)_{n \in \mathbb{N}}$ of $(M, *)$-programs such that each P_n is

on Σ^n for $n \in \mathbb{N}$ *recognises* L if and only if for each $n \in \mathbb{N}$, P_n recognises $L^{=n}$, i.e. there exists a sequence $(F_n)_{n\in\mathbb{N}}$ of subsets of M such that $L^{=n} = P_n^{-1}(F_n)$ for each $n \in \mathbb{N}$.

For all $s\colon \mathbb{N} \to \mathbb{N}$ and alphabet Σ, we define $\mathcal{P}(\Sigma^*, (M,*), s(n))$ as the class of languages L over the alphabet Σ such that there exists a constant $\alpha \in \mathbb{R}_{>0}$ and a sequence of $(M,*)$-programs $(P_n)_{n\in\mathbb{N}}$ recognising L, where for all $n \in \mathbb{N}$, $|P_n| \leq \alpha \cdot s(n)$. We define

$$\mathcal{P}(\Sigma^*, (M,*)) = \bigcup_{k\in\mathbb{N}} \mathcal{P}\big(\Sigma^*, (M,*), n^k\big)$$

as the set of languages over Σ that are recognised by sequences of polynomial-length programs over $(M,*)$; in that case, we also say that these languages are p-*recognised* by these sequences of programs as well as by the finite monoid $(M,*)$.

Let now $\mathbf{V}$ be a variety of finite monoids, we similarly define

$$\mathcal{P}(\Sigma^*, \mathbf{V}, s(n)) = \bigcup_{(M,*)\in\mathbf{V}} \mathcal{P}(\Sigma^*, (M,*), s(n))$$

and $\mathcal{P}(\Sigma^*, \mathbf{V}) = \bigcup_{(M,*)\in\mathbf{V}} \mathcal{P}(\Sigma^*, (M,*))$.

Finally $\mathcal{P}((M,*), s(n))$, $\mathcal{P}((M,*))$, $\mathcal{P}(\mathbf{V}, s(n))$ and $\mathcal{P}(\mathbf{V})$ will just denote the union, over all alphabets Σ, of the classes $\mathcal{P}(\Sigma^*, (M,*), s(n))$, $\mathcal{P}(\Sigma^*, (M,*))$, $\mathcal{P}(\Sigma^*, \mathbf{V}, s(n))$ and $\mathcal{P}(\Sigma^*, \mathbf{V})$ respectively.

Example 3.3.6. Continuing with Example 3.3.4, for all $n \in \mathbb{N}$, let us define the program $P_{2n+1} = \varepsilon$ over $(\mathbb{Z}/2\,\mathbb{Z}, \times)$ on $\{a,b\}^{2n+1}$ that recognises $\emptyset$. Then, the sequence of $(\mathbb{Z}/2\,\mathbb{Z}, \times)$-programs $(P_n)_{n\in\mathbb{N}}$ recognises $\{a^n b^n \mid n \in \mathbb{N}\}$ which is, by the way, well known to be a non-regular language. Thus, we have $\{a^n b^n \mid n \in \mathbb{N}\} \in \mathcal{P}((\mathbb{Z}/2\,\mathbb{Z}, \times), n) \subseteq \mathcal{P}((\mathbb{Z}/2\,\mathbb{Z}, \times))$.

The most striking aspect of this definition is that, as $\mathbf{V}$ ranges over varieties of finite monoids, polynomial-length programs over monoids in $\mathbf{V}$ capture NC^1 and all its known subclasses[3], and allow even to get a very fine variety-parametrised structure in NC^1. This points to deep links between computational complexity theory within NC^1 and algebraic automata theory.

Theorem 3.3.7 (Barrington and Thérien [1988])**.** *We have the following equalities.*

1. $\mathsf{AC}^0 = \mathcal{P}(\mathbf{A})$.

[3]Except TC^0, defined using circuits having arbitrary fan-in threshold gates, in addition to NOT and arbitrary fan-in AND and OR gates — although see Krebs et al. [2007] for such a characterisation using the notion of finitely typed infinite monoids we briefly talked about in the beginning of this section.

2. $\mathsf{ACC}^0 = \mathcal{P}(\mathbf{M}_{sol})$.

3. $\mathsf{NC}^1 = \mathcal{P}(\mathbf{M})$.

Unfortunately, though this formalism sharpened the computational complexity theorists' understanding of NC^1 and its subclasses and gave a new, algebraic way, to approach those, it did not help to prove new separation results inside NC^1 for the moment. At the time of writing of this thesis, nobody has in fact even been able to reprove the now well-established results on constant-depth circuits (e.g., that $\mathrm{MOD}_2 \notin \mathsf{AC}^0$) by using novel semigroup-theoretic-flavoured techniques. Thus, as this already seems to be a difficult problem, proving new separation results directly using the "program over monoid" formalism seems currently to be out of reach, and the work around this formalism presented in the two remaining chapters of this thesis represents a much more modest contribution.

"If you can't solve a problem, then there is an easier problem you can solve: find it." Applying this principle by George Pólya (to be found in his book "How to Solve It"), in the remainder of this chapter and the two that follow, we try to understand the computational power of (polynomial-length) programs over monoids taken from small varieties of finite (aperiodic) monoids[4], using some more general observations on the link between regular languages and program-recognition we first do. The hope behind this line of work is that a better understanding of $\mathcal{P}(\mathbf{V})$ for varieties of finite monoids $\mathbf{V} \subset \mathbf{A}$ will trigger new insights into how to tackle questions about $\mathcal{P}(\mathbf{A})$ without the need for constant-depth circuit lower bounds technology such as Håstad's switching lemma and Razborov's method of approximations (see Jukna [2012]). The most fundamental of these questions is to show that $\mathrm{MOD}_m \notin \mathcal{P}(\mathbf{A})$ for all $m \in \mathbb{N}, m \geq 2$. Solving this could in turn give new ideas to attack open questions concerning the other common classes in NC^1.

Our work in the remainder of this chapter and the two that follow builds on a considerable body of previous research. McKenzie and Thérien in McKenzie and Thérien [1989], and later together with Péladeau in McKenzie et al. [1991], were the first to define, and propose a systematic investigation of, $\mathcal{P}(\mathbf{V})$ when $\mathbf{V}$ ranges over all varieties of finite monoids, pinpointing the strong link between regular languages and program-recognition as well as giving various results in the framework of such an investigation. Around the same time, Barrington, Compton, Straubing and Thérien Barrington et al. [1992] managed to give an exact characterisation of the regular languages on $\{0,1\}$ in AC^0 and $\mathsf{ACC}^0[p]$ for all $p \in \mathbb{N}$ prime, building on the at the time recent constant-depth circuits size lower bound results. Péladeau, in his Ph.D. thesis Péladeau [1990], introduced the notion of

[4] "Small" is not well defined here and just means, informally, that those varieties are quite restricted, e.g. well within $\mathbf{A}$, at the bottom of the infinite hierarchy of varieties of finite monoids it contains.

a p-variety of finite monoids, in an attempt to prove an Eilenberg-type theorem in the case of p-recognition by finite monoids (relying on a still unproven conjecture). Along with Straubing and Thérien in a subsequent article Péladeau et al. [1997], he gave a general characterisation of the class of regular languages p-recognised by semigroups taken from any p-variety of finite semigroups of the form $\mathbf{V} * \mathbf{LI}$ where $\mathbf{V}$ is a variety of finite monoids (when everything is appropriately carried over to the case of finite semigroups). The property of being a p-variety has later been proven to hold for various varieties of finite monoids by Straubing Straubing [2000, 2001] (although he used a stronger non-equivalent notion of p-recognition by a finite monoid), as well as by Lautemann, Tesson and Thérien Lautemann et al. [2006], but also for the variety of finite semigroups $\mathbf{J} * \mathbf{LI}$ by Maciel, Péladeau and Thérien Maciel et al. [2000].

Concerning more specifically the computational power of programs over monoids taken from small varieties of finite aperiodic monoids, we note an early article by Thérien Thérien [1989] that precisely studies programs over aperiodic monoids and their unability to decide MOD_m for some $m \in \mathbb{N}, m \geq 2$, using the at the time recent results on lower bounds for the size of constant-depth Boolean circuits deciding these languages. Later on, along with Tesson in Tesson and Thérien [2001], he carried out an in-depth study of the notions of universality and polynomial-length property already at least implicitly considered in Thérien [1989]. Tesson pushed this study slightly further in his Ph.D. thesis Tesson [2003]. Gavaldà and Thérien Gavaldà and Thérien [2003] investigated the link between programs over monoids taken from small varieties of finite aperiodic monoids (or semigroups) and restricted Boolean computation models such as DNFs and bounded-rank decision trees to establish correspondences between classes.

Our contributions within this line of work, presented in the remainder of this chapter and the two that follow, can be summarised in short as follows: we introduce a notion that strengthens that of a p-variety of finite monoids, we apply the notion to study the classes of regular languages p-recognised by monoids taken from, respectively, **DA** and **J**, and we report some findings about the polynomial-length property for those monoids. We point out that, to the best of our knowledge, all the definitions and results stated from here onwards are new, unless otherwise stated. Subsection 3.4.2 in the next section and Chapter 4 are based on the article Grosshans et al. [2017].

3.3.2 Some general properties of programs

In this subsection, we review some basic closure properties for recognition by programs, most of them to be found in McKenzie et al. [1991] in the context of p-recognition, often

refining their statements and proofs.

The first, and very obvious, property of programs, is that, no matter what variety of finite monoids we consider, any trivial language can be recognised by a sequence of programs of constant length (actually 0 length) over some finite monoid of that variety (actually, any such monoid).

Proposition 3.3.8. *For any variety of finite monoids* $\mathbf{V}$*, for each alphabet* Σ*, we have* $\{\emptyset, \Sigma^*\} \subseteq \mathcal{P}(\mathbf{V}, 1)$.

Sequences of programs over monoids, each sequence recognising some language, can be combined to recognise some Boolean combination of those languages.

Lemma 3.3.9. *Let* $(M_1, *_1), \ldots, (M_r, *_r)$ *be* $r \in \mathbb{N}_{>0}$ *finite monoids. For all* $n \in \mathbb{N}$ *and for any Boolean combination* L *of languages* $L_1, \ldots, L_r \subseteq \Sigma^n$ *such that each* L_i *for* $i \in [r]$ *is recognised by an* $(M_i, *_i)$*-program* P_i *on* Σ^n*, there exists an* $(M_1, *_1) \times \cdots \times (M_r, *_r)$*-program* P *on* Σ^n *recognising* L *with* $|P| = \sum_{i=1}^{r} |P_i|$.

Proof. Let $n \in \mathbb{N}$ and let $L \subseteq \Sigma^n$ be some Boolean combination of languages $L_1, \ldots, L_r \subseteq \Sigma^n$ such that each L_i for $i \in [r]$ is recognised by an $(M_i, *_i)$-program P_i on Σ^n.

This means that for each $i \in [r]$, there exists $F_i \subseteq M_i$ verifying $L_i = P_i^{-1}(F_i)$ and

$$P_i = (p_{i,1}, f_{i,1})(p_{i,2}, f_{i,2}) \cdots (p_{i,l_i}, f_{i,l_i})$$

where $l_i = |P_i|$ and $(p_{i,j}, f_{i,j}) \in [n] \times {M_i}^{\Sigma}$ for each $j \in [l_i]$.

Now for each $i \in [r]$, we can define an $(M_1, *_1) \times \cdots \times (M_r, *_r)$-program P_i' on Σ^n of the same length as P_i so that

$$P_i' = (p_{i,1}, f_{i,1}')(p_{i,2}, f_{i,2}') \cdots (p_{i,l_i}, f_{i,l_i}')$$

where for each $j \in [l_i]$, $f_{i,j}' \in (M_1 \times \cdots \times M_r)^{\Sigma}$ is defined by

$$f_{i,j}'(a) = (1_{(M_1,*_1)}, \ldots, 1_{(M_{i-1},*_{i-1})}, f_{i,j}(a), 1_{(M_{i+1},*_{i+1})}, \ldots, 1_{(M_r,*_r)}) \ .$$

It is obvious that for all $w \in \Sigma^n$, we have

$$P_i'(w) = (1_{(M_1,*_1)}, \ldots, 1_{(M_{i-1},*_{i-1})}, P_i(w), 1_{(M_{i+1},*_{i+1})}, \ldots, 1_{(M_r,*_r)}) \ .$$

Hence, we can set P to be the $(M_1, *_1) \times \cdots \times (M_r, *_r)$-program on Σ^n obtained by concatenating the programs P_1' to P_r', i.e. $P = P_1' \cdots P_r'$. By construction, we can directly

see that for all $w \in \Sigma^n$, we have

$$P(w) = P'_1(w) * \cdots * P'_r(w) = (P_1(w), \ldots, P_r(w))$$

where $*$ denotes the product of $(M_1, *_1) \times \cdots \times (M_r, *_r)$.

As L is some Boolean combination of $L_1, \ldots, L_r$, there exist $q \in \mathbb{N}_{>0}$ and for each $i \in [q]$, languages $K_{i,1}, \ldots, K_{i,r} \subseteq \Sigma^n$ verifying that $K_{i,j}$ is equal to L_j or its complement $\Sigma^n \setminus L_j$ for all $j \in [r]$, such that

$$L = \bigcup_{i=1}^{q} \bigcap_{j=1}^{r} K_{i,j} \ .$$

For all $i \in [q]$ and $j \in [r]$, we can now define $F'_{i,j} \subseteq M_j$ as F_j if $K_{i,j} = L_j$ and $M_j \setminus F_j$ otherwise ($K_{i,j} = \Sigma^n \setminus L_j$), so that $K_{i,j} = P_j^{-1}(F'_{i,j})$. It is obvious that $\bigcap_{j=1}^{r} K_{i,j} = P^{-1}(F'_{i,1} \times \cdots \times F'_{i,r})$ for all $i \in [q]$, so that if we set $F = \bigcup_{i=1}^{q} F'_{i,1} \times \cdots \times F'_{i,r}$, we get that $L = \bigcup_{i=1}^{q} \bigcap_{j=1}^{r} K_{i,j} = P^{-1}(F)$, as inverses of functions commute with unions.

Hence, P is a $(M_1, *_1) \times \cdots \times (M_r, *_r)$-program on Σ^n that recognises L and is of length exactly $\sum_{i=1}^{r} |P_i|$. □

As a corollary, we can derive the following result.

Proposition 3.3.10. *For any variety of finite monoids* $\mathbf{V}$ *and any function* $s\colon \mathbb{N} \to \mathbb{N}$*, we have that* $\mathcal{P}(\mathbf{V}, s(n))$ *is closed under Boolean combinations.*

No one was ever able to prove the analogous proposition for varieties of finite semigroups, because it is most certainly false; however, to the best of our knowledge, no one ever proved it to be false either.

To prove that a given language L is recognised by a sequence of programs over a given finite monoid $(M, *)$, it might sometimes be easier to think about that sequence of programs as transforming input words into other words that will be tested for membership in another language L' that we already know to be recognised by a sequence of programs over $(M, *)$. This can be formalised through the notion of a program-reduction, most probably first defined by Péladeau in his Ph.D. thesis Péladeau [1990].

Definition 3.3.11 (Program-reduction)**.** Let Σ and Γ be two alphabets. A Γ-*program on* Σ^n, for some $n \in \mathbb{N}$, is a word $\Psi \in ([n] \times \Gamma^\Sigma)^*$ over the alphabet $[n] \times \Gamma^\Sigma$, each letter of Ψ being an *instruction*. The *length of* Ψ, denoted by $|\Psi|$ is the length of Ψ as a word. If we let $l = |\Psi|$, we have $\Psi = (p_1, f_1)(p_2, f_2) \cdots (p_l, f_l)$ with $p_i \in [n]$ and $f_i\colon \Sigma \to \Gamma$ for all

$i \in [l]$. Ψ uniquely defines a function from Σ^n to Γ^l by transforming each word $w \in \Sigma^n$ into a word in Γ^l according to the instructions of Ψ, i.e.

$$\Psi(w) = f_1(w_{p_1})f_2(w_{p_2})\cdots f_l(w_{p_l}) \ .$$

Let now $L \subseteq \Sigma^*$ and $K \subseteq \Gamma^*$. We say that L *program-reduces to* K if and only if there exists a sequence $(\Psi_n)_{n\in\mathbb{N}}$ of Γ-programs such that Ψ_n is on Σ^n and $L^{=n} = \Psi_n^{-1}(K^{=|\Psi_n|})$ for each $n \in \mathbb{N}$. Defining $s\colon \mathbb{N} \to \mathbb{N}$ by $s(n) = |\Psi_n|$ for all $n \in \mathbb{N}$, we say that $(\Psi_n)_{n\in\mathbb{N}}$ is a *program-reduction from* L *to* K *of length* $s(n)$; when moreover there exists $\alpha \in \mathbb{R}_{>0}$ and $k \in \mathbb{N}$ verifying $s(n) \leq \alpha \cdot n^k$ for all $n \in \mathbb{N}$, we can say it is a p-*reduction from* L *to* K.

Proposition 3.3.12. *Let* Σ *and* Γ *be two alphabets. Let* $(M, *)$ *be a finite monoid. Given some language* K *over* Γ *that is in* $\mathcal{P}((M, *), s(n))$ *for some function* $s\colon \mathbb{N} \to \mathbb{N}$ *and some other language* L *over* Σ *from which there exists a program-reduction to* K *of length* $t(n)$*, for* $t\colon \mathbb{N} \to \mathbb{N}$ *some function, we have that* $L \in \mathcal{P}((M, *), s(t(n)))$.

In particular, when K *is* p-*recognised by* $(M, *)$ *and* L p-*reduces to* K*, we have that* L *is also* p-*recognised by* $(M, *)$.

Proof. Let us take some language K over Γ that is in $\mathcal{P}((M, *), s(n))$ for some function $s\colon \mathbb{N} \to \mathbb{N}$. This means there exists a sequence $(P_n)_{n\in\mathbb{N}}$ of $(M, *)$-programs that recognises L, that is to say, there exists a sequence $(F_n)_{n\in\mathbb{N}}$ of subsets of M such that $K^{=n} = P_n^{-1}(F_n)$ for each $n \in \mathbb{N}$. Moreover, there exists some $\alpha \in \mathbb{R}_{>0}$ verifying that $|P_n| \leq \alpha \cdot s(n)$ for all $n \in \mathbb{N}$.

Let us also take some other language L over Σ from which there exists a program-reduction to K of length $t(n)$, for $t\colon \mathbb{N} \to \mathbb{N}$ some function. This means there exists a sequence $(\Psi_n)_{n\in\mathbb{N}}$ of Γ-programs that is a program-reduction of L to K of length $t(n)$, which implies that for each $n \in \mathbb{N}$, $|\Psi_n| = t(n)$ and $L^{=n} = \Psi_n^{-1}(K^{=t(n)})$.

Let $n \in \mathbb{N}$. Assume that

$$P_{t(n)} = (p_1, f_1)(p_2, f_2)\cdots(p_l, f_l)$$

where $l = \left|P_{t(n)}\right|$, $p_i \in [t(n)]$ and $f_i \in M^\Gamma$ for all $i \in [l]$, as well as

$$\Psi_n = (q_1, g_1)(q_2, g_2)\cdots(q_{t(n)}, g_{t(n)})$$

where $q_i \in [n]$ and $g_i \in \Gamma^\Sigma$ for all $i \in [t(n)]$. We can now build a new $(M, *)$-program on Σ^n as

$$Q_n = (q_{p_1}, f_1 \circ g_{p_1})(q_{p_2}, f_2 \circ g_{p_2})\cdots(q_{p_l}, f_l \circ g_{p_l}) \ .$$

Then, for all $w \in \Sigma^n$, if we set $\hat{w} = \Psi_n(w) = g_1(w_{q_1})g_2(w_{q_2}) \cdots g_{t(n)}(w_{q_{t(n)}})$, we have

$$\begin{aligned} Q_n(w) &= f_1\big(g_{p_1}(w_{q_{p_1}})\big) * f_2\big(g_{p_2}(w_{q_{p_2}})\big) * \cdots * f_l\big(g_{p_l}(w_{q_{p_l}})\big) \\ &= f_1(\hat{w}_{p_1}) * f_2(\hat{w}_{p_2}) * \cdots * f_l(\hat{w}_{p_l}) \\ &= P_{t(n)}(\hat{w}) = P_{t(n)}(\Psi_n(w)) \ . \end{aligned}$$

This finally means that $Q_n = P_{t(n)} \circ \Psi_n$, so that

$$L^{=n} = \Psi_n^{-1}(K^{=t(n)}) = \Psi_n^{-1}(P_{t(n)}^{-1}(F_{t(n)})) = Q_n^{-1}(F_{t(n)}) \ .$$

Moreover, we have that $|Q_n| = l \leq \alpha \cdot s(t(n))$.

Therefore, L is recognised by the sequence $(Q_n)_{n\in\mathbb{N}}$ of $(M, *)$-programs verifying that $|Q_n| \leq \alpha \cdot s(t(n))$ for all $n \in \mathbb{N}$. The conclusions of the proposition follow. □

This result allows us to prove the following additional closure property.

Proposition 3.3.13. *For any variety of finite monoids* $\mathbf{V}$ *and any* $k \in \mathbb{N}$*, we have that* $\mathcal{P}\big(\mathbf{V}, n^k\big)$ *is closed under quotients and inverses of* lm*-morphisms.*

Proof. Let $\mathbf{V}$ be a variety of finite monoids and $k \in \mathbb{N}$. Fix some alphabet Σ and let $L \subseteq \Sigma^*$ be some language over Σ that is in $\mathcal{P}\big(\mathbf{V}, n^k\big)$.

Closure under quotients. Let $u \in \Sigma^*$. Consider $u^{-1}L$. When $|u| = 0$, it is obvious that $u^{-1}L = \varepsilon^{-1}L = L$ belongs to $\mathcal{P}\big(\mathbf{V}, n^k\big)$. Otherwise, we have $|u| > 0$. It is easy to prove that $\{\varepsilon\}$ and Σ^+ both belong to $\mathcal{P}(\mathbf{V}, 1) \subseteq \mathcal{P}\big(\mathbf{V}, n^k\big)$, so that the language L' over Σ defined by

$$L' = \begin{cases} L \cup \{\varepsilon\} & \text{if } u \in L \\ L \cap \Sigma^+ & \text{otherwise } (u \notin L) \end{cases}$$

belongs to $\mathcal{P}\big(\mathbf{V}, n^k\big)$ by Proposition 3.3.10. We are now going to define a program-reduction from $u^{-1}L$ to L'. For this, for each $i \in [|u|]$ we define $f_{u_i} \in \Sigma^\Sigma$ by $f_{u_i}(a) = u_i$ for all $a \in \Sigma$. Let $n \in \mathbb{N}_{>0}$; we define the Σ-program Ψ_n on Σ^n as

$$\Psi_n = (1, f_{u_1}) \cdots (1, f_{u_{|u|}})(1, \mathrm{id}_\Sigma) \cdots (n, \mathrm{id}_\Sigma) \ .$$

It is direct to see that for all $w \in \Sigma^n$, $\Psi_n(w) = uw$, so that $(u^{-1}L)^{=n} = \{w \in \Sigma^n \mid uw \in L\} = \{w \in \Sigma^n \mid \Psi_n(w) \in L'\} = \Psi_n^{-1}(L'^{=n+|u|})$. Moreover, $|\Psi_n| = n + |u|$.

Now, we specifically define the Σ-program $\Psi_0 = \varepsilon$ on Σ^0, the only way it can be defined. By definition of L', we necessarily have $(u^{-1}L)^{=0} = \{u\} \cap L = \Psi_0^{-1}(L'^{=0})$ and moreover, $|\Psi_0| = 0$.

Therefore, $(\Psi_n)_{n\in\mathbb{N}}$ is a program-reduction from $u^{-1}L$ to L' of length $t(n)$ where $t\colon \mathbb{N} \to \mathbb{N}$ is defined by

$$t(n) = \begin{cases} n + |u| & \text{if } n > 0 \\ 0 & \text{otherwise } (n = 0) \end{cases}$$

for all $n \in \mathbb{N}$. This means, by Proposition 3.3.12, that $u^{-1}L \in \mathcal{P}\big(\mathbf{V}, (t(n))^k\big)$. But we have $0^k = (1 + |u|)^k \cdot 0^k$ and $(t(n))^k = (n + |u|)^k \leq (1 + |u|)^k \cdot n^k$ for all $n \in \mathbb{N}_{>0}$, so that, in conclusion, $u^{-1}L \in \mathcal{P}\big(\mathbf{V}, n^k\big)$.

We can prove that $Lu^{-1} \in \mathcal{P}\big(\mathbf{V}, n^k\big)$ in a very similar way.

Hence, $\mathcal{P}\big(\mathbf{V}, n^k\big)$ is closed under left and right quotients of L.

Closure under inverses of *lm*-morphisms. Let $\varphi\colon \Gamma^* \to \Sigma^*$ be an *lm*-morphism from $(\Gamma^*, \cdot)$ to $(\Sigma^*, \cdot)$ where Γ is some alphabet. We are going to define a program-reduction from $\varphi^{-1}(L)$ to L. For this, if we denote by $l \in \mathbb{N}$ the non-negative integer such that $\varphi(\Gamma) \subseteq \Sigma^l$, for each $i \in [l]$, we define $f_i \in \Sigma^\Gamma$ by $f_i(a) = \varphi(a)_i$ for all $a \in \Sigma$. Let $n \in \mathbb{N}$; we define the Σ-program Ψ_n on Γ^n as

$$\Psi_n = (1, f_1) \cdots (1, f_l)(2, f_1) \cdots (2, f_l) \cdot \cdots \cdot (n, f_1) \cdots (n, f_l) \ .$$

It is direct to see that for all $w \in \Gamma^n$, $\Psi_n(w) = \varphi(w)$, so that $(\varphi^{-1}(L))^{=n} = \{\varphi(w) \in L \mid w \in \Gamma^n\} = \Psi_n^{-1}(L^{=l\cdot n})$. Moreover, $|\Psi_n| = l \cdot n$.

Therefore, $(\Psi_n)_{n\in\mathbb{N}}$ is a program-reduction from $\varphi^{-1}(L)$ to L of length $l \cdot n$. This means, by Proposition 3.3.12, that $\varphi^{-1}(L) \in \mathcal{P}\big(\mathbf{V}, (l \cdot n)^k\big)$. But we have $(l \cdot n)^k = l^k \cdot n^k$ for all $n \in \mathbb{N}$, so that, in conclusion, $\varphi^{-1}(L) \in \mathcal{P}\big(\mathbf{V}, n^k\big)$.

Hence, $\mathcal{P}\big(\mathbf{V}, n^k\big)$ is closed under inverse images of L through *lm*-morphisms.

In conclusion, since this holds for any language L over some alphabet Σ, we proved the proposition. □

We end this section with a definition, that is folklore, that will be useful when doing syntactical manipulations of programs.

Definition 3.3.14. Let Σ be an alphabet and $(M, *)$ be a finite monoid. For two programs P and P' over $(M, *)$ on Σ^n for some $n \in \mathbb{N}$, we shall say that P' is a *subprogram*, a *prefix*

or a *suffix* of P if and only if it is, respectively, a subword, a prefix or a suffix of P as a word on $[n] \times M^{\Sigma}$. We shall also say that P and P' are *equivalent* if and only if any language $L \subseteq \Sigma^n$ recognised by P is also recognised by P' and vice versa.

3.4 Regular languages and programs

As we already mentioned briefly in the previous section, in McKenzie et al. [1991], McKenzie, Péladeau and Thérien uncovered a deep link between the computational power of polynomial-length programs over monoids taken from some variety of finite monoids $\mathbf{V}$ and the class of regular languages that are p-recognised by finite monoids in $\mathbf{V}$. And, in fact, Péladeau's notion of a p-variety of finite monoids Péladeau [1990] expresses, for a given variety of finite monoids $\mathbf{V}$, a certain notion of "well-behaviour" of p-recognition of regular languages by monoids of $\mathbf{V}$ (compared to classical morphism-recognition by monoids of $\mathbf{V}$) that suffices to restate almost all conjectures about the internal structure of NC^1 as the fact that all varieties of finite monoids $\mathbf{V}$ respectively corresponding to the concerned subclasses of NC^1 are p-varieties. In this section, we shall elaborate on that link and introduce a stronger notion of an *sp*-variety of finite monoids whose advantage over the notion of a p-variety of finite monoids is that when it is verified for a given variety of finite monoids $\mathbf{V}$, it directly entails an almost exact characterisation of the class of regular languages p-recognised by monoids of $\mathbf{V}$ (which happens to be exact in many cases).

3.4.1 First results and importance of regular languages in the realm of programs

The first obvious link between recognition by morphisms and recognition by programs is that for any finite monoid $(M, *)$, any language over some alphabet Σ recognised by $(M, *)$ in the classical, morphism-sense, is also p-recognised by $(M, *)$. This is formalised in the following proposition, that is straightforward to prove.

Proposition 3.4.1. *Let $(M, *)$ be a finite monoid and Σ an alphabet. Then, any language over Σ recognised by $(M, *)$ is a also recognised by a sequence of linear-length $(M, *)$-programs. More generally, for any variety of finite monoids $\mathbf{V}$, $\mathcal{L}(\mathbf{V})(\Sigma^*) \subseteq \mathcal{P}(\Sigma^*, \mathbf{V}, n)$ for any alphabet Σ, so that $\mathcal{L}(\mathbf{V}) \subseteq \mathcal{P}(\mathbf{V}, n)$.*

Moreover, the class of regular languages p-recognised by monoids taken from some variety of finite monoids is an *lm*-variety of languages. This is proved by combining

Propositions 3.3.8, 3.3.10 and 3.3.13 with the fact that the class of all regular languages $\mathcal{R}\mathsf{eg}$ is a variety of languages.

Proposition 3.4.2. *Let* $\mathbf{V}$ *be a variety of finite monoids. Then* $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg}$ *forms an* lm*-variety of languages.*

The concept of a program-reduction will be particularly useful in many cases in which it will be easier, at no extra cost in final program length, to give a program-reduction from some language L to some regular language in $\mathcal{L}(\mathbf{V})$ for some variety of finite monoids $\mathbf{V}$, rather than giving directly a sequence of programs over a finite monoid in $\mathbf{V}$ recognising L. This is stated formally in the following easy corollary of Proposition 3.3.12.

Corollary 3.4.3. *Let* Σ *and* Γ *be two alphabets. Let* $(M, *)$ *be a finite monoid. Given some language* K *over* Γ *that is recognised (classically) by* $(M, *)$ *and some other language* L *over* Σ *from which there exists a program-reduction to* K *of length* $s(n)$*, for* $s\colon \mathbb{N} \to \mathbb{N}$ *some function, we have that* $L \in \mathcal{P}((M, *), s(n))$*.*

This concept allows us to prove that for a wealth of varieties of finite monoids $\mathbf{V}$, we not only have that $\mathcal{P}(\mathbf{V})$ contains all of $\mathcal{L}(\mathbf{V})$, but also a good deal of other regular languages.

Let us say a stamp $\varphi\colon \Sigma^* \to M$ from the free monoid $(\Sigma^*, \cdot)$ generated by the alphabet Σ to a finite monoid $(M, *)$ is *cyclic* if and only if $\varphi(a) = \varphi(b)$ for all $a, b \in \Sigma$. Observe that, in this case, $(M, *)$ is necessarily generated by its unique element $m = \varphi(a)$ for any $a \in \Sigma$, and is hence what is called a *cyclic monoid*. Since a stamp that *lm*-divides a cyclic stamp is also cyclic and the stamp product of two cyclic stamps is cyclic as well (as is directly seen from the definition of these operations), we have that the class of all cyclic stamps is an *lm*-variety of stamps, that we will denote by **CYC**. Another example of an *lm*-variety of cyclic stamps is **MOD**, that is simply **CYC** where stamps are restricted to go into finite cyclic groups. The following result says that no matter what variety of finite monoids $\mathbf{V}$ we consider, programs over monoids in $\mathbf{V}$ can always decorate each of their input letters with the value of a counter given as the image through a cyclic stamp ψ of the word of letters preceding it (i.e. transform the input word w into $\sigma_\psi(w)$; see Subsection 3.2.3 for the definition of the sequential function σ_ψ); in particular, they can always handle the sort of "counting" introduced by the wreath product of a stamp into a monoid of $\mathbf{V}$ with some cyclic stamp when it comes to recognising regular languages, contrary to most cases of morphism-recognition.

Lemma 3.4.4. *Let* $\mathbf{V}$ *be a variety of finite monoids. Let* $\psi\colon \Sigma^* \to N$ *be a cyclic stamp from the free monoid* $(\Sigma^*, \cdot)$ *generated by some alphabet* Σ *to a finite monoid* $(N, \bot)$ *and*

let V be a language over $\Sigma \times N$ that belongs to $\mathcal{P}(\mathbf{V}, s(n))$ for $s\colon \mathbb{N} \to \mathbb{N}$ some function. Then, $\sigma_\psi^{-1}(V) \in \mathcal{P}(\mathbf{V}, s(n))$.

Proof. We know that $L = \sigma_\psi^{-1}(V)$ is a language over Σ.

We shall now define a program-reduction from $\sigma_\psi^{-1}(V)$ to V of linear length. Observe first that since ψ is a cyclic stamp, for all $n \in \mathbb{N}$, there exists $x_n \in N$ verifying $\psi(\Sigma^n) = \{x_n\}$. For all $n \in \mathbb{N}_{>0}$, we define $f_n \in (\Sigma \times N)^\Sigma$ by $f_n(a) = (a, x_{n-1})$ for all $a \in \Sigma$. Fix $n \in \mathbb{N}$. We define the $(\Sigma \times N)$-program Ψ_n on Σ^n as

$$\Psi_n = (1, f_1)(2, f_2)(3, f_3) \cdots (n, f_n) \ .$$

Then, for all $w \in \Sigma^n$, we have

$$\begin{aligned}
\Psi_n(w) &= f_1(w_1) f_2(w_2) f_3(w_3) \cdots f_n(w_n) \\
&= (w_1, x_0)(w_2, x_1)(w_3, x_2) \cdots (w_n, x_{n-1}) \\
&= (w_1, \psi(\varepsilon))(w_2, \psi(w_1))(w_3, \psi(w_1 w_2)) \cdots (w_n, \psi(w_1 w_2 \cdots w_{n-1})) \\
&= \sigma_\psi(w) \ .
\end{aligned}$$

This means that $\Psi_n = \sigma_\psi|_{\Sigma^n}$, so that, since $\sigma_\psi^{-1}((\Sigma \times N)^n) = \Sigma^n$, we have

$$L^{=n} = \sigma_\psi^{-1}(V) \cap \Sigma^n = \sigma_\psi^{-1}(V \cap (\Sigma \times N)^n) = (\sigma_\psi|_{\Sigma^n})^{-1}(V^{=n}) = \Psi_n^{-1}(V^{=n}) \ ,$$

as inverses of functions commute with intersections.

Therefore, $(\Psi_n)_{n \in \mathbb{N}}$ is a program-reduction from L to V of length n. Since there exists some monoid $(M, *) \in \mathbf{V}$ such that V belongs to $\mathcal{P}((M, *), s(n))$, it follows by Proposition 3.3.12 that $\sigma_\psi^{-1}(V) = L \in \mathcal{P}(\mathbf{V}, s(n))$. □

Proposition 3.4.5. *For any variety of finite monoids $\mathbf{V}$ and any* lm*-variety of cyclic stamps $\mathbf{W}$, we have $\mathcal{L}(\langle \mathbf{V} \rangle_{\text{all}} * \mathbf{W}) \subseteq \mathcal{P}(\mathbf{V}, n)$.*

Proof. Let $\mathbf{V}$ be a variety of finite monoids and $\mathbf{W}$ an *lm*-variety of cyclic stamps.

Fix some alphabet Σ.

Let $L \in \mathcal{L}(\mathbf{W})(\Sigma^*)$. Let $\eta\colon \Sigma^* \to M$ be its syntactic morphism, where $(M, *)$ is its syntactic monoid and let $F \subseteq M$ be such that $L = \eta^{-1}(F)$.

By definition of $\mathcal{L}(\mathbf{W})$, η is a stamp of $\mathbf{W}$; in particular, it is a cyclic stamp. This means that for all $n \in \mathbb{N}$, there exists $m_n \in M$ verifying $\eta(\Sigma^n) = \{m_n\}$, so that either $L^{=n} = \Sigma^n$ when $m_n \in F$ or $L^{=n} = \emptyset$ when $m_n \notin F$.

Let now $(N, \perp)$ be any finite monoid in $\mathbf{V}$ (that contains at least all trivial monoids by definition). For each $n \in \mathbb{N}$, we define $P_n = \varepsilon$ to be the empty $(N, \perp)$-program on Σ^n; for all $n \in \mathbb{N}$, we then have that $P_n(\Sigma^n) = \{1_{(N,\perp)}\}$, so that $L^{=n} = P_n^{-1}(1_{(N,\perp)})$ when $m_n \in F$ and $L^{=n} = P_n^{-1}(\emptyset)$ when $m_n \notin F$. Hence, $(P_n)_{n\in\mathbb{N}}$ recognises L and as $|P_n| = 0$ for all $n \in \mathbb{N}$, we have $L \in \mathcal{P}(\mathbf{V}, n)$.

Hence, since this is true for any $L \in \mathcal{L}(\mathbf{W})(\Sigma^*)$, we have that $\mathcal{L}(\mathbf{W})(\Sigma^*) \subseteq \mathcal{P}(\mathbf{V}, n)$.

Let now $\psi\colon \Sigma^* \to N$ be some stamp in $\mathbf{W}$ from $(\Sigma^*, \cdot)$ to a finite monoid $(N, \perp)$ and $V \in \mathcal{L}(\langle\mathbf{V}\rangle_{all})((\Sigma \times N)^*) = \mathcal{L}(\mathbf{V})((\Sigma \times N)^*)$. This means that $V \in \mathcal{P}(\mathbf{V}, n)$ by Proposition 3.4.1, so that by Lemma 3.4.4 we have that $\sigma_\psi^{-1}(V) \in \mathcal{P}(\mathbf{V}, n)$.

Now since $\mathcal{P}(\mathbf{V}, n)$ contains all languages of $\mathcal{L}(\mathbf{W})(\Sigma^*)$ and all languages of the form $\sigma_\psi^{-1}(V)$, where $\psi\colon \Sigma^* \to N$ is some stamp in $\mathbf{W}$ from $(\Sigma^*, \cdot)$ to a finite monoid $(N, \perp)$ and $V \in \mathcal{L}(\langle\mathbf{V}\rangle_{all})((\Sigma \times N)^*)$, and since $\mathcal{P}(\mathbf{V}, n)$ is closed under finite Boolean combinations by Proposition 3.3.10, it follows by Theorem 3.2.15 that $\mathcal{L}(\langle\mathbf{V}\rangle_{all} * \mathbf{W})(\Sigma^*) \subseteq \mathcal{P}(\mathbf{V}, n)$.

In conclusion, as this is true for any alphabet Σ, we have $\mathcal{L}(\langle\mathbf{V}\rangle_{all} * \mathbf{W}) \subseteq \mathcal{P}(\mathbf{V}, n)$. □

But why be interested at all in the class of regular languages p-recognised by monoids taken from some variety of finite monoids $\mathbf{V}$ beyond the quest for understanding this class for its own sake? The work of McKenzie, Péladeau and Thérien McKenzie et al. [1991] showed that any class of languages p-recognised by a monoid belonging to a variety of finite monoids is uniquely characterised by the class of regular languages contained in the variety, i.e. for all varieties of finite monoids $\mathbf{V}$ and $\mathbf{W}$, we have $\mathcal{P}(\mathbf{V}) = \mathcal{P}(\mathbf{W})$ if and only if $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg} = \mathcal{P}(\mathbf{W}) \cap \mathcal{R}\mathsf{eg}$. So this means in particular that determining exactly what regular languages are in $\mathcal{P}(\mathbf{V})$ is a fundamental problem that is often at least as difficult as proving a separation between $\mathcal{P}(\mathbf{V})$ and some other class $\mathcal{P}(\mathbf{W})$.

Barrington, Compton, Straubing and Thérien Barrington et al. [1992] exactly characterised the regular languages over $\{0, 1\}$ in AC^0 and $\mathsf{ACC}^0[p]$ for all $p \in \mathbb{N}$ prime, building on the at the time recent constant-depth circuits size lower bound results. Let $L_1 = \mathrm{MOD}_2$ and $L_2 = \big((0 + 1)^2\big)^*(0 + 1)$ be two languages in $\{0, 1\}^*$. On the one hand, thanks to the result of Furst, Saxe and Sipser Furst et al. [1984] and, independently, Ajtai Ajtai [1983] (Theorem 1.3.7), we know that $L_1 \notin \mathsf{AC}^0$, while it is easy to see that $L_2 \in \mathsf{AC}^0$. On the other hand, $\mathrm{M}(L_1)$ and $\mathrm{M}(L_2)$ are isomorphic and if we define $\varphi\colon \{0, 1\}^* \to \{0, 1\}^*$ as the unique monoid endomorphism of $(\{0, 1\}^*, \cdot)$ such that $\varphi(0) = 11$ and $\varphi(1) = 1$, we have that $L_1 = \varphi^{-1}(L_2)$. So this implies that membership of a regular language in the class $\mathsf{AC}^0 \cap \mathcal{R}\mathsf{eg}$ does not solely depend on its *syntactic monoid* and that this class is not a variety of languages, since it isn't closed under inverses of monoid morphisms. But

the authors of Barrington et al. [1992] managed to characterise that class in terms of the *syntactic morphisms* of the regular languages it contains.

This prompted Straubing Straubing [2002] to define more general notions of varieties of languages, each corresponding to a certain notion of varieties of stamps, namely the notions of $\mathcal{C}$-varieties of languages and associated $\mathcal{C}$-varieties of stamps. Those notions of varieties are interesting because they are parameterised by the class $\mathcal{C}$ of morphisms between finitely generated free monoids that on one hand go through when considering the property of stability under taking inverses of morphisms in the language varieties, and on the other hand adapt the notion of division considered for the property of stability under stamp division in the stamp varieties. And this is precisely what is needed for the study of $\mathsf{AC}^0 \cap \mathcal{R}\mathsf{eg}$, because while $\mathsf{AC}^0 \cap \mathcal{R}\mathsf{eg}$ is not closed under taking inverses of every morphism between finitely generated free monoids, it is closed under taking inverses of any such morphism which is length-multiplying.

Let $\varphi\colon \Sigma^* \to M$ be some stamp from the free monoid $(\Sigma^*, \cdot)$ generated by some alphabet Σ to some finite monoid $(M, *)$. Then, since M is finite, there exists $s \in \mathbb{N}_{>0}$ such that $\varphi(\Sigma^s) = \varphi(\Sigma^{2s})$. We call the semigroup $(\varphi(\Sigma^s), *|_{\varphi(\Sigma^s)})$ for the smallest such s the *stable semigroup of* φ and $\left(\varphi((\Sigma^s)^*), *|_{\varphi((\Sigma^s)^*)}\right)$ its *stable monoid*. Given $\mathbf{V}$ some variety of finite semigroups (monoids), we will denote by $\mathbf{QV}$ the class of *quasi-*$\mathbf{V}$ *stamps*, which is the class of all stamps whose stable semigroup (monoid) is in $\mathbf{V}$; it is an *lm*-variety of stamps. (See Pin and Straubing [2005] and Dartois and Paperman [2014] for further details.)

Why define all this? It is because the class of regular languages $\mathsf{AC}^0 \cap \mathcal{R}\mathsf{eg}$, as shown in Barrington et al. [1992] for the case of languages over $\{0, 1\}$ and as we will prove in the next subsection, is exactly the *lm*-variety of languages whose syntactic morphisms are quasi-aperiodic, i.e. belong to $\mathbf{QA}$.[5] That is to say, $\mathcal{P}(\mathbf{A}) \cap \mathcal{R}\mathsf{eg} = \mathsf{AC}^0 \cap \mathcal{R}\mathsf{eg} = \mathcal{L}(\mathbf{QA})$. Basically, this result means that, over finite aperiodic monoids, p-recognition does not allow recognising many more regular languages than classical morphism-recognition: in fact, as we shall see in the next subsection, we have $\langle \mathbf{A} \rangle_{all} * \mathbf{MOD} = \mathbf{QA}$, so that p-recognition over monoids in $\mathbf{A}$ informally only gives the ability to "count modulo some

[5]The way quasi-aperiodicity of some stamp $\eta\colon \Sigma^* \to M$ from the free monoid $(\Sigma^*, \cdot)$ generated by an alphabet Σ to some finite monoid $(M, *)$ is defined in Barrington et al. [1992] is as follows: for each $t \in \mathbb{N}$, $\eta(\Sigma^t)$ contains no set S such that $(S, *|_S)^1$ is a non-aperiodic monoid. The class of all stamps verifying this condition happens to be equal to $\mathbf{QA}$, but it seems like, in general, it is not necessarily the case that the class of all stamps $\eta\colon \Sigma^* \to M$ verifying that for each $t \in \mathbb{N}$, $\eta(\Sigma^t)$ contains no set S such that $(S, *|_S)^1$ does not belong to $\mathbf{V}$ is equal to $\mathbf{QV}$. In fact, this first class does not even seem to always be an *lm*-varieties of stamps. The standard definition of quasi-$\mathbf{V}$ stamps we use in this thesis was apparently introduced by Straubing in Straubing [2002], about 10 years after the publication of Barrington et al.'s article.

positive integer" in addition to what can be done through morphism-recognition over monoids in **A**. To study more systematically the varieties of finite monoids that are similarly "well-behaved" with respect to regular language program-recognition, we have to take a step back and try to characterise and understand the general properties these varieties do share.

3.4.2 Tame varieties of finite monoids

For this we introduce the notion of an *sp*-variety of finite monoids. This notion is inspired by the notion of *p*-varieties (program-varieties) of finite monoids originally defined by Péladeau in his Ph.D. thesis Péladeau [1990]. The notion was used later by Straubing in Straubing [2000, 2001] (noting that Straubing's notion of *p*-recognition by a finite monoid is strictly stronger), as well as by Lautemann, Tesson and Thérien in Lautemann et al. [2006] and by Tesson alone in his own Ph.D. thesis Tesson [2003]. Furthermore, the notion of a *p*-variety has also been defined for varieties of finite semigroups by Péladeau, Straubing and Thérien in Péladeau et al. [1997] (using an adequate notion of *programs over semigroups*, that we won't define in this thesis) who obtained results similar to ours for varieties of finite semigroups of the form $\mathbf{V} * \mathbf{LI}$ (equal to $\mathbf{V} * \mathbf{D}$ when $\mathbf{V}$ is a non-trivial variety of finite monoids Straubing [1985]).

Let us first define the notion of the set of word problems over a given stamp, finite semigroup or monoid.

Definition 3.4.6 (Following McKenzie et al. [1991])**.** Let $\varphi\colon \Sigma^* \to M$ be a stamp from the free monoid $(\Sigma^*, \cdot)$ generated by some alphabet Σ to a finite monoid $(M, *)$. We denote by $\mathcal{W}(\varphi)$ the *set of word problems over* φ, made of all languages L over Σ such that $L = \varphi^{-1}(F)$ for some subset F of M, a *word problem over* φ. Given a finite semigroup (monoid) $(S, *)$, the *set of word problems over* $(S, *)$, denoted by $\mathcal{W}((S, *))$, is simply $\mathcal{W}(\eta_{(S,*)})$, the set of word problems over the evaluation morphism $\eta_{(S,*)}$ of $(S, *)$.

Intuitively, when the set of word problems over some finite semigroup $(S, *)$ satisfies the property that each language in the set is *p*-recognised by a monoid taken from some fixed variety of finite monoids $\mathbf{V}$, it means that polynomial-length programs over monoids in $\mathbf{V}$ can *simulate* the multiplication in $(S, *)$.

The first obvious thing to observe is that, for any variety of finite monoids $\mathbf{V}$ and for any monoid $(M, *) \in \mathbf{V}$, we have that $\mathcal{W}((M, *)) \subseteq \mathcal{L}(\mathbf{V})$, so that, by Proposition 3.4.1, it follows that $\mathcal{W}((M, *)) \subseteq \mathcal{P}(\mathbf{V}, n)$. For the same reasons, any finite semigroup $(S, \perp)$ verifying that $(S, \perp)^1 \in \mathbf{V}$ also has its set of word problems contained in $\mathcal{P}(\mathbf{V}, n)$. The

notion of tameness we introduce requires a converse to the latter implication for finite semigroups.

Definition 3.4.7. An sp-*variety of finite monoids*, which we shall also call a *tame* such variety, is a variety $\mathbf{V}$ of finite monoids such that for any finite semigroup $(S, *)$, if $\mathcal{W}((S, *)) \subseteq \mathcal{P}(\mathbf{V})$ then $(S, *)^1 \in \mathbf{V}$.

The notion of a *p*-variety of finite monoids in Lautemann et al. [2006] is similar to our notion of an *sp*-variety, but the former only requires that any finite *monoid* $(M, *)$ verifying $\mathcal{W}((M, *)) \subseteq \mathcal{P}(\mathbf{V})$ must in fact belong to $\mathbf{V}$. Given some alphabet Σ, a language L over Σ has a neutral letter if and only if there exists $e \in \Sigma$ verifying that for all $u, v \in \Sigma^*$, $uv \in L \Leftrightarrow uev \in L$; an equivalent definition of a *p*-variety of finite monoids is that it is any variety of finite monoids $\mathbf{V}$ such that every regular language with a neutral letter in $\mathcal{P}(\mathbf{V})$ is in $\mathcal{L}(\mathbf{V})$. The notion of *p*-variety thus inherently corresponds to the kind of notion of "well-behaviour" we are looking for, but only for regular languages with a neutral letter; it does in fact not necessarily imply "well-behaviour" for *p*-recognition of languages without a neutral letter, as we shall see later in this subsection. This is why our notion of tameness is stronger, as any *sp*-variety of finite monoids is also a *p*-variety of finite monoids, but the converse is not always true. For instance, $\mathbf{J}$ is a *p*-variety of finite monoids Tesson [2003], but our discussion at the end of this subsection shows that $\mathbf{J}$ is not an *sp*-variety of finite monoids and in fact does not "behave well".

The intuition behind this definition is that when a variety of finite monoids $\mathbf{V}$ is tame, it cannot simulate multiplication in any finite semigroup $(S, *)$ that does not belong to $\mathbf{V}$ after adjoining, when necessary, an identity to make a monoid out of it. The next proposition nails down the "well behaviour", i.e., the limited ability for *p*-recognising regular languages, that the tameness of $\mathbf{V}$ entails.

Proposition 3.4.8. *Let* $\mathbf{V}$ *be an* sp-*variety of finite monoids. Then* $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{QV})$.

A similar result was proven for varieties of finite semigroups of the form $\mathbf{V} * \mathbf{LI}$ in Péladeau et al. [1997], where languages are considered to be subsets of Σ^+ for some alphabet Σ. Our proof follows the same lines.

Proof. Let L be a regular language in $\mathcal{P}((M, *))$ for some finite monoid $(M, *) \in \mathbf{V}$. Let $\mathrm{M}(L)$ be the syntactic monoid of L and η_L its syntactic morphism. Let $(S, \perp)$ be the stable semigroup of η_L, in particular $S = \eta_L(\Sigma^k)$ for some $k \in \mathbb{N}_{>0}$. We wish to show that $(S, \perp)^1$, which is isomorphic to the submonoid of $\mathrm{M}(L)$ of underlying subset $\eta_L((\Sigma^k)^*)$, is in $\mathbf{V}$.

We show that $\mathcal{W}((S,\perp)) \subseteq \mathcal{P}(\mathbf{V})$ and conclude from the fact that $\mathbf{V}$ is an *sp*-variety of finite monoids that $(S,\perp)^1 \in \mathbf{V}$ as desired. Let $\eta_{(S,\perp)}\colon S^* \to S^1$ be the evaluation morphism of $(S,\perp)$. For each $m \in S$, consider $L'_m = \eta^{-1}_{(S,\perp)}(m)$; we wish to show that $L'_m \in \mathcal{P}(\mathbf{V})$ for all $m \in S$. Since additionally when $(S,\perp)$ is not a monoid, we have $\eta^{-1}_{(S,\perp)}(1) = \{\varepsilon\}$ that is easily seen to belong to $\mathcal{P}(\mathbf{V})$, this implies that $\mathcal{W}((S,\perp)) \subseteq \mathcal{P}(\mathbf{V})$ by closure under union of $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg}$, Proposition 3.4.2.

Fix $m \in S$. Let $L''_m = \eta_L^{-1}(m)$. Since m belongs to the syntactic monoid of L and η_L is the syntactic morphism of L, a classical algebraic argument [Pin, 1986, Chapter 2, proof of Lemma 2.6] shows that L''_m is a Boolean combination of quotients of L or their complements. By Proposition 3.4.2, we conclude that $L''_m \in \mathcal{P}(\mathbf{V})$.

By definition of $(S,\perp)$, for any element s of S there is a word u_s of length k such that $\eta_L(u_s) = s$. Notice that this is precisely where we need to work with $(S,\perp)$ and not $(S,\perp)^1$.

Let $f\colon S^* \to \Sigma^*$ be the unique *lm*-morphism from $(S^*,\cdot)$ to $(\Sigma^*,\cdot)$ sending s to u_s and notice that $L'_m = f^{-1}(L''_m)$. The result follows by closure of $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg}$ under inverse images of *lm*-morphisms, Proposition 3.4.2. □

Dartois and Paperman showed [Dartois and Paperman, 2014, Corollary 18] (see also Dartois [2014], Paperman [2014]) that, while $\langle \mathbf{V} \rangle_{all} * \mathbf{MOD} \subseteq \mathbf{QV}$ for any variety of finite monoids $\mathbf{V}$, locality of $\mathbf{V}$ implies that equality holds. This allows us to give an exact characterisation of the *lm*-variety of regular languages *p*-recognised by monoids taken from some local *sp*-variety of finite monoids.

Proposition 3.4.9. *Let* $\mathbf{V}$ *be a local* sp*-variety of finite monoids. Then* $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg} = \mathcal{L}(\mathbf{QV})$.

Proof. By [Dartois and Paperman, 2014, Corollary 18], as $\mathbf{V}$ is local, we have $\mathbf{QV} = \langle \mathbf{V} \rangle_{all} * \mathbf{MOD}$. Moreover, we know that $\mathbf{MOD}$ is an *lm*-variety of cyclic stamps, so by Proposition 3.4.5, we have that $\mathcal{L}(\mathbf{QV}) \subseteq \mathcal{P}(\mathbf{V})$. We can conclude applying Proposition 3.4.8, as $\mathbf{V}$ is tame. □

We don't know whether it is always true that for non-local *sp*-varieties of finite monoids $\mathbf{V}$, $\mathcal{L}(\mathbf{QV})$ is contained in $\mathcal{P}(\mathbf{V})$.

An example of an *sp*-variety of finite monoids is the class of finite aperiodic monoids $\mathbf{A}$. This is a consequence of the celebrated result of Furst, Saxe and Sipser, and, independently, Ajtai, that for any integer $m \in \mathbb{N}, m \geq 2$, the language MOD_m of words over $\{0,1\}$ that contain a number of 1's not congruent to 0 modulo m is not in AC^0 (Theorem 1.3.7).

Proposition 3.4.10. $\mathbf{A}$ *is an* sp*-variety of finite monoids.*

Proof. Towards a contradiction, assume there would exist a finite semigroup $(S, *)$ such that $(S, *)^1$ is not aperiodic but still $\mathcal{W}((S, *)) \subseteq \mathcal{P}(\mathbf{A})$.

Then there is an x in S such that $x^{*,\omega} \neq x^{*,\omega+1}$ where $\omega \in \mathbb{N}_{>0}$ is the idempotent power of $(S, *)^1$. Consider the morphism $\mu\colon \{0,1\}^* \to S^*$ from the free monoid $(\{0,1\}^*, \cdot)$ to the free monoid $(S^*, \cdot)$ sending 1 to $x^{*,\omega+1}$ and 0 to $x^{*,\omega}$: it is an *lm*-morphism (even an *lp*-morphism). Let us now consider the language $(\eta_{(S,*)} \circ \mu)^{-1}(S^1 \setminus \{x^{*,\omega}\})$ where $\eta_{(S,*)}\colon S^* \to S^1$ is the evaluation morphism of $(S, *)$: it is easy to see that it is $\mathrm{MOD}_{\mathrm{m}}$, the language of all words over $\{0,1\}$ with a number of 1's not congruent to 0 modulo m, where $m \in \mathbb{N}, m \geq 2$ is the smallest positive integer such that $x^{*,\omega+m} = x^{*,\omega}$, that cannot be 1 because $x^{*,\omega} \neq x^{*,\omega+1}$.

From the hypothesis that $\mathcal{W}((S, *)) \subseteq \mathcal{P}(\mathbf{A})$, it follows that $\eta^{-1}_{(S,*)}(S^1 \setminus \{x^{*,\omega}\}) \in \mathcal{P}(\mathbf{A})$, hence since μ is an *lm*-morphism and $\mathcal{P}(\mathbf{A}) \cap \mathcal{R}\mathsf{eg}$ is closed under inverses of *lm*-morphisms by Proposition 3.4.2, we have $\mathrm{MOD}_{\mathrm{m}} = \mu^{-1}(\eta^{-1}_{(S,*)}(S^1 \setminus \{x^{*,\omega}\})) \in \mathcal{P}(\mathbf{A})$: a contradiction to Theorem 1.3.7. □

As $\mathbf{A}$ is local [Tilson, 1987, Example 15.5] and an *sp*-variety, it follows from Proposition 3.4.9 that the regular languages in $\mathcal{P}(\mathbf{A})$, hence in AC^0, are precisely those in $\mathcal{L}(\mathbf{QA})$. (This has originally been proven in Barrington et al. [1992] for the case of regular languages over $\{0,1\}$.)

As partially observed by Tesson Tesson [2003], we can prove that for some variety of finite monoids $\mathbf{V}$ to be a *p*-variety of finite monoids it is necessary and sufficient that for any variety of finite monoids $\mathbf{W}$ not contained in $\mathbf{V}$, $\mathcal{P}(\mathbf{V}) \neq \mathcal{P}(\mathbf{W})$ — put differently, $\mathbf{V}$ is maximal in the sense that it is the unique inclusion-wise biggest variety of finite monoids $\mathbf{W}$ verifying $\mathcal{P}(\mathbf{V}) = \mathcal{P}(\mathbf{W})$. This means that proving $\mathbf{V}$ to be a *p*-variety of finite monoids is equivalent to giving an optimal separation result of $\mathcal{P}(\mathbf{V})$ from all other complexity classes defined using *p*-recognition by monoids taken from some variety of finite monoids not contained in $\mathbf{V}$. For the case of *sp*-varieties of finite monoids, we can only prove that this optimal separation result is a necessary condition.

Proposition 3.4.11. *Let* $\mathbf{V}$ *be an* sp*-variety of finite monoids. Then, for any variety of finite monoids* $\mathbf{W}$ *such that* $\mathbf{W} \not\subseteq \mathbf{V}$*, we have* $\mathcal{P}(\mathbf{V}) \neq \mathcal{P}(\mathbf{W})$*.*

Proof. Let $\mathbf{W}$ be a variety of finite monoids such that $\mathbf{W} \not\subseteq \mathbf{V}$. This means that there exists a finite monoid $(M, *) \in \mathbf{W} \setminus \mathbf{V}$. Since $\mathbf{V}$ is an *sp*-variety of finite monoids, we necessarily have $\mathcal{W}((M, *)) \not\subseteq \mathcal{P}(\mathbf{V})$, while $\mathcal{W}((M, *)) \subseteq \mathcal{P}(\mathbf{W})$. Therefore, $\mathcal{P}(\mathbf{V}) \neq \mathcal{P}(\mathbf{W})$. □

The converse statement is false for *sp*-varieties of finite monoids, and we will for instance see that this optimal separation result holds for $\mathbf{J}$ which isn't tame. This converse statement fails precisely because of the ability for monoids taken from some variety of finite monoids $\mathbf{V}$ to *p*-recognise any language recognised by the wreath product of a stamp into a monoid of $\mathbf{V}$ with some cyclic stamp, Proposition 3.4.5. This ability allows, in particular, for all interesting cases of varieties of finite monoids $\mathbf{V}$, to *p*-recognise the regular language of words over $\{a, b\}$ starting with an a.

Lemma 3.4.12. *For any non-trivial variety of finite monoids* $\mathbf{V}$*, we have* $a(a+b)^* \in \mathcal{P}(\mathbf{V})$.

Proof. Let $\mathbf{V}$ be a non-trivial variety of finite monoids. Let $(M, *)$ be a monoid from $\mathbf{V}$ of order at least 2.

Consider the finite monoid $(\mathbb{Z}/2\,\mathbb{Z}, \times)$ where $\times$ denotes the canonical product modulo 2. Let us define the cyclic stamp $\psi\colon \{a, b\}^* \to \mathbb{Z}/2\,\mathbb{Z}$ from the free monoid $(\{a, b\}^*, \cdot)$ to $(\mathbb{Z}/2\,\mathbb{Z}, \times)$ as the unique monoid morphism from $(\{a, b\}^*, \cdot)$ to $(\mathbb{Z}/2\,\mathbb{Z}, \times)$ verifying $\psi(a) = \psi(b) = 0$. Then, it is direct to see that $\sigma_\psi(\varepsilon) = \varepsilon$ and $\sigma_\psi(w_1w_2w_3\cdots w_n) = (w_1, 1)(w_2, 0)(w_3, 0)\cdots(w_n, 0)$ for all $n \in \mathbb{N}_{>0}$, $w \in \Sigma^n$.

Let $m \in M$ be an element of M different from the identity $1_{(M,*)}$ and define the monoid morphism $\varphi\colon (\{a, b\} \times \mathbb{Z}/2\,\mathbb{Z})^* \to M$ from the free monoid $((\{a, b\} \times \mathbb{Z}/2\,\mathbb{Z})^*, \cdot)$ to $(M, *)$ as the unique one satisfying $\varphi(a, 1) = m$ and $\varphi(a, 0) = \varphi(b, 1) = \varphi(b, 0) = 1_{(M,*)}$. Let $V = \varphi^{-1}(m)$. By definition, V does belong to $\mathcal{L}(\mathbf{V})((\{a, b\} \times \mathbb{Z}/2\,\mathbb{Z})^*) = \mathcal{L}(\langle\mathbf{V}\rangle_{all})((\{a, b\} \times \mathbb{Z}/2\,\mathbb{Z})^*)$ and since $\psi \in \mathbf{CYC}$, it follows that $\sigma_\psi^{-1}(V)$ is a language of $\mathcal{L}(\langle\mathbf{V}\rangle_{all} * \mathbf{CYC})(\{a, b\}^*)$ by Theorem 3.2.15.

It is easy to see that V does contain all words over $\{a, b\} \times \mathbb{Z}/2\,\mathbb{Z}$ that have one unique letter $(a, 1)$ but does not contain any of the words over that same alphabet that has no letter $(a, 1)$, so, since for all $w \in \{a, b\}^*$, $\sigma_\psi(w)$ does contain one unique letter $(a, 1)$ when w is of length at least 1 and starts with the letter a, and doesn't contain any letter $(a, 1)$ otherwise, we have that $\sigma_\psi^{-1}(V) = a(a+b)^*$.

In conclusion, as Proposition 3.4.5 tells us that $\mathcal{L}(\langle\mathbf{V}\rangle_{all} * \mathbf{CYC}) \subseteq \mathcal{P}(\mathbf{V})$, we have $a(a+b)^* \in \mathcal{P}(\mathbf{V})$, which ends the proof of the proposition. □

It is straightforward to see that $a(a+b)^*$ has the property that the stable monoid of its syntactic morphism is equal to its syntactic monoid, which implies the following using the contrapositive of Proposition 3.4.8.

Lemma 3.4.13. *If* $\mathbf{V}$ *is a non-trivial* sp*-variety of finite monoids, then* $\mathrm{M}(a(a+b)^*) \in \mathbf{V}$.

But the fact is exactly that there are p-varieties of finite monoids, like $\mathbf{J}$ Tesson [2003], that do not contain $\mathrm{M}(a(a+b)^*)$. Thus, we have that $\mathcal{P}(\mathbf{J}) \neq \mathcal{P}(\mathbf{W})$ for any variety of finite monoids $\mathbf{W}$ such that $\mathbf{W} \not\subseteq \mathbf{J}$ while $\mathbf{J}$ is not an sp-variety of finite monoids. This shows that moving from the notion of a p-variety of finite monoids to the stronger notion of an sp-variety of finite monoids, we lose the equivalence with the optimal separation result of Proposition 3.4.11.

That being said, this example also makes clear why we need to replace the notion of a p-variety of finite monoids by the stronger notion of an sp-variety of finite monoids to be able to guarantee "well-behaviour" of p-recognition of regular languages as stated by Proposition 3.4.8: indeed, while $\mathbf{J}$ is a p-variety of finite monoids, we just showed that $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg} \not\subseteq \mathcal{L}(\mathbf{QJ})$.

Taking stock, there are two main motivations for an exhaustive study of exactly which varieties of finite monoids are tame:

- because proving the tameness of a variety of finite monoids $\mathbf{V}$ is a big step towards the exact characterisation of $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg}$, the lm-variety of regular languages p-recognised by monoids in $\mathbf{V}$;
- because proving the tameness of a variety of finite monoids $\mathbf{V}$ entails separation of $\mathcal{P}(\mathbf{V})$ from every complexity class of the form $\mathcal{P}(\mathbf{W})$ where $\mathbf{W}$ is a variety of finite monoids not contained in $\mathbf{V}$.

To complete such a study would thus be of great interest for computational complexity theory in general, because many of the conjectures about the internal structure of NC^1 would be proven by showing some appropriate variety of finite monoids is tame. This for instance is the case for the conjectured strict containment in NC^1 of the class ACC^0 (corresponding to p-recognition by monoids from $\mathbf{M}_{\mathrm{sol}}$) Lautemann et al. [2006], the current frontier of knowledge for lower bounds in computational complexity theory.

In the two next chapters of this thesis, we are pursuing a much humbler objective: to start this study for two classical "small" varieties of finite aperiodic monoids, $\mathbf{DA}$ and $\mathbf{J}$, along with some other investigations about p-recognition by monoids drawn from these.

Chapter 4

The power of programs over monoids in DA

In this chapter we focus on p-recognition by monoids taken from the variety of finite monoids **DA**. **DA** is a well-known "small" variety of finite aperiodic monoids whose importance in algebraic automata theory and connections with other fields is well established (see Tesson and Thérien [2002] for an eloquent testimony).

We start off with some specific preliminaries about **DA** in Section 4.1. The main result of this chapter, the tameness of **DA**, is proven in Section 4.2; as a consequence, we obtain that $\mathcal{P}(\mathbf{DA}) \cap \mathcal{R}\mathsf{eg}$ is exactly the lm-variety of languages $\mathcal{L}(\mathbf{QDA})$. Finally, we investigate Tesson and Thérien's polynomial-length property for monoids in **DA** Tesson and Thérien [2001] in Section 4.3, where we give a fine hierarchy inside $\mathcal{P}(\mathbf{DA})$ for some parameterisation of **DA**, refining Tesson and Thérien's result.

The content of this chapter is based on the article Grosshans et al. [2017].

4.1 Specific preliminaries about DA

As we have seen in Section 3.1 of Chapter 3, **DA** is a strict subvariety of the variety of finite aperiodic monoids **A**.[1] Its equational characterisation is the following: a finite monoid $(M, *)$ of idempotent power ω belongs to **DA** if and only if $(xy)^\omega = (xy)^\omega x (xy)^\omega$ for all $x, y \in M$.

The property we now state is crucial to many proofs involving **DA** and our work will rely on this property as well.

[1] Is is, in fact, included in the lowest levels of the so-called *dot-depth hierarchy* inside **A** (see Pin [2017]).

Lemma 4.1.1. *[Folklore] Let $(M, *)$ be some monoid in* **DA**. *For all $u, u', r \in M$, we have that $u \mathfrak{R} u'$ and $ur \mathfrak{R} u$ imply $u'r \mathfrak{R} u$. Similarly $u \mathfrak{L} u'$ and $ru \mathfrak{L} u$ imply $ru' \mathfrak{L} u$.*

Informally, this lemma states that, given a monoid $(M, *) \in \mathbf{DA}$, for any $u \in M$, whether ur for some $r \in M$ is strictly smaller than u according to the $\leq_{\mathfrak{R}}$ preorder (i.e., whether right-multiplication by r changes to a lower $\mathfrak{R}$-class) only depends on the $\mathfrak{R}$-class of u and not on u itself.

4.1.1 Characterisation of $\mathcal{L}(\mathbf{DA})$

The variety of languages $\mathcal{L}(\mathbf{DA})$ was first characterised by Schützenberger Schützenberger [1976] using so-called unambiguous polynomials. Given an alphabet Σ, an *unambiguous monomial over* Σ is a language L over Σ of the form $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ for some $k \in \mathbb{N}$, where $A_0, A_1, \ldots, A_{k-1}, A_k$ are subsets of Σ and $a_1, a_2, \ldots, a_k$ are letters of Σ, verifying that for each $w \in L$, there exists a unique decomposition of the form $w = w_0 a_1 w_1 a_2 \cdots w_{k-1} a_k w_k$ where $w_i \in A_i^*$ for each $i \in [\![0, k]\!]$. An *unambiguous polynomial over* Σ is then simply a (finite) disjoint union of unambiguous monomials over Σ. Schützenberger's result can then be stated as follows.

Theorem 4.1.2 (Schützenberger [1976])**.** *For any alphabet Σ, $\mathcal{L}(\mathbf{DA})(\Sigma^*)$ is exactly the set of all unambiguous polynomials over Σ.*

Following Gavaldà and Thérien [2003], we use an alternative definition of the languages recognised by a monoid in **DA**. We define by induction a hierarchy of classes of languages $\mathcal{SUM}_k$, where $\mathcal{SUM}$ stands for *strongly unambiguous monomial*.

Definition 4.1.3. For a fixed finite alphabet Σ and for each $k \in \mathbb{N}$, we define the set of languages of *strongly unambiguous monomials of degree at most k over* Σ, denoted by $\mathcal{SUM}_k(\Sigma^*)$, by induction on k. A language $L \subseteq \Sigma^*$ is in $\mathcal{SUM}_0(\Sigma^*)$ if it is of the form A^* for some alphabet $A \subseteq \Sigma$. For each $k \in \mathbb{N}_{>0}$, a language $L \subseteq \Sigma^*$ is in $\mathcal{SUM}_k(\Sigma^*)$ if it is in $\mathcal{SUM}_{k-1}(\Sigma^*)$ or $L = L_1 a L_2$ for some languages $L_1 \in \mathcal{SUM}_i(\Sigma^*)$ and $L_2 \in \mathcal{SUM}_j(\Sigma^*)$, where $i, j \in \mathbb{N}$ verify $i + j = k - 1$ and $a \in \Sigma$ is such that no word of L_1 contains the letter a or no word of L_2 contains the letter a. A language L of $\mathcal{SUM}_k(\Sigma^*)$ for $k \in \mathbb{N}$ is called a *strongly unambiguous monomial* (SUM) *over* Σ and is said to be of *degree* l for $l \in \mathbb{N}$ the minimum non-negative integer verifying $L \in \mathcal{SUM}_l(\Sigma^*)$.

For each $k \in \mathbb{N}$, we then define the class of languages of *strongly unambiguous monomials of degree at most k*, denoted by $\mathcal{SUM}_k$, simply as the union over all finite alphabets Σ of $\mathcal{SUM}_k(\Sigma^*)$.

Observe that a SUM is a more constrained form of an unambiguous monomial.

Gavaldà and Thérien stated without proof that a language L over some alphabet Σ is recognised by a monoid in **DA** if and only if there is a $k \in \mathbb{N}$ such that L is a Boolean combination of languages in $\mathcal{SUM}_k$ Gavaldà and Thérien [2003]. This characterisation can be deduced from results shown by Kufleitner and Weil in Kufleitner and Weil [2012] (more precisely, by combining Corollary 3.16, Theorem 3.21 and Proposition 3.23 of their article), but we move on to prove this result directly in this subsection, for completeness.

Lemma 4.1.4. *For all $k \in \mathbb{N}$, any language in $\mathcal{SUM}_k$ has its syntactic monoid in* **DA**.

Proof. We shall prove it by induction on k.

Base case $k = 0$. Let $L \in \mathcal{SUM}_0(\Sigma^*)$ for some alphabet Σ. This means that $L = A^*$ for some $A \subseteq \Sigma$. Let $\sim_L$ be the syntactic congruence of L and let ω_L be the idempotent power of L's syntactic monoid $\mathrm{M}(L) = (M, *)$.

Let $u, v \in \Sigma^*$, we are now going to show that $(uv)^{\omega_L} \sim_L (uv)^{\omega_L} u (uv)^{\omega_L}$.

Let $x, y \in \Sigma^*$ satisfy $x(uv)^{\omega_L} y \in A^* = L$. This means that all letters appearing in x, u, v and y belong to A^*, so that $x(uv)^{\omega_L} u (uv)^{\omega_L} y \in A^* = L$.

Conversely, for all $x, y \in \Sigma^*$, we can show that when $x(uv)^{\omega_L} u (uv)^{\omega_L} y \in L$, we have $x(uv)^{\omega_L} y \in L$.

This shows that $(uv)^{\omega_L} \sim_L (uv)^{\omega_L} u (uv)^{\omega_L}$ and as it is true for all $u, v \in \Sigma^*$, by definition of the syntactic monoid of L, we have that $(m*n)^{*,\omega_L} = (m*n)^{*,\omega_L} * m * (m*n)^{*,\omega_L}$ for all $m, n \in M$, so that $(M, *)$ does belong to **DA**.

This proves the base case.

Induction. Let $k \in \mathbb{N}_{>0}$ and assume that any language in $\mathcal{SUM}_{k-1}$ has its syntactic monoid in **DA**.

Let $L \in \mathcal{SUM}_k(\Sigma^*)$ for some alphabet Σ. This means that either L is in $\mathcal{SUM}_{k-1}(\Sigma^*)$ and so its syntactic monoid belongs to **DA** by inductive hypothesis, or $L = L_1 a L_2$ for some languages $L_1 \in \mathcal{SUM}_i(\Sigma^*)$ and $L_2 \in \mathcal{SUM}_j(\Sigma^*)$, where $i, j \in \mathbb{N}$ verify $i + j = k - 1$ and $a \in \Sigma$ is such that no word of L_1 contains the letter a or no word of L_2 contains the letter a. We shall only treat the case in which a does not appear in any of the words of L_1; the other case is treated symmetrically.

As L_1 and L_2 both belong to $\mathcal{SUM}_{k-1}(\Sigma^*)$, by inductive hypothesis, their respective syntactic monoids belong to **DA**.

Let $\sim_L$, $\sim_{L_1}$ and $\sim_{L_2}$ be the syntactic congruences of L, L_1 and L_2 respectively and let ω_L, ω_{L_1} and ω_{L_2} be the idempotent power of, respectively, L's, L_1's and L_2's syntactic monoid. We shall moreover denote by $\mathrm{M}(L) = (M, *)$ the syntactic monoid of L.

Let $u, v \in \Sigma^*$, we are now going to show that $(uv)^{\omega_L} \sim_L (uv)^{\omega_L} u (uv)^{\omega_L}$. By definition of the syntactic monoid of L and of ω_L, it is not too difficult to see that this is equivalent to showing that $(uv)^{\omega} \sim_L (uv)^{\omega} u (uv)^{\omega}$ where $\omega \in \mathbb{N}_{>0}$ is the smallest multiple of ω_L, ω_{L_1} and ω_{L_2} strictly bigger than ω_{L_2}.

Let $x, y \in \Sigma^*$ such that $w = x(uv)^{\omega} y \in L$ and consider $w' = x(uv)^{\omega} u (uv)^{\omega} y$. Let $i \in [|w|]$ be the minimum integer in $[|w|]$ such that $w_i = a$; by definition of L, we must have $w_1 \cdots w_{i-1} \in L_1$ and $w_{i+1} \cdots w_{|w|} \in L_2$. We also let $i' \in [|w'|]$ be the minimum integer in $[|w'|]$ such that $w'_{i'} = a$ (which necessarily exists, as $a \in \mathrm{alph}(w) \subseteq \mathrm{alph}(w')$).

- If i corresponds to a position in the factor x of w, then $i = i'$ and we have that $w'_1 \cdots w'_{i'-1} = x_1 \cdots x_{i-1} \in L_1$. But, now, since ω is a multiple of ω_{L_2} and the syntactic monoid of L_2 is in **DA**, we have $(uv)^{\omega} \sim_{L_2} (uv)^{\omega} u (uv)^{\omega}$, so that, because $x_{i+1} \cdots x_{|x|} (uv)^{\omega} y = w_{i+1} \cdots w_{|w|} \in L_2$, it follows that $w'_{i'+1} \cdots w'_{|w'|} = x_{i+1} \cdots x_{|x|} (uv)^{\omega} u (uv)^{\omega} y \in L_2$. Hence, $w' \in L$.

- If i corresponds to a position in one of the factors uv of w, then this position is necessarily in the first factor uv, say position $\kappa \in [|uv|]$, by minimality of i, and we have $i = i'$. So we have that $w'_1 \cdots w'_{i'-1} = x(uv)_1 \cdots (uv)_{\kappa-1} \in L_1$. Moreover, since ω is strictly bigger than ω_{L_2} and the syntactic monoid of L_2 is in **DA**, hence is in particular aperiodic, we have $(uv)^{\omega-1} \sim_{L_2} (uv)^{\omega} \sim_{L_2} (uv)^{\omega} u (uv)^{\omega} \sim_{L_2} (uv)^{\omega-1} u (uv)^{\omega}$, so that, because $(uv)_{\kappa+1} \cdots (uv)_{|uv|} (uv)^{\omega-1} y = w_{i+1} \cdots w_{|w|} \in L_2$, it follows that $w'_{i'+1} \cdots w'_{|w'|} = (uv)_{\kappa+1} \cdots (uv)_{|uv|} (uv)^{\omega-1} u (uv)^{\omega} y \in L_2$. Hence, $w' \in L$.

- If i corresponds to a position in the factor y of w, say position $\kappa \in [|y|]$, then i' also corresponds to position κ in the factor y of w', otherwise it would violate the minimality of i. So we have that $w'_{i'+1} \cdots w'_{|w'|} = y_{\kappa+1} \cdots y_{|y|} \in L_2$. But, now, since ω is a multiple of ω_{L_1} and the syntactic monoid of L_1 is in **DA**, we have $(uv)^{\omega} \sim_{L_1} (uv)^{\omega} u (uv)^{\omega}$, so that, because $x(uv)^{\omega} y_1 \cdots y_{\kappa-1} = w_1 \cdots w_{i-1} \in L_1$, it follows that $w'_1 \cdots w'_{i'-1} = x(uv)^{\omega} u (uv)^{\omega} y_1 \cdots y_{\kappa-1} \in L_1$. Hence, $w' \in L$.

Therefore, in any case we have $x(uv)^{\omega} u (uv)^{\omega} y \in L$.

Let $x, y \in \Sigma^*$ satisfy $x(uv)^{\omega} u (uv)^{\omega} y \in L$. In a way similar to above, we can show that then, $x(uv)^{\omega} y \in L$.

This shows that $(uv)^{\omega_L} \sim_L (uv)^{\omega_L} u (uv)^{\omega_L}$ and as it is true for all $u, v \in \Sigma^*$, by definition of the syntactic monoid of L, we have that $(m*n)^{*,\omega_L} = (m*n)^{*,\omega_L} * m * (m*n)^{*,\omega_L}$

for all $m, n \in M$, so that $(M, *)$ does belong to **DA**.

This concludes the inductive step and therefore the proof of the lemma. □

Now, since $\mathcal{L}(\mathbf{DA})$ is by definition closed under Boolean combinations, we have shown the following.

Proposition 4.1.5. *For all $k \in \mathbb{N}$, any Boolean combination of languages in $\mathcal{SUM}_k$ does belong to $\mathcal{L}(\mathbf{DA})$.*

We now turn to the converse statement.

Lemma 4.1.6. *Let $(M, *)$ be some monoid in **DA** and $\varphi\colon \Sigma^* \to M$ a monoid morphism from some free monoid $(\Sigma^*, \cdot)$ for Σ an alphabet to $(M, *)$. Then, there exists $k \in \mathbb{N}$ such that for all $F \subseteq M$, $\varphi^{-1}(F)$ is a Boolean combination of languages in $\mathcal{SUM}_k(\Sigma^*)$.*

Proof. Let $(M, *)$ be some monoid in **DA** and $\varphi\colon \Sigma^* \to M$ a monoid morphism from some free monoid $(\Sigma^*, \cdot)$ for Σ an alphabet to $(M, *)$.

For each element $m \in M$, we define its $\mathfrak{J}$-depth, denoted by $\mathrm{d}_{\mathfrak{J}}(m)$, to be the maximal $l \in \mathbb{N}$ such that there exist $u_0, u_1, \ldots, u_l \in M$ verifying $m = u_0 <_{\mathfrak{J}} u_1 <_{\mathfrak{J}} \cdots <_{\mathfrak{J}} u_l = 1_{(M,*)}$; in a similar way we define its $\mathfrak{R}$-depth, denoted by $\mathrm{d}_{\mathfrak{R}}(m)$ and its $\mathfrak{L}$-depth, denoted by $\mathrm{d}_{\mathfrak{L}}(m)$. It is direct to see that for all $m \in M$, the $\mathfrak{J}$-, $\mathfrak{R}$- and $\mathfrak{L}$-depth of m each are at most $|M| - 1$.

Let us also define the functions

$$\begin{aligned} f\colon [\![0, |\Sigma|\,]\!] \times [\![0, |M| - 1]\!]^3 &\to \mathbb{N} \\ (x_1, x_2, x_3, x_4) &\mapsto |M|^3 \cdot x_1 + |M|^2 \cdot x_2 + |M| \cdot x_3 + x_4 \end{aligned}$$

and

$$\begin{aligned} \langle\cdot\rangle\colon \mathfrak{P}(\Sigma) \times M^3 &\to [\![0, |\Sigma|\,]\!] \times [\![0, |M| - 1]\!]^3 \\ (\Gamma, m, s, t) &\mapsto \left(|\Gamma|\,, \mathrm{d}_{\mathfrak{J}}(m), |M| - 1 - \mathrm{d}_{\mathfrak{R}}(s), |M| - 1 - \mathrm{d}_{\mathfrak{L}}(t)\right). \end{aligned}$$

It is obvious that f is increasing.

We can now formulate the central claim in the proof of the present lemma.

Claim 4.1.7. *For every alphabet $\Gamma \subseteq \Sigma$ and $m, s, t \in M$, the language $\{u \in \Gamma^* \mid s * \varphi(u) * t = m\}$ over Γ is a (possibly empty) union of languages in $\mathcal{SUM}_{2^{f(\langle \Gamma, m, s, t\rangle)} - 1}(\Sigma^*)$.*

It is direct to see that, for all $F \subseteq M$, since $\varphi^{-1}(F) = \bigcup_{m \in F} \varphi^{-1}(m)$, this claim entails that $\varphi^{-1}(F)$ is a Boolean combination of languages in $\mathcal{SUM}_{2^{(|\Sigma|+1)\cdot|M|^3 - 1} - 1}(\Sigma^*)$.

Proof of the claim. We prove it by induction on the quadruplet $\langle \Gamma, m, s, t\rangle$.

Base case $|\Gamma| = 0$**.** Let $\Gamma = \emptyset$ and let $m, s, t \in M$. Then

$$\{u \in \Gamma^* \mid s * \varphi(u) * t = m\} = \begin{cases} \{\varepsilon\} & \text{if } s * t = m \\ \emptyset & \text{otherwise } (s * t \neq m) \ . \end{cases}$$

Hence, since $\{\varepsilon\} = \emptyset^*$, we have that $\{u \in \Gamma^* \mid s * \varphi(u) * t = m\}$ is obviously a (possibly empty) union of languages in $\mathcal{SUM}_0(\Sigma^*) \subseteq \mathcal{SUM}_{2^{f(\langle \Gamma, m, s, t\rangle)}-1}(\Sigma^*)$.

This proves the base case.

Induction. Let $\Gamma \subseteq \Sigma$ be an alphabet with at least one letter and let $m, s, t \in M$. Assume that for any alphabet $\Gamma' \subseteq \Sigma$ and $m', s', t' \in M$ verifying $\langle \Gamma', m', s', t'\rangle < \langle \Gamma, m, s, t\rangle$, we have that the language $\{u' \in \Gamma'^* \mid s' * \varphi(u') * t' = m'\}$ over Γ' is a (possibly empty) union of languages in $\mathcal{SUM}_{2^{f(\langle \Gamma', m', s', t'\rangle)}-1}(\Sigma^*)$.

Let $L = \{u \in \Gamma^* \mid s * \varphi(u) * t = m\}$. If $m \not\leq_{\mathfrak{R}} s$ or $m \not\leq_{\mathfrak{L}} t$, then $L = \emptyset$, which is trivially an empty union of languages in $\mathcal{SUM}_{2^{f(\langle \Gamma, m, s, t\rangle)}-1}(\Sigma^*)$. So we now assume that $m \leq_{\mathfrak{R}} s$ and $m \leq_{\mathfrak{L}} t$. There are five possible cases.

Case 1: there exists $a \in \Gamma$ such that $s * \varphi(a) <_{\mathfrak{R}} s$.

Let us set $\Gamma' = \Gamma \setminus \{a\}$. Consider the language $L' = L \cap \Gamma'^* = \{u' \in \Gamma'^* \mid s * \varphi(u') * t = m\}$ over Γ'. Then, by inductive hypothesis for Γ', m, s and t, since $\langle \Gamma', m, s, t\rangle < \langle \Gamma, m, s, t\rangle$, we have that L' is a (possibly empty) union of languages in $\mathcal{SUM}_{2^{f(\langle \Gamma', m, s, t\rangle)}-1}(\Sigma^*) \subseteq \mathcal{SUM}_{2^{f(\langle \Gamma, m, s, t\rangle)}-1}(\Sigma^*)$.

Moreover, consider $S' = s * \varphi(\Gamma'^*) * \varphi(a)$ and, for each $s' \in S'$, define the language $L_{s'} = \{u' \in \Gamma'^* \mid s * \varphi(u') * \varphi(a) = s'\}$ over Γ' and the language $K_{s'} = \{u \in \Gamma^* \mid s' * \varphi(u) * t = m\}$ over Γ. Let now $s' \in S'$. By inductive hypothesis for Γ', s', s and $\varphi(a)$, since $\langle \Gamma', s', s, \varphi(a)\rangle < \langle \Gamma, m, s, t\rangle$, $L_{s'}$ is a (possibly empty) union of languages in $\mathcal{SUM}_{2^{f(\langle \Gamma', s', s, \varphi(a)\rangle)}-1}(\Sigma^*) \subseteq \mathcal{SUM}_{2^{f(\langle \Gamma, m, s, t\rangle)-1}-1}(\Sigma^*)$. Further, since there exists some $u' \in \Gamma'^*$ such that $s' = s * \varphi(u') * \varphi(a)$, we have that

$$s' = s * \varphi(u') * \varphi(a) \leq_{\mathfrak{R}} s * \varphi(u') \leq_{\mathfrak{R}} s \ .$$

This implies that, either $s * \varphi(u') <_{\mathfrak{R}} s$ and then we also have $s' <_{\mathfrak{R}} s$, or $s * \varphi(u') \ \mathfrak{R} \ s$, which means by Lemma 4.1.1 that we necessarily also have $s' <_{\mathfrak{R}} s * \varphi(u') \ \mathfrak{R} \ s$, since $s * \varphi(a) <_{\mathfrak{R}} s$. Therefore, the $\mathfrak{R}$-depth $\mathrm{d}_{\mathfrak{R}}(s')$ of s' is greater than the $\mathfrak{R}$-depth $\mathrm{d}_{\mathfrak{R}}(s)$ of s. Hence, by inductive hypothesis for Γ, m, s' and t, since $\langle \Gamma, m, s', t\rangle < \langle \Gamma, m, s, t\rangle$, we have that $K_{s'}$ is a (possibly empty) union of languages in $\mathcal{SUM}_{2^{f(\langle \Gamma, m, s', t\rangle)}-1}(\Sigma^*) \subseteq$

$\mathcal{SUM}_{2^{f(\langle\Gamma,m,s,t\rangle)-1}-1}(\Sigma^*)$. By distributivity of concatenation over union and since $L_{s'}$ does not contain any word with the letter a, we have that $L_{s'}aK_{s'}$ is a (possibly empty) union of languages in $\mathcal{SUM}_{2^{f(\langle\Gamma,m,s,t\rangle)}-1}(\Sigma^*)$, and this is true for any $s' \in S'$.

We shall now conclude the proof for the present case by showing that

$$L = L' \cup \bigcup_{s' \in S'} L_{s'}aK_{s'} \ ,$$

which is a (possibly empty) union of languages in $\mathcal{SUM}_{2^{f(\langle\Gamma,m,s,t\rangle)}-1}(\Sigma^*)$ by construction.

Let $w \in L' \cup \bigcup_{s' \in S'} L_{s'}aK_{s'}$. If $w \in L'$, then it is obvious that $w \in L$. Otherwise, there must exist $s' \in S'$ such that $w \in L_{s'}aK_{s'}$. This means that w can be decomposed as $w = u'au$ where $u' \in L_{s'}$ and $u \in K_{s'}$, so that $s * \varphi(w) * t = s * \varphi(u') * \varphi(a) * \varphi(u) * t = s' * \varphi(u) * t = m$. Therefore, $w \in L$ in that case also. Since this is true for any $w \in L' \cup \bigcup_{s' \in S'} L_{s'}aK_{s'}$, we have $L' \cup \bigcup_{s' \in S'} L_{s'}aK_{s'} \subseteq L$.

Now let $w \in L$. Either w does not contain the letter a, and then $w \in L'$. Or w does contain the letter a and can be uniquely decomposed as $w = u'au$ where $u' \in \Gamma'^*$ and $u \in \Gamma^*$. Since $u' \in \Gamma'^*$, this means that $s' = s * \varphi(u') * \varphi(a)$ belongs to S', so that u' belongs to $L_{s'}$. Moreover, as we know that $s' * \varphi(u) * t = s * \varphi(u') * \varphi(a) * \varphi(u) * t = s * \varphi(w) * t = m$, we have that u belongs to $K_{s'}$. Therefore, $w = uau' \in L_{s'}aK_{s'}$. Hence we can conclude that $w \in L' \cup \bigcup_{s' \in S'} L_{s'}aK_{s'}$, so that, since this is true for any $w \in L$, we have $L \subseteq L' \cup \bigcup_{s' \in S'} L_{s'}aK_{s'}$.

This concludes the proof for the present case.

Case 2: there exists $a \in \Gamma$ such that $\varphi(a) * t <_{\mathfrak{L}} t$.

We proceed as for Case 1 by symmetry.

Case 3: for all $a \in \Gamma$, we have $s * \varphi(a)\ \mathfrak{R}\ s$, but $m <_{\mathfrak{R}} s$.

In this case, by Lemma 4.1.1 we have that for all $u \in \Gamma^*$, $s * \varphi(u)\ \mathfrak{R}\ s$. Let $M' = \mathfrak{R}(s) \cap \{m' \in M \mid m' * t = m\}$ and, for each $m' \in M'$, define the language $L_{m'} = \{u \in \Gamma^* \mid s * \varphi(u) = m'\}$ over Γ. Let now $m' \in M'$. Since $m'\ \mathfrak{R}\ s$ and $m <_{\mathfrak{R}} s$, we necessarily have that $m <_{\mathfrak{J}} m'$, otherwise we would have $m\ \mathfrak{R}\ m'$ by Lemma 3.1.20. This implies that the $\mathfrak{J}$-depth $\mathrm{d}_{\mathfrak{J}}(m')$ of m' is smaller than the $\mathfrak{J}$-depth $\mathrm{d}_{\mathfrak{J}}(m)$ of m. Hence, by inductive hypothesis for Γ, m', s and $1_{(M,*)}$, since $\langle\Gamma, m', s, 1_{(M,*)}\rangle < \langle\Gamma, m, s, t\rangle$, we have that $L_{m'}$ is a (possibly empty) union of language in $\mathcal{SUM}_{2^{f(\langle\Gamma,m',s,1_{(M,*)}\rangle)}-1}(\Sigma^*) \subseteq \mathcal{SUM}_{2^{f(\langle\Gamma,m,s,t\rangle)}-1}(\Sigma^*)$, and this is true for any $m' \in M'$.

We shall now conclude the proof for the present case by showing that

$$L = \bigcup_{m' \in M'} L_{m'} \ ,$$

which is a (possibly empty) union of languages in $\mathcal{SUM}_{2^{f(\langle \Gamma, m, s, t \rangle)-1}}(\Sigma^*)$ by construction.

Let $w \in \bigcup_{m' \in M'} L_{m'}$. There must exist $m' \in M'$ such that $w \in L_{m'}$. This means that $s * \varphi(w) * t = m' * t = m$. Therefore, $w \in L$ and since this is true for any $w \in \bigcup_{m' \in M'} L_{m'}$, we have $\bigcup_{m' \in M'} L_{m'} \subseteq L$.

Now let $w \in L$. If we let $m' = s * \varphi(w)$, then, by what we have seen just above, we have $m' \mathfrak{R} s$. Moreover, $m' * t = s * \varphi(w) * t = m$ by definition of L. This means that $m' \in M'$, so that $w \in L_{m'}$. Hence, since this is true for any $w \in L$, we have $L \subseteq \bigcup_{m' \in M'} L_{m'}$.

This concludes the proof for the present case.

Case 4: for all $a \in \Gamma$, we have $\varphi(a) * t \mathfrak{L} t$, but $m <_{\mathfrak{L}} t$.

We proceed as for Case 3 by symmetry.

Case 5: for all $a \in \Gamma$, we have $s * \varphi(a) \mathfrak{R} s$ and $\varphi(a) * t \mathfrak{L} t$, and $m \mathfrak{R} s$ and $m \mathfrak{L} t$.

Then, as in the two previous cases, we have that for all $u \in \Gamma^*$, $s * \varphi(u) \mathfrak{R} s$ and $\varphi(u) * t \mathfrak{L} t$.

If we have that $s * t <_{\mathfrak{R}} s$, then we have that $s * \varphi(w) * t <_{\mathfrak{R}} s \mathfrak{R} m$ for all $w \in \Gamma^*$ by Lemma 4.1.1. If we have $s * t <_{\mathfrak{L}} t$, then, similarly, we have that $s * \varphi(w) * t <_{\mathfrak{L}} t \mathfrak{L} m$ for all $w \in \Gamma^*$. This implies that in both cases $L = \emptyset$, which is trivially an empty union of languages in $\mathcal{SUM}_{2^{f(\langle \Gamma, m, s, t \rangle)-1}}(\Sigma^*)$.

Otherwise, we have that $s * t \mathfrak{R} s$ and $s * t \mathfrak{L} t$. This means, by Lemma 4.1.1, that $s * \varphi(w) * t \in \mathfrak{R}(s) \cap \mathfrak{L}(t)$. As $(M, *)$ is aperiodic and hence $\mathfrak{H}$-trivial, since we have $m \in \mathfrak{R}(s) \cap \mathfrak{L}(t)$, it means that $\mathfrak{R}(s) \cap \mathfrak{L}(t) = \mathfrak{R}(m) \cap \mathfrak{L}(m) = \mathfrak{H}(m)$, so by $\mathfrak{H}$-triviality of $(M, *)$, $\mathfrak{R}(s) \cap \mathfrak{L}(t) = \{m\}$, showing that $s * \varphi(w) * t = m$ for all $w \in \Gamma^*$. This implies that $L = \Gamma^*$, which is trivially a union of languages in $\mathcal{SUM}_0(\Sigma^*) \subseteq \mathcal{SUM}_{2^{f(\langle \Gamma, m, s, t \rangle)-1}}(\Sigma^*)$.

This concludes the proof of the last case.

This ends the inductive step and therefore the proof of the claim. □

As well as that of the lemma. □

We can directly conclude the following from this lemma.

Proposition 4.1.8. *Any language over some alphabet Σ recognised by a monoid in* **DA** *is a Boolean combination of languages in $\mathcal{SUM}_k(\Sigma^*)$ for some $k \in \mathbb{N}$.*

Finally, we get the desired characterisation.

Theorem 4.1.9. *For any alphabet Σ, a language L belongs to $\mathcal{L}(\mathbf{DA})(\Sigma^*)$ if and only if there exists $k \in \mathbb{N}$ such that L is a Boolean combination of languages in $\mathcal{SUM}_k(\Sigma^*)$.*

4.1.2 A parameterisation of DA

For each $k \in \mathbb{N}$, we denote by $\mathcal{SUL}_k$ the class of regular languages that are Boolean combinations of languages in $\mathcal{SUM}_k$; it is a variety of languages as shown just below. For each $k \in \mathbb{N}$, we denote by $\mathbf{DA_k}$ the variety of finite monoids generated by the syntactic monoids of the languages in $\mathcal{SUL}_k$; by Eilenberg's theorem (Theorem 3.2.10), we have that $\mathcal{L}(\mathbf{DA_k})$, the variety of languages whose syntactic monoid belongs to $\mathbf{DA_k}$, is exactly $\mathcal{SUL}_k$.

We emphasise the fact that we cannot restrict $\mathcal{SUL}_k$ for $k \in \mathbb{N}$ to be the class of regular languages that are unions of languages in $\mathcal{SUM}_k$, because then, while $b^*aa^*, b^*ba^* \in \mathcal{SUL}_1(\{a,b\}^*)$, we would have $b^*aa^* \cap b^*ba^* = b^*baa^* \notin \mathcal{SUL}_1(\{a,b\}^*)$, therefore $\mathcal{SUL}_1$ wouldn't be a variety of languages.

Let us now show the following.

Proposition 4.1.10. *For all $k \in \mathbb{N}$, $\mathcal{SUL}_k$ is a variety of languages.*

Closure under Boolean operations and trivial languages containment is obvious by construction. Closure under quotients and inverses of monoid morphisms is respectively given by the following two lemmata and by the fact that both quotients and inverses of monoid morphisms commute with Boolean operations.

Lemma 4.1.11. *For all $k \in \mathbb{N}$, for all $L \in \mathcal{SUM}_k(\Sigma^*)$ where Σ is some alphabet and $u \in \Sigma^*$, $u^{-1}L$ and Lu^{-1} both are unions of languages in $\mathcal{SUM}_k(\Sigma^*)$.*

Proof. We prove it by induction on k.

Base case $k = 0$. Let $L \in \mathcal{SUM}_0(\Sigma^*)$ where Σ is some alphabet and $u \in \Sigma^*$. This means that $L = A^*$ for some $A \subseteq \Sigma$. We have two cases: either $\mathrm{alph}(u) \nsubseteq A$ and then $u^{-1}L = Lu^{-1} = \emptyset$; or $\mathrm{alph}(u) \subseteq A$ and then $u^{-1}L = Lu^{-1} = A^* = L$. So $u^{-1}L$ and Lu^{-1} both are unions of languages in $\mathcal{SUM}_0(\Sigma^*)$. The base case is hence proved.

Inductive step. Let $k \in \mathbb{N}_{>0}$ and assume that the lemma is true for all $k' \in \mathbb{N}, k' < k$.

Let $L \in \mathcal{SUM}_k(\Sigma^*)$ where Σ is some alphabet and $u \in \Sigma^*$. This means that either L is in $\mathcal{SUM}_{k-1}(\Sigma^*)$ and hence $u^{-1}L$ and Lu^{-1} both are unions of languages in $\mathcal{SUM}_k(\Sigma^*)$ by

applying the inductive hypothesis directly for L and u, or $L = L_1 a L_2$ for some languages $L_1 \in \mathcal{SUM}_i(\Sigma^*)$ and $L_2 \in \mathcal{SUM}_j(\Sigma^*)$ and some letter $a \in \Sigma$ with $i + j = k - 1$ verifying that either no word of L_1 contains the letter a or no word of L_2 contains the letter a. We shall only treat the case in which a does not appear in any of the words of L_1; the other case is treated symmetrically.

There are again two cases to consider, depending on whether a does appear in u or not.

If $a \notin \text{alph}(u)$, then it is straightforward to check that $u^{-1}L = (u^{-1}L_1)aL_2$ and $Lu^{-1} = L_1 a(L_2 u^{-1})$. By the inductive hypothesis, we get that $u^{-1}L_1$ is a union of languages in $\mathcal{SUM}_i(\Sigma^*)$ and that $L_2 u^{-1}$ is a union of languages in $\mathcal{SUM}_j(\Sigma^*)$. Moreover, it is direct to see that no word of $u^{-1}L_1$ contains the letter a. By distributivity of concatenation over union, we finally get that $u^{-1}L$ and Lu^{-1} both are unions of languages in $\mathcal{SUM}_k(\Sigma^*)$.

If $a \in \text{alph}(u)$, then let $u = u_1 a u_2$ with $u_1, u_2 \in \Sigma^*$ and $a \notin \text{alph}(u_1)$. It is again straightforward to see that

$$u^{-1}L = \begin{cases} {u_2}^{-1}L_2 & \text{if } u_1 \in L_1 \\ \emptyset & \text{otherwise} \end{cases}$$

and

$$Lu^{-1} = L_1 a(L_2 u^{-1}) \cup \begin{cases} L_1 {u_1}^{-1} & \text{if } u_2 \in L_2 \\ \emptyset & \text{otherwise} \end{cases}.$$

As before, by the inductive hypothesis, we get that $L_1 {u_1}^{-1}$ is a union of languages in $\mathcal{SUM}_i(\Sigma^*)$ and that both ${u_2}^{-1}L_2$ and $L_2 u^{-1}$ are unions of languages in $\mathcal{SUM}_j(\Sigma^*)$. And, again, by distributivity of concatenation over union, we get that $u^{-1}L$ and Lu^{-1} both are unions of languages in $\mathcal{SUM}_k(\Sigma^*)$.

This concludes the inductive step and therefore the proof of the lemma. □

Lemma 4.1.12. *For all $k \in \mathbb{N}$, for all $L \in \mathcal{SUM}_k(\Sigma^*)$ where Σ is some alphabet and $\varphi\colon \Gamma^* \to \Sigma^*$ a monoid morphism from a free monoid $(\Gamma^*, \cdot)$ for Γ an alphabet to $(\Sigma^*, \cdot)$, $\varphi^{-1}(L)$ is a union of languages in $\mathcal{SUM}_k(\Gamma^*)$.*

Proof. We prove it by induction on k.

Base case $k = 0$. Let $L \in \mathcal{SUM}_0(\Sigma^*)$ where Σ is some alphabet and $\varphi\colon \Gamma^* \to \Sigma^*$ a monoid morphism from a free monoid $(\Gamma^*, \cdot)$ for Γ an alphabet to $(\Sigma^*, \cdot)$. This means

that $L = A^*$ for some $A \subseteq \Sigma$. It is straightforward to check that $\varphi^{-1}(L) = B^*$ where $B = \{b \in \Gamma \mid \varphi(b) \in A^*\}$. B^* is certainly a union of languages in $\mathcal{SUM}_0(\Gamma^*)$. The base case is hence proved.

Inductive step. Let $k \in \mathbb{N}_{>0}$ and assume that the lemma is true for all $k' \in \mathbb{N}, k' < k$.

Let $L \in \mathcal{SUM}_k(\Sigma^*)$ where Σ is some alphabet and $\varphi\colon \Gamma^* \to \Sigma^*$ a monoid morphism from a free monoid $(\Gamma^*, \cdot)$ for Γ an alphabet to $(\Sigma^*, \cdot)$. This means that either L is in $\mathcal{SUM}_{k-1}(\Sigma^*)$ and hence $\varphi^{-1}(L)$ is a union of languages in $\mathcal{SUM}_k(\Gamma^*)$ by applying the inductive hypothesis directly for L and φ, or $L = L_1 a L_2$ for some languages $L_1 \in \mathcal{SUM}_i(\Sigma^*)$ and $L_2 \in \mathcal{SUM}_j(\Sigma^*)$ and some letter $a \in \Sigma$ with $i + j = k - 1$ verifying that either no word of L_1 contains the letter a or no word of L_2 contains the letter a. We shall only treat the case in which a does not appear in any of the words of L_1; the other case is treated symmetrically.

Let us define $B = \{b \in \Gamma \mid a \in \mathrm{alph}(\varphi(b))\}$ as the set of letters of Γ whose image word by φ contains the letter a. For each $b \in B$, we shall also let $\varphi(b) = u_{b,1} a u_{b,2}$ with $u_{b,1}, u_{b,2} \in \Sigma^*$ and $a \notin \mathrm{alph}(u_{b,1})$. It is not too difficult to see that we then have

$$\varphi^{-1}(L) = \bigcup_{b \in B} \varphi^{-1}(L_1 {u_{b,1}}^{-1}) b \varphi^{-1}({u_{b,2}}^{-1} L_2) \ .$$

By the inductive hypothesis, by Lemma 4.1.11 and by the fact that inverses of monoid morphisms commute with unions, we get that $\varphi^{-1}(L_1 {u_{b,1}}^{-1})$ is a union of languages in $\mathcal{SUM}_i(\Gamma^*)$ and that $\varphi^{-1}({u_{b,2}}^{-1} L_2)$ is a union of languages in $\mathcal{SUM}_j(\Gamma^*)$. Moreover, it is direct to see that no word of $\varphi^{-1}(L_1 {u_{b,1}}^{-1})$ contains the letter b for all $b \in B$. By distributivity of concatenation over union, we finally get that $\varphi^{-1}(L)$ is a union of languages in $\mathcal{SUM}_k(\Gamma^*)$.

This concludes the inductive step and therefore the proof of the lemma. □

Let us finish with some positive and negative examples of languages in $\mathcal{L}(\mathbf{DA}) = \mathcal{SUL}$. For instance, the language $a(a+b)^*$ of words over $\{a, b\}$ starting with an a is in $\mathcal{SUL}$, because it is in $\mathcal{SUL}_1(\{a, b\}^*)$. Similarly, the language of words over $\{a, b, c\}$ containing at least one occurrence of c, the first of all the occurrences of c preceded by an a and the last of those followed by a b, but not starting with $abba$ is in $\mathcal{SUL}_4(\{a, b, c\}^*)$, since it is equal to $\big((a+b)^* ac (a+b+c)^* cb (a+b)^* \cup (a+b)^* acb (a+b)^*\big) \cap \big(abba(a+b+c)^*\big)^{\complement}$.

However, both the language $(ab)^*$ of words over $\{a, b\}$ where a and b alternate, starting with an a and ending with a b, and the language $(a+b)^* bb (a+b)^*$ of words over $\{a, b\}$ containing bb as a factor are provably not in $\mathcal{SUL}$, because their respective syntactic

monoids do not belong to **DA**.

4.2 Tameness of DA

The main result of this chapter is that **DA** is tame.

Theorem 4.2.1. DA *is an* sp*-variety of finite monoids.*

Combined with the fact that **DA** is local Almeida [1996], we obtain the following result by Proposition 3.4.9.

Theorem 4.2.2. $\mathcal{P}(\mathbf{DA}) \cap \mathcal{R}\mathsf{eg} = \mathcal{L}(\mathbf{QDA})$.

We devote the rest of this section to proving Theorem 4.2.1. The result follows from the following main technical contribution.

Proposition 4.2.3. $(c+ab)^*$, $(b+ab)^*$ *and* $b^*((ab^*)^k)^*$ *for any* $k \in \mathbb{N}, k \geq 2$ *are regular languages not in* $\mathcal{P}(\mathbf{DA})$.

Before proving the proposition we first show that it implies that **DA** is an *sp*-variety of finite monoids. This implication is a consequence of the following lemma, which is a result inspired by an observation in Tesson and Thérien [2002] stating that non-membership of a given finite monoid $(M, *)$ in **DA** implies non-aperiodicity of $(M, *)$ or division of it by (at least) one of two specific finite monoids.

Lemma 4.2.4. *Let* $(S, *)$ *be a finite semigroup such that* $(S, *)^1 \notin \mathbf{DA}$. *Then, one of* $(c+ab)^*$, $(b+ab)^*$ *or* $b^*((ab^*)^k)^*$ *for some* $k \in \mathbb{N}, k \geq 2$ *is recognised by a monoid morphism* $\mu\colon \Sigma^* \to S^1$ *from the appropriate free monoid* $(\Sigma^*, \cdot)$ *to* $(S, *)^1$ *such that* $\mu(\Sigma^+) \subseteq S$.

Proof. Let $\omega \in \mathbb{N}_{>0}$ be the idempotent power of $(S, *)^1$.

Aperiodic case. Assume first that $(S, *)^1$ is aperiodic. Then, since $(S, *)^1 \notin \mathbf{DA}$, there exist x, y in S such that $(xy)^\omega \neq (xy)^\omega x (xy)^\omega$.

Set $e = (xy)^\omega$, $f = (yx)^\omega$, $s = ex$ and $t = ye$. Our hypothesis says that $exe \neq e$. We now have two cases, depending on whether $fyf = f$ or not.

Subcase $fyf \neq f$**.** Suppose $fyf \neq f$. In that case, let $\mu\colon \{a, b, c\}^* \to S^1$ be the monoid morphism from the free monoid $(\{a, b, c\}^*, \cdot)$ to $(S, *)^1$ sending a to s, b to t and c to e and consider the language $L = \mu^{-1}(\{1, e\})$. We are now going to show that no word of L can contain aa, bb, ac or cb as a factor.

- Assume that L contains a word w with two consecutive a's. Then $w = w_1aaw_2$ with $w_1, w_2 \in \{a, b, c\}^*$ and as $w \in L$, either $e = \mu(w_1)exex\mu(w_2)$ or $1 = \mu(w_1)exex\mu(w_2)$. In both cases $e = u_1exeu_2$ for some suitable values of u_1 and u_2 taken from S. This implies that

$$e = u_1e(xeu_2) = u_1^2e(xeu_2)^2 = u_1^3e(xeu_2)^3 = \cdots = u_1^\omega e(xeu_2)^\omega$$

and, similarly, that $e = (u_1ex)^\omega eu_2^\omega$. Because $(S, *)^1$ is aperiodic, this in turn entails

$$exeu_2 = u_1^\omega e(xeu_2)^\omega(xeu_2) = u_1^\omega e(xeu_2)^\omega = e$$

and

$$eu_2 = (u_1ex)^\omega eu_2^\omega u_2 = (u_1ex)^\omega eu_2^\omega = e\ .$$

Hence $exe = exeu_2 = e$, contradicting the fact that $exe \neq e$. So L does not contain any word with two consecutive a's.

- Assume that L contains a word w with the factor ac. Then $w = w_1acw_2$ with $w_1, w_2 \in \{a, b, c\}^*$ and as $w \in L$, either $e = \mu(w_1)exe\mu(w_2)$ or $1 = \mu(w_1)exe\mu(w_2)$. So, as just before, in both cases $e = u_1exeu_2$ for some suitable values of u_1 and u_2 taken from S, which entails $exe = e$, contradicting the fact $exe \neq e$. So L does not contain any word with the factor ac.

- Assume that L contains a word w with two consecutive b's. Then $w = w_1bbw_2$ with $w_1, w_2 \in \{a, b, c\}^*$ and as $w \in L$, either $e = \mu(w_1)fyfy\mu(w_2)$ or $1 = \mu(w_1)fyfy$ $\mu(w_2)$, as $ye = y(xy)^\omega = (yx)^\omega y = fy$. In both cases $f = u_1fyfu_2$ for some suitable values of u_1 and u_2 taken from S, because, by aperiodicity of $(S, *)^1$, we have $yex = y(xy)^\omega x = (yx)^{\omega+1} = (yx)^\omega = f$. Similarly to what we did for the factor aa, this implies that $f = u_1^\omega f(yfu_2)^\omega = (u_1fy)^\omega fu_2^\omega$, which in turn entails $f = fyfu_2 = fu_2$. Hence $fyf = fyfu_2 = f$, contradicting the fact that $fyf \neq f$. So L does not contain any word with two consecutive b's.

- Assume that L contains a word w with the factor cb. Then $w = w_1cbw_2$ with $w_1, w_2 \in \{a, b, c\}^*$ and as $w \in L$, either $e = \mu(w_1)eye\mu(w_2)$ or $1 = \mu(w_1)eye\mu(w_2)$. So, similarly to what we did for the factor aa, in both cases $e = u_1eyeu_2$ for some suitable values of u_1 and u_2 taken from S, which entails $eye = e$. Now this means

that

$$\begin{aligned} eye &= e \\ yeye &= ye \\ fyfy &= fy && \text{since } ye = fy \\ fyfyx &= fyx \\ fyf &= f && \text{as } fyx = (yx)^\omega yx = (yx)^{\omega+1} = (yx)^\omega = f \ , \end{aligned}$$

contradicting the fact $fyf \neq f$. So L does not contain any word with the factor cb.

Because L is a language over the alphabet $\{a, b, c\}$, any word w in it is of the form $u_0v_1u_1 \cdots u_{k-1}v_ku_k$ where $k \in \mathbb{N}$, $v_1, \ldots, v_k \in c^+$ and $u_0, \ldots, u_k \in (a+b)^*$. As w does not contain aa nor bb as a factor, we have that $u_0, \ldots, u_k \in (b+\varepsilon)(ab)^*(a+\varepsilon)$. When $k \geq 1$, as moreover w does not contain ac nor cb as a factor, it follows that $u_1, \ldots, u_{k-1} \in (ab)^*$, $u_0 \in (b+\varepsilon)(ab)^*$ and $u_k \in (ab)^*(a+\varepsilon)$; u_0 can therefore be written as $\beta u_0'$ where $u_0' \in (ab)^*$ and β is b if $u_0 \in b(ab)^*$ and the empty word otherwise, and u_k can be written as $u_k'\alpha$ where $u_k' \in (ab)^*$ and α is a if $u_1 \in (ab)^*a$ and the empty word otherwise. We now observe that $\mu(ab) = exye = (xy)^{2\omega+1} = (xy)^\omega = e$ by aperiodicity and we consider four different cases.

- $\beta = \alpha = \varepsilon$.

  Then, $\mu(w) = \mu(u_0'v_1u_1 \cdots u_{k-1}v_ku_k') = e$.

- $\beta = b$ and $\alpha = \varepsilon$.

  Then $\mu(w) = \mu(b)\mu(u_0'v_1u_1 \cdots u_{k-1}v_ku_k') = yee = ye$ that does not belong to $\{1, e\}$, otherwise we would have $eye = e$ which would entail $fyf = f$, as shown in the previous paragraph. But this contradicts the fact that $w \in L$, so this case cannot occur.

- $\beta = \varepsilon$ and $\alpha = a$.

  Then $\mu(w) = \mu(u_0'v_1u_1 \cdots u_{k-1}v_ku_k')\mu(a) = eex = ex$ that does not belong to $\{1, e\}$, otherwise we would have $exe = e$. But this contradicts the fact that $w \in L$, so this case cannot occur.

- $\beta = b$ and $\alpha = a$.

  Then $\mu(w) = \mu(b)\mu(u_0'v_1u_1 \cdots u_{k-1}v_ku_k')\mu(a) = yeeex = yex = f$ by aperiodicity. We have that f does not belong to $\{1, e\}$. Indeed, suppose for the sake of contra-

diction that it does: there are two cases to examine. Either $(yx)^\omega = f = e = (xy)^\omega$, and then $exe = exf = (xy)^\omega x(yx)^\omega = (xy)^\omega(xy)^\omega x = (xy)^\omega x = ex$. But $ex = (xy)^\omega x = x(yx)^\omega = x(xy)^\omega xy = xexy$ by aperiodicity, so $ex = x^\omega exy^\omega$. Hence $exy = x^\omega exy^\omega y = x^\omega exy^\omega = ex$ by aperiodicity, while $exy = (xy)^\omega xy = (xy)^\omega = e$ by aperiodicity again. So $exe = ex = e$, contradicting the fact $exe \neq e$. Or $(yx)^\omega = f = 1$, and then $fyf = y$. But $y = y(yx)^\omega = y(yx)^\omega yx = yyx$ by aperiodicity, so $y = y^\omega yx^\omega$. Hence $yx = y^\omega yx^\omega x = y^\omega yx^\omega = y$ by aperiodicity, while $yx = yx(yx)^\omega = (yx)^\omega = f$ by aperiodicity again. So $fyf = y = f$, contradicting the fact $fyf \neq f$. Therefore, $\mu(w)$ does not belong to $\{1, e\}$, contradicting the fact that $w \in L$, so this case cannot occur either.

This means that, necessarily, $\alpha = \beta = \varepsilon$, so that $u_0, u_k \in (ab)^*$. And for the same reasons, $u_0 \in (ab)^*$ when $k = 0$. Therefore, we have $w \in (c + ab)^*$ and since it is true for any $w \in L$, it follows that $L \subseteq (c + ab)^*$. Combined with the fact that $\mu((c + ab)^*) = \{1, e\}$, we can conclude that $\mu^{-1}(\{1, e\}) = L = (c + ab)^*$, showing $(c + ab)^*$ is recognised by μ verifying $\mu(\{a, b, c\}^+) \subseteq S$.

Subcase $fyf = f$. Suppose now $fyf = f$. In that case, let $\mu\colon \{a, b\}^* \to S^1$ be the monoid morphism from the free monoid $(\{a, b\}^*, \cdot)$ to $(S, *)^1$ sending a to s and b to t and consider the language $L = \mu^{-1}(\{1, e, t\})$. Assume that L contains a word w with two consecutive a's. Then $w = w_1aaw_2$ with $w_1, w_2 \in \{a, b\}^*$ and as $w \in L$, we have that $\mu(w_1)exex\mu(w_2)$ is equal to t, e or 1. Since $xt = xye = xy(xy)^\omega = (xy)^\omega = e$ by aperiodicity, in all cases $e = u_1exeu_2$ for some suitable values of u_1 and u_2 taken from S, which, as for the subcase $fyf \neq f$, implies $exe = e$, contradicting the fact $exe \neq e$. So L does not contain any word with two consecutive a's.

This means that any word in L belongs to $(b + ab)^*(a + \varepsilon)$ so that any word w in L is of the form $u\alpha$ where $u \in (b + ab)^*$ and α is a if $w \in (b + ab)^*a$ and the empty word otherwise. Since, as for the previous case, $\mu(ab) = e$, but also $\mu(bb) = tt = yeye = fyfy = fy = ye = t = \mu(b)$, $te = yee = ye = t$ and $et = eye = efy = xfyfy = xfy = e$ (by aperiodicity), we have that $\mu(u) \in \{1, e, t\}$. Assume now that $\alpha = a$. There are three different cases.

- $\mu(u) = 1$.

  Then $\mu(w) = 1\mu(a) = ex$ that does not belong to $\{1, e, t\}$, otherwise we would have $exe = e$, because $ete = et = e$ by the equalities proved just above. But this contradicts the fact that $w \in L$, so this case cannot occur.

- $\mu(u) = e$.

  Then $\mu(w) = e\mu(a) = eex = ex$ that does not belong to $\{1, e, t\}$ (see the previous case). But this contradicts the fact that $w \in L$, so this case cannot occur.

- $\mu(u) = t$.

  Then $\mu(w) = t\mu(a) = yeex = yex = f$ by aperiodicity. We have that f does not belong to $\{1, e, t\}$. Indeed, suppose for the sake of contradiction that is does: there are three cases to examine. Either $(yx)^\omega = f = t = ye = y(xy)^\omega$, and then $exe = (xy)^\omega x(xy)^\omega = x(yx)^\omega(xy)^\omega = xy(xy)^\omega(xy)^\omega = (xy)^\omega = e$ by aperiodicity, contradicting the fact $exe \neq e$. Or $(yx)^\omega = f = e = (xy)^\omega$, and then $exe = (xy)^\omega x(xy)^\omega = (xy)^\omega x(yx)^\omega = (xy)^\omega x(yx)^\omega y(yx)^\omega = (xy)^\omega(xy)^{\omega+1}(xy)^\omega = (xy)^\omega = e$ by aperiodicity and since $fyf = f$, contradicting the fact $exe \neq e$. Or $(yx)^\omega = f = 1$, and then $y = fyf = f = 1$. But $e = (xy)^\omega = x^\omega$, so $exe = x^\omega x x^\omega = x^\omega = e$, contradicting the fact $exe \neq e$. Therefore, $\mu(w)$ does not belong to $\{1, e, t\}$, contradicting the fact that $w \in L$, so this case cannot occur either.

This means that, necessarily, $\alpha = \varepsilon$, so that $w \in (b + ab)^*$ and since it is true for any $w \in L$, it follows that $L \subseteq (b+ab)^*$. Combined with the fact that $\mu((b+ab)^*) = \{1, e, t\}$, we can conclude that $\mu^{-1}(\{1, e, t\}) = L = (b + ab)^*$, showing $(b + ab)^*$ is recognised by μ verifying $\mu(\{a, b\}^+) \subseteq S$.

Non-aperiodic case. Assume now that $(S, *)^1$ is not aperiodic. Then there is an x in S such that $x^\omega \neq x^{\omega+1}$. Consider the monoid morphism $\mu\colon \{a, b\}^* \to S^1$ from the free monoid $(\{a, b\}^*, \cdot)$ to $(S, *)^1$ sending a to $x^{\omega+1}$ and b to x^ω, and the language $L = \mu^{-1}(x^\omega)$. Let $k \in \mathbb{N}, k \geq 2$ be the smallest positive integer such that $x^{\omega+k} = x^\omega$, that cannot be 1 because $x^\omega \neq x^{\omega+1}$. Using this, for all $w \in \{a, b\}^*$, we have

$$\mu(w) = x^{|w|\cdot\omega+|w|_a} = x^{\omega+(|w|_a \mod k)},$$

so that w belongs to L if and only if $|w|_a = 0 \mod k$, that is, L is the language of all words with a number of a's divisible by k, $b^*((ab^*)^k)^*$. In conclusion, $b^*((ab^*)^k)^*$ is recognised by μ verifying $\mu(\{a, b\}^+) \subseteq S$. □

Let now $(S, *)$ be any finite semigroup such that $\mathcal{W}((S, *)) \subseteq \mathcal{P}(\mathbf{DA})$. Let $\eta_{(S,*)}\colon S^* \to S^1$ be the evaluation morphism of $(S, *)$. To show that $(S, *)^1$ is in $\mathbf{DA}$, we assume for the sake of contradiction that it is not the case. Then Lemma 4.2.4 tells us that one of $(c+ab)^*$,

$(b + ab)^*$ or $b^*((ab^*)^k)^*$ for some $k \in \mathbb{N}, k \geq 2$ is recognised by a monoid morphism $\mu\colon \Sigma^* \to S^1$ from the appropriate free monoid $(\Sigma^*, \cdot)$ to $(S, *)^1$ such that $\mu(\Sigma^+) \subseteq S$.

In all cases, we thus have a language $L \subseteq \Sigma^*$ equal to $\mu^{-1}(Q)$ for some subset Q of S^1 with the morphism μ sending letters of Σ to elements of S. Consider then the monoid morphism $\varphi\colon \Sigma^* \to S^*$ from the free monoid $(\Sigma^*, \cdot)$ to the free monoid $(S^*, \cdot)$ sending each letter $a \in \Sigma$ to $\mu(a)$, a letter of S: it is obvious that $\mu = \eta_{(S,*)} \circ \varphi$, so that $L = \mu^{-1}(Q) = (\eta_{(S,*)} \circ \varphi)^{-1}(Q) = \varphi^{-1}(\eta^{-1}_{(S,*)}(Q))$. As $\mathcal{W}((S, *)) \subseteq \mathcal{P}(\mathbf{DA})$, it follows that $\eta^{-1}_{(S,*)}(Q) \in \mathcal{P}(\mathbf{DA})$, hence since φ is an *lm*-morphism and $\mathcal{P}(\mathbf{DA}) \cap \mathcal{R}\mathsf{eg}$ is closed under inverses of *lm*-morphisms by Proposition 3.4.2, we have $L = \varphi^{-1}(\eta^{-1}_{(S,*)}(Q)) \in \mathcal{P}(\mathbf{DA})$: a contradiction to Proposition 4.2.3.

In the remaining part of this section we prove Proposition 4.2.3.

Proof of Proposition 4.2.3. The idea of the proof is the following. We work by contradiction and assume that we have a sequence of programs $(P_n)_{n\in\mathbb{N}}$ over some monoid $(M, *)$ of **DA** recognising one of the targeted languages. Let n be much larger than the order of $(M, *)$. Consider a language of the form Δ^* for some finite set Δ of words (for instance assume $\Delta = \{c, ab\}$, $\Delta = \{b, ab\}$, ...). We will show that we can fix a constant (depending on $(M, *)$ and Δ but not on n) number of entries to P_n such that P_n always outputs the same value and there is a completion of the fixed entries in Δ^*; hence, if Δ was chosen so that there is actually a completion of the fixed entries in the targeted language and one outside of it, P_n cannot recognise the restriction of that language to words of length n. We cannot prove this for all Δ, in particular it will not work for $\Delta = \{ab\}$ and indeed $(ab)^*$ is in $\mathcal{P}(\mathbf{DA})$. The key property of our Δ is that after fixing any letter at any position, except maybe for a constant number of positions, one can still complete the word into one within Δ^*. This is not true for $\Delta = \{ab\}$ because after fixing a b at an odd position all completions fall outside of $(ab)^*$.

We now spell out the technical details.

Let Δ be a finite non-empty set of non-empty words. Let Σ be the corresponding finite alphabet and let $\perp$ be a letter not in Σ. A *mask* is a word over $\Sigma \cup \{\perp\}$. The positions of a mask carrying a $\perp$ are called *free* while the positions carrying a letter in Σ are called *fixed*. A mask λ' is a *submask* of a mask λ if it is formed from λ by replacing some occurrences of $\perp$ by a letter in Σ or is simply equal to λ.

A *completion* of a mask λ is a word w over Σ that is built from λ by replacing all occurrences of $\perp$ by a letter in Σ. Notice that all completions of a mask have the same length as the mask itself. A mask λ is Δ-*compatible* if it has a completion in Δ^*.

The *dangerous* positions of a mask λ are the positions within distance $2l - 2$ of the

fixed positions or within distance $l-1$ of the beginning or the end of the mask, where l is the maximal length of a word in Δ. A position that is not dangerous is said to be *safe* and is necessarily free.

We say that Δ is *safe* if the following holds. Let λ be a Δ-compatible mask. Let i be any free position of λ that is not dangerous. Let a be any letter in Σ. Then the submask of λ constructed by fixing a at position i is Δ-compatible. We have already seen that $\Delta = \{ab\}$ is not safe. However our targeted Δ, $\Delta = \{c, ab\}$, $\Delta = \{b, ab\}$, $\Delta = \{a, b\}$ (and, in fact, all Δ containing at least one word of length 1), are safe. We always consider Δ to be safe in the following.

Finally, we say that a completion w of a mask λ is *safe* if w is a completion of λ belonging to Δ^* or is constructed from a completion of λ in Δ^* by modifying only letters at safe positions of λ, the dangerous positions remaining unchanged.

Let $(M, *)$ be a monoid in **DA** whose identity we will denote by 1.

An element r of M is $\mathfrak{R}$-bad for $u \in M$ if $ur <_{\mathfrak{R}} u$. Similarly an element r of M is $\mathfrak{L}$-bad for $v \in M$ if $rv <_{\mathfrak{L}} v$. By Lemma 4.1.1, it follows from $(M, *) \in \mathbf{DA}$ that being $\mathfrak{R}$-bad or $\mathfrak{L}$-bad only depends on the $\mathfrak{R}$- or $\mathfrak{L}$-class, respectively.

Let $n \in \mathbb{N}$. We are now going to prove the main technical lemma that allows us to assert that after fixing a constant number of positions in the input of an $(M, *)$-program on Σ^n, the input can still be completed into a word of Δ^*, but the program can no longer make the difference between any two possible completions. To prove the lemma, we define a relation $\prec$ on the set of quadruplets (λ, P, u, v) where λ is a mask of length n, P is a program over $(M, *)$ on Σ^n and u and v are two elements of M. We will say that an element $(\lambda_1, P_1, u_1, v_1)$ is strictly smaller than $(\lambda_2, P_2, u_2, v_2)$, written $(\lambda_1, P_1, u_1, v_1) \prec (\lambda_2, P_2, u_2, v_2)$, if and only if λ_1 is a submask of λ_2, P_1 is a subprogram of P_2 and one of the following cases occurs:

1. $u_1 <_{\mathfrak{R}} u_2$ and $v_1 = v_2$ and P_1 is a suffix of P_2 and $u_1 P_1(w) v_1 = u_2 P_2(w) v_2$ for all safe completions w of λ_1;

2. $v_1 <_{\mathfrak{L}} v_2$ and $u_1 = u_2$ and P_1 is a prefix of P_2 and $u_1 P_1(w) v_1 = u_2 P_2(w) v_2$ for all safe completions w of λ_1;

3. $u_1 = u_2$ and $v_1 = 1$ and P_1 is a prefix of P_2 and $u_2 P_2(w) v_2 <_{\mathfrak{J}} u_1 P_1(w) v_1$ for all safe completions w of λ_1;

4. $v_1 = v_2$ and $u_1 = 1$ and P_1 is a suffix of P_2 and $u_2 P_2(w) v_2 <_{\mathfrak{J}} u_1 P_1(w) v_1$ for all safe completions w of λ_1.

Note that, since $(M, *)$ is finite, this relation is well-founded (that is, it has no infinite decreasing chain, an infinite sequence of quadruplets $\mu_0, \mu_1, \mu_2, \ldots$ such that $\mu_{i+1} \prec \mu_i$ for all $i \in \mathbb{N}$) and the maximal length of any decreasing chain depends only on $(M, *)$ (it is at most $2 \cdot |M|^2$). For a given quadruplet, we shall also call its height the length of the longest decreasing chain starting with that element minus 1.

To count the number of fixed positions, we define $f\colon \mathbb{N} \to \mathbb{N}$ by $f(m) = (4l-3)m + 2l-2$ for all $m \in \mathbb{N}$. It is obvious that f is non-decreasing. For all $h \in \mathbb{N}$, we will denote by $g_h = (2^h \cdot f)^{2^h}$ the function $(2^h \cdot f)\colon \mathbb{N} \to \mathbb{N}, m \mapsto 2^h f(m)$ composed 2^h times. It is direct to see that for each $h \in \mathbb{N}$, g_h is non-decreasing and that for each $h_1, h_2 \in \mathbb{N}, h_1 \leq h_2$, we have $g_{h_1}(m) \leq g_{h_2}(m)$ for all $m \in \mathbb{N}$. Finally, for a given mask λ, we denote by $|\lambda|_\Sigma$ the number of letters in λ belonging to Σ, that is to say, the number of fixed positions in λ.

The following lemma is the key to the proof. It shows that modulo fixing a few entries, one can fix the output.

Lemma 4.2.5. *Let λ be a Δ-compatible mask of length n, let P be a program over $(M, *)$ on Σ^n, let u and v be elements of M such that (λ, P, u, v) is of height h. Then there is an element t of M and a Δ-compatible submask λ' of λ obtained by fixing a number of free positions which is at most $g_h(|\lambda|_\Sigma + h) - |\lambda|_\Sigma$, such that any safe completion w of λ' verifies $uP(w)v = t$.*

Proof. The proof goes by induction on $\prec$.

Let λ be a Δ-compatible mask of length n, let P be a program over $(M, *)$ on Σ^n, let u and v be elements of M such that (λ, P, u, v) is of height h, and assume that for any quadruplet (λ', P', u', v') strictly smaller than (λ, P, u, v), the lemma is verified. Consider the following conditions concerning the quadruplet (λ, P, u, v):

(a) there does not exist any instruction (x, f) of P such that for some letter a the submask λ' of λ formed by setting position x to a (if it wasn't already the case) is Δ-compatible and $f(a)$ is $\mathfrak{R}$-bad for u;

(b) v is not $\mathfrak{R}$-bad for u;

(c) there does not exist any instruction (x, f) of P such that for some letter a the submask λ' of λ formed by setting position x to a (if it wasn't already the case) is Δ-compatible and $f(a)$ is $\mathfrak{L}$-bad for v;

(d) u is not $\mathfrak{L}$-bad for v.

We will now do a case analysis based on which of these conditions are violated or not.

Case 1: condition (a) is violated. So there exists some instruction (x, f) of P such that for some letter a the submask λ' of λ formed by setting position x to a (if it wasn't already the case) is Δ-compatible and $f(a)$ is $\mathfrak{R}$-bad for u. Let i be the smallest number of such an instruction.

Let P' be the subprogram of P until instruction $i-1$. Let w be a safe completion of λ: for any instruction (y, g) of P', as the submask λ'' of λ formed by setting position y to w_y (if it wasn't already the case) is Δ-compatible (by the fact that Δ is safe and w is a safe completion of λ), $g(w_y)$ cannot be $\mathfrak{R}$-bad for u, otherwise it would contradict the minimality of i, so $u \mathfrak{R} ug(w_y)$. Hence, by Lemma 4.1.1, $u \mathfrak{R} uP'(w)$ for all safe completions w of λ.

So, because $f(a)$ is $\mathfrak{R}$-bad for u, any safe completion w of λ', which is also a safe completion of λ, is such that $uP(w)v \leq_{\mathfrak{R}} uP'(w)f(a) <_{\mathfrak{R}} u \mathfrak{R} uP'(w)$ by Lemma 4.1.1, hence $uP(w)v <_{\mathfrak{J}} uP'(w)$ by Lemma 3.1.20. So $(\lambda', P', u, 1) \prec (\lambda, P, u, v)$, therefore, by induction we get a Δ-compatible submask λ_1 of λ' and a monoid element t_1 such that $uP'(w) = t_1$ for all safe completions w of λ_1.

Let P'' be the subprogram of P starting from instruction $i+1$. Notice that, since $u \mathfrak{R} t_1$ (by what we have proven just above), $t_1 f(a) <_{\mathfrak{R}} u$ (by Lemma 4.1.1) and $t_1 f(a) P''(w)v = uP'(w)f(a)P''(w)v = uP(w)v$ for all safe completions w of λ_1. Hence, $(\lambda_1, P'', t_1 f(a), v)$ is strictly smaller than (λ, P, u, v) and by induction we get a Δ-compatible submask λ_2 of λ_1 and a monoid element t such that $t_1 f(a) P''(w)v = t$ for all safe completions w of λ_2.

Thus, any safe completion w of λ_2 is such that

$$uP(w)v = uP'(w)f(a)P''(w)v = t_1 f(a)P''(w)v = t \ .$$

Therefore λ_2 and t form the desired couple of a Δ-compatible submask of λ and an element of M.

By induction, since $(\lambda', P', u, 1)$ is of height $h' \leq h-1$, the number of free positions of λ fixed in λ_1, i.e. $|\lambda_1|_\Sigma - |\lambda|_\Sigma$, is at most

$$\begin{aligned} |\lambda'|_\Sigma - |\lambda|_\Sigma + g_{h'}(|\lambda'|_\Sigma + h') - |\lambda'|_\Sigma &\leq g_{h'}(|\lambda|_\Sigma + h) - |\lambda|_\Sigma \\ &\leq g_{h-1}(|\lambda|_\Sigma + h) - |\lambda|_\Sigma \ , \end{aligned}$$

because of the properties of $g_{h'}$ and g_{h-1} spelled out before stating the lemma. Consequently, by induction again, as $(\lambda_1, P'', t_1 f(a), v)$ is of height $h'' \leq h-1$, the

number of free positions of λ fixed in λ_2, i.e. $|\lambda_2|_\Sigma - |\lambda|_\Sigma$, is at most

$$\begin{aligned}
|\lambda_1|_\Sigma - |\lambda|_\Sigma + g_{h''}(|\lambda_1|_\Sigma + h'') - |\lambda_1|_\Sigma &\leq g_{h-1}(|\lambda_1|_\Sigma + h'') - |\lambda|_\Sigma \\
&\leq g_{h-1}\big(g_{h-1}(|\lambda|_\Sigma + h) + h - 1\big) - |\lambda|_\Sigma \\
&\leq g_{h-1}\big((2^h \cdot f)^{2^{h-1}}(|\lambda|_\Sigma + h)\big) - |\lambda|_\Sigma \\
&\leq (2^h \cdot f)^{2^{h-1}}\big((2^h \cdot f)^{2^{h-1}}(|\lambda|_\Sigma + h)\big) - |\lambda|_\Sigma \\
&= (2^h \cdot f)^{2^h}(|\lambda|_\Sigma + h) - |\lambda|_\Sigma \\
&= g_h(|\lambda|_\Sigma + h) - |\lambda|_\Sigma \ ,
\end{aligned}$$

the desired bound, because of the properties of $g_{h'}$ and g_{h-1} mentioned before stating the lemma and because, as seen easily, $g_{h-1}(m) \leq 2g_{h-1}(m) = 2(2^{h-1} \cdot f)^{2^{h-1}}(m) \leq (2^h \cdot f)^{2^{h-1}}(m)$ for all $m \in \mathbb{N}$, as well as $h - 1 \leq f(|\lambda|_\Sigma + h) \leq g_{h-1}(|\lambda|_\Sigma + h)$.

Case 2: condition (a) is verified but condition (b) is violated, so v is $\mathfrak{R}$-bad for u and Case 1 does not apply.

Let w be a safe completion of λ: for any instruction (x, f) of P, as the submask λ' of λ formed by setting position x to w_x (if it wasn't already the case) is Δ-compatible (by the fact that Δ is safe and w is a safe completion of λ), $f(w_x)$ cannot be $\mathfrak{R}$-bad for u, otherwise condition (a) would be violated, so $u \ \mathfrak{R} \ uf(w_x)$. Hence, by Lemma 4.1.1, $u \ \mathfrak{R} \ uP(w)$ for all safe completions w of λ. Notice then that $uP(w)v <_\mathfrak{R} uP(w) \ \mathfrak{R} \ u$ (by Lemma 4.1.1), hence $uP(w)v <_\mathfrak{J} uP(w)$ (by Lemma 3.1.20) for all safe completions w of λ. So $(\lambda, P, u, 1) \prec (\lambda, P, u, v)$, therefore, by induction, we obtain a Δ-compatible submask λ' of λ and a monoid element t_1 such that $uP(w) = t_1$ for all safe completions w of λ'. If we set $t = t_1 v$, we get that any safe completion w of λ' is such that $uP(w)v = t_1 v = t$. Therefore λ' and t form the desired couple of a Δ-compatible submask of λ and an element of M.

By induction, since $(\lambda, P, u, 1)$ is of height $h' \leq h - 1$, the number of free positions of λ fixed in λ', i.e. $|\lambda'|_\Sigma - |\lambda|_\Sigma$, is at most

$$g_{h'}(|\lambda|_\Sigma + h') - |\lambda|_\Sigma \leq g_{h'}(|\lambda|_\Sigma + h) - |\lambda|_\Sigma \leq g_h(|\lambda|_\Sigma + h) - |\lambda|_\Sigma \ ,$$

the desired bound, because of the properties of $g_{h'}$ and g_h explicited before stating the lemma.

Case 3: condition (c) is violated. So there exists some instruction (x, f) of P such that for some letter a the submask λ' of λ formed by setting position x to a (if it wasn't

already the case) is Δ-compatible and $f(a)$ is $\mathfrak{L}$-bad for v.

We proceed as for Case 1 by symmetry.

Case 4: condition (c) is verified but condition (d) is violated, so u is $\mathfrak{L}$-bad for v and Case 3 does not apply.

We proceed as for Case 2 by symmetry.

Case 5: conditions (a), (b), (c) and (d) are verified.

As it was in Case 2 and Case 4, using Lemma 4.1.1, the fact that condition (a) and condition (c) are verified implies that $u \ \mathfrak{R} \ uP'(w)$ and $v \ \mathfrak{L} \ P''(w)v$ for any prefix P' of P, any suffix P'' of P and all safe completions w of λ. Moreover, since condition (b) and condition (d) are verified, by Lemma 4.1.1, we get that $uP(w)v \ \mathfrak{R} \ u$ and $uP(w)v \ \mathfrak{L} \ v$ for all safe completions w of λ. This implies that (λ, P, u, v) is minimal for $\prec$ and that $h = 0$.

Let w_0 be a completion of λ that is in Δ^*. Let λ' be the submask of λ fixing all free dangerous positions of λ using w_0. Then, for any completion w of λ', which is a safe completion of λ by construction, we have that $uP(w)v \ \mathfrak{R} \ u$ and $uP(w)v \ \mathfrak{L} \ v$. As $(M, *)$ is aperiodic and hence $\mathfrak{H}$-trivial, since we have $uP(w_0)v \in \mathfrak{R}(u) \cap \mathfrak{L}(v)$, it means that $\mathfrak{R}(u) \cap \mathfrak{L}(v) = \mathfrak{R}(uP(w_0)v) \cap \mathfrak{L}(uP(w_0)v) = \mathfrak{H}(uP(w_0)v)$, so by $\mathfrak{H}$-triviality of $(M, *)$, $\mathfrak{R}(u) \cap \mathfrak{L}(v) = \{t\}$ where $t = uP(w_0)v$, showing that $uP(w)v = t$ for all completions w of λ'. Therefore λ' and t form the desired couple of a Δ-compatible submask of λ and an element of M.

Now, since the number of free positions of λ fixed in λ', i.e. $|\lambda'|_\Sigma - |\lambda|_\Sigma$, is exactly the number of free dangerous positions in λ, and as a position in λ is dangerous if it is within distance $2l-2$ of a fixed position or within distance $l-1$ of the beginning or the end of λ, this number is at most

$$\begin{aligned} 2 \cdot |\lambda|_\Sigma \cdot (2l-2) + 2 \cdot (l-1) &= |\lambda|_\Sigma \cdot (4l-4) + 2l - 2 \\ &= |\lambda|_\Sigma \cdot (4l-3) + 2l - 2 - |\lambda|_\Sigma \\ &= f(|\lambda|_\Sigma) - |\lambda|_\Sigma \\ &= (2^0 \cdot f)^{2^0}(|\lambda|_\Sigma + 0) - |\lambda|_\Sigma \\ &= g_0(|\lambda|_\Sigma + 0) - |\lambda|_\Sigma \ , \end{aligned}$$

the desired bound.

This concludes the proof of the lemma. □

Setting $\Delta = \{c, ab\}$ or $\Delta = \{b, ab\}$ with Σ the associated alphabet, when applying Lemma 4.2.5 with the trivial Δ-compatible mask λ of length n containing only free positions, with P some program over $(M, *)$ on Σ^n and with u and v equal to 1, the resulting mask λ' has the property that we have an element t of M such that $P(w) = t$ for any safe completion w of λ'. Since the mask λ' is Δ-compatible and has a number of free positions upper-bounded by $g_h(|\lambda|_\Sigma + h) - |\lambda|_\Sigma$ where h is the height of (λ, P, u, v), a number that does not depend on n (because the maximal length of any decreasing chain depends only on $(M, *)$), as long as n is big enough, we have a safe completion $w_0 \in \Delta^*$ and a safe completion $w_1 \notin \Delta^*$. Hence P cannot recognise $\Delta^* \cap \Sigma^n$. This implies that $(c + ab)^* \notin \mathcal{P}((M, *))$ and $(b + ab)^* \notin \mathcal{P}((M, *))$. Finally, for any $k \in \mathbb{N}, k \geq 2$, we can prove that $b^*((ab^*)^k)^* \notin \mathcal{P}((M, *))$ by setting $\Delta = \{a, b\}$ and completing the mask given by the lemma by setting the letters in such a way that we have the right number of a's modulo k in one case and not in the other case.

This concludes the proof of Proposition 4.2.3 because the argument above holds for any monoid in **DA**.

4.3 A fine hierarchy in $\mathcal{P}(\mathbf{DA})$

The definition of p-recognition by a sequence of programs over a finite monoid given in Definition 3.3.5 requires that for each $n \in \mathbb{N}$, the program working on words of length n has a length upper-bounded by some fixed polynomial in n. In the case of $\mathcal{P}(\mathbf{DA})$, the polynomial-length restriction is without loss of generality: any monoid $(M, *)$ in **DA** has what Tesson and Thérien call the *polynomial-length property* in Tesson and Thérien [2001], that is, there exists $k \in \mathbb{N}$ such that for any alphabet Σ there is a constant $c \in \mathbb{N}_{>0}$ such that any $(M, *)$-program on Σ^n for $n \in \mathbb{N}$ is equivalent to an $(M, *)$-program on Σ^n of length at most $c \cdot n^k$ (which they proved in the same article). In this section, we prove two things: first, that the exponent k can be arbitrarily large depending on the monoid $(M, *)$, so that while $\mathcal{P}(\mathbf{DA}, s(n))$ collapses to $\mathcal{P}(\mathbf{DA})$ for any super-polynomial function $s\colon \mathbb{N} \to \mathbb{N}$, there does not exist any $k \in \mathbb{N}$ such that $\mathcal{P}(\mathbf{DA})$ collapses to $\mathcal{P}\big(\mathbf{DA}, n^k\big)$; and second, that when we restrict to $\mathbf{DA_d}$ for some $d \in \mathbb{N}$, in this case $\mathcal{P}(\mathbf{DA_d})$ collapses to $\mathcal{P}\big(\mathbf{DA_d}, n^{\max\{d,1\}}\big)$ and this is optimal.

4.3.1 Strict hierarchy

For each $k \in \mathbb{N}_{>0}$, we exhibit a language $L_k \subseteq \{0, 1\}^*$ that can be recognised by a sequence of programs of length $\mathrm{O}(n^k)$ over a monoid $(M_k, *_k)$ in $\mathbf{DA_k}$ but cannot be recognised by

any sequence of programs of length $\mathrm{O}(n^{k-1})$ over any monoid in **DA**.

For a given $k \in \mathbb{N}_{>0}$, the language L_k expresses a property of the first k occurrences of 1 for words over $\{0,1\}$. To define L_k, we say that for some $n \in \mathbb{N}$, S is a *k-set over n* if S is a set where each element is an ordered tuple of k distinct elements of $[n]$. For any sequence $\Delta = (S_n)_{n\in\mathbb{N}}$ such that for each $n \in \mathbb{N}$, S_n is a k-set over n — a sequence we will call a *sequence of k-sets* — we set $L_\Delta = \bigcup_{n\in\mathbb{N}} K_{n,S_n}$, where for each $n \in \mathbb{N}$, K_{n,S_n} is the set of words over $\{0,1\}$ of length n such that for each of them, it contains at least k occurrences of 1 and the ordered k-tuple of the positions of the first k occurrences of 1 belongs to S_n.

On the one hand, we show that for all $k \in \mathbb{N}_{>0}$ there is a monoid $(M_k, *_k)$ in $\mathbf{DA_k}$ such that for any sequence of k-sets Δ, the language L_Δ is recognised by a sequence of programs over $(M_k, *_k)$ of length $\mathrm{O}(n^k)$. The proof is done by an inductive argument on k.

On the other hand, we show that for all $k \in \mathbb{N}_{>0}$ there is a sequence of k-sets Δ such that for any finite monoid $(M, *)$ and any sequence of programs $(P_n)_{n\in\mathbb{N}}$ over $(M, *)$ of length $\mathrm{O}(n^{k-1})$, L_Δ is not recognised by $(P_n)_{n\in\mathbb{N}}$. This is done using a counting argument: for some monoid order $i \in \mathbb{N}_{>0}$, for $n \in \mathbb{N}$ big enough, the number of languages in $\{0,1\}^n$ recognised by a program over some monoid of order i on $\{0,1\}^n$ of length at most $\alpha \cdot n^{k-1}$ for $\alpha \in \mathbb{N}_{>0}$ some constant is upper-bounded by a number that turns out to be smaller than the number of different possible K_{n,S_n}, when the k-set S_n varies.

Upper bound. We start with the upper bound. Given $k \in \mathbb{N}_{>0}$, we define the alphabet $Y_k = \{e\} \cup \{\bot_l, \top_l \mid l \in [k]\}$; we are going to prove that for all $k \in \mathbb{N}_{>0}$ there exists a language $Z_k \in \mathcal{SUL}_k(Y_k^*)$ such that for all $\Delta = (S_n)_{n\in\mathbb{N}}$ sequences of k-sets, there exists a program-reduction from L_Δ to Z_k of length $s(n)$, where $s\colon \mathbb{N} \to \mathbb{N}$ is some function verifying $s(n) \leq 2n^k$ for all $n \in \mathbb{N}$, using the following proposition and the fact that the language of words of length $n \in \mathbb{N}$ of L_Δ is exactly K_{n,S_n}.

Proposition 4.3.1. *For all $k \in \mathbb{N}_{>0}$ there is a language $Z_k \in \mathcal{SUL}_k(Y_k^*)$ such that for all $n \in \mathbb{N}$ and all k-sets S_n over n, we have $K_{n,S_n} = \Psi_{n,S_n}^{-1}(Z_k^{=|\Psi_{n,S_n}|})$ where Ψ_{n,S_n} is a Y_k-program on $\{0,1\}^n$ of length at most $2n^k$.*

Proof. We first define by induction on k a family of languages Z_k over the alphabet Y_k. For $k = 0$, Z_0 is Y_0^* where $Y_0 = \{e\}$. For $k \in \mathbb{N}_{>0}$, Z_k is the set of words containing $\top_k$ and such that the first occurrence of $\top_k$ has no $\bot_k$ to its left, and the sequence between the first occurrence of $\top_k$ and the first occurrence of $\bot_k$ or $\top_k$ to its right, or the end of the word if there is no such letter, belongs to Z_{k-1}. A simple induction on k shows that

Z_k for $k \in \mathbb{N}$ is defined by the expression

$$Y_{k-1}^* \top_k Y_{k-2}^* \top_{k-1} \cdots Y_1^* \top_2 Y_0^* \top_1 Y_k^* \ ,$$

and therefore it is in $\mathcal{SUM}_k(Y_k^*) \subseteq \mathcal{SUL}_k(Y_k^*)$.

Fix $n \in \mathbb{N}$. If $n = 0$, the proposition follows trivially; otherwise, we define by induction on k a Y_k-program $P_k(i, S)$ on $\{0,1\}^n$ for every k-set S over n and every $i \in [n+1]$.

For any $k \in \mathbb{N}_{>0}$, $j \in [n]$ and S a k-set over n, let $f_{j,S} \colon \{0,1\} \to Y_k$ be the function defined by $f_{j,S}(0) = e$ and $f_{j,S}(1) = \top_k$ if j is the first element of some ordered k-tuple of S, $f_{j,S}(1) = \bot_k$ otherwise. We also let $g_k \colon \{0,1\} \to Y_k$ be the function defined by $g_k(0) = e$ and $g_k(1) = \bot_k$. Moreover, $S|j$ denotes the $(k-1)$-set over n containing the ordered $(k-1)$-tuples $\bar{t}$ such that $(j, \bar{t}) \in S$.

For $k \in \mathbb{N}_{>0}$, $i \in [n+1]$ and S a k-set over n, the Y_k-program $P_k(i, S)$ on $\{0,1\}^n$ is the following sequence of instructions:

$$(i, f_{i,S}) P_{k-1}(i+1, S|i)(i, g_k) \cdots (n, f_{n,S}) P_{k-1}(n+1, S|n)(n, g_k) \ .$$

In other words, the program guesses the first occurrence $j \in [\![i, n]\!]$ of 1, returns $\bot_k$ or $\top_k$ depending on whether it is the first element of an ordered k-tuple in S, and then proceeds for the next occurrences of 1 by induction.

For $k = 0$, $i \in [n+1]$ and S a 0-set over n (that is empty or contains ε, the only ordered 0-tuple of elements of $[n]$), the Y_0-program $P_0(i, S)$ is the empty program ε.

A simple computation shows that for any $k \in \mathbb{N}$, $i \in [n+1]$ and S a k-set over n, the number of instructions in $P_k(i, S)$ is at most $2n^k$ and even 0 when $i = n+1$.

A simple induction on k shows that when running on a word $w \in \{0,1\}^n$, for any $k \in \mathbb{N}$, $i \in [n+1]$ and S a k-set over n, $P_k(i, S)$ returns a word in Z_k if and only if $k = 0$ or the ordered k-tuple of the positions of the first k occurrences of 1 starting at position i in w exists and is an element of S.

Therefore, for any $k \in \mathbb{N}_{>0}$ and S_n a k-set over n, if we set $\Psi_{n,S_n} = P_k(1, S_n)$, we have $K_{n,S_n} = \Psi_{n,S_n}^{-1}(Z_k^{=|\Psi_{n,S_n}|})$ where Ψ_{n,S_n} is a Y_k-program on $\{0,1\}^n$ of length at most $2n^k$. $\square$

Consequently, for all $k \in \mathbb{N}_{>0}$ and any sequence of k-sets Δ, since the language Z_k is in $\mathcal{SUL}_k(Y_k^*)$ and thus recognised by a monoid from $\mathbf{DA_k}$, we have, by Corollary 3.4.3, that $L_\Delta \in \mathcal{P}\big(\mathbf{DA_k}, n^k\big)$, as there is a program-reduction from it to Z_k of length $s(n)$, where $s \colon \mathbb{N} \to \mathbb{N}$ is some function verifying $s(n) \leq 2n^k$ for all $n \in \mathbb{N}$.

Lower bound. The following claim is a simple counting argument.

Claim 4.3.2. *For all $i \in \mathbb{N}_{>0}$ and $n \in \mathbb{N}$, the number of languages in $\{0,1\}^n$ recognised by programs over a monoid of order i on $\{0,1\}^n$, with at most $l \in \mathbb{N}$ instructions, is upper-bounded by $i^{i^2}2^i \cdot (n \cdot i^2)^l$.*

Proof. Fix a monoid $(M, *)$ of order i. Since a program over $(M, *)$ on $\{0,1\}^n$ with less than l instructions can always be completed into an equivalent such program with exactly l instructions (using the identity of $(M, *)$), we only consider programs with exactly l instructions. As $\Sigma = \{0,1\}$, there are $n \cdot i^2$ choices for each of the l instructions of an $(M, *)$-program on $\{0,1\}^n$. Such a program can recognise at most 2^i different languages in $\{0,1\}^n$. Hence, the number of languages in $\{0,1\}^n$ recognised by $(M, *)$-programs on $\{0,1\}^n$ of length at most l is at most $2^i \cdot (n \cdot i^2)^l$. The result follows from the facts that there are at most i^{i^2} isomorphism classes of monoids of order i and that two isomorphic monoids allow to recognise the same languages in $\{0,1\}^n$ through programs. □

If for some $k \in \mathbb{N}_{>0}$ and $i \in [\alpha]$, $\alpha \in \mathbb{N}_{>0}$, we apply Claim 4.3.2 for all $n \in \mathbb{N}$, $l = \alpha \cdot n^{k-1}$, we get a number $\mu_i(n)$ of languages that is in $2^{\mathrm{O}(n^{k-1}\log_2(n))}$, which is asymptotically strictly smaller than the number of distinct K_{n,S_n} when the k-set S_n over n varies, which is $2^{\binom{n}{k}}$, i.e. $\mu_i(n)$ is in $\mathrm{o}\big(2^{\binom{n}{k}}\big)$.

Hence, for all $j \in \mathbb{N}_{>0}$, there exist an $n_j \in \mathbb{N}$ and T_j a k-set over n_j such that no program over a monoid of order $i \in [j]$ on $\{0,1\}^{n_j}$ and of length at most $j \cdot {n_j}^{k-1}$ recognises K_{n_j,T_j}. Moreover, we can assume without loss of generality that the sequence $(n_j)_{j\in\mathbb{N}_{>0}}$ is increasing. Let $\Delta = (S_n)_{n\in\mathbb{N}}$ be such that $S_{n_j} = T_j$ for all $j \in \mathbb{N}_{>0}$ and $S_n = \emptyset$ for any $n \in \mathbb{N}$ verifying that it is not equal to any n_j for $j \in \mathbb{N}_{>0}$. We show that no sequence of programs over a finite monoid of length $\mathrm{O}(n^{k-1})$ can recognise L_Δ. If this were the case, then let i be the order of the monoid. Let $j \in \mathbb{N}, j \geq i$ be such that for any $n \in \mathbb{N}$, the n-th program has length at most $j \cdot n^{k-1}$. But, by construction, we know that there does not exist any such program on $\{0,1\}^{n_j}$ recognising K_{n_j,T_j}, a contradiction.

This implies the following hierarchy, using the fact that for all $k \in \mathbb{N}$ and all $d \in \mathbb{N}, d \leq \max\{k-1, 0\}$, any monoid from $\mathbf{DA_d}$ is also a monoid from $\mathbf{DA_k}$, observing that $a^* \in \mathcal{P}(\{a,b\}^*, \mathbf{DA_0}, n) \setminus \mathcal{P}(\{a,b\}^*, \mathbf{DA_0}, 1)$ simply because any program over some finite monoid $(M, *)$ on $\{a,b\}^n$ for $n \in \mathbb{N}$ recognising a^n must have at least n instructions, one for each input letter.

Proposition 4.3.3. *For all $k \in \mathbb{N}$, $\mathcal{P}\big(\mathbf{DA}, n^k\big) \subset \mathcal{P}\big(\mathbf{DA}, n^{k+1}\big)$. More precisely, for all $k \in \mathbb{N}$ and $d \in \mathbb{N}, d \leq \max\{k-1, 0\}$, $\mathcal{P}\big(\mathbf{DA_k}, n^d\big) \subset \mathcal{P}\big(\mathbf{DA_k}, n^{d+1}\big)$.*

4.3.2 Collapse

Tesson and Thérien showed that for any monoid $(M, *)$ in $\mathbf{DA}$, there exists $k \in \mathbb{N}$ such that for any alphabet Σ there is a constant $c \in \mathbb{N}_{>0}$ such that any $(M, *)$-program on Σ^n for $n \in \mathbb{N}$ is equivalent to an $(M, *)$-program on Σ^n of length at most $c \cdot n^k$ Tesson and Thérien [2001]. We now show that if we further assume that $(M, *)$ is in $\mathbf{DA_d}$ for some $d \in \mathbb{N}$, then the exponent k can be assumed to be equal to $\max\{d, 1\}$ — but not less by Proposition 4.3.3. This shows that proposition to be optimal in some sense.

Proposition 4.3.4. *Let $k \in \mathbb{N}$. Let $(M, *) \in \mathbf{DA_k}$ and Σ be an alphabet. Then there exists a constant $c \in \mathbb{N}_{>0}$ such that any program over $(M, *)$ on Σ^n for $n \in \mathbb{N}$ is equivalent to a program over $(M, *)$ on Σ^n of length at most $c \cdot n^{\max\{k,1\}}$.*

In particular, $\mathcal{P}(\mathbf{DA_k}) = \mathcal{P}\big(\mathbf{DA_k}, n^{\max\{k,1\}}\big)$ for all $k \in \mathbb{N}$.

Observe that given P a program over some finite monoid $(M, *)$ on Σ^n for $n \in \mathbb{N}$ and Σ an alphabet, given some $F \subseteq M$, we have

$$P^{-1}(F) = \xi_P^{-1}\big(\eta_{(M,*)}^{-1}(F) \cap M^{|P|}\big)$$

where $\eta_{(M,*)}$ is the evaluation morphism of $(M, *)$. Therefore, a language $L \subseteq \Sigma^n$ is recognised by P if and only if there exists a language $K \subseteq M^*$ recognised by $\eta_{(M,*)}$ verifying that $L = \xi_P^{-1}(K^{=|P|})$, i.e. $w \in \Sigma^n$ belongs to L if and only if $\xi_P(w) \in K$. Thus, a subprogram P' of P is equivalent to P if and only if for every language $K \subseteq M^*$ recognised by $\eta_{(M,*)}$ we have $\xi_P(w) \in K \Leftrightarrow \xi_{P'}(w) \in K$ for all $w \in \Sigma^n$. Moreover, every language recognised by $\eta_{(M,*)}$ is precisely a language of $\mathcal{SUL}_k(M^*)$ when $(M, *) \in \mathbf{DA_k}$ for some $k \in \mathbb{N}$.

The result is hence a consequence of the following lemma and the fact that every language in $\mathcal{SUL}_k(M^*)$ is a Boolean combination of languages in $\mathcal{SUM}_k(M^*)$, that is a finite set of languages.

Lemma 4.3.5. *Let Σ be an alphabet and $(M, *)$ a finite monoid.*

For all $k \in \mathbb{N}$, there exists a constant $c \in \mathbb{N}_{>0}$ verifying that for any program P over $(M, *)$ on Σ^n for $n \in \mathbb{N}$ and any SUM $K \in \mathcal{SUM}_k(M^*)$ of degree k, there exists a subprogram Q of P of length at most $c \cdot n^{\max\{k,1\}}$ such that for any subprogram Q' of P that has Q as a subprogram, for all $w \in \Sigma^n$ we have

$$\xi_P(w) \in K \Leftrightarrow \xi_{Q'}(w) \in K \ .$$

Proof. A program P over $(M, *)$ on Σ^n for $n \in \mathbb{N}$ is a finite sequence (p_i, f_i) of instructions where each p_i is a positive natural number which is at most n and each f_i is a function from Σ to M. We denote by l the number of instructions of P. For each set $I \subseteq [l]$ we denote by $P[I]$ the subprogram of P consisting of the subsequence of instructions of P obtained after removing all instructions whose index is not in I. In particular, $P[1, m]$ denotes the initial sequence of instructions of P, until instruction number m.

We prove the lemma by induction on k, fixing the constant to be $c_k = |M| \cdot 3^k \cdot |\Sigma|^{\max\{k,1\}}$ for a given $k \in \mathbb{N}$.

The intuition behind the proof for a program P on inputs of length n and some SUM $K_1 \gamma K_2$ of degree at least 2 is as follows. We assume that K_1 does not contain any word with the letter γ, the other case is done symmetrically. Consider the subset of all indices $I_\gamma \subseteq [l]$ that correspond, for a fixed letter a and a fixed position p in the input, to the first instruction of P that would output the element γ when reading a at position p. We then have that, given some w as input, $\xi_P(w) \in K_1 \gamma K_2$ if and only if there exists $i \in I_\gamma$ verifying that the element at position i of $\xi_P(w)$ is γ, $\xi_{P[1,i-1]}(w) \in K_1$ and $\xi_{P[i+1,l]}(w) \in K_2$. The idea is then that if we set I to contain I_γ as well as all indices obtained by induction for $P[1, i-1]$ and K_1 and for $P[i+1, l]$ and K_2, we would have that for all w, $\xi_P(w) \in K_1 \gamma K_2$ if and only if $\xi_{P[I]}(w) \in K_1 \gamma K_2$, that is $\xi_P(w)$ where only the elements at indices in I have been kept.

The intuition behind the proof when the SUM is of degree less than 2 is essentially the same, but without induction.

We now spell out the details of the proof, starting with the inductive step.

Inductive step. Let $k \in \mathbb{N}, k \geq 2$ and assume the lemma proven for all $k' \in \mathbb{N}, k' < k$. Let P be a program over $(M, *)$ on Σ^n for $n \in \mathbb{N}$ of length $l \in \mathbb{N}$ and some language K in $\mathcal{SUM}_k(M^*) \setminus \mathcal{SUM}_{k-1}(M^*)$. By definition, $K = K_1 \gamma K_2$ for $\gamma \in M$ and some SUMs $K_1 \in \mathcal{SUM}_{k_1}(M^*)$ of degree k_1 and $K_2 \in \mathcal{SUM}_{k_2}(M^*)$ of degree k_2 with $k_1 + k_2 = k - 1$. Moreover either γ does not occur in any of the words of K_1 or it does not occur in any of the words of K_2. We only treat the case where γ does not appear in any of the words in K_1, the other case is treated similarly by symmetry.

Observe that when $n = 0$, we necessarily have $P = \varepsilon$, so that the lemma is trivially proven in that case. So we now assume $n > 0$.

For each $p \in [n]$ and each $a \in \Sigma$ consider within the sequence of instructions of P the first instruction of the form (p, f) with $f(a) = \gamma$, if it exists. We let I_γ be the set of indices of these instructions for all a and p. Notice that the size of I_γ is at most $|\Sigma| \cdot n$.

For all $i \in I_\gamma$, we let $J_{i,1}$ be the set of indices of the instructions within $P[1, i-1]$

appearing in its subprogram obtained by induction for $P[1, i-1]$ and K_1, and $J_{i,2}$ be the same for $P[i+1, l]$ and K_2.

We now let I be the union of I_γ and $J_{i,1}$ and $J'_{i,2} = \{j + i \mid j \in J_{i,2}\}$ for all $i \in I_\gamma$ (the translation being required because the first instruction in $P[i+1, l]$ is the $(i+1)$-th instruction in P). We claim that $Q = P[I]$, a subprogram of P, has the desired properties.

First notice that by induction the sizes of $J_{i,1}$ for all $i \in I_\gamma$ are upper bounded by

$$\begin{aligned} |M| \cdot 3^{k_1} \cdot |\Sigma|^{\max\{k_1,1\}} \cdot n^{\max\{k_1,1\}} &\leq |M| \cdot 3^{k-1} \cdot |\Sigma|^{\max\{k-1,1\}} \cdot n^{\max\{k-1,1\}} \\ &= c_{k-1} \cdot n^{\max\{k-1,1\}} \ , \end{aligned}$$

as well as are those of $J'_{i,2}$ for all $i \in I_\gamma$. Hence, since $n \geq 1$ and $k - 1 \geq 1$, the size of I is at most

$$\begin{aligned} |I_\gamma| + \sum_{i \in I_\gamma} |J_{i,1}| + \sum_{i \in I_\gamma} \left|J'_{i,2}\right| &\leq |I_\gamma| + |I_\gamma| \cdot c_{k-1} \cdot n^{\max\{k-1,1\}} + |I_\gamma| \cdot c_{k-1} \cdot n^{\max\{k-1,1\}} \\ &\leq 3 \cdot |I_\gamma| \cdot c_{k-1} \cdot n^{\max\{k-1,1\}} \\ &\leq 3 \cdot |\Sigma| \cdot n \cdot |M| \cdot 3^{k-1} \cdot |\Sigma|^{\max\{k-1,1\}} \cdot n^{\max\{k-1,1\}} \\ &= |M| \cdot 3^k \cdot |\Sigma|^{\max\{k,1\}} \cdot n^{\max\{k,1\}} \\ &= c_k \cdot n^{\max\{k,1\}} \ , \end{aligned}$$

so that $P[I]$ has at most the required length.

Let Q' be a subprogram of P that has Q as a subprogram: it means that there exists some set $I' \subseteq [l]$ containing I such that $Q' = P[I']$.

Take $w \in \Sigma^n$.

Assume now that $\xi_P(w) \in K$. Let i be the position in $\xi_P(w)$ of label γ witnessing the membership in K. Let (p_i, f_i) be the corresponding instruction of P. In particular we have that $f_i(w_{p_i}) = \gamma$. Because γ does not occur in any word of K_1, for all $j \in [i-1]$ such that $p_j = p_i$ we cannot have $f_j(w_{p_j}) = \gamma$. Hence $i \in I_\gamma$. By induction we have that $\xi_{P[1,i-1][J]}(w) \in K_1$ for any set $J \subseteq [i-1]$ containing $J_{i,1}$ and $\xi_{P[i+1,l][J]}(w) \in K_2$ for any set $J \subseteq [l-i]$ containing $J_{i,2}$. Hence, if we set $I'_1 = \{j \in I' \mid j < i\}$ as the subset of I' of elements less than i and $I'_2 = \{j - i \mid j \in I', j > i\}$ as the subset of I' of elements greater than i translated by $-i$, we have $\xi_{P[I']}(w) = \xi_{P[1,i-1][I'_1]}(w)\gamma\xi_{P[i+1,l][I'_2]}(w) \in K_1\gamma K_2 = K$.

Assume finally that $\xi_{P[I']}(w) \in K$. Let i be the index in I' whose instruction provides the letter γ witnessing the fact that $\xi_{P[I']}(w) \in K$. This means that if we set $I'_1 = \{j \in I' \mid j < i\}$ as the subset of I' of elements less than i and $I'_2 = \{j - i \mid j \in I', j > i\}$ as the subset of I' of elements greater than i translated by $-i$, we have $\xi_{P[I']}(w) =$

$\xi_{P[1,i-1][I'_1]}(w)\gamma\xi_{P[i+1,l][I'_2]}(w)$ with $\xi_{P[1,i-1][I'_1]}(w) \in K_1$ and $\xi_{P[i+1,l][I'_2]}(w) \in K_2$. If $i \in I_\gamma$, then it means that $I'_1 \subseteq [i-1]$ contains $J_{i,1}$ and that $I'_2 \subseteq [l-i]$ contains $J_{i,2}$ by construction, so that, by induction, $\xi_P(w) = \xi_{P[1,i-1]}(w)\gamma\xi_{P[i+1,l]}(w) \in K_1\gamma K_2 = K$. If not this shows that there is an instruction (p_j, f_j) with $j \in [i-1]$, $j \in I'$, $p_j = p_i$ and $f_j(w_{p_j}) = \gamma$. But that would contradict the fact that γ cannot occur in K_1.

Therefore, $\xi_P(w) \in K \Leftrightarrow \xi_{Q'}(w) = \xi_{P[I']}(w) \in K$, as desired.

Base case. There are two subcases to consider.

Subcase $k = 1$. Let P be a program over $(M, *)$ on Σ^n for $n \in \mathbb{N}$ of length $l \in \mathbb{N}$ and some language K in $\mathcal{SUM}_1(M^*) \setminus \mathcal{SUM}_0(M^*)$.

Then $K = A_1^*\gamma A_2^*$ for $\gamma \in M$ and some alphabets $A_1 \subseteq M$ and $A_2 \subseteq M$. Moreover either $\gamma \notin A_1$ or $\gamma \notin A_2$. We only treat the case where γ does not belong to A_1, the other case is treated similarly by symmetry.

We use the same idea as in the inductive step.

Observe that when $n = 0$, we necessarily have $P = \varepsilon$, so that the lemma is trivially proven in that case. So we now assume $n > 0$.

For each $p \in [n]$, each $\alpha \in M$ and $a \in \Sigma$ consider within the sequence of instructions of P the first and last instruction of the form (p, f) with $f(a) = \alpha$, if they exist. We let I be the set of indices of these instructions for all a, α and p. Notice that the size of I is at most $2 \cdot |M| \cdot |\Sigma| \cdot n \leq |M| \cdot 3^1 \cdot |\Sigma|^{\max\{1,1\}} \cdot n^{\max\{1,1\}} = c_1 \cdot n^{\max\{1,1\}}$.

We claim that $Q = P[I]$, a subprogram of P, has the desired properties. We just showed it has at most the required length.

Let Q' be a subprogram of P that has Q as a subprogram: it means that there exists some set $I' \subseteq [l]$ containing I such that $Q' = P[I']$.

Take $w \in \Sigma^n$.

Assume now that $\xi_P(w) \in K$. Let i be the position in $\xi_P(w)$ of label γ witnessing the membership in K. Let (p_i, f_i) be the corresponding instruction of P. In particular we have that $f_i(w_{p_i}) = \gamma$ and this is the γ witnessing the membership in K. Because $\gamma \notin A_1$, for all $j \in [i-1]$ such that $p_j = p_i$ we cannot have $f_j(w_{p_j}) = \gamma$. Hence $i \in I \subseteq I'$. From $\xi_{P[1,i-1]}(w) \in A_1^*$ and $\xi_{P[i+1,l]}(w) \in A_2^*$ it follows that $\xi_{P[I' \cap [\![1,i-1]\!]]}(w) \in A_1^*$ and $\xi_{P[I' \cap [\![i+1,l]\!]]}(w) \in A_2^*$, showing that $\xi_{P[I']}(w) = \xi_{P[I' \cap [\![1,i-1]\!]]}(w)\gamma\xi_{P[I' \cap [\![i+1,l]\!]]}(w) \in K$.

Assume finally that $\xi_{P[I']}(w) \in K$. Let i be the index in I' whose instruction provides the letter γ witnessing the fact that $\xi_{P[I']}(w) \in K$. This means that $\xi_{P[I' \cap [\![1,i-1]\!]]}(w) \in A_1^*$ and $\xi_{P[I' \cap [\![i+1,l]\!]]}(w) \in A_2^*$. If there is an instruction (p_j, f_j), with $j \in [i-1]$ and $f_j(w_{p_j}) \notin A_1$ then either $j \in I'$ and we get a direct contradiction with the fact that $\xi_{P[I' \cap [\![1,i-1]\!]]}(w) \in A_1^*$,

or $j \notin I'$ and we get a smaller $j' \in I \subseteq I'$ with the same property, contradicting again the fact that $\xi_{P[I' \cap [\![1,i-1]\!]]}(w) \in A_1^*$. Hence, for all $j \in [i-1]$, $f_j(w_{p_j}) \in A_1$. By symmetry we have that for all $j \in [\![i+1, l]\!]$, $f_j(w_{p_j}) \in A_2$, showing that $\xi_P(w) \in A_1^* \gamma A_2^* = K$.

Therefore, $\xi_P(w) \in K \Leftrightarrow \xi_{Q'}(w) = \xi_{P[I']}(w) \in K$, as desired.

Subcase $k = 0$**.** Let P be a program over $(M, *)$ on Σ^n for $n \in \mathbb{N}$ of length $l \in \mathbb{N}$ and some language K in $\mathcal{SUM}_0(M^*)$.

Then $K = A^*$ for some alphabet $A \subseteq M$.

We again use the same idea as before.

Observe that when $n = 0$, we necessarily have $P = \varepsilon$, so that the lemma is trivially proven in that case. So we now assume $n > 0$.

For each $p \in [n]$, each $\alpha \in M$ and $a \in \Sigma$ consider within the sequence of instructions of P the first instruction of the form (p, f) with $f(a) = \alpha$, if it exists. We let I be the set of indices of these instructions for all a, α and p. Notice that the size of I is at most $|M| \cdot |\Sigma| \cdot n = |M| \cdot 3^0 \cdot |\Sigma|^{\max\{0,1\}} \cdot n^{\max\{0,1\}} = c_0 \cdot n^{\max\{0,1\}}$.

We claim that $Q = P[I]$, a subprogram of P, has the desired properties. We just showed it has at most the required length.

Let Q' be a subprogram of P that has Q as a subprogram: it means that there exists some set $I' \subseteq [l]$ containing I such that $Q' = P[I']$.

Take $w \in \Sigma^n$.

Assume now that $\xi_P(w) \in K$. As $\xi_{P[I']}(w)$ is a subword of $\xi_P(w)$, it follows directly that $\xi_{P[I']}(w) \in A^* = K$.

Assume finally that $\xi_{P[I']}(w) \in K$. If there is an instruction (p_j, f_j), with $j \in [l]$ and $f_j(w_{p_j}) \notin A$ then either $j \in I'$ and we get a direct contradiction with the fact that $\xi_{P[I']}(w) \in A^* = K$, or $j \notin I'$ and we get a smaller $j' \in I \subseteq I'$ with the same property, contradicting again the fact that $\xi_{P[I']}(w) \in A^* = K$. Hence, for all $j \in [l]$, $f_j(w_{p_j}) \in A$, showing that $\xi_P(w) \in A^* = K$.

Therefore, $\xi_P(w) \in K \Leftrightarrow \xi_{Q'}(w) = \xi_{P[I']}(w) \in K$, as desired. □

Chapter 5

The power of programs over monoids in J

The last chapter of this thesis is similar in its objectives to the previous one, except that we focus on p-recognition by monoids taken from the variety of finite monoids $\mathbf{J}$. It is another well-known "small" variety of finite aperiodic monoids, that is even strictly included in $\mathbf{DA}$. As explained in Chapter 3, $\mathbf{J}$ is especially interesting because it is not an sp-variety of finite monoids, and we spend most of this long chapter trying to understand the unexpected power p-recognition offers over classical morphism-recognition by monoids from $\mathbf{J}$ when it comes to regular languages.

Section 5.1 first presents some specific preliminaries about $\mathbf{J}$. Section 5.2 is then dedicated to a fine hierarchy result inside $\mathcal{P}(\mathbf{J})$ based on the usual parameterisation of $\mathbf{J}$, very similar to the fine hierarchy result for $\mathbf{DA}$. The remainder of this chapter is then devoted to the study of $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg}$: in Section 5.3, we investigate the possibilities offered by p-recognition by monoids from $\mathbf{J}$ and show that, actually, a whole strict superclass of languages of $\mathcal{L}(\mathbf{J})$ can be recognised that way. We conjecture that this class essentially suffices to characterise $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg}$ and Section 5.4 presents a partial result that supports this conjecture. The proof we give is, alas, long and complicated.

5.1 Specific preliminaries about J

As we did in the previous chapter, we also start this one with some specific definitions and results about $\mathbf{J}$ that we will use, based essentially on Klíma and Polák [2010], but also on [Pin, 2016, Chapter XI].

As we have seen in Section 3.1 of Chapter 3, $\mathbf{J} \subset \mathbf{DA} \subset \mathbf{A}$. (This is because

while any finite $\mathfrak{J}$-trivial monoid $(M, *)$ definitely verifies that each of its $\mathfrak{D}$-classes whose sole element is an idempotent forms an aperiodic subsemigroup of $(M, *)$, the monoid $\mathrm{M}(a(a+b)^*)$ in **DA** is not $\mathfrak{J}$-trivial.) Its equational characterisation is the following: a finite monoid $(M, *)$ of idempotent power ω belongs to **J** if and only if $(xy)^\omega = (xy)^\omega x = y(xy)^\omega$ for all $x, y \in M$.

For each $k \in \mathbb{N}$ and each alphabet Σ, let us define the equivalence relation $\sim_k^\Sigma$ on Σ^* by $u \sim_k^\Sigma v$ if and only if u and v have the same set of k-subwords, for all $u, v \in \Sigma^*$. $\sim_k^\Sigma$ is a congruence of finite index on Σ^*.

We define the *variety of piecewise testable languages* $\mathcal{PT}$ as the class of regular languages such that for every alphabet Σ, $\mathcal{PT}(\Sigma^*)$ is the set of all languages over Σ that are Boolean combinations of languages of the form $\Sigma^* a_1 \Sigma^* a_2 \Sigma^* \cdots \Sigma^* a_l \Sigma^*$ (i.e. the language of all words over Σ having $a_1 a_2 \cdots a_l$ as a subword) where $a_1, a_2, \ldots, a_l \in \Sigma$ and $l \in \mathbb{N}$. In fact, for each alphabet Σ, $\mathcal{PT}(\Sigma^*)$ is the set of languages over Σ equal to a union of $\sim_k^\Sigma$-classes for some $k \in \mathbb{N}$ (see Simon [1975]). Simon showed Simon [1975] that a language is piecewise testable if and only if its syntactic monoid is $\mathfrak{J}$-trivial, i.e. $\mathcal{PT} = \mathcal{L}(\mathbf{J})$.

We can define a hierarchy of piecewise testable languages in a natural way. For $k \in \mathbb{N}$, let the *variety of k-piecewise testable languages* $\mathcal{PT}_k$ be the class of regular languages such that for every alphabet Σ, $\mathcal{PT}_k(\Sigma^*)$ is the set of all languages over Σ that are Boolean combinations of languages of the form $\Sigma^* a_1 \Sigma^* a_2 \Sigma^* \cdots \Sigma^* a_l \Sigma^*$ where $a_1, a_2, \ldots, a_l \in \Sigma$ and $l \in \mathbb{N}, l \leq k$. We then unsurprisingly have that for each alphabet Σ, $\mathcal{PT}_k(\Sigma^*)$ is the set of languages on Σ equal to a union of $\sim_k^\Sigma$-classes. If we then define the varieties of finite monoids

$$\mathbf{J_k} = \left\langle \bigcup_{\Sigma \text{ alphabet}} \{(\Sigma^*/\sim_k^\Sigma, \cdot/\sim_k^\Sigma)\} \right\rangle_{\mathbf{M}},$$

that is, the variety of finite monoids generated by the quotients of $(\Sigma^*, \cdot)$ by $\sim_k^\Sigma$ for any alphabet Σ, we have that a language is k-piecewise testable if and only if its syntactic monoid belongs to $\mathbf{J_k}$, i.e. $\mathcal{PT}_k = \mathcal{L}(\mathbf{J_k})$. (See [Klíma and Polák, 2010, Section 3].)

Finally, we define the shuffle of two languages.

Definition 5.1.1. Let Σ be an alphabet.

Let $L_1, L_2 \subseteq \Sigma^*$ be two languages over Σ. The *shuffle of L_1 and L_2*, denoted by $L_1 \sqcup\!\sqcup L_2$, is the language $L_1 \sqcup\!\sqcup L_2 = \{w \in \Sigma^* \mid w = u_1 v_1 \cdots u_k v_k$ where $k \in \mathbb{N}_{>0}, u_1, v_1, \ldots, u_k, v_k \in \Sigma^*$ s.t. $u_1 \cdots u_k \in L_1, v_1 \cdots v_k \in L_2\}$ over Σ.

In the following, we consider that $\sqcup\!\sqcup$ has precedence over $\cup$ and $\cap$ (but of course not over concatenation). An example of the shuffle of two languages is given by the language

$(a+b+c)^*bc^*b(a+b+c)^*$ over the alphabet $\{a,b,c\}$, which is equal to the shuffle of the languages $(a+b)^*bb(a+b)^*$ and c^* over $\{a,b,c\}$.

For a given alphabet Σ and a word $u \in \Sigma^*$, we shall use this notion to simply write $u \sqcup\!\sqcup \Sigma^*$ for the language of all words over Σ having u as a subword.

Let us finish with some positive and negative examples of piecewise testable languages. For instance, the language a^* of words over $\{a,b\}$ not containing the letter b is (1-)piecewise testable, since $a^* = (b \sqcup\!\sqcup \{a,b\}^*)^\complement$. Similarly, the language of words over $\{a,b,c\}$ containing $abba$ or $abacab$ as a subword but containing less than three times the letter c is (6-)piecewise testable, since it is equal to $(abba \sqcup\!\sqcup \{a,b,c\}^* \cup abacab \sqcup\!\sqcup \{a,b,c\}^*) \cap (ccc \sqcup\!\sqcup \{a,b,c\}^*)^\complement$.

However, both the language $a(a+b)^*$ of words over $\{a,b\}$ starting with an a and the language $(a+b)^*bb(a+b)^*$ of words over $\{a,b\}$ containing bb as a factor are provably not piecewise testable, because their respective syntactic monoids aren't $\mathfrak{J}$-trivial.

5.2 A fine hierarchy in $\mathcal{P}(\mathbf{J})$

As Tesson and Thérien proved in Tesson and Thérien [2001], any monoid in **DA** has the polynomial-length property, so since $\mathbf{J} \subset \mathbf{DA}$, we have that any monoid in **J** also has the polynomial-length property. Proposition 4.3.3 in the previous chapter shows that there does not exist any $k \in \mathbb{N}$ such that $\mathcal{P}(\mathbf{DA})$ collapses to $\mathcal{P}\big(\mathbf{DA}, n^k\big)$, but this does not rule out the possibility that such a k would exist for $\mathcal{P}(\mathbf{J})$; moreover, we know by that same proposition and Proposition 4.3.4 that for all $k \in \mathbb{N}$, $\mathcal{P}(\mathbf{DA_k})$ collapses to $\mathcal{P}\big(\mathbf{DA_k}, n^{\max\{k,1\}}\big)$ but not further, not telling us much about $\mathcal{P}(\mathbf{J_k})$. In this section, very similar to Section 4.3 of the previous chapter, we show on the one hand that, as for **DA**, while $\mathcal{P}(\mathbf{J}, s(n))$ collapses to $\mathcal{P}(\mathbf{J})$ for any super-polynomial function $s\colon \mathbb{N} \to \mathbb{N}$, there does not exist any $k \in \mathbb{N}$ such that $\mathcal{P}(\mathbf{J})$ collapses to $\mathcal{P}\big(\mathbf{J}, n^k\big)$; and on the other hand that, for each $k \in \mathbb{N}$, $\mathcal{P}(\mathbf{J_k})$ does optimally collapse to $\mathcal{P}\big(\mathbf{J_k}, n^{\lceil k/2 \rceil}\big)$.

5.2.1 Strict hierarchy

In a way similar to Subsection 4.3.1 in Chapter 4, for each $k \in \mathbb{N}$, we exhibit a language $L_k \subseteq \{0,1\}^*$ that can be recognised by a sequence of programs of length $\mathrm{O}(n^{k+1})$ over a monoid $(M_k, *_k)$ in $\mathbf{J_{2k+1}}$ but cannot be recognised by any sequence of programs of length $\mathrm{O}(n^k)$ over any monoid in **J**.

Given $k \in \mathbb{N}$, we say that for some $n \in \mathbb{N}$, σ is a *k-selector over n* if σ is a function of $\mathfrak{P}([n])^{[n]^k}$ that associates a subset of $[n]$ to each vector in $[n]^k$. For any sequence

$\Delta = (\sigma_n)_{n\in\mathbb{N}}$ such that for each $n \in \mathbb{N}$, σ_n is a k-selector over n — a sequence we will call a *sequence of k-selectors* — we set $L_\Delta = \bigcup_{n\in\mathbb{N}} K_{n,\sigma_n}$, where for each $n \in \mathbb{N}$, K_{n,σ_n} is the set of words over $\{0,1\}$ of length $(k+1)\cdot n$ that can be decomposed into $k+1$ consecutive blocks $u^{(1)}, u^{(2)}, \dots, u^{(k)}, v$ of n letters where the first k blocks each contain 1 exactly once and uniquely define a vector ρ in $[n]^k$, where for all $i \in [k]$, ρ_i is given by the position of the only 1 in $u^{(i)}$ (i.e. $u^{(i)}_{\rho_i} = 1$) and v is such that there exists $j \in \sigma_n(\rho)$ verifying that v_j is 1. Observe that for any k-selector σ_0 over 0, we have $L_{0,\sigma_0} = \emptyset$.

We now proceed similarly to what we did in Subsection 4.3.1 to show, on one hand, that for all $k \in \mathbb{N}$, there is a monoid $(M_k, *_k)$ in $\mathbf{J_{2k+1}}$ such that for any sequence of k-selectors Δ, the language L_Δ is recognised by a sequence of programs over $(M_k, *_k)$ of length $\mathrm{O}(n^{k+1})$; and, on the other hand, that for all $k \in \mathbb{N}$ there is a sequence of k-selectors Δ such that for any finite monoid $(M, *)$ and any sequence of programs $(P_n)_{n\in\mathbb{N}}$ over $(M, *)$ of length $\mathrm{O}(n^k)$, L_Δ is not recognised by $(P_n)_{n\in\mathbb{N}}$.

Upper bound. We start with the upper bound. Given $k \in \mathbb{N}$, we define the alphabet $Y_k = \{e, \#\} \cup \{\bot_l, \top_l \mid l \in [k]\}$; we are going to prove that for all $k \in \mathbb{N}$ there exists a language $Z_k \in \mathcal{PT}_{2k+1}(Y_k^*)$ such that for all $\Delta = (\sigma_n)_{n\in\mathbb{N}}$ sequences of k-selectors, there exists a program-reduction from L_Δ to Z_k of length $s(n)$, where $s\colon \mathbb{N} \to \mathbb{N}$ is some function verifying $s(n) \leq 2\cdot(k+1)^{-k}\cdot n^{k+1}$ for all $n \in \mathbb{N}$, using the following proposition and the fact that the language of words of length $n \in \mathbb{N}$ of L_Δ is exactly $K_{n',\sigma_{n'}}$ when there exists $n' \in \mathbb{N}$ verifying $n = (k+1)\cdot n'$ (which implies a program length of at most $2\cdot(k+1)^{-k}\cdot n^{k+1} = 2\cdot(k+1)\cdot n'^{k+1}$) and $\emptyset$ otherwise.

Proposition 5.2.1. *For all $k \in \mathbb{N}$ there is a language $Z_k \in \mathcal{PT}_{2k+1}(Y_k^*)$ such that $\varepsilon \notin Z_k$ and for all $n \in \mathbb{N}$ and all k-selectors σ_n over n, we have $K_{n,\sigma_n} = \Psi^{-1}_{(k+1)\cdot n,\sigma_n}(Z_k^{=|\Psi_{(k+1)\cdot n,\sigma_n}|})$ where $\Psi_{(k+1)\cdot n,\sigma_n}$ is a Y_k-program on $\{0,1\}^{(k+1)\cdot n}$ of length at most $2\cdot(k+1)\cdot n^{k+1}$.*

Proof. We first define by induction on k a family of languages Z_k over the alphabet Y_k. For $k = 0$, Z_0 is $Y_0^*\#Y_0^*$. For $k \in \mathbb{N}_{>0}$, Z_k is the set of words containing each of $\top_k$ and $\bot_k$ exactly once, the first before the latter, and verifying that the sequence between the occurrence of $\top_k$ and the occurrence of $\bot_k$ belongs to Z_{k-1}, i.e. $Z_k = Y_{k-1}^*\top_k Z_{k-1}\bot_k Y_{k-1}^*$. A simple induction on k shows that Z_k for $k \in \mathbb{N}$ is defined by the expression

$$Y_{k-1}^*\top_k Y_{k-2}^*\top_{k-1}\cdots Y_1^*\top_2 Y_0^*\top_1 Y_0^*\#Y_0^*\bot_1 Y_0^*\bot_2 Y_1^*\cdots\bot_{k-1}Y_{k-2}^*\bot_k Y_{k-1}^*\ ,$$

hence in particular it does not contain the empty word ε, and belongs to $\mathcal{PT}_{2k+1}(Y_k^*)$.

Fix $n \in \mathbb{N}$. If $n = 0$, the proposition follows trivially since for any k-selector σ_0 over

0, we have $L_{0,\sigma_0} = \emptyset$ and $\varepsilon \notin Z_k$; otherwise, we define by induction on k a Y_k-program $P_k(d, \sigma)$ on $\{0,1\}^{(d+k+1)\cdot n}$ for every k-selector σ over n and every $d \in \mathbb{N}$.

For any $j \in [n]$ and σ a 0-selector over n, which is just a function in $\mathfrak{P}([n])^{\{\varepsilon\}}$, let $h_{j,\sigma} \colon \{0,1\} \to Y_0$ be the function defined by $h_{j,\sigma}(0) = e$ and $h_{j,\sigma}(1) = \#$ if $j \in \sigma(\varepsilon)$, $h_{j,\sigma}(1) = e$ otherwise. For all $k \in \mathbb{N}_{>0}$, we also let f_k and g_k be the functions in ${Y_k}^{\{0,1\}}$ defined by $f_k(0) = g_k(0) = e$, $f_k(1) = \top_k$ and $g_k(1) = \bot_k$. Moreover, for any k-selector σ over n, $\sigma|j$ for $j \in [n]$ denotes the $(k-1)$-selector over n such that for all $\rho' \in [n]^{k-1}$, $i \in \sigma|j(\rho')$ if and only if $i \in \sigma((j, \rho'))$.

For $k \in \mathbb{N}_{>0}$, $d \in \mathbb{N}$ and σ a k-selector over n, the Y_k-program $P_k(d, \sigma)$ on $\{0,1\}^{(d+k+1)\cdot n}$ is the following sequence of instructions:

$$(d \cdot n + 1, f_k) P_{k-1}(d+1, \sigma|1)(d \cdot n + 1, g_k) \cdots (d \cdot n + n, f_k) P_{k-1}(d+1, \sigma|n)(d \cdot n + n, g_k) \ .$$

In other words, for each position $i \in [\![d \cdot n + 1, d \cdot n + n]\!]$ with a 1 in the $(d+1)$-th block of n letters in the input, the program runs, between the symbols $\top_k$ and $\bot_k$, the program obtained by induction for the $(k-1)$-selector over n obtained by restricting σ to all vectors in $[n]^k$ whose first coordinate is i.

For $k = 0$, $d \in \mathbb{N}$ and σ a 0-selector over n, the Y_0-program $P_0(d, \sigma)$ on $\{0,1\}^{(d+1)\cdot n}$ is the following sequence of instructions:

$$(d \cdot n + 1, h_{1,\sigma})(d \cdot n + 2, h_{2,\sigma}) \cdots (d \cdot n + n, h_{n,\sigma}) \ .$$

In other words, for each position $i \in [\![d \cdot n + 1, d \cdot n + n]\!]$ with a 1 in the $(d+1)$-th block of n letters in the input, the program outputs $\#$ if and only if i does belong to the set $\sigma(\varepsilon)$.

A simple computation shows that for any $k \in \mathbb{N}$, $d \in \mathbb{N}$ and σ a k-selector over n, the number of instructions in $P_k(d, \sigma)$ is at most $2 \cdot (k+1) \cdot n^{k+1}$.

A simple induction on k shows that for any $k \in \mathbb{N}$ and $d \in \mathbb{N}$, when running on a word $w \in \{0,1\}^{(d+k+1)\cdot n}$, for any σ a k-selector over n, $P_k(d, \sigma)$ returns a word in Z_k if and only if when $u^{(1)}, u^{(2)}, \ldots, u^{(k)}, v$ are the last $k+1$ consecutive blocks of n letters of w, then $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ each contain 1 exactly once and define the vector ρ in $[n]^k$ where for all $i \in [k]$, ρ_i is given by the position of the only 1 in $u^{(i)}$, verifying that there exists $j \in \sigma_n(\rho)$ such that v_j is 1.

Therefore, for any $k \in \mathbb{N}$ and σ_n a k-selector over n, if we set $\Psi_{(k+1)\cdot n, \sigma_n} = P_k(0, \sigma_n)$, we have $K_{n,\sigma_n} = \Psi^{-1}_{(k+1)\cdot n, \sigma_n}({Z_k}^{=|\Psi_{(k+1)\cdot n, \sigma_n}|})$ where $\Psi_{(k+1)\cdot n, \sigma_n}$ is a Y_k-program on $\{0,1\}^{(k+1)\cdot n}$ of length at most $2 \cdot (k+1) \cdot n^{k+1}$. □

Consequently, for all $k \in \mathbb{N}$ and any sequence of k-selectors Δ, since the language Z_k is

in $\mathcal{PT}_{2k+1}(Y_k^*)$ and thus recognised by a monoid from $\mathbf{J_{2k+1}}$, we have, by Corollary 3.4.3, that $L_\Delta \in \mathcal{P}\big(\mathbf{J_{2k+1}}, n^{k+1}\big)$, as there is a program-reduction from it to Z_k of length $s(n)$, where $s\colon \mathbb{N} \to \mathbb{N}$ is some function verifying $s(n) \leq 2 \cdot (k+1)^{-k} \cdot n^{k+1}$ for all $n \in \mathbb{N}$.

Lower bound. If for some $k \in \mathbb{N}$ and $i \in [\alpha]$, $\alpha \in \mathbb{N}_{>0}$, we apply Claim 4.3.2 for all $n \in \mathbb{N}$, $l = \alpha \cdot ((k+1) \cdot n)^k$, we get a number $\mu_i(n)$ of languages in $\{0,1\}^{(k+1)\cdot n}$ recognised by programs over a monoid of order i on $\{0,1\}^{(k+1)\cdot n}$ with at most l instructions that is in $2^{\mathrm{O}(n^k \log_2(n))}$, which is asymptotically strictly smaller than the number of distinct K_{n,σ_n} when the k-selector σ_n over n varies, which is $2^{n^{k+1}}$, i.e. $\mu_i(n)$ is in $\mathrm{o}(2^{n^{k+1}})$.

Hence, for all $j \in \mathbb{N}_{>0}$, there exist an $n_j \in \mathbb{N}$ and τ_j a k-selector over n_j such that no program over a monoid of order $i \in [j]$ on $\{0,1\}^{(k+1)\cdot n_j}$ and of length at most $j \cdot ((k+1)\cdot n_j)^k$ recognises K_{n_j,τ_j}. Moreover, we can assume without loss of generality that the sequence $(n_j)_{j\in\mathbb{N}_{>0}}$ is increasing. Let $\Delta = (\sigma_n)_{n\in\mathbb{N}}$ be such that $\sigma_{n_j} = \tau_j$ for all $j \in \mathbb{N}_{>0}$ and $\sigma_n\colon [n]^k \to \mathfrak{P}([n]), \rho \mapsto \emptyset$ for any $n \in \mathbb{N}$ verifying that it is not equal to any n_j for $j \in \mathbb{N}_{>0}$. We show that no sequence of programs over a finite monoid of length $\mathrm{O}(n^k)$ can recognise L_Δ. If this were the case, then let i be the order of the monoid. Let $j \in \mathbb{N}, j \geq i$ be such that for any $n \in \mathbb{N}$, the n-th program has length at most $j \cdot n^k$. But, by construction, we know that there does not exist any such program on $\{0,1\}^{(k+1)\cdot n_j}$ recognising K_{n_j,τ_j}, a contradiction.

This implies the following hierarchy, using the fact that for all $k \in \mathbb{N}$ and all $d \in \mathbb{N}, d \leq \lceil \frac{k}{2} \rceil - 1$, any monoid from $\mathbf{J_d}$ is also a monoid from $\mathbf{J_k}$.

Proposition 5.2.2. *For all $k \in \mathbb{N}$, $\mathcal{P}\big(\mathbf{J}, n^k\big) \subset \mathcal{P}\big(\mathbf{J}, n^{k+1}\big)$. More precisely, for all $k \in \mathbb{N}$ and $d \in \mathbb{N}, d \leq \lceil \frac{k}{2} \rceil - 1$, $\mathcal{P}\big(\mathbf{J_k}, n^d\big) \subset \mathcal{P}\big(\mathbf{J_k}, n^{d+1}\big)$.*

5.2.2 Collapse

Looking at Proposition 5.2.2, it looks at first glance rather strange that, when restricting the monoids to be taken from $\mathbf{J_k}$ for some $k \in \mathbb{N}$, we can only prove strictness of the hierarchy inside $\mathcal{P}(\mathbf{J_k})$ up to exponent $\lceil \frac{k}{2} \rceil$. We now show, in a way similar to Subsection 4.3.2 in Chapter 4, that in fact $\mathcal{P}(\mathbf{J_k})$ does collapse to $\mathcal{P}\big(\mathbf{J_k}, n^{\lceil k/2 \rceil}\big)$ for all $k \in \mathbb{N}$, showing Proposition 5.2.2 to be optimal in some sense.

Proposition 5.2.3. *Let $k \in \mathbb{N}$. Let $(M, *) \in \mathbf{J_k}$ and Σ be an alphabet. Then there exists a constant $c \in \mathbb{N}_{>0}$ such that any program over $(M,*)$ on Σ^n for $n \in \mathbb{N}$ is equivalent to a program over $(M,*)$ on Σ^n of length at most $c \cdot n^{\lceil k/2 \rceil}$.*

In particular, $\mathcal{P}(\mathbf{J_k}) = \mathcal{P}\big(\mathbf{J_k}, n^{\lceil k/2 \rceil}\big)$ for all $k \in \mathbb{N}$.

As we observed in Subsection 4.3.2 of the previous chapter, given P a program over some finite monoid $(M, *)$ on Σ^n for $n \in \mathbb{N}$ and Σ an alphabet, a subprogram P' of P is equivalent to P if and only if for every language $K \subseteq M^*$ recognised by the evaluation morphism $\eta_{(M,*)}$ of $(M, *)$ we have $\xi_P(w) \in K \Leftrightarrow \xi_{P'}(w) \in K$ for all $w \in \Sigma^n$. Moreover, every language recognised by $\eta_{(M,*)}$ is precisely a language of $\mathcal{PT}_k(M^*)$ when $(M, *) \in \mathbf{J_k}$ for some $k \in \mathbb{N}$.

The result is hence a consequence of the following lemma and the fact that every language in $\mathcal{PT}_k(M^*)$ is a union of $\sim_k^M$-classes, each of those classes corresponding to all words over M having the same set of k-subwords, that is finite.

Lemma 5.2.4. *Let Σ be an alphabet and $(M, *)$ a finite monoid.*

For all $k \in \mathbb{N}$, there exists a constant $c \in \mathbb{N}_{>0}$ verifying that for any program P over $(M, *)$ on Σ^n for $n \in \mathbb{N}$ and any word $t \in M^k$, there exists a subprogram Q of P of length at most $c \cdot n^{\lceil k/2 \rceil}$ such that for any subprogram Q' of P that has Q as a subprogram, for all $w \in \Sigma^n$, t is a subword of $\xi_P(w)$ if and only if t is a subword of $\xi_{Q'}(w)$.

Proof. A program P over $(M, *)$ on Σ^n for $n \in \mathbb{N}$ is a finite sequence (p_i, f_i) of instructions where each p_i is a positive natural number which is at most n and each f_i is a function from Σ to M. We denote by l the number of instructions of P. As in the proof of Lemma 4.3.5, for each set $I \subseteq [l]$ we denote by $P[I]$ the subprogram of P consisting of the subsequence of instructions of P obtained after removing all instructions whose index is not in I. In particular, $P[1, m]$ denotes the initial sequence of instructions of P, until instruction number m.

We prove the lemma by induction on k, fixing the constant to be $c_k = k! \cdot |\Sigma|^{\lceil k/2 \rceil}$ for a given $k \in \mathbb{N}$.

The intuition behind the proof for a program P on inputs of length n and some t of length at least 3 is as follows. Consider all the indices $1 \leq i_1 < i_2 < \cdots < i_s \leq l$ that correspond, for a fixed letter a and a fixed position p in the input, to the first instruction of P that would output the element t_1 when reading a at position p or to the last instruction of P that would output the element t_k when reading a at position p. We then have that, given some w as input, t is a subword of $\xi_P(w)$ if and only if there exist $1 \leq \gamma < \delta \leq s$ verifying that the element at position i_γ of $\xi_P(w)$ is t_1, the element at position i_δ of $\xi_P(w)$ is t_k and $t_2 \cdots t_{k-1}$ is a subword of $\xi_{P[i_\gamma+1, i_\delta-1]}(w)$. The idea is then that if we set I to contain $i_1, i_2, \ldots, i_s$ as well as all indices obtained by induction for $P[i_\gamma + 1, i_\delta - 1]$ and $t_2 \cdots t_{k-1}$ for all $1 \leq \gamma < \delta \leq s$, we would have that for all w, t is a subword of $\xi_P(w)$ if and only if it is a subword of $\xi_{P[I]}(w)$, that is $\xi_P(w)$ where only the elements at indices in I have been kept. The only problem is that in doing this, I

would possibly be too big (possibly at least $\alpha \cdot n^{\lceil (k-2)/2 \rceil + 2}$ for α some constant), because the number of pairs $1 \leq \gamma < \delta \leq s$ is quadratic in s and hence possibly in n. We solve this problem by setting I to contain, in addition to $i_1, i_2, \ldots, i_s$, all indices obtained by induction for $P[i_j + 1, i_{j+1} - 1]$ and $t_\alpha \cdots t_\beta$ for all $1 \leq j \leq s-1$ (for which we have a number of possibilities linear in s and hence at most linear in n) and $1 < \alpha \leq \beta < k$ (for which we have a number of possibilities quadratic in k, a constant).

The intuition behind the proof when t is of length less than 3 is essentially the same, but without induction.

Inductive step. Let $k \in \mathbb{N}, k \geq 3$ and assume the lemma proven for all $k' \in \mathbb{N}, k' < k$. Let P be a program over $(M, *)$ on Σ^n for $n \in \mathbb{N}$ of length $l \in \mathbb{N}$ and some word $t \in M^k$.

Observe that when $n = 0$, we necessarily have $P = \varepsilon$, so that the lemma is trivially proven in that case. So we now assume $n > 0$.

For each $p \in [n]$ and each $a \in \Sigma$ consider within the sequence of instructions of P the first instruction of the form (p, f) with $f(a) = t_1$ and the last instruction of that form with $f(a) = t_k$, if they exist. We let $I_{(1,k)}$ be the set of indices of these instructions for all a and p. Notice that the size of $I_{(1,k)}$ is at most $2 \cdot |\Sigma| \cdot n$.

Let $s = \left|I_{(1,k)}\right|$ and let us denote $I_{(1,k)} = \{i_1, i_2, \ldots, i_s\}$ where $i_1 < i_2 < \cdots < i_s$. Given $\alpha, \beta \in [k]$, we also set $t^{(\alpha,\beta)} = t_\alpha t_{\alpha+1} \cdots t_\beta$. For all $\alpha, \beta \in [k]$ such that $1 < \alpha \leq \beta < k$ and $j \in [s-1]$, we let $J_{j,(\alpha,\beta)}$ be the set of indices of the instructions within $P[i_j + 1, i_{j+1} - 1]$ appearing in its subprogram obtained by induction for $P[i_j + 1, i_{j+1} - 1]$ and $t^{(\alpha,\beta)}$.

We now let I be the union of $I_{(1,k)}$ and $J'_{j,(\alpha,\beta)} = \{e + i_j \mid e \in J_{j,(\alpha,\beta)}\}$ for all $\alpha, \beta \in [k]$ such that $1 < \alpha \leq \beta < k$ and $j \in [s-1]$ (the translation being required because the first instruction in $P[i_j + 1, i_{j+1} - 1]$ is the $(i_j + 1)$-th instruction in P). We claim that $Q = P[I]$, a subprogram of P, has the desired properties.

First notice that by induction the size of $J'_{j,(\alpha,\beta)}$ for all $\alpha, \beta \in [k]$ such that $1 < \alpha \leq \beta < k$ and $j \in [s-1]$ is upper bounded by

$$(\beta - \alpha + 1)! \cdot |\Sigma|^{\lceil (\beta-\alpha+1)/2 \rceil} \cdot n^{\lceil (\beta-\alpha+1)/2 \rceil} \leq (k-2)! \cdot |\Sigma|^{\lceil (k-2)/2 \rceil} \cdot n^{\lceil (k-2)/2 \rceil} .$$

Hence, the size of I is at most

$$\begin{aligned} &\left|I_{(1,k)}\right| + \sum_{j=1}^{s-1} \sum_{1<\alpha\leq\beta<k} \left|J'_{j,(\alpha,\beta)}\right| \\ \leq &2 \cdot |\Sigma| \cdot n + (2 \cdot |\Sigma| \cdot n - 1) \cdot \frac{(k-1) \cdot (k-2)}{2} \cdot (k-2)! \cdot |\Sigma|^{\lceil (k-2)/2 \rceil} \cdot n^{\lceil (k-2)/2 \rceil} \end{aligned}$$

$$\leq 2 \cdot |\Sigma| \cdot n + (2 \cdot |\Sigma| \cdot n - 1) \cdot \frac{k \cdot (k-1)}{2} \cdot (k-2)! \cdot |\Sigma|^{\lceil (k-2)/2 \rceil} \cdot n^{\lceil (k-2)/2 \rceil}$$
$$\leq k! \cdot |\Sigma|^{\lceil k/2 \rceil} \cdot n^{\lceil k/2 \rceil} = c_k \cdot n^{\lceil k/2 \rceil}$$

as $\left|\{(\alpha, \beta) \in \mathbb{N}^2 \mid 1 < \alpha \leq \beta < k\}\right| = \sum_{j=2}^{k-1}(k-j) = \sum_{j=1}^{k-2} j = \frac{(k-1)\cdot(k-2)}{2}$ and $2 \cdot |\Sigma| \cdot n \leq \frac{k!}{2} \cdot |\Sigma|^{\lceil (k-2)/2 \rceil} \cdot n^{\lceil (k-2)/2 \rceil}$ since $k \geq 3$, so that $P[I]$ has at most the required length.

Let Q' be a subprogram of P that has Q as a subprogram: it means that there exists some set $I' \subseteq [l]$ containing I such that $Q' = P[I']$.

Take $w \in \Sigma^n$.

Assume now that t is a subword of $\xi_P(w)$. It means that there exist $r_1, r_2, \ldots, r_k \in [l]$, $r_1 < r_2 < \cdots < r_k$, such that for all $j \in [k]$, $f_{r_j}(w_{p_{r_j}}) = t_j$. By definition of $I_{(1,k)}$, there exist $\gamma, \delta \in [s], \gamma < \delta$, such that $i_\gamma \leq r_1 < r_k \leq i_\delta$ and $f_{i_\gamma}(w_{p_{i_\gamma}}) = t_1$ and $f_{i_\delta}(w_{p_{i_\delta}}) = t_k$. For each $j \in [\![\gamma, \delta - 1]\!]$, let $m_j \in [\![2, k]\!]$ be the smallest integer in $[\![2, k-1]\!]$ such that $i_j \leq r_{m_j} < i_{j+1}$, k if it does not exist, and $M_j \in [\![1, k-1]\!]$ be the biggest integer in $[\![2, k-1]\!]$ such that $i_j \leq r_{M_j} < i_{j+1}$, 1 if it does not exist. Observe that, since for each $j \in [\![\gamma, \delta - 1]\!]$, $t^{(m_j, M_j)} = t^{(k,1)} = \varepsilon$ if there does not exist any $o \in [\![2, k-1]\!]$ verifying $i_j \leq r_o < i_{j+1}$, we have $t^{(2,k-1)} = \prod_{j=\gamma}^{\delta-1} t^{(m_j, M_j)}$. For all $j \in [\![\gamma, \delta - 1]\!]$, we have that for any set $J \subseteq [i_{j+1} - i_j - 1]$ containing $\bigcup_{1<\alpha\leq\beta<k} J_{j,(\alpha,\beta)}$, $t^{(m_j, M_j)}$ is a subword of $f_{i_j}(w_{p_{i_j}})\xi_{P[i_j+1, i_{j+1}-1][J]}(w)$ when $m_j < k$ and $r_{m_j} = i_j$, and of $\xi_{P[i_j+1, i_{j+1}-1][J]}(w)$ otherwise. Indeed, let $j \in [\![\gamma, \delta - 1]\!]$.

- If $m_j < k$ and $r_{m_j} = i_j$, then $f_{i_j}(w_{p_{i_j}}) = f_{r_{m_j}}(w_{p_{r_{m_j}}}) = t_{m_j}$ and $i_j = r_{m_j} < r_{m_j+1} < \cdots < r_{M_j} < i_{j+1}$, so $t^{(m_j+1, M_j)}$ is a subword of $\xi_{P[i_j+1, i_{j+1}-1]}(w)$. This implies, directly when $m_j = M_j$ or by induction otherwise, that for any set $J \subseteq [i_{j+1} - i_j - 1]$ containing $\bigcup_{1<\alpha\leq\beta<k} J_{j,(\alpha,\beta)}$, $t^{(m_j+1, M_j)}$ is a subword of $\xi_{P[i_j+1, i_{j+1}-1][J]}(w)$. This implies in turn that $t^{(m_j, M_j)}$ is a subword of $f_{i_j}(w_{p_{i_j}})\xi_{P[i_j+1, i_{j+1}-1][J]}(w)$.

- Otherwise, when $m_j = k$, there does not exist any $o \in [\![2, k-1]\!]$ verifying $i_j \leq r_o < i_{j+1}$, so $t^{(m_j, M_j)} = \varepsilon$ is trivially a subword of $\xi_{P[i_j+1, i_{j+1}-1][J]}(w)$ for any set $J \subseteq [i_{j+1} - i_j - 1]$ containing $\bigcup_{1<\alpha\leq\beta<k} J_{j,(\alpha,\beta)}$. And when $m_j < k$ but $r_{m_j} \neq i_j$, it means that $r_{m_j} > i_j$, hence $i_j < r_{m_j} < r_{m_j+1} < \cdots < r_{M_j} < i_{j+1}$, so $t^{(m_j, M_j)}$ is a subword of $\xi_{P[i_j+1, i_{j+1}-1]}(w)$. This implies, by induction, that $t^{(m_j, M_j)}$ is a subword of $\xi_{P[i_j+1, i_{j+1}-1][J]}(w)$ for any set $J \subseteq [i_{j+1} - i_j - 1]$ containing $\bigcup_{1<\alpha\leq\beta<k} J_{j,(\alpha,\beta)}$.

Therefore, using the convention that $i_0 = 0$ and $i_{s+1} = l + 1$, if we set, for each $j \in [\![0, s]\!]$, $I'_j = \{e - i_j \mid e \in I', i_j < e < i_{j+1}\}$ as the subset of I' of elements strictly between i_j and

i_{j+1} translated by $-i_j$, we have that $t^{(2,k-1)}$ is a subword of

$$\xi_{P[i_\gamma+1,i_{\gamma+1}-1][I'_\gamma]}(w) f_{i_{\gamma+1}}(w_{p_{i_{\gamma+1}}}) \xi_{P[i_{\gamma+1}+1,i_{\gamma+2}-1][I'_{\gamma+1}]}(w) \cdots \\ f_{i_{\delta-1}}(w_{p_{i_{\delta-1}}}) \xi_{P[i_{\delta-1}+1,i_\delta-1][I'_{\delta-1}]}(w)$$

(since we have $r_{m_\gamma} \geq r_2 > r_1 \geq i_\gamma$), so that, as $f_{i_\gamma}(w_{p_{i_\gamma}}) = t_1$ and $f_{i_\delta}(w_{p_{i_\delta}}) = t_k$, we have that $t = t_1 t^{(2,k-1)} t_k$ is a subword of

$$\xi_{P[1,i_1-1][I'_0]}(w) f_{i_1}(w_{p_{i_1}}) \xi_{P[i_1+1,i_2-1][I'_1]}(w) \cdots f_{i_s}(w_{p_{i_s}}) \xi_{P[i_s+1,l][I'_s]}(w) = \xi_{P[I']}(w) \ .$$

Assume finally that t is a subword of $\xi_{P[I']}(w)$. Then it is obviously a subword of $\xi_P(w)$, as $\xi_{P[I']}(w)$ is a subword of $\xi_P(w)$.

Therefore, t is a subword of $\xi_P(w)$ if and only if t is a subword of $\xi_{Q'}(w) = \xi_{P[I']}(w)$, as desired.

Base case. There are three subcases to consider.

Subcase $k = 2$. Let P be a program over $(M, *)$ on Σ^n for $n \in \mathbb{N}$ of length $l \in \mathbb{N}$ and some word $t \in M^2$.

We use the same idea as in the inductive step.

Observe that when $n = 0$, we necessarily have $P = \varepsilon$, so that the lemma is trivially proven in that case. So we now assume $n > 0$.

For each $p \in [n]$ and each $a \in \Sigma$ consider within the sequence of instructions of P the first instruction of the form (p, f) with $f(a) = t_1$ and the last instruction of that form with $f(a) = t_2$, if they exist. We let I be the set of indices of these instructions for all a and p. Notice that the size of I is at most $2 \cdot |\Sigma| \cdot n = 2! \cdot |\Sigma|^{\lceil 2/2 \rceil} \cdot n^{\lceil 2/2 \rceil} = c_2 \cdot n^{\lceil 2/2 \rceil}$.

We claim that $Q = P[I]$, a subprogram of P, has the desired properties. We just showed it has at most the required length.

Let Q' be a subprogram of P that has Q as a subprogram: it means that there exists some set $I' \subseteq [l]$ containing I such that $Q' = P[I']$.

Take $w \in \Sigma^n$.

Assume now that t is a subword of $\xi_P(w)$. It means there exist $i_1, i_2 \in [l], i_1 < i_2$ such that $f_{i_1}(w_{p_{i_1}}) = t_1$ and $f_{i_2}(w_{p_{i_2}}) = t_2$. By definition of I, there exist ${i_1}', {i_2}' \in I$, such that ${i_1}' \leq i_1 < i_2 \leq {i_2}'$ and $f_{{i_1}'}(w_{p_{{i_1}'}}) = t_1$ and $f_{{i_2}'}(w_{p_{{i_2}'}}) = t_2$. Hence, as $f_{{i_1}'}(w_{p_{{i_1}'}}) f_{{i_2}'}(w_{p_{{i_2}'}})$ is a subword of $\xi_{P[I']}(w)$ (because $I \subseteq I'$), we get that $t = t_1 t_2$ is a subword of $\xi_{P[I']}(w)$.

Assume finally that t is a subword of $\xi_{P[I']}(w)$. Then it is obviously a subword of $\xi_P(w)$, as $\xi_{P[I']}(w)$ is a subword of $\xi_P(w)$.

Therefore, t is a subword of $\xi_P(w)$ if and only if t is a subword of $\xi_{Q'}(w) = \xi_{P[I']}(w)$, as desired.

Subcase $k = 1$**.** Let P be a program over $(M, *)$ on Σ^n for $n \in \mathbb{N}$ of length $l \in \mathbb{N}$ and some word $t \in M^1$.

We again use the same idea as before.

Observe that when $n = 0$, we necessarily have $P = \varepsilon$, so that the lemma is trivially proven in that case. So we now assume $n > 0$.

For each $p \in [n]$ and each $a \in \Sigma$ consider within the sequence of instructions of P the first instruction of the form (p, f) with $f(a) = t_1$, if it exists. We let I be the set of indices of these instructions for all a and p. Notice that the size of I is at most $|\Sigma| \cdot n = 1! \cdot |\Sigma|^{\lceil 1/2 \rceil} \cdot n^{\lceil 1/2 \rceil} = c_1 \cdot n^{\lceil 1/2 \rceil}$.

We claim that $Q = P[I]$, a subprogram of P, has the desired properties. We just showed it has at most the required length.

Let Q' be a subprogram of P that has Q as a subprogram: it means that there exists some set $I' \subseteq [l]$ containing I such that $Q' = P[I']$.

Take $w \in \Sigma^n$.

Assume now that t is a subword of $\xi_P(w)$. It means there exists $i \in [l]$ such that $f_i(w_{p_i}) = t_1$. By definition of I, there exists $i' \in I$ such that $i' \leq i$ and $f_{i'}(w_{p_{i'}}) = t_1$. Hence, as $f_{i'}(w_{p_{i'}})$ is a subword of $\xi_{P[I']}(w)$ (because $I' \subseteq I$), we get that $t = t_1$ is a subword of $\xi_{P[I']}(w)$.

Assume finally that t is a subword of $\xi_{P[I']}(w)$. Then it is obviously a subword of $\xi_P(w)$, as $\xi_{P[I']}(w)$ is a subword of $\xi_P(w)$.

Therefore, t is a subword of $\xi_P(w)$ if and only if t is a subword of $\xi_{Q'}(w) = \xi_{P[I']}(w)$, as desired.

Subcase $k = 0$**.** Let P be a program over $(M, *)$ on Σ^n for $n \in \mathbb{N}$ of length $l \in \mathbb{N}$ and some word $t \in M^0$.

We claim that $Q = \varepsilon$, a subprogram of P, has the desired properties.

First notice that the length of Q is $0 \leq 0! \cdot |\Sigma|^{\lceil 0/2 \rceil} \cdot n^{\lceil 0/2 \rceil} = c_0 \cdot n^{\lceil 0/2 \rceil}$, at most the required length.

Let Q' be a subprogram of P that has Q as a subprogram. As $t \in M^0$, we necessarily have that $t = \varepsilon$, which is a subword of any word in M^*. Therefore, we immediately get

that for all $w \in \Sigma^n$, t is a subword of $\xi_P(w)$ if and only if t is a subword of $\xi_{Q'}(w)$, as desired. □

5.3 Regular languages in $\mathcal{P}(\mathbf{J})$

As we showed in Chapter 3, $\mathbf{J}$ is not an *sp*-variety of finite monoids and it even contains finite monoids that p-recognise languages outside $\mathcal{L}(\mathbf{QJ})$ (see Lemma 3.4.13).

In this section, we present some results that give a better understanding of what regular languages can be p-recognised by monoids in $\mathbf{J}$.

5.3.1 Non-tameness of J

Lemma 3.4.12 shows that the language $a(a+b)^*$ over $\{a,b\}$ does belong to $\mathcal{P}(\mathbf{J})$, while it does not belong to $\mathcal{L}(\mathbf{QJ})$, showing $\mathbf{J}$ isn't tame. But actually, the first thing to notice is that, though none of them is in $\mathcal{L}(\mathbf{QJ})$, all languages of the form $\Sigma^* u$ and $u\Sigma^*$ for Σ an alphabet and $u \in \Sigma^+$ are in $\mathcal{P}(\mathbf{J})$. Indeed, for each of these languages, a sequence of constant-length programs over the $\mathfrak{J}$-trivial monoid $(\mathbb{Z}/2\,\mathbb{Z}, \times)$ (where $\times$ denotes the canonical product modulo 2) recognising it can be built, where for each possible input length, the associated program just checks if the $|u|$ first (or last) letters are correct. So, informally stated, programs over monoids in $\mathbf{J}$ can check for some constant-length beginning or ending of their input words.

But programs over monoids in $\mathbf{J}$ can do even more. Indeed, it is easily seen that the language $(a+b)^*ac^+$ does not belong to $\mathcal{L}(\mathbf{QJ})$ (just observe that the stable monoid of its syntactic morphism is equal to its syntactic monoid, which is not $\mathfrak{J}$-trivial), yet $(a+b)^*ac^+$ is p-recognised by a monoid in $\mathbf{J}$. The crucial insight is that it can be program-reduced in linear length to the piecewise testable language L of all words over $\{a,b,c\}$ having ca as a subword but not the subwords cca, caa and cb by using the following trick for input length $n \in \mathbb{N}$: reading the input letters in the order $2, 1, 3, 2, 4, 3, 5, 4, \ldots, n, n-1$ (a reading technique we shall call "feedback-sweeping"), just outputting the letters read. We are now going to formalise this proof.

Lemma 5.3.1. $(a+b)^*ac^+ \in \mathcal{P}(\mathbf{J}, n)$.

Proof. Let $\Sigma = \{a,b,c\}$.

Let

$$L = ca \sqcup\!\sqcup \Sigma^* \cap (cca \sqcup\!\sqcup \Sigma^*)^{\complement} \cap (caa \sqcup\!\sqcup \Sigma^*)^{\complement} \cap (cb \sqcup\!\sqcup \Sigma^*)^{\complement}$$

be the language of all words over Σ having ca as a subword but not the subwords cca, caa and cb, that by construction is piecewise testable, i.e. belongs to $\mathcal{L}(\mathbf{J})$.

We are now going to build a program-reduction from $(a+b)^*ac^+$ to L. Let $n \in \mathbb{N}$. If $n \leq 1$, we set Ψ_n to be ε, the empty Σ-program on Σ^n. Otherwise, if $n \geq 2$, we set

$$\Psi_n = (2, \mathrm{id}_\Sigma)(1, \mathrm{id}_\Sigma)(3, \mathrm{id}_\Sigma)(2, \mathrm{id}_\Sigma)(4, \mathrm{id}_\Sigma)(3, \mathrm{id}_\Sigma)(5, \mathrm{id}_\Sigma)(4, \mathrm{id}_\Sigma)\cdots(n, \mathrm{id}_\Sigma)(n-1, \mathrm{id}_\Sigma) \ .$$

Let us define $s\colon \mathbb{N} \to \mathbb{N}$ by $s(n) = |\Psi_n|$ for all $n \in \mathbb{N}$, which is such that

$$s(n) = \begin{cases} 0 & \text{if } n \leq 1 \\ 2n-2 & \text{otherwise } (n \geq 2) \end{cases}$$

for all $n \in \mathbb{N}$. Fix $n \in \mathbb{N}$.

Let $w \in ((a+b)^*ac^+)^{=n}$: it means $n \geq 2$ and there exist $u \in (a+b)^{n_1}$ with $n_1 \in [\![0, n-2]\!]$ and $n_2 \in [\![0, n-2]\!]$ verifying that $w = uacc^{n_2}$ and $n_1 + n_2 = n-2$. We therefore have

$$\Psi_n(w) = \begin{cases} cac^{2n_2} & \text{when } n_1 = 0 \\ u_2u_1\cdots u_{n_1}u_{n_1-1}au_{n_1}cac^{2n_2} & \text{otherwise } (n_1 > 0) \ , \end{cases}$$

a word easily seen to belong to $L^{=2n-2}$. Since this is true for all $w \in ((a+b)^*ac^+)^{=n}$, it follows that $((a+b)^*ac^+)^{=n} \subseteq \Psi_n^{-1}(L^{=s(n)})$.

Let conversely $w \in \Psi_n^{-1}(L^{=s(n)})$. Since this means that $\Psi_n(w) \in L^{=s(n)}$, we necessarily have $n \geq 2$ as it must contain ca as a subword, so that

$$\Psi_n(w) = w_2w_1w_3w_2w_4w_3\cdots w_nw_{n-1} \ .$$

Let $i, j \in [n]$ verifying that $w_i = c$, $w_j = a$ and w_iw_j is a subword of $\Psi_n(w)$. This means that $j \geq i-1$, and we will now show that, actually, $j = i-1$. Assume that $j \geq i+2$; by construction, this would mean that $w_iw_jw_j = caa$ is a subword of $\Psi_n(w)$, a contradiction to the fact it belongs to L. Assume otherwise that $j = i+1$; by construction, this would either mean that $w_iw_{i-1}w_{i+1}w_i$ is a subword of $\Psi_n(w)$, which would imply one of caa, cba and cca is a subword of $\Psi_n(w)$, or that $w_{i+1}w_iw_{i+2}w_{i+1}$ is a subword of $\Psi_n(w)$, which would imply one of caa, cba and cca is a subword of $\Psi_n(w)$, in both cases contradicting the fact $\Psi_n(w)$ belongs to L. Hence, we indeed have $j = i-1$, and in particular that $i \geq 2$. Now, by construction, for each $t \in [i-2]$, we have that $w_tw_iw_{i-1} = w_tca$ is a subword of $\Psi_n(w) \in L$, so that w_t cannot be equal to c. Similarly, for each $t \in [\![i+1, n]\!]$, we have that

$w_i w_{i-1} w_t = caw_t$ is a subword of $\Psi_n(w) \in L$, so that w_t must be equal to c. This means that $w_1 \cdots w_{i-1} \in (a+b)^*$ and $w_{i+1} \cdots w_n \in c^*$, so that $w \in (a+b)^* acc^* = (a+b)^* ac^+$. Since this is true for all $w \in \Psi_n^{-1}(L^{=s(n)})$, it follows that $((a+b)^* ac^+)^{=n} \supseteq \Psi_n^{-1}(L^{=s(n)})$.

Therefore, we have that $((a+b)^* ac^+)^{=n} = \Psi_n^{-1}(L^{=s(n)})$ for all $n \in \mathbb{N}$, so $(\Psi_n)_{n\in\mathbb{N}}$ is a program reduction from $(a+b)^* ac^+$ to L of length $s(n)$. So since $L \in \mathcal{L}(\mathbf{J})$, we can conclude that $(a+b)^* ac^+ \in \mathcal{P}(\mathbf{J}, s(n)) = \mathcal{P}(\mathbf{J}, n)$ by Corollary 3.4.3. □

Using variants of this "feedback-sweeping" technique, we can prove that the phenomenon just described is not an isolated case.

Lemma 5.3.2. *We have the following.*

$$\begin{aligned}
(a+b)^* ac^+ &\in \mathcal{P}(\mathbf{J}) \setminus \mathcal{L}(\mathbf{QJ}) \ , \\
(a+b)^* ac^+ a(a+b)^* &\in \mathcal{P}(\mathbf{J}) \setminus \mathcal{L}(\mathbf{QJ}) \ , \\
c^+ a(a+b)^* ac^+ &\in \mathcal{P}(\mathbf{J}) \setminus \mathcal{L}(\mathbf{QJ}) \ , \\
\left((a+c)^2\right)^* (bb)^* &\in \mathcal{P}(\mathbf{J}) \cap \mathcal{L}(\mathbf{QJ}) \ , \\
(a+b)^* bac^+ &\in \mathcal{P}(\mathbf{J}) \setminus \mathcal{L}(\mathbf{QJ}) \ , \\
(a+b)^* ac^+ (a+b)^* ac^+ &\in \mathcal{P}(\mathbf{J}) \setminus \mathcal{L}(\mathbf{QJ}) \ .
\end{aligned}$$

Hence, we are tempted to say that there are "much more" regular languages in $\mathcal{P}(\mathbf{J})$ than just those in $\mathcal{L}(\mathbf{QJ})$, even if it is not clear to us whether all of the languages in $\mathcal{L}(\mathbf{QJ})$ are in $\mathcal{P}(\mathbf{J})$ or not. Consequently, by Proposition 3.4.8, we are tempted to add that it contains "much more" sets of word problems than just those over semigroups $(S, *)$ such that in $(S, *)^1 \in \mathbf{J}$. But can we show any limit on the "algebraic complexity" of a finite semigroup whose set of word problems is contained in $\mathcal{P}(\mathbf{J})$? It turns out that we can, relying on a known result and one proven in this thesis.

Obviously, since $\mathbf{J} \subseteq \mathbf{DA}$, we have $\mathcal{P}(\mathbf{J}) \subseteq \mathcal{P}(\mathbf{DA})$, so the tameness of $\mathbf{DA}$, Theorem 4.2.1, implies what follows, observing that $\langle \mathbf{DA} \rangle_{\mathbf{S}}$ is exactly the class of all finite semigroups $(S, *)$ such that $(S, *)^1 \in \mathbf{DA}$ (see [Eilenberg, 1976, Chapter V, Proposition 1.2]).

Lemma 5.3.3. *Let $(S, *)$ be a finite semigroup such that $(S, *) \notin \langle \mathbf{DA} \rangle_{\mathbf{S}}$. Then, we have $\mathcal{W}((S, *)) \nsubseteq \mathcal{P}(\mathbf{J})$.*

Moreover $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{QDA})$.

Moreover, since $\mathbf{J} \subseteq \mathbf{J} * \mathbf{D}$, we have $\mathcal{P}(\mathbf{J}) \subseteq \mathcal{P}(\mathbf{J} * \mathbf{D})$ (if we extend the program-over-monoid formalism in the obvious way to finite semigroups) and Maciel, Péladeau

and Thérien, in [Maciel et al., 2000, Theorem 7], proved[1] that $\mathbf{J} * \mathbf{D}$ is a p-variety of finite semigroups, i.e. for any finite semigroup $(S, *)$, $\mathcal{W}((S, *)) \subseteq \mathcal{P}(\mathbf{J} * \mathbf{D})$ if and only if $(S, *) \in \mathbf{J} * \mathbf{D}$. This implies the lemma below.

Lemma 5.3.4. *Let $(S, *)$ be a finite semigroup such that $(S, *) \notin \mathbf{J} * \mathbf{D}$. Then, we have $\mathcal{W}((S, *)) \nsubseteq \mathcal{P}(\mathbf{J})$.*

Moreover $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{Q}(\mathbf{J} * \mathbf{D}))$.

These two lemmata together finally show that if the set of word problems over some finite semigroup is contained in $\mathcal{P}(\mathbf{J})$, then this semigroup must belong to the variety of finite semigroups $\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}}$ (intersection of two varieties of finite semigroups being trivially seen to be itself a variety of finite semigroups).

In fact, Lemma 5.3.2 gives us good reasons to conjecture that the converse is also true, i.e. that for any finite semigroup $(S, *)$, $(S, *) \in \mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}}$ if and only if $\mathcal{W}((S, *)) \subseteq \mathcal{P}(\mathbf{J})$. Since a word problem over some semigroup in $\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}}$ is a language in $\mathcal{L}(\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}})$, this would precisely follow from a proof of this conjecture in addition to what we already know.

Conjecture 1. *For any $L \in \mathcal{L}(\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}})$, there exists a sequence $(P_n)_{n \in \mathbb{N}}$ of programs over some monoid in $\mathbf{J}$ that decides L.*

In fact, we even conjecture the following about the precise characterisation of the *lm*-variety of regular languages p-recognised by a monoid taken from $\mathcal{P}(\mathbf{J})$.

Conjecture 2. $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg} = \mathcal{L}(\mathbf{Q}(\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}}))$.

This second conjecture does not directly follow from the first one, because we do not know whether $\mathbf{Q}(\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}}) = \langle \mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}} \rangle_{ne} * \mathbf{MOD}$ or not, a fact that would allow us to deduce the truth of Conjecture 2 straightforwardly from the truth of Conjecture 1 by Proposition 3.4.5.

Let us try to explain what we mean when we say we have "good reasons" for at least Conjecture 1 to be true. Let Σ be an alphabet and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$); we define $[u_1, \ldots, u_k]_\Sigma = \Sigma^* u_1 \Sigma^* \cdots \Sigma^* u_k \Sigma^*$. The *ne*-variety of languages $\mathcal{L}(\mathbf{J} * \mathbf{D})$ is the *ne*-variety of *dot-depth one languages*, that are Boolean combinations of languages of the form $\Sigma^* u$, $u\Sigma^*$ and $[u_1, \ldots, u_k]_\Sigma$ for Σ an alphabet, $k \in \mathbb{N}_{>0}$ and $u, u_1, \ldots, u_k \in \Sigma^+$ (see Straubing [1985], Maciel et al. [2000], Pin [2017]). Though, for a given alphabet Σ, we cannot decide

[1] To be entirely rigorous, this is not exactly what they proved; to get precisely this, we need to appeal to results of Straubing in Straubing [1985] identifying $\mathbf{J} * \mathbf{D}$ as the variety of finite semigroups generated by the syntactic semigroups of dot-depth one languages.

whether some word $u \in \Sigma^+$ of length at least 2 appears as a factor of any given word w in Σ^* with programs over finite $\mathfrak{J}$-trivial monoids (because $\Sigma^* u \Sigma^* \notin \mathcal{L}(\mathbf{QDA})$), Lemma 5.3.2 and the possibilities offered by the "feedback-sweeping" technique give the impression that we can do it when we are guaranteed that u appears at most a fixed number of times in w (i.e. there exists some fixed $l \in \mathbb{N}$ such that u^l is not a subword of w), which seems somehow to be the effect of intersecting $\mathbf{J} * \mathbf{D}$ with $\langle \mathbf{DA} \rangle_{\mathbf{S}}$ on the class of associated languages. This motivates the definition of *threshold dot-depth one languages*.

5.3.2 Threshold dot-depth one languages

The idea behind the definition of threshold dot-depth one languages is that we take the basic building blocks of dot-depth one languages, of the form $[u_1, \ldots, u_k]_\Sigma$ for Σ an alphabet, $k \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$ (defined at the end of the previous section, just above), and restrict them in the sense that, given some $l \in \mathbb{N}_{>0}$, membership of a word does really depend on the presence of a given word u_i as a factor if and only if it appears less than l times as a subword, as defined formally below.

Definition 5.3.5. Let Σ be an alphabet. For all $u \in \Sigma^+$ and $l \in \mathbb{N}_{>0}$, we define $[u]_{\Sigma,l}$ to be the language of words over Σ containing l copies of u as a subword or u as a factor, i.e. $[u]_{\Sigma,l} = \Sigma^* u \Sigma^* \cup u^l \sqcup\!\sqcup \Sigma^*$, as well as the following three languages:

$$
\begin{aligned}
[u[_{\Sigma,l} &= \Sigma^* u \\
]u]_{\Sigma,l} &= u\Sigma^* \\
]u[_{\Sigma,l} &= u \ .
\end{aligned}
$$

Then, for all $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}, k \geq 2$) and $l \in \mathbb{N}_{>0}$, we also define

$$
\begin{aligned}
[u_1, \ldots, u_k]_{\Sigma,l} &= [u_1]_{\Sigma,l} \ \cdots [u_k]_{\Sigma,l} \\
[u_1, \ldots, u_k[_{\Sigma,l} &= [u_1]_{\Sigma,l} \ \cdots [u_{k-1}]_{\Sigma,l} [u_k[_{\Sigma,l} \\
]u_1, \ldots, u_k]_{\Sigma,l} &=]u_1]_{\Sigma,l} [u_2]_{\Sigma,l} \ \cdots [u_k]_{\Sigma,l} \\
]u_1, \ldots, u_k[_{\Sigma,l} &=]u_1]_{\Sigma,l} [u_2]_{\Sigma,l} \ \cdots [u_{k-1}]_{\Sigma,l} [u_k[_{\Sigma,l} \ .
\end{aligned}
$$

We additionally use the convention for $l \in \mathbb{N}_{>0}$ and ε an empty list of words in Σ^+, that $[\varepsilon]_{\Sigma,l} = \Sigma^*$ and $[\varepsilon[_{\Sigma,l} =]\varepsilon]_{\Sigma,l} =]\varepsilon[_{\Sigma,l} = \emptyset$.

It is obvious that for each Σ an alphabet, $k \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$, the language $[u_1, \ldots, u_k]_{\Sigma,1}$ is equal to $u_1 \cdots u_k \sqcup\!\sqcup \Sigma^*$. The language $[ab, c]_{\{a,b,c\},3}$ contains all words

over $\{a, b, c\}$ such that they contain a letter c verifying that in the prefix up to that letter, *ababab* appears as a subword or *ab* appears as a factor. Finally, the language $(a + b)^* ac^+$ over $\{a, b, c\}$, that Lemma 5.3.1 shows to belong to $\mathcal{P}(\mathbf{J})$, is equal to $[c, a]_{\{a,b,c\},2}{}^{\complement} \cap [c, b]_{\{a,b,c\},2}{}^{\complement} \cap [ac]_{\{a,b,c\},2}$.

We can then define a *threshold dot-depth one language* as any language that is a Boolean combination of languages of the form $[u_1, \dots, u_k]_{\Sigma,l}$, $[u_1, \dots, u_k[_{\Sigma,l}$, $]u_1, \dots, u_k]_{\Sigma,l}$ or $]u_1, \dots, u_k[_{\Sigma,l}$ for Σ an alphabet, $k, l \in \mathbb{N}_{>0}$ and $u_1, \dots, u_k \in \Sigma^+$. Intuitively, a threshold dot-depth one language is a language of dot-depth one where detection of the presence of a given factor works if and only if it does not appear too often as a subword.

Confirming the intuition briefly given at the end of the previous subsection, the technique of "feedback-sweeping" can indeed be generalised and pushed further to prove that the whole class of all threshold dot-depth one languages is contained in $\mathcal{P}(\mathbf{J})$, and we dedicate the remainder of this section to prove it. Let us just say a few words about how this could help to prove Conjecture 1. Our intuition, as we explained in the previous subsection, leads us to believe that, in fact, the class of all threshold dot-depth one languages is exactly the *ne*-variety of languages $\mathcal{L}(\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}})$. Sadly, we were not able to prove it and in the next section, we shall only give a partial result that supports this belief.

Let us now move on to the proof of the following theorem.

Theorem 5.3.6. *The class of all threshold dot-depth one languages is contained in* $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg}$.

Knowing that $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg}$ is closed under Boolean operations (see Proposition 3.4.2), our goal is to prove, given an alphabet Σ, $l \in \mathbb{N}_{>0}$ and $u_1, \dots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$), that $[u_1, \dots, u_k]_{\Sigma,l}$ is in $\mathcal{P}(\mathbf{J})$, and we can then easily handle the case of $[u_1, \dots, u_k[_{\Sigma,l}$, $]u_1, \dots, u_k]_{\Sigma,l}$ and $]u_1, \dots, u_k[_{\Sigma,l}$. To do this, we need to put $[u_1, \dots, u_k]_{\Sigma,l}$ in some normal form. It is readily seen that $[u_1, \dots, u_k]_{\Sigma,l} = \bigcup_{q_1,\dots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)}$ where each $L^{(\Sigma,l)}_{(u_i,q_i)}$ for $i \in [k]$ is defined in the following way.

Definition 5.3.7. Let Σ be an alphabet.

For all $u \in \Sigma^+$, $l \in \mathbb{N}_{>0}$ and $\alpha \in [l]$,

$$L^{(\Sigma,l)}_{(u,\alpha)} = \begin{cases} \Sigma^* u \Sigma^* & \text{if } \alpha < l \\ u^l \sqcup\!\sqcup \Sigma^* & \text{otherwise} \end{cases}.$$

Building directly a sequence of programs over a finite $\mathfrak{J}$-trivial monoid that decides $L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)}$ for some $q_1, \dots, q_k \in \{1, l\}$ seems however tricky. We need to split things

even further by controlling precisely how many times each u_i for $i \in [k]$ appears in the right place when it does less than l times. To do this, we consider, for each $\alpha_1, \dots, \alpha_k \in [l]$, the language $R^{(\Sigma,l)}_{(u_1,\alpha_1),\dots,(u_k,\alpha_k)}$ of words w over Σ containing ${u_1}^{\alpha_1} \cdots {u_k}^{\alpha_k}$ as a subword and such that for each $i \in [k]$ verifying $\alpha_i < l$, given the shortest prefix v_0 of w containing ${u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}$ as a subword and the shortest suffix v_2 of w containing ${u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}$ as a subword, the factor v_1 of w such that $w = v_0 v_1 v_2$ does not contain more than α_i times u_i as a subword. This is formalised below.

Definition 5.3.8. Let Σ be an alphabet.

For all $u_1, \dots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$), $l \in \mathbb{N}_{>0}$, $\alpha_1, \dots, \alpha_k \in [l]$, we set

$$R^{(\Sigma,l)}_{(u_1,\alpha_1),\dots,(u_k,\alpha_k)} = ({u_1}^{\alpha_1} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^* \cap \bigcap_{i \in [k], \alpha_i < l} \big(({u_1}^{\alpha_1} \cdots {u_i}^{\alpha_i+1} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\big)^{\complement} .$$

Now, for given $\alpha_1, \dots, \alpha_k \in [l]$, we are interested in the words of $R^{(\Sigma,l)}_{(u_1,\alpha_1),\dots,(u_k,\alpha_k)}$ that are such that for each $i \in [k]$ verifying $\alpha_i < l$, u_i indeed appears as a factor in the right place. We thus introduce a last language $S^{(\Sigma,l)}_{(u_1,\alpha_k),\dots,(u_k,\alpha_k)}$ of words over Σ such that for each $i \in [k]$ verifying $\alpha_i < l$, it has a prefix containing ${u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}$ as a subword, a suffix containing ${u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}$ as a subword, and a factor u_i in between. We formalise this in the definition that follows.

Definition 5.3.9. Let Σ be an alphabet.

For all $u_1, \dots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$), $l \in \mathbb{N}_{>0}$, $\alpha_1, \dots, \alpha_k \in [l]$, we set

$$S^{(\Sigma,l)}_{(u_1,\alpha_1),\dots,(u_k,\alpha_k)} = \bigcap_{i \in [k], \alpha_i < l} \big(({u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}) \sqcup\!\sqcup \Sigma^*\big) u_i \big(({u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\big)$$

(where $({u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}) \sqcup\!\sqcup \Sigma^* = \Sigma^*$ when $i = 1$ and $({u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^* = \Sigma^*$ when $i = k$).

Using these two previous definitions, we now have the normal form we were looking for to prove Theorem 5.3.6: $[u_1, \dots, u_k]_{\Sigma,l}$ is equal to the union, over all $\alpha_1, \dots, \alpha_k \in [l]$, of the intersection of $R^{(\Sigma,l)}_{(u_1,\alpha_1),\dots,(u_k,\alpha_k)}$ and $S^{(\Sigma,l)}_{(u_1,\alpha_k),\dots,(u_k,\alpha_k)}$. To prove that $[u_1, \dots, u_k]_{\Sigma,l} \in \mathcal{P}(\mathbf{J})$, we can then prove $R^{(\Sigma,l)}_{(u_1,\alpha_1),\dots,(u_k,\alpha_k)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_k),\dots,(u_k,\alpha_k)}$ to be in $\mathcal{P}(\mathbf{J})$ for all $\alpha_1, \dots, \alpha_k \in [l]$ and conclude by closure of $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg}$ under union (see Proposition 3.4.2).

Though rather intuitive, the correctness of this way of decomposing $[u_1, \dots, u_k]_{\Sigma,l}$ is not so straightforward to prove and, actually, we can only prove it when for each $i \in [k]$, the letters in u_i are all distinct.

Lemma 5.3.10. *Let Σ be an alphabet, $l \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$) such that for each $i \in [k]$, the letters in u_i are all distinct. Then,*

$$\bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)} = \bigcup_{\alpha_1,\ldots,\alpha_k \in [l]} (R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)}) \ .$$

Proof. Let Σ be an alphabet and $l \in \mathbb{N}_{>0}$. We prove it by induction on $k \in \mathbb{N}_{>0}$.

Base case $k = 1$. Let $u_1 \in \Sigma^+$ such that the letters in u_1 are all distinct. It is clear that

$$\begin{aligned}
\bigcup_{q_1 \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} &= (\Sigma^* u_1 \Sigma^* \cup {u_1}^l \sqcup\!\sqcup \Sigma^*) \\
&= \Big(\bigcup_{\alpha_1=1}^{l-1} \big({u_1}^{\alpha_1} \sqcup\!\sqcup \Sigma^* \cap ({u_1}^{\alpha_1+1} \sqcup\!\sqcup \Sigma^*)^{\complement} \cap \Sigma^* u_1 \Sigma^* \big) \cup ({u_1}^l \sqcup\!\sqcup \Sigma^*) \Big) \\
&= \bigcup_{\alpha_1 \in [l]} (R^{(\Sigma,l)}_{(u_1,\alpha_1)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1)}) \ .
\end{aligned}$$

Induction. Let $k \in \mathbb{N}_{>0}$ and assume that for all $u_1, \ldots, u_k \in \Sigma^+$ such that for each $i \in [k]$, the letters in u_i are all distinct, we have

$$\bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)} = \bigcup_{\alpha_1,\ldots,\alpha_k \in [l]} (R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)}) \ .$$

Let now $u_1, \ldots, u_{k+1} \in \Sigma^+$ such that for each $i \in [k+1]$, the letters in u_i are all distinct.

Right-to-left inclusion. Let

$$w \in \bigcup_{\alpha_1,\ldots,\alpha_{k+1} \in [l]} (R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_{k+1},\alpha_{k+1})} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_{k+1},\alpha_{k+1})}) \ .$$

Let $\alpha_1, \ldots, \alpha_{k+1} \in [l]$ witnessing this fact. As $w \in R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_{k+1},\alpha_{k+1})}$, we can decompose it as $w = xy$ where $x \in ({u_1}^{\alpha_1} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*$, $y \in {u_{k+1}}^{\alpha_{k+1}} \sqcup\!\sqcup \Sigma^*$ and $|y|$ is minimal. What we are going to do is, on the one hand, to prove that $x \in R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)}$, so that we can apply the inductive hypothesis on x and get that there exist $q_1, \ldots, q_k \in \{1, l\}$ such that $x \in L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)}$; and, on the other hand, we are going to prove that there exists $q_{k+1} \in \{1, l\}$ verifying $y \in L^{(\Sigma,l)}_{(u_{k+1},q_{k+1})}$. We now spell out the details.

For each $i \in [k], \alpha_i < l$, we have $x \notin ({u_1}^{\alpha_1} \cdots {u_i}^{\alpha_i+1} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*$, otherwise we would have $w = xy \in ({u_1}^{\alpha_1} \cdots {u_i}^{\alpha_i+1} \cdots {u_{k+1}}^{\alpha_{k+1}}) \sqcup\!\sqcup \Sigma^*$. Also, for all $i \in [k], \alpha_i < l$, we have that $x \in \big(({u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}) \sqcup\!\sqcup \Sigma^*\big) u_i \big(({u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\big)$, otherwise it would mean that $y = y_1 y_2$ with $|y_1| > 0$, $xy_1 \in \big(({u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}) \sqcup\!\sqcup \Sigma^*\big) u_i \big(({u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\big)$ and $y_2 \in {u_{k+1}}^{\alpha_{k+1}} \sqcup\!\sqcup \Sigma^*$, contradicting the minimality of $|y|$. So $x \in R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)}$, which means by inductive hypothesis that there exist $q_1, \ldots, q_k \in \{1, l\}$ such that $x \in L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)}$.

Remember now that the letters in u_{k+1} are all distinct. If $\alpha_{k+1} < l$, since $w \in \big(({u_1}^{\alpha_1} \ldots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\big) u_{k+1} \Sigma^*$, we must have $y \in \Sigma^* u_{k+1} \Sigma^*$. Indeed, by minimality of $|y|$, y starts with the first letter of u_{k+1}, which has pairwise distinct letters, so that u_{k+1} cannot appear as a factor of xy partly in x and partly in y; so if it were the case that y does not contain u_{k+1} as a factor, we would have $x \in \big(({u_1}^{\alpha_1} \ldots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\big) u_{k+1} \Sigma^*$, so that $xy = w \in ({u_1}^{\alpha_1} \ldots {u_k}^{\alpha_k} {u_{k+1}}^{\alpha_{k+1}+1}) \sqcup\!\sqcup \Sigma^*$, a contradiction with the hypothesis on w. Hence, $y \in L^{(\Sigma,l)}_{(u_{k+1},\alpha_{k+1})}$. If $\alpha_{k+1} = l$, then $y \in {u_{k+1}}^{\alpha_{k+1}} \sqcup\!\sqcup \Sigma^* = L^{(\Sigma,l)}_{(u_{k+1},\alpha_{k+1})}$. So, if we set $q_{k+1} = \begin{cases} 1 & \text{if } \alpha_{k+1} < l \\ l & \text{otherwise} \end{cases}$, then we get that $y \in L^{(\Sigma,l)}_{(u_{k+1},q_{k+1})}$.

We can conclude that $w = xy \in L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)} L^{(\Sigma,l)}_{(u_{k+1},q_{k+1})}$.

Left-to-right inclusion. Let $w \in \bigcup_{q_1,\ldots,q_{k+1} \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_{k+1},q_{k+1})}$. The rough idea of our proof here is to take $\alpha_{k+1} \in [l]$ the biggest integer in $[l]$ such that $w \in \big(\bigcup_{q_1,\ldots,q_{k+1} \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)}\big)({u_{k+1}}^{\alpha_{k+1}} \sqcup\!\sqcup \Sigma^*)$ and decompose w as $w = xy$ where $x \in \bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)}$, $y \in {u_{k+1}}^{\alpha_{k+1}} \sqcup\!\sqcup \Sigma^*$ and $|y|$ is minimal. By inductive hypothesis, we know there exist $\alpha_1, \ldots, \alpha_k \in [l]$ such that $x \in R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)}$ and we then prove that $xy \in R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_{k+1},\alpha_{k+1})} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_{k+1},\alpha_{k+1})}$ by distinguishing between the case in which $\alpha_{k+1} = l$ and the case in which $\alpha_{k+1} < l$. The first one is easy to handle, the second one is much trickier.

We now spell out the details.

- Suppose we have

$$\begin{aligned} w \in & \bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \ldots L^{(\Sigma,l)}_{(u_k,q_k)} L^{(\Sigma,l)}_{(u_{k+1},l)} \\ = & \Big(\bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \ldots L^{(\Sigma,l)}_{(u_k,q_k)} \Big) ({u_{k+1}}^{l} \sqcup\!\sqcup \Sigma^*) \ . \end{aligned}$$

Then w can be decomposed as $w = xy$ where $x \in \bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)}$, $y \in$

${u_{k+1}}^l \sqcup\!\sqcup \Sigma^*$ and $|y|$ is minimal. So by inductive hypothesis, there exist $\alpha_1, \ldots, \alpha_k \in [l]$ such that $x \in R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)}$. Observe that this means we have $w \in ({u_1}^{\alpha_1} \cdots {u_k}^{\alpha_k} {u_{k+1}}^l) \sqcup\!\sqcup \Sigma^*$ and for each $i \in [k], \alpha_i < l$, $w \notin ({u_1}^{\alpha_1} \cdots {u_i}^{\alpha_i+1} \cdots {u_k}^{\alpha_k} {u_{k+1}}^l) \sqcup\!\sqcup \Sigma^*$, otherwise it would mean that $x \in ({u_1}^{\alpha_1} \cdots {u_i}^{\alpha_i+1} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*$ by minimality of $|y|$. Similarly, for all $i \in [k], \alpha_i < l$, it is obvious that we have

$$w = xy \in \big(({u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}) \sqcup\!\sqcup \Sigma^*\big) u_i \big(({u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k} {u_{k+1}}^l) \sqcup\!\sqcup \Sigma^*\big)$$

as $x \in \big(({u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}) \sqcup\!\sqcup \Sigma^*\big) u_i \big(({u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\big)$ and $y \in {u_{k+1}}^l \sqcup\!\sqcup \Sigma^*$. Hence, $w \in R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k),(u_{k+1},l)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k),(u_{k+1},l)}$.

- Or we have

$$\begin{aligned} w \notin & \bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)} L^{(\Sigma,l)}_{(u_{k+1},l)} \\ = & \Big(\bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)} \Big) ({u_{k+1}}^l \sqcup\!\sqcup \Sigma^*) \end{aligned}$$

but

$$w \in \bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)} L^{(\Sigma,l)}_{(u_{k+1},1)} \ .$$

Let $\alpha_{k+1} \in [l-1]$ be the biggest integer in $[l-1]$ such that

$$w \in \Big(\bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)} \Big) ({u_{k+1}}^{\alpha_{k+1}} \sqcup\!\sqcup \Sigma^*)$$

which does exist by hypothesis. We can decompose w as $w = xy$ where $x \in \bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)}$, $y \in {u_{k+1}}^{\alpha_{k+1}} \sqcup\!\sqcup \Sigma^*$ and $|y|$ is minimal. So by inductive hypothesis, there exist $\alpha_1, \ldots, \alpha_k \in [l]$ such that $x \in R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)}$. We are now going to prove that

$$w = xy \in R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_{k+1},\alpha_{k+1})} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_{k+1},\alpha_{k+1})} \ .$$

Among the obvious things to observe is that we have $w \in ({u_1}^{\alpha_1} \cdots {u_k}^{\alpha_k} {u_{k+1}}^{\alpha_{k+1}}) \sqcup\!\sqcup \Sigma^*$ and for each $i \in [k], \alpha_i < l$,

$$w \notin ({u_1}^{\alpha_1} \cdots {u_i}^{\alpha_i+1} \cdots {u_k}^{\alpha_k} {u_{k+1}}^{\alpha_{k+1}}) \sqcup\!\sqcup \Sigma^* \ ,$$

otherwise it would mean that $x \in ({u_1}^{\alpha_1} \cdots {u_i}^{\alpha_i+1} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*$ by minimality of $|y|$. Similarly, for all $i \in [k], \alpha_i < l$, it is obvious that we have

$$w = xy \in \big(({u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}) \sqcup\!\sqcup \Sigma^*\big) u_i \big(({u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k} {u_{k+1}}^{\alpha_{k+1}}) \sqcup\!\sqcup \Sigma^*\big)$$

as $x \in \big(({u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}) \sqcup\!\sqcup \Sigma^*\big) u_i \big(({u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\big)$ and $y \in {u_{k+1}}^{\alpha_{k+1}} \sqcup\!\sqcup \Sigma^*$. Now let us show that we have $y \in \Sigma^* u_{k+1} \Sigma^*$. Assume it weren't the case: the letters in u_{k+1} are pairwise distinct and moreover y starts with the first letter of u_{k+1} by minimality of $|y|$, so u_{k+1} cannot appear as a factor of xy partly in x and partly in y and, additionally,

$$\begin{aligned} w &\in \bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(l)}_{(u_k,q_k)} L^{(\Sigma,l)}_{(u_{k+1},1)} \\ &= \Big(\bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)} \Big) \Sigma^* u_{k+1} \Sigma^* \ , \end{aligned}$$

so we would have $x \in (\bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)}) \Sigma^* u_{k+1} \Sigma^*$. But this either contradicts the maximality of α_{k+1} or the fact that

$$w \notin \Big(\bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)} \Big) ({u_{k+1}}^{l} \sqcup\!\sqcup \Sigma^*) \ .$$

Thus, we have $w = xy \in \big(({u_1}^{\alpha_1} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\big) u_{k+1} \Sigma^*$ as $x \in ({u_1}^{\alpha_1} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*$.

Let us finish with the trickiest part, showing that $w \notin ({u_1}^{\alpha_1} \cdots {u_k}^{\alpha_k} {u_{k+1}}^{\alpha_{k+1}+1}) \sqcup\!\sqcup \Sigma^*$. Assume that $w \in ({u_1}^{\alpha_1} \cdots {u_k}^{\alpha_k} {u_{k+1}}^{\alpha_{k+1}+1}) \sqcup\!\sqcup \Sigma^*$. We then have that $x \in ({u_1}^{\alpha_1} \cdots {u_k}^{\alpha_k} u_{k+1}) \sqcup\!\sqcup \Sigma^*$, otherwise it would mean that $y = y_1 y_2$ with $|y_1| > 0$, $xy_1 \in ({u_1}^{\alpha_1} \cdots {u_k}^{\alpha_k} u_{k+1}) \sqcup\!\sqcup \Sigma^*$ and $y_2 \in {u_{k+1}}^{\alpha_{k+1}} \sqcup\!\sqcup \Sigma^*$, contradicting the minimality of $|y|$. We can decompose x as $x = x_1 x_2$ where $x_1 \in ({u_1}^{\alpha_1} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*$, $x_2 \in u_{k+1} \sqcup\!\sqcup \Sigma^*$ and $|x_2|$ is minimal. We claim that, actually, $x_1 \in R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)}$, so that by inductive hypothesis, $x_1 \in \bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)}$. But since $x_2 y \in {u_{k+1}}^{\alpha_{k+1}+1} \sqcup\!\sqcup \Sigma^*$, this means that

$$w = x_1 x_2 y \in \Big(\bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)} \Big) ({u_{k+1}}^{\alpha_{k+1}+1} \sqcup\!\sqcup \Sigma^*) \ ,$$

contradicting the maximality of α_{k+1} or the fact that

$$w \notin \Big(\bigcup_{q_1,\dots,q_k \in \{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_k,q_k)} \Big) ({u_{k+1}}^l \sqcup\!\sqcup \Sigma^*) \ .$$

So we can conclude that $w \notin ({u_1}^{\alpha_1} \cdots {u_k}^{\alpha_k} {u_{k+1}}^{\alpha_{k+1}+1}) \sqcup\!\sqcup \Sigma^*$.

The claim that $x_1 \in R^{(\Sigma,l)}_{(u_1,\alpha_1),\dots,(u_k,\alpha_k)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\dots,(u_k,\alpha_k)}$ remains to be shown. We directly see that for all $i \in [k], \alpha_i < l$, $x_1 \notin ({u_1}^{\alpha_1} \cdots {u_i}^{\alpha_i+1} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*$, otherwise it would mean that $x \in ({u_1}^{\alpha_1} \cdots {u_i}^{\alpha_i+1} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*$. Let now $i \in [k], \alpha_i < l$, and assume that $x_1 \notin \big(({u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}) \sqcup\!\sqcup \Sigma^*\big) u_i \big(({u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\big)$. We can decompose x_1 as $x_1 = x_{1,1}x_{1,2}$ where $x_{1,1} \in ({u_1}^{\alpha_1} \cdots {u_i}^{\alpha_i}) \sqcup\!\sqcup \Sigma^*$, $x_{1,2} \in ({u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*$ and $|x_{1,1}|$ is minimal. By hypothesis, we have $x_{1,1} \notin \big(({u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}) \sqcup\!\sqcup \Sigma^*\big) u_i \Sigma^*$, otherwise we would have

$$x_1 = x_{1,1}x_{1,2} \in \big(({u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}) \sqcup\!\sqcup \Sigma^*\big) u_i \big(({u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\big) \ .$$

As previously, the letters in u_i are pairwise distinct, and $x_{1,1}$ ends with the last letter of u_i by minimality of $|x_{1,1}|$, so u_i cannot appear as a factor of x partly in $x_{1,1}$ and partly in $x_{1,2}x_2$. Thus, we have that

$$x_{1,2}x_2 \in \Sigma^* u_i \big(({u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\big)$$

because we know that $x \in \big(({u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}) \sqcup\!\sqcup \Sigma^*\big) u_i \big(({u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\big)$. But this means that $x = x_{1,1}x_{1,2}x_2 \in ({u_1}^{\alpha_1} \cdots {u_i}^{\alpha_i+1} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*$, a contradiction. Hence, we can deduce that for all $i \in [k], \alpha_i < l$, $x_1 \in \big(({u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}) \sqcup\!\sqcup \Sigma^*\big) u_i \big(({u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\big)$. This finishes to show that

$$x_1 \in R^{(\Sigma,l)}_{(u_1,\alpha_1),\dots,(u_k,\alpha_k)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\dots,(u_k,\alpha_k)} \ .$$

Putting all together, we indeed also have that

$$w \in R^{(\Sigma,l)}_{(u_1,\alpha_1),\dots,(u_{k+1},\alpha_{k+1})} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\dots,(u_{k+1},\alpha_{k+1})}$$

in the present case.

In conclusion, in both cases,

$$w \in \bigcup_{\alpha_1,\ldots,\alpha_{k+1}\in[l]} (R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_{k+1},\alpha_{k+1})} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_{k+1},\alpha_{k+1})}) \ .$$

So we can finally conclude that

$$\begin{aligned} &\bigcup_{q_1,\ldots,q_k\in\{1,l\}} L^{(\Sigma,l)}_{(u_1,q_1)} \cdots L^{(\Sigma,l)}_{(u_{k+1},q_{k+1})} \\ =& \bigcup_{\alpha_1,\ldots,\alpha_{k+1}\in[l]} (R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_{k+1},\alpha_{k+1})} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_{k+1},\alpha_{k+1})}) \ . \end{aligned}$$

This concludes the proof of the lemma. □

Before proving the main lemma to be used in the proof of Theorem 5.3.6, we need a useful decomposition lemma, that is straightforward to prove.

Lemma 5.3.11. *Let Σ be an alphabet and $u \in \Sigma^+$. Then, for all $\alpha \in \mathbb{N}_{>0}$, each $w \in u^\alpha \sqcup\!\sqcup \Sigma^* \cap (u^{\alpha+1} \sqcup\!\sqcup \Sigma^*)^{\complement}$ verifies*

$$w = (\prod_{i=1}^{\alpha} \prod_{j=1}^{|u|} (v_{i,j}u_j))y$$

where $v_{i,j} \in (\Sigma \setminus \{u_j\})^*$ for all $i \in [\alpha]$, $j \in [|u|]$ and $y \in \bigcup_{i=1}^{|u|} \Big(\prod_{j=1}^{i-1} \big((\Sigma \setminus \{u_j\})^* u_j\big)(\Sigma \setminus \{u_i\})^* \Big)$.

Proof. Let Σ be an alphabet and $u \in \Sigma^+$.

Take $\alpha \in \mathbb{N}_{>0}$ and $w \in u^\alpha \sqcup\!\sqcup \Sigma^* \cap (u^{\alpha+1} \sqcup\!\sqcup \Sigma^*)^{\complement}$.

As $w \in u^\alpha \sqcup\!\sqcup \Sigma^*$, w can be decomposed as $w = xy$ where $x \in u^\alpha \sqcup\!\sqcup \Sigma^*$ and $|x|$ is minimal. Then, it is clearly necessarily the case that $x = \prod_{i=1}^{\alpha} \prod_{j=1}^{|u|} (v_{i,j}u_j)$ with $v_{i,j} \in (\Sigma \setminus \{u_j\})^*$ for all $i \in [\alpha]$, $j \in [|u|]$. Moreover, as $xy \notin u^{\alpha+1} \sqcup\!\sqcup \Sigma^*$, we necessarily have that $y \notin u \sqcup\!\sqcup \Sigma^*$, so that there exists some $i \in [|u|]$ verifying that $u_1 \cdots u_{i-1}$ is a subword of y but not $u_1 \cdots u_i$. Thus, we have that $y \in \prod_{j=1}^{i-1} \big((\Sigma \setminus \{u_j\})^* u_j\big)(\Sigma \setminus \{u_i\})^*$.

This concludes the proof. □

Remember that our goal is to prove, given an alphabet Σ, $l \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$) such that for each $i \in [k]$, the letters in u_i are all distinct, that for any $\alpha_1, \ldots, \alpha_k \in [l]$, the language $R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_k),\ldots,(u_k,\alpha_k)}$ is in $\mathcal{P}(\mathbf{J})$. The way $R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)}$ and $S^{(\Sigma,l)}_{(u_1,\alpha_k),\ldots,(u_k,\alpha_k)}$ are defined allows us to reason as follows. For each

$i \in [k]$ verifying $\alpha_i < l$, let L_i be the language of words w over Σ containing $x_{i,1}{u_i}^{\alpha_i}x_{i,2}$ as a subword and such that, given the shortest prefix v_0 of w containing $x_{i,1}$ as a subword and the shortest suffix v_2 of w containing $x_{i,2}$ as a subword, the factor v_1 of w such that $w = v_0v_1v_2$ does not contain more than α_i times u_i as a subword and has u_i as a factor, with $x_{i,1} = {u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}$ and $x_{i,2} = {u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}$. If we manage to prove that for each $i \in [k]$ verifying $\alpha_i < l$, $L_i \in \mathcal{P}(\mathbf{J})$, then we can conclude that

$$R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_k),\ldots,(u_k,\alpha_k)} = ({u_1}^{\alpha_1} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^* \cap \bigcap_{i\in[k],\alpha_i<l} L_i$$

does belong to $\mathcal{P}(\mathbf{J})$ by closure of $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg}$ under intersection, Proposition 3.4.2. The lemma that follows, the main lemma in the proof of Theorem 5.3.6, exactly shows that $L_i \in \mathcal{P}(\mathbf{J})$ for each $i \in [k]$ verifying $\alpha_i < l$. The proof of that lemma crucially uses the "feedback sweeping" technique, but note that we actually don't know how to prove it when we do not enforce that for each $i \in [k]$, the letters in u_i are all distinct.

Lemma 5.3.12. *Let Σ be an alphabet and $u \in \Sigma^+$ such that its letters are all distinct. For all $\alpha \in \mathbb{N}_{>0}$ and $x_1, x_2 \in \Sigma^*$, we have*

$$(x_1u^\alpha x_2) \sqcup\!\sqcup \Sigma^* \cap \left((x_1u^{\alpha+1}x_2) \sqcup\!\sqcup \Sigma^*\right)^{\complement} \cap (x_1 \sqcup\!\sqcup \Sigma^*)u(x_2 \sqcup\!\sqcup \Sigma^*) \in \mathcal{P}(\mathbf{J}) \ .$$

Proof. Let Σ be an alphabet and $u \in \Sigma^+$ such that its letters are all distinct. Let $\alpha \in \mathbb{N}_{>0}$ and $x_1, x_2 \in \Sigma^*$. We let

$$L = (x_1u^\alpha x_2) \sqcup\!\sqcup \Sigma^* \cap \left((x_1u^{\alpha+1}x_2) \sqcup\!\sqcup \Sigma^*\right)^{\complement} \cap (x_1 \sqcup\!\sqcup \Sigma^*)u(x_2 \sqcup\!\sqcup \Sigma^*) \ .$$

If $|u| = 1$, the lemma follows trivially because L is piecewise testable and hence belongs to $\mathcal{L}(\mathbf{J})$, so we assume $|u| > 1$.

For each letter $a \in \Sigma$, we shall use $2|u| - 1$ distinct decorated letters of the form $a^{(i)}$ for some $i \in [\![0, 2|u| - 2]\!]$, using the convention that $a^{(0)} = a$; of course, for two distinct letters $a, b \in \Sigma$, we have that $a^{(i)}$ and $b^{(j)}$ are distinct for all $i, j \in [\![0, 2|u| - 2]\!]$. We denote by A the alphabet of these decorated letters. The main idea of the proof is, for a given input length $n \in \mathbb{N}$, to build an A-program Ψ_n over Σ^n such that, given an input word $w \in \Sigma^n$, it first ouputs the $|u| - 1$ first letters of w and then, for each i going from $|u|$ to n, outputs w_i, followed by $w^{(1)}_{i-1} \cdots w^{(|u|-1)}_{i-|u|+1}$ (a "sweep" of $|u| - 1$ letters backwards down to position $i - |u| + 1$, decorating the letters incrementally) and finally by $w^{(|u|)}_{i-|u|+2} \cdots w^{(2|u|-2)}_i$ (a "sweep" forwards up to position i, continuing the incremental decoration of the letters). The idea behind this way of rearranging and decorating letters

is that, given an input word $w \in \Sigma^n$, as long as we make sure that w and thus $\Psi_n(w)$ do contain $x_1 u^\alpha x_2$ as a subword but not $x_1 u^{\alpha+1} x_2$, then $\Psi_n(w)$ can be decomposed as $\Psi_n(w) = y_1 z y_2$ where $y_1 \in x_1 \sqcup\!\sqcup \Sigma^*$, $y_2 \in x_2 \sqcup\!\sqcup \Sigma^*$, and $|y_1|, |y_2|$ are minimal such that z contains $u^\beta u_{|u|-1}^{(1)} \cdots u_1^{(|u|-1)} u_2^{(|u|)} \cdots u_{|u|}^{(2|u|-2)} u^{\alpha-\beta}$ as a subword for some $\beta \in [\alpha]$ if and only if $w \in (x_1 \sqcup\!\sqcup \Sigma^*) u (x_2 \sqcup\!\sqcup \Sigma^*)$. This means we can check whether $w \in L$ by testing whether w belongs to some fixed piecewise testable language over A. Let's now write the proof formally.

For each $i \in [\![0, 2|u| - 2]\!]$, let

$$\begin{aligned} f^{(i)} \colon \Sigma &\to A \\ a &\mapsto a^{(i)} \end{aligned} .$$

For all $i \in \mathbb{N}, i \geq |u|$, we define

$$\Phi_i = (i, f^{(0)}) \prod_{j=1}^{|u|-1} (i-j, f^{(j)}) \prod_{j=2}^{|u|} (i - |u| + j, f^{(|u|+j-2)}) .$$

For all $n \in \mathbb{N}, n < |u|$, we define $\Psi_n = \varepsilon$. For all $n \in \mathbb{N}, n \geq |u|$, we define

$$\Psi_n = \prod_{i=1}^{|u|-1} (i, f^{(0)}) \prod_{i=|u|}^{n} \Phi_i .$$

Finally, let K be the language of words over A having

$$\zeta_\beta = x_1 u^{\beta-1} u \prod_{j=1}^{|u|-1} u_{|u|-j}^{(j)} \prod_{j=2}^{|u|} u_j^{(|u|+j-2)} u^{\alpha-\beta} x_2$$

for some $\beta \in [\alpha]$ as a subword but not $x_1 u^{\alpha+1} x_2$.

Claim 5.3.13. *The sequence $(\Psi_n)_{n\in\mathbb{N}}$ of A-programs is a program-reduction from L to K.*

Let

$$\begin{aligned} s \colon \mathbb{N} &\to \mathbb{N} \\ n &\mapsto \begin{cases} 0 & \text{if } n < |u| \\ |u| - 1 + (n - |u| + 1) \cdot (2|u| - 1) & \text{otherwise} . \end{cases} \end{aligned}$$

It is direct to see that $s(n) = |\Psi_n| \leq (2|u| - 1) \cdot n$ for all $n \in \mathbb{N}$.

Therefore, using this claim, $(\Psi_n)_{n\in\mathbb{N}}$ is a program-reduction from L to K of length

$s(n)$, so since K is piecewise testable and hence is recognised (classically) by some monoid from $\mathbf{J}$, Corollary 3.4.3 tells us that $L \in \mathcal{P}(\mathbf{J}, s(n)) = \mathcal{P}(\mathbf{J}, n)$.

Proof of claim. Let $n \in \mathbb{N}$. If $n < |u|$, then it is obvious that for all $w \in \Sigma^n$, $w \notin (x_1 \sqcup\!\sqcup \Sigma^*)u(x_2 \sqcup\!\sqcup \Sigma^*)$ so $w \notin L^{=n}$ and also $\Psi_n(w) = \varepsilon \notin K^{=s(n)}$, hence $L^{=n} = \emptyset = \Psi_n^{-1}(K^{=s(n)})$. Otherwise, $n \geq |u|$. We are going to show that $L^{=n} = \Psi_n^{-1}(K^{=s(n)})$.

Left-to-right inclusion. Let $w \in L^{=n}$. We want to show that $\Psi_n(w) \in K^{=s(n)}$.

We are first going to show that there exists some $\beta \in [\alpha]$ such that ζ_β is a subword of $\Psi_n(w)$. The fact that $w \in L^{=n}$ means in particular that $w \in (x_1 \sqcup\!\sqcup \Sigma^*)u(x_2 \sqcup\!\sqcup \Sigma^*)$ and we can hence decompose w as $w = y_1 z y_2$ where $y_1 \in (x_1 \sqcup\!\sqcup \Sigma^*)$, $y_2 \in (x_2 \sqcup\!\sqcup \Sigma^*)$, and $|y_1|$, $|y_2|$ are minimal. It follows necessarily that $z \in u^\alpha \sqcup\!\sqcup \Sigma^* \cap (u^{\alpha+1} \sqcup\!\sqcup \Sigma^*)^\complement \cap \Sigma^* u \Sigma^*$ by minimality of $|y_1|$ and $|y_2|$. By Lemma 5.3.11, we have $z = \big(\prod_{i=1}^{\alpha} \prod_{j=1}^{|u|} (v_{i,j} u_j)\big) y$ where $v_{i,j} \in (\Sigma \setminus \{u_j\})^*$ for all $i \in [\alpha]$, $j \in [|u|]$ and $y \in \bigcup_{i=1}^{|u|} \Big(\prod_{j=1}^{i-1} \big((\Sigma \setminus \{u_j\})^* u_j\big) (\Sigma \setminus \{u_i\})^* \Big)$. We know the letters in u are all distinct, so this means that there is no $\beta \in [\alpha - 1]$ such that u is a factor of z partly in $\prod_{j=1}^{|u|} (v_{\beta,j} u_j)$ and partly in $\prod_{j=1}^{|u|} (v_{\beta+1,j} u_j)$, and that u cannot appear as a factor of z partly in $\prod_{j=1}^{|u|} (v_{\alpha,j} u_j)$ and partly in y either. Hence, since $z \in \Sigma^* u \Sigma^*$, by the way we decomposed z, there necessarily exists $\beta \in [\alpha]$ such that $\prod_{j=1}^{|u|} (v_{\beta,j} u_j) \in \Sigma^* u \Sigma^*$. Let $\gamma, \delta \in [n]$ such that $w_\gamma \cdots w_\delta = \prod_{j=1}^{|u|} (v_{\beta,j} u_j)$, $w_1 \cdots w_{\gamma-1} = y_1 \big(\prod_{i=1}^{\beta-1} \prod_{j=1}^{|u|} (v_{i,j} u_j) \big)$ and $w_{\delta+1} \cdots w_n = \big(\prod_{i=\beta+1}^{\alpha} \prod_{j=1}^{|u|} (v_{i,j} u_j) \big) y y_2$. By the way β is defined, we have $w_{\delta-|u|+1} \cdots w_\delta = u$, because δ is the first and only position in w with the letter $u_{|u|}$ within the interval $[\![\lambda, \delta]\!]$ verifying that $w_\gamma \cdots w_{\delta-1}$ contains $u_1 \cdots u_{|u|-1}$ as a subword, and we observe additionally that $\delta \geq \gamma + |u| - 1 \geq |u|$. This means that

$$\begin{aligned} \Phi_\delta(w) &= f^{(0)}(w_\delta) f^{(1)}(w_{\delta-1}) \cdots f^{(|u|-1)}(w_{\delta-|u|+1}) f^{(|u|)}(w_{\delta-|u|+2}) \cdots f^{(2|u|-2)}(w_\delta) \\ &= u_{|u|} \prod_{j=1}^{|u|-1} u_{|u|-j}^{(j)} \prod_{j=2}^{|u|} u_j^{(|u|+j-2)} \ . \end{aligned}$$

Moreover,

$$\prod_{i=1}^{\gamma-1} f^{(0)}(w_i) = w_1 \cdots w_{\gamma-1} = y_1 \big(\prod_{i=1}^{\beta-1} \prod_{j=1}^{|u|} (v_{i,j} u_j) \big) \ ,$$

$$\prod_{i=\delta-|u|+1}^{\delta-1} f^{(0)}(w_i) = w_{\delta-|u|+1} \cdots w_{\delta-1} = u_1 \cdots u_{|u|-1}$$

and

$$\prod_{i=\delta+1}^{n} f^{(0)}(w_i) = w_{\delta+1}\cdots w_n = \big(\prod_{i=\beta+1}^{\alpha}\prod_{j=1}^{|u|}(v_{i,j}u_j)\big)yy_2 \ .$$

So as $\prod_{i=1}^{\gamma-1}(i,f^{(0)})\prod_{i=\delta-|u|+1}^{\delta-1}(i,f^{(0)})\Phi_\delta\prod_{i=\delta+1}^{n}(i,f^{(0)})$ is a subword of Ψ_n, we have that

$$\zeta_\beta = x_1u^{\beta-1}u\prod_{j=1}^{|u|-1}u_{|u|-j}^{(j)}\prod_{j=2}^{|u|}u_j^{(|u|+j-2)}u^{\alpha-\beta}x_2$$

is a subword of $\Psi_n(w)$.

We secondly show that $x_1u^{\alpha+1}x_2$ cannot be a subword of $\Psi_n(w)$. But this is direct by construction of Ψ_n, otherwise we would have that $x_1u^{\alpha+1}x_2$ is a subword of w, contradicting the fact that $w \in L^{=n}$.

Hence, $\Psi_n(w) \in K^{=s(n)}$, and since this is true for all $w \in L^{=n}$, we have $L^{=n} \subseteq \Psi_n^{-1}(K^{=s(n)})$.

Right-to-left inclusion. We are going to prove the "contrapositive inclusion".

Let $w \in \Sigma^n \setminus L^{=n}$. We want to show that $\Psi_n(w) \notin K^{=s(n)}$.

Let us start with the easy cases. If we have $w \notin (x_1u^\alpha x_2) \sqcup\!\sqcup \Sigma^*$, then it means that $x_1u^\alpha x_2$ is not a subword of w and hence, by construction of Ψ_n, not a subword of $\Psi(w)$ either, so that there does not exist any $\beta \in [\alpha]$ such that ζ_β is a subword of $\Psi_n(w)$. Similarly, if we have $w \in (x_1u^{\alpha+1}x_2) \sqcup\!\sqcup \Sigma^*$, then it means that $x_1u^{\alpha+1}x_2$ is a subword of w and hence, by construction of Ψ_n, a subword of $\Psi_n(w)$.

We now assume that $w \in (x_1u^\alpha x_2) \sqcup\!\sqcup \Sigma^* \cap \big((x_1u^{\alpha+1}x_2) \sqcup\!\sqcup \Sigma^*\big)^{\complement}$ while $w \notin (x_1 \sqcup\!\sqcup \Sigma^*)u(x_2 \sqcup\!\sqcup \Sigma^*)$. We want to show that in this case, there does not exist any $\beta \in [\alpha]$ such that ζ_β is a subword of $\Psi_n(w)$. Suppose for a contradiction that such a β exists; our goal is to show, through a careful observation of what this implies on the letters in w by examining how Ψ_n decorates the letters, that this contradictingly entails $x_1u^{\alpha+1}x_2$ is a subword of w.

Since ζ_β is a subword of $\Psi_n(w)$, it is not too difficult to see there exist

$$p_1,\ldots,p_{|x_1|+(\beta-1)\cdot|u|},q_1,\ldots,q_{3|u|-2},r_1,\ldots,r_{(\alpha-\beta)\cdot|u|+|x_2|} \in [n]$$

verifying that

$$w_{p_1}\cdots w_{p_{|x_1|+(\beta-1)\cdot|u|}} = x_1u^{\beta-1} \ ,$$

$$w_{q_1} \cdots w_{q_{3|u|-2}} = u \prod_{j=1}^{|u|-1} u_{|u|-j} \prod_{j=2}^{|u|} u_j \ ,$$

$$w_{r_1} \cdots w_{r_{(\alpha-\beta)\cdot|u|+|x_2|}} = u^{\alpha-\beta} x_2$$

and

$$(p_1, f^{(0)}) \cdots (p_{|x_1|+(\beta-1)\cdot|u|}, f^{(0)})(q_1, f^{(0)}) \cdots (q_{|u|}, f^{(0)})(q_{|u|+1}, f^{(1)}) \cdots (q_{2|u|-1}, f^{(|u|-1)})$$
$$(q_{2|u|}, f^{(|u|)}) \cdots (q_{3|u|-2}, f^{(2|u|-2)})(r_1, f^{(0)}) \cdots (r_{(\alpha-\beta)\cdot|u|+|x_2|}, f^{(0)})$$

is a subword of Ψ_n. By construction of Ψ_n, we have

$$p_1 < \cdots < p_{|x_1|+(\beta-1)\cdot|u|} < q_1 < \cdots < q_{|u|} < r_1 < \cdots < r_{(\alpha-\beta)\cdot|u|+|x_2|} \ ,$$

so this implies that w can be decomposed as $w = y_1 z y_2$ where $y_1 \in x_1 \sqcup\!\sqcup \Sigma^*$, $z \in u^\alpha \sqcup\!\sqcup \Sigma^*$ and $y_2 \in x_2 \sqcup\!\sqcup \Sigma^*$, the positions $p_1, \ldots, p_{|x_1|}$ corresponding to letters in y_1, the positions $p_{|x_1|+1}, \ldots, p_{|x_1|+(\beta-1)\cdot|u|}, q_1, \ldots, q_{|u|}, r_1, \ldots, r_{(\alpha-\beta)\cdot|u|}$ corresponding to letters in z and the positions $r_{(\alpha-\beta)\cdot|u|+1}, \ldots, r_{(\alpha-\beta)\cdot|u|+|x_2|}$ corresponding to letters in y_2.

We are now going to show that, in fact, $q_{|u|} < q_{2|u|-1} < q_{2|u|} < \cdots < q_{3|u|-2} < r_1$, which implies $z \in u^{\alpha+1} \sqcup\!\sqcup \Sigma^*$ and thus the contradiction we are aiming for. Since $w \notin (x_1 \sqcup\!\sqcup \Sigma^*)u(x_2 \sqcup\!\sqcup \Sigma^*)$, we have $z \notin \Sigma^* u \Sigma^*$, hence as $w_{q_{|u|}} = u_{|u|}$ and $|u| > 1$, there must exist $j \in [|u| - 1]$ such that $w_{q_{|u|}-j} \neq u_{|u|-j}$ and $w_{q_{|u|}-\iota} = u_{|u|-\iota}$ for all $\iota \in [\![0, j-1]\!]$. By construction of Ψ_n, we know that $q_{|u|+j} \geq q_{|u|} - j$ (because the instructions with $f^{(j)}$ after an instruction with $f^{(0)}$ querying position $p \in [n]$ all query a position at least equal to $p - j$), but since $u_{|u|-j} \neq w_{q_{|u|}-j}$ and $u_{|u|-j} \neq u_{|u|-\iota} = w_{q_{|u|}-\iota}$ for all $\iota \in [\![0, j-1]\!]$ as the letters in u are all distinct, we get that $q_{|u|+j} > q_{|u|}$. By (backward) induction, we can show that for all $\iota \in [\![j+1, |u|-1]\!]$, $q_{|u|+\iota} > q_{|u|}$. Indeed, given $\iota \in [\![j+1, |u|-1]\!]$, we have $q_{|u|+\iota-1} > q_{|u|}$, either by inductive hypothesis or directly in the base case $\iota = j+1$ by what we have just seen. So by construction of Ψ_n, we know that $q_{|u|+\iota} \geq q_{|u|}$ (because the instructions with $f^{(\iota)}$ after an instruction with $f^{(\iota-1)}$ querying position $p \in [n]$ all query a position at least equal to $p - 1$), but since $u_{|u|-\iota} \neq u_{|u|} = w_{q_{|u|}}$ as the letters in u are all distinct, it follows that $q_{|u|+\iota} > q_{|u|}$. Therefore, we have that $q_{2|u|-1} > q_{|u|}$. Moreover, by construction of Ψ_n, we also have $q_{2|u|-1} < q_{2|u|} < \cdots < q_{3|u|-2} < r_1$ (because for each $\iota \in [\![0, |u|-2]\!]$, the instructions with $f^{(|u|+\iota)}$ after an instruction with $f^{(|u|+\iota-1)}$ querying position $p \in [n]$ all query a position at least equal to $p + 1$). So, to conclude, we have $p_1 < \cdots < p_{|x_1|+(\beta-1)\cdot|u|} < q_1 < \cdots < q_{|u|} < q_{2|u|-1} < q_{2|u|} < \cdots < q_{3|u|-2} < r_1 < \cdots <$

$r_{(\alpha-\beta)\cdot|u|+|x_2|}$ and

$$\begin{aligned}&w_{p_1}\cdots w_{p_{|x_1|+(\beta-1)\cdot|u|}}w_{q_1}\cdots w_{q_{|u|}}w_{q_{2|u|-1}}w_{q_{2|u|}}\cdots w_{q_{3|u|-2}}w_{r_1}\cdots w_{r_{(\alpha-\beta)\cdot|u|+|x_2|}}\\ =&x_1u^{\beta-1}uu_1u_2\cdots u_{|u|}u^{\alpha-\beta}x_2 = x_1u^{\alpha+1}x_2\ .\end{aligned}$$

This implies that $w \in (x_1u^{\alpha+1}x_2) \sqcup\!\sqcup \Sigma^*$, a contradiction. So there does not exist $\beta \in [\alpha]$ such that ζ_β is a subword of $\Psi(w)$.

Therefore, in every case $\Psi_n(w) \notin K^{=s(n)}$, and since this is true for all $w \in \Sigma^n \setminus L^{=n}$, we have $\Sigma^n \setminus L^{=n} \subseteq A^{s(n)} \setminus \Psi_n^{-1}(K^{=s(n)})$, which is equivalent to $L^{=n} \supseteq \Psi_n^{-1}(K^{=s(n)})$.

This concludes the proof of the claim. □

And the one of the lemma. □

As we explained just before stating the previous lemma, we can now use it as the basic building block to straightforwardly prove the result we were aiming for.

Lemma 5.3.14. *Let Σ be an alphabet, $l \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$) such that for each $i \in [k]$, the letters in u_i are all distinct. For all $\alpha_1, \ldots, \alpha_k \in [l]$, we have*

$$R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)} \in \mathcal{P}(\mathbf{J})\ .$$

Proof. Let Σ be an alphabet, $l \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$) such that for each $i \in [k]$, the letters in u_i are all distinct. Let $\alpha_1, \ldots, \alpha_k \in [l]$.

For each $i \in [k]$ verifying $\alpha_i < l$, we define

$$\begin{aligned}L_i =&({u_1}^{\alpha_1}\cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^* \cap \big(({u_1}^{\alpha_1}\cdots {u_i}^{\alpha_i+1}\cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\big)^{\complement} \cap\\ &\big(({u_1}^{\alpha_1}\cdots {u_{i-1}}^{\alpha_{i-1}}) \sqcup\!\sqcup \Sigma^*\big)u_i\big(({u_{i+1}}^{\alpha_{i+1}}\cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\big)\ .\end{aligned}$$

It is immediate to show that

$$R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_k),\ldots,(u_k,\alpha_k)} = ({u_1}^{\alpha_1}\cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^* \cap \bigcap_{i\in[k],\alpha_i<l} L_i\ .$$

By Lemma 5.3.12, $L_i \in \mathcal{P}(\mathbf{J})$ for each $i \in [k]$ verifying $\alpha_i < l$. Moreover, since $({u_1}^{\alpha_1}\cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*$ obviously is a piecewise testable language, it belongs to $\mathcal{P}(\mathbf{J})$. Thus, we can conclude that $R^{(\Sigma,l)}_{(u_1,\alpha_1),\ldots,(u_k,\alpha_k)} \cap S^{(\Sigma,l)}_{(u_1,\alpha_k),\ldots,(u_k,\alpha_k)}$ belongs to $\mathcal{P}(\mathbf{J})$ by closure of $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg}$ under intersection, Proposition 3.4.2. □

As already stated, we now can conclude by Lemma 5.3.10 and closure of $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg}$ under Boolean operations, Proposition 3.4.2.

Corollary 5.3.15. *Let Σ be an alphabet, $l \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$) such that for each $i \in [k]$, the letters in u_i are all distinct. Then, $[u_1, \ldots, u_k]_{\Sigma,l} \in \mathcal{P}(\mathbf{J})$.*

However, what we really want to obtain, is, given an alphabet Σ, $l \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$) without putting any restriction on them, that $[u_1, \ldots, u_k]_{\Sigma,l}$ is in $\mathcal{P}(\mathbf{J})$. But, in fact, to remove the constraint that the letters must be all distinct in each of the factors u_i in the previous result, we simply have to decorate each of the input letters with its position minus 1 modulo a big enough positive integer d. This is what we do now.

Lemma 5.3.16. *Let Σ be an alphabet, $l \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$). Then $[u_1, \ldots, u_k]_{\Sigma,l} \in \mathcal{P}(\mathbf{J})$.*

Proof. Let Σ be an alphabet, $l \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$).

Let $d = \max_{i \in [k]} |u_i|$. If $d = 1$, then the result is straightforward because the language $[u_1, \ldots, u_k]_{\Sigma,l}$ then belongs to $\mathcal{L}(\mathbf{J})$, so now we assume $d \geq 2$. We let $\Sigma_d = \Sigma \times \mathbb{Z}/d\,\mathbb{Z}$ and for all $w \in \Sigma^*$, for all $i \in \mathbb{Z}/d\,\mathbb{Z}$, we define $\widetilde{w}^i = \prod_{j=1}^{|w|} (w_j, (j + i - 1) \mod d)$. We also let $\widetilde{w} = \widetilde{w}^0$ for all $w \in \Sigma^*$. Let us finally define $\psi \colon \Sigma^* \to \mathbb{Z}/d\,\mathbb{Z}$ as the cyclic stamp from $(\Sigma^*, \cdot)$ to $(\mathbb{Z}/d\,\mathbb{Z}, +)$ (the cyclic group over $\mathbb{Z}/d\,\mathbb{Z}$ with canonical addition modulo d) such that $\psi(a) = 1 \mod d$ for all $a \in \Sigma$. Observe that for all $w \in \Sigma^*$, $\widetilde{w} = \sigma_\psi(w)$.

For all $v \in \Sigma^+, |v| \leq d$, we define $\mu(v, 1) = v$ and

$$\mu(v, l) = \underbrace{v_1, \ldots, v_{|v|}, \ldots\ldots\ldots, v_1, \ldots, v_{|v|}}_{l \text{ times}} .$$

For all $v_1, \ldots, v_{k'} \in \Sigma^+$ ($k' \in \mathbb{N}_{>0}$) such that $|v_i| \leq d$ for each $i \in [k']$, we let

$$[v_1, \ldots, v_{k'}]_{\Sigma,l,d} = \bigcup_{i_1, \ldots, i_{k'} \in \mathbb{Z}/d\,\mathbb{Z}} [\widetilde{v_1}^{i_1}, \ldots, \widetilde{v_{k'}}^{i_{k'}}]_{\Sigma_d,l} ,$$

that does belong to $\mathcal{P}(\mathbf{J})$ by Corollary 5.3.15 and closure of $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg}$ under finite union (Proposition 3.4.2), because since $|v_i| \leq d$ for each $i \in [k']$, each $\widetilde{v_i}^j$ for $j \in \mathbb{Z}/d\,\mathbb{Z}$ has all distinct letters.

This implies that for all $q_1, \ldots, q_k \in \{1, l\}$, we have that $[\mu(u_1, q_1), \ldots, \mu(u_k, q_k)]_{\Sigma,l,d}$

does belong to $\mathcal{P}(\mathbf{J})$, so that

$$\bigcup_{q_1,\dots,q_k\in\{1,l\}} [\mu(u_1,q_1),\dots,\mu(u_k,q_k)]_{\Sigma,l,d}$$

is a language over Σ_d belonging to $\mathcal{P}(\mathbf{J})$.

To conclude, it is not so difficult to see that

$$\begin{aligned}[u_1,\dots,u_k]_{\Sigma,l} &= \Big\{ w\in\Sigma^* \;\Big|\; \widetilde{w}^d \in \bigcup_{q_1,\dots,q_k\in\{1,l\}} [\mu(u_1,q_1),\dots,\mu(u_k,q_k)]_{\Sigma,l,d} \Big\} \\ &= \sigma_\psi^{-1}\Big(\bigcup_{q_1,\dots,q_k\in\{1,l\}} [\mu(u_1,q_1),\dots,\mu(u_k,q_k)]_{\Sigma,l,d} \Big) \ ,\end{aligned}$$

so that $[u_1,\dots,u_k]_{\Sigma,l}$ does also belong to $\mathcal{P}(\mathbf{J})$ by Lemma 3.4.4. □

We can eventually straightforwardly extend the result to $[u_1,\dots,u_k[_{\Sigma,l}$, $]u_1,\dots,u_k]_{\Sigma,l}$ and $]u_1,\dots,u_k[_{\Sigma,l}$.

Proposition 5.3.17. *Let Σ be an alphabet, $l\in\mathbb{N}_{>0}$ and $u_1,\dots,u_k\in\Sigma^+$ ($k\in\mathbb{N}_{>0}$). Then, $[u_1,\dots,u_k]_{\Sigma,l}$, $[u_1,\dots,u_k[_{\Sigma,l}$, $]u_1,\dots,u_k]_{\Sigma,l}$ and $]u_1,\dots,u_k[_{\Sigma,l}$ do all belong to $\mathcal{P}(\mathbf{J})$.*

Proof. Let Σ be an alphabet, $l\in\mathbb{N}_{>0}$ and $u_1,\dots,u_k\in\Sigma^+$ ($k\in\mathbb{N}_{>0}$).

Then, Lemma 5.3.16 tells us that $[u_1,\dots,u_k]_{\Sigma,l}\in\mathcal{P}(\mathbf{J})$. Moreover, we have

$$[u_1,\dots,u_k[_{\Sigma,l} = [u_1,\dots,u_k]_{\Sigma,l}\cap\Sigma^*u_k \ ,$$

$$]u_1,\dots,u_k]_{\Sigma,l} = [u_1,\dots,u_k]_{\Sigma,l}\cap u_1\Sigma^*$$

and

$$]u_1,\dots,u_k[_{\Sigma,l} = [u_1,\dots,u_k]_{\Sigma,l}\cap\Sigma^*u_k\cap u_1\Sigma^* \ ;$$

hence they all belong to $\mathcal{P}(\mathbf{J})$ since Σ^*u_k as well as $u_1\Sigma^*$ do (see the discussion right at the beginning of Subsection 5.3.1), and $\mathcal{P}(\mathbf{J})\cap\mathcal{R}\mathsf{eg}$ is closed under intersection, Proposition 3.4.2. □

This finishes to prove Theorem 5.3.6 by closure of $\mathcal{P}(\mathbf{J})\cap\mathcal{R}\mathsf{eg}$ under Boolean combinations (Proposition 3.4.2).

5.4 Dot-depth one strongly unambiguous monomials

As explained in the previous section, we believe that the class of all threshold dot-depth one languages, proven to be contained in $\mathcal{P}(\mathbf{J})$, is in fact exactly the *ne*-variety of languages $\mathcal{L}(\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}})$, easily seen to be equal to $\mathcal{L}(\mathbf{J} * \mathbf{D}) \cap \mathcal{L}(\mathbf{DA})$, which would imply Conjecture 1 to be true.

One direction of the conjectured classes equality, that any threshold dot-depth one language is in $\mathcal{L}(\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}})$, is quite straightforward but a bit cumbersome to prove. We now state and prove this result.

Proposition 5.4.1. *Any threshold dot-depth one language belongs to* $\mathcal{L}(\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}})$.

Proof. To prove the proposition, by closure under Boolean operations of both $\mathcal{L}(\mathbf{J} * \mathbf{D})$ and $\mathcal{L}(\mathbf{DA})$, it suffices to prove that for any Σ an alphabet, $l \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$), the languages $[u_1, \ldots, u_k]_{\Sigma,l}$, $[u_1, \ldots, u_k[_{\Sigma,l}$, $]u_1, \ldots, u_k]_{\Sigma,l}$ and $]u_1, \ldots, u_k[_{\Sigma,l}$ do all belong to $\mathcal{L}(\mathbf{J} * \mathbf{D}) \cap \mathcal{L}(\mathbf{DA})$. This is what we show in the following.

Let Σ be an alphabet, $l \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$). We first show that $[u_1, \ldots, u_k]_{\Sigma,l} \in \mathcal{L}(\mathbf{J} * \mathbf{D}) \cap \mathcal{L}(\mathbf{DA})$, and consequently we will be able to directly handle the cases of $[u_1, \ldots, u_k[_{\Sigma,l}$, $]u_1, \ldots, u_k]_{\Sigma,l}$ and $]u_1, \ldots, u_k[_{\Sigma,l}$.

Membership in $\mathcal{L}(\mathbf{J} * \mathbf{D})$. As given by Definition 5.3.7, we have that

$$[u_1, \ldots, u_k]_{\Sigma,l} = \bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(l)}_{(u_1,q_1)} \cdots L^{(l)}_{(u_k,q_k)} \ ,$$

where for all $q_1, \ldots, q_k \in \{1, l\}$, $L^{(l)}_{(u_1,q_1)} \cdots L^{(l)}_{(u_k,q_k)}$ is easily seen to be a dot-depth one language. Hence, by closure of $\mathcal{L}(\mathbf{J} * \mathbf{D})$ under finite union, $[u_1, \ldots, u_k]_{\Sigma,l} \in \mathcal{L}(\mathbf{J} * \mathbf{D})$.

Membership in $\mathcal{L}(\mathbf{DA})$. Let now $L = [u_1, \ldots, u_k]_{\Sigma,l}$, let $\sim$ be its syntactic congruence and let ω be the idempotent power of its syntactic monoid $(M, *)$. Using the equational characterisation of **DA**, we are now going to prove that $(M, *) \in \mathbf{DA}$: that is, we are going to prove that $(m * n)^{*,\omega} = (m * n)^{*,\omega} * m * (m * n)^{*,\omega}$ for all $m, n \in M$, so that $(M, *)$ does belong to **DA** and thus $[u_1, \ldots, u_k]_{\Sigma,l}$ to $\mathcal{L}(\mathbf{DA})$. To show that each pair of elements of M verifies the previous equation, by definition of the syntactic monoid of L, it suffices to show that $(uv)^\omega \sim (uv)^\omega u (uv)^\omega$ for all $u, v \in \Sigma^*$.

Let $u, v \in \Sigma^*$. Our aim is to show that $(uv)^\omega \sim (uv)^\omega u (uv)^\omega$. By definition of the syntactic monoid of L and of ω, it is not too difficult to see that this is equivalent to

showing that $(uv)^{\omega'} \sim (uv)^{\omega'}u(uv)^{\omega'}$ where $\omega' \in \mathbb{N}_{>0}$ is the smallest multiple of ω not smaller than $\sum_{i=1}^{k} l \cdot |u_i|$ (why we need ω' to be as big will become clear later on).

When both u and v are equal to the empty word, we trivially have that $(uv)^{\omega'} \sim (uv)^{\omega'}u(uv)^{\omega'}$. So we now assume that at least one of u and v is not equal to the empty word.

Let $x, y \in \Sigma^*$ such that $w = x(uv)^{\omega'}y \in L$ and consider $w' = x(uv)^{\omega'}u(uv)^{\omega'}y$. Let's now prove that w' does also belong to L. When x or y belongs to L, then it is obvious that w' does also belong to it. We now assume that it is not the case. Let $i_1 \in [k]$ be the smallest integer in $[k]$ such that x does not belong to $[u_1, \dots, u_{i_1}]_{\Sigma,l}$ and $i_2 \in [k]$ the biggest integer in $[k]$ such that y does not belong to $[u_{i_2}, \dots, u_k]_{\Sigma,l}$, that do exist by the hypothesis we just made. Let $\kappa_1 \in [\![0, |x|]\!]$ be the smallest integer in $[|x|]$ such that $x_1 \cdots x_{\kappa_1} \in [u_1, \dots, u_{i_1-1}]_{\Sigma,l}$ when $i_1 > 1$ and 0 otherwise; let symmetrically $\kappa_2 \in [\![1, |y|+1]\!]$ be the biggest integer in $[|y|]$ such that $y_{\kappa_2} \cdots y_{|y|} \in [u_{i_2+1}, \dots, u_k]_{\Sigma,l}$ when $i_2 < k$ and $|y|+1$ otherwise. The idea to prove $w' \in L$ is to distinguish between three cases when $i_1 \leq i_2$, otherwise it is direct. When both the prefix $x(uv)^{l\cdot|u_{i_1}|}$ of w and w' belongs to $[u_1, \dots, u_{i_1}]_{\Sigma,l}$ and the suffix $(uv)^{l\cdot|u_{i_2}|}y$ of w and w' belongs to $[u_{i_2}, \dots, u_k]_{\Sigma,l}$, then we can conclude by using the fact that all the letters of the words u_{i_1+1} to u_{i_2-1} are to be found in the remaining factor in the middle of w, made solely of powers of uv. Otherwise, the prefix $x(uv)^{l\cdot|u_{i_1}|}$ of w and w' does not belong to $[u_1, \dots, u_{i_1}]_{\Sigma,l}$ or the suffix $(uv)^{l\cdot|u_{i_2}|}y$ of w and w' does not belong to $[u_{i_2}, \dots, u_k]_{\Sigma,l}$. When the first possibility is true, we can show that we necessarily have that the prefix $x(uv)^{\omega'}$ of w and w' as a whole does not belong to $[u_1, \dots, u_{i_1}]_{\Sigma,l}$ and then conclude after analysing how w does consequently decompose into one prefix in $[u_1, \dots, u_{i_1-1}]_{\Sigma,l}$, one middle factor in $[u_{i_1}]_{\Sigma,l}$ and one suffix in $[u_{i_1+1}, \dots, u_k]_{\Sigma,l}$, using κ_1 and κ_2. We proceed by symmetry when the second possibility is true. We now move on to the details.

If $i_1 > i_2$, then we have that x belongs to $[u_1, \dots, u_{i_1-1}]_{\Sigma,l}$ (which is well defined as $i_1 > i_2 \geq 1$) and that y belongs belongs to $[u_{i_1}, \dots, u_k]_{\Sigma,l}$ (which is also well defined as $k \geq i_1$), so that w' obviously belongs to L. Otherwise, $i_1 \leq i_2$. We first observe that if $x(uv)^{\omega'}$ belongs to $[u_1, \dots, u_{i_1}]_{\Sigma,l}$, then $x(uv)^{l\cdot|u_{i_1}|}$ does also belong to it. Indeed, assume the hypothesis of the implication is true; there are two possible cases. Either all letters of u_{i_1} appear in uv: in that case we have $(uv)^{l\cdot|u_{i_1}|} \in {u_{i_1}}^l \sqcup\!\sqcup \Sigma^* \subseteq [u_{i_1}]_{\Sigma,l}$ and hence $x(uv)^{l\cdot|u_{i_1}|} \in [u_1, \dots, u_{i_1}]_{\Sigma,l}$. Or there is at least one letter in u_{i_1} not appearing in uv: since $x \notin [u_1, \dots, u_{i_1}]_{\Sigma,l}$, either

- u_{i_1} is a factor of $x(uv)^{\omega'}$ whose first letter is in $x_{\kappa_1+1} \cdots x_{|x|}$ and whose last letter is in $(uv)^{\omega'}$, so that because $|uv| \geq 1$, we have $x_{\kappa_1+1} \cdots x_{|x|}(uv)^{l\cdot|u_{i_1}|} \in \Sigma^* u_{i_1} \Sigma^* \subseteq [u_{i_1}]_{\Sigma,l}$

and hence $x(uv)^{l\cdot|u_{i_1}|} \in [u_1, \ldots, u_{i_1}]_{\Sigma,l}$;

- or ${u_{i_1}}^l$ is a subword of $x_{\kappa_1+1}\cdots x_{|x|}(uv)^{\omega'}$ such that only its at most $|u_{i_1}|-1$ last letters appear in the factor $(uv)^{\omega'}$, so that we have $x_{\kappa_1+1}\cdots x_{|x|}(uv)^{l\cdot|u_{i_1}|} \in {u_{i_1}}^l \sqcup\!\sqcup \Sigma^* \subseteq [u_{i_1}]_{\Sigma,l}$ and hence $x(uv)^{l\cdot|u_{i_1}|} \in [u_1, \ldots, u_{i_1}]_{\Sigma,l}$.

Symmetrically, we can prove that if $(uv)^{\omega'}y$ belongs to $[u_{i_2}, \ldots, u_k]_{\Sigma,l}$, then $(uv)^{l\cdot|u_{i_2}|}y$ does also belong to it. We now distinguish between three different cases.

- $x(uv)^{l\cdot|u_{i_1}|}$ does not belong to $[u_1, \ldots, u_{i_1}]_{\Sigma,l}$. By what we have shown just above, this means that $x(uv)^{\omega'}$ does not belong to $[u_1, \ldots, u_{i_1}]_{\Sigma,l}$ either. Since $w \in L$, this necessarily means that $\kappa_2 > 1$ and that there exists $\kappa_2' \in [\![2, \kappa_2]\!]$ verifying $x_1\cdots x_{\kappa_1} \in [u_1, \ldots, u_{i_1-1}]_{\Sigma,l}$, $x_{\kappa_1+1}\cdots x_{|x|}(uv)^{\omega'}y_1\cdots y_{\kappa_2'-1} \in [u_{i_1}]_{\Sigma,l}$ and $y_{\kappa_2'}\cdots y_{|y|} \in [u_{i_1+1}, \ldots, u_k]_{\Sigma,l}$, implying $i_1 = i_2$ and that κ_2' can be taken equal to κ_2 by the fact that $y \notin [u_{i_2}, \ldots, u_k]_{\Sigma,l}$ and $y_{\kappa_2}\cdots y_{|y|} \in [u_{i_2+1}, \ldots, u_k]_{\Sigma,l}$. If $(uv)^{\omega'}y$ belongs to $[u_{i_1}, \ldots, u_k]_{\Sigma,l}$, then as $x \in [u_1, \ldots, u_{i_1-1}]_{\Sigma,l}$, we have that $w' = x(uv)^{\omega'}u(uv)^{\omega'}y \in L$. Otherwise, since u_{i_1} contains at least one letter not appearing in uv and $|uv| \geq 1$, ${u_{i_1}}^l$ must be a subword of $x_{\kappa_1+1}\cdots x_{|x|}(uv)^{\omega'}y_1\cdots y_{\kappa_2'-1} \in [u_{i_1}]_{\Sigma,l}$ with at most $|u_{i_1}|-1$ of its letters appearing in the factor $(uv)^{\omega'}$, so that $x_{\kappa_1+1}\cdots x_{|x|}(uv)^{\omega'}u(uv)^{\omega'}y_1\cdots y_{\kappa_2'-1} \in {u_{i_1}}^l \sqcup\!\sqcup \Sigma^* \subseteq [u_{i_1}]_{\Sigma,l}$, also showing $w' = x(uv)^{\omega'}u(uv)^{\omega'}y \in L$.

- $(uv)^{l\cdot|u_{i_2}|}y$ does not belong to $[u_{i_2}, \ldots, u_k]_{\Sigma,l}$. Symmetrically to the previous case, we can show that $w' \in L$.

- $x(uv)^{l\cdot|u_{i_1}|}$ belongs to $[u_1, \ldots, u_{i_1}]_{\Sigma,l}$ and $(uv)^{l\cdot|u_{i_2}|}y$ belongs to $[u_{i_2}, \ldots, u_k]_{\Sigma,l}$. In this case, for all $i \in [\![i_1+1, i_2-1]\!]$, we have that $\mathrm{alph}(u_i) \subseteq \mathrm{alph}(uv)$. Indeed, assume there would exist $i \in [\![i_1+1, i_2-1]\!]$ such that $\mathrm{alph}(u_i) \nsubseteq \mathrm{alph}(uv)$; this would mean that at least one letter of u_i does appear in x or y but not in uv. So this would imply that either there exists $\kappa \in [|x|]$ such that $x_\kappa \ldots x_{|x|}(uv)^{\omega'}y \in [u_i, \ldots, u_k]_{\Sigma,l}$ but $x_{\kappa+1} \ldots x_{|x|}(uv)^{\omega'}y \notin [u_i, \ldots, u_k]_{\Sigma,l}$ so that necessarily $x_1\cdots x_{\kappa-1}$ and hence x do belong to $[u_1, \ldots, u_{i-1}]_{\Sigma,l}$, contradicting the fact that $x \notin [u_1, \ldots, u_{i_1}]_{\Sigma,l}$ since $i-1 \geq i_1$; or that $(uv)^{\omega'}y \in [u_i, \ldots, u_k]_{\Sigma,l}$, which would imply that y does belong to $[u_{i+1}, \ldots, u_k]_{\Sigma,l}$, contradicting the fact that $y \notin [u_{i_2}, \ldots, u_k]_{\Sigma,l}$ since $i+1 \leq i_2$. Hence, we have that $(uv)^{\omega'-l\cdot|u_{i_1}|}u(uv)^{\omega'-l\cdot|u_{i_2}|}$, containing $(uv)^{\sum_{i=i_1+1}^{i_2-1} l\cdot|u_i|}$ as a subword, belongs to ${u_{i_1+1}}^l\cdots{u_{i_2-1}}^l \sqcup\!\sqcup \Sigma^* \subseteq [u_{i_1+1}, \ldots, u_{i_2-1}]_{\Sigma,l}$. Thus, putting all together, we get that $w' = x(uv)^{\omega'}u(uv)^{\omega'}y \in L$.

Therefore, in any case we have $x(uv)^{\omega'}u(uv)^{\omega'}y \in L$.

Let $x, y \in \Sigma^*$ such that $x(uv)^{\omega'}u(uv)^{\omega'}y \in L$. In a way similar to above, we can show that then, $x(uv)^{\omega'}y \in L$.

This shows that $(uv)^\omega \sim (uv)^\omega u(uv)^\omega$ and as it is true for all $u, v \in \Sigma^*$, we eventually get that $[u_1, \ldots, u_k]_{\Sigma,l} \in \mathcal{L}(\mathbf{DA})$.

Now that we have $[u_1, \ldots, u_k]_{\Sigma,l} \in \mathcal{L}(\mathbf{J} * \mathbf{D}) \cap \mathcal{L}(\mathbf{DA})$, since both $u_1\Sigma^*$ and $\Sigma^* u_k$ obviously belong to $\mathcal{L}(\mathbf{J} * \mathbf{D}) \cap \mathcal{L}(\mathbf{DA})$, by closure under Boolean operations of both $\mathcal{L}(\mathbf{J} * \mathbf{D})$ and $\mathcal{L}(\mathbf{DA})$, we have that $[u_1, \ldots, u_k[_{\Sigma,l} = [u_1, \ldots, u_k]_{\Sigma,l} \cap \Sigma^* u_k$, $]u_1, \ldots, u_k]_{\Sigma,l} = [u_1, \ldots, u_k]_{\Sigma,l} \cap u_1\Sigma^*$ and $]u_1, \ldots, u_k[_{\Sigma,l} = [u_1, \ldots, u_k]_{\Sigma,l} \cap \Sigma^* u_k \cap u_1\Sigma^*$ do all also belong to $\mathcal{L}(\mathbf{J} * \mathbf{D}) \cap \mathcal{L}(\mathbf{DA})$.

This concludes the proof of the proposition. □

That being done, how do we prove, conversely, that any language in $\mathcal{L}(\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}})$ is a threshold dot-depth one language? We do not know, or, more precisely, we only know how to prove this in a particular case and we do not know whether the general case is true. The most fruitful path we followed on the quest for such a proof is to try to understand how unambiguous polynomials, languages from $\mathcal{L}(\mathbf{DA})$, look like when they are restricted to also be dot-depth one languages, languages from $\mathcal{L}(\mathbf{J} * \mathbf{D})$, and prove that they actually are threshold dot-depth one languages. What we managed to prove is only that a strongly unambiguous monomial that is also a language of dot-depth one is a threshold dot-depth one language.

Proposition 5.4.2. *Any strongly unambiguous monomial over some alphabet Σ that is also a dot-depth one language is in fact a threshold dot-depth one language.*

We dedicate the remainder (and almost all) of this section to prove this proposition — the main, partial, result of this section. Let us first just give some short remarks about how we could fully prove the *ne*-variety of languages $\mathcal{L}(\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}})$ to be contained in the class of threshold dot-depth one languages and, as a consequence, Conjecture 1.

Given Proposition 5.4.2, it would "suffice" to prove that, in fact, any language in $\mathcal{L}(\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}})$, i.e. any unambiguous polynomial that is also a language of dot-depth one, can be expressed as a Boolean combination of SUMs that each also are dot-depth one languages. However, we are not sure this is true and have no idea about how complicated it would be to prove it if it were the case. That being said, we must admit that the upcoming proof of Proposition 5.4.2 has a complexity that is much too high for the result it proves and that there must be better ways to prove any language in $\mathcal{L}(\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}})$ to be a threshold dot-depth one language; one of these ways could be adapting the technique

based on graph congruences used by Knast to characterise algebraically dot-depth one languages Knast [1983], later reformulated in terms of finite categories by Tilson Tilson [1987].

5.4.1 Strongly unambiguous monomial trees

Definitions and properties

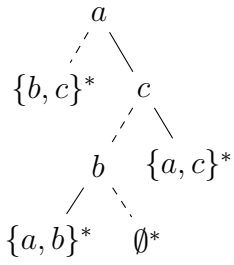
The very rough idea of our proof of Proposition 5.4.2 is to do an induction on a strongly unambiguous monomial (SUM) that is also a language of dot-depth one so as to inductively build a threshold dot-depth one language equal to this SUM. To do this, though, we need to exploit some structure of SUMs.

The definition of a SUM directly gives a way to decompose it as a tree that is of the type given by the definition below, not necessarily in a unique way.

Definition 5.4.3. Given an alphabet Σ, a *strongly unambiguous monomial tree over* Σ (SUMT) is a rooted binary tree whose internal nodes are labelled by letters from Σ and whose leaves are labelled by Kleene closures of alphabets subsets of Σ with two types of edges, full and dashed edges, each internal node being linked to exactly one child with a full edge and exactly one child with a dashed edge, such that the letter of this internal node never appears in the subtree corresponding to this last child as a label of an internal node or an element of a starred alphabet labelling a leaf.

To each such SUMT T we associate a unique SUM $\mathcal{L}(T)$ over Σ obtained by concatenating the labels of it in the order given by an infix depth-first search. We say that $\mathcal{L}(T)$ can be decomposed as T, which is a decomposition of $\mathcal{L}(T)$.

Example 5.4.4. The following SUMT over $\{a, b, c\}$



is a decomposition of the SUM $(b + c)^*a(a + b)^*bc(a + c)^*$ over $\{a, b, c\}$.

Observe that a SUMT that is a decomposition of a SUM can be arbitrary, which is not very practical when it comes to manipulating it in our proof by induction of Propos-

ition 5.4.2. Fortunately, any SUM can always be decomposed as a SUMT that is in a certain normal form, defined below.

Definition 5.4.5. A SUMT over an alphabet Σ is a *straight right normalised SUMT* (respectively a *straight left normalised SUMT*) over Σ if and only if either:

- it consists only of a root;
- its right (respectively left) branch has only full edges and all subtrees linked to this branch are left (respectively right) normalised SUMTs over Σ.

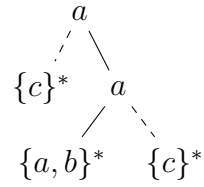
We will say a SUMT over Σ is a *bent right normalised SUMT* (respectively a *bent left normalised SUMT*) over Σ if and only if it is made of a right (respectively left) normalised SUMT that is not only a root and such that its rightmost (respectively leftmost) leaf has been replaced by a left (respectively right) normalised SUMT that is not only a root.

We shall abbreviate normalised SUMT with NSUMT and call such any straight or bent, right or left normalised SUMT.

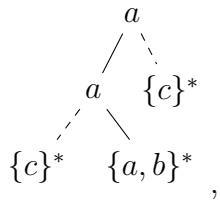
To put it in a (informal) nutshell, a straight NSUMT is a SUMT in which the only path from the root to a leaf using only full edges always goes to the same side (that is to say, it always goes to the right child or always goes to the left child), while a bent NSUMT is a SUMT in which the only such path changes sides only once.

Example 5.4.6. Example 5.4.4 gives an example of a straight right NSUMT over $\{a, b, c\}$.

The following SUMT over $\{a, b, c\}$



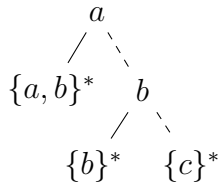
is a bent right NSUMT over $\{a, b, c\}$. It is a decomposition of the SUM $c^*a(a + b)^*ac^*$ over $\{a, b, c\}$ and the only other SUMT over $\{a, b, c\}$ decomposing it is this one



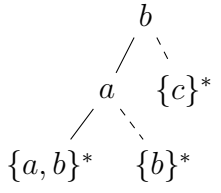
,

a bent left NSUMT over $\{a, b, c\}$.

Finally, the following SUMT over $\{a, b, c\}$



is a decomposition of the SUM $(a+b)^*ab^*bc^*$ over $\{a, b, c\}$ that is not normalised. However, it can also be decomposed as the following straight left NSUMT over $\{a, b, c\}$



.

We now state and prove the following lemma that basically tells that any SUM can always be decomposed as a normalised SUMT.

Lemma 5.4.7. *Let L be a SUM over an alphabet Σ. Then L can be decomposed as an NSUMT over Σ. Additionally, when L can be decomposed as a SUMT over Σ whose right or left branch contains only full edges, then it can also be decomposed, respectively, as a straight right or left NSUMT over Σ.*

Proof. Let us fix an alphabet Σ.

We use the following observation: given a SUM $A_0^*a_1A_1^*a_2 \cdots A_{k-1}^*a_kA_k^*$ with $k \in \mathbb{N}$, any language $A_i^*a_{i+1} \cdots A_{j-1}^*a_jA_j^*$ for $i, j \in [\![0, k]\!], i \leq j$, that is equal to A_i^* when $i = j$, is a SUM.

A root of a SUM $A_0^*a_1A_1^*a_2 \cdots A_{k-1}^*a_kA_k^*$ with $k \in \mathbb{N}_{>0}$ is any letter a_i which does not occur to the left (a *right-root*) or to the right (a *left-root*). A non-zero-degree SUM must have at least one root, may have several roots. A root may be left or right or both.

If a non-zero-degree SUM contains a right-root, we call it a *right-SUM*. Define *left-SUM* similarly. A given SUM may be right or left or both.

The following claim proves the lemma.

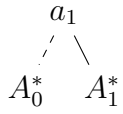
Claim 5.4.8. *Let L be a SUM over Σ of degree at least 1.*

- *If L is a right-SUM, it can be decomposed as a straight or bent right NSUMT. If additionally L is not a left-SUM or can be decomposed as a SUMT whose right branch contains only full edges, it can be decomposed as a straight right NSUMT.*

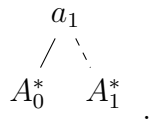
- *If L is a left-SUM, it can be decomposed as a straight or bent left NSUMT. If additionally L is not a right-SUM or can be decomposed as a SUMT whose left branch contains only full edges, it can be decomposed as a straight left NSUMT.*

Proof of the claim. We prove the claim by induction on the degree k of the SUMs.

Base case $k = 1$. Let $A_0^* a_1 A_1^*$ be a SUM over Σ of degree 1. It is straightforward to see that if it is a right-SUM, then it can be decomposed as the straight right NSUMT



and if it is a left-SUM, then it can be decomposed as the straight left NSUMT



.

Induction. Let $k \in \mathbb{N}, k \geq 2$ and assume the claim is true for all SUMs over Σ of degree less than k.

Let $L = A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ be a SUM over Σ of degree k.

Assume L is a right-SUM. Let $i \in [k]$ be the smallest integer in $[k]$ such that a_i is a right-root of L (a_i is the leftmost right-root of L). Then L can be written as $L = L_0 a_i L_1$ where L_0 and L_1 are SUMs over Σ and L_0 is such that $L_0 \cap \Sigma^* a_i \Sigma^* = \emptyset$. We now aim to apply the inductive hypothesis to get an NSUMT Δ_0 that is a decomposition of L_0, as well as an NSUMT Δ_1 that is a decomposition of L_1 and is necessary a straight right NSUMT when L is not a left-SUM or can be decomposed as a SUMT whose right branch contains only full edges. We can then combine them into a desired NSUMT T that is a decomposition of L.

Note first that when L is not a left-SUM and $i < k$, then L_1 is not a left-SUM either by the simple fact that any left-root of L_1 would also be a left-root of L. Moreover, when L can be decomposed as a SUMT whose right branch contains only full edges and $i < k$, then L_1 can also be decomposed as a SUMT whose right branch contains only full edges. Indeed, by hypothesis, we have $L = K_0 a_{\delta_1} \cdots K_{l-1} a_{\delta_l} A_k^*$ where $l \in [k]$, $K_0, \ldots, K_{l-1}$ are SUMs over Σ such that $K_{j-1} \cap \Sigma^* a_{\delta_j} \Sigma^* = \emptyset$ for all $j \in [l]$ and $\delta_l = k$. Denote by $j \in [\![0, l-1]\!]$ the unique integer in $[\![0, l-1]\!]$ such that $\delta_j \leq i < \delta_{j+1}$, using the convention that $\delta_0 = 0$, and set $K'_j = A_i^* a_{i+1} A_{i+1}^* \cdots A_{\delta_{j+1}-2}^* a_{\delta_{j+1}-1} A_{\delta_{j+1}-1}^*$. We have that

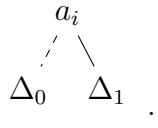
$L_1 = K'_j a_{\delta_{j+1}} K_{j+1} \cdots K_{l-1} a_{\delta_l} A_k^*$ where K'_j is a SUM over Σ such that $K'_j \cap \Sigma^* a_{\delta_{j+1}} \Sigma^* = \emptyset$, so that L_1 can indeed also be decomposed as a SUMT whose right branch contains only full edges.

Now, if $i = 1$, we have $L_0 = A_0^*$ and let $\Delta_0 = A_0^*$. Otherwise, we know that L_0 is a SUM over Σ of degree at least 1 and less than k, that cannot be a right-SUM by minimality of i and hence also necessarily has the property that it is a left-SUM: in that case we therefore apply the inductive hypothesis to let Δ_0 be a straight left NSUMT that is a decomposition of L_0. As a consequence, in any case, Δ_0 is a straight left NSUMT that is a decomposition of L_0.

For L_1, there are three cases to consider.

- If $i = k$, we have $L_1 = A_k^*$ and we let $\Delta_1 = A_k^*$, a straight right NSUMT.
- Otherwise, we know that L_1 is a SUM over Σ of degree at least 1 and less than k. Assume L_1 is a right-SUM. Then, by inductive hypothesis, we let Δ_1 be a straight or bent right NSUMT that is a decomposition of L_1. If, in addition, we have that L is not a left-SUM or can be decomposed as a SUMT whose right branch contains only full edges, we know by what we proved just above that L_1 is as such, so that by inductive hypothesis, we let Δ_1 be a straight right NSUMT.
- Otherwise, L_1 is a SUM over Σ of degree at least 1 and less than k that is not a right-SUM. Hence, it is necessarily a left-SUM: in that case we therefore apply the inductive hypothesis to let Δ_1 be a straight left NSUMT that is a decomposition of L_1.

Let now

$T =$ 

.

It is straightforward to see T is a SUMT over Σ that is a decomposition of L. Moreover, since Δ_0 is a straight left NSUMT, when Δ_1 is a straight or bent right NSUMT given by one of the two first cases just above, T is a straight or bent right NSUMT as well, and when Δ_1 is a straight left NSUMT given by the last case just above, T is a bent right NSUMT. If additionally L is not a left-SUM or can be decomposed as a SUMT whose right branch contains only full edges, we know by what we showed before that L_1 is as such when $i < k$, so that it must be a right-SUM. This in turn means Δ_1 is necessarily a straight right NSUMT given by one of the two first cases just above, so we can conclude T is a straight right NSUMT, as wished.

Assume now L is a left-SUM. In a symmetric way, we can show that it can be decomposed as a straight or bent left NSUMT and that, if additionally L is not a right-SUM or can be decomposed as a SUMT whose left branch contains only full edges, it can be decomposed as a straight left NSUMT.

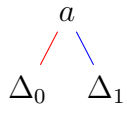
This concludes the proof of the claim. □

And the one of the lemma. □

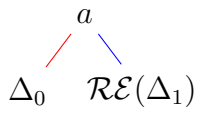
One thing will shall quite often do on SUMTs when manipulating them in the proof of Proposition 5.4.2 is to remove a certain number of their rightmost or leftmost internal nodes. We now define formally this operation of removing the rightmost or leftmost internal node, called respectively right or left elimination.

Definition 5.4.9. Let T be a SUMT over an alphabet Σ. We define the *right elimination operation on* T inductively on T as the operation associating $\mathcal{RE}(T)$ to T in the following way:

- if $T = A^*$, then $\mathcal{RE}(T) = A^*$;
- if

$T =$ 

  where Δ_0 and Δ_1 are SUMTs over Σ and the red edge is either full or dashed while the blue one is of the opposite type, then $\mathcal{RE}(T) = \Delta_0$ when Δ_1 consists only of a root and

$\mathcal{RE}(T) =$ 

  otherwise.

We define *left elimination on* T, denoted by $\mathcal{LE}(T)$, in a symmetric way.

When we apply right or left elimination $l \in \mathbb{N}_{>0}$ times successively on a SUMT T over an alphabet Σ (that is, we respectively apply the composition $\mathcal{RE}^l$ of l copies of $\mathcal{RE}$ or $\mathcal{LE}^l$ of l copies of $\mathcal{LE}$ to T), we respectively remove T's l rightmost or leftmost internal

nodes. If T is the decomposition of the SUM $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ ($k \in \mathbb{N}$) over Σ, it is obvious that we have

$$\mathcal{L}\big(\mathcal{RE}^l(T)\big) = A_0^* a_1 A_1^* a_2 \cdots A_{k-l-1}^* a_{k-l} A_{k-l}^*$$

and

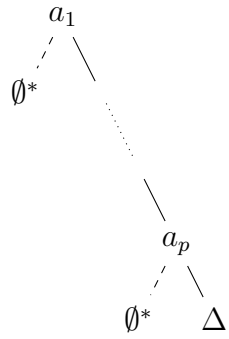
$$\mathcal{L}\big(\mathcal{LE}^l(T)\big) = A_l^* a_{l+1} A_{l+1}^* a_{l+2} \cdots A_{k-1}^* a_k A_k^* \ .$$

Finally, it will also be important to note that right and left elimination both preserve fullness of the opposite branch. More precisely, for any SUMT T over an alphabet Σ whose left (right) branch contains only full edges, $\mathcal{RE}(T)$'s ($\mathcal{LE}(T)$'s) left (right) branch does also have the same property.

This notion of elimination and the associated observation we just made allow us to normalise straight NSUMTs even more: given, for instance, a straight right NSUMT over some alphabet Σ that is a decomposition of a SUM $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ ($k \in \mathbb{N}$) over Σ, if we consider $p \in [\![0, k]\!]$ the smallest integer in $[\![0, k]\!]$ such that A_p is not empty, k if it does not exist, then $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ can be decomposed as a straight right NSUMT whose right branch starts with a sequence of p nodes labelled by the letters from a_1 to a_p. This result will be useful in our proof of Proposition 5.4.2 and we now state and prove if formally.

Lemma 5.4.10. *Let $L = A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ be a SUM over a finite alphabet Σ and T an NSUMT over Σ that is a decomposition of it.*

When the right branch of T contains only full edges, let $p \in [\![0, k]\!]$ be the smallest integer in $[\![0, k]\!]$ such that $A_p \neq \emptyset$, k if it does not exist. Then L can also be decomposed as the straight right NSUMT

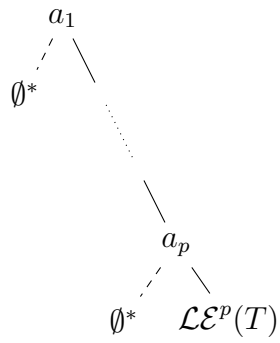


where Δ is a straight right NSUMT over Σ that is a decomposition of $A_p^* a_{p+1} \cdots A_{k-1}^* a_k A_k^*$ and the edges between the nodes labelled a_1 to a_p are all full.

When the left branch of T contains only full edges, the symmetric conclusion follows.

Proof. Assume the right branch of T contains only full edges and let $p \in [\![0, k]\!]$ be the smallest integer in $[\![0, k]\!]$ such that $A_p \neq \emptyset$, k if it does not exist.

It is rather direct to see that L can then be decomposed as



where the edges between the nodes labelled a_1 to a_p are all full.

Moreover, since $\mathcal{LE}^p(T)$ still has the property that its right branch contains only full edges, by Lemma 5.4.7, we get that $A_p^* a_{p+1} \cdots A_{k-1}^* a_k A_k^*$ can be decomposed as a straight right NSUMT Δ over Σ.

The desired conclusion follows.

When the left branch of T contains only full edges, we proceed by symmetry. □

SUMTs for languages of dot-depth one

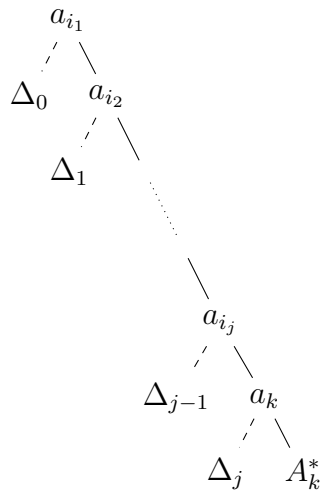
Recall now that what we want to do is, given a SUM L over some alphabet Σ that is also a dot-depth one language, prove that it is equal to a threshold dot-depth one language. Now that we know by Lemma 5.4.7 that there necessarily exists an NSUMT T over Σ that is a decomposition of L, we need to understand what properties T has that characterise the fact that the SUM L it is a decomposition of is a language of dot-depth one.

We can actually give an inductive characterisation of the NSUMTs that are decompositions of SUMs which are also languages of dot-depth one. This characterisation will be very useful when manipulating those SUMTs in our inductive proof of Proposition 5.4.2. The formal statement of the lemma giving that characterisation as well as its proof, both technical, are left for the end of this section (Lemma 5.4.22); here we might only give an intuitive presentation of that characterisation.

Let $L = A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ ($k \in \mathbb{N}$) be a SUM over an alphabet Σ and T an NSUMT over Σ that is a decomposition of it. Whether or not L is a language of dot-depth one depends on the type, the shape and the labels of the normalised SUMT T.

- If T only consists of a root, then L is always a language of dot-depth one.

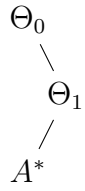
- If T is a straight right NSUMT of the form



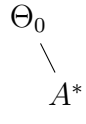
  where $\Delta_0, \Delta_1, \ldots, \Delta_{j-1}, \Delta_j$ ($j \in \mathbb{N}$) are straight left NSUMTs over Σ and the edges between the nodes labelled a_{i_1} to a_{i_j} are all full, then L is a language of dot-depth one if and only if the language $\mathcal{L}(\Delta_0)a_{i_1}\mathcal{L}(\Delta_1)a_{i_2}\cdots\mathcal{L}(\Delta_{j-1})a_{i_j}\mathcal{L}(\Delta_j)$ obtained by applying right elimination on T is of dot-depth one and some technical condition on A_k is verified. This condition states that for each $p \in [\![0, k-1]\!]$, we have that the intersection between A_p and A_k is empty, or at least some letter a_{p+1} to a_k does not belong to A_k, or the SUM $A_p^* a_{p+1} \cdots A_{k-1}^* a_k A_k^*$ and SUMT $\mathcal{LE}^p(T)$ that is a decomposition of it are very constrained.

- If T is a straight left NSUMT, then L is a language of dot-depth one if and only if the symmetric conditions are verified.

- If T is a bent right NSUMT of the form



  where

$\Delta_0 =$ 

is a straight right NSUMT and

$$\Delta_1 = \begin{array}{c} \Theta_1 \\ / \\ A^* \end{array}$$

is a straight left NSUMT, then L is a language of dot-depth one if and only if $\mathcal{L}(\Delta_0)$ and $\mathcal{L}(\Delta_1)$ both are.

- If T is a bent left NSUMT, then L is a language of dot-depth one if and only if the symmetric conditions are verified.

Some examples to illustrate this characterisation (that is not easy to grasp) as well as the additional notions introduced in the next subsection are given in Subsection 5.4.3.

5.4.2 Specific preliminaries to the main proof

In our proof of Proposition 5.4.2, we need an additional set of specific definitions and associated observations that we are going to present now.

Boolean combinations

The goal of the proof is to inductively build, given a SUM that is a language of dot-depth one, a threshold dot-depth one language equal to this SUM. Threshold dot-depth one languages are Boolean combinations of languages of the form $[u_1, \ldots, u_k]_{\Sigma,l}$, $[u_1, \ldots, u_k[_{\Sigma,l}$, $]u_1, \ldots, u_k]_{\Sigma,l}$ or $]u_1, \ldots, u_k[_{\Sigma,l}$ for Σ an alphabet, $k, l \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$. In the proof of Proposition 5.4.2, we will need to consider Boolean combinations in a more syntactic (rather than semantic) way, so as to be able to make the aforementioned construction work cleanly.

This is why we shall abuse terminology and also use the term "Boolean combination" for a loose notion of a formula over $\{^\complement, \cup, \cap\}$, a tree whose internal nodes are labelled by Boolean operations and its leaves by languages over some alphabet Σ. Given such a formula ψ, we shall denote by $\mathcal{L}(\psi)$ the language over Σ it defines.

Zouave languages

In fact, we will see that the proof we give for Proposition 5.4.2 is even more precise than what we claimed up to that point. Namely, the basic building blocks in the Boolean combination that we build inductively can always be assumed to verify $l = 2$.

Given an alphabet Σ, we will call a *Zouave language*[2] *over* Σ the language Σ^* or any language of the form $[u_1, \dots, u_k]_{\Sigma,2}$, $[u_1, \dots, u_k[_{\Sigma,2}$, $]u_1, \dots, u_k]_{\Sigma,2}$ or $]u_1, \dots, u_k[_{\Sigma,2}$ for $k \in \mathbb{N}_{>0}$ and $u_1, \dots, u_k \in \Sigma^+$.

To give some examples, for each Σ an alphabet and $u \in \Sigma^+$, the piecewise testable language $u \sqcup\!\sqcup \Sigma^*$ is also a Zouave language, because $u \sqcup\!\sqcup \Sigma^* = [u_1, \dots, u_{|u|}]_{\Sigma,2}$. The language $[ab, c]_{\{a,b,c\},2}$, containing all words over $\{a, b, c\}$ such that they contain a letter c verifying that in the prefixes up to that letter, *abab* appears a subword or *ab* appears as a factor, is also a Zouave language. However, the language $[ab, c]_{\{a,b,c\},3}$, which description is the same as the previous one except that *abab* is replaced with *ababab*, is not a Zouave language.

What our proof of Proposition 5.4.2 then does is to inductively build, given a SUM that is a language of dot-depth one, a Boolean combination of Zouave languages equal to this SUM.

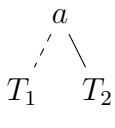
Reversal

As the reader may have observed, there are two symmetric, right and left, versions of straight NSUMTs and two symmetric, right and left, versions of bent NSUMTs. To avoid treating symmetric cases in the proof of Proposition 5.4.2, we will reverse the NSUMT at hand when needed to maintain the invariant that it always either is just a root or has a full right edge at the root. The definition of the reversal of a SUM is straightforward and is linked to the classical definition of the reversal of a language; we now give both.

Fix an alphabet Σ. For any word $w \in \Sigma^*$, we shall denote by $w^{\mathcal{R}} = w_{|w|}w_{|w|-1}\cdots w_1$ the *reversed word of* w. For each language $L \subseteq \Sigma^*$, we can then define $L^{\mathcal{R}} = \{w^{\mathcal{R}} \mid w \in L\}$, the *reversed language of* L.

If T is a SUMT over Σ, then we define the *reversed SUMT* $T^{\mathcal{R}}$ *over* Σ in the following way, by induction:

- if $T = A^*$ for some alphabet $A \subseteq \Sigma$, then $T^{\mathcal{R}} = T$;
- if

$T =$ 

  where a is a letter and T_1, T_2 are SUMTs over Σ, then

[2] In honour of the Zouave statue of the Alma bridge in Paris, whose feet are a well-known popular threshold for the height of the river Seine such that, once reached, usually implies trouble for Parisians.

$$T^{\mathcal{R}} = \begin{array}{ccc} & a & \\ & \diagup\ \diagdown & \\ T_2^{\mathcal{R}} & & T_1^{\mathcal{R}} \end{array};$$

- if

$$T = \begin{array}{ccc} & a & \\ & \diagup\ \diagdown & \\ T_1 & & T_2 \end{array}$$

where a is a letter and T_1, T_2 are SUMTs over Σ, then

$$T^{\mathcal{R}} = \begin{array}{ccc} & a & \\ & \diagup\ \diagdown & \\ T_2^{\mathcal{R}} & & T_1^{\mathcal{R}} \end{array}.$$

It is fairly easy to see that for each SUM L over Σ and SUMT T over Σ that is a decomposition of L, we have $L^{\mathcal{R}} = \mathcal{L}\big(T^{\mathcal{R}}\big)$ and that $L^{\mathcal{R}}$ is also a language of dot-depth one if L is such a language.

Given a Boolean combination ψ of Zouave languages over Σ, we then need to define the reversed such Boolean combination $\psi^{\mathcal{R}}$. We do it straightforwardly in the following way, by induction:

- if $\psi = [u_1, \ldots, u_k]_{\Sigma,2}$, then $\psi^{\mathcal{R}} = [u_k, \ldots, u_1]_{\Sigma,2}$;
- if $\psi = [u_1, \ldots, u_k[_{\Sigma,2}$, then $\psi^{\mathcal{R}} = [u_k, \ldots, u_1[_{\Sigma,2}$;
- if $\psi =]u_1, \ldots, u_k]_{\Sigma,2}$, then $\psi^{\mathcal{R}} =]u_k, \ldots, u_1]_{\Sigma,2}$;
- if $\psi =]u_1, \ldots, u_k[_{\Sigma,2}$, then $\psi^{\mathcal{R}} =]u_k, \ldots, u_1[_{\Sigma,2}$;
- if $\psi = \psi_0^{\complement}$, then $\psi^{\mathcal{R}} = (\psi_0^{\mathcal{R}})^{\complement}$;
- if $\psi = \psi_1 \cup \psi_2$, then $\psi^{\mathcal{R}} = \psi_1^{\mathcal{R}} \cup \psi_2^{\mathcal{R}}$;
- if $\psi = \psi_1 \cap \psi_2$, then $\psi^{\mathcal{R}} = \psi_1^{\mathcal{R}} \cap \psi_2^{\mathcal{R}}$.

Again, it is fairly easy to see that for each language L over Σ that can be expressed as a Boolean combination ψ of Zouave languages over Σ, $L^{\mathcal{R}}$ can be expressed by the reversed such Boolean combination $\psi^{\mathcal{R}}$.

Composition

Given several Boolean combinations of Zouave languages obtained inductively, it is not so direct to see how to combine them so as to obtain a new Boolean combination of Zouave languages corresponding to a given SUM. For instance, given a SUM $A_0^* a_1 A_1^* a_2 A_2^*$ over some alphabet Σ, we might have, by induction, a Boolean combination ψ_0 of Zouave languages for $A_0^* a_1 A_1^*$ and another Boolean combination ψ_1 of Zouave languages for $A_1^* a_2 A_2^*$ that we need to combine in some way to obtain a Boolean combination of Zouave languages for $A_0^* a_1 A_1^* a_2 A_2^*$. But it is not direct to see how to do that, considering that each of ψ_0 and ψ_1 should "be applied" only to either a strict prefix or a strict suffix of the whole word for which we want to determine membership in $A_0^* a_1 A_1^* a_2 A_2^*$. Assume that a_1 does not belong to A_0 and a_2 does not belong to A_2: the idea is that we can change each Zouave language $[u_1, \ldots, u_k]_{\Sigma,2}$ with $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$) to $[u_1, \ldots, u_k, a_2]_{\Sigma,2}$ in ψ_0 and to $[a_1, u_1, \ldots, u_k]_{\Sigma,2}$ in ψ_1, and similarly for Zouave languages of the other forms; we then obtain Boolean combinations of Zouave languages ψ_0' and ψ_1'. The intent behind that transformation is to make sure that, given some word w in $\Sigma^* a_2 (\Sigma \setminus \{a_2\})^*$, w belongs to $A_0^* a_1 A_1^* a_2 (\Sigma \setminus \{a_2\})^*$ if and only if it belongs to the language corresponding to ψ_0', and, similary, that, given some word w in $(\Sigma \setminus \{a_1\})^* a_1 \Sigma^*$, w belongs to $(\Sigma \setminus \{a_1\})^* a_1 A_1^* a_2 A_2^*$ if and only if it belongs to the language corresponding to ψ_1'.

We now formalise this idea through the notion of composition. Given a Boolean combination ψ of Zouave languages over some alphabet Σ and L a Zouave language over Σ, we define the *left composition of ψ with L*, denoted by $L * \psi$, the Boolean combination of Zouave languages defined in the following way, by induction:

- if $\psi = \Sigma^*$, then
  - if $L = \Sigma^*$, then $L * \psi = \Sigma^*$;
  - if $L = [v_1, \ldots, v_l]_{\Sigma,2}$, then $L * \psi = [v_1, \ldots, v_l]_{\Sigma,2}$;
  - if $L = [v_1, \ldots, v_l[_{\Sigma,2}$, then $L * \psi = [v_1, \ldots, v_l]_{\Sigma,2}$;
  - if $L =]v_1, \ldots, v_l]_{\Sigma,2}$, then $L * \psi =]v_1, \ldots, v_l]_{\Sigma,2}$;
  - if $L =]v_1, \ldots, v_l[_{\Sigma,2}$, then $L * \psi =]v_1, \ldots, v_l]_{\Sigma,2}$;
- if $\psi = [u_1, \ldots, u_k]_{\Sigma,2}$, then
  - if $L = \Sigma^*$, then $L * \psi = [u_1, \ldots, u_k]_{\Sigma,2}$;
  - if $L = [v_1, \ldots, v_l]_{\Sigma,2}$, then $L * \psi = [v_1, \ldots, v_l, u_1, \ldots, u_k]_{\Sigma,2}$;
  - if $L = [v_1, \ldots, v_l[_{\Sigma,2}$, then $L * \psi = [v_1, \ldots, v_l, u_1, \ldots, u_k]_{\Sigma,2}$;

  - if $L =]v_1, \ldots, v_l]_{\Sigma,2}$, then $L * \psi =]v_1, \ldots, v_l, u_1, \ldots, u_k]_{\Sigma,2}$;
  - if $L =]v_1, \ldots, v_l[_{\Sigma,2}$, then $L * \psi =]v_1, \ldots, v_l, u_1, \ldots, u_k]_{\Sigma,2}$;
- if $\psi = [u_1, \ldots, u_k[_{\Sigma,2}$, then
  - if $L = \Sigma^*$, then $L * \psi = [u_1, \ldots, u_k[_{\Sigma,2}$;
  - if $L = [v_1, \ldots, v_l]_{\Sigma,2}$, then $L * \psi = [v_1, \ldots, v_l, u_1, \ldots, u_k[_{\Sigma,2}$;
  - if $L = [v_1, \ldots, v_l[_{\Sigma,2}$, then $L * \psi = [v_1, \ldots, v_l, u_1, \ldots, u_k[_{\Sigma,2}$;
  - if $L =]v_1, \ldots, v_l]_{\Sigma,2}$, then $L * \psi =]v_1, \ldots, v_l, u_1, \ldots, u_k[_{\Sigma,2}$;
  - if $L =]v_1, \ldots, v_l[_{\Sigma,2}$, then $L * \psi =]v_1, \ldots, v_l, u_1, \ldots, u_k[_{\Sigma,2}$;
- if $\psi =]u_1, \ldots, u_k]_{\Sigma,2}$, then
  - if $L = \Sigma^*$, then $L * \psi = [u_1, \ldots, u_k]_{\Sigma,2}$;
  - if $L = [v_1, \ldots, v_l]_{\Sigma,2}$, then $L * \psi = [v_1, \ldots, v_l, u_1, \ldots, u_k]_{\Sigma,2}$;
  - if $L = [v_1, \ldots, v_l[_{\Sigma,2}$, then $L * \psi = [v_1, \ldots, v_l u_1, \ldots, u_k]_{\Sigma,2}$;
  - if $L =]v_1, \ldots, v_l]_{\Sigma,2}$, then $L * \psi =]v_1, \ldots, v_l, u_1, \ldots, u_k]_{\Sigma,2}$;
  - if $L =]v_1, \ldots, v_l[_{\Sigma,2}$, then $L * \psi =]v_1, \ldots, v_l u_1, \ldots, u_k]_{\Sigma,2}$;
- if $\psi =]u_1, \ldots, u_k[_{\Sigma,2}$, then
  - if $L = \Sigma^*$, then $L * \psi = [u_1, \ldots, u_k[_{\Sigma,2}$;
  - if $L = [v_1, \ldots, v_l]_{\Sigma,2}$, then $L * \psi = [v_1, \ldots, v_l, u_1, \ldots, u_k[_{\Sigma,2}$;
  - if $L = [v_1, \ldots, v_l[_{\Sigma,2}$, then $L * \psi = [v_1, \ldots, v_l u_1, \ldots, u_k[_{\Sigma,2}$;
  - if $L =]v_1, \ldots, v_l]_{\Sigma,2}$, then $L * \psi =]v_1, \ldots, v_l, u_1, \ldots, u_k[_{\Sigma,2}$;
  - if $L =]v_1, \ldots, v_l[_{\Sigma,2}$, then $L * \psi =]v_1, \ldots, v_l u_1, \ldots, u_k[_{\Sigma,2}$;
- if $\psi = \psi_0^{\complement}$, then $L * \psi = (L * \psi_0)^{\complement}$;
- if $\psi = \psi_1 \cup \psi_2$, then $L * \psi = (L * \psi_1) \cup (L * \psi_2)$;
- if $\psi = \psi_1 \cap \psi_2$, then $L * \psi = (L * \psi_1) \cap (L * \psi_2)$.

We define the *right composition of ψ with L*, denoted by $\psi * L$, in a symmetric way.

Composition realises what we need to combine several Boolean combinations of Zouave languages in a way that each of them allows to check that a part of a given word belongs to a part of a fixed SUM, as we showcased in our example before defining formally

composition. The next lemma is crucial to prove the correctness of the construction given in the proof of Proposition 5.4.2: it says that under some conditions, composition allows to check part of a word for membership in some SUM given by a Boolean combination of Zouave languages, as we explained intuitively in our example. One of those conditions is, in some cases, to verify a convenience property on a Boolean combination ψ of Zouave languages, that intuitively states that the prefix or the suffix of words are not constrained by the Zouave languages in ψ. Formally, a Boolean combination of Zouave languages over some alphabet Σ will be called *left-convenient* if it does not contain any language of the form $]u_1, \dots, u_k]_{\Sigma,2}$ or $]u_1, \dots, u_k[_{\Sigma,2}$ for Σ an alphabet, $k \in \mathbb{N}_{>0}$ and $u_1, \dots, u_k \in \Sigma^+$ and *right-convenient* if it does not contain any language of the form $[u_1, \dots, u_k[_{\Sigma,2}$ or $]u_1, \dots, u_k[_{\Sigma,2}$ for Σ an alphabet, $k \in \mathbb{N}_{>0}$ and $u_1, \dots, u_k \in \Sigma^+$.

Lemma 5.4.11. *Let Σ be an alphabet. Let ψ be a Boolean combination of Zouave languages over Σ, let L be the corresponding language over Σ, let $a_1, \dots, a_k, b_1, \dots, b_l \in \Sigma$ ($k \in \mathbb{N}_{>0}$ and $l \in \mathbb{N}$), $u \in \Sigma^*$ and $c \in \mathrm{alph}(ua_k)$. We have the following.*

1. *If ψ is left-convenient, then*

$$\begin{aligned}&\mathcal{L}\Big([a_1, \dots, a_k]_{\Sigma,2} \cap [a_1, \dots, a_k]_{\Sigma,2} * \psi\Big)\\ =&(\Sigma \setminus \{a_1\})^* a_1 (\Sigma \setminus \{a_2\})^* a_2 \cdots (\Sigma \setminus \{a_k\})^* a_k L\ .\end{aligned}$$

2. *If $|u| \geq 1$, then $\mathcal{L}\Big(]u]_{\Sigma,2} \cap]u[_{\Sigma,2} * \psi\Big) = uL$.*

3. *If ψ is right-convenient, $a_k \notin \mathrm{alph}(b_1 \cdots b_l u)$ and $L \subseteq [a_1, \dots, a_{k-1}]_{\Sigma,2} \cap [a_1, \dots, a_{k-1}, a_k]_{\Sigma,2}{}^{\complement}$, then*

$$\begin{aligned}&\mathcal{L}\left(\begin{matrix}[a_1, \dots, a_{k-1}, b_1, \dots, b_l, ua_k]_{\Sigma,2} \cap [a_1, \dots, a_{k-1}, a_k, c]_{\Sigma,2}{}^{\complement} \cap \\ \psi * [b_1, \dots, b_l, ua_k]_{\Sigma,2}\end{matrix}\right)\\ =&Lb_1(\Sigma \setminus \{b_1, a_k\})^* b_2 (\Sigma \setminus \{b_2, a_k\})^* \cdots b_l (\Sigma \setminus \{b_l, a_k\})^* ua_k (\Sigma \setminus \{c\})^*\ .\end{aligned}$$

4. *If $a_k \notin \mathrm{alph}(u)$ and $L \subseteq [a_k]_{\Sigma,2}{}^{\complement}$, then*

$$\mathcal{L}\Big([ua_k]_{\Sigma,2} \cap [a_k, c]_{\Sigma,2}{}^{\complement} \cap \psi *]ua_k]_{\Sigma,2}\Big) = Lua_k(\Sigma \setminus \{c\})^*\ .$$

5. *If $a_k \notin \mathrm{alph}(u)$ and $L \subseteq [c]_{\Sigma,2}{}^{\complement}$, then*

$$\mathcal{L}\Big([a_1, \dots, a_{k-1}, ua_k]_{\Sigma,2} \cap [a_1, \dots, a_{k-1}, a_k, c]_{\Sigma,2}{}^{\complement} \cap [a_1, \dots, a_{k-1}, ua_k[_{\Sigma,2} * \psi\Big)$$

$$=(\Sigma \setminus \{a_1\})^* a_1 (\Sigma \setminus \{a_2\})^* a_2 \cdots (\Sigma \setminus \{a_{k-1}\})^* a_{k-1} (\Sigma \setminus \{a_k\})^* u a_k L \ .$$

Note that in each case, a symmetric reversed version holds.

Proof. Let Σ be an alphabet. Let ψ be a Boolean combination of Zouave languages over Σ, let L be the corresponding language over Σ, let $a_1, \ldots, a_k, b_1, \ldots, b_l \in \Sigma$ ($k \in \mathbb{N}_{>0}$ and $l \in \mathbb{N}$), $u \in \Sigma^*$ and $c \in \mathrm{alph}(ua_k)$. Let us use a specific notation here to make things clearer in this proof: for all K a language over Σ, we shall write K^+ for K and K^- for its complement $K^{\complement}$. We know that there exist $\alpha \in \mathbb{N}_{>0}$ and for each $i \in [\alpha]$, $\beta_i \in \mathbb{N}_{>0}$ Zouave languages $L_{i,1}, \ldots, L_{i,\beta_i}$ over Σ and associated signs $\gamma_{i,1}, \ldots, \gamma_{i,\beta_i} \in \{+, -\}$ so that

$$\mathcal{L}(\psi) = L = \bigcup_{i=1}^{\alpha} \bigcap_{j=1}^{\beta_i} {L_{i,j}}^{\gamma_{i,j}} \ .$$

The proof in each case is rather straightforward, even if it is sometimes a bit tedious to write.

Case 1. Assume ψ is left-convenient. Let $w \in (\Sigma \setminus \{a_1\})^* a_1 (\Sigma \setminus \{a_2\})^* a_2 \cdots (\Sigma \setminus \{a_k\})^* a_k \Sigma^*$. Then it can be uniquely decomposed as $w = vw'$ where $v \in (\Sigma \setminus \{a_1\})^* a_1 (\Sigma \setminus \{a_2\})^* a_2 \cdots (\Sigma \setminus \{a_k\})^* a_k$ and $w' \in \Sigma^*$. For all $\kappa \in \mathbb{N}_{>0}$ and $\nu_1, \ldots, \nu_\kappa \in \Sigma^+$, it is direct to see that

$$\begin{aligned} &[a_1, \ldots, a_k]_{\Sigma,2} * [\nu_1, \ldots, \nu_\kappa]_{\Sigma,2} = [a_1, \ldots, a_k, \nu_1, \ldots, \nu_\kappa]_{\Sigma,2} = [a_1, \ldots, a_k]_{\Sigma,2} [\nu_1, \ldots, \nu_\kappa]_{\Sigma,2} \\ =&(\Sigma \setminus \{a_1\})^* a_1 (\Sigma \setminus \{a_2\})^* a_2 \cdots (\Sigma \setminus \{a_k\})^* a_k [\nu_1, \ldots, \nu_\kappa]_{\Sigma,2} \end{aligned}$$

and

$$\begin{aligned} &[a_1, \ldots, a_k]_{\Sigma,2} * [\nu_1, \ldots, \nu_\kappa[_{\Sigma,2} = [a_1, \ldots, a_k, \nu_1, \ldots, \nu_\kappa[_{\Sigma,2} = [a_1, \ldots, a_k]_{\Sigma,2} [\nu_1, \ldots, \nu_\kappa[_{\Sigma,2} \\ =&(\Sigma \setminus \{a_1\})^* a_1 (\Sigma \setminus \{a_2\})^* a_2 \cdots (\Sigma \setminus \{a_k\})^* a_k [\nu_1, \ldots, \nu_\kappa[_{\Sigma,2} \ . \end{aligned}$$

This means that, since ψ is a Boolean combination of languages of the form $[\nu_1, \ldots, \nu_\kappa]_{\Sigma,2}$ or $[\nu_1, \ldots, \nu_\kappa[_{\Sigma,2}$ for $\kappa \in \mathbb{N}_{>0}$ and $\nu_1, \ldots, \nu_\kappa \in \Sigma^+$,

$$w \in \mathcal{L}\Big([a_1, \ldots, a_k]_{\Sigma,2} * \psi\Big) = \bigcup_{i=1}^{\alpha} \bigcap_{j=1}^{\beta_i} \big((\Sigma \setminus \{a_1\})^* a_1 (\Sigma \setminus \{a_2\})^* a_2 \cdots (\Sigma \setminus \{a_k\})^* a_k L_{i,j}\big)^{\gamma_{i,j}}$$

if and only if $w' \in L = \bigcup_{i=1}^{\alpha} \bigcap_{j=1}^{\beta_i} {L_{i,j}}^{\gamma_{i,j}}$.

Therefore, since $[a_1,\ldots,a_k]_{\Sigma,2} = (\Sigma\setminus\{a_1\})^*a_1(\Sigma\setminus\{a_2\})^*a_2\cdots(\Sigma\setminus\{a_k\})^*a_k\Sigma^*$, we have that

$$\begin{aligned}&\mathcal{L}\Big([a_1,\ldots,a_k]_{\Sigma,2}\cap[a_1,\ldots,a_k]_{\Sigma,2}*\psi\Big)\\ =&(\Sigma\setminus\{a_1\})^*a_1(\Sigma\setminus\{a_2\})^*a_2\cdots(\Sigma\setminus\{a_k\})^*a_kL\ .\end{aligned}$$

Case 2. Assume $|u|\geq 1$. Let $w\in u\Sigma^*$. Then it can be uniquely decomposed as $w=uw'$ where $w'\in\Sigma^*$. For all $\kappa\in\mathbb{N}_{>0}$ and $\nu_1,\ldots,\nu_\kappa\in\Sigma^+$, it is direct to see that

$$]u[_{\Sigma,2}*[\nu_1,\ldots,\nu_\kappa]_{\Sigma,2}=]u,\nu_1,\ldots,\nu_\kappa]_{\Sigma,2}=]u[_{\Sigma,2}[\nu_1,\ldots,\nu_\kappa]_{\Sigma,2}=u[\nu_1,\ldots,\nu_\kappa]_{\Sigma,2}\ ,$$

$$]u[_{\Sigma,2}*[\nu_1,\ldots,\nu_\kappa[_{\Sigma,2}=]u,\nu_1,\ldots,\nu_\kappa[_{\Sigma,2}=]u[_{\Sigma,2}[\nu_1,\ldots,\nu_\kappa[_{\Sigma,2}=u[\nu_1,\ldots,\nu_\kappa[_{\Sigma,2}\ ,$$

$$]u[_{\Sigma,2}*]\nu_1,\ldots,\nu_\kappa]_{\Sigma,2}=]u\nu_1,\nu_2,\ldots,\nu_\kappa]_{\Sigma,2}=]u[_{\Sigma,2}]\nu_1,\ldots,\nu_\kappa]_{\Sigma,2}=u]\nu_1,\ldots,\nu_\kappa]_{\Sigma,2}$$

and

$$]u[_{\Sigma,2}*]\nu_1,\ldots,\nu_\kappa[_{\Sigma,2}=]u\nu_1,\nu_2,\ldots,\nu_\kappa[_{\Sigma,2}=]u[_{\Sigma,2}]\nu_1,\ldots,\nu_\kappa[_{\Sigma,2}=u]\nu_1,\ldots,\nu_\kappa[_{\Sigma,2}\ .$$

This means that, since ψ is a Boolean combination of Zouave languages,

$$w\in\mathcal{L}\Big(]u[_{\Sigma,2}*\psi\Big)=\bigcup_{i=1}^{\alpha}\bigcap_{j=1}^{\beta_i}(uL_{i,j})^{\gamma_{i,j}}$$

if and only if $w'\in L=\bigcup_{i=1}^{\alpha}\bigcap_{j=1}^{\beta_i}{L_{i,j}}^{\gamma_{i,j}}$.

Therefore, since $]u]_{\Sigma,2}=u\Sigma^*$, we have that $\mathcal{L}\Big(]u]_{\Sigma,2}\cap]u[_{\Sigma,2}*\psi\Big)=uL$.

Case 3. Assume ψ is right-convenient, $a_k\notin\mathrm{alph}(b_1\cdots b_lu)$ and $L\subseteq[a_1,\ldots,a_{k-1}]_{\Sigma,2}\cap{[a_1,\ldots,a_{k-1},a_k]_{\Sigma,2}}^{\complement}$. Let $w\in\Sigma^*b_1(\Sigma\setminus\{b_1,a_k\})^*b_2(\Sigma\setminus\{b_2,a_k\})^*\cdots b_l(\Sigma\setminus\{b_l,a_k\})^*ua_k(\Sigma\setminus\{c\})^*$. Then it can be uniquely decomposed as $w=w'v$ where $v\in b_1(\Sigma\setminus\{b_1,a_k\})^*b_2(\Sigma\setminus\{b_2,a_k\})^*\cdots b_l(\Sigma\setminus\{b_l,a_k\})^*ua_k(\Sigma\setminus\{c\})^*$ and $w'\in\Sigma^*$. For all $\kappa\in\mathbb{N}_{>0}$, $\nu_1,\ldots,\nu_\kappa\in\Sigma^+$ and $\gamma\in\{+,-\}$, we can show that

$$\begin{aligned}&\Sigma^*b_1(\Sigma\setminus\{b_1,a_k\})^*b_2(\Sigma\setminus\{b_2,a_k\})^*\cdots b_l(\Sigma\setminus\{b_l,a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap\\ &{[\nu_1,\ldots,\nu_\kappa]_{\Sigma,2}}^{\gamma}*[b_1,\ldots,b_l,ua_k]_{\Sigma,2}\\ =&\Sigma^*b_1(\Sigma\setminus\{b_1,a_k\})^*b_2(\Sigma\setminus\{b_2,a_k\})^*\cdots b_l(\Sigma\setminus\{b_l,a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap\\ &{[\nu_1,\ldots,\nu_\kappa,b_1,\ldots,b_l,ua_k]_{\Sigma,2}}^{\gamma}\end{aligned}$$

$$
\begin{aligned}
=&\Sigma^* b_1(\Sigma \setminus \{b_1, a_k\})^* b_2(\Sigma \setminus \{b_2, a_k\})^* \cdots b_l(\Sigma \setminus \{b_l, a_k\})^* u a_k (\Sigma \setminus \{c\})^* \cap \\
&\Big([\nu_1, \ldots, \nu_\kappa]_{\Sigma,2} b_1(\Sigma \setminus \{b_1\})^* b_2(\Sigma \setminus \{b_2\})^* \cdots b_l(\Sigma \setminus \{b_l\})^* \big(u a_k \Sigma^* \cup (u a_k)^2 \sqcup\!\sqcup \Sigma^* \big) \Big)^\gamma \\
=&[\nu_1, \ldots, \nu_\kappa]_{\Sigma,2}{}^\gamma b_1(\Sigma \setminus \{b_1, a_k\})^* b_2(\Sigma \setminus \{b_2, a_k\})^* \cdots b_l(\Sigma \setminus \{b_l, a_k\})^* u a_k (\Sigma \setminus \{c\})^*
\end{aligned}
$$

and, similarly,

$$
\begin{aligned}
&\Sigma^* b_1(\Sigma \setminus \{b_1, a_k\})^* b_2(\Sigma \setminus \{b_2, a_k\})^* \cdots b_l(\Sigma \setminus \{b_l, a_k\})^* u a_k (\Sigma \setminus \{c\})^* \cap \\
&]\nu_1, \ldots, \nu_\kappa]_{\Sigma,2}{}^\gamma * [b_1, \ldots, b_l, u a_k]_{\Sigma,2} \\
=&]\nu_1, \ldots, \nu_\kappa]_{\Sigma,2}{}^\gamma b_1(\Sigma \setminus \{b_1, a_k\})^* b_2(\Sigma \setminus \{b_2, a_k\})^* \cdots b_l(\Sigma \setminus \{b_l, a_k\})^* u a_k (\Sigma \setminus \{c\})^* \ .
\end{aligned}
$$

This means that, since ψ is a Boolean combination of languages of the form $[\nu_1, \ldots, \nu_\kappa]_{\Sigma,2}$ or $]\nu_1, \ldots, \nu_\kappa]_{\Sigma,2}$ for $\kappa \in \mathbb{N}_{>0}$ and $\nu_1, \ldots, \nu_\kappa \in \Sigma^+$,

$$
\begin{aligned}
w \in &\mathcal{L}\begin{pmatrix} \Sigma^* b_1(\Sigma \setminus \{b_1, a_k\})^* b_2(\Sigma \setminus \{b_2, a_k\})^* \cdots b_l(\Sigma \setminus \{b_l, a_k\})^* u a_k (\Sigma \setminus \{c\})^* \cap \\ \psi * [b_1, \ldots, b_l, u a_k]_{\Sigma,2} \end{pmatrix} \\
&= \bigcup_{i=1}^{\alpha} \bigcap_{j=1}^{\beta_i} L_{i,j}{}^{\gamma_{i,j}} b_1(\Sigma \setminus \{b_1, a_k\})^* b_2(\Sigma \setminus \{b_2, a_k\})^* \cdots b_l(\Sigma \setminus \{b_l, a_k\})^* u a_k (\Sigma \setminus \{c\})^*
\end{aligned}
$$

if and only if $w' \in L = \bigcup_{i=1}^{\alpha} \bigcap_{j=1}^{\beta_i} L_{i,j}{}^{\gamma_{i,j}}$.

Therefore, since $[a_1, \ldots, a_{k-1}, b_1, \ldots, b_l, u a_k]_{\Sigma,2} \cap [a_1, \ldots, a_{k-1}, a_k, c]_{\Sigma,2}{}^{\complement} \subseteq \Sigma^* b_1(\Sigma \setminus \{b_1, a_k\})^* b_2(\Sigma \setminus \{b_2, a_k\})^* \cdots b_l(\Sigma \setminus \{b_l, a_k\})^* u a_k (\Sigma \setminus \{c\})^*$ and $L \subseteq [a_1, \ldots, a_{k-1}]_{\Sigma,2} \cap [a_1, \ldots, a_{k-1}, a_k]_{\Sigma,2}{}^{\complement}$, we have that

$$
\begin{aligned}
&\mathcal{L}\begin{pmatrix} [a_1, \ldots, a_{k-1}, b_1, \ldots, b_l, u a_k]_{\Sigma,2} \cap [a_1, \ldots, a_{k-1}, a_k, c]_{\Sigma,2}{}^{\complement} \cap \\ \psi * [b_1, \ldots, b_l, u a_k]_{\Sigma,2} \end{pmatrix} \\
=&L b_1(\Sigma \setminus \{b_1, a_k\})^* b_2(\Sigma \setminus \{b_2, a_k\})^* \cdots b_l(\Sigma \setminus \{b_l, a_k\})^* u a_k (\Sigma \setminus \{c\})^* \ .
\end{aligned}
$$

Case 4. Assume $a_k \notin \mathrm{alph}(u)$ and $L \subseteq [a_k]_{\Sigma,2}{}^{\complement}$. Let $w \in (\Sigma \setminus \{a_k\})^* u a_k (\Sigma \setminus \{c\})^*$. Then it can be uniquely decomposed as $w = w'v$ where $v \in u a_k (\Sigma \setminus \{c\})^*$ and $w' \in (\Sigma \setminus \{a_k\})^*$. For all $\kappa \in \mathbb{N}_{>0}$, $\nu_1, \ldots, \nu_\kappa \in \Sigma^+$ and $\gamma \in \{+, -\}$, as no word of $(\Sigma \setminus \{a_k\})^* u a_k (\Sigma \setminus \{c\})^*$ has $(u a_k)^2$ or $(\nu_\kappa u a_k)^2$ as a subword, we can show that

$$
\begin{aligned}
&(\Sigma \setminus \{a_k\})^* u a_k (\Sigma \setminus \{c\})^* \cap [\nu_1, \ldots, \nu_\kappa]_{\Sigma,2}{}^\gamma *]u a_k]_{\Sigma,2} \\
=&(\Sigma \setminus \{a_k\})^* u a_k (\Sigma \setminus \{c\})^* \cap [\nu_1, \ldots, \nu_\kappa, u a_k]_{\Sigma,2}{}^\gamma
\end{aligned}
$$

$$
\begin{aligned}
&=(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap\Big([\nu_1,\dots,\nu_\kappa]_{\Sigma,2}\big(ua_k\Sigma^*\cup(ua_k)^2\sqcup\Sigma^*\big)\Big)^\gamma\\
&=\big([a_k]_{\Sigma,2}{}^{\complement}\cap[\nu_1,\dots,\nu_\kappa]_{\Sigma,2}{}^{\gamma}\big)ua_k(\Sigma\setminus\{c\})^*\ ,
\end{aligned}
$$

$$
\begin{aligned}
&(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap[\nu_1,\dots,\nu_\kappa[_{\Sigma,2}{}^{\gamma}*]ua_k]_{\Sigma,2}\\
=&(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap[\nu_1,\dots,\nu_{\kappa-1},\nu_\kappa ua_k]_{\Sigma,2}{}^{\gamma}\\
=&(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap\Big([\nu_1,\dots,\nu_{\kappa-1}]_{\Sigma,2}\big(\nu_\kappa ua_k\Sigma^*\cup(\nu_\kappa ua_k)^2\sqcup\Sigma^*\big)\Big)^\gamma\\
=&\big([a_k]_{\Sigma,2}{}^{\complement}\cap[\nu_1,\dots,\nu_\kappa[_{\Sigma,2}{}^{\gamma}\big)ua_k(\Sigma\setminus\{c\})^*\ ,
\end{aligned}
$$

as well as, similarly,

$$
\begin{aligned}
&(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap]\nu_1,\dots,\nu_\kappa]_{\Sigma,2}{}^{\gamma}*]ua_k]_{\Sigma,2}\\
=&\big([a_k]_{\Sigma,2}{}^{\complement}\cap]\nu_1,\dots,\nu_\kappa]_{\Sigma,2}{}^{\gamma}\big)ua_k(\Sigma\setminus\{c\})^*
\end{aligned}
$$

and

$$
\begin{aligned}
&(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap]\nu_1,\dots,\nu_\kappa[_{\Sigma,2}{}^{\gamma}*]ua_k]_{\Sigma,2}\\
=&\big([a_k]_{\Sigma,2}{}^{\complement}\cap]\nu_1,\dots,\nu_\kappa[_{\Sigma,2}{}^{\gamma}\big)ua_k(\Sigma\setminus\{c\})^*\ .
\end{aligned}
$$

This means that, since ψ is a Boolean combination of Zouave languages,

$$
w\in\mathcal{L}\Big((\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap\psi*]ua_k]_{\Sigma,2}\Big)=\bigcup_{i=1}^{\alpha}\bigcap_{j=1}^{\beta_i}\big([a_k]_{\Sigma,2}{}^{\complement}\cap L_{i,j}{}^{\gamma_{i,j}}\big)ua_k(\Sigma\setminus\{c\})^*
$$

if and only if $w'\in L=[a_k]_{\Sigma,2}{}^{\complement}\cap L=\bigcup_{i=1}^{\alpha}\bigcap_{j=1}^{\beta_i}\big([a_k]_{\Sigma,2}{}^{\complement}\cap L_{i,j}{}^{\gamma_{i,j}}\big)$.

Therefore, since $[ua_k]_{\Sigma,2}\cap[a_k,c]_{\Sigma,2}{}^{\complement}=(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*$, we have that

$$
\mathcal{L}\Big([ua_k]_{\Sigma,2}\cap[a_k,c]_{\Sigma,2}{}^{\complement}\cap\psi*]ua_k]_{\Sigma,2}\Big)=Lua_k(\Sigma\setminus\{c\})^*\ .
$$

Case 5. Assume $a_k\notin\mathrm{alph}(u)$ and $L\subseteq[c]_{\Sigma,2}{}^{\complement}$. Let $w\in(\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}$ $(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*$. Then it can be uniquely decomposed as $w=vw'$ where $v\in(\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}(\Sigma\setminus\{a_k\})^*ua_k$ and $w'\in(\Sigma\setminus\{c\})^*$. For all $\kappa\in\mathbb{N}_{>0}$, $\nu_1,\dots,\nu_\kappa\in\Sigma^+$ and $\gamma\in\{+,-\}$, as no word of $(\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*$ has $a_1\cdots a_{k-1}(ua_k)^2$ or $a_1\cdots a_{k-1}(ua_k\nu_1)^2$ as a subword, we can show

that

$$
\begin{aligned}
&(\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap\\
&[a_1,\ldots,a_{k-1},ua_k[_{\Sigma,2}*[\nu_1,\ldots,\nu_\kappa]_{\Sigma,2}{}^{\gamma}\\
=&(\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap\\
&[a_1,\ldots,a_{k-1},ua_k,\nu_1,\ldots,\nu_\kappa]_{\Sigma,2}{}^{\gamma}\\
=&(\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap\\
&\Big((\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}(\Sigma\setminus\{a_k\})^*\big(ua_k\cup(ua_k)^2\sqcup\Sigma^*\big)[\nu_1,\ldots,\nu_\kappa]_{\Sigma,2}\Big)^{\gamma}\\
=&(\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}(\Sigma\setminus\{a_k\})^*ua_k\big([c]_{\Sigma,2}{}^{\complement}\cap[\nu_1,\ldots,\nu_\kappa]_{\Sigma,2}{}^{\gamma}\big)\ ,
\end{aligned}
$$

$$
\begin{aligned}
&(\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap\\
&[a_1,\ldots,a_{k-1},ua_k]_{\Sigma,2}*]\nu_1,\ldots,\nu_\kappa]_{\Sigma,2}{}^{\gamma}\\
=&(\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap\\
&[a_1,\ldots,a_{k-1},ua_k\nu_1,\nu_2\ldots,\nu_\kappa]_{\Sigma,2}{}^{\gamma}\\
=&(\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap\\
&\Big((\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}(\Sigma\setminus\{a_k\})^*\big(ua_k\nu_1\cup(ua_k\nu_1)^2\sqcup\Sigma^*\big)[\nu_2,\ldots,\nu_\kappa]_{\Sigma,2}\Big)^{\gamma}\\
=&(\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}(\Sigma\setminus\{a_k\})^*ua_k\big([c]_{\Sigma,2}{}^{\complement}\cap]\nu_1,\ldots,\nu_\kappa]_{\Sigma,2}{}^{\gamma}\big)\ ,
\end{aligned}
$$

as well as, similarly,

$$
\begin{aligned}
&(\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap\\
&[a_1,\ldots,a_{k-1},ua_k[_{\Sigma,2}*[\nu_1,\ldots,\nu_\kappa[_{\Sigma,2}{}^{\gamma}\\
=&(\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}(\Sigma\setminus\{a_k\})^*ua_k\big([c]_{\Sigma,2}{}^{\complement}\cap[\nu_1,\ldots,\nu_\kappa[_{\Sigma,2}{}^{\gamma}\big)
\end{aligned}
$$

and

$$
\begin{aligned}
&(\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}(\Sigma\setminus\{a_k\})^*ua_k(\Sigma\setminus\{c\})^*\cap\\
&[a_1,\ldots,a_{k-1},ua_k[_{\Sigma,2}*]\nu_1,\ldots,\nu_\kappa[_{\Sigma,2}{}^{\gamma}\\
=&(\Sigma\setminus\{a_1\})^*a_1\cdots(\Sigma\setminus\{a_{k-1}\})^*a_{k-1}(\Sigma\setminus\{a_k\})^*ua_k\big([c]_{\Sigma,2}{}^{\complement}\cap]\nu_1,\ldots,\nu_\kappa[_{\Sigma,2}{}^{\gamma}\big)\ .
\end{aligned}
$$

This means that, since ψ is a Boolean combination of Zouave languages,

$$\begin{aligned} w \in \mathcal{L}\begin{pmatrix} (\Sigma \setminus \{a_1\})^* a_1 \cdots (\Sigma \setminus \{a_{k-1}\})^* a_{k-1} (\Sigma \setminus \{a_k\})^* u a_k (\Sigma \setminus \{c\})^* \cap \\ [a_1, \ldots, a_{k-1}, u a_k[_{\Sigma,2} * \psi \end{pmatrix} \\ = \bigcup_{i=1}^{\alpha} \bigcap_{j=1}^{\beta_i} (\Sigma \setminus \{a_1\})^* a_1 \cdots (\Sigma \setminus \{a_{k-1}\})^* a_{k-1} (\Sigma \setminus \{a_k\})^* u a_k \big({[c]_{\Sigma,2}}^{\complement} \cap {L_{i,j}}^{\gamma_{i,j}} \big) \end{aligned}$$

if and only if $w' \in L = {[c]_{\Sigma,2}}^{\complement} \cap L = \bigcup_{i=1}^{\alpha} \bigcap_{j=1}^{\beta_i} ({[c]_{\Sigma,2}}^{\complement} \cap {L_{i,j}}^{\gamma_{i,j}})$.

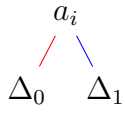
Therefore, since $[a_1, \ldots, a_{k-1}, u a_k]_{\Sigma,2} \cap {[a_1, \ldots, a_{k-1}, a_k, c]_{\Sigma,2}}^{\complement} = (\Sigma \setminus \{a_1\})^* a_1 \cdots (\Sigma \setminus \{a_{k-1}\})^* a_{k-1} (\Sigma \setminus \{a_k\})^* u a_k (\Sigma \setminus \{c\})^*$, we have that

$$\begin{aligned} &\mathcal{L}\Big([a_1, \ldots, a_{k-1}, u a_k]_{\Sigma,2} \cap {[a_1, \ldots, a_{k-1}, a_k, c]_{\Sigma,2}}^{\complement} \cap [a_1, \ldots, a_{k-1}, u a_k[_{\Sigma,2} * \psi \Big) \\ =&(\Sigma \setminus \{a_1\})^* a_1 (\Sigma \setminus \{a_2\})^* a_2 \cdots (\Sigma \setminus \{a_{k-1}\})^* a_{k-1} (\Sigma \setminus \{a_k\})^* u a_k L \ . \end{aligned}$$ □

Three measures on NSUMTs

To write the proof of Proposition 5.4.2, we have to introduce some slightly far-fetched measures on NSUMTs to be used as parameters on which to do the induction. We could probably use more natural parameters, but we did not find better ones.

Given an alphabet Σ, we shall define three measures on NSUMTs over Σ, *major weight*, *minor weight* and *constraint level*, that will help us for the upcoming proof by induction. If T is just a root or a bent NSUMT over Σ, these three measures are equal to -1 on T. If T is a straight NSUMT over Σ that is a decomposition of a SUM $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ over Σ of degree $k \in \mathbb{N}_{>0}$, of the form



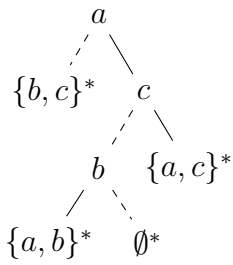
where $i \in [k]$, Δ_0 is a straight left NSUMT over Σ, Δ_1 is a straight right NSUMT over Σ and the red edge is either full or dashed while the blue one is of the opposite type. Assuming the blue edge is full (and the red edge is therefore dashed), we define that

- the *major weight of* T is equal to k minus the number of nodes on the right branch of T, not counting the nodes whose left subtrees only have leaves labelled by $\emptyset^*$;
- the *minor weight of* T is equal to k minus the number of nodes on the left branch of Δ_0, not counting the nodes whose right subtrees only have leaves labelled by $\emptyset^*$;

- the *constraint level of* T is equal to the number of $p \in [k-1]$ such that $\{a_{p+1}, \ldots, a_k\} \subseteq A_k$ and $A_p \cap A_k \neq \emptyset$.

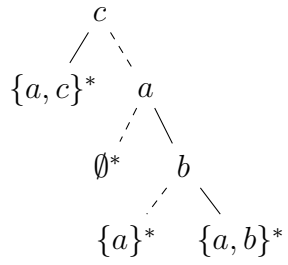
When the red edge is full (and the blue edge is therefore dashed), we define the major weight, the minor weight and the constraint level of T in a symmetric way, replacing "right" with "left", "left" with "right", "Δ_0" with "Δ_1", "$\{a_{p+1}, \ldots, a_k\} \subseteq A_k$" with "$\{a_1, \ldots, a_{p-1}\} \subseteq A_0$" and "$A_p \cap A_k \neq \emptyset$" with "$A_0 \cap A_p \neq \emptyset$".

Example 5.4.12. The following straight right NSUMT over $\{a, b, c\}$



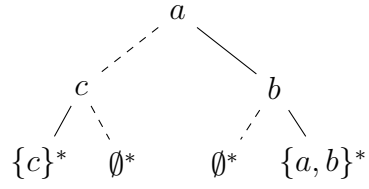
has major weight $3 - 2 = 1$, minor weight $3 - 0 = 3$ and constraint level 0.

Example 5.4.13. The following straight right NSUMT over $\{a, b, c\}$



has major weight $3 - 1 = 2$, minor weight $3 - 1 = 2$ and constraint level 1.

Example 5.4.14. The following straight right NSUMT over $\{a, b, c\}$



has major weight $3 - 1 = 2$, minor weight $3 - 0 = 3$ and constraint level 0.

Equational characterisation of $\mathbf{J} * \mathbf{D}$

Let us finish with the equational characterisation of $\mathbf{J} * \mathbf{D}$, that we shall use several times to prove that a language is of dot-depth one, i.e. belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$.

Let L be some language over some alphabet Σ. In the following proofs of the technical lemmata used in the main proof, to show that $L \in \mathcal{L}(\mathbf{J} * \mathbf{D})$, we might use Knast's equational characterisation of $\mathbf{J} * \mathbf{D}$ [Knast, 1983, Theorem 8] (see also [Straubing, 1985, Theorem 8.1]): given η the syntactic morphism of L and $\sim$ its syntactic congruence, given its syntactic monoid $(M, *)$ and ω its idempotent power, the semigroup $(\eta(\Sigma^+), *|_{\eta(\Sigma^+)})$, that necessarily has the same idempotent power, belongs to $\mathbf{J} * \mathbf{D}$ if and only if for all $e_1, a, e_2, b, c, d \in \eta_L(\Sigma^+)$ such that e_1 and e_2 are idempotents, we have

$$\begin{aligned}&(e_1 * a * e_2 * b * e_1)^{*,\omega} * (e_1 * c * e_2 * d * e_1)^{*,\omega}\\ =&(e_1 * a * e_2 * b * e_1)^{*,\omega} * e_1 * a * e_2 * d * e_1 * (e_1 * c * e_2 * d * e_1)^{*,\omega} \ .\end{aligned}$$

By definition of the syntactic monoid and morphism of L, this means that, equivalently, L belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ if and only if for all $e, s, f, t, u, v \in \Sigma^+$ such that $\eta(e)$ and $\eta(f)$ are idempotents, we have

$$(esfte)^\omega (eufve)^\omega \sim (esfte)^\omega esfve(eufve)^\omega \ .$$

We shall also use the fact that if L belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$, then for all $e, u, v \in \Sigma^+$ such that $\eta(e)$ is idempotent, we have $(eueve)^\omega \sim (eveue)^\omega$, because

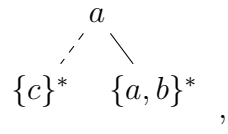
$$\begin{aligned}(eueve)^\omega &\sim (eeeee)^\omega (eueve)^\omega\\ &\sim (eeeee)^\omega eeeve(eueve)^\omega\\ &\sim (eveue)^\omega eve\\ &\sim (eveue)^\omega eveee(eeeee)^\omega\\ &\sim (eveue)^\omega (eeeee)^\omega\\ &\sim (eveue)^\omega \ .\end{aligned}$$

5.4.3 Some examples

The main proof of Proposition 5.4.2 heavily relies on Lemma 5.4.22, a technical result introduced intuitively at the end of Subsection 5.4.1 and proved separately later that gives an exact characterisation of the conditions a given NSUMT should verify in order for the

SUM it is a decomposition of to belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$. Before moving to this main proof, we start with some positive and negative examples of SUMs with respect to membership in $\mathcal{L}(\mathbf{J} * \mathbf{D})$, along with an adequate Boolean combination of Zouave languages when membership holds. This should give the reader an idea about the conditions that any SUM belonging to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ verifies and how those can help to express them as a Boolean combinations of Zouave languages, using the notions introduced in the previous subsection.

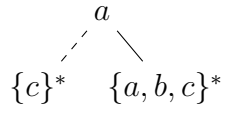
Example 5.4.15. The SUM $c^*a(a+b)^*$ over $\{a,b,c\}$, that can be decomposed as the following straight right NSUMT



belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ because

$$c^*a(a+b)^* = [a]_{\Sigma,2} \cap \Big(\big([c]_{\Sigma,2}{}^{\complement} \cap]a]_{\Sigma,2} \big) \cup \big([c]_{\Sigma,2} \cap [a,c]_{\Sigma,2}{}^{\complement} \cap [b,c]_{\Sigma,2}{}^{\complement} \cap [ca]_{\Sigma,2} \big) \Big) \ .$$

However, the SUM $c^*a(a+b+c)^*$ over $\{a,b,c\}$, that can be decomposed as the following straight right NSUMT

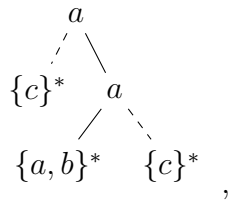


does not belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ because, if we denote by $\sim$ its syntactic congruence and ω the idempotent power of its syntatic monoid, if it were the case we would have

$$(c^\omega a c^\omega b c^\omega)^\omega \sim (c^\omega b c^\omega a c^\omega)^\omega$$

noting that while the first word belongs to $c^*a(a+b+c)^*$, the second one does not, a contradiction.

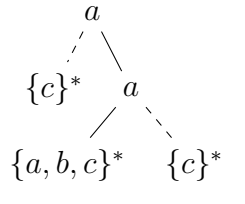
Example 5.4.16. The SUM $c^*a(a+b)^*ac^*$ over $\{a,b,c\}$, that can be decomposed as the following bent right NSUMT



belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ because, if we denote by ψ_1 the Boolean combination of Zouave languages given for $c^*a(a+b)^*$ in the previous example and by $\psi_0 = \psi_1^{\mathcal{R}}$ one suited for $(a+b)^*ac^* = \big(c^*a(a+b)^*\big)^{\mathcal{R}}$, we have

$$c^*a(a+b)^*ac^* = [a]_{\Sigma,2} \cap [a]_{\Sigma,2} * \psi_1 \cap \psi_0 * [a]_{\Sigma,2} \ .$$

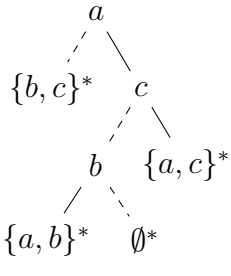
However, the SUM $c^*a(a+b+c)^*ac^*$ over $\{a,b,c\}$, that can be decomposed as the following bent right NSUMT



,

does not belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ because otherwise, by virtue of the fact that $\mathcal{L}(\mathbf{J} * \mathbf{D})$ is an *ne*-variety of languages, we would have that $c^*a(a+b+c)^* = \big(c^*a(a+b+c)^*ac^*\big)a^{-1}$ belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$, a contradiction to what we showed in the previous example.

Example 5.4.17. The SUM $(b+c)^*a(a+b)^*bc(a+c)^*$ over $\{a,b,c\}$, that can be decomposed as the following straight right NSUMT

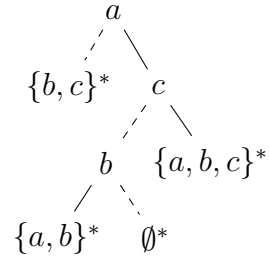


,

belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ because

$$(b+c)^*a(a+b)^*bc(a+c)^* = [a,b,c]_{\Sigma,2} \cap [a,c,b]_{\Sigma,2}^{\complement} \cap [a,bc]_{\Sigma,2} \ .$$

However, the SUM $(b+c)^*a(a+b)^*bc(a+b+c)^*$ over $\{a,b,c\}$, that can be decomposed as the following straight right NSUMT



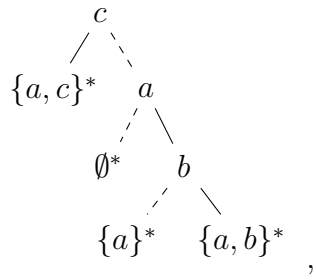
,

does not belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ because, if we denote by $\sim$ its syntactic congruence and ω the idempotent power of its syntatic monoid, if it were the case we would have

$$(c^\omega abcc^\omega acc^\omega)^\omega \sim (c^\omega acc^\omega abcc^\omega)^\omega$$

noting that while the first word belongs to $(b+c)^*a(a+b)^*bc(a+b+c)^*$, the second one does not, a contradiction.

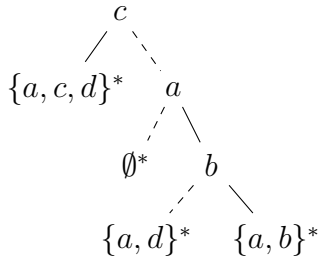
Example 5.4.18. The SUM $(a+c)^*caa^*b(a+b)^*$ over $\{a,b,c\}$, that can be decomposed as the following straight right NSUMT



belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ because

$$(a+c)^*caa^*b(a+b)^* = [cb]_{\Sigma,2}{}^{\complement} \cap [c,a,b]_{\Sigma,2} \cap [b,c]_{\Sigma,2}{}^{\complement} .$$

However, the SUM $(a+c+d)^*ca(a+d)^*b(a+b)^*$ over $\{a,b,c,d\}$, that can be decomposed as the following straight right NSUMT



does not belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ because, if we denote by $\sim$ its syntactic congruence and ω the idempotent power of its syntatic monoid, if it were the case we would have

$$(d^\omega cd^\omega cad^\omega)^\omega b \sim (d^\omega cad^\omega cd^\omega)^\omega b$$

noting that while the first word belongs to $(a+c+d)^*ca(a+d)^*b(a+b)^*$, the second one does not, a contradiction.

5.4.4 The main proof

We now give the main proof of Proposition 5.4.2, leaving the proof of some needed technical lemmata for the next subsection.

Proof of Proposition 5.4.2. Let us fix an alphabet Σ.

We are going to prove something a bit stronger by induction on k, on the number of non-empty alphabets in $\{A_1, \ldots, A_{k-1}\}$ (non-tail alphabets), on $\sum_{i=1}^{k-1} |A_i|$, on the major and the minor weights and on the constraint level (parameters taken in that order): that to any NSUMT T over Σ that is a decomposition of a SUM $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ ($k \in \mathbb{N}$) over Σ in $\mathcal{L}(\mathbf{J} * \mathbf{D})$ we can associate a Boolean combination $\varphi(T)$ of Zouave languages over Σ verifying that:

- $\mathcal{L}(\varphi(T)) = \mathcal{L}(T) = A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$;
- if $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ can be decomposed as a straight left NSUMT, then $\varphi(T)$ is left-convenient;
- if $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ can be decomposed as a straight right NSUMT, then $\varphi(T)$ is right-convenient.

The basic idea of the proof goes like this: depending on the type of the NSUMT T, use the characterisation given by Lemma 5.4.22 and the inductive hypothesis to build $\varphi(T)$. This way of proceeding gives rise to several cases, subcases and even subsubcases that we have to examine carefully; this is the reason why the proof is long and cumbersome in some places, even if its basic idea is simple.

Note that given some $A \subseteq \Sigma$, we shall write $a \notin A$ instead of $a \in \Sigma \setminus A$.

Base case $k = 0$. For any alphabet $A \subseteq \Sigma$,

$$\varphi(A^*) = \bigcap_{b \notin A} [b]_{\Sigma,2}^{\complement}$$

is the desired Boolean combination of Zouave languages.

Induction. Let $(k, \delta_1, \delta_2, \delta_3, \delta_4, \delta_5) \in \mathbb{N}_{>0} \times \mathbb{N}^2 \times (\mathbb{N} \cup \{-1\})^3$ and assume the statement is true for any NSUMT over Σ that is a decomposition of a SUM over Σ of degree l in $\mathcal{L}(\mathbf{J} * \mathbf{D})$, with number of non-empty non-tail alphabets σ_1, sum of cardinals of non-tail alphabets σ_2, major weight σ_3, minor weight σ_4 and constraint level σ_5 such that $(l, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5) < (k, \delta_1, \delta_2, \delta_3, \delta_4, \delta_5)$.

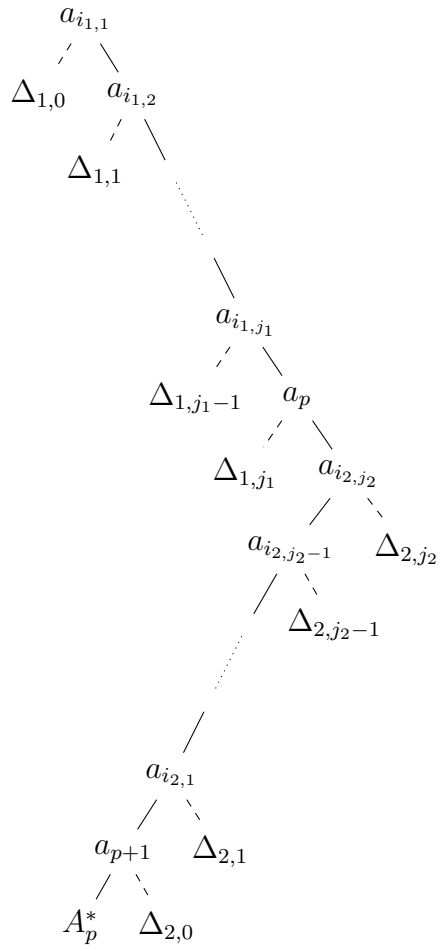
Let T be an NSUMT over Σ that is a decomposition of a SUM $L = A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ over Σ of degree k in $\mathcal{L}(\mathbf{J} * \mathbf{D})$, with a number of non-empty alphabets in $\{A_1, \ldots, A_{k-1}\}$ equal to δ_1, $\sum_{i=1}^{k-1} |A_i| = \delta_2$, major weight δ_3, minor weight δ_4 and constraint level δ_5.

By what we discussed in Subsection 5.4.2, we can assume without loss of generality that T's right edge at the root is full, otherwise we shall just get the expression $\varphi(T^{\mathcal{R}})$ for the reversed NSUMT $T^{\mathcal{R}}$ over Σ, that is a decomposition of the reversed SUM $A_k^* a_k A_{k-1}^* \cdots a_2 A_1^* a_1 A_0^*$ over Σ, and set

$$\varphi(T) = \varphi(T^{\mathcal{R}})^{\mathcal{R}} \ .$$

There are several cases to consider, depending on the shape of T.

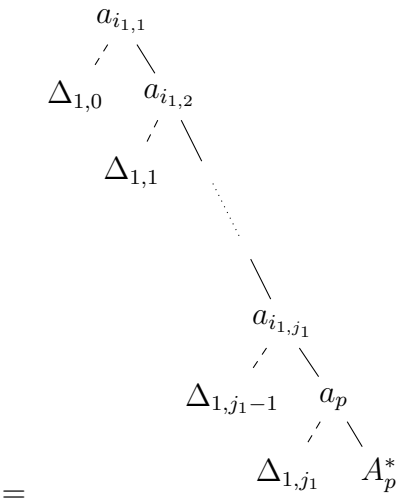
Case 1. T is a bent right NSUMT of the form



where $\Delta_{1,0}, \Delta_{1,1}, \ldots, \Delta_{1,j_1-1}, \Delta_{1,j_1}$ ($j_1 \in \mathbb{N}$) are straight left NSUMTs over Σ, $\Delta_{2,0}, \Delta_{2,1}, \ldots, \Delta_{2,j_2-1}, \Delta_{2,j_2}$ ($j_2 \in \mathbb{N}$) are straight right NSUMTs over Σ, $p \in [\![1, k-1]\!]$ and the edges

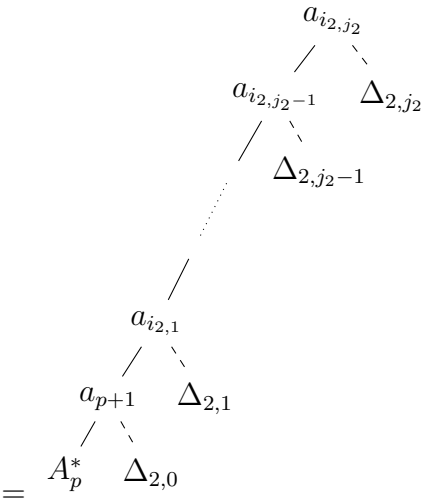
between the nodes labelled $a_{i_{1,1}}$ to $a_{i_{1,j_1}}$ and $a_{i_{2,1}}$ to $a_{i_{2,j_2}}$ are all full. We let

$\Delta_0 =$



and

$\Delta_1 =$



.

By Lemma 5.4.22, both $\mathcal{L}(\Delta_0)$ and $\mathcal{L}(\Delta_1)$ are SUMs over Σ of degree less than k in $\mathcal{L}(\mathbf{J} * \mathbf{D})$. Moreover, $\mathcal{L}(\Delta_0)$ can be decomposed as a straight left NSUMT when L can and $\mathcal{L}(\Delta_1)$ can be decomposed as a straight right NSUMT when L can. Hence, by inductive hypothesis, the Boolean combination $\varphi(\Delta_1)$ of Zouave languages is left-convenient, as well as right-convenient when L can be decomposed as a straight right NSUMT, and verifies $\mathcal{L}(\varphi(\Delta_1)) = \mathcal{L}(\Delta_1)$, and the Boolean combination $\varphi(\Delta_0)$ of Zouave languages is right-convenient, as well as left-convenient when L can be decomposed as a straight left NSUMT, and verifies $\mathcal{L}(\varphi(\Delta_0)) = \mathcal{L}(\Delta_0)$. It is quite direct to see that

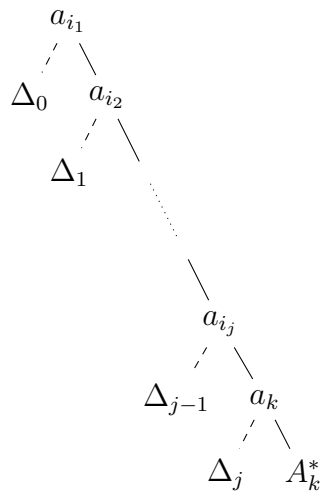
$$\begin{aligned} L =& (\Sigma \setminus \{a_{i_{1,1}}\})^* a_{i_{1,1}} \cdots (\Sigma \setminus \{a_{i_{1,j_1}}\})^* a_{i_{1,j_1}} (\Sigma \setminus \{a_p\})^* a_p \mathcal{L}(\Delta_1) \cap \\ & \mathcal{L}(\Delta_0) a_{p+1} (\Sigma \setminus \{a_{p+1}\})^* a_{i_{2,1}} (\Sigma \setminus \{a_{i_{2,1}}\})^* \cdots a_{i_{2,j_2}} (\Sigma \setminus \{a_{i_{2,j_2}}\})^* \ , \end{aligned}$$

so that by Lemma 5.4.11 (and its reversed version),

$$\begin{aligned}\varphi(T) =&[a_{i_{1,1}}, a_{i_{1,2}}, \dots, a_{i_{1,j_1}}, a_p]_{\Sigma,2} \cap [a_{i_{1,1}}, a_{i_{1,2}}, \dots, a_{i_{1,j_1}}, a_p]_{\Sigma,2} * \varphi(\Delta_1) \cap \\ & [a_{p+1}, a_{i_{2,1}}, \dots, a_{i_{2,j_2-1}}, a_{i_{2,j_2}}]_{\Sigma,2} \cap \varphi(\Delta_0) * [a_{p+1}, a_{i_{2,1}}, \dots, a_{i_{2,j_2-1}}, a_{i_{2,j_2}}]_{\Sigma,2}\end{aligned}$$

is the desired Boolean combination of Zouave languages.

Case 2. T is a straight right NSUMT of the form



where $\Delta_0, \Delta_1, \dots, \Delta_{j-1}, \Delta_j$ ($j \in \mathbb{N}$) are straight left NSUMTs over Σ and the edges between the nodes labelled a_{i_1} to a_{i_j} are all full. We use the convention that $i_0 = 0$ and $i_{j+1} = k$.

In this case, it is less direct to see how to split T and apply the inductive hypothesis on smaller NSUMTs (with respect to the order we consider for the proof by induction) so as to build $\varphi(T)$.

Recall the characterisation given by Lemma 5.4.22: it says in particular that the fact that L is a language of dot-depth one implies that for each $p \in [\![0, k-1]\!]$, the intersection between A_p and A_k is empty, or at least some letter a_{p+1} to a_k does not belong to A_k, or the SUM $A_p^* a_{p+1} \cdots A_{k-1}^* a_k A_k^*$ and SUMT $\mathcal{RE}^p(T)$ that is a decomposition of it are very constrained. How can we use this condition to build a Boolean combination of Zouave languages corresponding to L?

Imagine we can find some $q \in [j+1]$ and some $p \in [\![i_{q-1}, i_q - 1]\!]$ (i.e., such that Δ_{q-1} contains a_p as label of an internal node) verifying that all of $A_{p+1}, \dots, A_{i_q-1}$ are empty, while A_p is not empty. If we denote by Θ_{q-1} some straight left normalised version of the SUM Δ_{q-1} whose $i_q - 1 - p$ internal nodes have been removed, this means that

for any $w \in L$, either there exists some $c \in A_p$ such that w can be decomposed as $w = v_0 a_{i_1} \cdots v_{q-2} a_{i_{q-1}} v_{q-1} c a_{p+1} \cdots a_{i_q-1} a_{i_q} v_q a_{i_{q+1}} \cdots v_j a_k v_{j+1}$ such that $v_\iota \in \mathcal{L}(\Delta_j)$ for all $\iota \in [\![0, j]\!] \setminus \{q-1\}$, $v_{q-1} \in \Theta_{q-1}$ and $v_{j+1} \in A_k^*$, or w belongs to the language L_p that is the SUM L where A_p^* has been replaced by $\emptyset^*$. The idea is that if we have the additional guarantee that while A_k does not contain any element of A_p, all words of the language $a_{p+1} \cdots a_{i_q-1} a_{i_q} \mathcal{L}(\Delta_q) a_{i_{q+1}} \cdots \mathcal{L}(\Delta_j) a_k A_k^*$ do only contain letters from A_k, such a letter c in a word w can be used so that the factor $c a_{p+1} \cdots a_{i_q-1} a_{i_q}$ forms a delimiter that allows to "wedge" the Boolean combinations of Zouave languages obtained by induction for NSUMTs that are decompositions of $\mathcal{L}(\Delta_0) a_{i_1} \cdots \mathcal{L}(\Delta_{q-2}) a_{i_{q-1}} A_{i_{q-1}}^*$, $\mathcal{L}(\Theta_{q-1})$ and $\mathcal{L}(\Delta_q) a_{i_{q+1}} \cdots \mathcal{L}(\Delta_j) a_k A_k^*$ and "apply" them to the correct factor of w so as to test its membership in L. When no such letter c exists, we can essentially just use a Boolean combination of Zouave languages obtained by induction for an NSUMT that is a decomposition of L_p to test membership of w in L. This is the first subcase.

Now, if we do not have this additional guarantee, it may be because at least one of $a_{p+1}, \ldots, a_{i_q}$ does not belong to A_k while, still, all words of the language $\mathcal{L}(\Delta_q) a_{i_{q+1}} \cdots \mathcal{L}(\Delta_j) a_k A_k^*$ do only contain letters from A_k. If we take $p' \in [\![p+1, i_q]\!]$ to be such that $a_{p'}$ does not belong to A_k while all words of the language $a_{p'+1} \cdots a_{i_q} \mathcal{L}(\Delta_q) a_{i_{q+1}} \cdots \mathcal{L}(\Delta_j) a_k A_k^*$ do only contain letters from A_k, we can essentially use this fact to rearrange T into a bent right NSUMT that is also a decomposition of L (where the bend happens at an internal node labelled by $a_{p'}$) and conclude by induction. The second case is therefore the one in which we can find some $q \in [j+1]$ and some $p' \in [\![i_{q-1}+1, i_q]\!]$ verifying that all of $A_{p'}, \ldots, A_{i_q-1}$ are empty, and $a_{p'}$ does not belong to A_k while all words of the language $a_{p'+1} \cdots a_{i_q} \mathcal{L}(\Delta_q) a_{i_{q+1}} \cdots \mathcal{L}(\Delta_j) a_k A_k^*$ do only contain letters from A_k.

The third and last subcase is then the complement of the disjunction of the two first subcases. The idea for this case is that we consider $q \in [j+1]$ the biggest integer in $[j+1]$ such that at least one of $A_{i_{q-1}+1}, \ldots, A_{i_q-1}$ is non-empty or $A_{i_{q-1}}$ is not contained in A_k, 1 if it does not exist. Let then $p \in [\![i_{q-1}, i_q - 1]\!]$ be the biggest integer in $[\![i_{q-1}, i_q - 1]\!]$ such that A_p is not empty, i_{q-1} if it does not exist. By construction and since we are not in the second subcase, it then follows that all words of the language $a_{p+1} \cdots a_{i_q-1} a_{i_q} \mathcal{L}(\Delta_q) a_{i_{q+1}} \cdots \mathcal{L}(\Delta_j) a_k A_k^*$ do only contain letters from A_k. When A_p is empty, then it necessarily means that $q = 1$ and $p = 0$, a specific subsubcase that we handle separately. Otherwise, when A_p is not empty, then since we are not in the first subcase, we must have that the intersection between A_p and A_k is non-empty. The following strong restrictions hence are implied by Lemma 5.4.22:

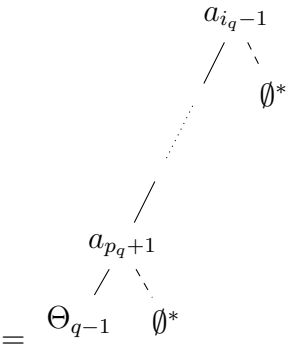
- for each $\iota \in [\![q, j]\!]$, the subtree Δ_ι is reduced to the root $(A_k \setminus \{a_{i_{\iota+1}}\})^*$;

- if $p+1 < i_q$, then there exists some letter $d \in \Sigma$ verifying that all letters from a_{p+1} to a_{i_q-1} are equal to d and the intersection between A_p and A_k only contains d;
- all elements in A_k except a_{i_q} are also in A_p.

We can then consider different subsubcases depending on the values of q and p and for each of them, use the previous properties in different ways to build a Boolean combination of Zouave languages: we won't explain further how we do that to avoid losing the reader with too much details for this intuitive presentation.

We now spell out all the details for each of the three subcases, one after the other.

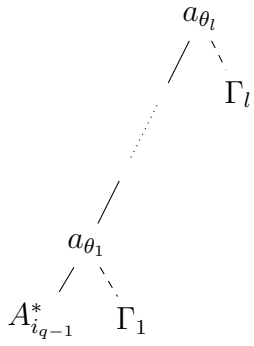
For all $q \in [j+1]$, let $p_q \in [\![i_{q-1}, i_q - 1]\!]$ be the biggest integer in $[\![i_{q-1}, i_q - 1]\!]$ such that $A_{p_q} \neq \emptyset$, i_{q-1} if it does not exist; by Lemma 5.4.10, $\mathcal{L}(\Delta_{q-1})$ can be decomposed as the NSUMT

$\Delta'_{q-1} =$ 

where Θ_{q-1} is a straight left NSUMT over Σ that is a decomposition of $A^*_{i_{q-1}} a_{i_{q-1}+1} \cdots A^*_{p_q-1} a_{p_q} A^*_{p_q}$ and the edges between the nodes labelled a_{p_q+1} to a_{i_q-1} are all full.

Subcase 1. Assume there exists some $q \in [j+1]$ such that $A_{p_q} \neq \emptyset$, $A_{p_q} \cap A_k = \emptyset$, $A_\iota \subseteq A_k$ for all $\iota \in [\![i_q, k-1]\!]$ and $\{a_{p_q+1}, \ldots, a_k\} \subseteq A_k$.

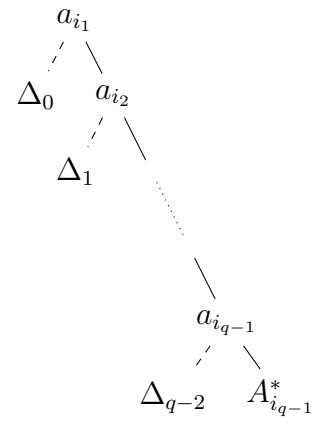
We know that Θ_{q-1} is a straight left NSUMT of the form



where $\Gamma_1, \ldots, \Gamma_l$ ($l \in \mathbb{N}$) are straight right NSUMTs over Σ and the edges between the nodes labelled a_{θ_1} to a_{θ_l} are all full, using the convention that Θ_{q-1} is the sole root $A^*_{i_{q-1}}$ when $l = 0$. By Corollary 5.4.21 (because Δ_{q-1} can be replaced by Δ'_{q-1} in T), we get that $\mathcal{L}(\Theta_{q-1})$ also belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$. Since Θ_{q-1} is a straight left NSUMT and $\mathcal{L}(\Theta_{q-1})$ is a SUM over Σ of degree less than k, by inductive hypothesis, the Boolean combination $\varphi(\Theta_{q-1})$ of Zouave languages is left-convenient and verifies $\mathcal{L}(\varphi(\Theta_{q-1})) = \mathcal{L}(\Theta_{q-1})$.

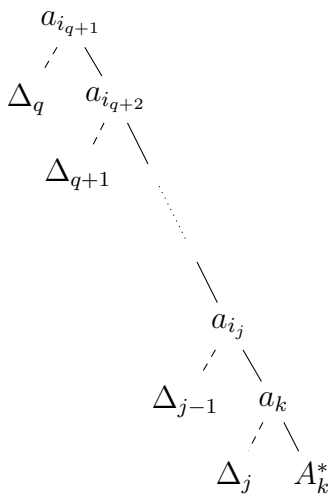
Let

$\Lambda_0 =$



where the edges between the nodes labelled a_{i_1} to $a_{i_{q-1}}$ are all full, using the convention that Λ_0 is simply the root A^*_0 when $q = 1$. By Corollary 5.4.23 when $q > 1$, otherwise using Lemma 5.4.19, we get that $\mathcal{L}(\Lambda_0)$ also belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$. Moreover, $\mathcal{L}(\Lambda_0)$ can be decomposed as a straight left NSUMT when L can. Since Λ_0 is a straight right NSUMT and $\mathcal{L}(\Lambda_0)$ is a SUM over Σ of degree less than k, by inductive hypothesis, the Boolean combination $\varphi(\Lambda_0)$ of Zouave languages is right-convenient, as well as left-convenient when L can be decomposed as a straight left NSUMT, and verifies $\mathcal{L}(\varphi(\Lambda_0)) = \mathcal{L}(\Lambda_0)$.

Let also

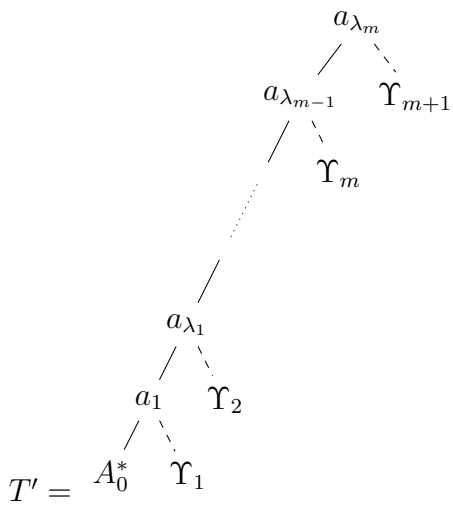
$\Lambda_1 =$ 

where the edges between the nodes labelled $a_{i_{q+1}}$ to a_{i_j} are all full, using the convention that Λ_1 is simply the root A_k^* when $q = j+1$. By Corollary 5.4.21, we get that $\mathcal{L}(\Lambda_1)$ also belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$. Since Λ_1 is a straight right NSUMT and $\mathcal{L}(\Lambda_1)$ is a SUM over Σ of degree less than k, by inductive hypothesis, the Boolean combination $\varphi(\Lambda_1)$ of Zouave languages is right-convenient and verifies $\mathcal{L}(\varphi(\Lambda_1)) = \mathcal{L}(\Lambda_1)$.

Let us denote by T_{p_q} the straight right NSUMT T in which the label at the leaf labelled by $A_{p_q}^*$ has been changed to $\emptyset^*$ and L_{p_q} the associated SUM, which is $L_{p_q} = A_0^* a_1 A_1^* \cdots A_{p_q-1}^* a_{p_q} a_{p_q+1} A_{p_q+1}^* \cdots A_{k-1}^* a_k A_k^*$. By Lemma 5.4.25 when $p_q = 0$ or $a_{p_q} \in A_{p_q}$, or otherwise Lemma 5.4.26 when $a_{p_q} \notin A_{p_q}$, we get that L_{p_q} does also belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$.

When $p_q > 0$, as L_{p_q} contains one non-tail non-empty alphabet less as L and can be decomposed as a straight left NSUMT when L can, by inductive hypothesis, the Boolean combination $\varphi(T_{p_q})$ of Zouave languages is right-convenient, as well as left-convenient when L can be decomposed as a straight left NSUMT, and verifies $\mathcal{L}\big(\varphi(T_{p_q})\big) = \mathcal{L}\big(T_{p_q}\big) = L_{p_q}$.

Finally, when $p_q = 0$, assume L can also be decomposed as the following straight left NSUMT,



where $\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_m, \Upsilon_{m+1}$ ($m \in \mathbb{N}$) are straight right NSUMTs over Σ and the edges between the nodes labelled a_{λ_1} to a_{λ_m} are all full, using the convention that $\lambda_0 = 1$ and $\lambda_{m+1} = k + 1$. Then L_0 can be decomposed as the bent left NSUMT T_0' that is T' in which the label at the leaf labelled by A_0^* has been changed to $\emptyset^*$ and the arcs going from the node labelled a_1 to its children have been switched, the full one becoming dashed and vice versa. The major and minor weights of T_0' both are equal to -1 while at least one of them is positive for T, and L_0 can both be decomposed as the straight right NSUMT T_0 and the straight left NSUMT T' in which the label at the leaf labelled by A_0^* has been changed to $\emptyset^*$, so by inductive hypothesis, the Boolean combination $\varphi(T_0')$ of Zouave languages is both left- and right-convenient and verifies $\mathcal{L}(\varphi(T_0')) = \mathcal{L}(T_0') = L_0$.

Let $w \in L$. This means there exists a unique decomposition $w = u_0 a_1 u_1 a_2 \cdots u_{k-1} a_k u_k$ such that $u_\iota \in A_\iota^*$ for all $\iota \in [\![0, k]\!]$. When $u_{p_q} = \varepsilon$, then either $p_q > 0$ and we have $w \in L_{p_q}$, or $p_q = 0$, which necessarily means $q = 1$, and we have $w \in L_0 = a_1 \cdots a_{i_1 - 1} a_{i_1} \mathcal{L}(\Lambda_1)$. Otherwise, when $u_{p_q} \neq \varepsilon$, there exist $c \in A_{p_q}$ and $v \in A_{p_q}^*$ such that $u_{p_q} = vc$. But note that as $A_{p_q} \cap A_k = \emptyset$, we have $c \notin A_k$, so that $\mathcal{L}(\Lambda_1) \subseteq (\Sigma \setminus \{c\})^*$ because $A_\iota \subseteq A_k$ for all $\iota \in [\![i_q, k-1]\!]$ and $\{a_{p_q+1}, \ldots, a_k\} \subseteq A_k$. So, as $u_0 a_1 \cdots u_{i_{q-1}-1} a_{i_{q-1}} u_{i_{q-1}} \in \mathcal{L}(\Lambda_0)$, $u_{i_{q-1}} a_{i_{q-1}+1} \cdots u_{p_q - 1} a_{p_q} v \in \mathcal{L}(\Theta_{q-1})$ and $u_{i_q} a_{i_q+1} \cdots u_{k-1} a_k u_k \in \mathcal{L}(\Lambda_1)$, this means that

$$\begin{aligned} w \in & \mathcal{L}(\Lambda_0) a_{\theta_1} (\Sigma \setminus \{a_{\theta_1}, a_{i_q}\})^* \cdots a_{\theta_l} (\Sigma \setminus \{a_{\theta_l}, a_{i_q}\})^* c a_{p_q+1} \cdots a_{i_q - 1} a_{i_q} (\Sigma \setminus \{c\})^* \cap \\ & (\Sigma \setminus \{a_{i_1}\})^* a_{i_1} \cdots (\Sigma \setminus \{a_{i_{q-1}}\})^* a_{i_{q-1}} \mathcal{L}(\Theta_{q-1}) c a_{p_q+1} \cdots a_{i_q - 1} a_{i_q} (\Sigma \setminus \{c\})^* \cap \\ & (\Sigma \setminus \{a_{i_1}\})^* a_{i_1} \cdots (\Sigma \setminus \{a_{i_{q-1}}\})^* a_{i_{q-1}} (\Sigma \setminus \{a_{i_q}\})^* c a_{p_q+1} \cdots a_{i_q - 1} a_{i_q} \mathcal{L}(\Lambda_1) \ . \end{aligned}$$

Thus, we can show that, in fact, if we let

$$K = \begin{cases} L_{p_q} & \text{when } p_q > 0 \\ L_0 = a_1 \cdots a_{i_1-1} a_{i_1} \mathcal{L}(\Lambda_1) & \text{otherwise } (p_q = 0) \ , \end{cases}$$

we have

$$\begin{aligned} & L \\ = & \bigcup_{c \in A_{p_q}} \Big(\mathcal{L}(\Lambda_0) a_{\theta_1} (\Sigma \setminus \{a_{\theta_1}, a_{i_q}\})^* \cdots a_{\theta_l} (\Sigma \setminus \{a_{\theta_l}, a_{i_q}\})^* c a_{p_q+1} \cdots a_{i_q-1} a_{i_q} (\Sigma \setminus \{c\})^* \cap \\ & \quad (\Sigma \setminus \{a_{i_1}\})^* a_{i_1} \cdots (\Sigma \setminus \{a_{i_{q-1}}\})^* a_{i_{q-1}} \mathcal{L}(\Theta_{q-1}) c a_{p_q+1} \cdots a_{i_q-1} a_{i_q} (\Sigma \setminus \{c\})^* \cap \\ & \quad (\Sigma \setminus \{a_{i_1}\})^* a_{i_1} \cdots (\Sigma \setminus \{a_{i_{q-1}}\})^* a_{i_{q-1}} (\Sigma \setminus \{a_{i_q}\})^* c a_{p_q+1} \cdots a_{i_q-1} a_{i_q} \mathcal{L}(\Lambda_1) \Big) \\ & \cup K \ . \end{aligned}$$

So by Lemma 5.4.11, if we let

$$\psi = \begin{cases} \varphi(T_{p_q}) & \text{when } p_q > 0 \\ \varphi(T'_0) & \text{when } p_q = 0 \text{ and } L \text{ can be decomposed as } T' \\ ]a_1 \cdots a_{i_1-1} a_{i_1}] \cap]a_1 \cdots a_{i_1-1} a_{i_1}[* \varphi(\Lambda_1) & \text{otherwise } (p_q = 0 \text{ and } L \text{ can't be decomposed as } T') \ , \end{cases}$$

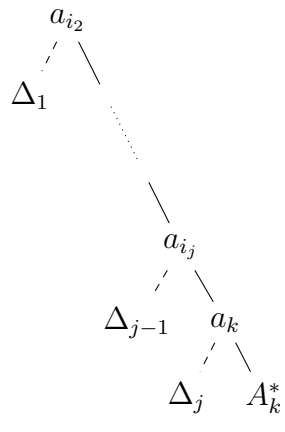
we have that

$$\begin{aligned} & \varphi(T) \\ = & \bigcup_{c \in A_{pq}} \Big([a_{i_1}, \ldots, a_{i_{q-1}}, a_{\theta_1}, \ldots, a_{\theta_l}, c a_{p_q+1} \cdots a_{i_q-1} a_{i_q}]_{\Sigma,2} \cap [a_{i_1}, \ldots, a_{i_{q-1}}, a_{i_q}, c]_{\Sigma,2}{}^{\complement} \cap \\ & \quad \varphi(\Lambda_0) * [a_{\theta_1}, \ldots, a_{\theta_l}, c a_{p_q+1} \cdots a_{i_q-1} a_{i_q}]_{\Sigma,2} \cap \\ & \quad [a_{i_1}, \ldots, a_{i_{q-1}}]_{\Sigma,2} * \big([c a_{p_q+1} \cdots a_{i_q-1} a_{i_q}]_{\Sigma,2} \cap [a_{i_q}, c]_{\Sigma,2}{}^{\complement} \cap \\ & \qquad \varphi(\Theta_{q-1}) *]c a_{p_q+1} \cdots a_{i_q-1} a_{i_q}]_{\Sigma,2} \big) \cap \\ & \quad [a_{i_1}, \ldots, a_{i_{q-1}}, c a_{p_q+1} \cdots a_{i_q-1} a_{i_q}[_{\Sigma,2} * \varphi(\Lambda_1) \Big) \\ & \cup \psi \end{aligned}$$

is the desired Boolean combination of Zouave languages.

Subcase 2. Assume there exists some $q \in [j+1]$ and some $p' \in [\![p_q + 1, i_q]\!]$ such that $a_{p'} \notin A_k$, $A_\iota \subseteq A_k$ for all $\iota \in [\![i_q, k-1]\!]$ and $\{a_{p'+1}, \ldots, a_k\} \subseteq A_k$.

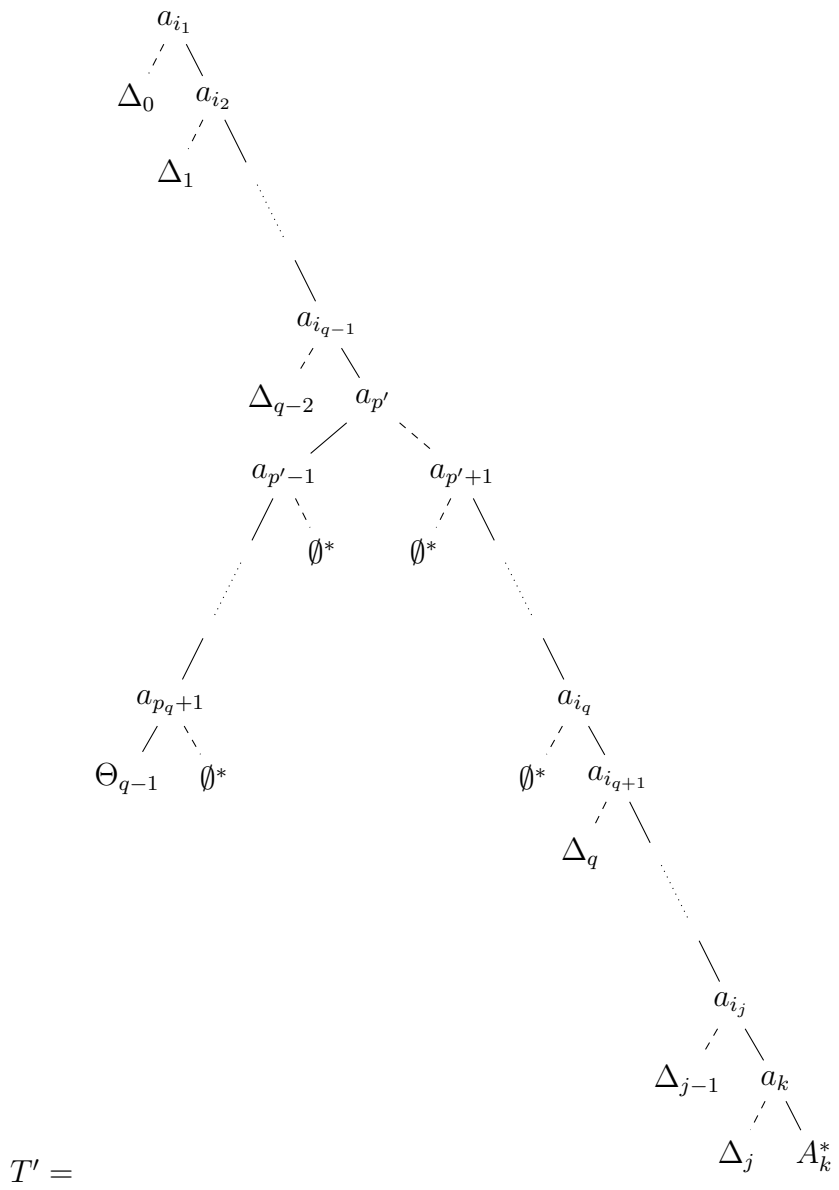
When $q = 1$, let

$\Lambda =$ 

where the edges between the nodes labelled a_{i_2} to a_{i_j} are all full, using the convention that Λ is simply the root A_k^* when $j = 0$. By Corollary 5.4.21 (because Δ_0 can be replaced by Δ_0' in T), we get that both $\mathcal{L}(\Lambda)$ and $\mathcal{L}(\Theta_0)$ also belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$. Since Λ is a straight right NSUMT, Θ_0 is a straight left NSUMT and $\mathcal{L}(\Lambda)$ and $\mathcal{L}(\Theta_0)$ both are SUMs over Σ of respective degrees less than k, by inductive hypothesis, the Boolean combination $\varphi(\Lambda)$ of Zouave languages is right-convenient and verifies $\mathcal{L}(\varphi(\Lambda)) = \mathcal{L}(\Lambda)$, and the Boolean combination $\varphi(\Theta_0)$ of Zouave languages is left-convenient and verifies $\mathcal{L}(\varphi(\Theta_0)) = \mathcal{L}(\Theta_0)$. Note that as $p' \in [\![p_1 + 1, i_1]\!]$ and $a_{p'} \notin A_k$, we have that $\mathcal{L}(\Lambda) \subseteq (\Sigma \setminus \{a_{p'}\})^*$ because $A_\iota \subseteq A_k$ for all $\iota \in [\![i_1, k-1]\!]$ and $\{a_{p'+1}, \ldots, a_k\} \subseteq A_k$. So, as $\mathcal{L}(\Theta_0) \subseteq (\Sigma \setminus \{a_{i_1}\})^*$, we can show that

$$L = \mathcal{L}(\Theta_0) a_{p_1+1} \cdots a_{i_1-1} a_{i_1} (\Sigma \setminus \{a_{p'}\})^* \cap (\Sigma \setminus \{a_{i_1}\})^* a_{p_1+1} \cdots a_{i_1-1} a_{i_1} \mathcal{L}(\Lambda) \ .$$

When $q > 1$, since $a_{p'} \neq a_\iota \in A_k$ and $a_{p'} \notin A_\iota \subseteq A_k$ for all $\iota \in [\![p' + 1, k]\!]$ (because $a_{p'} \notin A_k$), we can define another SUMT T' that is a decomposition of L in the following way:



$T' =$

where the edges between the nodes labelled a_{i_1} to $a_{i_{q-1}}$, a_{p_q+1} to $a_{p'-1}$, $a_{p'+1}$ to a_{i_q} and a_{i_q+1} to a_{i_j} are all full. The major and minor weights of T' both are equal to -1 while at least one of them is positive for T, and L can be decomposed as the straight right NSUMT T, so by inductive hypothesis, the Boolean combination $\varphi(T')$ of Zouave languages is right-convenient, as well as left-convenient when L can be decomposed as a straight left NSUMT, and verifies $\mathcal{L}(\varphi(T')) = \mathcal{L}(T') = L$.

Therefore, by Lemma 5.4.11,

$$\varphi(T) = \begin{cases} \varphi(T') & \text{when } q > 1 \\ [a_{p_1+1} \cdots a_{i_1-1} a_{i_1}]_{\Sigma,2} \cap [a_{i_1}, a_{p'}]_{\Sigma,2}{}^{\complement} \cap & \text{otherwise } (q = 1) \\ \varphi(\Theta_0) *]a_{p_1+1} \cdots a_{i_1-1} a_{i_1}]_{\Sigma,2} \cap \\ [a_{p_1+1} \cdots a_{i_1-1} a_{i_1}[_{\Sigma,2} * \varphi(\Lambda) \end{cases}$$

is the desired Boolean combination of Zouave languages.

Subcase 3. Assume now that for all $q \in [j+1]$, there exists $\iota \in [\![i_q, k-1]\!]$ such that $A_\iota \not\subseteq A_k$ or $\{a_{i_q+1}, \ldots, a_k\} \not\subseteq A_k$ or ($\{a_{p_q+1}, \ldots, a_{i_q}\} \subseteq A_k$ and ($A_{p_q} = \emptyset$ or $A_{p_q} \cap A_k \neq \emptyset$)). Denote by ($\star$) this condition, which is the negation of the disjunction of the conditions for the two first cases.

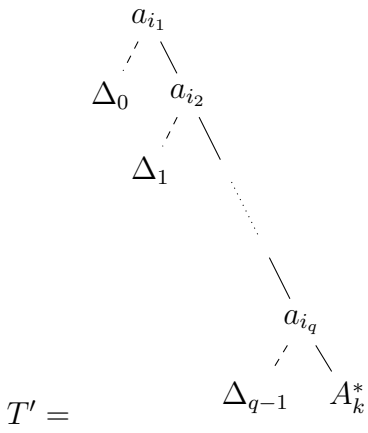
Let $q \in [j+1]$ be the biggest integer in $[j+1]$ such that $p_q > i_{q-1}$ or $A_{p_q} \not\subseteq A_k$, 1 if it does no exist. When $q = j+1$, by condition ($\star$), since there does not exist any $\iota \in [\![k, k-1]\!]$ such that $A_\iota \not\subseteq A_k$ and $\emptyset \subseteq A_k$, we necessarily have that $\{a_{p_{j+1}+1}, \ldots, a_k\} \subseteq A_k$ and ($A_{p_{j+1}} = \emptyset$ or $A_{p_{j+1}} \cap A_k \neq \emptyset$). Otherwise, it is easy to see by (backward) induction that for each $q' \in [\![q+1, j+1]\!]$, we necessarily have $A_\iota \subseteq A_k$ for all $\iota \in [\![i_{q'-1}, k-1]\!]$ and $\{a_{i_{q'-1}+1}, \ldots, a_k\} \subseteq A_k$. Indeed, let $q' \in [\![q+1, j+1]\!]$.

- Assume $q' = j+1$. Because $q < q'$, we have $p_{j+1} = i_j$ and $A_{p_{j+1}} \subseteq A_k$ by maximality of q. By definition of p_{j+1}, this entails $A_{i_j+1} = \cdots = A_{k-1} = \emptyset \subseteq A_k$, so that we necessarily have $A_\iota \subseteq A_k$ for all $\iota \in [\![i_j, k-1]\!]$. Additionally, by condition ($\star$), we necessarily have $\{a_{p_{j+1}+1}, \ldots, a_{i_{j+1}}\} = \{a_{i_j+1}, \ldots, a_k\} \subseteq A_k$.
- Assume $q' < j+1$. Because $q < q'$, we have $p_{q'} = i_{q'-1}$ and $A_{p_{q'}} \subseteq A_k$ by maximality of q. By definition of $p_{q'}$, this entails $A_{i_{q'-1}+1} = \cdots = A_{i_{q'}-1} = \emptyset \subseteq A_k$, so that, since we have $A_\iota \subseteq A_k$ for all $\iota \in [\![i_{q'}, k-1]\!]$ (by induction), we necessarily have $A_\iota \subseteq A_k$ for all $\iota \in [\![i_{q'-1}, k-1]\!]$. Additionally, by condition ($\star$), since there does not exist any $\iota \in [\![i_{q'}, k-1]\!]$ such that $A_\iota \not\subseteq A_k$ and $\{a_{i_{q'}+1}, \ldots, a_k\} \subseteq A_k$ (by induction), we necessarily have $\{a_{p_{q'}+1}, \ldots, a_{i_{q'}}\} \cup \{a_{i_{q'}+1}, \ldots, a_k\} = \{a_{i_{q'-1}+1}, \ldots, a_k\} \subseteq A_k$.

Hence $A_\iota \subseteq A_k$ for all $\iota \in [\![i_q, k-1]\!]$ and $\{a_{i_q+1}, \ldots, a_k\} \subseteq A_k$, so that we necessarily have $\{a_{p_q+1}, \ldots, a_{i_q}\} \cup \{a_{i_q+1}, \ldots, a_k\} = \{a_{p_q+1}, \ldots, a_k\} \subseteq A_k$ and ($A_{p_q} = \emptyset$ or $A_{p_q} \cap A_k \neq \emptyset$) by condition ($\star$). When $A_{p_q} \cap A_k \neq \emptyset$, Lemma 5.4.22 tells us that:

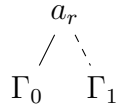
- for all $\iota \in [\![q, j]\!]$, $\Delta_\iota = (A_k \setminus \{a_{i_{\iota+1}}\})^*$;
- if $p_q + 1 < i_q$, then there exists $d \in \Sigma$ verifying $a_{p_q+1} = \cdots = a_{i_q-1} = d$ and $A_{p_q} \cap A_k = \{d\}$;
- $A_k \subseteq A_{p_q} \cup \{a_{i_q}\}$.

Assume that $A_{p_q} \cap A_k \neq \emptyset$ and $q < j + 1$. Let

$T' =$ 

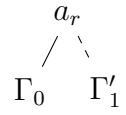
where the edges between the nodes labelled a_{i_1} and a_{i_q} are all full be a SUMT over Σ and L' the SUM over Σ it is a decomposition of. By Lemma 5.4.24, since $\{a_{i_q+1}, \ldots, a_k\} \subseteq A_k$, we get that L' does also belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$. As T' is a straight right NSUMT, L' is a SUM of degree less than k and can be decomposed as a straight left NSUMT when L can (see the next paragraph for more details), by inductive hypothesis, the Boolean combination $\varphi(T')$ of Zouave languages is right-convenient, as well as left-convenient when L can be decomposed as a straight left NSUMT, and verifies $\mathcal{L}(\varphi(T')) = \mathcal{L}(T') = L'$.

Let us justify the fact that L' can be decomposed as a straight left NSUMT when L can, because this is not so easy to see at first glance. So, assume L' can be decomposed as



where Γ_0 is a straight left NSUMT over Σ and Γ_1 is a straight right NSUMT over Σ. As $\{a_{i_q}, \ldots, a_k\} \subseteq A_k$ and the rightmost leaf of Γ_1 is labelled by A_k^*, we necessarily have $r < i_q$, because it must be that $a_r \notin A_k$. Now, since $\mathcal{LE}^r(T')$ is a SUMT over

Σ that is a decomposition of $A_r^* a_{r+1} \cdots A_{i_q-1}^* a_{i_q} A_k^*$ and whose right branch contains only full edges, by Lemma 5.4.7, we get a straight right NSUMT Γ_1' over Σ that is a decomposition of $A_r^* a_{r+1} \cdots A_{i_q-1}^* a_{i_q} A_k^*$. Therefore, as by hypothesis we have $a_r \notin A_\iota$ for all $\iota \in [\![r, k]\!]$ and $a_r \notin \{a_{r+1}, \ldots, a_k\}$, we get that



is a straight left NSUMT over Σ that is a decomposition of L'.

By inductive hypothesis, we also directly have that the Boolean combination $\varphi(A_k^*)$ of Zouave languages is both left- and right-convenient and verifies $\mathcal{L}(\varphi(A_k^*)) = A_k^*$.

Because in the present case,

$$L = \mathcal{L}(\Delta_0) a_{i_1} \cdots \mathcal{L}(\Delta_{q-1}) a_{i_q} (A_k \setminus \{a_{i_{q+1}}\})^* a_{i_{q+1}} \cdots (A_k \setminus \{a_k\})^* a_k A_k^* \ ,$$

it is rather easy to see that

$$\begin{aligned} L =& (\Sigma \setminus \{a_{i_1}\})^* a_{i_1} (\Sigma \setminus \{a_{i_2}\})^* a_{i_2} \cdots (\Sigma \setminus \{a_{i_j}\})^* a_{i_j} (\Sigma \setminus \{a_k\})^* a_k \Sigma^* \cap \\ & (\Sigma \setminus \{a_{i_1}\})^* a_{i_1} (\Sigma \setminus \{a_{i_2}\})^* a_{i_2} \cdots (\Sigma \setminus \{a_{i_q}\})^* a_{i_q} A_k^* \cap L' \ , \end{aligned}$$

so that by Lemma 5.4.11,

$$\varphi(T) = [a_{i_1}, a_{i_2}, \ldots, a_{i_j}, a_k]_{\Sigma,2} \cap [a_{i_1}, a_{i_2}, \ldots, a_{i_q}]_{\Sigma,2} * \varphi(A_k^*) \cap \varphi(T')$$

is the desired Boolean combination of Zouave languages.

Otherwise, in all situations between which we now distinguish, we have $A_{p_q} \cap A_k \neq \emptyset$ and $q = j + 1$, or $A_{p_q} = \emptyset$.

- $p_q = 0$ $(q = 1)$.
  If $A_0 \cap A_k \neq \emptyset$, we have $i_1 = k$, so by what we discussed just above, it follows that
  $$L = \begin{cases} A_0^* d^{k-1} a_k A_k^* & \text{when } 0 < k-1 \\ A_0^* a_k A_k^* & \text{otherwise } (0 = k-1) \end{cases}$$
  with $A_k \subseteq A_0 \cup \{a_k\}$ and, when $0 < k-1$, $A_0 \cap A_k = \{d\}$. This means that

when $0 < k-1$, we have $A_k \subseteq \{d, a_k\}$ and hence

$$\begin{aligned} L &= A_0^* d^{k-1} a_k A_k^* \\ &= A_0^* a_k A_k^* \cap \bigcap_{i=0}^{k-2} \Big((d^i a_k \Sigma^*)^{\complement} \cap \bigcap_{c \in A_0 \setminus \{d\}} \big((\Sigma \setminus \{a_k\})^* c d^i a_k (\Sigma \setminus \{c\})^* \big)^{\complement} \Big) \ . \end{aligned}$$

When $0 < k-1$ and L can also be decomposed as a straight left NSUMT, then it necessarily means that $a_k \notin A_k$ and that its root is this last letter of the monomial, because $d \in A_k$, so that in this case $L \subseteq (\Sigma \setminus \{a_k\})^* d^{k-1} a_k (\Sigma \setminus \{a_k\})^*$. We can then show that, if we let

$$K = \begin{cases} (\Sigma \setminus \{a_k\})^* d^{k-1} a_k (\Sigma \setminus \{a_k\})^* & \text{when } 0 < k-1 \text{ and } L \\ & \text{can be decomposed as a} \\ & \text{straight left NSUMT} \\ \bigcap_{i=0}^{k-2} \Big((d^i a_k \Sigma^*)^{\complement} \cap & \text{when } 0 < k-1 \text{ and } L \\ \qquad \bigcap_{c \in A_0 \setminus \{d\}} & \text{can't be decomposed as} \\ \qquad \big((\Sigma \setminus \{a_k\})^* c d^i a_k (\Sigma \setminus \{c\})^* \big)^{\complement} \Big) & \text{a straight left NSUMT} \\ \Sigma^* & \text{otherwise } (0 = k-1) \ , \end{cases}$$

we have

$$L = (\Sigma \setminus \{a_k\})^* a_k A_k^* \cap \bigcap_{b \notin A_0 \cup \{a_k\}} (\Sigma^* b \Sigma^*)^{\complement} \cap K \ .$$

So, by Lemma 5.4.11 and since, when $0 < k-1$, for each $i \in [\![0, k-2]\!]$ and $c \in A_0 \setminus \{d\}$, no word of $A_0^* a_k A_k^*$ has $(c d^i a_k)^2$ as a subword, if we let

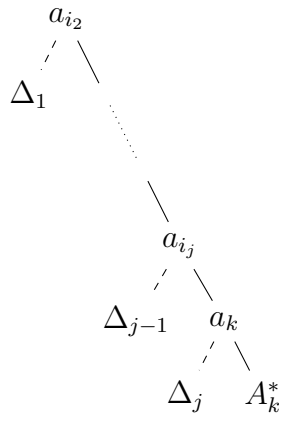
$$\psi = \begin{cases} [d^{k-1} a_k]_{\Sigma,2} & \text{when } 0 < k-1 \text{ and } L \text{ can be decom-} \\ & \text{posed as a straight left NSUMT} \\ \bigcap_{i=0}^{k-2} \Big( ]d^i a_k]_{\Sigma,2}{}^{\complement} \cap & \text{when } 0 < k-1 \text{ and } L \text{ can't be decom-} \\ \qquad \bigcap_{c \in A_0 \setminus \{d\}} [c d^i a_k]_{\Sigma,2}{}^{\complement} \Big) & \text{posed as a straight left NSUMT} \\ \Sigma^* & \text{otherwise } (0 = k-1) \ , \end{cases}$$

we have that

$$\varphi(T) = [a_k]_{\Sigma,2} \cap [a_k]_{\Sigma,2} * \varphi(A_k^*) \cap \bigcap_{b \notin A_0 \cup \{a_k\}} [b]_{\Sigma,2}{}^{\complement} \cap \psi$$

is the desired Boolean combination of Zouave languages.

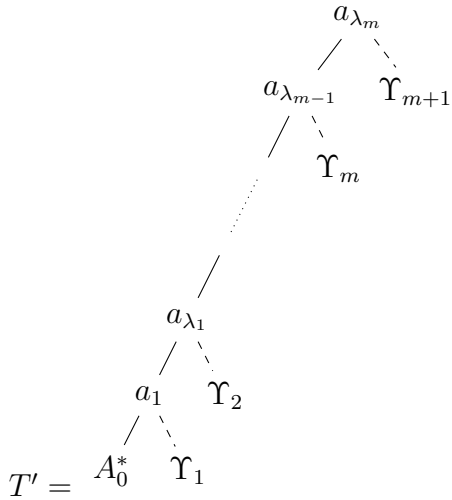
Otherwise, when $A_0 \cap A_k = \emptyset$, then we necessarily have $A_0 = \emptyset$. Let

$\Lambda =$ 

where the edges between the nodes labelled a_{i_2} to a_{i_j} are all full, using the convention that Λ is simply the root A_k^* when $j = 0$. Applying Lemma 5.4.20, we get that $\mathcal{L}(\Lambda)$ also belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$. Since Λ is a straight right NSUMT and $\mathcal{L}(\Lambda)$ is a SUM over Σ of degree less than k, by inductive hypothesis, the Boolean combination $\varphi(\Lambda)$ of Zouave languages is right-convenient and verifies $\mathcal{L}(\varphi(\Lambda)) = \mathcal{L}(\Lambda)$. It is rather straightforward to check that

$$L = a_1 \cdots a_{i_1-1} a_{i_1} \mathcal{L}(\Lambda) \ .$$

Assume now L can also be decomposed as the following straight left NSUMT,

$T' =$ 

where $\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_m, \Upsilon_{m+1}$ ($m \in \mathbb{N}$) are straight right NSUMTs over Σ and the edges between the nodes labelled a_{λ_1} to a_{λ_m} are all full, using the convention

that $\lambda_0 = 1$ and $\lambda_{m+1} = k+1$. Then, as $A_0 = \emptyset$, L can also be decomposed as the bent left NSUMT T'' that is T' in which the arcs going from the node labelled a_1 to its children have been switched, the full one becoming dashed and vice versa. The major and minor weights of T'' both are equal to -1 while at least one of them is positive for T, and L can both be decomposed as the straight right NSUMT T and the straight left NSUMT T', so by inductive hypothesis, the Boolean combination $\varphi(T'')$ of Zouave languages is both left- and right-convenient and verifies $\mathcal{L}(\varphi(T'')) = \mathcal{L}(T'') = L$.

Therefore, by Lemma 5.4.11,

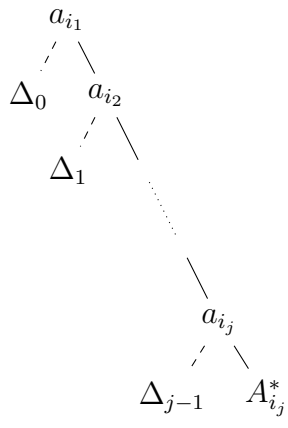
$$\varphi(T) = \begin{cases} \varphi(T'') & \text{when } L \text{ can be decomposed as } T' \\ ]a_1 \cdots a_{i_1-1}a_{i_1}]_{\Sigma,2} \cap]a_1 \cdots a_{i_1-1}a_{i_1}[_{\Sigma,2} * \varphi(\Lambda) & \text{otherwise } (L \text{ can't be decomposed as } T') \end{cases}$$

is the desired Boolean combination of Zouave languages.

- $p_q > 0$ and $p_q = i_{q-1}$.

  Then, it necessarily means that $q > 1$, otherwise we would have $p_q = i_0 = 0$, and consequently that $A_{p_q} \nsubseteq A_k$ by definition of q. So we must have $A_{p_q} \cap A_k \neq \emptyset$ and $q = j+1$ because A_{p_q} cannot be empty.

  Note that $j > 0$ as $j + 1 = q > 1$. Let

$\Lambda =$ 

  where the edges between the nodes labelled a_{i_1} to a_{i_j} are all full. By Corollary 5.4.23, we get that $\mathcal{L}(\Lambda)$ also belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$. Since Λ is a straight right NSUMT, $\mathcal{L}(\Lambda)$ is a SUM of degree less than k and can be decomposed as a straight left NSUMT when L can, by inductive hypothesis, the Boolean combination $\varphi(\Lambda)$ of Zouave languages is right-convenient, as well as left-

convenient when L can be decomposed as a straight left NSUMT, and verifies $\mathcal{L}(\varphi(\Lambda)) = \mathcal{L}(\Lambda)$.

Let us finally denote by T' the straight right NSUMT T in which the label at the leaf labelled by $A_{i_j}^*$ has been changed to $(A_k \setminus \{a_k\})^*$ and L' the associated SUM, which is $L' = A_0^* a_1 A_1^* \cdots A_{i_j-1}^* a_{i_j} (A_k \setminus \{a_k\})^* a_{i_j+1} A_{i_j+1}^* \cdots A_{k-1}^* a_k A_k^*$. By Lemma 5.4.27, since $\{a_{i_j+1}, \ldots, a_k\} \subseteq A_k$, $A_k \subseteq A_{i_j} \cup \{a_k\}$ and $A_{i_j+1} = \cdots = A_{k-1} = \emptyset \subseteq A_k$, we get that L' does also belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$. As the sum of non-tail alphabet cardinals in L' is less than this of L (because $A_{i_j} \nsubseteq A_k \setminus \{a_k\}$ and $i_j = p_q > 0$) and L' can be decomposed as a straight left NSUMT when L can, by inductive hypothesis, the Boolean combination $\varphi(T')$ of Zouave languages is right-convenient, as well as left-convenient when L can be decomposed as a straight left NSUMT, and verifies $\mathcal{L}(\varphi(T')) = \mathcal{L}(T') = L'$.

By inductive hypothesis, we also directly have that the Boolean combination $\varphi\big((A_{i_j} \cup \{a_k\})^*\big)$ of Zouave languages is both left- and right-convenient and verifies $\mathcal{L}\big(\varphi\big((A_{i_j} \cup \{a_k\})^*\big)\big) = (A_{i_j} \cup \{a_k\})^*$.

In the present case, we have

$$L = \begin{cases} \mathcal{L}(\Delta_0) a_{i_1} \cdots \mathcal{L}(\Delta_{j-1}) a_{i_j} A_{i_j}^* d^{k-1-i_j} a_k A_k^* & \text{when } i_j < k-1 \\ \mathcal{L}(\Delta_0) a_{i_1} \cdots \mathcal{L}(\Delta_{j-1}) a_{i_j} A_{i_j}^* a_k A_k^* & \text{otherwise } (i_j = k-1) \end{cases}$$

with $A_k \subseteq A_{i_j} \cup \{a_k\}$ and, when $i_j < k-1$, $A_{i_j} \cap A_k = \{d\}$. This means that when $i_j < k-1$, we have $A_k \subseteq \{d, a_k\}$ and hence for any $b \in A_{i_j} \setminus \{d\} = A_{i_j} \setminus A_k$, since $A_k^* \subseteq (\Sigma \setminus \{c\})^*$ for all $c \in A_{i_j} \setminus \{d\}$,

$$\begin{aligned} & (\Sigma \setminus \{a_{i_1}\})^* a_{i_1} \cdots (\Sigma \setminus \{a_{i_j}\})^* a_{i_j} A_{i_j}^* b (A_{i_j} \setminus \{b\})^* a_k A_k^* \cap \\ & (\Sigma \setminus \{a_{i_1}\})^* a_{i_1} \cdots (\Sigma \setminus \{a_{i_j}\})^* a_{i_j} A_{i_j}^* d^{k-1-i_j} a_k A_k^* \\ = & (\Sigma \setminus \{a_{i_1}\})^* a_{i_1} \cdots (\Sigma \setminus \{a_{i_j}\})^* a_{i_j} A_{i_j}^* b (A_{i_j} \setminus \{b\})^* a_k A_k^* \cap \\ & \bigcap_{i=0}^{k-2-i_j} \bigcap_{c \in A_{i_j} \setminus \{d\}} \big((\Sigma \setminus \{a_{i_1}\})^* a_{i_1} \cdots (\Sigma \setminus \{a_{i_j}\})^* a_{i_j} (\Sigma \setminus \{a_k\})^* \\ & \quad c d^i a_k (\Sigma \setminus \{c\})^* \big)^{\complement} . \end{aligned}$$

We let

$$K = \begin{cases} \bigcap_{i=0}^{k-2-i_j} \bigcap_{c \in A_{i_j} \setminus \{d\}} \big((\Sigma \setminus \{a_{i_1}\})^* a_{i_1} \cdots & \text{when } i_j < k-1 \\ \qquad (\Sigma \setminus \{a_{i_j}\})^* a_{i_j} (\Sigma \setminus \{a_k\})^* & \\ \qquad cd^i a_k (\Sigma \setminus \{c\})^* \big)^{\complement} & \\ \Sigma^* & \text{otherwise } (i_j = k-1) \ . \end{cases}$$

Let $w \in L$. This means there exists a unique decomposition $w = u_0 a_1 u_1 a_2 \cdots u_{k-1} a_k u_k$ such that $u_\iota \in A_\iota^*$ for all $\iota \in [\![0, k]\!]$. When $u_{i_j} \subseteq (A_k \setminus \{a_k\})^*$, then we have $w \in L'$. Otherwise, when $u_{i_j} \nsubseteq (A_k \setminus \{a_k\})^*$, there exist $b \in A_{i_j} \setminus A_k$, $v_0 \in A_{i_j}^*$ and $v_1 \in (A_{i_j} \setminus \{b\})^*$ such that $u_{i_j} = v_0 b v_1$. But note that, as $b \notin A_k \cup \{a_k\}$ (because $a_k \notin A_{i_j}$), we have $(A_{i_j} \setminus \{b\})^* a_k A_k^* \subseteq (\Sigma \setminus \{b\})^*$. So, as $u_0 a_1 \cdots u_{i_j - 1} a_{i_j} v_0 \in \mathcal{L}(\Lambda)$ and $v_1 a_{i_j+1} u_{i_j+1} a_{i_j+2} \cdots u_{k-1} a_k u_k = v_1 d^{k-1-i_j} a_k u_k \in (A_{i_j} \setminus \{b\})^* d^{k-1-i_j} a_k A_k^* \subseteq (A_{i_j} \setminus \{b\})^* a_k A_k^*$ when $i_j < k-1$ and $v_1 a_{i_j+1} u_{i_j+1} a_{i_j+2} \cdots u_{k-1} a_k u_k = v_1 a_k u_k \in (A_{i_j} \setminus \{b\})^* a_k A_k^*$ otherwise ($i_j = k-1$), this means that

$$\begin{aligned} w \in & (\Sigma \setminus \{a_{i_1}\})^* a_{i_1} \cdots (\Sigma \setminus \{a_{i_j}\})^* a_{i_j} A_{i_j}^* b (A_{i_j} \setminus \{b\})^* a_k A_k^* \cap \\ & \mathcal{L}(\Lambda) b (\Sigma \setminus \{b\})^* \cap K \ . \end{aligned}$$

Thus, we can show that, in fact, we have

$$\begin{aligned} L = & \bigcup_{b \in A_{i_j} \setminus A_k} \Big((\Sigma \setminus \{a_{i_1}\})^* a_{i_1} \cdots (\Sigma \setminus \{a_{i_j}\})^* a_{i_j} A_{i_j}^* b (A_{i_j} \setminus \{b\})^* a_k A_k^* \cap \\ & \mathcal{L}(\Lambda) b (\Sigma \setminus \{b\})^* \cap K \Big) \\ & \cup L' \ . \end{aligned}$$

So by Lemma 5.4.11, observing that

$$\begin{aligned} & (\Sigma \setminus \{a_{i_1}\})^* a_{i_1} \cdots (\Sigma \setminus \{a_{i_j}\})^* a_{i_j} A_{i_j}^* b (A_{i_j} \setminus \{b\})^* a_k A_k^* \\ = & (\Sigma \setminus \{a_{i_1}\})^* a_{i_1} \cdots (\Sigma \setminus \{a_{i_j}\})^* a_{i_j} (\Sigma \setminus \{a_k\})^* a_k A_k^* \cap \\ & (\Sigma \setminus \{a_{i_1}\})^* a_{i_1} \cdots (\Sigma \setminus \{a_{i_j}\})^* a_{i_j} (A_{i_j} \cup \{a_k\})^* \cap \\ & (\Sigma \setminus \{a_{i_1}\})^* a_{i_1} \cdots (\Sigma \setminus \{a_{i_j}\})^* a_{i_j} (\Sigma \setminus \{b\})^* b (\Sigma \setminus \{a_k\})^* a_k \Sigma^* \end{aligned}$$

for all $b \in A_{i_j} \setminus A_k$, and since, when $i_j < k-1$, for each $i \in [\![0, k-2-i_j]\!]$ and $c \in A_{i_j} \setminus \{d\}$, no word of $(\Sigma \setminus \{a_{i_1}\})^* a_{i_1} \cdots (\Sigma \setminus \{a_{i_j}\})^* a_{i_j} A_{i_j}^* b (A_{i_j} \setminus \{b\})^* a_k A_k^*$

has $a_{i_1} \cdots a_{i_j}(cd^i a_k)^2$ as a subword, if we let

$$\psi = \begin{cases} \bigcap_{i=0}^{k-2-i_j} \bigcap_{c \in A_{i_j} \setminus \{d\}} [a_{i_1}, \ldots, a_{i_j}, cd^i a_k]_{\Sigma,2}^{\complement} & \text{when } i_j < k-1 \\ \Sigma^* & \text{otherwise } (i_j = k-1) \end{cases},$$

we have that

$$\varphi(T) = \bigcup_{b \in A_{i_j} \setminus A_k} \Big([a_{i_1}, \ldots, a_{i_j}, b, a_k]_{\Sigma,2} \cap [a_{i_1}, \ldots, a_{i_j}, a_k]_{\Sigma,2} * \varphi(A_k^*) \cap [a_{i_1}, \ldots, a_{i_j}]_{\Sigma,2} * \varphi\big((A_{i_j} \cup \{a_k\})^*\big) \cap \varphi(\Lambda) * [b]_{\Sigma,2} \cap \psi \Big) \cup \varphi(T')$$

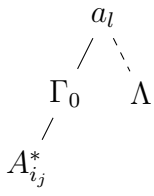
is the desired Boolean combination of Zouave languages.

- $p_q > i_{q-1}$.

  Then, it necessarily means that $A_{p_q} \neq \emptyset$ by definition of p_q (otherwise we would have $p_q = i_{q-1}$), so that $A_{p_q} \cap A_k \neq \emptyset$ and $q = j+1$.

  We shall denote $p_q = p_{j+1}$ simply by p.

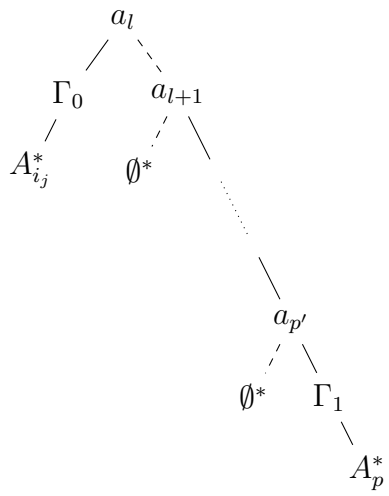
  We know Θ_j is of the form



  where $l \in [\![i_j + 1, p]\!]$, Γ_0 is a straight left NSUMT over Σ whose leftmost leaf has been suppressed (and hence which might be empty) and Λ is a straight right NSUMT over Σ. We can actually assume that l is the biggest integer in $[\![i_j + 1, p]\!]$ such that a_l does not appear in any word of $A_l^* a_{l+1} \cdots A_{p-1}^* a_p A_p^*$: indeed, if there would exist $l' \in [\![l+1, p]\!]$ such that $a_{l'}$ does not appear in any word of $A_{l'}^* a_{l'+1} \cdots A_{p-1}^* a_p A_p^*$, we could obtain a straight left NSUMT that is also a decomposition of $\mathcal{L}(\Theta_j)$ by putting $a_{l'}$ at the root, putting a straight right NSUMT that is a decomposition of $A_{l'}^* a_{l'+1} \cdots A_{p-1}^* a_p A_p^*$ as the right child (that exists since $\mathcal{LE}^{l'-l}(\Lambda)$'s right branch contains only full edges), and putting a straight left NSUMT that is a decomposition of $A_{i_j}^* a_{i_j+1} \cdots A_{l'-2}^* a_{l'-1} A_{l'-1}^*$ as the left child (that exists since $\mathcal{RE}^{p-l'+1}(\Theta_j)$'s left branch contains only full edges).

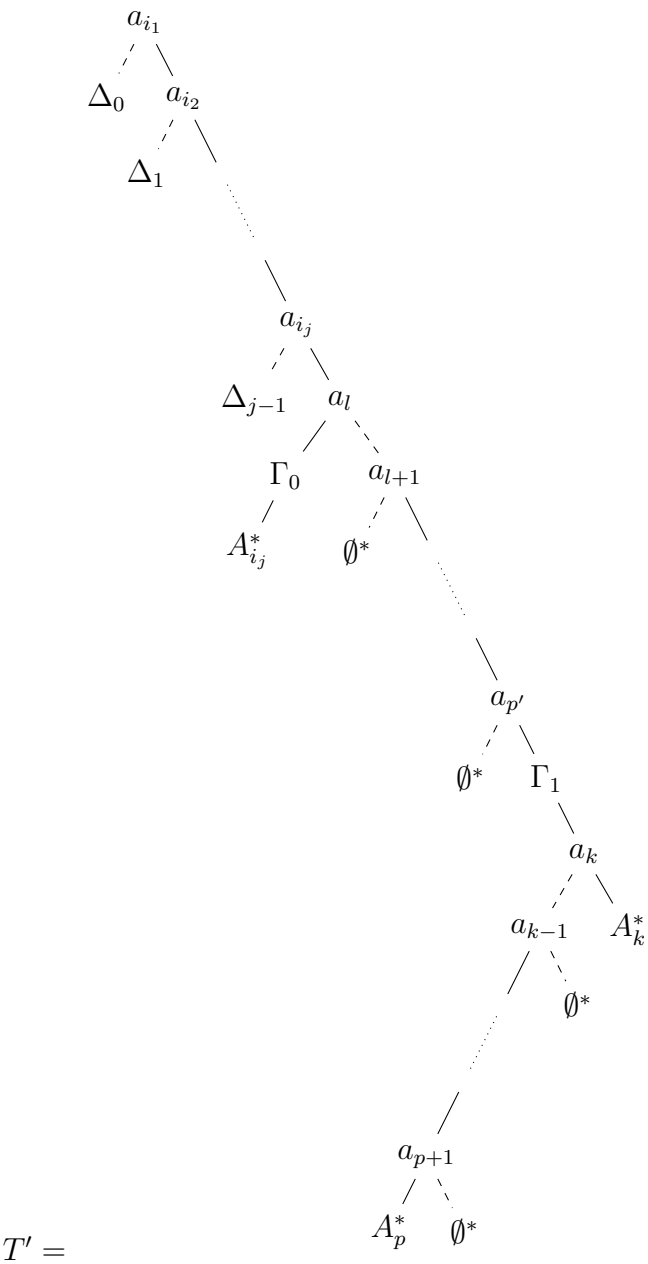
  Let now $p' \in [\![l, p]\!]$ be the smallest integer in $[\![l, p]\!]$ such that $A_{p'} \neq \emptyset$, which

necessarily exists since $A_p \neq \emptyset$. Then, by Lemma 5.4.10, $\mathcal{L}(\Theta_j)$ can also be decomposed as follows



where Γ_1 is a straight right NSUMT over Σ whose rightmost leaf has been suppressed (and hence which might be empty) and the edges between the nodes labelled a_{l+1} to $a_{p'}$ are all full.

Let



$T' =$

where the edges between the nodes labelled a_{i_1} to a_{i_j}, a_{l+1} to $a_{p'}$ and a_{p+1} to a_{k-1} are all full, using the convention that T' is rooted at a_l when $j = 0$. It is also a SUMT that is a decomposition of L because $\{a_{p+1}, \ldots, a_k\} \subseteq A_k$ and $a_l \notin A_p \cup \{a_k\}$, so that $a_l \notin A_k \subseteq A_p \cup \{a_k\}$. When $j > 0$, the major and minor weights of T' are both equal to -1 while at least one of them is positive for T. Further, when $j = 0$, when at least one of $A_1, \ldots, A_{l-1}$ is non-empty or it isn't the case but $p' < p$, then, respectively, the major weight of T' is at most $k - 2$ while that of T is $k - 1$, or the minor weight of T' is at most

$k-2$ while that of T is $k-1$ (because, by maximality of l, there can only be one node on the left branch of the left subtree of the root of T whose right subtree has at least one leaf not labelled by $\emptyset^*$), the major weight being the same. Finally, when $j=0$, when $A_1,\ldots,A_{l-1}$ are all empty and $p'=p$ but the constraint level of T' is 0, we have that its constraint level is 0 while that of T is 1 because $A_1=\cdots=A_{p-1}=A_{p+1}=\cdots=A_{k-1}=\emptyset$ and $a_l\notin A_k$. So, as L can be decomposed as the straight right NSUMT T, by inductive hypothesis, the Boolean combination $\varphi(T')$ of Zouave languages is right-convenient, as well as left-convenient when L can be decomposed as a straight left NSUMT, and verifies $\mathcal{L}(\varphi(T'))=\mathcal{L}(T')=L$. Therefore,

$$\varphi(T)=\varphi(T')$$

is the desired Boolean combination of Zouave languages.

The last case to examine is hence when $j=0$, $A_1=\cdots=A_{p-1}=A_{p+1}=\cdots=A_{k-1}=\emptyset$, $\{a_1,\ldots,a_p\}\subseteq A_0$ and $A_0\cap A_p\neq\emptyset$, the only way for the constraint level of T' to be greater than 0 (1 exactly), because $a_k\notin A_0$. Lemma 5.4.22 tells us that, since $A_1=\cdots=A_{l-1}=\emptyset$ and A_0 contains at least two elements, a_l and some element in $A_0\cap A_p$ different from a_l, we necessarily have that $l=1$. Moreover, we also have the following:

- if $p>1$, then there exists $e\in\Sigma$ verifying $a_2=\cdots=a_p=e$ and $A_0\cap A_p=\{e\}$;
- $A_0\subseteq A_p\cup\{a_1\}$;

besides the following:

- if $p<k-1$, then there exists $d\in\Sigma$ verifying $a_{p+1}=\cdots=a_{k-1}=d$ and $A_p\cap A_k=\{d\}$;
- $A_k\subseteq A_p\cup\{a_k\}$.

So in this very last case, we have

$$L=\begin{cases}A_0^*a_1e^{p-1}A_p^*d^{k-1-p}a_kA_k^* & \text{when } 1<p<k-1\\ A_0^*a_1e^{p-1}A_p^*a_kA_k^* & \text{when } 1<p=k-1\\ A_0^*a_1A_p^*d^{k-1-p}a_kA_k^* & \text{when } 1=p<k-1\\ A_0^*a_1A_p^*a_kA_k^* & \text{when } 1=p=k-1\ .\end{cases}$$

As in the previous subsubcases, we can show that if we let

$$K_1 = \begin{cases} \bigcap_{i=0}^{k-2-p} \bigcap_{c\in A_p\setminus\{d\}} \big((\Sigma\setminus\{a_k\})^* \\ \qquad\qquad cd^ia_k(\Sigma\setminus\{c\})^*\big)^{\complement} & \text{when } p < k-1 \\ \Sigma^* & \text{otherwise } (p = k-1)\ , \end{cases}$$

$$K_0 = \begin{cases} \bigcap_{i=0}^{p-2} \bigcap_{c\in A_p\setminus\{e\}} \big((\Sigma\setminus\{c\})^*a_1e^ic(\Sigma\setminus\{a_1\})^*\big)^{\complement} & \text{when } 1 < p \\ \Sigma^* & \text{otherwise } (1 = p)\ , \end{cases}$$

and

$$K = \begin{cases} \big((\Sigma\setminus\{a_k\})^*a_1A_p^{\leq k-3}a_k(\Sigma\setminus\{a_1\})^*\big)^{\complement} & \text{when } 1 < k-1 \\ \Sigma^* & \text{otherwise } (1 = k-1)\ , \end{cases}$$

we have

$$\begin{aligned} L =& A_0^*a_1A_p^*a_kA_k^* \cap K_1 \cap K_0 \cap K \\ =& (\Sigma\setminus\{a_k\})^*a_kA_k^* \cap A_0^*a_1(\Sigma\setminus\{a_1\})^* \cap (A_p\cup\{a_1,a_k\})^* \cap K_1 \cap K_0 \cap K\ , \end{aligned}$$

because $A_0^*a_1A_p^* \subseteq (\Sigma\setminus\{a_k\})^*$ and $A_p^*a_kA_k^* \subseteq (\Sigma\setminus\{a_1\})^*$.

By inductive hypothesis, we have that the Boolean combinations $\varphi(A_0^*)$ and $\varphi\big((A_p\cup\{a_1,a_k\})^*\big)$ of Zouave languages are both left- and right-convenient and verify, respectively, $\mathcal{L}(\varphi(A_0^*)) = A_0^*$ and $\mathcal{L}\big(\varphi\big((A_p\cup\{a_1,a_k\})^*\big)\big) = (A_p\cup\{a_1,a_k\})^*$.

By Lemma 5.4.11 (and its reversed version), and since

- when $p < k-1$, for each $i \in [\![0, k-2-p]\!]$ and $c \in A_p\setminus\{d\}$, no word of $A_0^*a_1A_p^*a_kA_k^*$ has $(cd^ia_k)^2$ as a subword,
- when $1 < p$, for each $i \in [\![0, p-2]\!]$ and $c \in A_p\setminus\{e\}$, no word of $A_0^*a_1A_p^*a_kA_k^*$ has $(a_1e^ic)^2$ as a subword,
- when $1 < k-1$, for each $u \in A_p^{k-3}$, no word of $A_0^*a_1A_p^*a_kA_k^*$ has $(a_1ua_k)^2$ as a subword,

if we let

$$\psi_1 = \begin{cases} \cap_{i=0}^{k-2-p} \cap_{c\in A_p\setminus\{d\}} \left[cd^ia_k\right]_{\Sigma,2}{}^{\complement} & \text{when } p < k-1 \\ \Sigma^* & \text{otherwise } (p = k-1)\ , \end{cases}$$

$$\psi_0 = \begin{cases} \cap_{i=0}^{p-2} \cap_{c \in A_p \setminus \{e\}} [a_1 e^i c]_{\Sigma,2}{}^{\complement} & \text{when } 1 < p \\ \Sigma^* & \text{otherwise } (1 = p) \end{cases}$$

and

$$\psi = \begin{cases} \cap_{u \in A_p^{\leqslant k-3}} [a_1 u a_k]_{\Sigma,2}{}^{\complement} & \text{when } 1 < k-1 \\ \Sigma^* & \text{otherwise } (1 = k-1)\ , \end{cases}$$

we have that

$$\begin{aligned} \varphi(T) = & [a_k]_{\Sigma,2} \cap [a_k]_{\Sigma,2} * \varphi(A_k^*) \cap [a_1]_{\Sigma,2} \cap \varphi(A_0^*) * [a_1]_{\Sigma,2} \cap \\ & \varphi\big((A_p \cup \{a_1, a_k\})^*\big) \cap \psi_1 \cap \psi_0 \cap \psi \end{aligned}$$

is the desired Boolean combination of Zouave languages.

This eventually concludes the proof of the proposition. □

5.4.5 Technical lemmata needed for the main proof

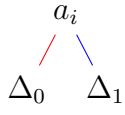
As announced before giving the main proof of Proposition 5.4.2, we now state and prove all the technical lemmata used in that proof. The only proof that is quite tricky is the one of Lemma 5.4.22, that gives the exact characterisation of NSUMTs for languages of dot-depth one.

Using the equational characterisation of $\mathbf{J} * \mathbf{D}$, we can trivially show the following.

Lemma 5.4.19. *Any SUM A^* over an alphabet Σ is in $\mathcal{L}(\mathbf{J} * \mathbf{D})$.*

Given a SUM that belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ and a SUMT that is a decomposition of it, we show that any SUM that can actually be decomposed as a subtree of that SUMT does also belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$, a result used several times in the proof of Proposition 5.4.2.

Lemma 5.4.20. *Let $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ ($k \in \mathbb{N}_{>0}$) be a SUM over an alphabet Σ in $\mathcal{L}(\mathbf{J} * \mathbf{D})$ and*



some decomposition of it where $i \in [k]$, Δ_0, Δ_1 are SUMTs over Σ, and the red edge is either full or dashed while the blue one is of the opposite type. Then, $\mathcal{L}(\Delta_0)$ and $\mathcal{L}(\Delta_1)$ are both also SUMs over Σ in $\mathcal{L}(\mathbf{J} * \mathbf{D})$.

Proof. Let $L = A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$.

Assume that the red edge is dashed while the blue one is full. We have that for any $w \in (\Sigma \setminus \{a_i\})^*$, $w \in \mathcal{L}(\Delta_0)$ if and only if $wa_i \cdots a_k \in L$. Similarly, for any $w \in \Sigma^*$, $w \in \mathcal{L}(\Delta_1)$ if and only if $a_1 \cdots a_i w \in L$. This means that $\mathcal{L}(\Delta_0) = L(a_i \cdots a_k)^{-1} \cap (\Sigma \setminus \{a_i\})^*$ and $\mathcal{L}(\Delta_1) = (a_1 \cdots a_i)^{-1} L$. As $\mathcal{L}(\mathbf{J} * \mathbf{D})$ is an *ne*-variety of languages and both L and $(\Sigma \setminus \{a_i\})^*$ belong to it (by virtue of Lemma 5.4.19), $\mathcal{L}(\Delta_0)$ and $\mathcal{L}(\Delta_1)$ do also belong to it.

When the red edge is full and the blue one is dashed, everything goes through in the symmetric way. □

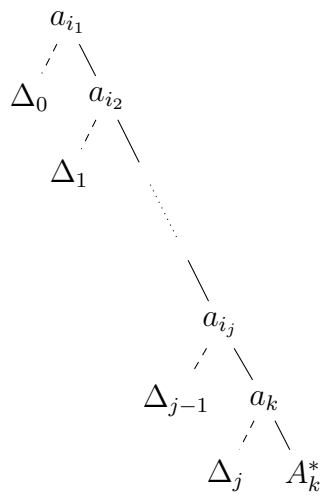
Corollary 5.4.21. *Let L be a SUM over an alphabet Σ in $\mathcal{L}(\mathbf{J} * \mathbf{D})$ and T a SUMT over Σ that is a decomposition of it. Then, for any SUMT Δ over Σ that is a subtree of T, $\mathcal{L}(\Delta)$ is also a SUM over Σ in $\mathcal{L}(\mathbf{J} * \mathbf{D})$.*

Proof sketch. Given a non-empty subtree Δ of T (that is a SUMT over Σ), simply apply Lemma 5.4.20 repeatedly following the unique path from the root ot T to the root of Δ. □

The fundamental lemma used in the proof of Proposition 5.4.2, intuitively presented at the end of Subsection 5.4.1, exactly characterises, given a SUM and an NSUMT that is a decomposition of it, what conditions this NSUMT should verify so as for the SUM to belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$. We now state it formally and eventually prove it.

Lemma 5.4.22. *Let $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ ($k \in \mathbb{N}$) be a SUM over an alphabet Σ and T an NSUMT over Σ that is a decomposition of it. Then, $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ is a language of $\mathcal{L}(\mathbf{J} * \mathbf{D})$ if and only if:*

- *T only consists of a root;*
- *when T is a straight right NSUMT of the form*



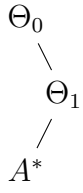
where $\Delta_0, \Delta_1, \ldots, \Delta_{j-1}, \Delta_j$ *(*$j \in \mathbb{N}$*) are straight left NSUMTs over* Σ *and the edges between the nodes labelled* a_{i_1} *to* a_{i_j} *are all full, we have that*

$$\mathcal{L}(\Delta_0)a_{i_1}\mathcal{L}(\Delta_1)a_{i_2}\cdots\mathcal{L}(\Delta_{j-1})a_{i_j}\mathcal{L}(\Delta_j) \in \mathcal{L}(\mathbf{J} * \mathbf{D})$$

and for all $p \in [\![0, k-1]\!]$*, using the convention that* $i_0 = 0$ *and* $i_{j+1} = k$*,* $A_p \cap A_k \neq \emptyset$ *implies that* $\{a_{p+1}, \ldots, a_k\} \nsubseteq A_k$ *or, if we take* $q \in [j+1]$ *to be the unique integer in* $[j+1]$ *verifying* $i_{q-1} \leq p < i_q$*, the following conditions are verified:*

- *for all* $\iota \in [\![q, j]\!]$*, there exists* $B_\iota \subseteq \Sigma$ *such that* $\Delta_\iota = B_\iota^*$ *and* $A_k \subseteq B_\iota \cup \{a_{i_{\iota+1}}\}$*;*
- *if* $p + 1 < i_q$*, then there exists* $b \in \Sigma$ *verifying* $a_{p+1} = \cdots = a_{i_q - 1} = b$ *and* $A_p \cap A_k = \{b\}$*;*
- $A_k \subseteq A_p \cup \{a_{i_q}\}$*.*

- *when* T *is a straight left NSUMT, the symmetric conditions are verified.*
- *when* T *is a bent right NSUMT of the form*



where

$$\Delta_0 = \begin{array}{c} \Theta_0 \\ \diagdown \\ A^* \end{array}$$

is a straight right NSUMT and

$$\Delta_1 = \begin{array}{c} \Theta_1 \\ \diagup \\ A^* \end{array}$$

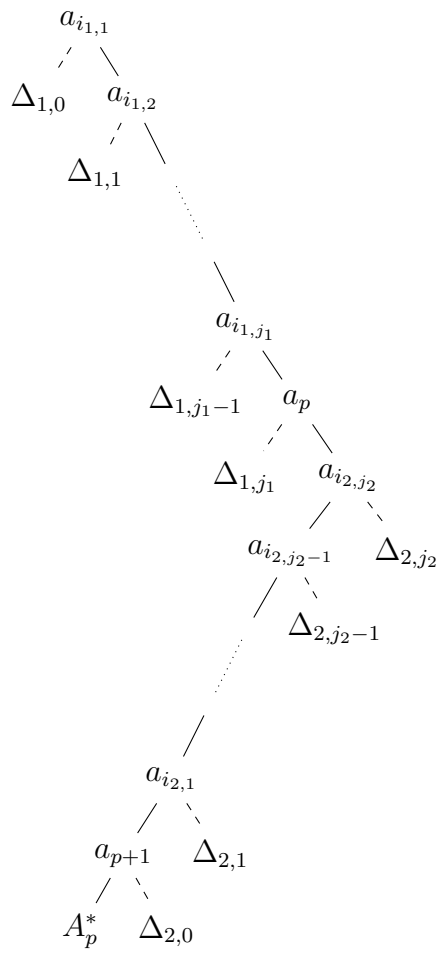
is a straight left NSUMT, $\mathcal{L}(\Delta_0)$ *and* $\mathcal{L}(\Delta_1)$ *both belong to* $\mathcal{L}(\mathbf{J} * \mathbf{D})$*;*

- *when* T *is a bent left NSUMT, the symmetric conditions are verified.*

Proof. Let $L = A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ ($k \in \mathbb{N}$) be a SUM over an alphabet Σ and T an NSUMT over Σ that is a decomposition of it. We consider each of the possible cases of the statement, one after the other.

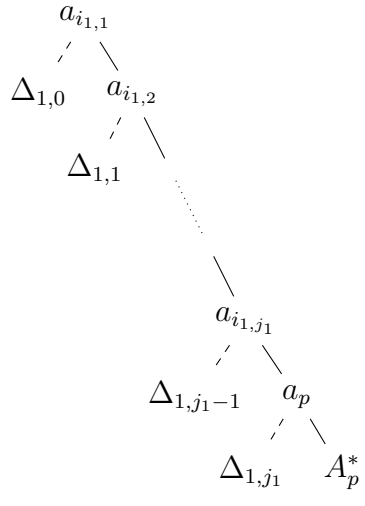
Case 1. Assume first that T only consists of a root. Then, it is trivial to see that $L \in \mathcal{L}(\mathbf{J} * \mathbf{D})$.

Case 2. Assume now that T is a bent right NSUMT of the form

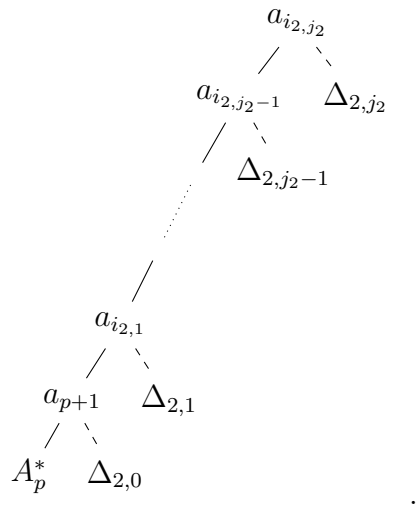


where $\Delta_{1,0}, \Delta_{1,1}, \ldots, \Delta_{1,j_1-1}, \Delta_{1,j_1}$ ($j_1 \in \mathbb{N}$) are straight left NSUMTs over Σ, $\Delta_{2,0}, \Delta_{2,1}$, $\ldots, \Delta_{2,j_2-1}, \Delta_{2,j_2}$ ($j_2 \in \mathbb{N}$) are straight right NSUMTs over Σ, $p \in [\![1, k-1]\!]$ and the edges between the nodes labelled $a_{i_{1,1}}$ to $a_{i_{1,j_1}}$ and $a_{i_{2,1}}$ to $a_{i_{2,j_2}}$ are all full. We let

$\Delta_0 =$



and

$\Delta_1 =$ 

.

We also let $L_0 = \mathcal{L}(\Delta_0)$ and $L_1 = \mathcal{L}(\Delta_1)$. Our goal now is to prove that $L \in \mathcal{L}(\mathbf{J} * \mathbf{D})$ if and only if $L_0, L_1 \in \mathcal{L}(\mathbf{J} * \mathbf{D})$.

Suppose first that $L \in \mathcal{L}(\mathbf{J} * \mathbf{D})$. We have that for any $w \in \Sigma^*$, $w \in L_0$ if and only if $wa_{p+1} \cdots a_k \in L$. Similarly, for any $w \in \Sigma^*$, $w \in L_1$ if and only if $a_1 \cdots a_p w \in L$. This means that $L_0 = L(a_{p+1} \cdots a_k)^{-1}$ and $L_1 = (a_1 \cdots a_p)^{-1} L$. As $\mathcal{L}(\mathbf{J} * \mathbf{D})$ is an *ne*-variety of languages and L belongs to it, L_0 and L_1 do also belong to it.

Suppose now that $L_0, L_1 \in \mathcal{L}(\mathbf{J} * \mathbf{D})$. We shall prove that L belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ using the equational characterisation of $\mathbf{J} * \mathbf{D}$. Let η_L, η_{L_0} and η_{L_1} be the syntactic morphisms of L, L_0 and L_1 respectively and $\sim_L$, $\sim_{L_0}$ and $\sim_{L_1}$ the syntactic congruences of L, L_0 and L_1 respectively. Let ω_L, ω_{L_0} and ω_{L_1} be the idempotent powers of $\mathrm{M}(L)$, $\mathrm{M}(L_0)$ and $\mathrm{M}(L_1)$ respectively (which are the syntactic monoids of L, L_0 and L_1 respectively).

Let $e, s, f, t, u, v \in \Sigma^+$ such that $\eta_L(e)$ and $\eta_L(f)$ are idempotents. Then

$$(esfte)^{\omega_L}(eufve)^{\omega_L} \sim_L (esfte)^{\omega_L} esfve (eufve)^{\omega_L}$$

if and only if

$$\begin{aligned}&(e^{\omega'} s f^{\omega'} t e^{\omega'})^{\omega_L \omega'} (e^{\omega'} u f^{\omega'} v e^{\omega'})^{\omega_L \omega'} \\ \sim_L &(e^{\omega'} s f^{\omega'} t e^{\omega'})^{\omega_L \omega'} e^{\omega'} s f^{\omega'} v e^{\omega'} (e^{\omega'} u f^{\omega'} v e^{\omega'})^{\omega_L \omega'} ,\end{aligned}$$

where $\omega' \in \mathbb{N}_{>0}$ is any multiple of both ω_{L_0} and ω_{L_1}. We shall fix ω' more precisely later on. We set $\omega = \omega_L \omega'$, $e' = e^{\omega'}$ and $f' = f^{\omega'}$. Note that then, $\eta_{L_0}(e')$ and $\eta_{L_0}(f')$ as well as $\eta_{L_1}(e')$ and $\eta_{L_1}(f')$ are idempotents and ω is a multiple of both ω_{L_0} and ω_{L_1}. Let

$\alpha, \beta, \gamma \in \Sigma^+$ such that $\alpha = e'sf'te'$, $\beta = e'uf've'$ and $\gamma = e'sf've'$. We are now going to prove that $\alpha^\omega\beta^\omega \sim_L \alpha^\omega\gamma\beta^\omega$.

Let $x, y \in \Sigma^*$ such that $w = x\alpha^\omega\beta^\omega y \in L$. Let $w' = x\alpha^\omega\gamma\beta^\omega y$; we want to prove that w' does also belong to L. We do this through a careful analysis of how w and w' can be decomposed according to T. Let $\iota \in [|w|]$ be the minimum integer in $[|w|]$ such that $w_1 \cdots w_\iota$ contains $a_{i_{1,1}} \cdots a_{i_{1,j_1}} a_p$ as a subword, and $\iota' \in [|w'|]$ the minimum integer in $[|w'|]$ such that $w'_1 \cdots w'_{\iota'}$ contains $a_{i_{1,1}} \cdots a_{i_{1,j_1}} a_p$ as a subword. Let $\delta \in [|w|]$ be the maximum integer in $[|w|]$ such that $w_\delta \cdots w_{|w|}$ contains $a_{p+1} a_{i_{2,1}} \cdots a_{i_{2,j_2}}$ as a subword, and $\delta' \in [|w'|]$ the maximum integer in $[|w'|]$ such that $w'_{\delta'} \cdots w'_{|w'|}$ contains $a_{p+1} a_{i_{2,1}} \cdots a_{i_{2,j_2}}$ as a subword. We now distinguish between four different cases.

- If ι corresponds to a position in the factor x of w, say position $\kappa \in [|x|]$, then $\iota = \iota'$ and we have that $x_1 \cdots x_\kappa \in L_0$ and $x_{\kappa+1} \cdots x_{|x|}\alpha^\omega\beta^\omega y \in L_1$. But since $\eta_{L_1}(e')$ and $\eta_{L_1}(f')$ are idempotents and ω is a multiple of ω_{L_1}, we have $w'_{\iota'+1} \cdots w'_{|w'|} = x_{\kappa+1} \cdots x_{|x|}\alpha^\omega\gamma\beta^\omega y \in L_1$. Hence, $x\alpha^\omega\gamma\beta^\omega y \in L$, because $w'_1 \cdots w'_{\iota'} = x_1 \cdots x_\kappa \in L_0$.

- If ι corresponds to a position in one of the factors α of w, then taking ω' to be a big enough multiple of ω_{L_0} and ω_{L_1} verifying $\omega > j_1 + j_2 + 2$, this position is in one of the $j_1 + 1$ first factors α of w by minimality and we have $\iota = \iota'$. Moreover, we also have that δ corresponds to a position that is at least in one of the $j_2 + 1$ last factors α of w by maximality, which implies that A_p contains all letters appearing in α, so as $w_{\iota+1} \cdots w_{|w|} \in L_1$, it follows that $\alpha^\omega\beta^\omega y \in L_1$. But since $\eta_{L_1}(e')$ and $\eta_{L_1}(f')$ are idempotents and ω is a multiple of ω_{L_1}, we have $\alpha^\omega\gamma\beta^\omega y \in L_1$. From what we know about δ, we can deduce that δ' also corresponds to a position that is at least in one of the $j_2 + 1$ last factors α of w' by maximality, hence we get that $w'_{\iota'+1} \cdots w'_{|w'|}$ must belong to L_1 for $\alpha^\omega\gamma\beta^\omega y$ to belong to L_1, as $\iota' < \delta'$. Therefore, since $w'_1 \cdots w'_{\iota'} \in L_0$, we have that $x\alpha^\omega\gamma\beta^\omega y \in L$.

- If δ corresponds to a position in one of the factors β of w, we can prove symmetrically to the previous case that we have $x\alpha^\omega\gamma\beta^\omega y \in L$.

- If δ corresponds to a position in the factor y of w, we can prove symmetrically to the first case that we have $x\alpha^\omega\gamma\beta^\omega y \in L$.

There are no other cases to consider, since when ι correponds to a position in one of the factors β of w or in the factor y of w, then δ necessarily corresponds to a position in one of the factors β of w or in the factor y of w.

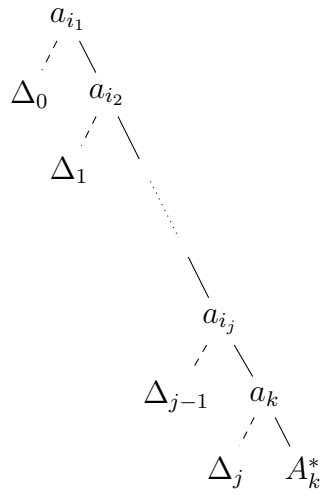
Let $x, y \in \Sigma^*$ such that $x\alpha^\omega\gamma\beta^\omega y \in L$. In a way similar to above, we can show that then, $x\alpha^\omega\beta^\omega y \in L$.

This finishes to show that

$$(esfte)^{\omega_L}(eufve)^{\omega_L} \sim_L (esfte)^{\omega_L} esfve(eufve)^{\omega_L} \ .$$

Since this holds for any $e, s, f, t, u, v \in \Sigma^+$ such that $\eta_L(e)$ and $\eta_L(f)$ are idempotents, it follows that $L \in \mathcal{L}(\mathbf{J} * \mathbf{D})$.

Case 3. Assume now that T is a straight right NSUMT of the form



where $\Delta_0, \Delta_1, \ldots, \Delta_{j-1}, \Delta_j$ $(j \in \mathbb{N})$ are straight left NSUMTs over Σ and the edges between the nodes labelled a_{i_1} to a_{i_j} are all full. We let

$$L' = \mathcal{L}(\Delta_0)a_{i_1}\mathcal{L}(\Delta_1)a_{i_2}\cdots\mathcal{L}(\Delta_{j-1})a_{i_j}\mathcal{L}(\Delta_j) \ .$$

Let η_L and $\eta_{L'}$ be the syntactic morphisms of L and L' respectively and $\sim_L$ and $\sim_{L'}$ the syntactic congruences of L and L' respectively. Let ω_L and $\omega_{L'}$ be the idempotent powers of $\mathrm{M}(L)$ and $\mathrm{M}(L')$ respectively (which are the syntactic monoids of L and L' respectively). Our goal is to prove the necessity and then the sufficiency of the conditions given in the statement of the lemma for L to belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$.

Necessity of the conditions. Suppose first that $L \in \mathcal{L}(\mathbf{J} * \mathbf{D})$.

We shall first prove that L' belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ using the equational characterisation

of $\mathbf{J} * \mathbf{D}$. Let $e, s, f, t, u, v \in \Sigma^+$ such that $\eta_{L'}(e)$ and $\eta_{L'}(f)$ are idempotents. Then

$$(esfte)^{\omega_{L'}}(eufve)^{\omega_{L'}} \sim_{L'} (esfte)^{\omega_{L'}} esfve(eufve)^{\omega_{L'}}$$

if and only if

$$\begin{aligned}&(e^{\omega'}sf^{\omega'}te^{\omega'})^{\omega_{L'}\omega'}(e^{\omega'}uf^{\omega'}ve^{\omega'})^{\omega_{L'}\omega'}\\ \sim_{L'}&(e^{\omega'}sf^{\omega'}te^{\omega'})^{\omega_{L'}\omega'}e^{\omega'}sf^{\omega'}ve^{\omega'}(e^{\omega'}uf^{\omega'}ve^{\omega'})^{\omega_{L'}\omega'}\ ,\end{aligned}$$

where $\omega' \in \mathbb{N}_{>0}$ is any multiple of ω_L. We shall fix ω' more precisely later on.

We set $\omega = \omega_{L'}\omega'$, $e' = e^{\omega'}$ and $f' = f^{\omega'}$. Since $\eta_L(e')$ and $\eta_L(f')$ are idempotents and ω is a multiple of ω_L, we have

$$(e'sf'te')^{\omega}(e'uf've')^{\omega} \sim_L (e'sf'te')^{\omega}e'sf've'(e'uf've')^{\omega}\ .$$

Let $\alpha, \beta, \gamma \in \Sigma^+$ such that $\alpha = e'sf'te'$, $\beta = e'uf've'$ and $\gamma = e'sf've'$. We are now going to prove that $\alpha^{\omega}\beta^{\omega} \sim_{L'} \alpha^{\omega}\gamma\beta^{\omega}$.

Let $x, y \in \Sigma^*$ such that $w = x\alpha^{\omega}\beta^{\omega}y \in L'$. Then, $x\alpha^{\omega}\beta^{\omega}ya_k \in L$ and so $w' = x\alpha^{\omega}\gamma\beta^{\omega}ya_k \in L$. We want to prove that $x\alpha^{\omega}\gamma\beta^{\omega}y$ does belong to L'; we do this through a careful analysis of how w and w' can be decomposed according to T. Let $\iota \in [|w|]$ be the minimum integer in $[|w|]$ such that $w_1 \cdots w_\iota$ contains $a_{i_1} \cdots a_{i_j}$ as a subword, and ι' the minimum integer in $[|w'|]$ such that $w'_1 \cdots w'_{\iota'}$ contains $a_{i_1} \cdots a_{i_j}$ as a subword. There are several cases to examine, but in any of them, we can prove that $|w'|$ is the smallest integer δ in $[|w'|]$ such that $w'_1 \cdots w'_\delta$ contains $a_{i_1} \cdots a_{i_j}a_k$ as a subword, which implies that $x\alpha^{\omega}\gamma\beta^{\omega}y = w'_1 \cdots w'_{|w'|-1} \in L'$, as $w' \in L$.

- If ι corresponds to a position in the factor x of w, then $\iota = \iota'$ and since $w_{\iota+1} \cdots w_{|w|} \in \mathcal{L}(\Delta_j)$, we in particular have that all letters in x appearing after position ι, all letters appearing in α, β, y and hence γ are different from a_k. Therefore, we reach the aforementioned conclusion.

- If ι corresponds to a position in one of the factors α of w, then taking ω' to be a big enough multiple of ω_L verifying $\omega > j$, this position is in one of the j first factors α of w by minimality and we have $\iota = \iota'$ and, since $w_{\iota+1} \cdots w_{|w|} \in \mathcal{L}(\Delta_j)$, that all letters appearing in α, β, y and hence γ are different from a_k. Therefore, we reach the aforementioned conclusion.

- If ι corresponds to a position in one of the factors β of w, then taking ω' to be a

big enough multiple of ω_L verifying $\omega > j$, ι' corresponds to a position in the factor ve' of γ of w' or one of the factors β of w' by minimality (otherwise, the minimality condition would be violated as ι would correspond to a position in one of the factors α of w). The position given by ι must be in one of the j first factors β of w by minimality, so we moreover have, since $w_{\iota+1} \cdots w_{|w|} \in \mathcal{L}(\Delta_j)$, that all letters appearing in β, y and hence in ve' are different from a_k. Therefore, we reach the aforementioned conclusion.

- If ι corresponds to a position in the factor y of w, say position $\kappa \in [|y|]$, then ι' also corresponds to position κ in the factor y of w', otherwise it would violate minimality of ι when we take ω' to be a big enough multiple of ω_L verifying $\omega > j$. This means that $y_{\kappa+1} \cdots y_{|y|} \in \mathcal{L}(\Delta_j)$, so that $y_{\kappa+1} \cdots y_{|y|}$ does not contain the letter a_k. Therefore, we reach the aforementioned conclusion.

Let $x, y \in \Sigma^*$ such that $x\alpha^\omega\gamma\beta^\omega y \in L'$. In a way similar to above, we can show that then, $x\alpha^\omega\beta^\omega y \in L'$.

This finishes to show that

$$(esfte)^{\omega_{L'}}(eufve)^{\omega_{L'}} \sim_{L'} (esfte)^{\omega_{L'}} esfve (eufve)^{\omega_{L'}} \ .$$

Since this holds for any $e, s, f, t, u, v \in \Sigma^+$ such that $\eta_{L'}(e)$ and $\eta_{L'}(f)$ are idempotents, it follows that $L' \in \mathcal{L}(\mathbf{J} * \mathbf{D})$.

We now prove that the technical condition for each $p \in [\![0, k-1]\!]$ given in the lemma's statement is verified.

Let $p \in [\![0, k-1]\!]$, let us use the convention that $i_0 = 0$ and $i_{j+1} = k$ and take $q \in [j+1]$ to be the unique integer in $[j+1]$ verifying $i_{q-1} \leq p < i_q$. Assume that $A_p \cap A_k \neq \emptyset$ and $\{a_{p+1}, \ldots, a_k\} \subseteq A_k$. We have to check that three properties are verified.

The first property says that for all $\iota \in [\![q, j]\!]$, there exists $B_\iota \subseteq \Sigma$ such that $\Delta_\iota = B_\iota^*$ and $A_k \subseteq B_\iota \cup \{a_{i_{\iota+1}}\}$. Let $b \in A_p \cap A_k$ and for each $\iota \in [\![q, j]\!]$ some $w_\iota \in (A_k \setminus \{a_{i_{\iota+1}}\})^*$. We have

$$\begin{aligned} &a_1 \cdots a_p (b^{\omega_L} a_{p+1} \cdots a_{i_q} \cdots a_k b^{\omega_L} a_{p+1} \cdots a_{i_q} w_q a_{i_{q+1}} w_{q+1} \cdots a_{i_j} w_j a_k b^{\omega_L})^{\omega_L} \\ \sim_L &a_1 \cdots a_p (b^{\omega_L} a_{p+1} \cdots a_{i_q} w_q a_{i_{q+1}} w_{q+1} \cdots a_{i_j} w_j a_k b^{\omega_L} a_{p+1} \cdots a_{i_q} \cdots a_k b^{\omega_L})^{\omega_L} \end{aligned}$$

by virtue of the fact that $L \in \mathcal{L}(\mathbf{J} * \mathbf{D})$. Since the first word belongs to L, the second does also, so we can conclude that $w_\iota \in \mathcal{L}(\Delta_\iota)$ for all $\iota \in [\![q, j]\!]$. Therefore, as this is true taking,

for each $\iota \in [\![q, j]\!]$, any $w_\iota \in (A_k \setminus \{a_{i_{\iota+1}}\})^*$ and in particular the empty word ε, we get that for each $\iota \in [\![q, j]\!]$, Δ_ι must be restricted to a sole root (because $\mathcal{L}(\Delta_\iota)$ contains the empty word) verifying that there exists $B_\iota \subseteq \Sigma$ such that $\Delta_\iota = B_\iota^*$ and $A_k \subseteq B_\iota \cup \{a_{i_{\iota+1}}\}$.

The second property says that if $p + 1 < i_q$, we have $a_{p+1} = \cdots = a_{i_q-1} = b$ and $A_p \cap A_k = \{b\}$. Assume that $p + 1 < i_q$ and suppose there would exist $d \in (A_p \cap A_k) \setminus (\{a_{p+1}\} \cap \cdots \cap \{a_{i_q-1}\})$. We would have

$$a_1 \cdots a_p (d^{\omega_L} a_{p+1} \cdots a_k d^{\omega_L} a_{i_q} \cdots a_k d^{\omega_L})^{\omega_L} \sim_L a_1 \cdots a_p (d^{\omega_L} a_{i_q} \cdots a_k d^{\omega_L} a_{p+1} \cdots a_k d^{\omega_L})^{\omega_L}$$

by virtue of the fact that $L \in \mathcal{L}(\mathbf{J} * \mathbf{D})$. Nevertheless, the first word belongs to L while the second one does not (because since $p + 1 < i_q$, there is at least one letter among $a_{p+1}, \ldots, a_{i_q-1}$ that is different from d and should appear between the first factor $a_1 \cdots a_p$ and the first letter a_{i_q} after it in the second word, which is not the case): contradiction. Hence, as $b \in A_p \cap A_k$, we must have $a_{p+1} = \cdots = a_{i_q-1} = b$ and $A_p \cap A_k = \{b\}$.

The third and last property says that $A_k \subseteq A_p \cup \{a_{i_q}\}$. Suppose now there would exist $c \in A_k \setminus (A_p \cup \{a_{i_q}\})$. We would have

$$\begin{aligned} &a_1 \cdots a_p (b^{\omega_L} a_{p+1} \cdots a_k b^{\omega_L} b^\alpha c a_{p+1} \cdots a_k b^{\omega_L})^{\omega_L} \\ \sim_L &a_1 \cdots a_p (b^{\omega_L} b^\alpha c a_{p+1} \cdots a_k b^{\omega_L} a_{p+1} \cdots a_k b^{\omega_L})^{\omega_L} \end{aligned}$$

for $\alpha \in \mathbb{N}$ the smallest non-negative integer such that $\omega_L + \alpha \geq p - i_{q-1}$, by virtue of the fact that $L \in \mathcal{L}(\mathbf{J} * \mathbf{D})$. Nevertheless, the first word belongs to L while the second one does not: a contradiction. To see that the second word, that we will denote by u, does not belong to L, suppose for a contradiction that it does: this would necessarily mean that we cannot have $i_{q-1} = p = i_q - 1$, because $c \notin A_p$. Then, there would only be two possibilities for u to belong to L. Either we would have that $p > i_{q-1}$ and in the unique decomposition of u according to T, the first occurrence of c would be the letter corresponding to the node labelled by the letter a_p in T. But then there would necessarily exist some $\iota \in [\![i_{q-1}, p-1]\!]$ verifying that $b \in A_\iota \cap A_k$ and $a_{\iota+1} = \cdots = a_{p-1} = b \in A_k$, so that since $a_p = c \in A_k$ and $\{a_{p+1}, \ldots, a_k\} \subseteq A_k$, we would have $a_p = b$ by what we have just proven above, which is not possible as $a_p = c \notin A_p$. Or we would have that $p+1 < i_q$ and in the unique decomposition of u according to T, the first occurrence of c would appear in the factor corresponding to the subtree of T corresponding to $a_{p+1} A_{p+1}^* \cdots a_{i_q-1} A_{i_q-1}^*$. Because then $a_{p+1} = \cdots = a_{i_q-1} = b$ by what we have proven just above, the first occurrence of c in that decomposition would in fact appear in a factor corresponding to one of the leaves labelled by $A_{p+1}^*, \ldots, A_{i_q-1}^*$ respectively in T. But then there would

necessarily exist some $\iota \in [\![p+1, i_q - 1]\!]$ verifying that $b \in A_\iota$, contradicting the fact that the word $a_1 \cdots a_p b^{\iota - p+1} a_{\iota+1} \cdots a_k \in L$ has a unique decomposition $w_0 a_1 w_1 a_2 \cdots w_{k-1} a_k w_k$ where $w_i \in A_i^*$ for all $i \in [\![0, k]\!]$. Therefore, u cannot belong to L. Hence, to conclude, $A_k \subseteq A_p \cup \{a_{i_q}\}$.

Therefore, all conditions listed in the lemma's statement are verified.

Sufficiency of the conditions. Now suppose that $L' \in \mathcal{L}(\mathbf{J} * \mathbf{D})$ and for all $p \in [\![0, k-1]\!]$, using the convention that $i_0 = 0$ and $i_{j+1} = k$, $A_p \cap A_k \neq \emptyset$ implies that $\{a_{p+1}, \ldots, a_k\} \not\subseteq A_k$ or, if we take $q \in [j+1]$ to be the unique integer in $[j+1]$ verifying $i_{q-1} \leq p < i_q$, the following conditions are verified:

- for all $\iota \in [\![q, j]\!]$, there exists $B_\iota \subseteq \Sigma$ such that $\Delta_\iota = B_\iota^*$ and $A_k \subseteq B_\iota \cup \{a_{i_{\iota+1}}\}$;
- if $p + 1 < i_q$, then there exists $b \in \Sigma$ verifying $a_{p+1} = \cdots = a_{i_q - 1} = b$ and $A_p \cap A_k = \{b\}$;
- $A_k \subseteq A_p \cup \{a_{i_q}\}$.

We shall prove that L belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ using the equational characterisation of $\mathbf{J} * \mathbf{D}$. Let $e, s, f, t, u, v \in \Sigma^+$ such that $\eta_L(e)$ and $\eta_L(f)$ are idempotents. Then

$$(esfte)^{\omega_L}(eufve)^{\omega_L} \sim_L (esfte)^{\omega_L} esfve (eufve)^{\omega_L}$$

if and only if

$$\begin{aligned}&(e^{\omega'} s f^{\omega'} t e^{\omega'})^{\omega_L \omega'} (e^{\omega'} u f^{\omega'} v e^{\omega'})^{\omega_L \omega'} \\ \sim_L &(e^{\omega'} s f^{\omega'} t e^{\omega'})^{\omega_L \omega'} e^{\omega'} s f^{\omega'} v e^{\omega'} (e^{\omega'} u f^{\omega'} v e^{\omega'})^{\omega_L \omega'} \ ,\end{aligned}$$

where $\omega' \in \mathbb{N}_{>0}$ is any multiple of $\omega_{L'}$. We shall fix ω' more precisely later on.

We set $\omega = \omega_L \omega'$, $e' = e^{\omega'}$ and $f' = f^{\omega'}$. Let $\alpha, \beta, \gamma \in \Sigma^+$ such that $\alpha = e' s f' t e'$, $\beta = e' u f' v e'$ and $\gamma = e' s f' v e'$. We are now going to prove that $\alpha^\omega \beta^\omega \sim_L \alpha^\omega \gamma \beta^\omega$.

Let $x, y \in \Sigma^*$ such that $w = x \alpha^\omega \beta^\omega y \in L$. We also let $w' = x \alpha^\omega \gamma \beta^\omega y$; we want to prove that w' does also belong to L. We do this, as before in the present proof, through a careful analysis of how w and w' can be decomposed according to T. Let $\iota \in [|w|]$ be the minimum integer in $[|w|]$ such that $w_1 \cdots w_\iota$ contains $a_{i_1} \cdots a_{i_j} a_k$ as a subword, and ι' the minimum integer in $[|w'|]$ such that $w'_1 \cdots w'_{\iota'}$ contains $a_{i_1} \cdots a_{i_j} a_k$ as a subword. We now distinguish between four different cases; one of them is really tricky.

- If ι corresponds to a position in the factor x of w, then $\iota = \iota'$ and since $w_{\iota+1} \cdots w_{|w|} \in A_k^*$, we in particular have that all letters in x appearing after position ι, all letters appearing in α, β, y and hence γ belong to A_k. Therefore, $w'_{\iota'+1} \cdots w'_{|w'|} \in A_k^*$ and as $w'_1 \cdots w'_{\iota-1} = w_1 \cdots w_{\iota-1} \in L'$, we have that $x\alpha^\omega\gamma\beta^\omega y \in L$.

- If ι corresponds to a position in one of the factors α of w, then taking ω' to be a big enough multiple of $\omega_{L'}$ verifying $\omega > j+1$, this position is in one of the $j+1$ first factors α of w by minimality and we have $\iota = \iota'$ and, since $w_{\iota+1} \cdots w_{|w|} \in A_k^*$, that all letters appearing in α, β, y and hence γ belong to A_k. Therefore, $w'_{\iota'+1} \cdots w'_{|w'|} \in A_k^*$ and as $w'_1 \cdots w'_{\iota-1} = w_1 \cdots w_{\iota-1} \in L'$, we have that $x\alpha^\omega\gamma\beta^\omega y \in L$.

- If ι corresponds to a position in one of the factors β of w, then taking ω' to be a big enough multiple of $\omega_{L'}$ verifying $\omega > j+1$, ι' corresponds to a position in the factor ve' of γ of w' or one of the factors β of w' by minimality (otherwise, the minimality condition would be violated as ι would correspond to a position in one of the factors α of w). The position given by ι must be in one of the $j+1$ first factors β of w by minimality, so we moreover have, since $w_{\iota+1} \cdots w_{|w|} \in A_k^*$, that all letters appearing in β, y and hence in ve' belong to A_k.

  Let $q \in [j+1]$ be the smallest integer in $[j+1]$ such that $x\alpha^\omega$ does not contain $a_{i_1} \cdots a_{i_q}$ as a subword. Let now $\delta_0 \in [\![0, |w|]\!]$ be the minimum integer in $[|w|]$ such that $w_1 \cdots w_{\delta_0}$ contains $a_{i_1} \cdots a_{i_{q-1}}$ as a subword or 0 if $q-1=0$, and $\delta'_0 \in [\![0, |w'|]\!]$ the minimum integer in $[|w'|]$ such that $w'_1 \cdots w'_{\delta'_0}$ contains $a_{i_1} \cdots a_{i_{q-1}}$ as a subword or 0 if $q-1=0$. Similarly, let $\delta_1 \in [|w|]$ be the minimum integer in $[|w|]$ such that $w_1 \cdots w_{\delta_1}$ contains $a_{i_1} \cdots a_{i_q}$ as a subword, and $\delta'_1 \in [|w'|]$ the minimum integer in $[|w'|]$ such that $w'_1 \cdots w'_{\delta'_1}$ contains $a_{i_1} \cdots a_{i_q}$ as a subword. The strategy we follow to prove that $w' \in L$ is to prove that, on the one hand, $w'_1 \cdots w'_{\delta'_1-1} \in \mathcal{L}(\Delta_0)a_{i_1}\mathcal{L}(\Delta_1) \cdots a_{i_{q-1}}\mathcal{L}(\Delta_{q-1})$, and on the other hand, $w'_{\delta'_1+1} \cdots w'_{|w'|} \in \mathcal{L}(\Delta_q)a_{i_{q+1}} \cdots \mathcal{L}(\Delta_j)a_k A_k^*$. The idea is that we exploit our hypothesis that the conditions given in the statement of the lemma for L to belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ are true by:

  1. finding some $p \in [\![i_{q-1}, i_q - 1]\!]$ such that $A_p \cap A_k \neq \emptyset$ and $\{a_{p+1}, \ldots, a_k\} \subseteq A_k$, which implies that the three properties of the statement hold for p;
  2. showing that $x\alpha^\omega$ belongs to $A_0^* a_1 A_1^* \cdots a_p A_p^*$;
  3. concluding that $x\alpha^\omega(e'e'f'e'e')^\omega a_{i_q} \cdots a_{k-1}$ belongs to L' by the equational characterisation of $\mathbf{J} * \mathbf{D}$ and showing that this means there exists some

$\sigma \in [\![\delta'_0 + 1, \delta'_1 - i_q + p]\!]$ verifying that $\sigma + 1$ corresponds to a position in the factor f' of γ of w' and that $w'_1 \cdots w'_\sigma$ belongs to $A_0^* a_1 A_1^* \cdots a_p A_p^*$;

4. showing that $w'_{\sigma+1} \cdots w'_{\delta'_1-1}$ belongs to $A_p^* a_{p+1} A_{p+1}^* \cdots a_{i_q-1} A_{i_q-1}^*$, which finishes to prove that $w'_1 \cdots w'_{\delta'_1-1} \in \mathcal{L}(\Delta_0) a_{i_1} \mathcal{L}(\Delta_1) \cdots a_{i_{q-1}} \mathcal{L}(\Delta_{q-1})$;
5. showing eventually that $w'_{\delta'_1+1} \cdots w'_{|w'|} \in \mathcal{L}(\Delta_q) a_{i_{q+1}} \cdots \mathcal{L}(\Delta_j) a_k A_k^*$.

We now give the details.

Looking at the way q was defined and ω' was fixed, we can see that $\delta_0 = \delta'_0$ and, when $q-1 > 0$, that they both correspond to the same position in the factor x of w and w' respectively, or both correspond to the same position in one of the $q-1$ first factors α of w and w' respectively. Moreover, we can also see that δ_1 corresponds to a position in the first factor β of w and that δ'_1 corresponds to a position in the factor ve' of γ of w' or the first factor β of w'. Further, δ_1 corresponds to a position at least at a position in the first factor u of β^ω of w. Let $w = v_0 a_1 v_1 a_2 \cdots v_{k-1} a_k v_k$ where $v_\lambda \in A_\lambda^*$ for all $\lambda \in [\![0, k]\!]$ be the unique decomposition of w according to T. By construction, we have $w_{\delta_0} = a_{i_{q-1}}$ when $q - 1 > 0$, as well as $w_{\delta_1} = a_{i_q}$.

Step 1. Taking ω' to be a big enough multiple of $\omega_{L'}$ verifying $|e'| > k-1$ and $|f'| > k-1$, we first show there exists $p \in [\![i_{q-1}, i_q - 1]\!]$ such that $A_p \cap A_k \neq \emptyset$ and $\{a_{p+1}, \ldots, a_k\} \subseteq A_k$. We have that δ_1 corresponds to a position in the first factor β of w and all letters appearing in β and y do belong to A_k, so it follows that $a_{i_q} v_{i_q} \cdots a_k v_k = w_{\delta_1} \cdots w_{|w|} \in A_k^*$ and consequently that $\{a_{i_q}, \ldots, a_k\} \subseteq A_k$. Moreover, because δ_0, when different from 0, corresponds to a position in the factor $x\alpha^\omega$ of w and δ_1 corresponds to a position at least in the first factor u of β^ω of w, the word $v_{i_{q-1}} a_{i_{q-1}+1} v_{i_{q-1}+1} \cdots a_{i_q-1} v_{i_q-1} = w_{\delta_0+1} \cdots w_{\delta_1-1}$ is guaranteed to have as a suffix a prefix of β of length at least $k \geq i_q - i_{q-1}$, that contains only letters of A_k, so that there must exist some $p \in [\![i_{q-1}, i_q - 1]\!]$ verifying that v_p contains at least one letter of A_k and $\{a_{p+1}, \ldots, a_{i_q-1}\} \subseteq A_k$. This entails there necessarily exists $p \in [\![i_{q-1}, i_q - 1]\!]$ such that $A_p \cap A_k \neq \emptyset$ and $\{a_{p+1}, \ldots, a_k\} \subseteq A_k$; we then take the smallest such p. Let us also set $\rho \in [\![i_{q-1}, i_q - 1]\!]$ to be the biggest integer in $[\![i_{q-1} + 1, i_q - 1]\!]$ verifying $a_\rho \notin A_k$ or i_{q-1} if it does not exist: by definition of p, we necessarily have $\rho \leq p$. By hypothesis, the following conditions are verified:

- for all $\kappa \in [\![q, j]\!]$, there exists $B_\kappa \subseteq \Sigma$ such that $\Delta_\kappa = B_\kappa^*$ and $A_k \subseteq B_\kappa \cup \{a_{i_{\kappa+1}}\}$;
- if $p + 1 < i_q$, then there exists $b \in \Sigma$ verifying $a_{p+1} = \cdots = a_{i_q-1} = b$ and $A_p \cap A_k = \{b\}$;

– $A_k \subseteq A_p \cup \{a_{i_q}\}$.

This implies in particular that if $p+1 < i_q$, $A_k = \{b, a_{i_q}\}$, so that e' and f' both contain only the letter b, since they do not contain the letter a_{i_q} by definition of q. We also observe that, if $p+1 < i_q$, none of $A_{p+1}, \ldots, A_{i_q-1}$ contains the letter b, because if there would exist some $\kappa \in [\![p+1, i_q-1]\!]$ verifying that $b \in A_\kappa$, it would contradict the fact that the word $a_1 \cdots a_p b^{\kappa-p+1} a_{\kappa+1} \cdots a_k \in L$ has a unique decomposition $u_0 a_1 u_1 a_2 u_2 \cdots u_{k-1} a_k u_k$ where $u_\lambda \in A_\lambda^*$ for all $\lambda \in [\![0,k]\!]$.

Step 2. The second step is to prove that $x\alpha^\omega$ belongs to $A_0^* a_1 A_1^* \cdots a_p A_p^*$. By the way it was defined, if $\rho > 0$, the letter a_ρ in w's decomposition is at a position at most in the last factor t of α^ω of w, as the letter $a_{i_{q-1}}$ in w's decomposition is at a position in one of the $q-1$ first factors α of w when $\rho = i_{q-1}$ and $e'\beta^\omega y \in A_k^*$. Moreover, if p is different from ρ, then as none of $A_\rho, \ldots, A_{p-1}$ contains a letter of A_k by minimality of p and as, in w, the last factor t of α^ω is followed by a factor e' containing only letters in A_k and of length at least $k-1 \geq p-\rho$, we have that the letter a_p in w's decomposition is at a position at most at the position $p-\rho$ of the last factor e' of α^ω of w. Since the letter a_{i_q} in w's decomposition is at position δ_1 in w, which is at least at a position in the first factor u of β^ω of w, it follows that, if $p+1 < i_q$, all letters $a_{p+1}, \ldots, a_{i_q-1}$ in w's decomposition are at positions in the first factor β of w. This is because otherwise, as there is a factor e' of length at least $k-1 \geq i_q-1-p$ preceding the first factor u of β^ω of w, at least one of $v_{p+1}, \ldots, v_{i_q-1}$ would contain the letter b. All in all, this means we have $x\alpha^\omega \in A_0^* a_1 A_1^* \cdots a_p A_p^*$.

Step 3. Since $\eta_{L'}(e')$ and $\eta_{L'}(f')$ are idempotents and ω is a multiple of $\omega_{L'}$, we have

$$(e'sf'te')^\omega (e'e'f'e'e')^\omega \sim_{L'} (e'sf'te')^\omega e'sf'e'e'(e'e'f'e'e')^\omega$$

by hypothesis, using the equational characterisation of $\mathbf{J} * \mathbf{D}$. Obviously,

$$x(e'sf'te')^\omega (e'e'f'e'e')^\omega a_{i_q} \cdots a_{k-1} = x\alpha^\omega (e'e'f'e'e')^\omega a_{i_q} \cdots a_{k-1} \in L'$$

by the simple fact that all letters of both e' and f' are in A_p (because they are in A_k and are different from a_{i_q} by definition of q) and, when $p+1 < i_q$, $a_{p+1}, \ldots, a_{i_q-1}$ are equal to the sole letter b in e' and f'. We thus have

$$z = x\alpha^\omega e'sf'e'e'(e'e'f'e'e')^\omega a_{i_q} \cdots a_{k-1} \in L' \ .$$

Let $z = u_0 a_1 u_1 a_2 \cdots u_{k-1} a_k u_k$ where $u_\lambda \in A_\lambda^*$ for all $\lambda \in [\![0,k]\!]$ be the unique de-

composition of z according to T. In a way similar to the previous case, we can show that if $\rho > 0$, then the letter a_ρ in z's decomposition is at a position at most in the last factor s of z, that if p is different from ρ, then the letter a_p in z's decomposition is at a position at most at the position $p - \rho$ of the factor f' after the last factor s of z, and that if $p + 1 < i_q$, then all letters $a_{p+1}, \dots, a_{i_q-1}$ in z's decomposition are at positions in the last factor e' of z. Putting all together, we can deduce that $x\alpha^\omega e'sf'_1 \ \cdots f'_{p-\rho} = x(e'sf'te')^\omega e'sf'_1 \cdots f'_{p-\rho} \in A_0^* a_1 A_1^* \cdots a_p A_p^*$. Letting $\sigma = \left|x\alpha^\omega e'sf'_1 \ \cdots f'_{p-\rho}\right|$, we indeed have that $w'_1 \cdots w'_\sigma = x\alpha^\omega e'sf'_1 \ \cdots f'_{p-\rho} \in A_0^* a_1 A_1^* \cdots a_p A_p^*$, where $\sigma + 1$ corresponds to a position in the factor f' of γ of w' (because $p + 1 \leq k \leq |f'|$) and σ is at least $\delta'_0 + 1$ and at most $\delta'_1 - i_q + p$ (because δ'_1 cannot correspond to a position that is before the first factor v of w', so that $\delta'_1 - \sigma > |f'| - p + \rho \geq k - p + \rho \geq i_q - p$).

Step 4. The fourth step is to prove that $w'_{\sigma+1} \cdots w'_{\delta'_1-1}$ belongs to $A_p^* a_p A_{p+1}^* \cdots a_{i_q-1} A_{i_q-1}^*$. We have that $w'_{\sigma+1} \cdots w'_{\delta'_1-1}$ does only contain letters belonging to A_k (because it only contains letters appearing in β by the fact that $\sigma + 1$ corresponds to a position in the factor f' of γ of w') that are different from a_{i_q} (since $\delta'_0 < \sigma$) and hence belong to $A_p \cap A_k$, as $A_k \subseteq A_p \cup \{a_{i_q}\}$. This means that $w'_{\sigma+1} \cdots w'_{\delta'_1-1} \in A_p^* a_p A_{p+1}^* \cdots a_{i_q-1} A_{i_q-1}^*$ as either $p + 1 = i_q$ and then all letters in $w'_{\sigma+1} \cdots w'_{\delta'_1-1}$ straightforwardly belong to A_p^*, or $p + 1 < i_q$ and then b is the sole letter in $w'_{\sigma+1} \cdots w'_{\delta'_1-1}$ since it is the sole one in $A_p \cap A_k$ and the word is long enough to contain $a_{p+1}, \dots, a_{i_q-1}$, all equal to b. Hence, we have eventually shown that $w'_1 \cdots w'_{\delta'_1-1} \in A_0^* a_1 A_1^* \ \cdots a_{i_q-1} A_{i_q-1}^* = \mathcal{L}(\Delta_0) a_{i_1} \mathcal{L}(\Delta_1) \cdots a_{i_{q-1}} \mathcal{L}(\Delta_{q-1})$.

Step 5. We finally show that $w'_{\delta'_1+1} \cdots w'_{|w'|} \in \mathcal{L}(\Delta_q) a_{i_{q+1}} \cdots \mathcal{L}(\Delta_j) a_k A_k^*$. We know that δ'_1 corresponds to a position in the factor ve' of γ of w' or the first factor β of w', that all letters appearing in ve', β and y do belong to A_k, that $\{a_{i_{q+1}}, \dots, a_k\} \subseteq A_k$ and that for all $\kappa \in [\![q, j]\!]$, there exists $B_\kappa \subseteq \Sigma$ such that $\Delta_\kappa = B_\kappa^*$ and $A_k \subseteq B_\kappa \cup \{a_{i_{\kappa+1}}\}$. So, as $w'_{\delta'_1+1} \cdots w'_{|w'|}$ contains $a_{i_{q+1}} \cdots a_{i_j} a_k$ as a subword because $w_{\delta_1+1} \cdots w_{|w|}$ does, it follows that $w'_{\delta'_1+1} \cdots w'_{|w'|} \in B_q^* a_{i_q} \cdots B_j^* a_k A_k^* = \mathcal{L}(\Delta_q) a_{i_{q+1}} \cdots \mathcal{L}(\Delta_j) a_k A_k^*$.

In the end, we can conclude that $x\alpha^\omega \gamma \beta^\omega y = w' \in L$.

- If ι corresponds to a position in the factor y of w, say position $\kappa \in [|y|]$, then ι' also corresponds to position κ in the factor y of w', otherwise it would violate minimality of ι when we take ω' to be a big enough multiple of $\omega_{L'}$ verifying $\omega > j + 1$. Since

$\eta_{L'}(e')$ and $\eta_{L'}(f')$ are idempotents and ω is a multiple of $\omega_{L'}$, we have

$$(e'sf'te')^{\omega}(e'uf've')^{\omega} \sim_{L'} (e'sf'te')^{\omega}e'sf've'(e'uf've')^{\omega}$$

by hypothesis, using the equational characterisation of $\mathbf{J} * \mathbf{D}$. So as $w_1 \cdots w_{\iota-1} = x\alpha^{\omega}\beta^{\omega}y_1 \cdots y_{\kappa-1} \in L'$, we have $w'_1 \cdots w'_{\iota'-1} = x\alpha^{\omega}\gamma\beta^{\omega}y_1 \cdots y_{\kappa-1} \in L'$. Therefore, as $w'_{\iota'+1} \cdots w'_{|w'|} = y_{\kappa+1} \cdots y_{|y|} = w_{\iota+1} \cdots w_{|w|} \in A_k^*$, we have that $x\alpha^{\omega}\gamma\beta^{\omega}y \in L$.

Let $x, y \in \Sigma^*$ such that $x\alpha^{\omega}\gamma\beta^{\omega}y \in L$. In a way similar to above, we can show that then, $x\alpha^{\omega}\beta^{\omega}y \in L$.

This finishes to show that

$$(esfte)^{\omega_L}(eufve)^{\omega_L} \sim_L (esfte)^{\omega_L}esfve(eufve)^{\omega_L} \ .$$

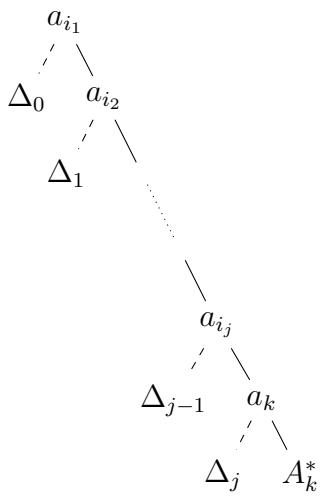
Since this holds for any $e, s, f, t, u, v \in \Sigma^+$ such that $\eta_L(e)$ and $\eta_L(f)$ are idempotents, it follows that $L \in \mathcal{L}(\mathbf{J} * \mathbf{D})$.

Case 4. Symmetric to case 2.

Case 5. Symmetric to case 3. □

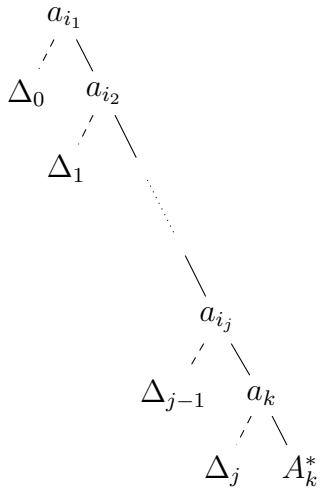
We derive the following corollary, that shows we can "cut off" a given right part of the right branch of a straight right NSUMT and still stay in $\mathcal{L}(\mathbf{J} * \mathbf{D})$. This corollary is used twice in the inductive construction of the proof of Proposition 5.4.2 and once in the lemma that is stated and proved just after this corollary.

Corollary 5.4.23. *Let $A_0^*a_1A_1^*a_2 \cdots A_{k-1}^*a_kA_k^*$ ($k \in \mathbb{N}, k \geq 2$) be a SUM over a finite alphabet Σ in $\mathcal{L}(\mathbf{J} * \mathbf{D})$ and*

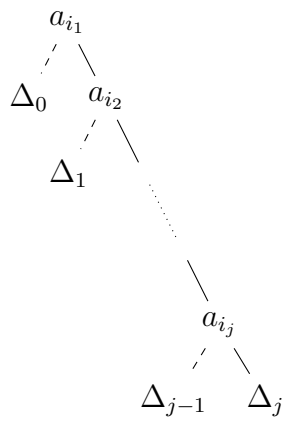


some decomposition of it where $\Delta_0, \Delta_1, \ldots, \Delta_{j-1}, \Delta_j$ $(j \in \mathbb{N}_{>0})$ *are straight left NSUMTs over* Σ *and the edges between the nodes labelled* a_{i_1} *to* a_{i_j} *are all full. Then, for all* $q \in [j]$, $\mathcal{L}(\Delta_0)a_{i_1}\mathcal{L}(\Delta_1)a_{i_2}\cdots\mathcal{L}(\Delta_{q-1})a_{i_q}\mathcal{L}(\Delta_q)$ *and* $\mathcal{L}(\Delta_0)a_{i_1}\mathcal{L}(\Delta_1)a_{i_2}\cdots\mathcal{L}(\Delta_{q-1})a_{i_q}A^*_{i_q}$ *are also SUMs over* Σ *in* $\mathcal{L}(\mathbf{J} * \mathbf{D})$.

Proof sketch. The SUMT



is a straight right NSUMT, so that since the SUM $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ it is a decomposition of belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$, Lemma 5.4.22 tells us that $\mathcal{L}(\Delta_0)a_{i_1}\mathcal{L}(\Delta_1)a_{i_2}\cdots\mathcal{L}(\Delta_{j-1})a_{i_j}\mathcal{L}(\Delta_j)$ is also a SUM over Σ in $\mathcal{L}(\mathbf{J} * \mathbf{D})$. When Δ_j is restricted to the root $A^*_{i_q}$, we obviously directly have that $\mathcal{L}(\Delta_0)a_{i_1}\mathcal{L}(\Delta_1)a_{i_2}\cdots\mathcal{L}(\Delta_{j-1})a_{i_j}A^*_{i_j}$ is also a SUM over Σ in $\mathcal{L}(\mathbf{J} * \mathbf{D})$. Otherwise, the SUMT

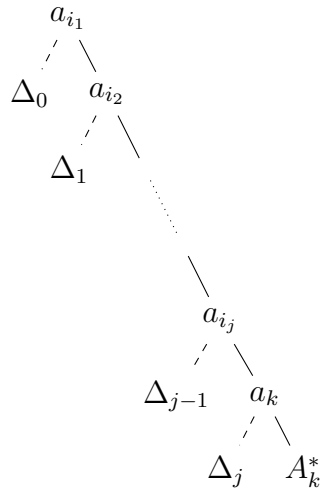


is a decomposition of $\mathcal{L}(\Delta_0)a_{i_1}\mathcal{L}(\Delta_1)a_{i_2}\cdots\mathcal{L}(\Delta_{j-1})a_{i_j}\mathcal{L}(\Delta_j)$ where Δ_j is a straight left NSUMT not restricted to a sole root, which means the former is a bent right NSUMT. Hence, by Lemma 5.4.22 again, we have, too, that $\mathcal{L}(\Delta_0)a_{i_1}\mathcal{L}(\Delta_1)a_{i_2}\cdots\mathcal{L}(\Delta_{j-1})a_{i_j}A_{i_j}^*$ is also a SUM over Σ in $\mathcal{L}(\mathbf{J}*\mathbf{D})$.

We then just have to apply Lemma 5.4.22 repeatedly in that way to prove the lemma for all $q \in [j]$. □

The next lemma in turn shows that under some conditions, we can even "cut off" a given right part of the right branch of a straight right NSUMT, but keeping the initial rightmost leaf, and still stay in $\mathcal{L}(\mathbf{J}*\mathbf{D})$. We use this result once in one of the subsubcases of the third subcase of the second case in the proof of Proposition 5.4.2.

Lemma 5.4.24. *Let $A_0^*a_1A_1^*a_2\cdots A_{k-1}^*a_kA_k^*$ ($k \in \mathbb{N}, k \geq 2$) be a SUM over an alphabet Σ in $\mathcal{L}(\mathbf{J}*\mathbf{D})$ and*



some decomposition of it where $\Delta_0, \Delta_1, \ldots, \Delta_{j-1}, \Delta_j$ ($j \in \mathbb{N}_{>0}$) are SUMTs over Σ and the

edges between the nodes labelled a_{i_1} to a_{i_j} are all full. Let $q \in [j]$ such that $\{a_{i_q+1}, \dots, a_k\} \subseteq A_k$. Then $\mathcal{L}(\Delta_0)a_{i_1}\mathcal{L}(\Delta_1)a_{i_2}\cdots\mathcal{L}(\Delta_{q-1})a_{i_q}A_k^*$ is also a SUM over Σ in $\mathcal{L}(\mathbf{J} * \mathbf{D})$.

Proof. Let $L = A_0^*a_1A_1^*a_2\cdots A_{k-1}^*a_kA_k^*$.

As $L \in \mathcal{L}(\mathbf{J} * \mathbf{D})$, by Corollary 5.4.23 when $q > 1$ and Lemma 5.4.20 otherwise ($q = 1$), we get that

$$\mathcal{L}(\Delta_0)a_{i_1}\mathcal{L}(\Delta_1)\cdots a_{i_{q-1}}\mathcal{L}(\Delta_{q-1})$$

belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ as well.

Now, for all $p \in [\![0, i_q - 1]\!]$, using the convention that $i_0 = 0$, when $A_p \cap A_k \neq \emptyset$ and $\{a_{p+1}, \dots, a_{i_q}\} \subseteq A_k$, if we take $q' \in [q]$ to be the unique integer in $[q]$ verifying $i_{q'-1} \leq p < i_{q'}$, since $\{a_{i_q+1}, \dots, a_k\} \subseteq A_k$, using Lemma 5.4.22 gives us in particular that the following conditions are verified:

- for all $\iota \in [\![q', q-1]\!]$, there exists $B_\iota \subseteq \Sigma$ such that $\Delta_\iota = B_\iota^*$ and $A_k \subseteq B_\iota \cup \{a_{i_{\iota+1}}\}$;
- if $p + 1 < i_{q'}$, then there exists $b \in \Sigma$ verifying $a_{p+1} = \cdots = a_{i_{q'}-1} = b$ and $A_p \cap A_k = \{b\}$;
- $A_k \subseteq A_p \cup \{a_{i_{q'}}\}$.

Putting all together, by Lemma 5.4.22 again, we can conclude that

$$\mathcal{L}(\Delta_0)a_{i_1}\mathcal{L}(\Delta_1)a_{i_2}\cdots\mathcal{L}(\Delta_{q-1})a_{i_q}A_k^*$$

is in $\mathcal{L}(\mathbf{J} * \mathbf{D})$. □

The two following lemmata show that, under different assumptions, in a given SUM $A_0^*a_1A_1^*a_2\cdots A_{k-1}^*a_kA_k^*$, we can replace an alphabet A_p by $\emptyset$ and still stay in $\mathcal{L}(\mathbf{J} * \mathbf{D})$, the essential difference between the two of them being that in the first one we assume that $a_p \in A_p$ when $p > 0$ and in the second one that $a_p \notin A_p$ when $p > 0$, which requires different additional hypotheses for the conclusion to hold. Each of them is used once in the first subcase of the second case in the proof of Proposition 5.4.2.

Lemma 5.4.25. *Let $A_0^*a_1A_1^*a_2\cdots A_{k-1}^*a_kA_k^*$ ($k \in \mathbb{N}$) be a SUM over an alphabet Σ in $\mathcal{L}(\mathbf{J} * \mathbf{D})$ and let $p \in [\![0, k]\!]$ such that $A_p \cap A_i = \emptyset$ for all $i \in [\![p+1, k]\!]$ and $a_p \in A_p$ when $p > 0$. Then $A_0^*a_1\cdots A_{p-1}^*a_pa_{p+1}A_{p+1}^*\cdots a_kA_k^*$ is also a SUM over Σ in $\mathcal{L}(\mathbf{J} * \mathbf{D})$.*

Proof. Let us fix an alphabet Σ.

The proof goes by induction on k.

Base case $k = 0$**.** Trivial.

Induction. Let $k \in \mathbb{N}_{>0}$ and assume the lemma is true for all SUMs of degree less than k over Σ.

Let $L = A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ be a SUM over Σ of degree k. Let us take some $p \in [\![0, k]\!]$ verifying $A_p \cap A_i = \emptyset$ for all $i \in [\![p+1, k]\!]$ and $a_p \in A_p$ when $p > 0$. Let us finally consider T some NSUMT over Σ that is a decomposition of L. We consider four different cases, one for each of the possible shapes of T.

Case 1. T is a bent right NSUMT of the form

$$\begin{array}{c} \Theta_0 \\ \quad \backslash \\ \quad\ \Theta_1 \\ \quad / \\ A_{p'}^* \end{array}$$

where $p' \in [\![1, k-1]\!]$,

$$\Delta_0 = \begin{array}{c} \Theta_0 \\ \quad \backslash \\ \quad\ A_{p'}^* \end{array}$$

is a straight right NSUMT and

$$\Delta_1 = \begin{array}{c} \quad\ \Theta_1 \\ \quad / \\ A_{p'}^* \end{array}$$

is a straight left NSUMT. By Lemma 5.4.22, $\mathcal{L}(\Delta_0) = A_0^* a_1 A_1^* a_2 \cdots A_{p'-1}^* a_{p'} A_{p'}^*$ and $\mathcal{L}(\Delta_1) = A_{p'}^* a_{p'+1} A_{p'+1}^* a_{p'+2} \cdots A_{k-1}^* a_k A_k^*$ both belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$. Now, if $p \leq p'$, it is easily seen that, by inductive hypothesis,

$$A_0^* a_1 \cdots A_{p-1}^* a_p a_{p+1} A_{p+1}^* \cdots a_{p'} A_{p'}^*$$

does also belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ and if $p \geq p'$, it is as easily seen that, by inductive hypothesis,

$$A_{p'}^* a_{p'+1} \cdots A_{p-1}^* a_p a_{p+1} A_{p+1}^* \cdots A_{k-1}^* a_k A_k^*$$

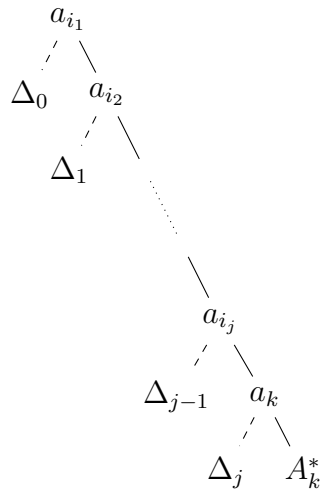
does belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ as well. Using Lemma 5.4.22, we can then conclude that

$$A_0^* a_1 \cdots A_{p-1}^* a_p a_{p+1} A_{p+1}^* \cdots a_k A_k^*$$

is in $\mathcal{L}(\mathbf{J} * \mathbf{D})$.

Case 2. T is a bent left NSUMT. This case is treated symmetrically to the previous one.

Case 3. T is a straight right NSUMT of the form



where $\Delta_0, \Delta_1, \ldots, \Delta_{j-1}, \Delta_j$ ($j \in \mathbb{N}$) are straight left NSUMTs over Σ and the edges between the nodes labelled a_{i_1} to a_{i_j} are all full. We use the convention that $i_0 = 0$ and $i_{j+1} = k$.

Lemma 5.4.22 tells us that $\mathcal{L}(\Delta_0) a_{i_1} \mathcal{L}(\Delta_1) a_{i_2} \cdots \mathcal{L}(\Delta_{j-1}) a_{i_j} \mathcal{L}(\Delta_j)$ is in $\mathcal{L}(\mathbf{J} * \mathbf{D})$.

When $p = k$, it is obvious that for all $p' \in [\![0, k-1]\!]$, we have $A_{p'} \cap \emptyset = \emptyset$. Therefore, by Lemma 5.4.22, $\mathcal{L}(\Delta_0) a_{i_1} \mathcal{L}(\Delta_1) a_{i_2} \cdots \mathcal{L}(\Delta_{j-1}) a_{i_j} \mathcal{L}(\Delta_j) a_k$ is in $\mathcal{L}(\mathbf{J} * \mathbf{D})$.

Otherwise, when $p < k$, by inductive hypothesis, it is readily seen that

$$A_0^* a_1 \cdots A_{p-1}^* a_p a_{p+1} A_{p+1}^* \cdots a_{k-1} A_{k-1}^*$$

belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$. Let $q \in [j+1]$ be the unique integer in $[j+1]$ verifying $i_{q-1} \leq p < i_q$. When $p > 0$, as $a_p \in A_p$ and $A_p \cap A_k = \emptyset$, we have $a_p \notin A_k$. Hence, for all $p' \in [\![0, p-1]\!]$, we have $A_{p'} \cap A_k = \emptyset$ or $\{a_{p'+1}, \ldots, a_k\} \not\subseteq A_k$. Obviously, $\emptyset \cap A_k = \emptyset$ and, by Lemma 5.4.22, for all $p' \in [\![p+1, k-1]\!]$, $A_{p'} \cap A_k \neq \emptyset$ implies that $\{a_{p'+1}, \ldots, a_k\} \not\subseteq A_k$ or, if we take

$q' \in [j+1]$ to be the unique integer in $[j+1]$ verifying $i_{q'-1} \leq p' < i_{q'}$, the following conditions are verified:

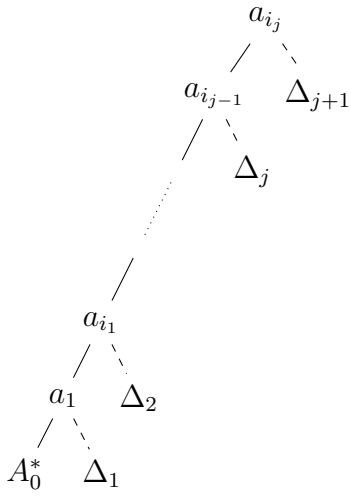
- for all $\iota \in [\![q', j]\!]$, there exists $B_\iota \subseteq \Sigma$ such that $\Delta_\iota = B_\iota^*$ and $A_k \subseteq B_\iota \cup \{a_{i_{\iota+1}}\}$;
- if $p'+1 < i_{q'}$, then there exists $b \in \Sigma$ verifying $a_{p'+1} = \cdots = a_{i_{q'}-1} = b$ and $A_{p'} \cap A_k = \{b\}$;
- $A_k \subseteq A_{p'} \cup \{a_{i_{q'}}\}$.

We observe that, for each $p' \in [\![p+1, k-1]\!]$, all these conditions are still verified for L in which A_p has been replaced by $\emptyset$, because they do not imply anything on A_p since the associated q' is at least as big as q. Therefore, by Lemma 5.4.22, we can conclude that

$$A_0^* a_1 \cdots A_{p-1}^* a_p a_{p+1} A_{p+1}^* \cdots a_k A_k^*$$

is in $\mathcal{L}(\mathbf{J} * \mathbf{D})$.

Case 4. T is a straight left NSUMT of the form



where $\Delta_1, \Delta_2, \ldots, \Delta_j, \Delta_{j+1}$ ($j \in \mathbb{N}$) are straight right NSUMTs over Σ and the edges between the nodes labelled a_{i_1} to a_{i_j} are all full. We use the convention that $i_0 = 1$ and $i_{j+1} = k+1$.

Lemma 5.4.22 tells us that $\mathcal{L}(\Delta_1) a_{i_1} \mathcal{L}(\Delta_2) \cdots a_{i_{j-1}} \mathcal{L}(\Delta_j) a_{i_j} \mathcal{L}(\Delta_{j+1})$ is in $\mathcal{L}(\mathbf{J} * \mathbf{D})$.

When $p = 0$, it is obvious that for all $p' \in [\![1, k]\!]$, we have $\emptyset \cap A_{p'} = \emptyset$. Therefore, by Lemma 5.4.22, $a_1 \mathcal{L}(\Delta_1) a_{i_1} \mathcal{L}(\Delta_2) \cdots a_{i_{j-1}} \mathcal{L}(\Delta_j) a_{i_j} \mathcal{L}(\Delta_{j+1})$ is in $\mathcal{L}(\mathbf{J} * \mathbf{D})$.

Otherwise, when $p > 0$, by inductive hypothesis, it is readily seen that

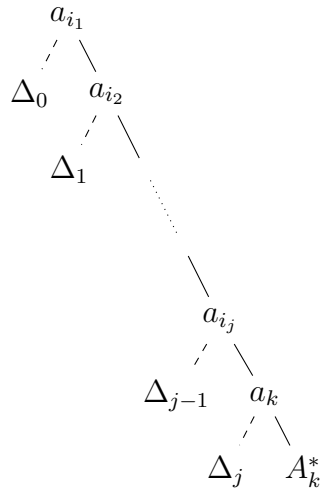
$$A_1^* a_2 \cdots A_{p-1}^* a_p a_{p+1} A_{p+1}^* \cdots a_k A_k^*$$

belongs to $\mathcal{L}(\mathbf{J} * \mathbf{D})$. Let $q \in [\![0, j]\!]$ be the unique integer in $[\![0, j]\!]$ verifying $i_q \leq p < i_{q+1}$. As $a_p \in A_p$, we have that for any $p' \in [\![i_{q+1}, k]\!]$, $A_0 \cap A_{p'} = \emptyset$ or $\{a_1, \ldots, a_{p'}\} \nsubseteq A_0$, otherwise we would have, by Lemma 5.4.22, that there exists $B_{q+1} \subseteq \Sigma$ such that $\Delta_{q+1} = B_{q+1}^*$ while we know Δ_{q+1} is not restricted to a sole root because $i_q < p$ (as otherwise it would mean that $a_p = a_{i_q} \notin A_p$), a contradiction. Obviously, $A_0 \cap \emptyset = \emptyset$ and, following the same reasoning as in the previous case, for all $p' \in [\![1, i_{q+1} - 1]\!] \setminus \{p\}$, the conditions of Lemma 5.4.22 are verified. Therefore, by Lemma 5.4.22, we can conclude that

$$A_0^* a_1 \cdots A_{p-1}^* a_p a_{p+1} A_{p+1}^* \cdots a_k A_k^*$$

is in $\mathcal{L}(\mathbf{J} * \mathbf{D})$. □

Lemma 5.4.26. *Let $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ ($k \in \mathbb{N}_{>0}$) be a SUM over an alphabet Σ in $\mathcal{L}(\mathbf{J} * \mathbf{D})$ and*



some decomposition of it where $\Delta_0, \Delta_1, \ldots, \Delta_{j-1}, \Delta_j$ ($j \in \mathbb{N}$) are straight left NSUMTs over Σ and the edges between the nodes labelled a_{i_1} to a_{i_j} are all full. Using the convention that $i_0 = 0$ and $i_{j+1} = k$, let $p \in [\![0, k-1]\!]$ such that $A_{p+1} = \cdots = A_{i_q - 1} = \emptyset$, $A_p \cap A_\iota = \emptyset$ for all $\iota \in [\![i_q, k]\!]$, $\{a_{i_q+1}, \ldots, a_k\} \subseteq A_k$ and $a_p \notin A_p$ when $p > 0$, where $q \in [j+1]$ is the unique integer in $[j+1]$ verifying $i_{q-1} \leq p < i_q$. Then $A_0^* a_1 \cdots A_{p-1}^* a_p a_{p+1} A_{p+1}^* \cdots a_k A_k^*$ is also a SUM over Σ in $\mathcal{L}(\mathbf{J} * \mathbf{D})$.

Proof. Since $A_p^* \cap \mathcal{L}(\Delta_q) a_{i_{q+1}} \cdots \mathcal{L}(\Delta_j) a_k A_k^* = \emptyset$, we have that

$$\begin{aligned} &A_0^* a_1 \cdots A_{p-1}^* a_p a_{p+1} A_{p+1}^* \cdots a_k A_k^* \\ =&A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^* \cap \bigcap_{c \in A_p} [a_{i_1}, \ldots, a_{i_{q-1}}, c a_{p+1} \cdots a_{i_q - 1} a_{i_q}]_\Sigma^{\complement} . \end{aligned}$$

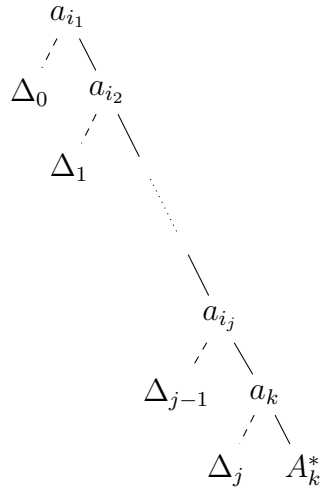
As $\mathcal{L}(\mathbf{J} * \mathbf{D})$ is an *ne*-variety of languages and $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ as well as the languages $[a_{i_1}, \ldots, a_{i_{q-1}}, c a_{p+1} \cdots a_{i_q - 1} a_{i_q}]_\Sigma$ for any $c \in A_p$ belong to it,

$$A_0^* a_1 \cdots A_{p-1}^* a_p a_{p+1} A_{p+1}^* \cdots a_k A_k^*$$

does also belong to it. □

Finally, this last lemma shows that, under some assumptions, in a SUM $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$, we can restrict all alphabets at the right of some letter a_p to contain only those letters belonging to A_k and still stay in $\mathcal{L}(\mathbf{J} * \mathbf{D})$. This result is used once in one of the subsubcases of the third subcase of the second case in the proof of Proposition 5.4.2.

Lemma 5.4.27. *Let $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ ($k \in \mathbb{N}, k \geq 2$) be a SUM over an alphabet Σ in $\mathcal{L}(\mathbf{J} * \mathbf{D})$ and*



some decomposition of it where $\Delta_0, \Delta_1, \ldots, \Delta_{j-1}, \Delta_j$ ($j \in \mathbb{N}_{>0}$) are straight left NSUMTs over Σ and the edges between the nodes labelled a_{i_1} to a_{i_j} are all full. Let $q \in [j]$ such that $\{a_{i_q+1}, \ldots, a_k\} \subseteq A_k$. Then $A_0^* a_1 \cdots A_{i_q-1}^* a_{i_q} (A_{i_q} \cap A_k)^* \cdots a_{k-1} (A_{k-1} \cap A_k)^* a_k A_k^*$ is also a SUM over Σ in $\mathcal{L}(\mathbf{J} * \mathbf{D})$.

Proof. We have that

$$A_0^* a_1 \cdots A_{i_q-1}^* a_{i_q} (A_{i_q} \cap A_k)^* \cdots a_{k-1} (A_{k-1} \cap A_k)^* a_k A_k^*$$

$$= A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^* \cap \bigcap_{b \notin A_k} [a_{i_1}, \ldots, a_{i_q}, b]_\Sigma^{\complement} ,$$

where we write $b \notin A_k$ instead of $b \in \Sigma \setminus A_k$. As $\mathcal{L}(\mathbf{J} * \mathbf{D})$ is an *ne*-variety of languages and $A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^*$ as well as the languages $[a_{i_1}, \ldots, a_{i_q}, b]_\Sigma$ for any $b \notin A_k$ belong to it, $A_0^* a_1 \cdots A_{i_q-1}^* a_{i_q} (A_{i_q} \cap A_k)^* \cdots a_{k-1} (A_{k-1} \cap A_k)^* a_k A_k^*$ does also belong to it. $\square$

Conclusion

In this thesis, we essentially made two contributions to the strand of research in computational complexity theory aiming for non-uniform complexity measure lower bounds within AC^1:

1. a formal, measure-independent, treatment and study of Nečiporuk's lower bound method;

2. a better understanding of the computational power of programs over monoids taken from small varieties of finite monoids.

Section 2.8 of Chapter 2 provides concluding remarks concerning our study of Nečiporuk's method, to which Chapter 2 is dedicated.

Concerning programs over monoids, in Section 3.4 of Chapter 3, we introduced a new notion of tameness for varieties of finite monoids that strengthens a similar notion introduced by Péladeau Péladeau [1990]. While the latter was equivalent, for a given variety of finite monoids $\mathbf{V}$, to an optimal separation of $\mathcal{P}(\mathbf{V})$ from all other classes of languages $\mathcal{P}(\mathbf{W})$ defined by p-recognition over a variety of finite monoids $\mathbf{W}$ not contained in $\mathbf{V}$, it did not necessarily imply a characterisation of the class of regular languages in $\mathcal{P}(\mathbf{V})$. Our notion of tameness ensures additionally that $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{QV})$, and we even know that equality holds when $\mathbf{V}$ is local. Thus, this notion serves as a tool to better understand the fundamental link between regular languages and programs over monoids, an understanding that provably holds the key to answering most open questions about the internal structure of NC^1.

A fundamental example of a tame variety of finite monoids is $\mathbf{A}$. Its tameness can be proven using the celebrated result of Furst, Saxe and Sipser, and, independently, Ajtai, that MOD_m is not in AC^0 for all $m \in \mathbb{N}, m \geq 2$; in fact, the tameness of $\mathbf{A}$ is even equivalent to this result. However, reproving this result using new finite-semigroup-theoretic techniques directly in the framework of p-recognition by finite aperiodic monoids

remains one of the major unresolved challenges when it comes to the program-over-monoid formalism.

We managed to use such techniques in Chapter 4 to prove tameness of $\mathbf{DA}$, a much smaller variety of finite monoids well within $\mathbf{A}$. As a corollary, we obtained that $\mathcal{P}(\mathbf{DA}) \cap \mathcal{R}\mathsf{eg} = \mathcal{L}(\mathbf{QDA})$. This proof of tameness in that restricted setting may or may not help to prove the same for the unrestricted case of $\mathbf{A}$, but at least it suggests a clear (though probably difficult) path towards this goal. For $\mathcal{F}$ a class of regular languages, we denote by $\mathfrak{B}(\mathcal{F})$ the Boolean closure of $\mathcal{F}$, i.e. the inclusion-wise smallest class of regular languages containing $\mathcal{F}$ and verifying that for each alphabet Σ, $\mathcal{F}(\Sigma^*)$ is closed under Boolean operations. We now define a hierarchy of classes of regular languages $\mathcal{V}_0, \mathcal{V}_{1/2}, \mathcal{V}_1, \mathcal{V}_{1+1/2}, \mathcal{V}_2, \dots$ in the following way:

- $\mathcal{V}_0$ is such that $\mathcal{V}_0(\Sigma^*) = \{\emptyset, \Sigma^*\}$ for each alphabet Σ;
- $\mathcal{V}_{1/2}$ is such that $\mathcal{V}_{1/2}(\Sigma^*) = \{\Sigma^* a \Sigma^* \mid a \in \Sigma\}$ for each alphabet Σ, and $\mathcal{V}_1 = \mathfrak{B}\big(\mathcal{V}_{1/2}\big)$;
- for any $n \in \mathbb{N}_{>0}$, $\mathcal{V}_{n+1/2}$ is such that

$$\mathcal{V}_{n+1/2}(\Sigma^*) = \{L_0 a_1 L_1 \cdots a_k L_k \mid k \in \mathbb{N}, a_1, \dots, a_k \in \Sigma, L_0, L_1, \dots L_k \in \mathcal{V}_n(\Sigma^*)\}$$

  for each alphabet Σ, and $\mathcal{V}_{n+1} = \mathfrak{B}\big(\mathcal{V}_{n+1/2}\big)$.

The so-called full levels $\mathcal{V}_k$ for $k \in \mathbb{N}$ happen to be varieties of star-free languages and verify $\mathcal{SF} = \bigcup_{k\in\mathbb{N}} \mathcal{V}_k$. When we consider the hierarchy of varieties of finite monoids $\mathbf{V}_0, \mathbf{V}_1, \mathbf{V}_2, \dots$ respectively associated to $\mathcal{V}_0, \mathcal{V}_1, \mathcal{V}_2, \dots$ through Eilenberg's correspondence, we get that $\mathbf{A} = \bigcup_{k\in\mathbb{N}} \mathbf{V}_k$. This hierarchy of classes of regular languages, a slightly twisted version of the Straubing-Thérien hierarchy (analogous and linked to the dot-depth hierarchy, see Pin [2017]) where the level 1 is a bit restricted, was introduced by Barrington and Thérien who showed that, for all $k \in \mathbb{N}$, p-recognition by monoids from $\mathbf{V}_k$ corresponds to the restriction of AC^0 to Boolean combinations of languages decided by sequences of AC^0-type circuits of depth at most k (see Barrington and Thérien [1988]). In light of all the properties of that hierarchy of varieties of finite monoids and knowing that $\mathbf{DA}$ is included in its lowest levels, we therefore think it makes particular sense to attack the question of the tameness of $\mathbf{A}$ by looking at the question of the tameness of $\mathbf{V}_k$ for increasing $k \in \mathbb{N}$. We observe that $\mathbf{V}_1 = \mathbf{J_1}$, which is not tame by Lemma 3.4.13, but we guess that $\mathbf{V}_k$ is tame for all $k \in \mathbb{N}, k \geq 2$.

Finally, in Chapter 5, we made progress towards an exact algebraic characterisation of the class of regular languages p-recognised by finite $\mathfrak{J}$-trivial monoids. In fact, $\mathbf{J}$ is not tame and its monoids are unexpectedly powerful when it comes to p-recognition of regular languages: while a result by Maciel, Péladeau and Thérien as well as our result of Chapter 4 together show that, in any case, $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{Q}(\mathbf{J} * \mathbf{D})) \cap \mathcal{L}(\mathbf{QDA})$, we presented good evidence that equality holds (this is Conjecture 2). Indeed, we observed that although programs over monoids in $\mathbf{J}$ cannot recognise languages requiring detection of the presence or absence of a word as a factor (basically dot-depth one languages), they can recognise such languages with the additional guarantee that this word does appear at most a constant number of times as a subword. The definition of threshold dot-depth one languages is based on this principle, and we proved that all the latter belong to $\mathcal{P}(\mathbf{J})$. Our intuition is that the class of threshold dot-depth one languages is in fact equal to $\mathcal{L}(\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}})$, but we were only able to prove that any SUM that is also of dot-depth one is in fact a threshold dot-depth one language. This is a first step towards a proof of Conjecture 1, that states that any language in $\mathcal{L}(\mathbf{J} * \mathbf{D} \cap \langle \mathbf{DA} \rangle_{\mathbf{S}})$ is in fact p-recognised by a finite $\mathfrak{J}$-trivial monoid, a conjecture whose truth would, we believe, rather easily imply the truth of Conjecture 2. However, we must admit that our proof of this particular case of Conjecture 1 is not satisfactory, because of its unreasonable length and complexity; we feel the final complete proof of Conjecture 1, if it exists, should be reasonably short and simple (in view of what was done by Knast in Knast [1983] for the algebraic characterisation of dot-depth one languages).

Future directions Below we give some ideas of lines along which one could conduct research in continuation of this Ph.D. thesis.

- One could explore even more general versions of Nečiporuk's method, as suggested in Section 2.8 of Chapter 2: for instance, one could consider a version of Nečiporuk's method where the bound given by any Nečiporuk function should only hold for a restricted class of Boolean functions or partitions of those.

- Obviously, one direction to follow would be to search for a full and reasonably short and simple proof of Conjecture 1. If one can find such a proof, then an algebraic characterisation of $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg}$, Conjecture 2, should not be far.

- As explained just above, in view of reproving MOD_m is not in $\mathcal{P}(\mathbf{A})$ for all $m \in \mathbb{N}, m \geq 2$ (or equivalently that $\mathbf{A}$ is tame) using novel semigroup-theoretic methods,

one path to follow would be to progressively prove, for growing $k \in \mathbb{N}, k \geq 2$, that $\mathbf{V}_k$ is tame, if true.

- More generally, carrying on with the exhaustive study of exactly which varieties of finite monoids are tame would definitely be interesting, because of the tight link between this question and many of the questions about the internal structure of NC^1. As this thesis focused on small varieties of finite *aperiodic* monoids, one place to continue would be to consider small varieties of finite *non-aperiodic* monoids (or even groups).

- Some questions about the general properties of tameness as we defined it still remain and could be investigated. For instance, for local and tame varieties $\mathbf{V}$ of finite monoids, we have shown that the regular languages recognised by programs over $\mathbf{V}$ are exactly those in $\mathcal{L}(\mathbf{QV})$. It is not clear whether locality is necessary, but we don't have any example of a tame variety $\mathbf{V}$ of finite monoids for which $\mathcal{L}(\mathbf{QV})$ is not included into $\mathcal{P}(\mathbf{V})$.

- One could imagine that throwing in randomness as well as non-determinism in the program-over-monoid formalism could give rise to interesting and tractable questions concerning derandomisation and the power of non-determinism for small complexity classes inside NC^1.

- Benjamin Rossman recently started to develop a wealth of new techniques to prove small-depth circuit lower bounds (see e.g. Li et al. [2014]). It is unclear whether those could help in the realm of programs over monoids, or even if those could be generalised in some direction (in terms of languages or models for which we can prove lower bounds using the technique), but it could be worth putting some effort into trying to see if it is the case or not.

Bibliography

Scott Aaronson. Why philosophers should care about computational complexity. *CoRR*, abs/1108.1791, 2011. URL http://arxiv.org/abs/1108.1791.

Miklós Ajtai. Σ_1^1-formulae on finite structures. *Annals of pure and applied logic*, 24(1): 1–48, 1983.

Eric Allender. A status report on the P versus NP question. *Advances in Computers*, 77:117–147, 2009. doi: 10.1016/S0065-2458(09)01204-2. URL http://dx.doi.org/10.1016/S0065-2458(09)01204-2.

Jorge Almeida. A syntactical proof of locality of DA. *IJAC*, 6(2):165–178, 1996. doi: 10.1142/S021819679600009X. URL https://doi.org/10.1142/S021819679600009X.

Jorge Almeida and Pascal Weil. Relatively free profinite monoids: an introduction and examples. *Semigroups, Formal Languages and Groups, JB Fountain, ed*, 466:73–117, 1995.

Noga Alon and Uri Zwick. On Nečiporuk's theorem for branching programs. *Theor. Comput. Sci.*, 64(3):331–342, 1989. doi: 10.1016/0304-3975(89)90054-6. URL http://dx.doi.org/10.1016/0304-3975(89)90054-6.

Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. ISBN 978-0-521-42426-4. URL http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264.

David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989. doi: 10.1016/0022-0000(89)90037-8. URL http://dx.doi.org/10.1016/0022-0000(89)90037-8.

David A. Mix Barrington and Denis Thérien. Finite monoids and the fine structure of NC^1. *J. ACM*, 35(4):941–952, 1988. doi: 10.1145/48014.63138. URL http://doi.acm.org/10.1145/48014.63138.

David A. Mix Barrington, Howard Straubing, and Denis Thérien. Non-uniform automata over groups. *Inf. Comput.*, 89(2):109–132, 1990. doi: 10.1016/0890-5401(90)90007-5. URL https://doi.org/10.1016/0890-5401(90)90007-5.

David A. Mix Barrington, Kevin J. Compton, Howard Straubing, and Denis Thérien. Regular languages in NC^1. *J. Comput. Syst. Sci.*, 44(3):478–499, 1992. doi: 10.1016/0022-0000(92)90014-A. URL http://dx.doi.org/10.1016/0022-0000(92)90014-A.

Paul Beame, Nathan Grosshans, Pierre McKenzie, and Luc Segoufin. Nondeterminism and an abstract formulation of Nečiporuk's lower bound method. *TOCT*, 9(1): 5:1–5:34, 2016. doi: 10.1145/3013516. URL http://doi.acm.org/10.1145/3013516.

Ravi B. Boppana and Michael Sipser. The complexity of finite functions. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*, pages 757–804. 1990.

Allan Borodin, Danny Dolev, Faith E. Fich, and Wolfgang J. Paul. Bounds for width two branching programs. *SIAM J. Comput.*, 15(2):549–560, 1986. doi: 10.1137/0215040. URL http://dx.doi.org/10.1137/0215040.

Laura Chaubard, Jean-Éric Pin, and Howard Straubing. Actions, wreath products of $\mathcal{C}$-varieties and concatenation product. *Theor. Comput. Sci.*, 356(1-2):73–89, 2006. doi: 10.1016/j.tcs.2006.01.039. URL https://doi.org/10.1016/j.tcs.2006.01.039.

Luc Dartois. *Méthodes algébriques pour la théorie des automates*. PhD thesis, Université Paris Diderot, Paris, 2014.

Luc Dartois and Charles Paperman. Adding modular predicates. *CoRR*, abs/1401.6576, 2014. URL http://arxiv.org/abs/1401.6576.

Samuel Eilenberg. *Automata, Languages, and Machines*, volume A. Academic Press, New York, 1974.

Samuel Eilenberg. *Automata, Languages, and Machines*, volume B. Academic Press, New York, 1976.

Lance Fortnow and Steven Homer. A short history of computational complexity. *Bulletin of the EATCS*, 80:95–133, 2003.

Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984. doi: 10.1007/BF01744431. URL http://dx.doi.org/10.1007/BF01744431.

Ricard Gavaldà and Denis Thérien. Algebraic characterizations of small classes of Boolean functions. In Helmut Alt and Michel Habib, editors, *STACS 2003, 20th Annual Symposium on Theoretical Aspects of Computer Science, Berlin, Germany, February 27 - March 1, 2003, Proceedings*, volume 2607 of *Lecture Notes in Computer Science*, pages 331–342. Springer, 2003. ISBN 3-540-00623-0. doi: 10.1007/3-540-36494-3_30. URL http://dx.doi.org/10.1007/3-540-36494-3_30.

Judy Goldsmith, Matthew A. Levy, and Martin Mundhenk. Limited nondeterminism. *SIGACT News*, 27(2):20–29, 1996. doi: 10.1145/235767.235769. URL http://doi.acm.org/10.1145/235767.235769.

Nathan Grosshans, Pierre McKenzie, and Luc Segoufin. The power of programs over monoids in DA. In Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin, editors, *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*, volume 83 of *LIPIcs*, pages 2:1–2:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. ISBN 978-3-95977-046-0. doi: 10.4230/LIPIcs.MFCS.2017.2. URL https://doi.org/10.4230/LIPIcs.MFCS.2017.2.

Johan Håstad. The shrinkage exponent of de Morgan formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998. doi: 10.1137/S0097539794261556. URL https://doi.org/10.1137/S0097539794261556.

Juraj Hromkovic and Georg Schnitger. Nondeterministic communication with a limited number of advice bits. *SIAM J. Comput.*, 33(1):43–68, 2003. doi: 10.1137/S0097539702414622. URL http://dx.doi.org/10.1137/S0097539702414622.

Russell Impagliazzo. Computational complexity since 1980. In Ramaswamy Ramanujam and Sandeep Sen, editors, *FSTTCS 2005: Foundations of Software Technology and*

Theoretical Computer Science, 25th International Conference, Hyderabad, India, December 15-18, 2005, Proceedings, volume 3821 of *Lecture Notes in Computer Science*, pages 19–47. Springer, 2005. ISBN 3-540-30495-9. doi: 10.1007/11590156_2. URL http://dx.doi.org/10.1007/11590156_2.

Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012. ISBN 978-3-642-24507-7. doi: 10.1007/978-3-642-24508-4. URL http://dx.doi.org/10.1007/978-3-642-24508-4.

Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*, pages 102–111. IEEE Computer Society, 1993. ISBN 0-8186-4070-7. doi: 10.1109/SCT.1993.336536. URL https://doi.org/10.1109/SCT.1993.336536.

Hartmut Klauck. One-way communication complexity and the Nečiporuk lower bound on formula size. *SIAM J. Comput.*, 37(2):552–583, 2007. doi: 10.1137/S009753970140004X. URL http://dx.doi.org/10.1137/S009753970140004X.

Ondrej Klíma and Libor Polák. Hierarchies of piecewise testable languages. *Int. J. Found. Comput. Sci.*, 21(4):517–533, 2010. doi: 10.1142/S0129054110007404. URL http://dx.doi.org/10.1142/S0129054110007404.

Robert Knast. A semigroup characterization of dot-depth one languages. *ITA*, 17(4): 321–330, 1983.

Andreas Krebs, Klaus-Jörn Lange, and Stephanie Reifferscheid. Characterizing TC^0 in terms of infinite groups. *Theory Comput. Syst.*, 40(4):303–325, 2007. doi: 10.1007/s00224-006-1310-2. URL http://dx.doi.org/10.1007/s00224-006-1310-2.

Kenneth Krohn and John L. Rhodes. Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines. *Transactions of the American Mathematical Society*, 116:450–464, 1965.

Manfred Kufleitner and Pascal Weil. On logical hierarchies within FO^2-definable languages. *Logical Methods in Computer Science*, 8(3), 2012. doi: 10.2168/LMCS-8(3:11)2012. URL https://doi.org/10.2168/LMCS-8(3:11)2012.

Richard E. Ladner and Nancy A. Lynch. Relativization of questions about log space computability. *Mathematical Systems Theory*, 10:19–32, 1976. doi: 10.1007/BF01683260. URL http://dx.doi.org/10.1007/BF01683260.

Clemens Lautemann, Pascal Tesson, and Denis Thérien. An algebraic point of view on the Crane Beach property. In Zoltán Ésik, editor, *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, volume 4207 of *Lecture Notes in Computer Science*, pages 426–440. Springer, 2006. ISBN 3-540-45458-6. doi: 10.1007/11874683_28. URL http://dx.doi.org/10.1007/11874683_28.

Yuan Li, Alexander A. Razborov, and Benjamin Rossman. On the AC^0 complexity of subgraph isomorphism. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 344–353. IEEE Computer Society, 2014. ISBN 978-1-4799-6517-5. doi: 10.1109/FOCS.2014.44. URL http://dx.doi.org/10.1109/FOCS.2014.44.

Oleg B. Lupanov. A method of circuit synthesis. *Izvestia V.U.Z. Radiofizika*, 1:120–140, 1958.

Oleg B. Lupanov. On the complexity of the realization of the functions of an algebra of logic by formulas. *Problemy Kibernet. Vyp*, 3:61–80, 1960.

Alexis Maciel, Pierre Péladeau, and Denis Thérien. Programs over semigroups of dot-depth one. *Theor. Comput. Sci.*, 245(1):135–148, 2000. doi: 10.1016/S0304-3975(99)00278-9. URL http://dx.doi.org/10.1016/S0304-3975(99)00278-9.

Meena Mahajan. Polynomial size log depth circuits: Between NC^1 and AC^1. *Bulletin of the EATCS*, 91:42–56, 2007.

William J. Masek. A fast algorithm for the string editing problem and decision graph complexity. Master's thesis, Massachusetts Institute of Technology, 1976.

Pierre McKenzie and Denis Thérien. Automata theory meets circuit complexity. In Giorgio Ausiello, Mariangiola Dezani-Ciancaglini, and Simona Ronchi Della Rocca, editors, *Automata, Languages and Programming, 16th International Colloquium, ICALP89, Stresa, Italy, July 11-15, 1989, Proceedings*, volume 372 of *Lecture Notes in Computer Science*, pages 589–602. Springer, 1989. ISBN 3-540-51371-X. doi: 10.1007/BFb0035785. URL https://doi.org/10.1007/BFb0035785.

Pierre McKenzie, Pierre Péladeau, and Denis Thérien. NC^1: The automata-theoretic viewpoint. *Computational Complexity*, 1:330–359, 1991. doi: 10.1007/BF01212963. URL http://dx.doi.org/10.1007/BF01212963.

Eduard I. Nečiporuk. On the complexity of schemes in some bases containing nontrivial elements with zero weights. *Problemy Kibernetiki*, 8:123–160, 1962.

Eduard I. Nečiporuk. On a Boolean function. *Doklady of the Academy of the USSR*, 169 (4):765–766, 1966. Translation: *Soviet Math. Doklady* 7:4, pp. 999-1000.

Charles Paperman. *Circuits booléens, prédicats modulaires et langages réguliers*. PhD thesis, Université Paris Diderot, Paris, 2014.

Wolfgang J. Paul. A 2.5 n-lower bound on the combinational complexity of Boolean functions. *SIAM J. Comput.*, 6(3):427–443, 1977. doi: 10.1137/0206030. URL https://doi.org/10.1137/0206030.

Pierre Péladeau. *Classes de circuits booléens et variétés de monoïdes*. PhD thesis, Université Pierre-et-Marie-Curie (Paris-VI), Paris, France, 1990.

Pierre Péladeau, Howard Straubing, and Denis Thérien. Finite semigroup varieties defined by programs. *Theor. Comput. Sci.*, 180(1-2):325–339, 1997. doi: 10.1016/S0304-3975(96)00297-6. URL http://dx.doi.org/10.1016/S0304-3975(96)00297-6.

Sylvain Perifel. *Complexité algorithmique*. Ellipses, 2014.

Jean-Éric Pin. The dot-depth hierarchy, 45 years later. In Stavros Konstantinidis, Nelma Moreira, Rogério Reis, and Jeffrey Shallit, editors, *The Role of Theory in Computer Science - Essays Dedicated to Janusz Brzozowski*, pages 177–202. World Scientific, 2017. ISBN 978-981-3148-19-2. doi: 10.1142/9789813148208_0008. URL https://doi.org/10.1142/9789813148208_0008.

Jean-Éric Pin. *Varieties Of Formal Languages*. Plenum Publishing Co., 1986. ISBN 0306422948.

Jean-Éric Pin. Syntactic semigroups. In *Handbook of formal languages*, pages 679–746. Springer, 1997.

Jean-Éric Pin. *Mathematical Foundations of Automata Theory*. 2016. URL https://www.irif.univ-paris-diderot.fr/~jep/PDF/MPRI/MPRI.pdf.

Jean-Éric Pin and Howard Straubing. Some results on $\mathcal{C}$-varieties. *ITA*, 39(1):239–262, 2005. doi: 10.1051/ita:2005014. URL http://dx.doi.org/10.1051/ita:2005014.

Pavel Pudlák. The hierarchy of Boolean circuits. *Computers and artificial intelligence*, 6 (5):449–468, 1987.

Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes*, 41(4):333–338, 1987.

Alexander A. Razborov. Lower bounds for deterministic and nondeterministic branching programs. In Lothar Budach, editor, *Fundamentals of Computation Theory, 8th International Symposium, FCT'91, Gosen, Germany, September 9-13, 1991, Proceedings*, volume 529 of *Lecture Notes in Computer Science*, pages 47–60. Springer, 1991. ISBN 3-540-54458-5. doi: 10.1007/3-540-54458-5_49. URL http://dx.doi.org/10.1007/3-540-54458-5_49.

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

John E. Savage. Computational work and time on finite machines. *J. ACM*, 19(4): 660–674, 1972. doi: 10.1145/321724.321731. URL http://doi.acm.org/10.1145/321724.321731.

John E. Savage. *The Complexity of Computing*. Wiley New York, 1976.

Marcel-Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965. doi: 10.1016/S0019-9958(65)90108-7. URL https://doi.org/10.1016/S0019-9958(65)90108-7.

Marcel-Paul Schützenberger. Sur le produit de concaténation non ambigu. *Semigroup Forum*, 13(1):47–75, Dec 1976. ISSN 1432-2137. doi: 10.1007/BF02194921. URL https://doi.org/10.1007/BF02194921.

Imre Simon. Piecewise testable events. In H. Barkhage, editor, *Automata Theory and Formal Languages, 2nd GI Conference, Kaiserslautern, May 20-23, 1975*, volume 33 of *Lecture Notes in Computer Science*, pages 214–222. Springer, 1975. ISBN 3-540-07407-4. doi: 10.1007/3-540-07407-4_23. URL https://doi.org/10.1007/3-540-07407-4_23.

Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82. ACM, 1987. ISBN 0-89791-221-7. doi: 10.1145/28395.28404. URL http://doi.acm.org/10.1145/28395.28404.

Howard Straubing. Families of recognizable sets corresponding to certain varieties of finite monoids. *Journal of Pure and Applied Algebra*, 15(3):305–318, 1979.

Howard Straubing. Finite semigroup varieties of the form $V * D$. *Journal of Pure and Applied Algebra*, 36:53–94, 1985.

Howard Straubing. When can one finite monoid simulate another? In *Algorithmic Problems in Groups and Semigroups*, pages 267–288. Springer, 2000.

Howard Straubing. Languages defined with modular counting quantifiers. *Inf. Comput.*, 166(2):112–132, 2001. doi: 10.1006/inco.2000.2923. URL https://doi.org/10.1006/inco.2000.2923.

Howard Straubing. On logical descriptions of regular languages. In Sergio Rajsbaum, editor, *LATIN 2002: Theoretical Informatics, 5th Latin American Symposium, Cancun, Mexico, April 3-6, 2002, Proceedings*, volume 2286 of *Lecture Notes in Computer Science*, pages 528–538. Springer, 2002. ISBN 3-540-43400-3. doi: 10.1007/3-540-45995-2_46. URL http://dx.doi.org/10.1007/3-540-45995-2_46.

Pascal Tesson. *Computational Complexity Questions Related to Finite Monoids and Semigroups*. PhD thesis, McGill University, Montreal, 2003.

Pascal Tesson and Denis Thérien. The computing power of programs over finite monoids. *J. Autom. Lang. Comb.*, 7(2):247–258, nov 2001. ISSN 1430-189X. URL http://dl.acm.org/citation.cfm?id=767345.767350.

Pascal Tesson and Denis Thérien. Diamonds are forever: the variety DA. *Semigroups, algorithms, automata and languages*, 1:475–500, 2002.

Denis Thérien. Programs over aperiodic monoids. *Theor. Comput. Sci.*, 64(3):271–280, 1989. doi: 10.1016/0304-3975(89)90051-0. URL http://dx.doi.org/10.1016/0304-3975(89)90051-0.

Bret Tilson. Categories as algebra: an essential ingredient in the theory of monoids. *Journal of Pure and Applied Algebra*, 48(1-2):83–198, 1987.

Heribert Vollmer. *Introduction to Circuit Complexity - A Uniform Approach*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 1999. ISBN 978-3-540-64310-4. doi: 10.1007/978-3-662-03927-4. URL https://doi.org/10.1007/978-3-662-03927-4.

Ingo Wegener. *The complexity of Boolean functions*. Wiley-Teubner, 1987. URL http://ls2-www.cs.uni-dortmund.de/monographs/bluebook/.

Ingo Wegener. *Branching Programs and Binary Decision Diagrams*. SIAM, 2000. ISBN 0-89871-458-3.

Avi Wigderson. P, NP and mathematics–a computational complexity perspective. In *Proc. of the 2006 International Congress of Mathematicians*, 2006.

Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014. doi: 10.1145/2559903. URL http://doi.acm.org/10.1145/2559903.

Appendix A

Résumé substantiel en langue française

Le cadre global. Cette thèse s'inscrit globalement dans le contexte de la recherche de minorants pour des modèles de calcul non uniformes : étant donné un langage, un modèle de calcul et une mesure de complexité, on cherche un minorant serré du coût donné par cette mesure pour la décision de ce langage dans ce modèle.

On s'intéresse tout particulièrement au modèle des *programmes de branchement*. Un programme de branchement sur mots binaires de longueur n est un graphe orienté acyclique avec un sommet source et deux sommets puits, l'un étiqueté 0, l'autre étiqueté 1, chaque sommet non puits étant étiqueté par une position dans l'entrée et ayant exactement deux arcs sortants, l'un étiqueté 0 et l'autre étiqueté 1. Un certain mot en entrée donne ensuite lieu à un unique chemin dans ce programme de branchement qui arrive soit au sommet puits étiqueté 0 (rejet), soit au sommet puits étiqueté 1 (acceptation). La mesure de complexité la plus fondamentale associée à ce modèle est, pour un langage donné, la fonction donnant pour chaque taille d'entrée possible la taille minimum, en nombre de sommets non puits, d'un programme de branchement décidant la restriction de ce langage aux mots de cette longueur donnée. Elle est fondamentale parce qu'elle capture la mesure de *complexité en espace pour les machines de Turing* de la manière suivante : tout langage décidé par une machine de Turing en espace $s(n)$ peut être décidé par une suite de programmes de branchement de taille majorée par $2^{\alpha \cdot s(n)}$, où α est une constante dépendant seulement de la machine de Turing. Ainsi, un minorant fort pour la taille des programmes de branchement nécessaire pour décider un certain langage implique directement un minorant fort pour l'espace nécessaire à une machine de Turing pour décider ce même langage. Par « fort », on entend « super-polynomial », puisqu'un tel minorant pour la

taille des programmes de branchement donnerait lieu à un minorant super-logarithmique pour l'espace des machines de Turing, ce qui est exactement ce que l'on vise pour séparer la classe L des langages décidables en espace logarithmique par une machine de Turing de la classe P des langages décidables en temps polynomial par une machine de Turing. La question de la relation entre L et P est une question ouverte centrale en théorie de la complexité algorithmique : il est bien connu que $\mathsf{L} \subseteq \mathsf{P}$, mais presque rien n'est connu à propos de l'inclusion inverse, bien qu'il soit largement admis qu'elle n'est pas vraie. L'importance de cette question est comparable à celle de la question fondamentale de la théorie de la complexité algorithmique, à savoir celle de la relation entre P et la classe NP des langages décidables en temps polynomial par une machine de Turing non déterministe.

La méthode de Nečiporuk. Le problème est que, dans l'état actuel de la recherche, séparer L de P de cette manière (ou, d'ailleurs, de quelque autre manière que ce soit) semble totalement hors d'atteinte. En effet, le meilleur minorant connu pour la taille des programmes de branchement est toujours, au moment d'écrire cette thèse, celui démontré par Nečiporuk il y a plus de 50 ans Nečiporuk [1966], qui est en $\Theta(n^2/\log^2 n)$. Nečiporuk utilisa implicitement une technique qui a plus tard été identifiée explicitement, définie et appliquée à plusieurs autres mesures de complexité pour différents modèles de calcul, et qui a également été exposée dans plusieurs livres de référence Savage [1976], Wegener [1987, 2000], Jukna [2012]. La première contribution de cette thèse, tirée d'un article coécrit avec Paul Beame, Pierre McKenzie et Luc Segoufin Beame et al. [2016], est une formulation de la méthode de minoration de Nečiporuk en amont de toute mesure de complexité spécifique (ce qui n'a jamais été fait systématiquement auparavant) et l'analyse des limites de cette méthode (en termes de meilleur minorant que l'on puisse obtenir) induites par des majorants pour la complexité de décision d'un langage spécifique. Ce cadre est ensuite utilisé pour appliquer la méthode et montrer ses limitations à la fois pour des modèles de calcul classiques avec les mesures de complexité associées et pour des variantes de ceux-ci jamais étudiées auparavant. Au bout du compte, dans ce cadre un peu plus général, des résultats de minoration et de limitation bien connus sont redémontrés et de nouveaux tels résultats sont prouvés.

Programmes sur monoïdes. Étant donné que l'on sait maintenant que la méthode de Nečiporuk ne permet pas d'obtenir de meilleurs minorants pour la taille des programmes de branchement qu'en $\Theta(n^2/\log^2 n)$ et puisque personne ne sait vraiment comment démontrer ne serait-ce qu'un minorant quadratique, que devrait-on faire ? Ce que les chercheurs firent une fois ce constat fait fut de commencer à chercher des minorants

pour la taille de *variantes restreintes de programmes de branchement* avec les mesures de complexité associées, en espérant que les connaissances acquises en démontrant de tels minorants aideraient à démontrer de meilleurs minorants pour le modèle non restreint. Parmi les restrictions étudiées, on trouvera le fait de restreindre le nombre de fois qu'une certaine position dans l'entrée peut être lue le long d'un chemin du sommet source à un sommet puits, ou encore de restreindre la largeur d'un programme de branchement. Les *programmes de branchement de largeur bornée* se sont révélés être particulièrement intéressants, surtout à partir du moment où Barrington Barrington [1989] prouva le résultat inattendu que les programmes de branchement de largeur bornée et de taille polynomiale permettent de décider tous les langages de NC^1. NC^1 est la classe des langages décidés par des circuits de taille polynomiale, profondeur logarithmique avec des portes NON et des portes OU, ET de degré entrant 2, une « petite » classe de complexité centrale et très étudiée. Son inclusion dans L lorsque l'on considère une variante uniforme appropriée est probablement stricte, mais il n'est, dans l'état actuel des connaissances, toujours pas exclu que $\mathsf{NP} \subseteq \mathsf{NC}^1$. En tant que suite directe de ce résultat et de sa démonstration, Barrington et Thérien Barrington and Thérien [1988] introduisirent le modèle des *programmes sur monoïdes*. Étant donné un monoïde fini $(M, *)$ et un alphabet fini Σ, un $(M, *)$-programme P sur Σ pour la longueur d'entrée n est simplement une suite d'instructions finie

$$(i_1, f_1)(i_2, f_2) \cdots (i_l, f_l) \ ,$$

où pour tout j, i_j indique une position dans l'entrée et f_j associe un élément de M à tout élément de Σ. De cette manière, à tout mot w de longueur n sur Σ en entrée, P associe un unique élément de M,

$$P(w) = f_1(w_{i_1}) * f_2(w_{i_2}) * \cdots * f_l(w_{i_l}) \ .$$

Un langage de mots de Σ^n est ensuite reconnu par P si et seulement s'il est égal à l'ensemble de mots dans Σ^n auxquels le programme P associe un élément dans un certain sous-ensemble F de M. Un langage de mots de longueur arbitraire est, quant à lui, reconnu par une suite de $(M, *)$-programmes.

Ce modèle de calcul et cette notion de reconnaissance peuvent être vus comme une généralisation de la notion de reconnaissance par morphismes dans des monoïdes finis, à la base même de la théorie algébrique des automates. D'une manière similaire au cas de la reconnaissance classique par de tels morphismes, on s'intéresse habituellement à la classe de langages reconnus par une suite de $(M, *)$-programmes de longueur polynomiale où

$(M, *)$ est un monoïde fini tiré d'une certaine variété de monoïdes finis $\mathbf{V}$ (une telle variété étant une classe de monoïdes finis close par produit direct et division, deux opérations basiques sur les monoïdes). Le résultat frappant montré par Barrington et Thérien (et dans quelques articles suivants) est que NC^1 et presque toutes ses sous-classes bien connues (à l'exception notable de TC^0) peuvent chacune être caractérisée comme une certaine classe $\mathcal{P}(\mathbf{V})$ de langages reconnus par des suites de programmes de longueur polynomiale sur monoïdes issus d'une telle variété $\mathbf{V}$. Par exemple, considérons AC^0, la classe des langages décidés par des circuits de taille polynomiale, profondeur constante avec des portes NON et des portes OU, ET de degré entrant non borné, et ACC^0 la même classe avec l'ajout de portes de comptage modulaire — deux des plus importantes sous-classes de NC^1. Alors, on a que NC^1, ACC^0 et AC^0 contiennent, respectivement, exactement les langages reconnus par des programmes, de longueur polynomiale, sur monoïdes de la variété de tous les monoïdes finis, des monoïdes finis résolubles et des monoïdes apériodiques finis. En théorie algébrique des automates, beaucoup de techniques ont été développées depuis les années 1960 pour caractériser algébriquement les classes de langages réguliers par les variétés de monoïdes finis les reconnaissant par morphismes. L'espoir était, et est toujours, que celles-ci puissent être généralisées à la reconnaissance par programmes et aider à aborder des questions ouvertes de théorie de la complexité algorithmique liées à NC^1 et sa structure interne ; ou, tout du moins, amener à de nouvelles démonstrations de résultats connus dans ce domaine en utilisant la théorie des semi-groupes finis. Le plus célèbre et fondamental de ces résultats est celui affirmant que pour tout entier $m \geq 2$, le langage MOD_m des mots sur $\{0, 1\}$ contenant un nombre de 1 non divisible par m n'est pas dans AC^0.

Mais aucun de ces espoirs ne s'est concrétisé pour le moment, à l'écriture de cette thèse. Comme expliqué pour le cas des minorants de la taille des programmes de branchement (non restreints), une approche que l'on pourrait avoir face à de telles difficultés serait de se concentrer sur des modèles encore plus restreints et essayer d'accumuler des connaissances et des techniques qui pourraient être utiles dans le cadre non restreint. Le formalisme des programmes sur monoïdes offre une façon directe de restreindre la puissance du modèle, simplement en restreignant la « puissance algébrique » à disposition en considérant des programmes sur monoïdes pris d'une certaine « petite » variété (au sens de l'inclusion) ; l'espoir est de pouvoir ensuite réutiliser ce que l'on apprend en étudiant la puissance des programmes sur monoïdes pris de ces « petites » variétés lorsque l'on mène cette étude avec des variétés plus grandes, comme celles données juste au-dessus. L'étude de l'expressivité des programmes sur monoïdes issus de « petites » variétés de monoïdes finis

amène également des questions intéressantes en tant que telles, souvent liées à la théorie algébrique des automates et la logique. Un nombre non négligeable de travaux précédents ont été faits dans ce contexte (par exemple Barrington et al. [1990], Gavaldà and Thérien [2003], Lautemann et al. [2006], Maciel et al. [2000], McKenzie et al. [1991], Tesson and Thérien [2001]).

La seconde contribution de cette thèse, qui correspond à des travaux en partie exposés dans un article de conférence co-publié avec Pierre McKenzie et Luc Segoufin Grosshans et al. [2017], se décompose en deux volets : à un niveau général, l'examen d'une propriété générale de la classe des langages réguliers contenus dans $\mathcal{P}(\mathbf{V})$ pour toute variété de monoïdes finis $\mathbf{V}$; à un niveau plus spécifique, l'étude de chacun des cas des variétés de monoïdes finis $\mathbf{DA}$ et $\mathbf{J}$, deux « petites » sous-variétés bien connues de celle des monoïdes apériodiques finis, importantes en théorie algébrique des automates et domaines connexes. Caractériser exactement la classe des langages réguliers dans $\mathcal{P}(\mathbf{V})$ alors que l'on fait varier $\mathbf{V}$ parmi toutes les variétés de monoïdes finis possibles est une tâche fondamentale car, comme il est montré dans McKenzie et al. [1991], deux classes $\mathcal{P}(\mathbf{V})$ et $\mathcal{P}(\mathbf{W})$ sont égales si et seulement si elles contiennent exactement les mêmes langages réguliers. En s'inspirant du travail de Péladeau, Straubing et Thérien Péladeau et al. [1997] concernant les programmes sur semi-groupes issus de variétés de semi-groupes finis d'une certaine forme, une nouvelle notion de *docilité* d'une variété de monoïdes finis $\mathbf{V}$ est introduite, capturant essentiellement la propriété que, sur les monoïdes issus de cette variété, la reconnaissance par programmes de longueur polynomiale ne permet pas de reconnaître imprévisiblement plus de langages réguliers que la reconnaissance classique par morphismes.

Le reste de la thèse s'attache ensuite d'abord à montrer que $\mathbf{DA}$ est docile, dérivant de cette propriété une caractérisation algébrique exacte de la classe des langages réguliers appartenant à $\mathcal{P}(\mathbf{DA})$. En outre, d'autres propriétés à propos de $\mathcal{P}(\mathbf{DA})$ intéressantes en elles-mêmes sont montrées. Cette thèse se tourne en dernier lieu vers $\mathbf{J}$, la variété des monoïdes finis appelés $\mathfrak{J}$-triviaux, un exemple notable d'une variété de monoïdes finis prouvée non docile. Ceci signifie, avant tout, que les programmes de longueur polynomiale sur monoïdes de $\mathbf{J}$ peuvent reconnaître « beaucoup plus » de langages réguliers qu'en considérant seulement la reconnaissance classique par morphismes sur monoïdes de $\mathbf{J}$. Le dernier chapitre de cette thèse, assez considérable, développe des résultats partiels (non publiés à ce jour) pour la caractérisation algébrique de la classe des langages réguliers appartenant à $\mathcal{P}(\mathbf{J})$ (en plus de quelques petits résultats à part concernant $\mathcal{P}(\mathbf{J})$).

Orientations futures. Voici enfin quelques idées de sujets de recherche en continuité avec cette thèse sur lesquels l'on pourrait travailler.

- On pourrait explorer des versions de la méthode de Nečiporuk encore plus générales que celle proposée dans cette thèse, dont les limitations sont en grande partie dues au fait qu'elle doive s'appliquer à tout langage binaire.
- Bien évidemment, une piste à suivre serait de chercher une preuve complète et à la fois raisonnablement courte et simple de la caractérisation algébrique exacte de la classe des langages réguliers contenus dans $\mathcal{P}(\mathbf{J})$ conjecturée dans cette thèse.
- Il y a une hiérarchie infinie de variétés de monoïdes finis entre **DA** et celle des monoïdes apériodiques finis. Si quelqu'un devait un jour redémontrer que MOD_m n'est pas dans AC^0 pour tout entier $m \geq 2$ en utilisant des méthodes de théorie des semi-groupes finis, une voie à suivre serait de montrer progressivement, si vraie, la docilité de chacun des niveaux de cette hiérarchie, dans l'ordre croissant des niveaux.
- La docilité pour les variétés de monoïdes finis semble être une notion intéressante, puisque, entre autres, la conjecture bien connue qu'ACC^0 est strictement incluse dans NC^1 est équivalente au fait que la variété des monoïdes résolubles finis est docile. Ceci motive la poursuite de l'étude exhaustive de la docilité des variétés de monoïdes finis débutée dans cette thèse. Comme cette dernière s'est concentrée sur des petites variétés de monoïdes *apériodiques*, un angle d'attaque possible serait d'examiner de petites variétés de monoïdes *non apériodiques* (ou même des groupes). Par ailleurs, cette thèse soulève aussi quelques questions à propos des propriétés générales de la docilité qu'il reste à résoudre.
- L'on pourrait imaginer qu'introduire des comportements probabilistes ainsi que non déterministes dans le formalisme des programmes sur monoïdes pourrait donner lieu à des questions intéressantes et accessibles concernant la déprobabilisation et la puissance du non déterminisme pour des petites classes de complexité dans NC^1.
- Benjamin Rossman a récemment commencé à développer une foule de techniques nouvelles pour prouver des minorants pour les circuits de faible profondeur (voir par exemple Li et al. [2014]). Il n'est pas clair si celles-ci pourraient aider dans le domaine des programmes sur monoïdes, ou même si celles-ci pourraient être généralisées dans une certaine direction (en termes de langages ou de modèles pour lesquels on peut démontrer des minorants en utilisant ces techniques), mais des efforts pour savoir ce qu'il en est pourraient s'avérer profitables.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Faculté des arts et des sciences
Université de Montréal

Titre : Les limites de la méthode de Nečiporuk et le pouvoir des programmes sur monoïdes issus de petites variétiés de monoïdes finis

Mots-clés : Complexité algorithmique, minorants, Nečiporuk, programmes sur monoïdes, DA, J

Résumé : Cette thèse porte sur des minorants pour des mesures de complexité liées à des sous-classes de la classe P de langages pouvant être décidés en temps polynomial par des machines de Turing. Nous considérons des modèles de calcul non uniformes tels que les programmes sur monoïdes et les programmes de branchement.

Notre première contribution est un traitement abstrait de la méthode de Nečiporuk pour prouver des minorants, indépendamment de toute mesure de complexité spécifique. Cette méthode donne toujours les meilleurs minorants connus pour des mesures telles que la taille des programmes de branchements déterministes et non déterministes ou des formules avec des opérateurs booléens binaires arbitraires ; nous donnons une formulation abstraite de la méthode et utilisons ce cadre pour démontrer des limites au meilleur minorant obtenable en utilisant cette méthode pour plusieurs mesures de complexité. Par là, nous confirmons, dans ce cadre légèrement plus général, des résultats de limitation précédemment connus et exhibons de nouveaux résultats de limitation pour des mesures de complexité auxquelles la méthode de Nečiporuk n'avait jamais été appliquée.

Notre seconde contribution est une meilleure compréhension de la puissance calculatoire des programmes sur monoïdes issus de petites variétés de monoïdes finis. Les programmes sur monoïdes furent introduits à la fin des années 1980 par Barrington et Thérien pour généraliser la reconnaissance par morphismes et ainsi obtenir une caractérisation en termes de semi-groupes finis de NC^1 et de ses sous-classes. Étant donné une variété $\mathbf{V}$ de monoïdes finis, on considère la classe $\mathcal{P}(\mathbf{V})$ de langages reconnus par une suite de programmes de longueur polynomiale sur un monoïde de $\mathbf{V}$: lorsque l'on fait varier $\mathbf{V}$ parmi toutes les variétés de monoïdes finis, on obtient différentes sous-classes de NC^1, par exemple AC^0, ACC^0 et NC^1 quand $\mathbf{V}$ est respectivement la variété de tous les monoïdes apériodiques finis, résolubles finis et finis. Nous introduisons une nouvelle notion de docilité pour les variétés de monoïdes finis, renforçant une notion de Péladeau. L'intérêt principal de cette notion est que quand une variété $\mathbf{V}$ de monoïdes finis est docile, nous avons que $\mathcal{P}(\mathbf{V})$ contient seulement des langages réguliers qui sont quasi reconnus par morphisme par des monoïdes de $\mathbf{V}$. De nombreuses questions ouvertes à propos de la structure interne de NC^1 seraient réglées en montrant qu'une variété de monoïdes finis appropriée est docile, et, dans cette thèse, nous débutons modestement une étude exhaustive de quelles variétés de monoïdes finis sont dociles. Plus précisément, nous portons notre attention sur deux petites variétés de monoïdes apériodiques finis bien connues : $\mathbf{DA}$ et $\mathbf{J}$. D'une part, nous montrons que $\mathbf{DA}$ est docile en utilisant des arguments de théorie des semi-groupes finis. Cela nous permet de dériver une caractérisation algébrique exacte de la classe des langages réguliers dans $\mathcal{P}(\mathbf{DA})$. D'autre part, nous montrons que $\mathbf{J}$ n'est pas docile. Pour faire cela, nous présentons une astuce par laquelle des programmes sur monoïdes de $\mathbf{J}$ peuvent reconnaître beaucoup plus de langages réguliers que seulement ceux qui sont quasi reconnus par morphisme par des monoïdes de $\mathbf{J}$. Cela nous amène à conjecturer une caractérisation algébrique exacte de la classe de langages réguliers dans $\mathcal{P}(\mathbf{J})$, et nous exposons quelques résultats partiels appuyant cette conjecture. Pour chacune des variétés $\mathbf{DA}$ et $\mathbf{J}$, nous exhibons également une hiérarchie basée sur la longueur des programmes à l'intérieur de la classe des langages reconnus par programmes sur monoïdes de la variété, améliorant par là les résultats de Tesson et Thérien sur la propriété de longueur polynomiale pour les monoïdes de ces variétés.

Title: The limits of Nečiporuk's method and the power of programs over monoids taken from small varieties of finite monoids

Keywords: Computational complexity, lower bounds, Nečiporuk, programs over monoids, DA, J

Abstract: This thesis deals with lower bounds for complexity measures related to subclasses of the class P of languages that can be decided by Turing machines in polynomial time. We consider non-uniform computational models like programs over monoids and branching programs.

Our first contribution is an abstract, measure-independent treatment of Nečiporuk's method for proving lower bounds. This method still gives the best lower bounds known on measures such as the size of deterministic and non-deterministic branching programs or formulæ with arbitrary binary Boolean operators; we give an abstract formulation of the method and use this framework to prove limits on the best lower bounds obtainable using this method for several complexity measures. We thereby confirm previously known limitation results in this slightly more general framework and showcase new limitation results for complexity measures to which Nečiporuk's method had never been applied.

Our second contribution is a better understanding of the computational power of programs over monoids taken from small varieties of finite monoids. Programs over monoids were introduced in the late 1980s by Barrington and Thérien as a way to generalise recognition by morphisms so as to obtain a finite-semigroup-theoretic characterisation of NC^1 and its subclasses. Given a variety $\mathbf{V}$ of finite monoids, one considers the class $\mathcal{P}(\mathbf{V})$ of languages recognised by a sequence of polynomial-length programs over a monoid from $\mathbf{V}$: as $\mathbf{V}$ ranges over all varieties of finite monoids, one obtains different subclasses of NC^1, for instance AC^0, ACC^0 and NC^1 when $\mathbf{V}$ respectively is the variety of all finite aperiodic, finite solvable and finite monoids. We introduce a new notion of tameness for varieties of finite monoids, strengthening a notion of Péladeau. The main interest of this notion is that when a variety $\mathbf{V}$ of finite monoids is tame, we have that $\mathcal{P}(\mathbf{V})$ does only contain regular languages that are quasi morphism-recognised by monoids from $\mathbf{V}$. Many open questions about the internal structure of NC^1 would be settled by showing that some appropriate variety of finite monoids is tame, and, in this thesis, we modestly start an exhaustive study of which varieties of finite monoids are tame. More precisely, we focus on two well-known small varieties of finite aperiodic monoids: $\mathbf{DA}$ and $\mathbf{J}$. On the one hand, we show that $\mathbf{DA}$ is tame using finite-semigroup-theoretic arguments. This allows us to derive an exact algebraic characterisation of the class of regular languages in $\mathcal{P}(\mathbf{DA})$. On the other hand, we show that $\mathbf{J}$ is not tame. To do this, we present a trick by which programs over monoids from $\mathbf{J}$ can recognise much more regular languages than only those that are quasi morphism-recognised by monoids from $\mathbf{J}$. This brings us to conjecture an exact algebraic characterisation of the class of regular languages in $\mathcal{P}(\mathbf{J})$, and we lay out some partial results that support this conjecture. For each of the varieties $\mathbf{DA}$ and $\mathbf{J}$, we also exhibit a program-length-based hierarchy within the class of languages recognised by programs over monoids from the variety, refining Tesson and Thérien's results on the polynomial-length property for monoids from those varieties.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Université de Montréal
2900, boul. Édouard-Montpetit
Montréal (Québec) H3T 1J4, Canada