LaBRI

université de BORDEAUX

THÈSE PRÉSENTÉE POUR OBTENIR LE GRADE DE

DOCTEUR
DE L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE MATHÉMATIQUES ET INFORMATIQUE

Par **Antonio CASARES**

Propriétés structurelles des automates sur les mots infinis et mémoire pour les jeux

---

Structural properties of automata over infinite words and memory for games

Sous la direction de :
Nathanaël Fijalkow et Igor Walukiewicz

Soutenue le 23 novembre 2023

Composition du jury :

Orna Kupferman	Professor, Hebrew University	Rapportrice
Christof Löding	Professor, RWTH Aachen University	Rapporteur
Christel Baier	Professor, TU Dresden	Examinatrice
Damien Pous	Directeur de Recherche, CNRS	Examinateur
Marc Zeitoun	Professeur, Université de Bordeaux	Examinateur
Nathanaël Fijalkow	Chargé de Recherche, CNRS	Directeur
Igor Walukiewicz	Directeur de Recherche, CNRS	Directeur

Membres invités :

Thomas Colcombet	Directeur de Recherche, CNRS	Co-Encadrant

Table of contents

Abstract

Automata and infinite duration games form the theoretical basis for the verification and synthesis of reactive systems. Contrary to the case of regular languages over finite words, several models of ω-automata exist, each one offering its own advantages. However, their structural and language-theoretical properties are still far from being fully understood. In this thesis, we study transformations between different types of ω-automata, as well as the minimisation problem. Moreover, we exhibit links between these automata and the complexity of winning strategies for games with ω-regular winning conditions, giving a complete characterisation of half-positionality for ω-regular languages.

Résumé

Les automates et les jeux à durée infinie constituent l'un des outils fondamentaux pour la vérification et la synthèse de systèmes réactifs. Contrairement au cas des langages réguliers sur les mots finis, plusieurs modèles d'automates existent, et de nombreuses questions sur leur structure et sur les langages qu'ils reconnaissent restent ouvertes. Dans cette thèse, nous étudions des transformations entre différents types d'automates sur les mots infinis, le problème de la minimisation, et le lien entre la structure de ces automates et la complexité des stratégies nécessaires pour gagner des jeux sur des graphes avec condition de gain ω-régulière.

Acknowledgments

First, I would like to thank my PhD advisors, Igor, Nath and Thomas for their guidance and the many mathematical insights they have shared with me, while also encouraging and motivating me to pursue my own research goals and collaborate with others. The lessons I take away from you go beyond purely academic knowledge. Thank you for showing me the importance of active engagement within the academic community and for exemplifying that it is possible to do science without forgetting ethical and environmental commitments.

I want to thank Orna Kupferman and Christof Löding for agreeing to review this document. Much of the research conducted on this thesis has been motivated and influenced by your work, and it brings me immense pleasure that you are the ones evaluating it. I would also want to thank Christel Baier, Damien Pous and Marc Zeitoun for accepting to be part of the jury.

I would like to express my deepest gratitude to Pierre Ohlmann, my older academic brother, whom I could actually consider a further PhD advisor (as if three were not enough!). From the very beginning of my thesis you have shown immense patience in working with me, and your enthusiasm and dynamism in research have made of the (absurdly numerous and lengthy) online working sessions a truly enjoyable experience.

I want to thank the many outstanding colleagues I have had the opportunity to work with during these years: Karoliina, Pierre V., Corto, León, Denis, Michał S., Michał P., Thejaswini, Patricia, Mickaël, Salomon, Klara, Alexandre, Florian, Aline, Irmak, and many others. Doing research with you has been greatly inspiring and fun!

Thank you to all the researchers from LaBRI for the interesting discussions and for fostering such a lively atmosphere. In particular, many thanks to Hugo, Laurent, Géraud, Anca, Asia, Diego, Guillaume, Marthe, Vincent P., Vincent D., Séb, Frédéric, Jérôme, Pascal, Sylvain, Marc and Meghyn.

Thanks to all the PhD students who have made my time at LaBRI such an enjoyable experience. Have you heard about Rémi Morvan and his nifty companion `knowledge-clustering`? He is a super nice guy whom I want to thank not only for saving me hundreds of hours of clustering knowledges, mais particulièrement pour son engagement dans les activités du laboratoire et pour organiser toute sorte d'activités qui contribuent tellement à notre bonheur quotidien. Merci à Thibault pour ~~me supporter~~ always supporting me, malgré mes utilisations parfois peut-être pas tout à fait appropriées des sachets de sauce, et à Aïda por sus visitas por el bureau 257 que siempre nos llenan de alegría. Merci à Zoé, pas seulement pour ses cookies délicieuses, mais aussi pour sa bienveillance et volonté à aider; à Clément pour ses leçons de escalade, slack, tricot (et en gros tout ce qu'on pourrait avoir envie de connaître !); thanks a lot to Alex, although your stay in Bordeaux has not been long, it has been a real pleasure to go dancing with you; à Théo pour m'aider à faire face à des enseignements de programmation effrayants; à Sarah et Adrian pour son accueil

récurrent pour des soirées jeux; à Joséphine pour toutes les sorties culturelles; à Alice pour ses cours intensifs d'acoustique; à Quentin pour son effort pour accueillir et intégrer les nouvelles.aux arrivant.e.s, les soirées jeux chez toi ont été un souffle de vie dans l'époque post-confinement; à Jonathan, pour toujours rayonner de la joie; à Aline, pour son rôle précieux au sein de l'AFoDIB; à Maximino pour des parties d'échecs passionnantes; à Claire pour me motiver à rédiger ma thèse (j'oublie pas les dessins que je te dois !); à Amaury, même s'il a pas encore fait une tortilla de patatas; et à Timothé(e), Paul, Edgar, Romain, Meghna, Abel, Colleen, Yanis, Pierre, Pierre-Marie et autant d'autres qui font de ce laboratoire un endroit si agréable.

Corto, I will never thank you enough for everything you have done for me during these last two years. Of course, having you in my office is extremely handy, as you can solve any problem I struggle with, and you are always willing to listen and check all my wrong proofs. But more importantly, tu as toujours su m'écouter et tu m'as apporté un soutien inconditionnel en toute circonstance. Cette thèse ne serait sûrement pas cette thèse si ce n'était pour toi. Merci beaucoup.

Je tiens à remercier toute l'équipe des Aigles de Bègles pour leur accueil chaleureux et pour la bonne ambiance qu'iels parviennent à créer avec ce grand sport qu'est l'ultimate frisbee ! En particulier, merci à toutes les personnes qui s'investissent dans le fonctionnement de l'équipe, et un grand merci à Paul Latouche, notre entraîneur d'or, toujours presque rationnel.

Merci à mes formidables colocs Joan, Emeline, Pilou, Guillaume et Paul pour les soirées salsa, karaoke, et pour rendre la cohabitation si facile. Merci a toutes les autres personnes que j'ai eu la chance de rencontrer à Bordeaux, et en particulier, un grand merci à Margaux pour son énergie inépuisable et pour ne jamais laisser me reposer un seul moment. Les semaines à Bordeaux sont bien plus passionantes avec toi.

Muchas gracias a Juanma por interesarse y dar respuesta a todas mis preguntas sobre teoría de conjuntos. Eres sin duda la persona que más ha compartido mis inquietudes lógico-filosóficas, y coincidir contigo durante mi paso por el LMFI hizo de ese año uno de los mejores de mi vida. Aunque ahora me haya pasado al lado computacional, espero que no me dejes olvidarme de todas las cuestiones que me atrajeron al estudio de la lógica y han hecho que esté hoy aquí.

Por supuesto, no olvido a todes mis amigues de Valladolid. Gracias a Jorge y Miguel por estar siempre ahí y sacar tiempo para vernos a cada oportunidad. ¡El próximo interrail no me lo pierdo! Gracias a Rodrigo por todas las conversaciones existenciales (don't forget to go check the album Cerúleo if you are reading this!), a Elena for being une excellente gamemaster, a Raquel por las tardes en la piscina y a Silvia, por no olvidarme aunque ya sea una matemática de éxito. No puedo terminar sin agradecer a Ana su permanente apoyo todos estos años.

Finalmente, quiero agradecer a mi familia todo su apoyo, en especial a mis padres, Juan y Paz, y a mi hermana, Chelo. Siempre habéis puesto mi bienestar y mis estudios por delante de cualquier otra cosa y durante toda mi vida habéis hecho un gran esfuerzo para darme todas las facilidades para que pueda perseguir mis metas. Si realizar esta tesis ha sido posible, es gracias a vosotros.

A mis padres, Juan y Paz.

Résumé étendu en français

Nous présentons ci-dessous un résumé du manuscrit en langue française. Pour une discussion plus approfondie, le lecteur pourra se référer à l'introduction générale (en langue anglaise).

1 Contexte

1.1 Décidabilité des logiques et synthèse de programmes

L'une des questions fondamentales en logique est celle de la décidabilité. Étant donné un système formel $\mathcal{L}$ (par exemple, la logique du premier ordre) et une structure mathématique $\mathcal{M}$ (par exemple, les entiers naturels), l'objectif est de trouver un algorithme permettant de résoudre le problème suivant :

> Étant donné une formule logique $\phi \in \mathcal{L}$, décider si ϕ est vraie dans $\mathcal{M}$.

Une solution à ce problème aurait, comme le lecteur peut imaginer, des implications exceptionnelles. La première conséquence réside dans la possibilité de démontrer automatiquement des théorèmes mathématiques. Si la logique $\mathcal{L}$ est suffisamment expressive, un tel algorithme pourrait conduire à la résolution des grands problèmes mathématiques ouverts. La deuxième conséquence concerne davantage le domaine de l'informatique. Les systèmes logiques ne servent pas uniquement à formaliser les mathématiques, mais aussi à spécifier le comportement des programmes informatiques et à préciser les contraintes à respecter. Ainsi, la résolution du problème de la décidabilité pour certaines logiques permettrait de vérifier automatiquement la correction des programmes informatiques.

En 1957, Church [Chu57] introduisit un problème encore plus ambitieux, à savoir celui de la *synthèse*. L'objective consiste à concevoir un algorithme capable d'accomplir la tâche suivante :

> Étant donné une formule logique $\phi \in \mathcal{L}$ spécifiant le comportement d'un système informatique, construire un système satisfaisant ϕ, ou déterminer qu'un tel système ne peut pas exister.

Les deux objets centraux de cette thèse, les automates sur les mots infinis et les jeux à durée infinie, ont été initialement introduits dans le but d'aborder les problèmes de la

décidabilité et de la synthèse pour certaines logiques [Bü62, Rab69, BL69b].

1.2 Propriétés structurelles des automates

Suite aux découvertes réalisées dans les années 1960, les automates sur les mots infinis (ou ω-automates) sont devenus un domaine à part entière. Une série de résultats fondamentaux ont été obtenus, contribuant ainsi à la maturation de cette théorie. Cependant, d'importants défis dans l'étude des ω-automates ont mis en évidence des lacunes majeures dans notre compréhension de leur structure.

Transformations d'automates. Il existe plusieurs formalismes permettant de définir quels sont les calculs acceptants dans un automate sur les mots infinis, engendrant différents types d'automates qui présentent des propriétés algorithmiques et d'expressivité variées. Une question fondamentale qui se pose est celle de la relation entre ces différents modèles, notamment en ce qui concerne les transformations permettant de convertir un automate d'un type X en un automate de type Y en ajoutant un nombre minimal d'états.

Minimisation d'automates. Contrairement aux automates sur les mots finis, il n'existe généralement pas d'automate minimal canonique permettant de reconnaître un langage ω-régulier donné. En 2010, Schewe démontra que la minimisation des automates déterministes de Büchi est un problème NP-complet [Sch10]. Cependant, cette preuve ne s'applique qu'aux automates avec la condition d'acceptation sur les états. En 2019, Abu Radi et Kupferman montrèrent que la minimisation de certains types d'automates est réalisable en temps polynomial lorsque la condition d'acceptation est définie sur les transitions [AK19]. La question de la minimisation des automates déterministes de Büchi avec la condition d'acceptation sur les transitions demeure ouverte.

1.3 Complexité des stratégies pour des jeux à durée infinie

Le problème de la synthèse est particulièrement intéressant et présente des défis uniques dans le cas des systèmes ouverts qui interagissent avec un environnement (ce que l'on appelle *synthèse réactive)*. L'approche principale pour résoudre ce problème consiste à modéliser l'interaction entre le système et l'environement sous la forme d'un jeu. Une stratégie gagnante dans ce jeu fournirait une implémentation du système tout en garantissant le respect des spécifications. Afin de faciliter la résolution de ces jeux et de simplifier l'implémentation finale, il est impératif que les stratégies utilisées soient aussi simples que possible.

Positionnalité. Les stratégies les plus simples sont celles qui ne considèrent que la position à un instant donné, sans tenir compte de l'historique du jeu. Ces stratégies sont désignées sous le terme de *stratégies positionnelles*. Une question qui se pose est la suivante : étant donné une condition de victoire, est-il possible pour le système de jouer de manière optimale quel que soit le jeu (avec cette condition de victoire) en utilisant des stratégies positionnelles ? Si la réponse est affirmative, on dit que la condition de victoire est positionnelle.

Mémoire. En général, les stratégies positionnelles ne suffisent pas, et les joueurs doivent retenir certaines informations sur le passé du jeu pour effectuer des coups optimaux. On peut représenter ce type de stratégies à l'aide d'automates finis, que l'on appelle des *structures de mémoire*. Comprendre la taille optimale de ces automates et leur structure est fondamental pour concevoir des algorithmes pour la résolution des jeux en aboutissant à des stratégies aussi simples que possible.

2 Contributions

Nous présentons ici les contributions principales de la thèse. Pour une présentation plus détaillée, on pourra se référer à l'introduction générale ainsi qu'aux introductions de chaque chapitre (en anglais). Nous utilisons une terminologie technique afin d'énoncer les résultats de manière précise. Des hyperliens incorporés dans les mots peuvent faciliter la recherche rapide des définitions requises par le lecteur.

▸ **Caractérisation de la positionnalité des langages ω-réguliers.** La contribution que nous considérons comme la plus importante de la thèse consiste en une caractérisation des langages positionnels parmi la classe des langages ω-réguliers (Théorème V.1). Cette caractérisation nous permet de répondre à la plupart des questions ouvertes concernant la positionnalité dans le contexte des langages ω-réguliers : obtenir la décidabilité en temps polynomial, établir l'équivalence de la positionnalité des langages ω-réguliers dans les graphes finis et infinis et montrer les conjectures de Kopczyński et Ohlmann. De plus, nous présentons quelques résultats concernant la positionnalité des langages en dehors de la classe des ω-réguliers (voir Théorèmes V.6, V.7 et V.8).

▸ **Transformations optimales d'automates.** Nous présentons une construction pour transformer des automates de Muller en automates de parité et démontrons un résultat d'optimalité : l'automate ainsi obtenu est le plus petit parmi les automates de parité qui peuvent être obtenus en dupliquant des états de l'automate initial, tout en utilisant la condition d'acceptation la plus simple possible (Théorèmes II.5 et II.6).

Pour formaliser ces résultats, nous introduisons des morphismes d'automates qui capturent la notion de transformation.

▸ **L'ACD : Une nouvelle approche à la structure des automates.** Afin de définir la transformation mentionnée dans le paragraphe précédent, nous introduisons une structure de données appelée la *décomposition en cycles alternants* (ACD, selon son acronyme en anglais). Cette structure permet de résumer les principales propriétés structurelles des automates de Muller. Les applications de l'ACD vont au-delà de la définition des transformations, l'ACD permet également d'obtenir de nombreux résultats concernant les automates sur les mots infinis. Parmi ces résultats figurent notamment :

- Déterminer si la condition d'acceptation d'un automate peut être simplifiée.
- Définir une forme normale pour les automates de parité.
- Démontrer que la minimisation du nombre de couleurs utilisées par un automate de Muller est un problème NP-difficile.

Nous étudions également la complexité du calcul et de l'utilisation de l'ACD.

▸ **Minimisation d'automates.** Nous établissons la complexité du problème de la minimisation de certaines classes d'automates (avec la condition d'acceptation définie sur les transitions) :

- La minimisation d'automates de Rabin déterministes est un problème NP-complet, même pour des automates reconnaissant des langages de Muller (Théorème III.1).
- Il est possible de minimiser les automates de parité déterministes reconnaissant des langages de Muller en temps polynomial (Théorème III.4).
- La minimisation d'automates de Büchi généralisés déterministes est un problème NP-complet (Théorème III.2).
- La minimisation d'automates de Muller déterministes est un problème NP-difficile (Théorème III.3).

▸ **Perspectives sur le déterminisme en histoire.** Le déterminisme en histoire (HD selon son acronyme en anglais) est une propriété d'automates qui a récemment attiré beaucoup d'attention en raison de ses applications potentielles dans le problème de la synthèse réactive. (Pour plus de détails sur ce sujet, le lecteur pourra consulter [Kup22, BL23b]).

Nous utilisons des automates déterministes en histoire tout au long de cette thèse. Cependant, nous ne nous limitons pas uniquement à l'étude du modèle en lui-même, dans le but de comprendre ses propriétés d'expressivité. Au contraire, dans cette thèse les automates HD sont un outil essentiel pour obtenir des résultats théoriques. Nous établissons une équivalence entre la mémoire pour les jeux de Muller et les automates HD de Rabin (Théorème IV.3). De plus, pour obtenir notre caractérisation de la positionnalité dans le Chapitre V, nous faisons un usage intensif des automates HD. Cela est particulièrement remarquable, car bien que les énoncés de nos théorèmes n'incluent pas explicitement les automates HD, les propriétés de canonicité qu'ils confèrent (voir [AK22]) jouent un rôle fondamental dans la réalisation de nos démonstrations. Ces résultats mettent en avant la canonicité des automates HD et soutient leur pertinence pour obtenir des résultats théoriques.

Nous établissons également des résultats sur la taille des automates HD. Nous montrons que les automates de parité HD reconnaissant des langages de Muller ne sont pas plus petits que leur homologues déterministes (Corollaire II.36), ce qui pose des limites importantes à l'applicabilité du déterminisme en histoire pour la synthèse réactive (voir également le Corollaire II.80). Cependant, nous montrons que les automates HD de Rabin reconnaissant des langages de Muller peuvent être exponentiellement plus petits que leur homologues déterministes (Théorème III.5).

▸ **Correspondance mémoire–automate.** Nous établissons une correspondance entre les automates de Rabin et les structures de mémoire pour les jeux de Muller. Plus précisément, les automates de Rabin déterministes correspondent exactement aux mémoires chromatiques pour les jeux de Muller (Théorème IV.1), tandis que les automates déterministes en histoire correspondent aux mémoires générales pour ce type de jeux (Théorème IV.3).

▸ **Automates avec la condition d'acceptation sur les transitions.** La grande majorité de travaux dans la littérature utilisent des ω-automates avec la condition d'accep-

tation sur les états. Dans cette thèse, le choix de placer la condition d'acceptation sur les transitions n'est pas seulement techniquement plus pratique, mais il est également essentiel, car sans cela, la plupart des résultats ne seraient tout simplement pas valides !

Nous croyons que nos contributions apportent des arguments supplémentaires en faveur du fait que les automates avec la condition d'acceptation sur les transitions sont plus canoniques et pratiques. Dans le Chapitre VI, nous présentons une synthèse des résultats qui mettent en évidence le contraste entre les deux modèles, comprenant à la fois des contributions préexistantes dans la littérature et des découvertes réalisées au cours de cette thèse.

Introduction

Outline

1 Automata and games for the synthesis problem

1.1 Motivation: From decidability of logic to program synthesis

The advancements in mathematics in the early 20th century breathed life into the idea of obtaining a logical formalism that would enable to establish the truth value of all mathematical statements. This idea found its highest expression in Hilbert's program, whose ultimate goal was to obtain an automatic procedure (an algorithm) solving the following problem:

> Given a logical formula stating a theorem, decide whether it is true or false.

In what is probably the most important set of results in the entire history of logic, Gödel showed with his incompleteness theorems the impossibility of this ambitious programs [Göd31]. Not all hope was lost, though. Gödel's theorems imply that no algorithm can decide the truth of statements expressed in first order logic in the standard model of arithmetic (natural numbers with the operations of addition and multiplication). But, what about other (weaker) logics? Some positive results were found: the decidability of Presburguer arithmetic, that is, the first order theory of integers without multiplication [Pre29], and the decidability of the first order theory of reals by Tarski [Tar49].

Model checking. A solution to the decidability problem has further implications when analysed from a computer science perspective. Indeed, logics do not only serve to formalise mathematics, they also allow us to describe the behaviour of computer programs and specify properties we want them to satisfy. A program can be modelled as a mathematical structure $\mathcal{P}$; checking whether the program satisfies a property represented by a logical formula ϕ comes down to determining whether ϕ holds over the structure $\mathcal{P}$. This is commonly referred to as the *model checking problem*.

The synthesis problem. As we have already mentioned, model checking plays a crucial role in the formal verification of the correctness of computer programs. In 1957, Church [Chu57] introduced an even more ambitious objective:

> Given a requirement expressed in some logical formalism, construct a system satisfying the requirement, or determine that no such system exists.

The system referred to (also called *controller*) is usually a finite state automaton, but one can think more generally of any computer program. Church stated the problem for systems working on-line (usually called *reactive systems*) whose task is to transform an infinite sequence of inputs predicates $i_0 i_1 i_2 \dots$ into an infinite sequence of outputs predicates $o_0 o_1 o_2 \dots$ on the fly, such that the pair of sequences satisfies the desired requirement. This is known as the *reactive synthesis problem*.

Automata for obtaining decidability results. The topics explored in this thesis originated from the use of automata to study the problem of decidability for some logics. The tight connections between automata and logic were brought to light around the 1960s, in the works of Trakhtenbrot [Tra58, Tra62], Büchi [Büc60] and Elgot [Elg61], where they established the equivalence between automata over finite words and weak monadic second order logic (weak MSO). Extending these ideas, Büchi proved the decidability of monadic second order logic with one successor over the natural numbers (S1S) [Bü62]. For this proof, he introduced the object that will constitute the central subject of study in this thesis: automata over infinite words (see Figure 1 for an example). An extension of S1S of special relevance, as the decidability of many other logics can reduce to this one, is monadic second order logic with two successor (S2S) – equivalent to the MSO theory of the full binary tree. The decidability of S2S, left open by Büchi, was established by Rabin in 1969 [Rab69]. Rabin's proof, known for being extremely intricate,[1,2] develops a theory of automata over infinite trees and introduces new acceptance conditions for automata. In the subsequent years, many advancements in automata theory were driven by the search for a simplified proof of Rabin's result [HR72, GH82, MS84, Muc84, EJ91].

Games come into play. The use of games in logic has a long history. The first appearance of infinite duration games can be traced back to Zermelo's work on (infinite

[1] It is worth including here a fragment from the abstract of Gurevich and Harrington [GH82] to highlight both the importance and intricacy of Rabin's result: "*In 1969 Rabin introduced tree automata and proved one of the deepest decidability results. If you worked on decision problems you did most probably use Rabin's result. But did you make your way through Rabin's cumbersome proof with its induction on countable ordinals?*".

[2] Apparently, not everyone agreed. Citing Büchi [Büc83]: "*The old proof is good enough for me*".

duration) chess [Zer13].[3] These games were formalised in the seminal work of Gale and Stewart [GS53], after which they found multiple applications in descriptive set theory (see for instance [Mos80, Chapter 6]).

The connection between games and automata theory appeared for the first time in the solution of the reactive synthesis problem for specifications given in the logic S1S by Büchi and Landweber [BL69b]. They used a game-theoretic approach suggested by McNaughton: one player represents the environment and provides input predicates, the other player, who represent the system, responds with output predicates, trying to build a sequence satisfying the requirement. A winning strategy for the system in this game yields an implementation for a controller. Since then, the use of games has been ubiquitous for studying the synthesis of reactive systems (see [Tho95, BCJ18]).

After the work of Büchi and Landweber, games became a convenient and frequently used tool for reasoning and elaborating proofs in automata theory [Büc77, Büc83]. In fact, the most significant progress in the simplification of Rabin's proof of the decidability of S2S came through the application of games. A major breakthrough was made by Gurevich and Harrington [GH82][4] by considering games with Muller winning conditions (a family of winning conditions for infinite duration games with its root in the work of Muller [Mul63]). A key step in their approach is to describe "simple" strategies for the winner in those games, more precisely, strategies implemented by a finite state automaton. A further important simplification was made by Emerson and Jutla [EJ91]. For their proof, they obtain one of the first positionality results: the winner of a game using a parity condition can win using a strategy that only depends on the current position and not on the history of the play.

Some logics for specifying the behaviour of programs. We have already mentioned monadic second order logic for its significance in the origins of the theory of automata over infinite words. We briefly discuss two further logical formalisms of particular relevance for their role in the historical development of automata and game-theoretic tools: the modal μ-calculus and linear temporal logic

The modal *μ-calculus*, developed in its modern form by Kozen [Koz83], is a logical formalism notable for its balance between expressiveness and complexity, which makes it well-suited for model checking purposes. Many of the central results in μ-calculus were obtained via automata and game-theoretic methods [SE84, Wal96], in particular, Streett and Emerson showed the equivalence in expressive power between the μ-calculus and automata on infinite trees [SE89]. Moreover, Emerson, Jutla and Sistla obtained a linear-time equivalence between the model checking of μ-calculus, checking the non-emptiness of parity tree-automata and the algorithmic problem of solving parity games [EJS93]. They proved that these problems belong to $\mathsf{NP} \cap \mathsf{coNP}$, and whether they can be solved in polynomial time remains one of the major open questions in the field.

Linear temporal logic (LTL) was proposed by Pnueli in 1977 as a logic for the verification of non-terminating reactive systems [Pnu77]. This logic gained popularity due to its simplicity and proximity to natural language. The synthesis problem for LTL was solved by Pnueli and Rosner [PR89] using an automata-theoretic approach working in

[3] There are many misunderstandings surrounding Zermelo's work on chess. See [SW01] for a discussion and a translation of Zermelo's paper into English.

[4] We remark that their proof is still quite complex. Citing Emerson and Jutla [EJ91]: "*While the presentation is brief, the argument is still extremely difficult, and is probably best appreciated when accompanied by the 40 page supplement of Monk.*"

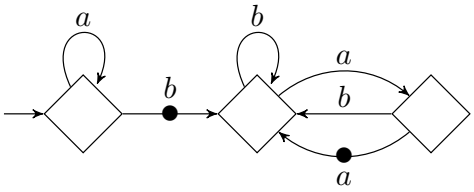
3 steps: (1) translate the LTL formula into an equivalent non-deterministic Büchi automaton on words, (2) transform this automaton into a deterministic Rabin one, and (3) use this automaton to produce a parity game and solve it.[5] The synthesis of controllers for LTL specifications continues to receive substantial attention nowadays – as evidenced by the existence of competitions such as SYNTCOMP [Jac+22] – and the automata-theoretic approach of Pnueli and Rosner is still at the heart of the state-of-the-art synthesis tools [Esp+17, LMS20, MC18, MS17]. Due to the efforts invested in improving these tools, and the quest for a polynomial time algorithm for parity games, game solvers have achieved impressive advancements that allow for the resolution of very large parity games in just a few seconds [Dij18, Jac+22]. However, the limiting factor for LTL synthesis algorithms is the transformation of the LTL formula to a deterministic parity automaton [EKS18, LMS20]. For this reason, synthesis procedures avoiding the construction of deterministic automata have been proposed, for example, via the use of universal coBüchi automata [KV05]. Another important challenge is to improve the quality of the obtained solutions, as for practical applications, the final system must be as small and simple as possible [Kup12].

1.2 A focus on automata

After the discoveries made in the 1960s culminating in Rabin's decidability theorem [Rab69], automata over infinite words (which we simply refer to as ω-automata) became a well-established area of study of independent interest. (We refer to Figure 1 for an example and explanations about automata.) The theory of automata over finite words had proven to be very rich and fruitful, yielding connections between diverse areas of study such as formal languages, logic, and algebra. Mirroring these advances, a series of grounding results on ω-automata shaped the emerging theory: the equivalence between ω-automata and ω-regular expressions [McN66], results about determinisation of automata [McN66, Saf88], or the study of recognisability of ω-regular languages by algebraic structures [Wil91, SPW91]. Additionally, connections between the structure of ω-automata and notions of topological complexity of languages coming from descriptive set theory were explored [Lan69, Wag79], adding a new dimension to the theory. However, major challenges in the study of ω-automata appeared, and a fundamental lack of understanding of their structure became evident.

Canonicity and minimisation of ω-automata. A remarkable property of automata over finite words is that each regular language admits a unique minimal deterministic automaton with a very nice structure: states correspond to residuals of the language (classes of the Myhill-Nerode congruence). However, in the case of infinite words, the automaton of residuals does not suffice to recognise the desired language. In the final quarter of the 20th century, an endeavour was initiated to characterise ω-regular languages through congruences à la Myhill-Nerode and to obtain canonical automata recognising them [Sta83, Saë90, SL94]. A syntactic congruence capturing a given ω-regular language was proposed by Arnold [Arn85], although it does not provide minimal automata. Other canonical – but not minimal – objects recognising ω-regular languages where proposed by Mahler and Staiger [MS97] (namely, families of right-congruences). The quest for canon-

[5] In the original paper [PR89], this last step was presented as checking the emptiness of a Rabin tree-automaton. The approach based on games rapidly took over this presentation [ALW89, Tho91].



✦ **Figure 1.** An automaton is a finite state machine with transitions labelled by letters. An infinite word induces an infinite path in the automaton by following step by step the transitions labelled with the letters of the word. We call such a path a run in the automaton. An automaton can be used to define a set of words (a language) by specifying which runs are accepting and which ones are rejecting. The automaton in the figure uses what is called a Büchi acceptance condition over transitions: some transitions carry a dot, and a run is accepting in the automaton if it visits a dot infinitely often. Therefore, the set of words accepted by this automaton is:

$L =$ Words that contain some b and afterwards infinitely many times the factor aa.

There are many other ways of specifying which runs are accepting (what we call acceptance conditions). We refer to Section I.2 for formal definitions.

ical representations of ω-regular languages remains very much alive to this day [ABF18, AK22, ES22, BL23a].

An algorithm minimising a subclass of ω-automata was first proposed by Gutleben [Gut96],[6] who gave a procedure for the minimisation of automata recognising weak languages, building on work of Staiger and Maler [Sta83, MS97] (a similar algorithm implicitly appears in the work of Maler and Pnueli [MP95]). This algorithm was improved by Löding [Löd01], giving a procedure working in time $\mathcal{O}(n \log n)$. In 2010, Schewe proved that the minimisation of deterministic Büchi automata is NP-complete [Sch10]. That appeared to be a conclusion to the minimisation problem, but was it? In Schewe's paper, the NP-completeness is established for automata with the acceptance condition over states. However, in 2019, Abu Radi and Kupferman showed that some classes of automata (namely history-deterministic coBüchi automata) can be minimised in polynomial time if the acceptance condition is defined over transitions [AK19]. The general problem of the minimisation of automata with transition-based acceptance remains wide open.

A variety of acceptance conditions. The simple Büchi condition (the one used by the automaton in Figure 1) suffices to define all ω-regular languages over infinite words using non-deterministic automata. However, to obtain the same expressive power with deterministic automata, or when using automata over infinite trees, we need to use more complex ways to represent accepting runs. A wide range of acceptance conditions has arisen from the study of ω-automata (Muller [Mul63], Rabin [Rab69], parity [Mos84, EJ91], Emerson-Lei [EL85, Bab+15]...), each one offering its own advantages.

A wide body of literature is devoted to the comparison of these conditions and the study of questions such as their expressive power [McN66, Mos84, KPB95], the possibility of simplifying the acceptance condition of a given automaton [KPB94, KMM06], the efficiency of transformations between the different models and the succinctness gap

[6] Gutleben's paper has almost gone unnoticed in the literature. I thank Christof Löding for sending this paper to me.

between them [Löd98, Löd99, BK12, Bok17, Bok19], or the complexity of decision procedures depending on the condition used [CDK93, KP09, Hug23]. (See [Bok18] for a general overview of all these questions.)

Two acceptance conditions stand out as the most commonly used in modern LTL synthesis tools. Emerson-Lei conditions – which are a concise representation of Muller conditions via boolean formulas – are well-suited for their use due to their succinctness and their proximity with logical formulas. This allows for simple translations from LTL logic [MS17, Maj+19a, DL+22] and the use of methods based on SAT-solvers [BDL15, Cas+22, SSM23]. Parity conditions, on the other hand, allow for the use of the powerful parity game solvers in the final steps of the LTL synthesis process [RDLP20, LMS20, MC18].

The parity condition. Amongst all the acceptance conditions mentioned in the preceding paragraph, one takes a prominent place in both the literature on ω-automata and in this thesis: the parity condition. Parity automata were introduced by Mostowski [Mos84] (under the name of Rabin automata in chain form), and he proved that they suffice to represent all ω-regular languages. Independently, Emerson and Jutla [EJ91] presented the modern formulation of parity conditions and established the paramount result of their bipositionality (also obtained independently by Mostowski [Mos91]). The use of parity conditions was rapidly adopted for multiple reasons outlined below.

We have already discussed the utility of the parity condition for LTL synthesis – mainly due to the existence of high-performing algorithms solving parity games – but its significant role in the theory of ω-automata extends beyond these practical applications. From a theoretical point of view, parity conditions can be considered as the simplest family of conditions that can be used to recognise all ω-regular languages with deterministic automata. Several properties justify the distinguished status of the parity condition:[7,8]

- **The parity hierarchy.** The optimal number of colours needed by a parity automaton to recognise a language L reveals a fundamental information about it, called its parity index. The parity index (sometimes called Mostowski index) yields a strict hierarchy both for deterministic automata over words and for non-deterministic automata over trees [Niw86, Bra98] (and these hierarchies are closely related [KSV96, NW98]). In both cases, this index is a measure of the structural complexity of automata recognising L [Wag79, NW98] and of its topological complexity [Arn+08, Skr13]. The parity index of a language of infinite words represented as a deterministic parity automaton can be computed in polynomial time [CM99], while the problem is NP-complete when the input is a Rabin automaton [KPB95]. Whether we can decide the parity index of a language of infinite trees represented as a non-deterministic parity tree-automaton is a long standing open problem [NW04, CL08].
- **Relation with μ-calculus.** The natural acceptance condition appearing when translating a μ-calculus formula to tree-automata is the parity condition [EJ91]. Moreover, the parity index of a language of trees corresponds to the minimal alternation depth of fixpoint operators of a μ-calculus formula defining it [Niw86].

[7] To enable a precise statement of the properties, the following discussion is quite technical and employs terminology that has not yet been introduced. Hyperlinks on words should help the reader to quickly check the missing definitions.

[8] Some of the arguments shown below are discussed by Niwiński and Walukiewicz [NW98, p.2].

- **Symmetry.** Parity languages are exactly Muller languages corresponding to families $\mathcal{F} \subseteq 2^{\Gamma}_{+}$ of subsets of colours such that both $\mathcal{F}$ and its complement are closed under union [Zie98] (see also Proposition II.103). This allows, for instance, for easy complementation of parity automata.
- **Positionality.** Parity languages are bipositional [EJ91]. Moreover, over infinite game graphs, these are the only bipositional languages [CN06], and over finite game graphs, these are the unique bipositional Muller languages [Zie98].
- **Complexity of parity games.** Solving parity games is both in NP and coNP [EJS93] (more precisely, the problem is in UP ∩ co-UP [Jur98]). They can be solved in quasi-polynomial time [Cal+17], and whether they can be solved in polynomial time is a major open question. This contrasts with the complexity of solving Rabin and Muller games, which is, respectively, NP-complete [EJ99] and PSPACE-complete [HD05].

(We refer to Section VII.2.2 and Question VII.5 in the conclusions for further discussions about the canonicity of the parity condition.)

History-deterministic automata. As discussed earlier, one of the main challenges for an efficient implementation of LTL synthesis algorithms is the translation of non-deterministic automata to deterministic ones. Known determinisation procedures are very costly, and deterministic automata can be exponentially larger than non-deterministic ones. To address this problem, Henzinger and Piterman [HP06] proposed a new intermediate model under the name of good-for-games automata (which we call, following the current terminology, history-deterministic). Similar ideas had been previously investigated by Kupferman, Safra and Vardi [KSV96], and Colcombet studied the notion of history-determinism in the context of the theory of cost functions [Col09].

The definition of history-determinism is rather natural: an automaton is history-deterministic if it can resolve its non-determinism on-the-fly, without guessing the future. The remarkable property of this definition is that it exactly captures the features of deterministic automata that make them suitable for synthesis purposes. A natural question that arises is whether history-deterministic automata can be more succinct that deterministic ones, and, in that case, which languages and automata types can benefit from this succinctness. It was not until several years after the introduction of history-determinism that an example of an ω-regular language for which history-deterministic automata are smaller than deterministic ones was exhibited [KS15]. History-deterministic automata are the focus of several lines of research (we refer to the surveys [Kup22, BL23b] for detailed expositions). Despite this, a complete understanding of history-deterministic automata remains elusive, and their scope of applicability is still uncertain.

State complexity. In this thesis, we study problems such as the transformation of different types of automata and their minimisation. Throughout the entire document, the main focus will be in *state complexity*, that is:

- The parameter we seek to minimise when minimising automata is the number of states.
- The efficiency of a construction is measured by the number of states of the resulting automaton.

The size of the information needed to completely represent an automaton (transitions and the acceptance condition) takes a secondary role for us, although it will be relevant in some parts of the document (mainly Section II.5). This approach is commonly adopted in the literature, and justified by several reasons:

a) In many cases, the size of the whole representation of an automaton is polynomial in the number of states.

b) In the case of automata over finite words, each state stands for a precise information of the recognised language: a residual of it. We expect that, similarly, in the case of infinite words, the states of automata represent and capture relevant information of the language.

c) In certain situations, it is possible to completely characterise the number of states of automata (see for instance [CZ09]). The focus on state complexity allows for elegant statements and tight (non-asymptotic) bounds on the size of automata.

1.3 A focus on strategy complexity in infinite duration games

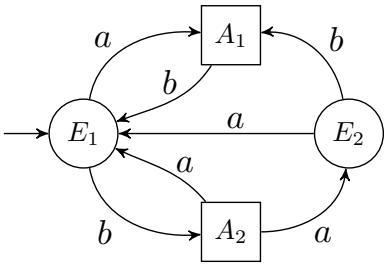
Solving a game consists of determining which player (if any) has a winning strategy, that is, a strategy that ensures a victory regardless of the actions of the opponent. A strategy can be defined as a set of instructions that indicates the player how to act after each possible finite sequence of moves. For algorithmic purposes, a major challenge arises: the set of all sequences of moves is infinite. Therefore, in order to be able to tackle the problem of solving games algorithmically we need to obtain a finite representation of the set of all possible plays, or, at least, to compress the information needed by a player to make the optimal moves. (We refer to Figure 2 for an example and explanations on games.)

In 1958, the criticality of the question of the computability of strategies was made evident by Rabin, who showed that there are games in which one player can force a victory, but for which no computable winning strategy exists [Rab58]. To provide a solution to the synthesis problem with specifications in MSO, Büchi and Landweber proved that in every game with an ω-regular winning condition there is a player with a winning strategy that can be implemented by a finite-state automaton [BL69b].[9] In their paper, they suggest the following type of problem [BL69b, p.299]:

> Given a class of games $\mathcal{C}$, how simple winning strategies do exist for games in $\mathcal{C}$?

Complexity of strategies takes on special relevance in the context of synthesis (see [Tho95]). First, the efficiency of algorithms solving a class of games will depend on the complexity of the strategies required to win in those games. Perhaps more importantly, the simplicity of the controller synthesised from the winning strategy directly depends on the simplicity of that strategy.

[9] The fact that games with ω-regular objective are determined (that is, one of the players can force a win) can be seen as a corollary of Martin's determinacy theorem for Borel objectives [Mar75], as ω-regular languages are Borel. However, Martin's theorem is posterior to Büchi and Landweber's result. Still, at that time, Davis [Dav64] had already obtained the determinacy of Σ^0_3-objectives, which is sufficient to imply that of ω-regular objectives.



✦ **Figure 2.** We consider games played on a graph facing two players: Eve, controlling the circles, and Adam, controlling the squares. They take turns in moving a token along the edges of the graph. Each time the token traverses an edge, a letter is produced (a or b, in this case). This process produces an infinite word w, used to determine the winner. The winning condition (also called objective) is given as a language W of infinite words. Eve wins if the produced word w belongs to W, and Adam wins otherwise. Consider the game above with the winning condition:

$$W = \text{Words that contain the factor } aa \text{ infinitely often.}$$

We show how Eve can force a victory in this game. Whenever the token arrives at E_1, she first goes down to A_2; the next time the token arrives at E_1, she takes the a-transition going to A_1. From E_2 she always takes the a-transition to E_1. Repeating this process guarantees producing aa infinitely often. However, to set up this strategy, she needs to retain some information, she cannot make a decision based solely on the current position, that is, she cannot win positionally.

Positionality. The simplest kind of strategies are positional ones, that is, those whose choices depend exclusively in the current position, and not in the history of the play. Given a language of infinite words W, a natural question is whether players can play optimally using positional strategies in games with W as winning condition. If both players can do so, we say that W is bipositional; if only Eve ("the system") can, we say that W is half-positional.

Bipositionality of various objectives was established since the middle of the 20th century: simple stochastic games by Shapley [Sha53], finite mean-payoff games by Ehrenfeucht and Mycielski [EM79], parity games by Emerson and Jutla [EJ91] (and independently by Mostowski [Mos91]) or finite energy games by Bouyer et al. [Bou+08].[10] The existence of positional strategies for both players implies that solving the previous games is in $\mathsf{NP} \cap \mathsf{coNP}$, and all existing algorithms solving them strongly rely on their bipositionality (the bibliography on these games is overwhelming, we refer to [Fij+23] for an overview). At the beginning of this century, there was an increasing interest in obtaining a precise understanding of positionality. Two major results characterising bipositionality were obtained. Gimbert and Zielonka [GZ05] completely characterised bipositionality over finite game graphs, and Colcombet and Niwiński [CN06] established that the only bipositional objectives over all game graphs were parity objectives (under an hypothesis of prefix-independence).

Nevertheless, for applications in synthesis, half-positionality is arguably a more compelling concept, as we are only interested in Eve's strategies to build a controller. However, advancing in the comprehension of half-positionality has proven challenging despite many

[10] Interestingly, parity games were shown to be linear-time reducible to mean-payoff games and energy games [Pur95], which are in turn reducible to simple stochastic games [ZP96].

efforts. One of the first non bipositional objectives proven to be half-positional were Rabin languages [Kla94], and Zielonka showed that these are the only half-positional objectives within the class of Muller languages [Zie98]. Other objectives have been proven to be half-positional, for example finitary Büchi [CF13], cost-parity [FZ14] or mean-payoff over infinite graphs [Ohl23b]. The first detailed investigation on half-positionality was conducted by Kopczyński in his PhD thesis [Kop08]. The driving question behind much of his research was: Are half-positional prefix-independent objectives closed under union? Recently, this question was answered negatively by Kozachinskiy [Koz22a] (in the case of finite game graphs and non ω-regular conditions). Kopczyński proposed various sufficient conditions for half-positionality, but they do not constitute a characterisation. His research focused on prefix-independent objectives; generalisations of some of his results to all objectives were further explored by Bianco et al. [Bia+11].

In a recent breakthrough, Ohlmann gave a characterisation of half-positionality by means of monotone universal graphs [Ohl23a]. Universal graphs (in the context of infinite duration games) were first introduced by Colcombet and Fijalkow for the study of algorithms for parity games [CF18, CF19]. A universal graph for an objective W contains, in some sense, all possible ways a strategy can interact with W in a game. Ohlmann's theorem establishes that an objective is half-positional if and only if it admits universal graphs equipped with a well-order with some further monotonicity properties. This provides a powerful tool to obtain half-positionality of many objectives. However, Ohlmann's characterisation is not constructive, and does not directly yield decidability results.

Memory for games. As already mentioned, the solution to the synthesis problem involved proving that strategies implemented by finite automata were sufficient in games with ω-regular conditions [BL69b]. We call such automata *memory structures*, and refer to their size as the memory used by the strategy. Gurevich and Harrington [GH82] (and independently Büchi [Büc83, Section 12]) went a step further: they explicitly describe memory structures implementing winning strategies in games using Muller conditions.[11] This understanding of the structure of strategies is an essential ingredient in McNaughton's algorithm for solving Muller games [McN93]. However, the memory structures proposed did not have an optimal size in general. A landmark result in the study of memory for games is the characterisation of the precise memory requirements of Muller conditions by Dziembowski, Jurdziński and Walukiewicz [DJW97], based on the analysis of the Zielonka tree of the condition (a combinatorial structure introduced by Zielonka under the name of split-tree [Zie98]). This result was extended by Horn by allowing stochasticity in games [Hor09]. To the best of the author's knowledge, the only further characterisation of the memory requirements of a class of languages was that of closed objectives by Colcombet, Fijalkow and Horn [CFH14] (extended recently to the model of chromatic memory, both for closed and open objectives [Bou+23]).

In recent years, there has been a renewed interest in questions related to memory for games. In the context of his PhD thesis [Van23], Vandenhove and coauthors have extended the results of Gimbert and Zielonka [Bou+20] and Colcombet and Niwiński [BRV23] to the setting of finite memory determinacy. As a consequence of that work, they characterised ω-regular languages as the only ones for which both players can play optimally

[11] Büchi [Büc83] attributes the idea behind the structure appearing in this work (the *later appearance record*) to McNaughton [McN66]. (However, for a less sharp reader it is not at all clear where to find it in McNaughton's paper.)

with finite memory in all games.[12] In their work, there is a focus on understanding when the analysis of the memory requirements in one-player games suffices to deduce results for two-player games (results encompassed under the name of 1-to-2-players lifts). This question has been further explored by Kozachinskiy [Koz22c], who has also made contributions to the understanding of the differences between various models of memory [Koz22d, Koz22b] (see Chapter IV for more details). Memory for games has also been studied in other settings, as concurrent games [BRT22] or timed games [MPR21].

Understanding the complexity of strategies in certain contexts remains the missing piece for establishing the decidability of some logics. For instance, the decidability of cost-MSO over infinite trees (which would imply the decidability of determining the parity index of languages of non-deterministic tree-automata [CL08]) would follow from the conjectured existence of optimal finite memory strategies for some games [Col13, Section 9.1].

2 Contributions and organisation

2.1 Division into chapters

The thesis is organised in 6 chapters (the first of them consisting in preliminary definitions). We have structured the chapters based on thematic considerations, resulting in 4 main research axes:

- Transformations and structural properties of Muller automata (Chapter II).
- Minimisation of automata (Chapter III).
- Memory for games (Chapter IV).
- Positionality (Chapter V).

The last chapter (Chapter VI) should be treated as an addendum, in which we discuss a particular aspect of our model: the use of transition-based acceptance.

This arrangement has resulted in chapters with imbalanced lengths. Chapters II and V are notably longer, and encompass most of the technical content of the thesis. On the contrary, Chapter IV is comparatively short, as many of its contributions can be easily obtained from the the previous technical work.

At the end of each chapter's introduction, we credit people that have collaborated in deriving the results of the chapter and give information about related publications.

2.2 Contributions

We present the primary contributions of this thesis at a high level. For a more detailed overview of the contributions related to each specific topic, we refer to the introductions of individual chapters.

[12] The characterisation is a bit more subtle, as players are required to have a fixed memory structure that allows them to play optimally in all games (what we call an arena-independent memory). The question of whether there are non ω-regular objectives for which both players can play optimally with finite (non-chromatic) memory is open.

Contribution 1. Characterisation of half-positionality for ω-regular languages

The contribution we deem to be the most important of the thesis is a complete characterisation of half-positional ω-regular languages (Theorem V.1). This allows us to answer most open questions about half-positionality in the context of ω-regular languages: decidability in polynomial time, existence of finite-to-infinite and 1-to-2-player lifts and proofs of conjectures by Kopczyński and Ohlmann. Moreover, we give a characterisation of bipositionality for all objectives (Theorem V.6), and characterise half-positionality for some languages beyond ω-regular ones, such as closed and open languages (Theorems V.7 and V.8).

Chapter V is devoted to the statement and proof of these results.

Contribution 2. Optimal transformations of automata

We introduce a construction for transforming Muller automata into parity automata. We provide very strong optimality guarantees: in all cases, the resulting automaton has a minimal number of states and uses a minimal number of priorities amongst automata that can be obtained by duplicating states of the input automaton (Theorems II.1, II.2, II.5 and II.6). Similar transformations are given for obtaining Rabin automata (Theorems II.4 and II.7).

To formalise the statement of these optimality results, we introduce various notions of morphisms of automata, capturing different kinds of transformations and granting the preservation of different semantic properties.

These transformations and their optimality are presented in Sections II.3 and II.4 of Chapter II. Morphisms for automata are defined in Section II.2.

Contribution 3. The ACD: Understanding the structure of automata

To define the aforementioned transformations, we introduce a data structure named the *alternating cycle decomposition* (ACD). This is a generalisation of the Zielonka tree, encapsulating many of the structural properties of Muller automata. Its applications go beyond the definition of transformations, as it can be used to derive many results about ω-regular automata. Some of them are:

- Deciding when a Muller automaton can be relabelled with a simpler acceptance condition (typeness results) (Section II.6).
- Defining a normal form for parity automata (Section II.7).
- Showing that the minimisation of colours for Muller automata is NP-hard (Section II.6.3).

We also study the complexity of computing and using the alternating cycle decomposition (Section II.5).

The ACD provides a fresh presentation of the structural properties of automata stemming from the work of Wagner [Wag79]. We believe that this structural perspective represents a fruitful approach to reasoning about automata, as proven by the numerous advances in the theory building upon Wagner's work since the 1980s. The insights gained from the ACD are the basis of most of the contributions in the thesis.

Contribution 4. Minimisation of transition-based automata

We establish the complexity of the minimisation of some classes of (transition-based) automata:

- The minimisation of deterministic Rabin automata is NP-complete, even for automata recognising Muller languages (Theorem III.1).
- Deterministic parity automata recognising Muller languages can be minimised in polynomial time (Theorem III.4).
- The minimisation of deterministic generalised (co)Büchi automata is NP-complete (Theorem III.2).
- The minimisation of deterministic Muller automata is NP-hard (Theorem III.3).

These results are proven in Chapter III.

Contribution 5. Correspondence memory–automata

We show a tight correspondence between different types of memories for Muller games and Rabin automata: chromatic memories correspond to deterministic Rabin automata (Theorem IV.1) and general memories correspond to good-for-games Rabin automata (Theorem IV.3). These contributions appear in Chapter IV.

The two following contributions span across every chapter.

Contribution 6. History-deterministic automata: A natural model for the study of ω-regular languages

Throughout the entire thesis, history-deterministic (HD) automata play a central role. However, our focus is not only on the model itself for the sole purpose of understanding its expressiveness or succinctness properties; instead, we make an instrumental use of it. In Chapter V we extensively employ HD automata to obtain our characterisation of half-positionality. This is quite striking, as the statements of our theorems do not need to mention HD automata; nevertheless, the canonicity properties they offer (see [AK22]) are fundamental to execute our proofs. Also, in Chapter IV we characterise memory for games by means of HD automata (Theorem IV.3). We believe that these contributions reinforce the role of HD automata as a natural and fundamental model and highlight their suitability for obtaining theoretical results.

Along the path, we also provide succinctness results showing that for some subclasses of languages, history-deterministic automata can (or cannot) be smaller than deterministic ones. We show that HD parity automata recognising Muller languages are not more succinct than deterministic ones (Corollary II.36), and set some important limits for the applicability of HD automata in the context of reactive synthesis (Corollary II.80). On the more positive side, we show that HD Rabin automata recognising Muller languages can be exponentially more succinct than deterministic ones (Theorem III.5).

Contribution 7. Transition-based vs state-based automata

Most works in the literature use ω-automata with the acceptance condition defined over states (although in recent years this trend is changing). In this thesis, the use of transition-based acceptance is not only technically more practical, but without its use most results would simply not hold! We believe that our contributions provide further evidence to the fact that transition-based automata are more canonical and practical.

The way in which the use of this model is critical to obtain different results is indicated in each chapter.

In Chapter VI we include a review of results – both existing from the literature and novel findings obtained during this thesis – in which the contrast between the two models is evident.

3 Reading tips and conventions

Non-linear reading and dependencies between chapters. This thesis – mainly due to its length – is not really well-suited for linear reading from start to finish (but the reader is very welcome to do so if they wish!). On the contrary, the focus has been on providing a comprehensive analysis of the topic under study in each part, including motivations and all necessary technical details.

For this reason, we have taken special care to facilitate the independent reading of the chapters. Nevertheless, some dependencies are unavoidable – given that establishing new connections among different areas is a central theme in the thesis – and the proofs of our theorems often rely on results from previous chapters. At the beginning of each chapter, we explicitly mention the notions and results previously introduced that will be used in that chapter. Also, the use of hyperlinks should help the reader to quickly see the definition of any notion, which facilitates a non-linear reading. These two features should also help the reader to know what parts of the preliminaries can be skipped.

Despite the comment above, this thesis constitutes a coherent document as a whole. There are some aspects that are best appreciated with a global view of the document, primarily the contributions related to: history-deterministic automata, the superiority of transition-based models, and, in general, the understanding of structural properties of ω-automata. These aspects are relevant across all chapters.

Use of appendices. Some of the content of the thesis is included in appendices at the end of the document. This has been done mainly for stylistic reasons. As some proofs are quite lengthy and technical, we have considered that placing them in appendices might help to highlight the contributions and the main proof ideas. We always provide intuitive explanations in the main body. The content in the appendix has received the same level of attention as the content in the main body, and we consider this material as an integral part of the thesis.

We note that the bibliography appears after the appendices (which might be unconventional, and even go against certain style guides). There are two reasons for this choice: first, as said before, the material in the appendix is an integral part of the thesis, and we use citations in there just like in any other part; secondly, we find it more convenient for the reader to have the list of references at the very end, making it easy to locate them quickly.

Numbering conventions. Except for theorems and conjectures, that are numbered independently, all environments (lemmas, propositions, definitions, examples...) are numbered together to facilitate the navigation in the document.[13] To avoid ambiguity, each identifier is preceded by the chapter in which it appears, in roman numerals (for instance, Proposition I.18).

[13] This has a drawback: the numbering reaches some embarrassing values, e.g. Proposition II.137.

Environments. Important results, definitions and examples are framed in boxes of different colours.[14] We note that lemmas are enclosed in boxes only occasionally; lemmas that we consider to be relevant results on their own are framed, while small technical results that are only used locally are not. We comment some particularities on the use of some environments.

Theorem (Theorem environment)**.**

This is a theorem. The theorem environment is employed exclusively to state original contributions of the thesis;[15] important results appearing in the literature are stated as propositions (and appropriate references are always included).

Conjecture (Conjecture environment)**.**

This is a conjecture (or a question). This environment is employed exclusively to state questions that are not solved within the thesis. These questions might be original or come from the literature; in the latter case, citations are always included.

Global hypothesis.

This environment is employed to state assumptions made throughout a whole chapter or section. We try to minimise its use, as it is in general preferable to explicitly state all hypothesis at each point. However, we do make use of it to a certain extent in Chapter V to improve readability.

This environment is used to emphasise informal questions and general research objectives.

Colours. We use colours in figures, but they are never essential for understanding the ideas we aim to convey. Whenever necessary, we ensure redundancy by supplementing the colour code with other visual markings.

Links between a notion and its definition. The design of this thesis is optimised for reading in an electronic device. Mainly, the document contains many, many, *hyperlinks*. Each occurrence of a notion (which we lightly colour in dark blue) is linked to its *definition*. On an electronic device, the reader can click on words or symbols (or just hover over them on some PDF readers) to see their definition.[16]

[14] I thank Pierre Ohlmann and Olivier Serre for sharing the LaTeX style document used as starting point for formatting the document of this thesis, and Rémi Morvan for his stylistic advice.

[15] Of course, there are high chances that some of these results already appear in some form in papers from the 20th century that have passed unnoticed to the author.

[16] Thanks to Thomas Colcombet, Rémi Morvan and Aliaume Lopez for developing the `knowledge` package and its nifty companion, the `knowledge-clustering` tool, which allow for the ~~not so~~ simple implementation of these very useful hyperlinks.

General preliminaries

Outline

Notational conventions

Sets and infinite words

For a set A we let $|A|$ denote its cardinality, 2^A its power set and $2^A_+ = 2^A \setminus \{\emptyset\}$. If B is a subset of A, and A is clear from the context, we note $\overline{B} = A \setminus B$. For a family of subsets $\mathcal{F} \subseteq 2^A$ and $A' \subseteq A$, we write $\mathcal{F}|_{A'} = \mathcal{F} \cap 2^{A'}$. We write $A \sqcup B$ to denote the union of two sets such that $A \cap B = \emptyset$. For natural numbers $i \leq j$, $[i, j]$ stands for $\{i, i+1, \ldots, j-1, j\}$.

For a set Σ, a word over Σ is a sequence of elements from Σ. An *ω-word* (or simply an *infinite word*) is a word of length ω. The sets of finite and infinite words over Σ will be written Σ^* and Σ^ω, respectively, and we let $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. Subsets of Σ^* and Σ^ω will be called languages (or *objectives*, in a context of games). For a word $w \in \Sigma^\infty$ we write w_i to represent the i^{th} letter of w. For $w = w_0 w_1 \ldots w_k \in \Sigma^*$ we write $\mathsf{first}(w) = w_0$ and

$\mathsf{last}(w) = w_k$. We let ε denote the *empty word*, and let $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. The concatenation of two words $u \in \Sigma^*$ and $v \in \Sigma^\infty$ is written $u \cdot v$, or simply uv. If $u = v \cdot w$ for $v \in \Sigma^*, u, w \in \Sigma^\infty$, we say that v is a *prefix* of u and we write $v \sqsubseteq u$. We write $v \sqsubset u$ if $v \sqsubseteq u$ and $v \neq u$. If $u = v_1 v_2 \cdot w$ for $v_1, v_2 \in \Sigma^*, u, w \in \Sigma^\infty$, we say that v_2 is a *factor* of u. For a word $w \in \Sigma^\omega$, we let

$$\mathsf{Inf}(w) = \{a \in \Sigma \mid w_i = a \text{ for infinitely many } i \in \mathbb{N}\}.$$

For $\Sigma' \subseteq \Sigma$ and a language $L \subseteq \Sigma^\infty$, the restriction of L to Σ' is $L|_{\Sigma'} = L \cap \Sigma'^\infty$.

We say that a language $L \subseteq \Sigma^\omega$ is *prefix-independent* if for all $w \in \Sigma^\omega$ and $u \in \Sigma^*$, $uw \in L$ if and only if $w \in L$.

Given a map $\alpha : A \to B$, we will extend α to words component-wise without explicitly mentioning it, i.e., $\alpha : A^\infty \to B^\infty$ is defined as $\alpha(w_0 w_1 w_2 \dots) = \alpha(w_0)\alpha(w_1)\alpha(w_2)\dots$. If $A' \subseteq A$, we note $\alpha|_{A'}$ the restriction of α to A'. We let Id_A be the identity function on A. We write $\alpha : A \rightharpoonup B$ if α is a *partial mapping* (it is defined only over some subset of A).

Graphs

A *graph*[1] is a tuple $G = (V, E, \mathsf{source}, \mathsf{target})$ where V is a set of vertices, E a set of edges and $\mathsf{source} \colon E \to V$ and $\mathsf{target} \colon E \to V$ are maps indicating the source and target for each edge. We may omit the maps source target in the description of a graph if they are clear from the context. A *path* is a (finite or infinite) sequence $\rho = e_0 e_1 \ldots \in E^\infty$ such that $\mathsf{source}(e_i) = \mathsf{target}(e_{i-1})$ for all $i > 0$. For notational convenience, we write $v_0 \xrightarrow{e_0} v_1 \cdots \xrightarrow{e_{n-1}} v_n$ to denote a finite path from $v_0 = \mathsf{source}(e_0)$ to $v_n = \mathsf{target}(e_{n-1})$ and we let $\mathsf{source}(\rho) = v_0$ and $\mathsf{target}(\rho) = v_n$. For $A \subseteq V$, we let $\mathcal{Path}_A^{\mathsf{fin}}(G)$ and $\mathcal{Path}_A(G)$ denote, respectively, the set of finite and infinite paths on G starting from some $v \in A$ (we omit the subscript if $A = V$). We let $\mathcal{Path}_A^\infty(G) = \mathcal{Path}_A^{\mathsf{fin}}(G) \cup \mathcal{Path}_A(G)$. For a subset of vertices $A \subseteq V$ we write:

- $\mathsf{In}(A) = \{e \in E \mid \mathsf{target}(e) \in A\}$,
- $\mathsf{Out}(A) = \{e \in E \mid \mathsf{source}(e) \in A\}$.

A graph is *strongly connected* if there is a path connecting each pair of vertices. A *subgraph* of $(V, E, \mathsf{source}, \mathsf{target})$ is a graph $(V', E', \mathsf{source}', \mathsf{target}')$ such that $V' \subseteq V$, $E' \subseteq E$ and source′ and target′ are the restrictions of source and target to E', respectively. A *strongly connected component* (SCC) is a maximal strongly connected subgraph. We say that a SCC is *final* if there is no edge leaving it. We say that a vertex v is *recurrent* if it belongs to some SCC, and that it is *transient* on the contrary. A *sink* is a vertex with no outgoing edges.

A *pointed graph* is a graph together with a non-empty subset of *initial vertices* $I \subseteq V$. We say that a vertex v is *accessible* (or *reachable*) from a vertex v_0 if there exists a finite path from v_0 to v. We say that a vertex v of a pointed graph is *accessible* if it is accessible from some initial vertex. A set of states $B \subseteq V$ is *accessible* if every state $v \in B$ is accessible. The *accessible part* of a pointed graph is the set of accessible states. We define analogously the *accessible part from a vertex* v_0.

[1] In this work, we will use the term *graph* to denote what is sometimes called a *directed multigraph* (edges are directed, and multiple edges between two vertices are allowed).

1 Games

1.1 Games and strategies

Games on graphs. A *game* is an edge-coloured graph together with a set of winning sequences of colours and a partition of the vertices into those controlled by a *player* named *Eve* and her opponent, named *Adam*. Formally, given a pointed graph[2] G with vertices V and edges E, a *game* over G is a tuple $\mathcal{G} = (G, V_{\text{Eve}}, V_{\text{Adam}}, \Gamma, \mathsf{col}\colon E \to \Gamma \cup \{\varepsilon\}, W)$, with $V = V_{\text{Eve}} \sqcup V_{\text{Adam}}$ and $W \subseteq \Gamma^\omega$. We call G the *underlying graph* of $\mathcal{G}$, I its set of *initial vertices* and W its *winning condition* (or *objective*). We say that $\mathcal{G}$ is an *W-game* if it uses W as winning condition.[3] We allow *ε-edges* (that we also call *uncoloured*) – edges e such that $\mathsf{col}(e) = \varepsilon$ – but we impose the condition that no infinite path in G is composed exclusively of ε-edges. For technical convenience, we make the assumptions that G contains no sink.

An *Eve-game* is a game $\mathcal{G}$ in which all the vertices are controlled by Eve, that is, $V = V_{\text{Eve}}$. A game is *ε-free* if it does not contain any uncoloured edge.

Unless stated otherwise, we take the point of view of player Eve; expressions as "winning" will implicitly stand for "winning for Eve", and strategies will be defined for her.

Plays. In a game, a pebble in placed in an initial vertex, which players move from one vertex to another for an infinite amount of time. The player who owns the vertex v where the pebble is placed chooses an edge $e \in \mathsf{Out}(v)$ and the pebble travels through this edge to its target, producing colour $\mathsf{col}(e)$. In this way, they produce a path $\rho = e_0 e_1 e_2 \cdots \in \mathcal{P}\mathit{ath}^\infty(G)$, that we call a *play*. We denote $\mathsf{col}(\rho)$ the sequence of colours labelling ρ omitting the ε labels (we remark that, if ρ is infinite, $\mathsf{col}(\rho) \in \Gamma^\omega$, since there are no cycles entirely labelled by ε). We refer to $\mathsf{col}(\rho)$ as the *output* of play ρ. We say that a play ρ is *winning* (for Eve) if $\mathsf{col}(\rho) \in W$ and that it is *losing* (or *winning for Adam*) on the contrary. A *subplay* ρ' of ρ is a strict prefix of it: $\rho' \sqsubset \rho$.

Strategies. A *strategy* (for Eve) is a partial function $\mathsf{strat} : \mathcal{P}\mathit{ath}^{\mathsf{fin}}(\mathcal{G}) \rightharpoonup E$, defined for finite plays ending in a vertex in V_{Eve} that tells Eve which move to choose after any possible finite play. We say that a play $\rho \in \mathcal{P}\mathit{ath}^\omega(\mathcal{G})$ is *consistent with the strategy* strat if after each subplay $\rho' \sqsubset \rho$ ending in a vertex controlled by Eve, the next edge in ρ is $\mathsf{strat}(\rho')$. We say that strategy strat is *winning from* $v \in V$ if all infinite plays starting in v consistent with strat are winning. Strategies for Adam are defined symmetrically.

Winning regions. We say that Eve *wins* the game $\mathcal{G}$ from v if she has a strategy that is winning from v. Given a game $\mathcal{G}$ with $I \subseteq V$ as set of initial vertices, the *winning region* of $\mathcal{G}$, written $\mathcal{W}\mathit{in}_{\text{Eve}}(\mathcal{G})$, is the set of initial vertices $v \in I$ such that Eve wins $\mathcal{G}$ from v. We say that a strategy is *optimal (for Eve)* if it is winning from $\mathcal{W}\mathit{in}_{\text{Eve}}(\mathcal{G})$.

[2] We remark that in our definition, we require G to be a pointed graph, that is, we specify a set of initial vertices I. This is to be interpreted as follows: these are the vertices that may be chosen to initialise the game; we will be interested in plays starting from some vertex in the set I. In most cases, I will be either a singleton, or all of V (see also Lemmas II.19 and II.20).

[3] As it will be the case in multiple future definitions, the set of colours Γ is of key importance when studying properties of the objective W. As in most cases it will be clear from the context, we do not include it in the notations.

✦ Remark I.1. *Eve always has an optimal strategy, that is, there is a strategy* $\mathsf{strat}_{\mathsf{opt}}$ *that is winning from* v *for every* $v \in \mathcal{W}in_{\mathrm{Eve}}(\mathcal{G})$.

The *full winning region* of $\mathcal{G}$ for Eve is her winning region in the game $\mathcal{G}_V$, obtained by setting all vertices to be initial; that is, the set of vertices $v \in V$ such that Eve wins the game $\mathcal{G}$ from v.

We will use the expression *solve a game* $\mathcal{G}$ to refer to the algorithmic problem of determining the winning region of $\mathcal{G}$.

Determinacy. We say that a game $\mathcal{G}$ is *determined* if from any initial vertex v, either Eve or Adam have a winning strategy from v. In this work, all games will be determined, as by Martin's theorem [Mar75] games using Borel objectives are determined, and all objectives that we will consider are Borel.

Graphical representation of games. We use circles to represent vertices controlled by Eve and squares to represent those controlled by Adam. We will allow ourselves to consider games with edges labelled by a finite word $w = w_1 w_2 \dots w_n \in \Sigma^*$. Formally, such transitions will stand for a sequence of n transitions, with $n-1$ intermediate vertices. We represent this kind of transitions by a wiggly arrow. We will also use this notation for infinite words: for $w \in \Sigma^\omega$ we write $v \overset{w}{\rightsquigarrow}$ for an infinite sequence of edges labelled with the letters of w starting from v. In this case, the resulting game graph is necessarily infinite.

1.2 Strategy complexity: Positionality and memory

Positionality. We say that a strategy $\mathsf{strat} : \mathcal{P}ath^{\mathsf{fin}}(\mathcal{G}) \rightharpoonup E$ is *positional* if there exists a mapping $\sigma : V_{\mathrm{Eve}} \to E$ such that for every finite play ρ ending in a vertex controlled by Eve we have:

$$\mathsf{strat}(\rho) = \sigma(\mathsf{last}(\rho)).$$

That is, a strategy is positional if the choice of the next transition only depends on the current position, and not on the history of the path.

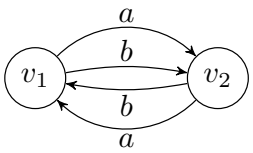
We say that Eve (resp. Adam) can *win positionally* from a subset $A \subseteq V$ if there is a positional strategy $\mathsf{strat}_{\mathsf{pos}}$ that is winning from any vertex in A. We say that Eve (resp. Adam) can *play optimally in* $\mathcal{G}$ *using a positional strategy* if she can win positionally from her winning region.

An objective $W \subseteq \Gamma^\omega$ is *half-positional* if for every W-game, Eve can play optimally using positional strategies.[4] We say that W is *bipositional* if both W and $\overline{W}$ are half-positional, or, equivalently, if both Eve and Adam can play optimally using positional strategies in W-games. If $\mathcal{X}$ is a subclass of W-games (notably, finite, ε-free and Eve-games), we say that W is *half-positional over* $\mathcal{X}$ *games* if for every W-game in $\mathcal{X}$, Eve can play optimally using positional strategies. The same terminology is used for bipositionality.

✦ Remark I.2. *Our notion of positionality uses what sometimes are called* uniform strategies*, that is, we require that a single positional strategy suffices to win independently of the*

[4] As in other definitions, the notion of half-positionality depends not only on the set W, but also on the set of colours Γ. As the set of colours will always be clear from the context, we omit including Γ in the notations.

initial vertex. This notion is strictly stronger than the non-uniform version in which we allow to use different strategies depending on the initial vertex. Said differently, Remark I.1 does not hold if we require strategies to be positional. See Figure 3 for an example.



✦ **Figure 3.** Consider the game above, where Eve controls both vertices v_1 and v_2. Let $W = ab(a+b)^\omega$ be the winning condition of the game, that is, Eve wins if the play starts by ab. She has two positional strategies strat_1 and strat_2 winning from v_1 and v_2, respectively. However, no positional strategy is winning from the entire winning region $\{v_1, v_2\}$.

Memory structures. A *memory skeleton over a set X* is a tuple $\mathcal{M} = (M, m_0, \mu)$ where M is a set of *memory states*, $m_0 \in M$ is an *initial memory state* and $\mu : M \times X \to M$ is an *update function*. We extend the function μ to sequences in X^* by induction: $\mu(m, \varepsilon) = m$, and $\mu(m, x_0 \dots x_{n-1}x_n) = \mu(\mu(m, x_0 \dots x_{n-1}), x_n)$.

A *memory structure for a game $\mathcal{G}$* with set of edges E is a memory skeleton over E together with a *next-move function* $\mathsf{next\text{-}move} : V_{\mathrm{Eve}} \times M \to E$. Such a memory structure *implements a* strategy $\mathsf{strat}_{\mathcal{M}}$ as:

$$\mathsf{strat}_{\mathcal{M}}(\rho) = \mathsf{next\text{-}move}(\mathsf{target}(\rho), \mu(m_0, \rho)), \quad \text{for all } \rho \in \mathcal{P}\mathit{ath}^{\mathsf{fin}}(\mathcal{G}) \text{ ending in } V_{\mathrm{Eve}}.$$

The *size* of a memory structure is the cardinality of its set M of states.

Memory requirements. We say that Eve (resp. Adam) can *play optimally in $\mathcal{G}$ using memory k* if she (resp. he) has a strategy strat_k implemented by a memory structure of size at most k that is winning from every vertex in her winning region Once again, we highlight the use of *uniformity* of strategies in this definition.

We define the *memory requirements* of an objective $W \subseteq \Gamma^\omega$, as the least integer k such that Eve can play optimally in W-games using memory k. We remark that an objective W is half-positional if and only if its memory requirements equal 1.

2 Automata

2.1 Automata over infinite words

Non-deterministic automata. A *(non-deterministic) automaton* over the alphabet Σ is a tuple $\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta, W)$, where Q is a set of states, Σ is a set of letters called the *input alphabet*, $I \subseteq Q$ is a non-empty set of *initial states*, Γ is a set of *output colours*, $\Delta \subseteq Q \times \Sigma \times \Gamma \times Q$ is a set of *transitions* and $W \subseteq \Gamma^\omega$ is a set of infinite sequences of colours, called the *acceptance set*.

We write $q \xrightarrow{a:c} q'$ to denote the transition $(q, a, c, q') \in \Delta$, and $q \overset{w:u}{\rightsquigarrow} q'$ to represent the existence of a path from q to q' labelled with the input letters $w \in \Sigma^*$ and output colours $u \in \Gamma^*$.

The *underlying graph* of $\mathcal{A}$ is the pointed graph having Q as set of vertices, Δ as set of edges and I as set of initial vertices. We refer to the tuple $(\mathsf{col}, \Gamma, W)$ as the *acceptance condition* of $\mathcal{A}$. For $a \in \Sigma$ and $q \in Q$, an *a-self loop* over q is a transition $q \xrightarrow{a:c} q$.

For a state $q \in Q$, we write $\mathcal{A}_q$ to denote the automaton obtaining by setting $I = \{q\}$.

We define:

$$\delta(q, a) = \{(q', c) \in Q \times \Gamma \mid \text{ there is } q \xrightarrow{a:c} q' \in \Delta\},$$

and we will sometimes replace Δ by δ in the representation of automata. Also, for a transition $e = (q, a, c, q')$, we let $\mathsf{let}(e) = a$ and $\mathsf{col}(e) = c$ be the projection into the input letter and output colour, respectively.

✦ Remark I.3. *We remark that, if necessary (and when complexity aspects are not under consideration), we can suppose that Γ is the set of transitions Δ and col is the identity function. Indeed, an equivalent acceptance condition can always be defined by using the acceptance set $W' = \{w \in E^\omega \mid \mathsf{col}(w) \in W\} \subseteq E^\omega$.*

The *size* of an automaton is its number of states, and we note it $|\mathcal{A}|$

State-based automata. We emphasise that in our definition of automata, the acceptance condition is put over the *transitions*. This will be a crucial element in all the chapters of this thesis (see also Chapter VI). We introduce automata with the acceptance condition over the states for comparison purposes.

A *state-based automaton* is a tuple $\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta_{\mathsf{st}}, \mathsf{col}_{\mathsf{st}}, W)$, where all the elements are as in the paragraph above except for the set of transitions, which is a subset $\Delta_{\mathsf{st}} \subseteq Q \times \Sigma \times Q$ and the colouring function $\mathsf{col}_{\mathsf{st}} \colon Q \to \Gamma$.

Determinism and completeness. We say that an automaton $\mathcal{A}$ is *deterministic* if I is a singleton and for every $q \in Q$ and $a \in \Sigma$, $|\delta(q, a)| \leq 1$. We say that $\mathcal{A}$ is *complete* if for every $q \in Q$ and $a \in \Sigma$, $|\delta(q, a)| \geq 1$. We remark that we can suppose that automata are complete without loss of generality by adding an extra state.

For deterministic automata, we will sometimes write $\mathsf{col}(q, a)$ to denote the output colour of the unique transition leaving from the state q reading letter a.

Runs and recognition of languages. Given an automaton $\mathcal{A}$ and a word $w \in \Sigma^\infty$, a *run over w* in $\mathcal{A}$ is a path

$$\rho = (q_0, w_0, c_0, q_1)(q_1, w_1, c_1, q_2)(q_2, w_2, c_2, q_3) \dots$$

such that $q_0 \in I$. The *output of* such a run is $\mathsf{col}(\rho) = c_0 c_1 c_2 \cdots \in \Gamma^\infty$, and we say that the run is *accepting* if $\mathsf{col}(\rho) \in W$, and *rejecting* otherwise.[5] A word $w \in \Sigma^\omega$ is *accepted* by $\mathcal{A}$ if it exists an accepting run over w. We remark that if $\mathcal{A}$ is deterministic (resp. complete), there is at most one (resp. at least one) run over w for each $w \in \Sigma^\omega$.

The *language accepted* (or *recognised*) by an automaton $\mathcal{A}$ is the set

$$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^\omega \mid w \text{ is accepted by } \mathcal{A}\}.$$

Two automata recognising the same language are said to be *equivalent*.

[5] Although we define runs over finite and infinite words, we will be interested in the acceptance of runs only over infinite words.

Subautomata. Given a subgraph $G' = (Q', \Delta')$ of the underlying graph of an automaton $\mathcal{A}$ and a subset of states $I' \subseteq Q'$, the *subautomaton induced by* G' with initial states I' is the automaton having Q' as set of states, Δ' as set of transitions, I' as set of initial states, and whose acceptance set is that of $\mathcal{A}$. To specify a subgraph, we will sometimes only give the subset of transitions Δ'.

Closed subautomata and X-Final SCC. Let $\mathcal{A}$ be an automaton, and let $X \subseteq \Sigma$ be a subset of the input alphabet. We say that a set of states $S \subset Q$ is *X-closed* if for every state q in S and every transition $q \xrightarrow{a:c} q'$, the state q' is in S. An *X-final strongly connected component* (X-*FSCC*) of $\mathcal{A}$ is an final SCC in the graph obtained by taking the restriction of the underlying graph of $\mathcal{A}$ to the edges labelled by letters in X. We remark that the states of a X-FSCC form an X-closed, and that a subset $S \subseteq Q$ is the set of states of an X-FSCC if and only if:

- for any two states $q, q' \in S$ there is a finite word $w \in X^*$ labelling a finite path from q to q', and
- if $q \in S$ and there is a finite path from q to q' labelled with a word $w \in X^*$, then $q' \in S$.

Lemma I.4 (Existence of X-FSCC)**.**

Let $\mathcal{A}$ be a complete automaton. For every subset $X \subseteq \Sigma$, $\mathcal{A}$ contains an accessible X-FSCC.

Proof. As any graph without sinks contains some final SCC, the accessible part of the restriction of $\mathcal{A}$ to edges labelled by letters in X contains a final SCC. By completeness of $\mathcal{A}$, one such final SCC has to be an X-closed subgraph, so it is an X-FSCC. ◀

Automaton structures and acceptance conditions on top of them. An *automaton structure* over Σ is a tuple $\mathcal{S} = (Q, \Sigma, I, \Delta)$, with $\Delta \subseteq Q \times \Sigma \times Q$. We apply the terms deterministic and complete to automaton structures in the natural way. An *acceptance condition on top of* $\mathcal{S}$ is a tuple $(\mathsf{col}, \Gamma, W)$, with $\mathsf{col}\colon \Delta \to \Gamma$ and $W \subseteq \Gamma^\omega$. This naturally defines an automaton $\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta', W)$, with $\Delta' \subseteq Q \times \Sigma \times \Gamma \times Q$ containing the tuples (q, a, c, q') such that $(q, a, q') \in \Delta$ is assigned colour c via col. We say that $\mathcal{A}$ is an *automaton on top of* $\mathcal{S}$.

✦ Remark I.5. *Given a automaton structure over Σ that is complete, and a language $L \subseteq \Sigma^\omega$, we can trivially define an acceptance condition on top of $\mathcal{S}$ obtaining an automaton recognising L. It suffices to take $\Gamma = \Sigma$, colour transitions with their input letters and use $W = L$ as acceptance set.*

2.2 History-deterministic automata

Sound resolvers. Let $\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta, W)$ be a (non-deterministic) automaton. A *resolver* for $\mathcal{A}$ is a pair (r_0, r), consisting of a choice of an initial state,[6] $r_0 \in I$, and a function $\mathsf{r}\colon \Delta^* \times \Sigma \to \Delta$ such that, for all words $w = w_0 w_1 \cdots \in \Sigma^\omega$, the sequence $e_0 e_1 \cdots \in \Delta^\omega$, called the *run induced by* r over w and defined by $e_i = \mathsf{r}(e_0 \dots e_{i-1}, w_i)$ is

actually a run over w in $\mathcal{A}$ starting from r_0. We write $r_0 \overset{w}{\rightsquigarrow}_{\mathsf{r}} q$ to denote that the run induced by r over w lands in q

We say that the resolver is *sound* if it satisfies that, for every $w \in \mathcal{L}(\mathcal{A})$, the run induced by r over w is an accepting run. In other words, r should be able to construct an accepting run in $\mathcal{A}$ letter-by-letter with only the knowledge of the word so far, for all words in $\mathcal{L}(\mathcal{A})$.

History-determinism. An automaton $\mathcal{A}$ is called *history-deterministic* (shortened HD) if there is a sound resolver for it. History-deterministic automata are sometimes called good-for-games in the literature; a definition of good-for-gameness and further explanations on this terminology will be given in Section II.1.3.

Determinisability by pruning. We say that we can *prune* $\mathcal{A}$ into $\mathcal{A}'$ if the latter automaton can be obtained by removing transitions from $\mathcal{A}$. Formally, there is a subset $\Delta' \subseteq \Delta$ and initial states $I' \subseteq I$ such that $\mathcal{A}'$ is the subautomaton induced by Δ' with initial states I'.

We say that an automaton $\mathcal{A}$ is *determinisable by pruning* if it can be pruned into an equivalent deterministic automaton.

✦ Remark I.6. *Deterministic automata are history-deterministic and they admit a unique resolver. Automata that are determinisable by pruning are also history-deterministic.*

Example I.7 (History-deterministic but not determinisable by pruning automaton)**.**

In Figure 4, we show an automaton $\mathcal{A}$ over $\Sigma = \{a, b, c\}$ that is not determinisable by pruning but is history-deterministic. Its set of output colours is $\Gamma = \{1, 2\}$ and its acceptance set is $W = \{u \in \{1, 2\}^\omega \mid u \text{ contains finitely many } 1s\}$ (this is a coBüchi condition, as introduced in Section I.3). It is easy to check that $\mathcal{A}$ recognises the language:

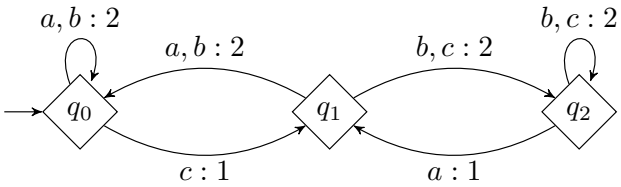
$$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^\omega \mid \mathsf{Inf}(w) \subseteq \{a, b\} \text{ or } \mathsf{Inf}(w) \subseteq \{b, c\}\}.$$

A resolver for $\mathcal{A}$ only has to take a decision when the automaton is in the state q_1 and letter b is provided. In this case, a sound resolver is obtained by using the following strategy: if the last letter seen was a, we take the transition leading to state q_0; if it was c, we take the transition leading to q_2. This strategy ensures that, if eventually only letters in $\{a, b\}$ (resp. $\{b, c\}$) are seen, the run will end up in state q_0 (resp. q_2) and remain there indefinitely, without producing any colour 1.

Reachability in HD automata. We say that an state q is *reachable using the resolver* (r_0, r) if for some word $w \in \Sigma^*$, $r_0 \overset{w}{\rightsquigarrow}_{\mathsf{r}} q$. That is, some run induced by r over some word $w \in \Sigma^*$ arrives to q.

Next remark indicates that we can suppose without loss of generality that all states in an HD automaton are reachable using some sound resolver.

[6] Sometimes in the literature [Bok+13, BL23b, HP06] the initial state r_0 is not required to be specified. This would permit to choose it after the first letter w_0 is given. We consider that a resolver constructing a run without guessing the future should pick the initial state before the first letter is revealed, hence the introduction of r_0 in the definition of a resolver. The suitability of this choice will be further supported by the generalisation of HD automata to HD mappings (Section II.2.3).



✦ **Figure 4.** An example of a history-deterministic automaton that is not determinisable by pruning. The acceptance set is $W = \{u \in \{1,2\}^\omega \mid u \text{ contains finitely many } 1s\}$. An arrow of the form $q \xrightarrow{a,b:2} q'$ represents two different transitions with input letters a and b, respectively. The initial state q_0 is marked with one incoming arrow.

✦ Remark I.8. *Let $\mathcal{A}$ be an HD automaton, let (r_0, r) be a sound resolver for it and let $\tilde{\mathcal{A}}$ be the subautomaton induced by the set of states reachable using (r_0, r), with initial state r_0. Then, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\tilde{\mathcal{A}})$, and $\tilde{\mathcal{A}}$ is HD.*

Simplification for prefix-independent languages. The following lemmas provide some simplifications for automata recognising prefix-independent languages. Together with Remark I.8, Lemma I.9 indicates that when dealing with HD automata for this kind of languages, we can suppose that any state of the automaton is the initial one. In particular there will be no need to specify the initial states of subautomata induced by subgraphs of HD automata recognising prefix-independent languages.

Lemma I.9 (Choice of an initial state)**.**

Let $\mathcal{A}$ be a history-deterministic automaton recognising a prefix-independent language and using as acceptance set a prefix-independent language. For any state q of $\mathcal{A}$ that is reachable using some sound resolver, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_q)$. Moreover, $\mathcal{A}_q$ is also history-deterministic. In particular, if $\mathcal{A}$ is deterministic, this is the case for any reachable state q.

Proof. Let (r_0, r) be a sound resolver for $\mathcal{A}$ such that q is reachable using (r_0, r), and fix a word $w_0 \in \Sigma^*$ such that r induces the run $\rho_0 = r_0 \overset{w_0}{\rightsquigarrow}_{\mathsf{r}} q$. We first show that $\mathcal{L}(\mathcal{A}_q) \subseteq \mathcal{L}(\mathcal{A})$. Let $w \in \Sigma^\omega$ be a word accepted from q. Then, $w_0 w$ admits an accepting run from the original initial state (by prefix-independence of the acceptance set), so $w_0 w \in \mathcal{L}(\mathcal{A})$, and by the prefix-independence of $\mathcal{L}(\mathcal{A})$, $w \in \mathcal{L}(\mathcal{A})$ too.

For the converse direction, we define a sound resolver (r'_0, r') for $\mathcal{A}_q$. We let $r'_0 = q$, and $\mathsf{r}'(\rho, a) = \mathsf{r}(\rho_0 \rho, a)$ be the strategy that acts as the resolver r assuming that ρ_0 has happened in the past. It is clear that for every word $w \in \Sigma^\omega$, the run induced by (r'_0, r') over w has a common suffix with the run induced by (r_0, r) over $w_0 w$. Therefore, by the prefix-independence assumptions:

$$w \in \mathcal{L}(\mathcal{A}) \iff w_0 w \text{ is accepted using } (r_0, \mathsf{r}) \iff w_0 w \text{ is accepted using } (r'_0, \mathsf{r}'). \quad \blacktriangleleft$$

Finite memory resolvers. Let $\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta, W)$ be a non-deterministic automaton. A *memory structure* for $\mathcal{A}$ is a memory skeleton $\mathcal{M}$ over Δ together with a function $\sigma \colon Q \times M \times \Sigma \to \Delta$, where M is the set of states of $\mathcal{M}$. We say that $(\mathcal{M}, \sigma)$ *implements* a resolver (q_0, r) if for all $a \in \Sigma$, $\mathsf{r}(\varepsilon, a) = \sigma(q_0, m_0, a)$ and for all $\rho \in \Delta^+$, $\mathsf{r}(\rho, a) = \sigma(\mathsf{target}(\rho), \mu(m_0, \rho), a)$, where m_0 is the initial state of $\mathcal{M}$ and $\mu \colon M \times \Delta^* \to M$

is its update function. For a state $q \in Q$ and $m \in M$, we say that (q, m) is *reachable using* r if there is some word $w \in \Sigma^*$ such that $\rho = q_0 \overset{w}{\rightsquigarrow}_{\mathsf{r}} q$ and $\mu(m_0, \rho) = m$.

Lemma I.10 ([Bok+13]).

Every history-deterministic parity automaton admits a sound resolver implemented by a finite memory structure.

3 Main classes of acceptance conditions and their representation

3.1 Main classes of acceptance conditions

We now define the main classes of languages used by ω-regular automata as acceptance sets. We let Γ stand for a finite set of colours.

Büchi. Given a subset $B \subseteq \Gamma$, we define the *Büchi language associated to* B as:

$$\mathsf{Buchi}_\Gamma(B) = \{w \in \Gamma^\omega \mid \mathsf{Inf}(w) \cap B \neq \emptyset\}.$$

We say that a language $L \subseteq \Gamma^\omega$ is a *Büchi language* if there is a set $B \subseteq \Gamma$ such that $L = \mathsf{Buchi}_\Gamma(B)$.

coBüchi. Given a subset $B \subseteq \Gamma$, we define the *coBüchi language associated to* B as:

$$\mathsf{coBuchi}_\Gamma(B) = \{w \in \Gamma^\omega \mid \mathsf{Inf}(w) \cap B = \emptyset\}.$$

We say that a language $L \subseteq \Gamma^\omega$ is a *coBüchi language* if there is a set $B \subseteq \Gamma$ such that $L = \mathsf{coBuchi}_\Gamma(B)$.

Generalised Büchi. Given k non-empty subsets $B_1, \dots, B_k \subseteq \Gamma$, we define the *generalised Büchi language associated to* $B = \{B_1, \dots, B_k\}$ as

$$\mathsf{genBuchi}_\Gamma(B) = \{w \in \Gamma^\omega \mid \mathsf{Inf}(w) \cap B_i \neq \emptyset \text{ for all } i \in \{1, \dots, k\}\}.$$

We say that a language $L \subseteq \Gamma^\omega$ is a *generalised Büchi language* if there is a family of sets $B = \{B_1, \dots, B_k\}$ such that $L = \mathsf{genBuchi}_\Gamma(B)$.

Generalised coBüchi. Given k non-empty subsets $B_1, \dots, B_k \subseteq \Gamma$, we define the *generalised coBüchi language associated to* $B = \{B_1, \dots, B_k\}$ as

$$\mathsf{genCoBuchi}_\Gamma(B) = \{w \in \Gamma^\omega \mid \mathsf{Inf}(w) \cap B_i = \emptyset \text{ for some } i \in \{1, \dots, k\}\}.$$

We say that a language $L \subseteq \Gamma^\omega$ is a *generalised coBüchi language* if there is a family of sets $B = \{B_1, \dots, B_k\}$ such that $L = \mathsf{genCoBuchi}_\Gamma(B)$.

Rabin. A Rabin language is represented by a family $R = \{(\mathfrak{g}_1, \mathfrak{r}_1), \dots, (\mathfrak{g}_r, \mathfrak{r}_r)\}$ of *Rabin pairs*, where $\mathfrak{g}_j, \mathfrak{r}_j \subseteq \Gamma$. The *Rabin language associated to* R is defined as:

$$\mathsf{Rabin}_\Gamma(R) = \{w \in \Gamma^\omega \mid [\mathsf{Inf}(w) \cap \mathfrak{g}_j \neq \emptyset \text{ and } \mathsf{Inf}(w) \cap \mathfrak{r}_j = \emptyset] \text{ for some index } j\}.$$

If $[\mathsf{Inf}(w) \cap \mathfrak{g}_j \neq \emptyset$ and $\mathsf{Inf}(w) \cap \mathfrak{r}_j = \emptyset]$, we say that w is *accepted by the Rabin pair* $(\mathfrak{g}_j, \mathfrak{r}_j)$. For $c \in \Gamma$, we will say that a Rabin pair $(\mathfrak{g}_j, \mathfrak{r}_j))$ is *green in* c if $c \in \mathfrak{g}_j$, *red in* if $c \in \mathfrak{r}_j$, or that is *orange in* if none of the previous occur. We say that a language $L \subseteq \Gamma^\omega$ is a *Rabin language* if there is a family of Rabin pairs R such that $L = \mathsf{Rabin}_\Gamma(R)$.

Streett. The *Streett language associated to* a family $R = \{(\mathfrak{g}_1, \mathfrak{r}_1), \dots, (\mathfrak{g}_r, \mathfrak{r}_r)\}$ of Rabin pairs is defined as:

$$\mathsf{Streett}_\Gamma(R) = \{w \in \Gamma^\omega \mid [\mathsf{Inf}(w) \cap \mathfrak{g}_j \neq \emptyset \text{ implies } \mathsf{Inf}(w) \cap \mathfrak{r}_j \neq \emptyset] \text{ for all indices } j\}.$$

We say that a language $L \subseteq \Gamma^\omega$ is a *Streett language* if there is a family of Rabin pairs R such that $L = \mathsf{Streett}_\Gamma(R)$.

Parity. We define the *parity language* over the alphabet $[d_{\min}, d_{\max}] \subseteq \mathbb{N}$ as:

$$\mathsf{parity}_{[d_{\min}, d_{\max}]} = \{w \in [d_{\min}, d_{\max}]^\omega \mid \min \mathsf{Inf}(w) \text{ is even}\}.$$

We say that a language $L \subseteq \Gamma^\omega$ is a *$[d_{\min}, d_{\max}]$-parity language* if there is a mapping $\phi\colon \Gamma \to [d_{\min}, d_{\max}]$ such that for all $w \in \Gamma^\omega$, $w \in L$ if and only if $\phi(w) \in \mathsf{parity}_{[d_{\min}, d_{\max}]}$. We say that L is a *parity language* if there are $d_{\min}, d_{\max} \in \mathbb{N}$ such that L is a $[d_{\min}, d_{\max}]$-parity language. When using exclusively parity languages, we will refer to colours as *priorities*.

Muller. We define the *Muller language associated to* a family $\mathcal{F} \subseteq 2^\Gamma_+$ of non-empty subsets of Γ as:

$$\mathsf{Muller}_\Gamma(\mathcal{F}) = \{w \in \Gamma^\omega \mid \mathsf{Inf}(w) \in \mathcal{F}\}.$$

We say that a language $L \subseteq \Gamma^\omega$ is a *Muller language* if there is a family $\mathcal{F} \subseteq 2^\Gamma_+$ such that $L = \mathsf{Muller}_\Gamma(\mathcal{F})$. We will often refer to sets in $\mathcal{F}$ as *accepting sets* and sets not in $\mathcal{F}$ as *rejecting sets*.

We drop the subscript Γ (resp. $[d_{\min}, d_{\max}]$) whenever the set of colours is clear from the context. We remark that all languages of the classes above are prefix-independent (for all $w \in \Gamma^\omega$ and $u \in \Gamma^*$, $uw \in L$ if and only if $w \in L$).

We say that an automaton is an *X automaton*, for X one of the classes of languages above, if its acceptance set is an X language. (We use analogous terminology for games or acceptance conditions.) In the case of parity automata, we will always suppose that the set of colours is a subset of $\mathbb{N}$ and ϕ is the identity function. We let *DPA* stand for deterministic parity automaton and *DMA* for deterministic Muller automaton.

We refer to the survey [Bok18] for a more detailed account on different types of acceptance conditions.

✦ Remark I.11. *A language $L \subseteq \Gamma^\omega$ is a Muller language if and only if it satisfies:*

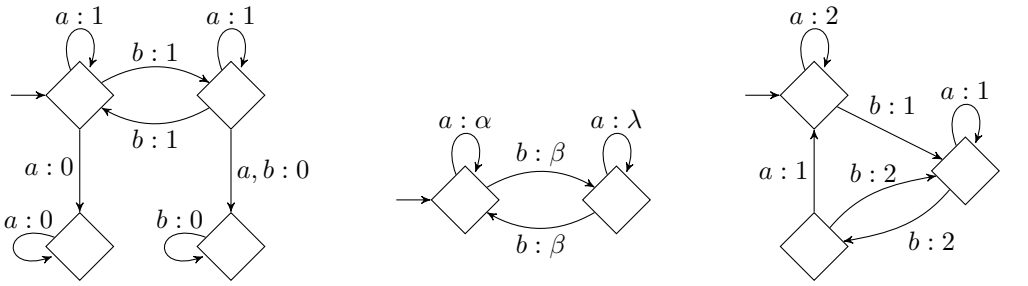
$$\textit{For all } w, w' \in \Gamma^\omega, \textit{ if } \mathsf{Inf}(w) = \mathsf{Inf}(w'), \textit{ then } w \in L \iff w' \in L.$$

Also, L is a Muller language if and only if it can be recognised by a deterministic Muller automaton with one state.

Example I.12.

In Figure 5 we show three different types of automata over the alphabet $\Sigma = \{a, b\}$ recognising the language of words that either eventually only contain letter b, or, after an even number of occurrences of letter b, only contain letter a. Formally:

$$L = \{w \in \Sigma^\omega \mid w = ub^\omega \text{ or } (w = ua^\omega \text{ and } u \text{ has an even number of 'b's })\}.$$



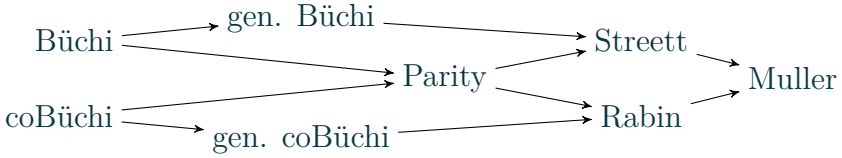
ND Büchi automaton | Det. Muller automaton $\mathcal{F} = \{\{\alpha\}, \{\beta\}\}$ | Det. coBüchi automaton

✦ **Figure 5.** Different types of automata recognising the language $L = \{w \in \Sigma^\omega \mid w = ub^\omega$ or $(w = ua^\omega$ and u has an even number of 'b's $)\}$.

Inclusions between classes

We observe that there are many inclusions between the classes of languages that we have introduced. For example, parity languages are Rabin languages, and generalised Büchi languages are Streett languages. In particular, all classes above are special cases of Muller languages. The relations between these classes of languages are outlined in Figure 6.

Moreover, we note that Büchi languages are exactly $[0, 1]$-parity languages and coBüchi languages are exactly $[1, 2]$-parity languages.



✦ **Figure 6.** Relations between subclasses of Muller languages. An arrow from a class X towards a class Y means that if a language $L \subseteq \Gamma^\omega$ is an X language, then it is also a Y language. Arrows obtained by transitivity have been omitted. Inclusions are strict: if an arrow from X to Y cannot be obtained by transitivity, then there are X languages that are not Y languages [Zie98].

✦ Remark I.13. *We remark that Streett languages are dual to Rabin ones: for a family of Rabin pairs R over a set of colours Γ, the following holds*

$$\mathsf{Rabin}_\Gamma(R) = \Gamma^\omega \setminus \mathsf{Streett}_\Gamma(R).$$

Similarly, (generalised) coBüchi languages are dual to (generalised) Büchi ones.

3.2 Representation of automata

In some parts of this thesis – mainly, in Chapter II – we will focus in the expressive power of acceptance conditions, and the results we present will not depend on how they are represented. On the other hand, in other parts (Chapter III, Sections II.5, etc...), we will consider decision problems for which it will be important to specify how we represent automata and what is the size of the input of the problem. Muller conditions are those

acceptance conditions for which the question of the representation is more delicate, as they admit multiple representations having very different properties. In this thesis, we will examine only a small number of decision problems involving Muller automata, so it will not be necessary for us to provide a comprehensive description of the different representation of Muller conditions.

✦ Remark I.14. *We can suppose without loss of generality that automata under consideration do not contain multiple transitions between the same two states labelled with the same input letter, that is, two transitions of the form* $q \xrightarrow{a:\alpha} q'$, $q \xrightarrow{a:\beta} q'$ *(see Appendix B.5).*

Representation of parity, Rabin and generalised Büchi automata

A parity automaton $\mathcal{A} = (Q, \Sigma, I, [d_{\min}, d_{\max}], \Delta, \mathsf{parity})$ is represented by the sets Q, I and Δ. The *size of its representation* is then $\mathcal{O}(|Q| + |\Delta||d_{\max}|)$, where $|d_{\max}|$ is the size of the representation of the integer $d_{\max}$ (for our purposes, it will not make a difference whether it is represented in binary or unary). By Remark I.14, we can assume that $|\Delta| \leq |Q|^2|\Sigma|$, and we can moreover assume that $d_{\max} \leq |\Delta|$. All in all, the size of the representation of a parity automaton is polynomial in its number of states, which will be enough precision for our purposes.

Similarly, a generalised Büchi (or generalised coBüchi) automaton $\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta, \mathsf{genBuchi}_\Gamma(B))$ is represented by the sets Q, I, Δ and $B = \{B_1, \dots, B_k\}$. The *size of its representation* is then $\mathcal{O}(|Q| + |\Delta| + k|\Gamma|)$. As we can assume that for each i, $B_i \not\subseteq \cup_{j\neq i} B_j$, we can suppose that $k \leq |\Gamma|$ and therefore we can also assume that the size of the representation of a generalised Büchi automaton is polynomial in its number of states.

A Rabin (or Streett) automaton $\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta, \mathsf{Rabin}_\Gamma(R))$ is represented by the sets Q, I, Δ and $R = \{(\mathfrak{g}_1, \mathfrak{r}_2), \dots, (\mathfrak{g}_r, \mathfrak{r}_r)\}$. The *size of its representation* is then $\mathcal{O}(|Q| + |\Delta| + 2r|\Gamma|)$. By Remark I.14, we can assume that $|\Delta| \leq |Q|^2|\Sigma|$, and we can moreover assume that $|\Gamma| \leq |\Delta|$. All in all, the size of the representation of a Rabin automaton is polynomial in $|Q| + r$, where r is the number of Rabin pairs it uses.

Representation of Muller automata

In practice, there exists a variety of ways to *represent* Muller languages; we cite here some of them and refer to [Bok19, Hug23] for a more detailed account. See also Section II.5 for a comparison between the sizes of the different representations.

Emerson-Lei conditions. A family $\mathcal{F} \subseteq 2^\Gamma_+$ is described as a positive boolean formula over the primitives $\mathtt{Inf}(c)$ and $\mathtt{Fin}(c)$, for $c \in \Gamma$. Automata using Emerson-Lei conditions are commonly used in practice, as they offer a succinct representation of the acceptance condition [Bab+15].

Explicit Muller. A family $\mathcal{F} \subseteq 2^\Gamma_+$ can be described explicitly as a list of the subsets appearing in $\mathcal{F}$.

Zielonka tree. We will define the Zielonka tree of a family $\mathcal{F} \subseteq 2^\Gamma_+$ in Section II.3.1. The Zielonka tree is a representation of $\mathcal{F}$ that captures all the fundamental information about it.

Zielonka DAG. The Zielonka DAG is a structure obtained from the Zielonka tree, and that can be more succinct that the latter. We formally define it in Section II.3.1. It has been first studied by Hunter and Dawar [HD05] and recently, Hugenroth has put forward its good algorithmic properties [Hug23]. Our results will further support

the idea that the Zielonka DAG is a fairly succinct representation that nevertheless allows to solve many important decision problems about Muller automata in polynomial time (see e.g. Theorems II.9, II.10 and II.11).

Parity automata. A Muller language $\mathsf{Muller}_\Gamma(\mathcal{F})$ can be described by a DPA recognising it. As we will prove in Theorems II.2 and Proposition III.18, this representation is equivalent as the one using the Zielonka tree of $\mathcal{F}$.

The complexity and practicality of algorithms manipulating Muller automata and Muller games may greatly differ depending on which of these representations is used [Hor08, HD05] (see also Section II.5.1). However, as commented previously, most of the results we present will not depend on the representation of the Muller acceptance condition. The representation of Muller automata will be important specially in Section II.5 of Chapter II.

3.3 Classic results

We state some well-known results about ω-automata that will be used in several points of the manuscript.

Proposition I.15 (Deciding language containment of Rabin automata [CDK93])**.**

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two deterministic Rabin automata. We can decide in polynomial time on the representation of the automata whether $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$.

Corollary I.16 (Deciding equivalence of automata)**.**

We can decide in polynomial time the containment and equivalence of deterministic automata using acceptance conditions of some of the types in {parity, Rabin, Streett, generalised Büchi, generalised coBüchi}.

Lemma I.17 (Folklore)**.**

Let $L_1, L_2 \subseteq \Sigma^\omega$ be two ω-regular languages. If $L_1 \not\subseteq L_2$, there are finite words $w_1, w_2 \in \Sigma^*$ such that $w_1 w_2^\omega \in L_1$ and $w_1 w_2^\omega \notin L_1$.

4 ω-regular languages, cycles and the parity hierarchy

4.1 ω-regular languages

The class of ω-regular languages plays a central role in the theory of formal languages and verification. The significance of ω-regular languages is (partly) due to the robustness of its definition, as they admit multiple equivalent characterisations relating different areas of study.

Proposition I.18 ([Mos84, McN66])**.**

Let $L \subseteq \Sigma^\omega$ be a language of infinite words. The following properties are equivalent:

- L can be recognised by a non-deterministic Büchi automaton.
- L can be recognised by a deterministic parity automaton.
- L can be recognised by a non-deterministic Muller automaton.

A language satisfying the previous conditions is called *ω-regular*. Many other equivalent definitions exist. Notably, ω-regular languages are exactly the languages that can be defined using *monadic second-order logic* [Bü62], those that can be described by using *ω-regular expressions* [McN66], and those that can be recognised by an *ω-semigroup* [PP04, Chapter 2].

4.2 Cycles

Let $\mathcal{A}$ be an automaton with Q and Δ as set of states and transitions, respectively. A *cycle* of $\mathcal{A}$ is a subset $\ell \subseteq \Delta$ such that there is a finite path $q_0 \xrightarrow{e_0} q_1 \xrightarrow{e_1} q_2 \to \dots q_r \xrightarrow{e_r} q_0$ with $e_i \in \Delta$ and $\ell = \{e_0, e_1, \dots, e_r\}$. We remark that we do not require this path to be simple, that is, edges and vertices may appear multiple times. The set of *states of the cycle* ℓ is $\mathsf{States}(\ell) = \{q_0, q_1, \dots q_r\}$. We say that two cycles ℓ_1, ℓ_2 have some *state in common* if $\mathsf{States}(\ell_1) \cap \mathsf{States}(\ell_2) \neq \emptyset$. The set of cycles of an automaton $\mathcal{A}$ is written $\mathcal{Cycles}(\mathcal{A})$. We will consider the set of cycles ordered by inclusion. For a state $q \in Q$, we note $\mathcal{Cycles}_q(\mathcal{A})$ the subset of cycles of Q containing q. We remark that a state q is recurrent in the underlying graph of $\mathcal{A}$ if and only if $\mathcal{Cycles}_q(\mathcal{A}) \neq \emptyset$. We note that $\mathcal{Cycles}_q(\mathcal{A})$ is closed under union; moreover, the union of two cycles $\ell_1, \ell_2 \in \mathcal{Cycles}(\mathcal{A})$ is again a cycle if and only if they have some state in common.

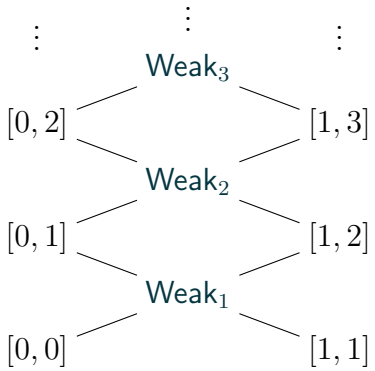
Let $\mathcal{A}$ be a Muller automaton with acceptance condition $(\mathsf{col}, \Gamma, \mathsf{Muller}_\Gamma(\mathcal{F}))$. Given a cycle $\ell \in \mathcal{Cycles}(\mathcal{A})$, we say that ℓ is *accepting* (resp. *rejecting*) if $\mathsf{col}(\ell) \in \mathcal{F}$ (resp. $\mathsf{col}(\ell) \notin \mathcal{F}$). We remark that the maximal cycles of an automaton are exactly the sets of edges of the strongly connected components of its underlying graph. In particular, we can apply the adjectives *accepting* and *rejecting* similarly to the SCCs of a Muller automaton.

We note that, by definition, the acceptance of a run in a Muller automaton only depends on the set of transitions taken infinitely often. For any infinite run ρ, the set of transitions taken infinitely often forms a cycle, $\mathsf{Inf}(\rho) = \ell_\rho \in \mathcal{Cycles}(\mathcal{A})$, and ρ is an accepting run if and only if ℓ_ρ is an accepting cycle.

4.3 The deterministic and history-deterministic parity hierarchy

As we have mentioned, every ω-regular language can be recognised by a deterministic parity automaton, but the number of colours required to do so might be arbitrarily large. We can assign to each ω-regular language the optimal number of colours needed to recognise it using a deterministic automaton. We obtain in this way the *deterministic parity hierarchy* (called *Mostowski hierarchy* in the context of tree-automata [CL08]), having its origins in the works of Wagner [Wag79], Kaminski [Kam85], and Mostowski [Mos84]. We represent this hierarchy in Figure 7. This hierarchy is strict, that is, for each level of the hierarchy there are languages that do not appear in lower levels [Wag79]. It is known that we can decide in polynomial time the parity index of an ω-regular language repre-

sented by a deterministic parity automaton [CM99], but this problem is NP-complete if the language is given by a deterministic Rabin or Streett automaton [KPB95].



✦ **Figure 7.** The (history-)deterministic parity hierarchy.

Definition I.19 (Parity index of a language)**.**

Let $L \subseteq \Sigma^\omega$ be an ω-regular language. We say that L has *parity index at least* $[0, d-1]$ (resp. $[1, d]$) if any DPA recognising L with a parity acceptance condition over the set of colours $[d_{\min}, d_{\max}]$ satisfies that $d_{\max} - d_{\min} \geq d - 1$, and in case of equality $d_{\min}$ is even (resp. odd). We say that the *parity index* of L is $[0, d-1]$ (resp. $[1, d]$) if, moreover, there is a DPA recognising L with a parity acceptance condition over the set of colours $[0, d-1]$ (resp. $[1, d]$).

We say that L has *parity index at least* Weak_d if any DPA recognising L with a parity acceptance condition over the set of colours $[d_{\min}, d_{\max}]$ satisfies that $d_{\max} - d_{\min} \geq d$. We say that the parity index of L is Weak_d if, moreover, there are DPAs $\mathcal{A}_1$ and $\mathcal{A}_2$ recognising L with parity acceptance conditions over the sets of colours $[0, d]$ and $[1, d+1]$, respectively.

Throughout the thesis, whenever we refer to the parity index of a language L, it will implicitly imply that L is ω-regular.

If follows from the definition that for each ω-regular language L, there is a unique d such that either L has parity index $[0, d-1]$, $[1, d]$ or Weak_d, and these options are mutually exclusive. See also Definition II.1 for more details about languages of parity index Weak_d.

As we will show in Section II.6.2, the parity index also applies to Muller automata: any deterministic or HD Muller automaton recognising an ω-regular language of parity index $[0, d-1]$ uses at least d different colours (Proposition II.117).

The following proposition states that the notion of parity index of a language does not change by using HD automata instead of deterministic ones in the definition. However, for non-deterministic automata, the hierarchy collapses at level $[0, 1]$ (Büchi automata) [McN66].

Proposition I.20 (Parity index for HD automata [BL19, Theorem 19])**.**

Let $\mathcal{A}$ be an HD parity automaton recognising a language L, and suppose that the parity index of L is $[0, d-1]$ (resp. $[1, d]$). Then, the acceptance condition of $\mathcal{A}$ uses at least d output colours, and if it uses exactly d colours, the least of them is even (resp. odd). If the parity index of L is Weak_d, then $\mathcal{A}$ uses at least $d+1$ output colours.

We show next that the parity index of an ω-regular language can be read directly from a deterministic Muller automaton.

Let $\mathcal{A}$ be an automaton using the Muller acceptance condition $(\mathsf{col}, \Gamma, \mathsf{Muller}_\Gamma(\mathcal{F}))$. A *$d$-flower* over a state q of $\mathcal{A}$ is a set of d cycles $\ell_1, \ell_2, \dots, \ell_d \in \mathcal{Cycles}_q(\mathcal{A})$ such that $\ell_i \supsetneq \ell_{i+1}$ and $\mathsf{col}(\ell_i) \in \mathcal{F} \iff \mathsf{col}(\ell_{i+1}) \notin \mathcal{F}$. We say that it is a *positive flower* if $\mathsf{col}(\ell_1) \in \mathcal{F}$ and that it is *negative* otherwise.

Lemma I.21 (Flower Lemma [NW98, Wag79])**.**

Let $\mathcal{A}$ be a DMA. If $\mathcal{A}$ admits an accessible positive (resp. negative) d-flower, then $\mathcal{L}(\mathcal{A})$ has parity index at least $[0, d-1]$ (resp. $[1, d]$). If $\mathcal{A}$ admits both accessible positive and negative d-flowers, then $\mathcal{L}(\mathcal{A})$ has parity index at least Weak_d.

Conversely, if an ω-regular language L has parity index at least $[0, d-1]$ (resp. $[1, d]$), then any DMA recognising L admits a positive (resp. negative) d-flower.

✦ Remark I.22. *Deterministic generalised Büchi (resp. generalised coBüchi) automata have the same expressive power than deterministic Büchi (resp. coBüchi) automata: they recognise languages of parity index at most $[0, 1]$ (resp. $[1, 2]$). This well-known property follows from the results in Chapter II, Section II.6.*

✦ Remark I.23 (Wagner hierarchy). *The parity hierarchy was refined in the work of Wagner [Wag79], where he introduces a hierarchy that takes into account the structure of Muller automata with greater precision. To avoid overcomplicating the presentation of this thesis, we will not consider Wagner's hierarchy in the subsequent sections.*

Optimal transformations of automata and games using Muller conditions

II

Outline

Hay objetos en el mundo que hacen varias funciones y todas las hacen bien.[...] Si tú eres un objeto que tiene varias funciones y todas las haces bien, entonces no tengo nada que reprocharte.

TER

Introduction

In this chapter, we focus on the problem of transforming Muller automata to simplify their acceptance condition: given an automaton using a Muller acceptance condition, our goal is to construct an equivalent one using a parity acceptance condition. We introduce the alternating cycle decomposition (ACD), a data structure that dissects the structure of Muller automata. We use the ACD to define optimal transformations of Muller automata into parity and Rabin ones, as well as to derive various theoretical results about these automata.

Applications of automata transformations. As remarked in the general introduction, the bottleneck in modern implementations of LTL synthesis algorithms is the transformation of the input logic formula into a deterministic parity automaton. Most solutions to this problem (including the top-ranked tools in the SYNTCOMP competitions, Strix [LMS20, MS21b] and `ltlsynt` [MC18]) first construct a Muller automaton, and then transform it into an equivalent parity automaton. The use of an intermediate Muller automaton is also present (although sometimes implicitly) in the most recent improvements in the determinisation of Büchi automata towards deterministic parity automata [Pit06, Sch09, LP19]. For these reasons, understanding transformations of Muller automata and finding efficient procedures for them holds significant importance.

Existing methods. There are various existing techniques to transform Muller automata or games into parity ones. The majority of these methods involve composing the input automaton $\mathcal{A}$ with a deterministic parity automaton recognising the acceptance condition used by $\mathcal{A}$. The first such parity automaton was introduced by Gurevich and Harrington in the 1980s [GH82] and is known as the Latest Appearance Record (LAR).

Löding proved that the LAR is optimal in the worst case [Löd99]: *there exists* a family of Muller languages L_i for which the LAR is minimal amongst deterministic parity automata recognising L_i. However, the LAR is far from being minimal in every case, as it only uses the information about the size of the alphabet. Since its introduction, many refinements of the LAR have been proposed for subclasses of Muller languages [Löd99, Kře+17]. The approach using composition of automata has one significant drawback: it disregards the structure of the original automaton, and only its acceptance condition is taken into account. Some works have explored heuristics to improve this aspect [RDLP20, Kře+21, MS21a]. These refined transformations do still have the following property: each original state q is turned into multiple states of the form (q, x) – although this is done in a non-uniform way, with each state possibly being copied a different number of times. In this work, we introduce morphisms of transition systems to formalise the idea of transformations of automata and games; if a parity automaton $\mathcal{B}$ has been obtained as a transformation of a Muller automaton $\mathcal{A}$, there will be a morphism $\varphi\colon \mathcal{B} \to \mathcal{A}$ that sends states of the form (q, x) to q. A theory of morphisms of transition systems is developed in Section II.2.

The Zielonka tree and the alternating cycle decomposition. The starting point of our work is the notion of Zielonka tree, introduced by Zielonka [Zie98] as an informative representation of Muller languages – languages that can be described by a boolean combination of atomic propositions of the form "the letter 'a' appears infinitely often". The Zielonka tree captures many important properties of Muller languages, such as being Rabin or parity [Zie98], and, most importantly, it characterises their exact memory requirements, both in two-player games [DJW97] and stochastic games [Hor09].

The contribution at the core of this work is a generalisation of Zielonka trees to general Muller automata recognising any ω-regular language, which we call the alternating cycle decomposition (ACD). The ACD, greatly inspired from Wagner's work on ω-automata [Wag79], is a data structure that provides an abridged representation of the accepting and rejecting cycles of the automaton, encapsulating the interplay between the structure of the underlying graph and the acceptance condition of a Muller automaton.

Contributions

In this chapter, we perform an extensive study of transformations of Muller automata and games. We outline next our main contributions.

1. **Minimal automata for Muller languages.** The basis on which we build up our work is a study of minimal automata recognising Muller languages. Using the Zielonka tree, we propose a construction of a deterministic parity automaton recognising a Muller language (Section II.3.2). This construction implicitly appears in the long version of [DJW97]. We show a strong optimality result: *for every* Muller language L, the parity automaton obtained from the Zielonka tree is minimal both amongst deterministic and history-deterministic parity automata recognising L (Theorem II.2).[1] Moreover, it uses the optimal number of output colours to recognise L (Theorem II.1). The optimality result we obtain is much stronger than the worst case optimality result of the LAR transformation [Löd99], since it applies to every Muller language. In particular, our characterisation yields an algorithm to

[1] The optimality of the Zielonka-tree-parity-automaton amongst deterministic automata has also been obtained independently by Meyer and Sickert in the unpublished work [MS21a].

minimise deterministic parity automata recognising Muller languages in polynomial time (Theorem III.4). In light of our result, we conclude that the use of history-determinism does not yield any gain in the state complexity of parity automata recognising Muller languages.

We further propose a construction of a history-deterministic Rabin automaton recognising a given Muller language (Section II.3.3), and prove that this automaton is minimal amongst history-deterministic Rabin automata (Theorem II.4). This construction is also based on the Zielonka tree.

In essence, our results reinforce the idea that the Zielonka tree precisely captures the fundamental properties of Muller languages.

2. **Introducing morphisms as witnesses of transformations.** In order to formalise transformations of games and automata, we develop a theory of morphisms of transition systems (Section II.2).[2] Intuitively, a morphism $\varphi\colon \mathcal{B} \to \mathcal{A}$ witnesses the fact that $\mathcal{B}$ has been obtained from $\mathcal{A}$ by blowing up each state $q \in \mathcal{A}$ to the states in $\varphi^{-1}(q)$. However, this property on its own does not suffice to guarantee the semantic equivalence of $\mathcal{A}$ and $\mathcal{B}$. It is for this reason that we introduce different variants of morphisms, offering a range of definitions with varying degrees of restrictiveness. Two kind of morphisms will be of central importance: (1) locally bijective morphisms, which generalise composition by deterministic automata and preserve determinism, and (2) history-deterministic mappings (HD mappings), which generalise composition by history-deterministic automata and are defined using a minimal set of hypothesis guaranteeing the semantic equivalence of $\mathcal{A}$ and $\mathcal{B}$.

3. **The alternating cycle decomposition and optimal transformations of Muller automata.** In order to generalise the fruitful applications of the Zielonka tree to Muller automata and games, we introduce the alternating cycle decomposition (ACD), a data structure that captures the interplay of the underlying graph of an automaton and its acceptance condition (Section II.4). Using the ACD, we describe a construction that transforms a Muller automaton $\mathcal{A}$ into an equivalent parity automaton $\mathcal{B}$ while preserving the determinism of $\mathcal{A}$ (formally, there is a locally bijective morphism $\varphi\colon \mathcal{B} \to \mathcal{A}$). This transformation comes with a strong optimality guarantee: for any other parity automaton $\mathcal{B}'$ admitting a locally bijective morphism (or even HD mapping) $\varphi'\colon \mathcal{B}' \to \mathcal{A}$, the automaton $\mathcal{B}$ is smaller than $\mathcal{B}'$ and it uses less output colours (Theorems II.5 and II.6). An interesting corollary of our result is the following: if $\mathcal{B}$ is an HD parity automaton that is strictly smaller than any deterministic parity automaton recognising $\mathcal{L}(\mathcal{B})$, then $\mathcal{B}$ cannot be derived from a deterministic Muller automaton (Corollary II.80). This result sheds light on the difficulty to obtain succinct HD automata and their potential applicability.

   We also provide a transformation that translates a Muller automaton $\mathcal{A}$ into a history-deterministic Rabin automaton $\mathcal{B}$ in an optimal way: for any other Rabin automaton $\mathcal{B}'$ admitting an HD mapping $\varphi'\colon \mathcal{B}' \to \mathcal{A}$, the automaton $\mathcal{B}$ is smaller than $\mathcal{B}'$.

   We show that the ACD can be computed in polynomial time if the acceptance condition is represented as a Zielonka tree (Theorem II.8).

4. **Structural results for Muller automata.** The ACD does not only provide optimal

[2] Very similar notions of morphisms have been studied by Sakarovitch and De Souza in the context of transducers over finite words [Sak98, SS10].

transformations of games and automata, it also exhibits some of their fundamental structural properties. As an application, we give a set of crisp characterisations for relabelling automata with different classes of acceptance conditions (Section II.6). For instance, we show that given a Muller automaton $\mathcal{A}$, we can define a Rabin condition over the underlying graph of $\mathcal{A}$ obtaining an equivalent automaton if and only if the union of rejecting cycles of $\mathcal{A}$ is again a rejecting cycle. Our results unify and extend those from [BJW01, BKS10, KPB94, Zie98]. These results will find application in Chapters III and IV, for the study of the minimisation of automata and memory for games, respectively. We also show that the minimisation of the number of colours used by the acceptance conditions of Muller automata is NP-hard (Theorem II.13).

In Section II.7, we conduct a comprehensive examination of a normal form for parity automata. This normal form implicitly appears in [CM99], and has since proven instrumental in proofs about history-deterministic automata [KS15, AK22, ES22], positionality of ω-regular languages [Bou+22] and learning of ω-automata [BL23a]. Similar normalisation procedures are commonly applied to parity games to speed up algorithms solving them [FL09]. We use the ACD to provide straightforward proofs of the fundamental properties which make automata in normal form practical in both theoretical proofs and applications. This normal form and its properties will be an essential tool in our characterisation of half-positionality in Chapter V.

Our model: transition systems. We want to point out a technical detail about the model used in this chapter. We work with general transition systems, that is, graphs with an acceptance condition on top of them. This choice has been made for two reasons: (1) to seamlessly encompass both automata and games models, and (2) to emphasise that the ACD and the transformations we propose do only depend on the underlying graph and the acceptance condition; we can view the input letters of an automaton or the partition of the vertices in a game as add-ons that do not affect the core of our approach.

Also, as discussed in the general introduction, we define acceptance conditions over the edges of transitions systems and we are concerned with state complexity. The representation of acceptance conditions is of secondary importance in this chapter.

Collaborators and related publications

The contributions of this chapter are the outcome of a series of collaborations, as we discuss now. This line of research initiated during my master's thesis, supervised by Thomas Colcombet and Nathanaël Fijalkow. Our first results – mainly the material concerning deterministic automata and locally bijective morphisms – was published in the conference paper [CCF21]. Together with Thomas Colcombet and Karoliina Lehtinen, we generalised our results to obtain minimal history-deterministic Rabin automata (Theorem II.4), result appearing (amongst others) in [CCL22]. These contributions, together with some additional content (Theorem II.2, Sections II.4.3 and II.7), constitute the material of a journal submission under review [Cas+23].

Some of the contents of the chapter are still unpublished, mainly results concerning computational aspects of the alternating cycle decomposition (Sections II.5 and II.6.3). These have been obtained jointly with Corto Mascle.

Transformations based on the alternating cycle decomposition brought the attention of the developers of two open-source tools for ω-automata and LTL synthesis: Spot [DL+22] and Owl [KMS18]. The ACD transformations were implemented in Spot 2.10 by Alexan-

dre Duret-Lutz and Florian Renkin, and in Owl 21.0 by Klara J. Meyer and Salomon Sickert. Together, we prepared the tool paper [Cas+22], where transformations based on the ACD are compared to the state-of-the-art existing paritizing methods. That paper also contains material on transformations for state-based automata (Section II.8).

1 Preliminaries

1.1 Transition systems

We introduce transition systems to encompass both automata and games with a definition oriented towards the study of the acceptance condition and its interaction with the underlying graph, giving secondary importance to semantic aspects such as languages recognised by automata or winners of games.

Acceptance conditions. An *acceptance condition* over a graph G with edges E is a tuple $\mathrm{Acc} = (\mathsf{col}, \Gamma, W)$ where Γ is a finite set of *colours*, $\mathsf{col} : E \to \Gamma \cup \{\varepsilon\}$ is an *edge-colouring* of G and $W \subseteq \Gamma^\omega$ is a language of infinite words called the *acceptance set*. We allow *uncoloured edges* (ε-edges), but we impose that no infinite path of G is eventually composed exclusively of ε-edges.

As mentioned in Remark I.3, we can assume whenever necessary that Γ equals E and col is the identity function.

We focus on acceptance sets of some of the classes defined in Section I.3. Of particular importance for this chapter will be parity, Rabin and Muller languages.

Transition systems. A *transition system* (abbreviated *TS*) is a tuple of the form $\mathcal{TS} = (G_{\mathcal{TS}}, \mathrm{Acc}_{\mathcal{TS}})$, where $G_{\mathcal{TS}} = (V, E, \mathsf{source}, \mathsf{target}, I)$ is a pointed graph, called the *underlying graph* of $\mathcal{TS}$, and $\mathrm{Acc}_{\mathcal{TS}} = (\mathsf{col}, \Gamma, W)$ is an acceptance condition over $G_{\mathcal{TS}}$. We will also refer to vertices and edges as states and transitions, respectively. We write $v \xrightarrow{c} v'$ if there is $e \in E$ such that $\mathsf{source}(e) = v$, $\mathsf{target}(e) = v'$ and $\mathsf{col}(e) = c$. We assume for technical convenience that transition systems contain no sink, that is, every vertex has at least one outgoing edge. For any non-empty subset of vertices $\tilde{I} \subseteq V$, we let $\mathcal{TS}_{\tilde{I}}$ be the transition system obtained from $\mathcal{TS}$ by setting $\tilde{I}$ to be its set of initial vertices. The *size* of a transition system $\mathcal{TS}$ is the cardinality of its set of vertices, written $|\mathcal{TS}|$.

We define *cycles* of transition systems in the exact same way as cycles for automata (that is, as a subset of edges that can be visited by a closed path) and use similar notations.

The comments we made about the size of the representation of automata apply similarly to transition systems. In this chapter, transitions systems are assumed to be finite.

As in the case of automata, we can consider *state-based transition systems*, for which the colouring of the acceptance condition is given by $\mathsf{col}_{\mathsf{st}} : V \to \Gamma \cup \{\varepsilon\}$. We will only use state-based TS for the sake of comparison in Section II.8. Transition systems will be transition-based by default.

Runs. A *run* on a transition system $\mathcal{TS}$ (or on a pointed graph) is a (finite or infinite) path $\rho = e_0 e_1 \cdots \in E^\infty$ starting from an initial vertex, that is, $\mathsf{source}(e_0) \in I$. We let $\mathcal{R}un^{\mathsf{fin}}(\mathcal{TS})$ and $\mathcal{R}un(\mathcal{TS})$ be the set of finite and infinite runs on $\mathcal{TS}$, respectively, and we let $\mathcal{R}un^{\infty}(\mathcal{TS}) = \mathcal{R}un^{\mathsf{fin}}(\mathcal{TS}) \cup \mathcal{R}un(\mathcal{TS})$. (We note that $\mathcal{R}un(\mathcal{TS}) = \mathcal{P}ath_I(G_{\mathcal{TS}})$.)

The *output* of a run $\rho \in \mathcal{Run}^\infty(\mathcal{TS})$ is the sequence of colours in Γ^∞ obtained by removing the occurrences of ε from $\mathsf{col}(\rho)$; which we will also denote $\mathsf{col}(\rho)$ by a small abuse of notation. A run ρ is *accepting* if $\mathsf{col}(\rho) \in W$, and *rejecting* otherwise (in particular, finite runs will be rejecting). We write $\rho = v \overset{w}{\rightsquigarrow} v'$ to denote a run with $\mathsf{source}(\rho) = v$, $\mathsf{target}(\rho) = v'$ and $\mathsf{col}(\rho) = w$.

Labelled transition systems. A *labelled graph* $(G, (l_V, L_V), (l_E, L_E))$ is a graph together with labelling functions $l_V : V \to L_V$, $l_E : E \to L_E$, where L_V and L_E are sets of labels for vertices and edges, respectively. If only the first (resp. the second) of these labelling functions appears, we will use the terms *vertex-labelled* (resp. *edge-labelled*) graphs. A *labelled transition system* is a transition system with labelled underlying graph.

Automata and games as transition systems. We can represent automata and games, as introduced in the general preliminaries, as labelled transition systems.

An automaton over Σ admits a representation as an edge-labelled transition system

$$\mathcal{A} = (G_\mathcal{A}, \mathrm{Acc}_\mathcal{A}, (\mathsf{let}, \Sigma)),$$

where $\mathsf{let}\colon E \to \Sigma$ is a labelling with input letters.

A game admits a representation as a vertex-labelled transition system

$$\mathcal{G} = (G_\mathcal{G}, \mathrm{Acc}_\mathcal{G}, (l_{\mathsf{Players}}, \{\mathrm{Eve}, \mathrm{Adam}\}))\,,$$

with $l_{\mathrm{Players}}\colon V \to \{\mathrm{Eve}, \mathrm{Adam}\}$ a vertex-labelling function inducing a partition of V into vertices controlled by two Eve and Adam: $V_{\mathrm{Eve}} = l^{-1}_{\mathsf{Players}}(\mathrm{Eve})$ and $V_{\mathrm{Adam}} = l^{-1}_{\mathsf{Players}}(\mathrm{Adam})$.

Generalised weak transition systems. In this chapter, we will also consider conditions that depend on the structure of a given transition system and not only on the set of colours, that we call *generalised weak conditions*.

Definition II.1 (Weak_d transition systems)**.**

Let $\mathcal{TS} = (G_\mathcal{TS}, \mathrm{Acc}_\mathcal{TS})$ be a transition system using a parity condition $\mathrm{Acc}_\mathcal{TS} = (\mathsf{col}, [d_{\min}, d_{\max}], \mathsf{parity})$. We say that $\mathcal{TS}$ (or $\mathrm{Acc}_\mathcal{TS}$) is Weak_d if in each strongly connected component $\mathcal{S} \subseteq G_\mathcal{TS}$ there are at most d different colours that appear, that is, $|\mathsf{col}(E_\mathcal{S})| \leq d$, where $E_\mathcal{S}$ is the set of edges of $\mathcal{S}$.

The adjective *Weak* has typically been used to refer to the condition corresponding to a partition of $\mathcal{TS}$ into accepting and rejecting SCC [MSS86]. A run will be accepting if the component it finally stays in is accepting. It corresponds to the Weak_1-condition with our notation.

As we will show (Corollary II.116), the notation is justified by the fact that an ω-regular language of parity index Weak_d can be recognised by a deterministic Weak_d automaton.

1.2 Equivalence of transition systems and typeness

Let $\mathcal{TS}_1 = (G, \mathrm{Acc}_1)$ and $\mathcal{TS}_2 = (G, \mathrm{Acc}_2)$ be two (labelled) transitions systems over the same underlying graph G (and using the same labels) with acceptance conditions $\mathrm{Acc}_i = (\mathsf{col}_i, \Gamma_i, W_i)$, for $i \in \{1, 2\}$. We say that Acc_1 and Acc_2 are *equivalent over* G,

written $\mathrm{Acc}_1 \simeq_G \mathrm{Acc}_2$, if for all runs $\rho \in \mathcal{R}un(G)$, ρ is accepting for $\mathcal{TS}_1$ if and only if it is accepting for $\mathcal{TS}_2$; that is, $\mathsf{col}_1(\rho) \in W_1 \iff \mathsf{col}_2(\rho) \in W_2$. In this case, we write $\mathcal{TS}_1 \simeq \mathcal{TS}_2$ and say that $\mathcal{TS}_1$ and $\mathcal{TS}_2$ are *equivalent*. This is equivalent to the fact that there is an isomorphism of transition systems between $\mathcal{TS}_1$ and $\mathcal{TS}_2$ (see Section II.2.1 for the definition of isomorphism).

For X one of types of languages defined in Section I.3 (Büchi, parity, Muller, etc...), we say that a transition system $\mathcal{TS}$ is *X type* if there exists an acceptance condition Acc_X over the underlying graph of $\mathcal{TS}$ of type X and such that it is equivalent to that of $\mathcal{TS}$. We say that this new acceptance condition has been defined *on top of* $\mathcal{TS}$. Similarly, we say that $\mathcal{TS}$ is *Weak_d-type* if there is an equivalent parity condition over $\mathcal{TS}$ making it Weak_d.

✦ Remark II.2. *Given a pointed graph G (whose states are accessible), the equivalence classes of Muller acceptance conditions for the relation $\simeq_G$ is given exactly by the mappings $f : \mathcal{C}ycles(G) \to \{\text{Accept}, \text{Reject}\}$.*

Clearly, two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $\mathcal{A}_1 \simeq \mathcal{A}_2$ recognise the same language. However, the converse only holds for deterministic automata.

Lemma II.3.

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two deterministic automata over the same underlying graph and with the same labelling by input letters. Then, $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$ if and only if $\mathcal{A}_1 \simeq \mathcal{A}_2$.

Proof. The implication from right to left is trivial. For the other implication, suppose that $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$, and let $\rho \in \mathcal{R}un(\mathcal{A}_1) = \mathcal{R}un(\mathcal{A}_2)$ be an infinite run over the underlying graph of $\mathcal{A}_1$. Let $w \in \Sigma^\omega$ be the word over the input alphabet Σ labelling the run ρ. Since $\mathcal{A}_1$ and $\mathcal{A}_2$ are deterministic, ρ is the only run over w, and therefore:

$$\rho \text{ is accepting for } \mathcal{A}_1 \iff w \in \mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2) \iff \rho \text{ is accepting for } \mathcal{A}_2. \qquad \blacktriangleleft$$

1.3 Composition of an automaton and a transition system

We now present the construction of the composition (or product) of a transition system with an automaton, which constitutes the standard method for transforming a transition system that uses an acceptance set W_1 to another one using a different acceptance set W_2. To guarantee the correctness of the resulting transition system (that is, that it has the same semantic properties as the original one), the automaton must be deterministic or history-deterministic (see Propositions II.4, II.6, and II.5).

The composition construction. Let $\mathcal{TS} = (G_{\mathcal{TS}}, \mathrm{Acc}_{\mathcal{TS}})$ be a transition system, with $G_{\mathcal{TS}} = (V, E, \mathsf{source}_{\mathcal{TS}}, \mathsf{target}_{\mathcal{TS}}, I_{\mathcal{TS}})$ and $\mathrm{Acc}_{\mathcal{TS}} = (\mathsf{col}_{\mathcal{TS}}, \Sigma, W_{\mathcal{TS}})$, and let $\mathcal{A} = (G_{\mathcal{A}}, \mathrm{Acc}_{\mathcal{A}}, (\mathsf{let}_{\mathcal{A}}, \Sigma))$ be a complete automaton over the alphabet Σ, where $G_{\mathcal{A}} = (Q, \Delta, \mathsf{source}_{\mathcal{A}}, \mathsf{target}_{\mathcal{A}}, I_{\mathcal{A}})$ and $\mathrm{Acc}_{\mathcal{A}} = (\mathsf{col}_{\mathcal{A}}, \Gamma, W_{\mathcal{A}})$. The *composition* of $\mathcal{TS}$ and $\mathcal{A}$ (also called their *product*) is the transition system $\mathcal{TS} \ltimes \mathcal{A}$ defined as follows:

- The set of vertices is the cartesian product $V \times Q$.
- The set of initial vertices is $I_{\mathcal{TS}} \times I_{\mathcal{A}}$.
- The set of edges $E^{\ltimes}$ contains a transition $(v, q) \xrightarrow{c} (v', q')$ if there is $a \in \Sigma$ and transitions $e_1 = v \xrightarrow{a} v' \in E$ and $e_2 = q \xrightarrow{a:c} q' \in \Delta$. It also contains ε-edges

$(v, q) \xrightarrow{\varepsilon} (v', q)$ if $v \xrightarrow{\varepsilon} v' \in E$. Formally,

$$E^{\ltimes} = \{(e_1, e_2) \in E \times \Delta \mid \mathsf{col}_{\mathcal{TS}}(e_1) = \mathsf{let}_{\mathcal{A}}(e_2)\} \cup \{e_1 \in E \mid \mathsf{col}_{\mathcal{TS}}(e_1) = \varepsilon\} \subseteq (E \times \Delta) \cup E.$$

- The acceptance condition is inherited from that of $\mathcal{A}$: the colouring function $\mathsf{col}' \colon E^{\ltimes} \to \Gamma$ is defined as $\mathsf{col}'(e_1, e_2) = \mathsf{col}_{\mathcal{A}}(e_2)$, and the acceptance set is $W_{\mathcal{A}} \subseteq \Gamma^{\omega}$.

We remark that if $\mathcal{TS}$ does not contain an uncoloured cycle, neither does $\mathcal{TS} \ltimes \mathcal{A}$. Also, $\mathcal{TS} \ltimes \mathcal{A}$ does not contain sinks by completeness of $\mathcal{A}$.

If $\mathcal{TS}$ is a labelled transition system, labelled by the functions l_V and l_E, we consider $\mathcal{TS} \ltimes \mathcal{A}$ as a labelled transition system with the functions $l_V^{\ltimes}(v, q) = l_V(v)$ and $l_E^{\ltimes}(e_1, e_2) = l_E(e_1)$ (resp. $l_E^{\ltimes}(e_1) = l_E(e_1)$ if e_1 is an uncoloured edge).

Intuitively, a computation in $\mathcal{TS} \ltimes \mathcal{A}$ happens as follows: we start from a vertex $v_0 \in I_{\mathcal{TS}}$ in $\mathcal{TS}$ and from $q_0 \in I_{\mathcal{A}}$. When we are in a position $(v, q) \in V \times Q$, a transition e between v and v' takes place in $\mathcal{TS}$, producing a letter $a \in \Sigma$ as output. Then, the automaton $\mathcal{A}$ proceeds using a transition corresponding to a, producing an output in Γ. In this way, a word in Γ^{ω} is generated and we can use the acceptance set $W_{\mathcal{A}} \subseteq \Gamma^{\omega}$ of the automaton as the acceptance set for $\mathcal{TS} \ltimes \mathcal{A}$.

If $\mathcal{A}$ recognises the acceptance set of $\mathcal{TS}$, and under some further hypothesis, their composition $\mathcal{TS} \ltimes \mathcal{A}$ share the semantic properties of $\mathcal{TS}$.

Composition of automata. In particular, we can perform this operation if $\mathcal{TS}$ is an automaton. We obtain in this way a new automaton that uses the acceptance condition of $\mathcal{A}$.

Proposition II.4 (Folklore)**.**

Let $\mathcal{B}$ be an automaton with acceptance set $W_{\mathcal{B}}$ and let $\mathcal{A}$ be an automaton recognising $\mathcal{L}(\mathcal{A}) = W_{\mathcal{B}}$. Then, $\mathcal{L}(\mathcal{B} \ltimes \mathcal{A}) = \mathcal{L}(\mathcal{B})$. Moreover, if $\mathcal{A}$ and $\mathcal{B}$ are deterministic (resp. history-deterministic), so is $\mathcal{B} \ltimes \mathcal{A}$.

Games suitable for transformations. We could apply this construction to a game $\mathcal{G}$, obtaining a new game $\mathcal{G} \ltimes \mathcal{A}$ in which the player who makes a move in $\mathcal{G}$ also chooses a transition in $\mathcal{A}$ corresponding to the letter produced by the selected move. However, in most applications, we intend to obtain an asymmetric form of product game in which one player has full control of the transitions of the automaton (we take the point of view of Eve and want her to choose these transitions). For this reason, we restrain the class of games to which we can apply the product construction by a non-deterministic automaton.

We say that a game is *suitable for transformations* if it satisfies that for every edge $e = v \to v'$ such that $v \in V_{\text{Adam}}$, the edge e is uncoloured ($\mathsf{col}(e) = \varepsilon$), $v' \in V_{\text{Eve}}$, and e is the only incoming edge to v' ($\mathsf{In}(v') = \{e\}$). We remark that any game $\mathcal{G}$ can be made suitable for transformations with at most a linear blow up on the size by inserting an intermediate Eve-vertex in each edge outgoing from an Adam-vertex. A formal construction, as well as further motivation for this definition, can be found in Appendix B.1.

Good-for-gameness. The goal of performing the composition of a game with an automaton is to obtain a game $\mathcal{G} \ltimes \mathcal{A}$ that uses a simpler winning condition than that of $\mathcal{G}$, so we can solve the new game more easily. Of course, this is of some interest only

if the games $\mathcal{G}$ and $\mathcal{G} \ltimes \mathcal{A}$ have the same winner, which is not in general the case if the automaton $\mathcal{A}$ is not deterministic. However, the winner of a game is preserved by composition with deterministic automata, and also with some non-deterministic ones.

We say that an automaton $\mathcal{A}$ is *good-for-games* if there is some initial vertex q_{init} of $\mathcal{A}$ such that for every game suitable for transformations $\mathcal{G}$ using the winning condition $\mathcal{L}(\mathcal{A})$, Eve wins the game $\mathcal{G}$ from a vertex v if and only if she wins $\mathcal{G} \ltimes \mathcal{A}$ from (v, q_{init}).

In fact, good-for-games automata are exactly history-deterministic ones, which was the main motivation for the introduction of history-determinism [HP06]. However, it should be noted that history-determinism and good-for-gameness have been generalised to other contexts in which they do not necessarily yield equivalent notions [Col09, BL21].

Proposition II.5 ([HP06])**.**

An automaton $\mathcal{A}$ is good-for-games if and only if it is history-deterministic.

In particular, we have the following property, that we generalise in Section II.2.4.

Proposition II.6 ([HP06])**.**

Let $\mathcal{G}$ be a game that is suitable for transformations with winning condition $W_{\mathcal{G}}$, and let $\mathcal{A}$ be a history-deterministic automaton recognising $\mathcal{L}(\mathcal{A}) = W_{\mathcal{G}}$. Then, the winning region of Eve in $\mathcal{G}$ is the projection of her winning region in $\mathcal{G} \ltimes \mathcal{A}$, that is, there is an initial vertex q_{init} of $\mathcal{A}$ such that Eve wins $\mathcal{G}$ from a vertex v if and only if she wins $\mathcal{G} \ltimes \mathcal{A}$ from (v, q_{init}).

1.4 Trees

We introduce some technical notations that will be used to define automata based on the Zielonka tree (Sections II.3.2 and II.3.3) and the transformations based on the ACD (Sections II.4.2 and II.4.3).

Basic notations about trees. A *tree* $T = (N, \preceq)$ is a non-empty finite set of *nodes* N equipped with an order relation $\preceq$ called the *ancestor relation* (we say that x is an *ancestor* of y, or that y is below x if $x \preceq y$), such that (1) there is a minimal node for $\preceq$, called *the root*, and (2) the ancestors of an element are totally ordered by $\preceq$. The converse relation $\succeq$ is the *descendant* relation. Maximal nodes are called *leaves*, and the set of leaves of T is denoted by $\mathsf{Leaves}(T)$. The minimal strict descendants of a node are called its *children*. The set of children of n in T is written $\mathsf{Children}_T(n)$. The *depth* of a node n is the number of strict ancestors of it. We note it $\mathsf{Depth}(n)$. The *height* of a tree T is the maximal length of a chain for the ancestor relation. A *subtree* of $T = (N, \preceq)$ is a tree $T' = (N', \preceq')$ such that $N' \subseteq N$, $\preceq'$ is the restriction of $\preceq$ to N' and $\mathsf{Children}_{T'}(n') \subseteq \mathsf{Children}_T(n')$ for all $n' \in N'$. Given a node n of a tree T, the *subtree of T rooted at n* is the subtree of T whose nodes are the nodes of T that have n as ancestor. A *branch* is a maximal chain of the order $\preceq$. An *A-labelled tree* is a tree T together with a labelling function $\nu \colon N \to A$. A set of trees is called a *forest*.

Ordered trees. An *ordered tree* is a tree $T = (N, \preceq)$ together with a total order $\leq_n$ over $\mathsf{Children}_T(n)$, for each node $n \in N$ that is not a leaf. We remark that a subtree of an

ordered tree can be seen as an ordered tree with the restrictions of these total orders to the existing children. These orders induce a total order $\leq_T$ on T (the depth-first order): let $n, n' \in N$. If $n \preceq n'$, we let $n \leq_T n'$. If n and n' are incomparable for the ancestor relation, let n_m be the deepest common ancestor, and let $n_1, n_2 \in \mathsf{Children}_T(n_m)$ such that $n_1 \preceq n$ and $n_2 \preceq n'$. We let $n \leq_T n'$ if and only if $n_1 \leq_{n_m} n_2$. In the latter case, we say that n is *on the left of* n'.

Navigating in ordered trees. We will make use of these orders through some auxiliary functions. The function $\mathsf{Next}(n)$ gives the next sibling of n in the tree, in a cyclic order. Two examples are shown on the left of Figure 8. The function $\mathsf{Jump}(n, n_m)$ (for n_m an ancestor of n) outputs the node given by the following procedure: we go up the tree from n to n_m; then, we change to the next branch below n_m (in a cyclic way) and go down again taking the leftmost leaf below it. Examples are given on the right of Figure 8.

$\mathsf{Next}(5) = 4$ and $\mathsf{Next}(2) = 3$. $\mathsf{Jump}(5, 1) = 4$ and $\mathsf{Jump}(6, 0) = 7$.

✦ **Figure 8.** Illustration of the functions Next and Jump.

We give the formal definition now. We also need to define these notions taking into account some subtree T' of T: the input can be any node in T, but the final output is restricted to be a node in T'. Examples II.29 (Section II.3.1) and II.55 (Section II.4.1) further illustrate these notations.

Let T' be a subtree of T and n' a node of T' that is not a leaf in T'. For $n \in \mathsf{Children}_T(n')$, we let

$$\mathsf{Next}_{T'}(n) = \begin{cases} \min_{\leq_{n'}}\{n'' \in \mathsf{Children}_{T'}(n') \mid n <_{n'} n''\} & \text{if this set is not empty,} \\ \min_{\leq_{n'}}\{n'' \in \mathsf{Children}_{T'}(n')\} & \text{otherwise.} \end{cases}$$

That is, the function $\mathsf{Next}_{T'}$ maps each child of n' to a sibling that is its successor in T' for the $\leq_n$-order, in a cyclic way.

Let $T' = (N', \preceq')$ be a subtree of $T = (N, \preceq)$. Let $n' \in N'$ and $n \in N$ such that n' is a (non-strict) ancestor of n ($n' \preceq n$). If n' is a leaf of T', we define $\mathsf{Jump}_{T'}(n, n') = n'$. For $n' = n$, we define $\mathsf{Jump}_{T'}(n, n')$ to be the leftmost leaf of T' below n'. In any other case, we define $\mathsf{Jump}_{T'}(n, n') = l_{\text{dest}} \in \mathsf{Leaves}(T')$ to be the only node satisfying that there are two children of n' in T, $n_1, n_2 \in \mathsf{Children}_T(n')$ such that:

- $n_1 \preceq n$,
- $n_2 = \mathsf{Next}_{T'}(n_1)$ (in particular, $n_2 \in N'$),
- $l_{\text{dest}} \succeq n_2$ is the leftmost[3] leaf in T' (minimal for $\leq_{T'}$) below n_2.

We remark that $n_1 = n_2$ if n_1 is the only child of n' in T'.

Directed acyclic graphs. A *directed acyclic graph* (DAG) $(N, \preceq)$ is a non-empty finite set of nodes N equipped with an order relation $\preceq$ called the *ancestor relation* such that there is a minimal node for $\preceq$, called *the root*. We apply to DAGs similar vocabulary than for trees (*children*, *leaves*, *depth*, *subDAG rooted at a node*...). An *A-labelled DAG* is a DAG together with a labelling function $\nu\colon D \to A$.

2 Morphisms as witnesses of transformations

As mentioned in the introduction, all existing transformations of automata follow a common approach: they turn each state q into multiple states of the form (q, x), where x stores some information about the acceptance condition. It is reasonable to put forward this characteristic as the defining trait establishing that an automaton has been obtained as a transformation of another. In this section, we introduce morphisms of transition systems, which formalise this idea: a morphism $\varphi\colon \mathcal{B} \to \mathcal{A}$ witnesses that each state $q \in \mathcal{A}$ has been augmented to $\varphi^{-1}(q)$. To ensure that $\mathcal{B}$ is semantically equivalent to $\mathcal{A}$, the morphism has to grant a further guarantee, namely, we need to be able to simulate runs of $\mathcal{A}$ in $\mathcal{B}$. We will examine two properties of morphisms that allow to do this: local bijectivity and history-determinism for mappings.

Almost identical notions of morphisms have been considered by Sakarovitch [Sak98, Section 2] and Sakarovitch and de Souza [SS10, Section 2.5] in the context of transducers over finite words.[4] Similar ideas to the ones presented here were used by Colcombet to characterise history-deterministic automata: an automaton is history-deterministic if it is the homomorphic image of a (possibly infinite) deterministic automaton for the same language [Col12, Definition 13].

In the entire section, $\mathcal{TS} = (G, \mathrm{Acc})$ and $\mathcal{TS}' = (G', \mathrm{Acc}')$ will stand for transition systems with underlying graphs $G = (V, E, \mathsf{source}, \mathsf{target}, I)$ and $G' = (V', E', \mathsf{source}', \mathsf{target}', I')$, and acceptance conditions $\mathrm{Acc} = (\mathsf{col}, \Gamma, W)$ and $\mathrm{Acc}' = (\mathsf{col}', \Gamma', W')$.

2.1 Morphisms of transition systems

Definition II.7.

A *morphism of graphs* from G to G' is a pair of mappings $\varphi = (\varphi_V : V \to V', \varphi_E : E \to E')$ preserving edges, that is:

- $\mathsf{source}'(\varphi_E(e)) = \varphi_V(\mathsf{source}(e))$ for every $e \in E$,
- $\mathsf{target}'(\varphi_E(e)) = \varphi_V(\mathsf{target}(e))$ for every $e \in E$.

We say that φ is a *morphism of pointed graphs* if, moreover, it preserves initial vertices:

- $\varphi_V(v_0) \in I'$ for every $v_0 \in I$.

[3] The choice of the leftmost leaf is arbitrary. In all our uses of the function Jump, it could be replaced by any leaf below n_2.

[4] Thanks to Géraud Sénizergues for pointing us to the works of Sakarovitch and De Souza.

If $(G, (l_V, L_V), (l_E, L_E))$ and $(G, (l'_V, L'_V), (l'_E, L'_E))$ are labelled graphs, we say that φ is a *morphism of labelled graphs* if, in addition, $L_V \subseteq L'_V$, $L_E \subseteq L'_E$ and φ preserves labels:

- $l'_V(\varphi_V(v)) = l_V(v)$ for every $v \in V$,
- $l'_E(\varphi_E(e)) = l_E(e)$ for every $e \in V$.

We will write $\varphi\colon G \to G'$ to denote a morphism φ. We will drop the subscript in φ_V and φ_E whenever it can be deduced from its use. We say that φ is *surjective* (resp. injective) if φ_V is.

Note that the mapping φ_V does not completely determine a morphism φ, as multiple edges might exist between two given vertices. However, if G has no isolated vertices, the mapping φ_E does determine it. It will be convenient nonetheless to also keep the notation for φ_V.

We remark that the image of a run in G by a morphism of pointed graphs is a run in G'. Therefore, a morphism of pointed graphs $\varphi\colon G \to G'$ induces a mapping

$$\varphi_{\mathcal{R}uns}\colon \mathcal{R}un^\infty(G) \to \mathcal{R}un^\infty(G').$$

Definition II.8.

Let $\mathcal{TS}$ and $\mathcal{TS}'$ be two (labelled) transition systems. A *weak morphism of (labelled) transition systems* $\varphi\colon \mathcal{TS} \to \mathcal{TS}'$ is a morphism of (labelled) pointed graphs between their underlying graphs, $\varphi : G \to G'$. We say that it is a *morphism of (labelled) transition systems* if it *preserves the acceptance* of runs, that is:

- for every infinite run $\rho \in \mathcal{R}un(\mathcal{TS})$, $\mathsf{col}(\rho) \in W \iff \mathsf{col}'(\varphi_{\mathcal{R}uns}(\rho)) \in W'$.

A morphism of labelled TS between automata (resp. between games) will be called a *morphism of automata* (resp. *morphism of games*).

We say that a morphism of TS $\varphi\colon \mathcal{TS} \to \mathcal{TS}'$ is an *isomorphism* if φ_V and φ_E are bijective and $\varphi^{-1} = (\varphi_V^{-1}, \varphi_E^{-1})$ is a morphism from $\mathcal{TS}'$ to $\mathcal{TS}$. In that case, we say that $\mathcal{TS}$ and $\mathcal{TS}'$ are *isomorphic*.

We recall that two acceptance conditions are equivalent over the same underlying graph if they define the same set of accepting runs.

✦ Remark II.9. *If $\varphi\colon \mathcal{TS}_1 \to \mathcal{TS}_2$ is an isomorphism, then $(\mathsf{col}_2 \circ \varphi, \Gamma_2, W_2)$ is an acceptance condition over the underlying graph of $\mathcal{TS}_1$ that is equivalent to $(\mathsf{col}_1, \Gamma_1, W_1)$ over this graph.*

Conversely, if two acceptance conditions Acc_1 and Acc_2 are equivalent over a same graph G, then the identity function is an isomorphism between $\mathcal{TS}_1 = (G, \mathrm{Acc}_1)$ and $\mathcal{TS}_2 = (G, \mathrm{Acc}_2)$.

Therefore, we can use interchangeably the terms of equivalent and isomorphic TS, and we note $\mathcal{TS} \simeq \mathcal{TS}'$ if $\mathcal{TS}$ and $\mathcal{TS}'$ are isomorphic. In particular, $\mathcal{TS}$ is X type, for some type of acceptance conditions X, if and only if there exists an isomorphic transition system $\mathcal{TS}' \simeq \mathcal{TS}$ using an X acceptance condition.

2.2 Local properties of morphisms

Definition II.10.

A morphism of pointed graphs $\varphi : G \to G'$ is called:

- *Locally surjective* if it satisfies:
  1. For every $v'_0 \in I'$ there exists $v_0 \in I$ such that $\varphi(v_0) = v'_0$.
  2. For every $v \in V$ and every $e' \in \mathsf{Out}(\varphi(v))$ there exists $e \in \mathsf{Out}(v)$ such that $\varphi(e) = e'$.
- *Locally injective* if it satisfies:
  1. For every $v'_0 \in I'$, there is at most one $v_0 \in I$ such that $\varphi(v_0) = v'_0$.
  2. For every $v \in V$ and every couple $e_1, e_2 \in \mathsf{Out}(v)$, $\varphi(e_1) = \varphi(e_2)$ implies $e_1 = e_2$.
- *Locally bijective* if it is both locally surjective and locally injective.

Equivalently, a morphism of pointed graphs φ is locally surjective (resp. locally injective) if for every $v \in V$ the restriction of φ_E to $\mathsf{Out}(v)$ is a surjection onto $\mathsf{Out}(\varphi(v))$ (resp. an injection into $\mathsf{Out}(\varphi(v))$), and the restriction of φ_V to I is a surjection onto I' (resp. an injection into I').

Let $\varphi : \mathcal{TS} \to \mathcal{TS}'$ be a (weak) morphism, and let $\rho' = v'_0 \xrightarrow{e'_0} v'_1 \xrightarrow{e'_1} \dots$ be a run in $\mathcal{TS}'$. If φ is locally surjective, we can pick an initial vertex v_0 in $\varphi^{-1}(v'_0)$ and build step-by-step a run ρ in $\mathcal{TS}$ from v_0 that is sent to ρ' under φ. If φ is moreover locally bijective, the choices of the initial vertex and the edges at each step are unique, so runs in $\mathcal{TS}'$ can be simulated in $\mathcal{TS}$ via φ in a unique way. Said differently, if $\varphi : \mathcal{TS} \to \mathcal{TS}'$ is a locally bijective morphism, we can see $\mathcal{TS}$ as an automaton that processes runs of $\mathcal{TS}'$ in a deterministic fashion (this idea is formalised in Section 4.4.3). This property will allow us to show that a locally bijective morphism witnesses the semantic equivalence of $\mathcal{TS}$ and $\mathcal{TS}'$ (see Section II.2.4).

We note that the notion of locally bijective morphisms of transition systems almost coincides with the usual concept of bisimulation for deterministic transition systems. The main difference is that locally bijective morphisms treat the acceptance of a run as a whole; we do not impose the output colour of an edge $\mathsf{col}(e)$ to coincide with the colour $\mathsf{col}'(\varphi(e))$. This allows us to compare transition systems using different types of acceptance conditions.

✦ Remark II.11. *Let φ be a morphism of pointed graphs.*

1. *If φ is locally surjective, then $\varphi_{\mathcal{R}uns}$ is surjective.*
2. *If φ is locally injective, then $\varphi_{\mathcal{R}uns}$ is injective.*
3. *If φ is locally bijective, then $\varphi_{\mathcal{R}uns}$ is bijective.*

In the following, all weak morphisms under consideration will be locally surjective. Next lemma ensures that we can assume that they are surjective without loss of generality.

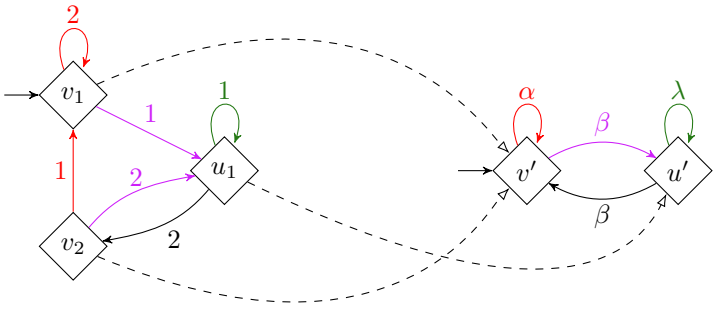
Lemma II.12.

If $\varphi\colon \mathcal{TS} \to \mathcal{TS}'$ is a locally surjective weak morphism, it is onto the accessible part of $\mathcal{TS}'$. That is, for every accessible state $v' \in \mathcal{TS}'$, there exists some state $v \in \mathcal{TS}$ such that $\varphi_V(v) = v'$. In particular, if every state of $\mathcal{TS}'$ is accessible, φ is surjective.

Proof. Let v' be an accessible state of $\mathcal{TS}'$. By definition, there exists a finite run ρ' from an initial vertex of $\mathcal{TS}'$ to v'. By surjectivity of $\varphi_{\mathcal{R}uns}$, there is a finite run $\rho \in \mathcal{R}un^{\mathsf{fin}}(\mathcal{TS})$ such that $\varphi_{\mathcal{R}uns}(\rho) = \rho'$. As φ is a morphism of graphs, we have that $\varphi(\mathsf{target}(\rho)) = v'$. ◀

Example II.13.

In Figure 9 we provide an example of a locally bijective morphism between the two rightmost transition systems from Figure 5 (we have removed input letters for simplicity). We recall that the acceptance set of the rightmost transition system is the Muller language associated to $\mathcal{F} = \{\{\alpha\}, \{\beta\}\}$. The morphism is given by $\varphi_V(v_1) = \varphi_V(v_2) = v'$ and $\varphi_V(v_2) = v'_2$. In this case, the mapping φ_V determines a unique morphism; the (uniquely determined) mapping φ_E is represented by the colours of the edges in the figure. It is easy to check that this mapping preserves the acceptance of runs and that it is locally bijective.



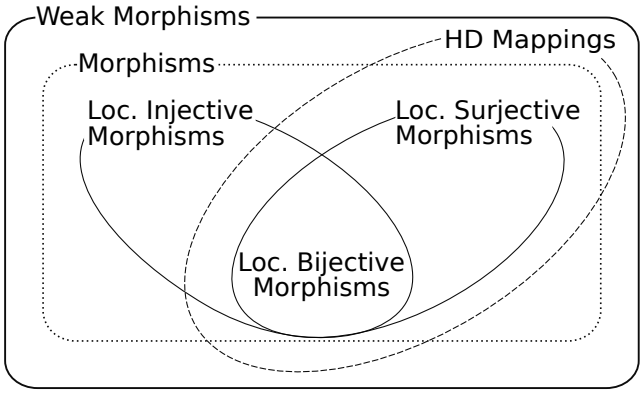
✦ **Figure 9.** A locally bijective morphism from a parity TS to a Muller TS with acceptance set given by $\mathcal{F} = \{\{\alpha\}, \{\beta\}\}$. We use dashed arrows to represent the images of vertices, and colours to represent the image of edges (that can be inferred from φ_V).

2.3 History-deterministic mappings

Locally bijective morphisms are a natural generalisation of the composition of a transition system with a deterministic automaton. They guarantee the semantic equivalence of the two involved transition systems, but at the cost of the use of some strong hypothesis, as the outgoing edges of a vertex v must exactly correspond to the outgoing edges of its image $\varphi(v)$. We can imagine correct transformations that do not satisfy this requirement. Notably, history-deterministic automata have been introduced as a method to bypass this restriction, with the hope of outperforming transformations that are witnessed by locally bijective morphisms. In general, if $\mathcal{A}$ is an HD automaton recognising the acceptance

set of $\mathcal{TS}$, the composition $\mathcal{TS} \ltimes \mathcal{A}$ does not admit a locally bijective morphism to $\mathcal{TS}$, although it shares most semantic properties with it (Proposition II.6).

We introduce next HD mappings, which are weak morphisms with the minimal set of hypothesis ensuring that, if $\varphi\colon \mathcal{TS} \to \mathcal{TS}'$ is an HD mapping, we can simulate runs of $\mathcal{TS}'$ in $\mathcal{TS}$ via φ while preserving their acceptance. This will allow us to show that φ witnesses the semantic equivalence of $\mathcal{TS}$ and $\mathcal{TS}'$ (Section II.2.4).



✦ **Figure 10.** Different types of morphisms and the relations between them. The fact that locally surjective morphisms are HD mappings is given by Lemma II.18. Note that HD mappings are also locally surjective weak morphisms (Remark II.14).

History-deterministic mappings

Let $\mathcal{TS}$ and $\mathcal{TS}'$ be transition systems and $\varphi : \mathcal{TS} \to \mathcal{TS}'$ a weak morphism between them. A *resolver simulating* φ consists of a pair of functions $\mathsf{r}_{\mathsf{Init}}\colon I' \to I$ and $\mathsf{r}\colon E^* \times E' \to E$ such that:

1. $\varphi(\mathsf{r}_{\mathsf{Init}}(v'_0)) = v'_0$ for all $v'_0 \in I'$,
2. $\varphi(\mathsf{r}(\rho, e')) = e'$, for all $\rho \in E^*$ and $e' \in E'$,
3. if $e'_0 \in \mathsf{Out}(I')$, $\mathsf{source}(\mathsf{r}(\varepsilon, e'_0)) = \mathsf{r}_{\mathsf{Init}}(\mathsf{source}(e'_0))$, and
4. if ρ is a finite run in $\mathcal{TS}$ ending in v and $e' \in \mathsf{Out}(\varphi(v))$, then $\mathsf{r}(\rho, e') \in \mathsf{Out}(v)$.

Given a run $\rho' = e'_0 e'_1 \cdots \in \mathcal{R}un^\infty(\mathcal{TS}')$ starting in some $v'_0 \in I'$, the *run induced by* r is the sequence $\mathsf{r}_{\mathcal{R}uns}(\rho') = e_0 e_1 e_2 \cdots \in \mathcal{R}un^\infty(\mathcal{TS})$ defined by $e_i = \mathsf{r}(e_0 \dots e_{i-1}, e'_i)$, which is indeed a run in $\mathcal{TS}$. We say that the resolver is *sound* if for every accepting run $\rho' \in \mathcal{R}un(\mathcal{TS}')$, the run $\mathsf{r}_{\mathcal{R}uns}(\rho')$ is accepting in $\mathcal{TS}$. Note that we do not impose $\mathsf{r}_{\mathcal{R}uns}(\rho')$ to be rejecting if ρ' is.

✦ Remark II.14. *Provided that all states of $\mathcal{TS}'$ are accessible, a resolver simulating φ can only exist if φ is a locally surjective weak morphism. We recall that φ does not need to be a morphism (see Figure 10).*

Said differently, a sound resolver simulating φ is a winning strategy for the player Duplicator in the following game:

- In round 0, Spoiler picks an initial vertex v'_0 in $\mathcal{TS}'$. Duplicator responds by picking an initial vertex v_0 in $\mathcal{TS}$ such that $\varphi(v_0) = v'_0$.
- In round $n > 0$, Spoiler picks an edge e'_n in $\mathcal{TS}'$, and Duplicator responds by picking an edge e_n in $\mathcal{TS}$ such that $\varphi(e_n) = e'_n$.

- Duplicator wins if either $e_1e_2\ldots$ is an accepting run in $\mathcal{TS}$ from v_0 or $e'_1e'_2\ldots$ is not an accepting run in $\mathcal{TS}'$ from v'_0 (it is either not a run from v_0 or not accepting). Spoiler wins otherwise.

Definition II.15.

Let $\mathcal{TS}$ and $\mathcal{TS}'$ be (labelled) transition systems. A *history-deterministic mapping* (HD mapping) of transition systems from $\mathcal{TS}$ to $\mathcal{TS}'$ is a pair of mappings $\varphi = (\varphi_V : V \to V', \varphi_E : E \to E')$ such that:

- φ is a weak morphism,
- φ *preserves accepting runs*: $\rho \in \mathcal{R}un(\mathcal{TS})$ and $\mathsf{col}(\rho) \in W \implies \mathsf{col}'(\varphi_{\mathcal{R}uns}(\rho)) \in W'$, and
- there exists a sound resolver simulating φ.

Even if a history-deterministic mapping is not necessarily locally bijective (and not even a morphism of transition systems), the existence of a sound resolver allows us to define a right inverse to $\varphi_{\mathcal{R}uns}$ preserving the acceptance of runs.

Lemma II.16.

Let $\varphi\colon \mathcal{TS} \to \mathcal{TS}'$ be an HD mapping and let $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r})$ be a sound resolver simulating it. The following holds:

- $\varphi_{\mathcal{R}uns} \circ \mathsf{r}_{\mathcal{R}uns} = \mathsf{Id}_{\mathcal{R}un^\infty(\mathcal{TS}')}$.
- $\mathsf{r}_{\mathcal{R}uns}$ preserves the acceptance of runs in $\mathcal{TS}'$, that is, for every run $\rho' \in \mathcal{R}un(\mathcal{TS}')$, ρ' is accepting if and only if $\mathsf{r}_{\mathcal{R}uns}(\rho')$ is accepting in $\mathcal{TS}$.

Proof. The first item follows from the fact that $\varphi(\mathsf{r}(\rho, e')) = e'$ for every $\rho \in E^*$ and $e' \in E'$.

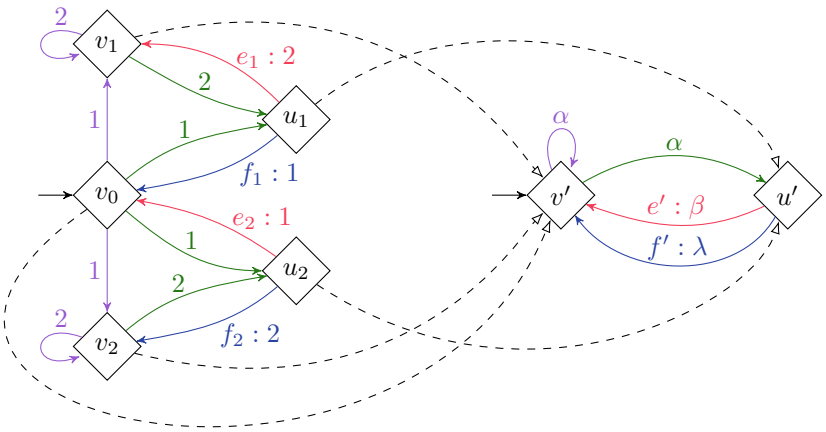
For the second item, the definition of a sound resolver imposes that if ρ' is accepting, so is $\mathsf{r}_{\mathcal{R}uns}(\rho')$. For the other direction, if $\mathsf{r}_{\mathcal{R}uns}(\rho')$ is accepting, then $\varphi_{\mathcal{R}uns}(\mathsf{r}_{\mathcal{R}uns}(\rho')) = \rho'$ has to be accepting, as an HD mapping preserves accepting runs. ◀

Example II.17.

In Figure 11 we give an example of a weak morphism $\varphi\colon \mathcal{TS} \to \mathcal{TS}'$ that is a history-deterministic mapping, but which is neither a morphism, nor locally bijective. Transition system $\mathcal{TS}$, on the left of the figure, is a parity TS (more precisely, a coBüchi TS). Transition system $\mathcal{TS}'$, depicted on the right of the figure, is a Muller TS using as acceptance set the Muller language associated to $\mathcal{F} = \{\{\alpha\}, \{\alpha, \beta\}, \{\alpha, \lambda\}\}$; that is, a run in $\mathcal{TS}'$ is accepting if and only if it eventually avoids either transition e' or transition f'. The weak morphism we propose is given by: $\varphi(v_0) = \varphi(v_1) = \varphi(v_2) = v'$, and $\varphi(u_1) = \varphi(u_2) = u'$. The image of most edges is uniquely determined, and we use colours to represent them. We have named the only edges whose image is not uniquely determined, and we define $\varphi(e_1) = \varphi(e_2) = e'$ and $\varphi(f_1) = \varphi(f_2) = f'$.

We remark that φ does not preserve rejecting runs. Indeed, a run in $\mathcal{TS}$ alternating between v_0 and u_1, taking transition f_1 infinitely often, is rejecting, but its image is accepting in $\mathcal{TS}'$. However, φ preserves accepting runs: a run is accepting in $\mathcal{TS}$ if and only if it eventually stays in $\{v_1, u_1\}$ or in $\{v_2, u_2\}$. In the first case, the image under φ avoids transition f' in $\mathcal{TS}'$, and in the second case, its image avoids transition e'.

Finally, we describe a sound resolver simulating φ. When simulating a run from $\mathcal{TS}'$ in $\mathcal{TS}$, we have a choice to make only when we are in state v_0. If the previous transition in $\mathcal{TS}'$ was e', we will go up, that is, $v' \xrightarrow{\alpha} u'$ is simulated by $v_0 \xrightarrow{1} u_1$ and $v' \xrightarrow{\alpha} v'$ is simulated by $v_0 \xrightarrow{1} v_1$. If the previous transition in $\mathcal{TS}'$ was f', we will go down in a symmetric manner. In this way, if transition f' is eventually not visited by the run in $\mathcal{TS}'$, we ensure to stay in $\{v_1, u_1\}$ in $\mathcal{TS}$ (and symmetrically, we ensure to stay in $\{v_2, u_2\}$ if e' is avoided in $\mathcal{TS}'$).



✦ **Figure 11.** A history-deterministic mapping from a parity TS to a Muller TS with acceptance set given by $\mathcal{F} = \{\{\alpha\}, \{\alpha, \beta\}, \{\alpha, \lambda\}\}$. We use dashed arrows to represent the images of vertices, and colours to represent the image of edges.

In the next lemma, we prove that HD mappings are a strict generalisation of locally surjective morphisms (and therefore, also of locally bijective ones). On the other hand, we remark that HD mappings must be locally surjective, but they are not necessarily morphisms (they might not preserve rejecting runs).

Lemma II.18.

If $\varphi : \mathcal{TS} \to \mathcal{TS}'$ is a locally surjective morphism, it is also an HD mapping.

Proof. We need to define a sound resolver simulating φ. Let $\mathsf{r}_{\mathsf{Init}} \colon I' \to I$ be any function choosing initial vertices satisfying that $\varphi \circ \mathsf{r}_{\mathsf{Init}} = \mathsf{Id}_{I'}$ (which exists by local surjectivity of φ). For each $v \in V$ and edge $e' \in \mathsf{Out}(\varphi(v))$ we choose one edge $f(v, e') \in \mathsf{Out}(v)$ such that $\varphi(f(v, e')) = e'$ (which exists by local surjectivity), and we let r be the resolver induced by these choices. Formally, we define $\mathsf{r} \colon E^* \times E' \to E$ recursively. For the base case, if $e'_0 \in \mathsf{Out}(v')$, with $v' \in I'$, we define $\mathsf{r}(\varepsilon, e'_0) = f(\mathsf{r}_{\mathsf{Init}}(v'), e'_0)$. Assume that r has been defined for runs of length $\leq n$, and let $\rho \in E^*$ be of length $n+1$ and $e' \in E'$. If ρ is not a run or $e' \notin \mathsf{Out}(\mathsf{target}(\varphi(\rho)))$, we let $\mathsf{r}(\rho, e')$ be any edge in $\varphi^{-1}(e')$. If not, let $v = \mathsf{target}(\rho)$ and we define $\mathsf{r}(\rho, e')$ to be the edge $f(v, e')$.

It is straightforward to check that $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r})$ is indeed a resolver (for every run $\rho' \in \mathcal{R}un(\mathcal{TS}')$, the sequence $\mathsf{r}_{\mathcal{R}uns}(\rho')$ is a run in $\mathcal{TS}$ and ρ' is its image under φ). Finally, since φ is a morphism, for every $\rho' \in \mathcal{R}un(\mathcal{TS}')$, $\mathsf{r}_{\mathcal{R}uns}(\rho')$ is accepting in $\mathcal{TS}$ if and only if $\rho' = \varphi_{\mathcal{R}uns}(\rho)$ is accepting in $\mathcal{TS}'$. We conclude that $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r})$ is a sound resolver and therefore φ is an HD mapping. ◀

Restrictions and extensions of initial sets

We now include some technical considerations regarding how modifying the set of initial vertices of the transition systems can impact the existence of HD mappings between them. The following simple lemma states that reducing the number of initial vertices preserves the history-determinism of mappings.

Lemma II.19.

Let $\mathcal{TS}$ and $\mathcal{TS}'$ be two TS such that there is an HD mapping $\varphi\colon \mathcal{TS} \to \mathcal{TS}'$. For any non-empty subset $\tilde{I} \subseteq I'$, φ is also an HD mapping between the transition systems $\mathcal{TS}_{\mathsf{r}_{\mathsf{Init}}(\tilde{I})}$ and $\mathcal{TS}'_{\tilde{I}}$; that is, the transitions systems obtained by setting $\mathsf{r}_{\mathsf{Init}}(\tilde{I})$ and $\tilde{I}$ as initial vertices, respectively.

For arbitrary acceptance conditions, enlarging the set of initial vertices does not preserve history-determinism. However, for transition systems using the acceptance conditions considered in this work, we can enlarge the set of initial vertices without loss of generality.

Lemma II.20.

Let $\mathcal{TS}$ and $\mathcal{TS}'$ be two TS such that all their states are accessible, and let $\varphi\colon \mathcal{TS} \to \mathcal{TS}'$ be an HD mapping between them. If the acceptance sets W and W' are prefix-independent, the mapping φ is also HD when considered between the transition systems $\mathcal{TS}_V$ and $\mathcal{TS}'_{V'}$, consisting of the transition systems $\mathcal{TS}$ and $\mathcal{TS}'$ where all the states are set to be initial.

Proof. First, $\varphi\colon \mathcal{TS}_V \to \mathcal{TS}'_{V'}$ is trivially a weak morphism. We claim that it preserves accepting runs. Let $v \in V$ be a state in $\mathcal{TS}$ and let $\rho = v \overset{w}{\rightsquigarrow}$ be an accepting run from v. Since all the states are reachable, there is some $v_0 \in I$ and finite run $\rho_v = v_0 \overset{u}{\rightsquigarrow} v$. As φ is a weak morphism we have that $\varphi_{\mathcal{R}uns}(\rho_v\rho) = \varphi(v_0) \overset{u'}{\rightsquigarrow} \varphi(v) \overset{w'}{\rightsquigarrow}$. It is satisfied:

$$w \in W \overset{W}{\underset{\text{pref-indep.}}{\Longrightarrow}} uw \in W \implies u'w' \in W' \overset{W'}{\underset{\text{pref-indep.}}{\Longrightarrow}} w' \in W',$$

where the central implication follows from the fact that φ preserves accepting runs starting in v_0. Therefore $\varphi_{\mathcal{R}uns}(\rho)$ is also an accepting run.

Let $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r})$ be a sound resolver simulating $\varphi\colon \mathcal{TS} \to \mathcal{TS}'$. We define a resolver $(\tilde{\mathsf{r}}_{\mathsf{Init}}, \tilde{\mathsf{r}})$ for the new mapping.

For every state v of $\mathcal{TS}$, we fix (whenever it exists) a finite run $\rho_v \in \mathcal{R}un^{\mathsf{fin}}(\mathcal{TS})$ ending in v such that there exists $\rho' \in \mathcal{R}un^{\mathsf{fin}}(\mathcal{TS}')$ such that $\rho_v = \mathsf{r}_{\mathcal{R}uns}(\rho'_{v'})$. We let $V_{\text{Reach}} \subseteq V$ be the set of vertices for which ρ_v is well-defined. We note that for each $v' \in V'$ there exists at least one $v \in \varphi^{-1}(v')$ such that $v \in V_{\text{Reach}}$; indeed, if $\rho'_{v'}$ is a finite run reaching v' in $\mathcal{TS}'$, one such v is $\mathsf{target}(\mathsf{r}_{\mathcal{R}uns}(\rho'_{v'}))$ (that is, the vertex to which we arrive in $\mathcal{TS}$ when simulating $\rho'_{v'}$ via the original resolver). We let $\tilde{\mathsf{r}}_{\mathsf{Init}}(v')$ be this vertex.

If $e' \in \mathsf{Out}(v')$ is an edge in $\mathcal{TS}'$, we let $\tilde{\mathsf{r}}(\varepsilon, e') = \mathsf{r}(\rho_v, e')$, for $v = \tilde{\mathsf{r}}_{\mathsf{Init}}(v')$ (which belongs to V_{Reach}). For ρ a non-empty finite run starting in $v \in V_{\mathrm{Reach}}$ and $e' \in E'$, we define $\tilde{\mathsf{r}}(\rho, e') = \mathsf{r}(\rho_v \rho, e')$. If ρ starts in $v \notin V_{\mathrm{Reach}}$ we let $\tilde{\mathsf{r}}(\rho, e')$ be any edge in $\varphi^{-1}(e')$ (if $e' \in \mathsf{Out}(\varphi(\mathsf{target}(\rho)))$ we pick it in $\mathsf{Out}(\mathsf{target}(\rho))$). We check that $(\tilde{\mathsf{r}}_{\mathsf{Init}}, \tilde{\mathsf{r}})$ satisfies the four requirements to be a resolver:

1. $\tilde{\mathsf{r}}_{\mathsf{Init}}(v')$ has been chosen in $\varphi^{-1}(v')$.
2. $\tilde{\mathsf{r}}(e')$ is chosen in $\varphi^{-1}(e')$.
3. Let $e' \in \mathsf{Out}(v')$. We have defined $\tilde{\mathsf{r}}(\varepsilon, e') = \mathsf{r}(\rho_v, e')$, where ρ_v is a finite run ending in v, so by Property 4 of a resolver, $\mathsf{r}(\rho_v, e') \in \mathsf{Out}(v)$.
4. Let $\rho = v_0 \rightsquigarrow v \in \mathcal{R}un^{\mathsf{fin}}(\mathcal{TS}_V)$ and $e' \in \mathsf{Out}(\varphi(v))$. If $v_0 \notin V_{\mathrm{Reach}}$, then we have picked $\tilde{\mathsf{r}}(\rho, e')$ in $\mathsf{Out}(v)$. If $v_0 \in V_{\mathrm{Reach}}$, then $\tilde{\mathsf{r}}(\rho, e') = \mathsf{r}(\rho_{v_0}\rho, e')$; as $\rho_{v_0}\rho$ is a run ending in v and r satisfies Property 4 of a resolver, $\mathsf{r}(\rho_{v_0}\rho, e') \in \mathsf{Out}(v)$.

Finally, we show that $(\tilde{\mathsf{r}}_{\mathsf{Init}}, \tilde{\mathsf{r}})$ is sound. Let $\rho' = v' \overset{w'}{\rightsquigarrow} \in \mathcal{R}un(\mathcal{TS}'_{V'})$ be an accepting run, and let $\rho = \mathsf{r}_{\mathcal{R}uns}(\rho') = v \overset{w'}{\rightsquigarrow}$ be the run induced by $(\tilde{\mathsf{r}}_{\mathsf{Init}}, \tilde{\mathsf{r}})$ over ρ'. In particular, $v = \tilde{\mathsf{r}}_{\mathsf{Init}}(v')$. Let $\rho_v = v_0 \overset{u}{\rightsquigarrow} v$ and $\rho'_v = v'_0 \overset{u'}{\rightsquigarrow} v'$ be the chosen runs such that $\rho_v = \mathsf{r}_{\mathcal{R}uns}(\rho'_v)$. It is immediate to check that $\rho_v \rho = \mathsf{r}_{\mathcal{R}uns}(\rho'_v \rho')$. Since ρ' is accepting, we have that $w' \in W'$, and by prefix-independence of the acceptance sets and the fact that $\rho_v\rho$ is accepting if $\rho'_v\rho'$ is, we have:

$$w' \in W' \implies u'w' \in W' \implies uw \in W \implies w \in W,$$

so we conclude that ρ is accepting in $\mathcal{TS}$, as we wanted to show. ◀

2.4 Preservation of semantic properties of automata and games

We start this section by showing that locally bijective morphisms and HD mappings are a strict generalisation of the composition of a TS by deterministic and history-deterministic automata, respectively (Proposition II.21). Then, we prove that these mappings witness the semantic equivalence of the transition systems under consideration. That is, the language recognised by automata (Proposition II.22) and the winner of games Proposition II.23).

Morphisms generalise composition by an automaton

Proposition II.21.

Let $\mathcal{A}$ be a complete automaton accepting the language $\mathcal{L}(\mathcal{A}) = W \subseteq \Sigma^\omega$, and let $\mathcal{TS}$ be a (labelled) TS with acceptance set W. Then, there exists a locally surjective weak morphism of (labelled) transition systems $\varphi \colon \mathcal{TS} \ltimes \mathcal{A} \to \mathcal{TS}$ that preserves accepting runs. Moreover:

1. If $\mathcal{A}$ is deterministic, φ can be chosen to be a locally bijective morphism.
2. If $\mathcal{A}$ is HD, then φ can be chosen to be an HD mapping.

Proof. We recall that the set of states of $\mathcal{TS} \ltimes \mathcal{A}$ is $V \times Q$ and its set of transitions $E^{\ltimes}$ is a subset of $(E \times \Delta) \sqcup E$, where V and Q (resp. E and Δ) are the states (resp.

transitions) of $\mathcal{TS}$ and $\mathcal{A}$, respectively. We let $W_{\mathcal{A}} \subseteq \Gamma^\omega$ be the acceptance set of $\mathcal{A}$. We define $\varphi_V(v,q) = v$ and $\varphi_E(e_1,e_2) = e_1$ for $(e_1,e_2) \in E \times \Delta$ and $\varphi_E(e_1) = e_1$ for $e_1 \in E$. It is immediate to check that φ is a weak morphism.

Given a run $\rho = (v_0,q_0) \xrightarrow{c_0} (v_1,q_1) \xrightarrow{c_1} \dots$ in $\mathcal{TS} \ltimes \mathcal{A}$, we can consider its projection over $\mathcal{TS}$, $\varphi_{\mathcal{R}uns}(\rho) = v_0 \xrightarrow{a_0} v_1 \xrightarrow{a_1} \dots$. We note that there must exist a unique run in $\mathcal{A}$ of the form

$$\varphi_{\mathcal{A}}(\rho) = q_0 \xrightarrow{a_0:c_0} q_1 \xrightarrow{a_1:c_1} \dots.$$

(Formally, some of the letters a_i might equal ε, and in this case $q_i \xrightarrow{a_i:c_i} q_{i+1}$ does not appear in the run $\varphi_{\mathcal{A}}(\rho)$).

We show that φ preserves accepting runs. Let ρ be an accepting run in $\mathcal{TS} \ltimes \mathcal{A}$. In that case, $c_0c_1c_2\dots \in W_{\mathcal{A}}$, and therefore $\varphi_{\mathcal{A}}(\rho)$ is an accepting run in $\mathcal{A}$ over $a_0a_1a_2\dots$, so we conclude that $a_0a_1a_2\dots \in W$ and $\varphi_{\mathcal{R}uns}(\rho)$ is an accepting run in $\mathcal{TS}$.

We prove next the local surjectivity of φ. Clearly, φ induces a surjection between the initial vertices of $\mathcal{TS} \ltimes \mathcal{A}$ (which are $I_{\mathcal{TS}} \times I_{\mathcal{A}}$) and those of $\mathcal{TS}$. Let $(v,q) \in V \times Q$ and $e_1 = v \xrightarrow{a} v' \in E$. If $a = \varepsilon$, the edge e_1 belongs to $E^{\ltimes}$ and $\varphi(e_1) = e_1$. If $a \neq \varepsilon$, since $\mathcal{A}$ is complete there is a transition $e_2 = q \xrightarrow{a} q' \in \Delta$ and $\varphi(e_1,e_2) = e_1$, so φ is locally surjective.

1. Since $\mathcal{A}$ has a single initial state q_0, φ induces a bijection between the initial vertices of $\mathcal{TS} \ltimes \mathcal{A}$ (which are $I_{\mathcal{TS}} \times \{q_0\}$) and those of $\mathcal{TS}$. Let $E^{\ltimes}_{\mathrm{col}} \subseteq E \times \Delta$ and $E^{\ltimes}_{\varepsilon} \subseteq E$ such that $E^{\ltimes} = E^{\ltimes}_{\mathrm{col}} \cup E^{\ltimes}_{\varepsilon}$. We remark that $\varphi|_{E^{\ltimes}_{\varepsilon}}$ is the identity function (so injective) and that $\varphi(E^{\ltimes}_{\mathrm{col}}) \cap \varphi(E^{\ltimes}_{\varepsilon}) = \emptyset$ because $\varphi(E^{\ltimes}_{\mathrm{col}})$ are exactly coloured transitions of $\mathcal{TS}$. Finally, let (e_1,e_2) and (e_1',e_2') in $\mathsf{Out}(v,q) \cap E^{\ltimes}_{\mathrm{col}}$. Their $\varphi(e_1,e_2) = \varphi(e_1',e_2')$ if and only if $e_1 = e_1'$. Let $a \in \Sigma$ be the colour of e_1. Since $\mathcal{A}$ is deterministic, there is at most one transition from q labelled by a, that must be $e_2 = e_2'$. We conclude that $(e_1,e_2) = (e_1',e_2')$ and that φ is locally injective.

   Let ρ be a rejecting run in $\mathcal{TS} \ltimes \mathcal{A}$ (we use the notations introduced above). In that case, $c_0c_1c_2\dots \notin W_{\mathcal{A}}$, and therefore $\varphi_{\mathcal{A}}(\rho)$ is a rejecting run over $a_0a_1a_2\dots$. Since $\mathcal{A}$ is deterministic, this is the only run over $a_0a_1a_2\dots$ so we conclude that it does not belong to W. We conclude that $\varphi_{\mathcal{R}uns}(\rho)$ is a rejecting run in $\mathcal{TS}$.

2. Let $(r_0,\mathsf{r}_{\mathcal{A}})$ be a resolver for $\mathcal{A}$. We define a resolver $(\mathsf{r}_{\mathsf{Init}},\mathsf{r}_{\varphi})$ simulating φ that follows the runs indicated by $(r_0,\mathsf{r}_{\mathcal{A}})$. First, we let $\mathsf{r}_{\mathsf{Init}}(v_0) = (v_0,r_0)$ for all $v_0 \in I_{\mathcal{TS}}$. We define $\mathsf{r}_{\varphi}\colon E^{\ltimes *} \times E \to E^{\ltimes}$ by induction on the length of the runs. Let $e_0 = v_0 \xrightarrow{a} v_1 \in \mathsf{Out}(v_0)$ be an edge in $\mathcal{TS}$. If e_0 is uncoloured ($a = \varepsilon$), we let $\mathsf{r}_{\varphi}(\varepsilon,e_0) = e_0 = (v_0,r_0) \xrightarrow{\varepsilon} (v_1,r_0)$. If not, we let $\mathsf{r}_{\varphi}(\varepsilon,e_0) = (e_0,e_a)$, where $e_a = \mathsf{r}(\varepsilon,a)$. Assume that r_{φ} has been defined for sequences of edges of $\mathcal{TS} \ltimes \mathcal{A}$ of length $< n$ and let $\rho = e_0e_1\dots e_{n-1} \in E^{\ltimes *}$ be a sequence length $n+1$ and $e_{\mathcal{TS}} = v_n \xrightarrow{a_n} v_{n+1}$ be an edge in $\mathcal{TS}$. If ρ is not a run or if it does not end in $\varphi^{-1}(v_n)$, we let $\mathsf{r}_{\varphi}(\rho,e_{\mathcal{TS}})$ be any edge in $\varphi^{-1}(e_{\mathcal{TS}})$. Assume that ρ is a run ending in $\varphi^{-1}(v_n)$. If $a_n = \varepsilon$, we define $\mathsf{r}_{\varphi}(\rho,e_{\mathcal{TS}}) = e_{\mathcal{TS}}$. As noted before, ρ induces a run $\varphi_{\mathcal{A}}(\rho) = q_0 \xrightarrow{a_0:c_0} q_1 \xrightarrow{a_1:c_1} \dots \to q_n$ in $\mathcal{A}$. We let $e_{\mathcal{A}} = \mathsf{r}_{\mathcal{A}}(\varphi_{\mathcal{A}}(\rho),a_n)$ be the transition chosen by the resolver of $\mathcal{A}$ after this run, and we define $\mathsf{r}_{\varphi}(\rho,e_{\mathcal{TS}}) = (e_{\mathcal{TS}},e_{\mathcal{A}})$.

   It directly follows from this definition that $(\mathsf{r}_{\mathsf{Init}},\mathsf{r}_{\varphi})$ is indeed a resolver. The proof that if $\rho \in \mathcal{R}un(\mathcal{TS})$ is an accepting run then $\mathsf{r}_{\varphi,\mathcal{R}uns}(\rho)$ is accepting follows the same lines than the previous item. ◀

Morphisms witness equivalence of automata

Proposition II.22.

Let $\mathcal{A}$, $\mathcal{A}'$ be two automata over the same input alphabet such that there is an HD mapping of automata $\varphi\colon \mathcal{A} \to \mathcal{A}'$. Then, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$, and $\mathcal{A}$ is HD if and only if $\mathcal{A}'$ is HD. If φ is moreover locally bijective and surjective, $\mathcal{A}$ is deterministic (resp. complete) if and only if $\mathcal{A}'$ is.

Proof. Since φ preserves accepting runs, it is clear that $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$. Since φ admits a sound resolver $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r})$, if ρ is an accepting run over $w \in \Sigma^\omega$ in $\mathcal{A}'$, then $\mathsf{r}_{\mathcal{R}uns}(\rho)$ is an accepting run over w in $\mathcal{A}$, so $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$.

Let $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r}_\varphi)$ be a sound resolver simulating φ. Assume that $\mathcal{A}$ is HD, admitting a resolver (r_0, r). A resolver (r'_0, r') for $\mathcal{A}'$ can be obtained just by composing r_φ and φ, that is: $\mathsf{r}'_0 = \varphi(r_0)$ and for $\rho' \in \mathcal{R}un^{\mathsf{fin}}(\mathcal{A}')$ and $a \in \Sigma$, $\mathsf{r}'(\rho', a) = \varphi(\mathsf{r}(\mathsf{r}_{\varphi,\mathcal{R}uns}(\rho'), a)))$. That is, given a run ρ' in $\mathcal{A}'$, we simulate it in $\mathcal{A}$ using r_φ, then, we look at what is the continuation proposed by the resolver r when we give the letter a, and we transfer back this choice to $\mathcal{A}'$ using φ. Assume now that $\mathcal{A}'$ is HD and that (r'_0, r') is a resolver for it. We define a resolver (r_0, r) for $\mathcal{A}$. We let $\mathsf{r}_0 = \mathsf{r}_{\mathsf{Init}}(r'_0)$, and for $\rho \in \mathcal{R}un^{\mathsf{fin}}(\mathcal{A})$ and $a \in \Sigma$, $\mathsf{r}(\rho, a) = \mathsf{r}_\varphi((\varphi_{\mathcal{R}uns}(\rho), \mathsf{r}(\varphi_{\mathcal{R}uns}(\rho), a))$. That is, given a run ρ in $\mathcal{A}$, we simulate it in $\mathcal{A}'$ using φ, then, we look at what is the continuation proposed by the resolver r' when we give the letter a, and we transfer back this choice to $\mathcal{A}$ using r_φ. It is a direct check that the resolvers defined this way witness that $\mathcal{A}'$ and $\mathcal{A}$, respectively, are HD.

The proof that $\mathcal{A}$ is deterministic (resp. complete) if and only if $\mathcal{A}'$ is deterministic (resp. complete), assuming surjectivity and local bijectivity of φ, follows the same lines. ◀

A subclass of automata with a restrictive amount of non-determinism that is widely study is that of *unambiguous* automata (we refer to [Col15, CM03] for a detailed exposition). An automaton is *unambiguous* if for every input word $w \in \Sigma^\omega$ there is at most one accepting run over w, and it is *strongly unambiguous* if there is at most one run over w. By Remark II.11, locally bijective morphisms also preserve (strongly) unambiguity: if $\varphi\colon \mathcal{A} \to \mathcal{A}'$ is a locally bijective morphism then $\mathcal{A}$ is (strongly) unambiguous if and only if $\mathcal{A}'$ is.

Morphisms preserve winning regions of games

It is not difficult to show that locally bijective morphisms preserve winning regions of games (a stronger result is proved in Appendix B.1).

Proposition II.23.

Let $\mathcal{G}, \mathcal{G}'$ be two games such that there is an locally bijective morphism $\varphi\colon \mathcal{G} \to \mathcal{G}'$. Eve's winning region in $\mathcal{G}'$ is the image of her winning region in $\mathcal{G}$: $\mathcal{W}in_{\mathrm{Eve}}(\mathcal{G}') = \varphi(\mathcal{W}in_{\mathrm{Eve}}(\mathcal{G}))$.

Applying Lemma II.20, we moreover obtain that if the winning condition used by the games $\mathcal{G}$ and $\mathcal{G}'$ in the previous proposition are prefix-independent, then Eve's full winning region in $\mathcal{G}'$ is the image of her full winning region in $\mathcal{G}$.

However, in the same way as HD automata are good-for-games (the composition of a game with an HD automaton preserves the winning region, c.f. Proposition II.5), we would like to have that HD mappings also preserve the winning regions of games. Unfortunately, this is not exactly the case, as HD mappings are oblivious to the owners of the vertices in the game. For instance, consider a game $\mathcal{G}$ consisting in a single vertex controlled by Adam with two self loops, labelled a and b. Let $\mathcal{G}'$ be a game with one vertex controlled by Adam, but just with a self loop labelled a. Let $W = a^\omega$ be the winning condition of both games. Clearly, Adam wins $\mathcal{G}$ (as he can produce, for example, b^ω), but loses $\mathcal{G}'$, as he can only produce a^ω. However, there is a (uniquely determined) HD mapping $\varphi\colon \mathcal{G} \to \mathcal{G}'$. The reason for this undesired behaviour is that, in the definition of HD mapping, it is Eve who choses how to simulate a play of $\mathcal{G}'$ in $\mathcal{G}$. However, in the previous example it was Adam who could choose what to do in $\mathcal{G}$.

To work out this problem, we introduce a further notion of mappings, oriented exclusively to games, in which we take into account the owner of the vertices. Our definition weakens the properties of locally bijective morphisms, and uses the minimal hypothesis to guarantee the preservation of winning regions in games. See Appendix B.1 for a definition of HD-for-games mappings and for stronger results about the preservation of winners in games.

3 The Zielonka tree: An optimal approach to Muller languages

In this section, we take a close look into the Zielonka tree, a structure introduced (under the name of *split trees*) to study Muller languages [Zie98]. We show how to use the Zielonka tree to construct minimal deterministic parity automata and minimal history-deterministic Rabin automata recognising Muller languages. In Section II.3.2, we describe the construction of a minimal deterministic parity automaton $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$ for a given Muller language $\mathsf{Muller}(\mathcal{F})$. Theorem II.2, the main contribution of this section, states the minimality of $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$ both amongst deterministic and HD parity automata. Theorem II.1 states the optimality on the number of priorities of the acceptance condition of $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$, and implies that we can determine the parity index of a Muller language from its Zielonka tree. We will use the optimality of automaton $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$ to provide a polynomial-time algorithm minimising DPAs recognising Muller languages in Section III.3.

In Section II.3.3, we describe the construction of a minimal history-deterministic Rabin automaton $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ for a Muller language $\mathsf{Muller}(\mathcal{F})$. Its minimality amongst HD automata is shown in Theorem II.4, by using the characterisation of the memory requirements of a Muller language in terms of its Zielonka tree [DJW97]. On the other hand, we will show in the next chapter that finding a minimal deterministic Rabin automaton recognising a given Muller language is NP-complete, if the language is represented by a parity or Rabin automaton, or even by its Zielonka tree (Theorem III.1). Therefore, unless $\mathsf{P} = \mathsf{NP}$, there are Muller languages for which minimal deterministic Rabin automata are strictly larger than minimal HD Rabin automata. We will explicitly show Muller languages for which minimal HD Rabin automata are exponentially smaller than deterministic ones in Section III.4. A summary of the minimal automata recognising Muller languages appears in Table 1.

Type of automata	Deterministic	History-deterministic
Parity	$\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$	$\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$
Rabin	NP-complete (Section III.1)	$\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$

✦ **Table 1.** Minimal automata recognising a Muller language $\mathsf{Muller}(\mathcal{F})$, according to the type of acceptance condition (parity or Rabin) and the form of determinism.

3.1 The Zielonka tree

Definition II.24 ([Zie98])**.**

Let $\mathcal{F} \subseteq 2^\Sigma_+$ be a family of non-empty subsets over a finite set Σ. A *Zielonka tree* for $\mathcal{F}$ (over Σ),[5] denoted $\mathcal{Z}_\mathcal{F} = (N, \preceq, \nu : N \to 2^\Sigma_+)$ is a 2^Σ_+-labelled tree with nodes partitioned into *round nodes* and *square nodes*, $N = N_\bigcirc \sqcup N_\square$, such that:

- The root is labelled Σ.
- If a node is labelled $X \subseteq \Sigma$, with $X \in \mathcal{F}$, then it is a round node, and it has a child for each maximal non-empty subset $Y \subseteq X$ such that $Y \notin \mathcal{F}$, which is labelled Y.
- If a node is labelled $X \subseteq \Sigma$, with $X \notin \mathcal{F}$, then it is a square node, and it has a child for each maximal non-empty subset $Y \subseteq X$ such that $Y \in \mathcal{F}$, which is labelled Y.

We write $|\mathcal{Z}_\mathcal{F}|$ to denote the number of nodes in $\mathcal{Z}_\mathcal{F}$.

✦ Remark II.25. *For each family of subsets $\mathcal{F} \subseteq 2^\Sigma_+$, there is only one Zielonka tree up to renaming of its nodes, so we will talk of* the *Zielonka tree of $\mathcal{F}$.*

✦ Remark II.26. *If n is a node of $\mathcal{Z}_\mathcal{F}$, then the subtree of $\mathcal{Z}_\mathcal{F}$ rooted at n is the Zielonka tree for the family $\mathcal{F}|_{\nu(n)}$ over the alphabet $\nu(n)$, that is, for the restriction of $\mathcal{F}$ to the subsets included in the label of n.*

✦ Remark II.27. *Let n be a node of $\mathcal{Z}_\mathcal{F}$ and let n_1 be a child of it. If $\nu(n_1) \subsetneq X \subseteq \nu(n)$, then $\nu(n_1) \in \mathcal{F} \iff X \notin \mathcal{F} \iff \nu(n) \notin \mathcal{F}$. In particular, if n_1, n_2 are two different children of n, then $\nu(n_1) \in \mathcal{F} \iff \nu(n_2) \in \mathcal{F} \iff \nu(n_1) \cup \nu(n_2) \notin \mathcal{F}$.*

Next lemma provides a simple way to decide if a subset $C \subseteq \Sigma$ belongs to $\mathcal{F}$ given the Zielonka tree. It follows directly from the previous remark.

✦ **Lemma II.28.** *Let $C \subseteq \Sigma$ and let n be a node of $\mathcal{Z}_\mathcal{F}$ such that $C \subseteq \nu(n)$ and that is maximal for $\preceq$ amongst nodes containing C in its label. Then, $C \in \mathcal{F}$ if and only if n is round.*

We equip Zielonka trees with an order to navigate in them. That is, we equip each set $\mathsf{Children}_{\mathcal{Z}_\mathcal{F}}(n)$ with a total order, making $\mathcal{Z}_\mathcal{F}$ an ordered tree. The precise order considered

[5] The definition of $\mathcal{Z}_\mathcal{F}$, as well as most subsequent definitions, do not only depend on $\mathcal{F}$ but also on the alphabet Σ. Although this dependence is important, we do not explicitly include it in the notations in order to lighten them, as most of the times the alphabet will be clear from the context.

will be irrelevant for our purposes. From now on, we will assume that all Zielonka trees are ordered, without explicitly mentioning it.

For a leaf $l \in \mathsf{Leaves}(\mathcal{Z}_\mathcal{F})$ and a letter $a \in \Sigma$ we define $\mathsf{Supp}(l, a) = n$ to be the deepest ancestor of l (maximal for $\preceq$) such that $a \in \nu(n)$.

Example II.29.

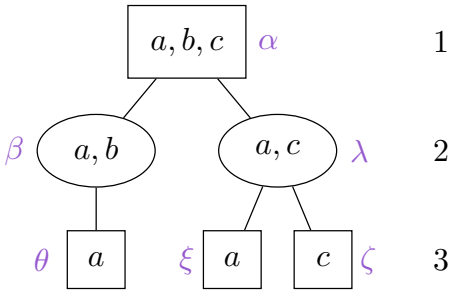
We will use the Muller language associated to the following family of subsets as a running example throughout the chapter. Let $\Sigma = \{a, b, c\}$ and let $\mathcal{F}$ be:

$$\mathcal{F} = \{\{a, b\}, \{a, c\}, \{b\}\}.$$

In Figure 12 we show the Zielonka tree of $\mathcal{F}$. We use Greek letters (in pink) to name the nodes of the tree. Integers appearing on the right of the tree will be used in next section.

We have that $\mathsf{Supp}(\xi, c) = \lambda$ and $\mathsf{Supp}(\xi, b) = \alpha$. Also, $\mathsf{Jump}(\xi, \lambda) = \zeta$ is the leaf reached by going from ξ to λ, then changing to the next branch (in a cyclic way) and re-descend by taking the leftmost path. Similarly, $\mathsf{Jump}(\xi, \alpha) = \theta$.

The subtree rooted at λ contains the nodes $\{\lambda, \xi, \zeta\}$. We note that this is the Zielonka tree of $\mathcal{F}|_{\{a,c\}} = \{\{a, c\}\}$ (over the alphabet $\{a, c\}$).



✦ **Figure 12.** Zielonka tree $\mathcal{Z}_\mathcal{F}$ for $\mathcal{F} = \{\{a, b\}, \{a, c\}, \{b\}\}$.

The Zielonka DAG

Another structure that will be useful, mostly when considering decision problems and the representation of Muller conditions (c.f. Section II.5), is the Zielonka DAG.

The *Zielonka DAG* of a family $\mathcal{F} \subseteq 2^\Sigma_+$ is just the labelled directed acyclic graph obtained by merging the nodes of $\mathcal{Z}_\mathcal{F}$ that share a common label. Formally, if is the labelled DAG $\mathcal{Z}\text{-}\mathsf{DAG}_\mathcal{F} = (N', \preceq', \nu')$ where $N' = \{C \subseteq \Sigma \mid \exists n$ node of the Zielonka tree such that $C = \nu(n)\}$ and $\nu'(C) = C$. The relation $\preceq'$ is inherited from the ancestor relation of the tree: $C \preceq' D$ if there are n_C, n_D nodes of the Zielonka tree such that $\nu(n_C) = C$, $\nu(n_D) = D$ and $n_C \preceq n_D$. In particular, $C \preceq' D$ implies $D \subseteq C$ (but the converse does not hold in general). We will denote the labelling just by ν. We remark that $\mathcal{Z}\text{-}\mathsf{DAG}_\mathcal{F}$ inherits the partition of nodes into round and square ones. Moreover, children of a round node of the Zielonka DAG are square nodes and vice-versa. We also note that Remark II.27 and Lemma II.28 hold similarly replacing $\mathcal{Z}_\mathcal{F}$ by $\mathcal{Z}\text{-}\mathsf{DAG}_\mathcal{F}$ in their statement.

We refer to Proposition B.32 for a comparison between the size of the Zielonka tree and the Zielonka DAG, and Figure 52 for an example (page 286).

3.2 A minimal deterministic parity automaton

We present next the Zielonka-tree-parity-automaton, a minimal deterministic parity automaton for a Muller language $\mathsf{Muller}(\mathcal{F})$ built from the Zielonka tree $\mathcal{Z}_\mathcal{F}$. Our construction will furthermore let us determine the parity index of the language $\mathsf{Muller}(\mathcal{F})$ from its Zielonka tree.

3.2.1 The Zielonka-tree-parity-automaton

We associate a non-negative integer to each level of a Zielonka tree $\mathcal{Z}_\mathcal{F} = (N, \preceq, \nu)$. We let $p_\mathcal{Z} : N \to \mathbb{N}$ be the function defined as:

- if $\Sigma \in \mathcal{F}$, $p_\mathcal{Z}(n) = \mathsf{Depth}(n)$,
- if $\Sigma \notin \mathcal{F}$, $p_\mathcal{Z}(n) = \mathsf{Depth}(n) + 1$.

We let $\min_\mathcal{F}$ (resp. $\max_\mathcal{F}$) be the minimum (resp. maximum) value taken by the function $p_\mathcal{Z}$.

✦ Remark II.30. *A node n in the Zielonka tree $\mathcal{Z}_\mathcal{F}$ verifies that $p_\mathcal{Z}(n)$ is even if and only if $\nu(n) \in \mathcal{F}$. If $\Sigma \in \mathcal{F}$, $\min_\mathcal{F} = 0$ and $\max_\mathcal{F}$ equals the height of the Zielonka tree minus one. If $\Sigma \notin \mathcal{F}$, $\min_\mathcal{F} = 1$ and $\max_\mathcal{F}$ equals the height of the Zielonka tree.*

Example II.31.

The Muller language from Example II.29 satisfies $\Sigma \notin \mathcal{F}$. The values taken by the function $p_\mathcal{Z}$ are represented at the right of the Zielonka tree in Figure 12. We have $p_\mathcal{Z}(\alpha) = 1$, $p_\mathcal{Z}(\beta) = p_\mathcal{Z}(\lambda) = 2$ and $p_\mathcal{Z}(\theta) = p_\mathcal{Z}(\xi) = p_\mathcal{Z}(\zeta) = 3$, so $\min_\mathcal{F} = 1$ and $\max_\mathcal{F} = 3$.

Definition II.32 (Zielonka-tree-parity-automaton)**.**

Given a family of non-empty subsets $\mathcal{F} \subseteq 2^\Sigma_+$, we define the *ZT-parity-automaton* $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}} = (Q, \Sigma, q_{\mathsf{init}}, [\min_\mathcal{F}, \max_\mathcal{F}], \delta, \mathsf{parity})$ as the deterministic parity automaton given by:

- $Q = \mathsf{Leaves}(\mathcal{Z}_\mathcal{F})$,
- q_{init} is the leftmost leaf of $\mathcal{Z}_\mathcal{F}$,[6]
- The transition reading $a \in \Sigma$ from $q \in Q$ goes to $\mathsf{Jump}(q, \mathsf{Supp}(q, a))$ and produces $p_\mathcal{Z}(\mathsf{Supp}(q, a))$ as output, that is,

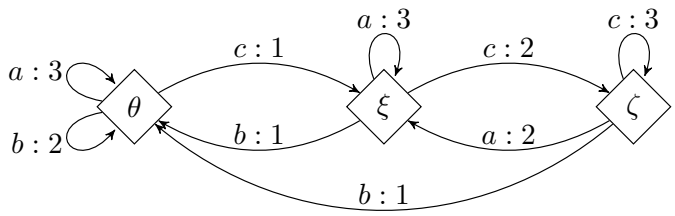
$$\delta(q, a) = (\mathsf{Jump}(q, \mathsf{Supp}(q, a)), p_\mathcal{Z}(\mathsf{Supp}(q, a)))\,.$$

Intuitively, the transitions of the automaton are determined as follows: if we are in a leaf l and we read a colour a, then we move up in the branch of l until we reach a node n that contains the letter a in its label. Then we pick the child of n just to the right of the branch that we took before (in a cyclic way) and we move to the leftmost leaf below it. The priority produced as output is $p_\mathcal{Z}(n)$, determined by the depth of n.

[6] Any state can be chosen as initial state (see Lemma I.9).

Example II.33.

In Figure 13 we show the ZT-parity-automaton $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$ of the family of subsets from Example II.29.



✦ **Figure 13.** ZT-parity-automaton recognising the Muller language associated to $\mathcal{F} = \{\{a, b\}, \{a, c\}, \{b\}\}$.

Correctness of the Zielonka-tree-parity-automaton

Proposition II.34 (Correctness)**.**

Let $\mathcal{F} \subseteq 2^\Sigma_+$ be a family of non-empty subsets. Then,

$$\mathcal{L}(\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}) = \mathsf{Muller}_\Sigma(\mathcal{F}).$$

That is, a word $w \in \Sigma^\omega$ is accepted by $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$ if and only if $\mathsf{Inf}(w) \in \mathcal{F}$.

The following useful lemma follows directly from the definition of Supp and Jump.

✦ **Lemma II.35.** *Let q be a leaf of $\mathcal{Z}_\mathcal{F}$ and let n be a node above q. Then, $\mathsf{Supp}(q, a)$ is a descendant of n if and only if $a \in \nu(n)$, and in this case, $\mathsf{Jump}(q, \mathsf{Supp}(q, a))$ is a descendant of n too.*

Proof of Proposition II.34. Let $w = w_0w_1w_2 \cdots \in \Sigma^\omega$ be an infinite word. For $i > 0$, let q_i be the leaf of $\mathcal{Z}_\mathcal{F}$ reached after the (only) run over $w_0w_1 \dots w_{i-1}$ in $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$. For $i \geq 0$ let $n_i = \mathsf{Supp}(q_i, w_i)$ be the "intermediate node" used to determine the next state and the output priority of each transition, and let $c_i = p_\mathcal{Z}(n_i) = \mathsf{col}(q_i, w_i) \in [\min_\mathcal{F}, \max_\mathcal{F}]$ be that output priority (the output of the run over w being therefore $c_0c_1c_2 \cdots \in \mathbb{N}^\omega$). Let q_∞ be a node appearing infinitely often in the sequence $q_0q_1q_2 \dots$, and let n_w be the deepest ancestor of q_∞ such that $\mathsf{Inf}(w) \subseteq \nu(n_w)$.

✧ Claim II.34.1. *There is $K \in \mathbb{N}$ such that for all $i \geq K$, $q_i \succeq n_w$ and $\mathsf{Supp}(q_i, w_i) \succeq n_w$. In particular, $c_i \geq p_\mathcal{Z}(n_w)$ for $i \geq K$.*

Proof. Let $K \in \mathbb{N}$ be a position such that $w_i \in \mathsf{Inf}(w)$ for all $i \geq K$ and $q_K = q_\infty$. The claim follows from Lemma II.35 and induction. ◁

✧ Claim II.34.2. *Let $n_{w,1}, \dots, n_{w,s}$ be an enumeration of $\mathsf{Children}_{\mathcal{Z}_\mathcal{F}}(n_w)$ from left to right. It is verified that:*

1. *$\mathsf{Supp}(q_i, w_i) = n_w$ infinitely often. In particular, $c_i = p_\mathcal{Z}(n_w)$ for infinitely many i's.*
2. *There is no $n_{w,k} \in \mathsf{Children}(n_w)$ such that $\mathsf{Inf}(w) \subseteq \nu(n_{w,k})$.*

Proof. We first remark that for all $n_{w,k}$ there are arbitrarily large positions i such that q_i is not below $n_{w,k}$. Suppose by contradiction that this is not the case. Then, for all i sufficiently large we have that $\mathsf{Supp}(q_i, w_i) \succeq n_{w,k}$, and by Lemma II.35, $\mathsf{Inf}(w) \subseteq \nu(n_{w,k})$. In particular, q_∞ is below $n_{w,k}$, contradicting the fact that n_w is the deepest ancestor of q_∞ containing $\mathsf{Inf}(w)$.

Let K be like in the Claim II.34.1. We show that if $i \geq K$ and $q_i \succeq n_{w,k}$, then there is $j > i$ such that $w_j \notin \nu(n_{w,k})$, $\mathsf{Supp}(q_j, w_j) = n_w$ and $q_{j+1} \succeq n_{w,k+1}$ (by an abuse of notation we let $s + 1 = 1$). It suffices to consider the least $j \geq i$ such that $\mathsf{Supp}(q_j, w_j) \not\succeq n_{w,k}$ (which exists by the previous remark). Since $\mathsf{Inf}(w) \subseteq \nu(n_w)$ we have that $\mathsf{Supp}(q_j, w_j) = n_w$, so $w_j \notin \nu(n_{w,j})$ (by Lemma II.35) and by definition of the transitions of $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$, q_{j+1} will be a leaf below $n_{w,k+1}$.

The fact that $q_{j+1} \succeq n_{w,k+1}$ implies that for any child $n_{w,k'}$, infinitely many states q_i will be below $n_{w,k'}$ (we go around the children in a round-robin fashion). Therefore, for any k, there are arbitrarily large j such that $w_j \notin \nu(n_{w,j})$ and $\mathsf{Supp}(q_j, w_j) = n_w$, implying both items in the claim. ◁

Combining both claims, we obtain that the minimum of the priorities that are produced as output infinitely often is $p_\mathcal{Z}(n_w)$. By Remark II.30, $p_\mathcal{Z}(n_w)$ is even if and only if n_w is a round node (if $\nu(n_w) \in \mathcal{F}$). It remains to show that $\mathsf{Inf}(w) \in \mathcal{F}$ if and only if $\nu(n_w) \in \mathcal{F}$, which holds by the second item in Claim II.34.2 and Remark II.27. ◀

3.2.2 Optimality of the Zielonka-tree-parity-automaton

We now state and prove the main results of this section: the optimality of the ZT-parity-automaton in both number of states (Theorem II.2) and number of priorities of the acceptance condition (Theorem II.1). The minimality of the ZT-parity-automaton comes in two version. A weaker one states its minimality only amongst deterministic automata (Theorem II.3), and a stronger one states its minimality amongst all history-deterministic automata (Theorem II.2). Although the weaker version is implied by the stronger one, we find it instructive to provide a proof for this easier case. The proof of the stronger statement is one of the most technical parts of the thesis, but the argument used in its proof is just a careful refinement of the ideas appearing in the weaker version.

Statement of the results

Theorem II.1 (Optimality of the parity condition)**.**

The parity index of a Muller language $\mathsf{Muller}_\Sigma(\mathcal{F})$ is $[\min_\mathcal{F}, \max_\mathcal{F}]$. That is, the ZT-parity-automaton of $\mathsf{Muller}_\Sigma(\mathcal{F})$ uses the optimal number of priorities to recognise this language.

Theorem II.2 (Minimality of the ZT-parity-automaton)**.**

Let $\mathcal{A}$ be a history-deterministic parity automaton recognising a Muller language $\mathsf{Muller}_\Sigma(\mathcal{F})$. Then, $|\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}| \leq |\mathcal{A}|$.

Corollary II.36.

For every Muller language L, a minimal deterministic parity automaton recognising L has the same size than a minimal HD parity automaton recognising L.

We remark that, nonetheless, there are non-trivial HD parity automata recognising Muller languages. The automaton provided in Example I.7 (Figure 4) is an HD coBüchi automaton recognising a Muller language that cannot be made deterministic just by removing transitions (it is not determinisable by pruning). We note that the (deterministic) ZT-parity-automaton for this Muller language has only 2 states.

Proposition II.37.

There exists an HD parity automaton recognising a Muller language that is not determinisable by pruning.

Optimality of the parity condition

Proof of Theorem II.1. Let $L = \mathsf{Muller}_\Sigma(\mathcal{F})$. The ZT-parity-automaton of L is a parity automaton recognising L using priorities in $[\min_\mathcal{F}, \max_\mathcal{F}]$, therefore, the parity index of L is at most $[\min_\mathcal{F}, \max_\mathcal{F}]$.

To prove that the parity index is not less than $[\min_\mathcal{F}, \max_\mathcal{F}]$, we use the Flower Lemma I.21. The language L is trivially recognised by a deterministic Muller automaton $\mathcal{A}_L$ with just one state q, transitions $q \xrightarrow{a:a} q$ for each $a \in \Sigma$, and acceptance condition given by L itself. Let $n_1 \preceq n_2 \preceq \ldots \preceq n_d$ be a branch of maximal length of $\mathcal{Z}_\mathcal{F}$ (that must verify $d = \max_\mathcal{F} - \min_\mathcal{F}$, and that the root n_1 is a round node if and only if $\min_\mathcal{F}$ is even). If we let ℓ_i be the cycle in $\mathcal{A}_L$ containing exactly the transitions corresponding to letters in $\nu(n_i)$, we obtain that $\ell_1 \supsetneq \ell_2 \supsetneq \cdots \supsetneq \ell_d$ is a d-flower over q, which is positive if and only if n_1 is a round node. Lemma I.21 allows us to conclude. ◀

Minimality of the ZT-parity-automaton with respect to deterministic automata

The proof of Theorem II.2 is quite technical, and we include it in Appendix B.2. Here, we prove a weaker result, namely, that the ZT-parity-automaton is minimal amongst *deterministic* parity automata recognising a Muller language. We find that a proof of this weaker result might be enlightening for the reader, and it will allow us to introduce the main ingredients used in the proof of the stronger optimality result in a simpler setting.

Theorem II.3 (Minimality with respect to deterministic automata)**.**

Let $\mathcal{A}$ be a DPA recognising a Muller language $\mathsf{Muller}_\Sigma(\mathcal{F})$. Then, $|\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}| \leq |\mathcal{A}|$.

We recall that, by Remark I.8 and Lemma I.9, we can suppose that all the states of automata recognising Muller languages are accessible, and that any of them can be chosen to be initial. When considering subautomata of these automata, we will sometimes not mention their initial state.

We recall that for a subset $X \subseteq \Sigma$, an X-FSCC of an automaton $\mathcal{A}$ over Σ is an X-closed final SCC in the graph obtained by taking the restriction of the underlying graph of $\mathcal{A}$ to the edges labelled by letters in X. We can find an X-FSCC in an automaton $\mathcal{A}$ recognising $\mathsf{Muller}_\Sigma(\mathcal{F})$ for any $X \subseteq \Sigma$ (Lemma I.4).

✦ **Lemma II.38.** *Let $\mathcal{A}$ be a DMA recognising a Muller language $\mathsf{Muller}_\Sigma(\mathcal{F})$, let $X \subseteq \Sigma$ and let $\mathcal{S}_X$ be an accessible X-FSCC of $\mathcal{A}$. Then, the automaton induced by $\mathcal{S}_X$ is a deterministic automaton recognising $\mathsf{Muller}_X(\mathcal{F}|_X) = \{w \in X^\omega \mid \mathsf{Inf}(w) \in \mathcal{F}\}$.*

Proof. Let $\mathcal{A} = (Q, \Sigma, q_0, \Gamma, \delta, W)$ (where W is a Muller language). Let q_S be the state in $\mathcal{S}_X$ chosen to be initial, and let $u_0 \in \Sigma^*$ be a finite word such that the run over u_0 from q_0 ends in q_S. By prefix-independence of Muller languages, a word $w \in X^\omega$ belongs to $\mathsf{Muller}_\Sigma(\mathcal{F})$ if and only if $u_0 w \in \mathsf{Muller}_\Sigma(\mathcal{F})$, and therefore, $\mathcal{A}$ accepts w if and only if it accepts $u_0 w$. Since the run in $\mathcal{A}$ over $u_0 w$ and the run in $\mathcal{S}_X$ over w have a suffix in common, and by prefix-independence of W, we have that $w \in \mathcal{L}(\mathcal{S}_X)$ if and only if $u_0 w \in \mathcal{L}(\mathcal{A})$ if and only if $\mathsf{Inf}(w) \in \mathcal{F}$. ◀

Next lemma states that, in a parity automaton, the union of two accepting cycles must be accepting, and similarly for rejecting cycles. In Section II.6, we will see that this property is actually a characterisation of parity transition systems (Proposition II.111).

✦ **Lemma II.39.** *Let $\mathcal{A}$ be a parity automaton. Let $\ell_1, \ell_2 \in \mathcal{C}\mathit{ycles}(\mathcal{A})$ be two cycles with some state in common. If ℓ_1 and ℓ_2 are both accepting (resp. rejecting), then $\ell_1 \cup \ell_2$ is also accepting (resp. rejecting).*

Proof. Let $\mathsf{col} \colon \Delta \to \mathbb{N}$ be the colouring function of $\mathcal{A}$. The cycles ℓ_1 and ℓ_2 are accepting if and only if $d_i = \min \mathsf{col}(\ell_i)$ is even, for $i = 1, 2$. In this case, $\min \mathsf{col}(\ell_1 \cup \ell_2) = \min\{d_1, d_2\}$ is even. The proof is symmetric if ℓ_1 and ℓ_2 are rejecting. ◀

By a small abuse of notation, we will say that two SCC $\mathcal{S}_1$ and $\mathcal{S}_2$ are disjoint, and write $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$, if their sets of states are disjoint.

✦ **Lemma II.40.** *Let $\mathcal{F} \subseteq 2^\Sigma_+$ be a family of subsets with Zielonka tree $\mathcal{Z}_\mathcal{F} = (N, \preceq, \nu)$, and let $\mathcal{A}$ be a DPA recognising $\mathsf{Muller}_\Sigma(\mathcal{F})$. Let $n \in N$ be a node of the Zielonka tree of $\mathcal{F}$, and let $n_1, n_2 \in \mathsf{Children}_{\mathcal{Z}_\mathcal{F}}(n)$ be two different children of n. If $\mathcal{S}_1$ and $\mathcal{S}_2$ are two accessible $\nu(n_1)$-FSCC and $\nu(n_2)$-FSCC in $\mathcal{A}$, respectively, then $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$.*

Proof. Without loss of generality, we can suppose that all states in $\mathcal{A}$ are accessible, and since the language that $\mathcal{A}$ recognises is prefix-independent, we can also suppose that $\mathcal{A}$ is complete. Let $\mathsf{let} \colon \Delta \to \Sigma$ be the labelling of the transitions of $\mathcal{A}$ with input letters. Let $\mathcal{S}_i$ be a $\nu(n_i)$-FSCC in $\mathcal{A}$, for $i = 1, 2$, and let ℓ_i be its set of edges, which form a cycle satisfying $\mathsf{let}(\ell_i) = \nu(n_i)$. Suppose by contradiction that $\mathcal{S}_1 \cap \mathcal{S}_2 \neq \emptyset$. Then ℓ_1 and ℓ_2 have some state in common, and their union is also a cycle satisfying $\mathsf{let}(\ell_1 \cup \ell_2) = \nu(n_1) \cup \nu(n_2)$. By Lemma II.39, we must have

$$\ell_1 \text{ accepting} \iff \ell_1 \cup \ell_2 \text{ accepting},$$

contradicting the fact that $\nu(n_1) \in \mathcal{F}$ if and only if $\nu(n_1) \cup \nu(n_2) \notin \mathcal{F}$ (Remark II.27). ◀

Proof of Theorem II.3. We proceed by induction in the height of $\mathcal{Z}_\mathcal{F}$. For height 1, the result is trivial, since $|\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}| = 1$. Let $\mathcal{A}$ be a DPA recognising $\mathsf{Muller}_\Sigma(\mathcal{F})$. Let n_0 be the root of $\mathcal{Z}_\mathcal{F}$ and $n_1, n_2, \dots, n_k$ be an enumeration of the children of n_0 in $\mathcal{Z}_\mathcal{F}$. By Lemma I.4, for each $i \in \{1, \dots, k\}$, $\mathcal{A}$ contains some accessible $\nu(n_i)$-FSCC $\mathcal{S}_i$, and by Lemma II.40 these must be pairwise disjoint. By Lemma II.38, each $\mathcal{S}_i$ induces a deterministic subautomaton recognising $\mathsf{Muller}_{\nu(n_i)}(\mathcal{F}|_{\nu(n_i)})$. Let $\mathcal{Z}_i$ by the subtree of $\mathcal{Z}_\mathcal{F}$

rooted at n_i, which we recall that is the Zielonka tree for $\mathcal{F}|_{\nu(n_i)}$. By induction hypothesis, it must be the case that $|\mathsf{Leaves}(\mathcal{Z}_i)| \leq |\mathcal{S}_i|$, so we can conclude:

$$|\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}| = |\mathsf{Leaves}(\mathcal{Z}_\mathcal{F})| = \sum_{i=1}^{k} |\mathsf{Leaves}(\mathcal{Z}_i)| \leq \sum_{i=1}^{k} |\mathcal{S}_i| \leq |\mathcal{A}|. \qquad \blacktriangleleft$$

Some comments about the general case (Theorem II.2). The steps we follow to prove Theorem II.2 are the same as the ones we have just presented. We perform an induction over the height of the Zielonka tree, and at each step we try to find disjoint subautomata recognising the language associated to the different children of a given node. However, some major technical difficulties appear. Due to the asymmetry of semantics of non-determinism (we accept a word if *there exists* an accepting run over it), we need to distinguish two cases: the one in which the root of the Zielonka tree is a square node, and the one in which it is round. The first case can be handle without too much trouble, as we can generalise Lemmas II.39 and II.40 by using resolvers given by finite memory structures: we take the product of $\mathcal{A}$ by such a memory, we find disjoint X-FSCC in there, and we prove that the image of these X-FSCC in $\mathcal{A}$ are also disjoint. However, in the case in which the root is a round node, this naive approach is not sufficient, and a fine analysis of the strategies used by the resolver is required. All the details can be found in Appendix B.2.

3.3 A minimal history-deterministic Rabin automaton

In this section, we present the construction of a history-deterministic Rabin automaton $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ for a Muller language $\mathsf{Muller}(\mathcal{F})$ using the Zielonka tree $\mathcal{Z}_\mathcal{F}$, and prove its minimality (Theorem II.4). The automaton $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ can be seen as a quotient of the ZT-parity-automaton; that is, $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ is obtained by merging some of the states of $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$. Thus, we replace the complexity in the number of states by complexity in the acceptance condition. The size of the automaton $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ is a well-studied parameter of Zielonka trees: its round-branching width, $\mathsf{rbw}(\mathcal{Z}_\mathcal{F})$. This parameter was introduced by Dziembowski, Jurdziński and Walukiewicz [DJW97] (under the name of *memory of* $\mathcal{Z}_\mathcal{F}$) and shown to coincide with the memory required by Eve to win in games using $\mathsf{Muller}(\mathcal{F})$ as an acceptance condition (see Proposition IV.1). Although in this chapter we are not concerned with the memory of objectives, we rely on their result to show the optimality of our construction.

We note that this construction is asymmetric, in the sense that we show it for Rabin automata, but not for Streett automata (their dual notion). The reason why we cannot dualize the construction is due to the semantics of non-deterministic automata. However, we could use the same idea to obtain a minimal *universal* history-deterministic Streett automaton (we refer to [BL19] for the definition of universal HD automata).

3.3.1 The Zielonka-tree-HD-Rabin-automaton

Definition II.41 ([DJW97]).

Let T be a tree with nodes partitioned into round and square nodes, and let $T_1, \ldots, T_k$ be the subtrees of T rooted at the children of the root of T. We define inductively the

round-branching width of T, denoted $\mathsf{rbw}(T)$ as:

$$\mathsf{rbw}(T) = \begin{cases} 1 & \text{if } T \text{ has exactly one node,} \\ \max\{\mathsf{rbw}(T_1), \dots, \mathsf{rbw}(T_k)\} & \text{if the root is square,} \\ \sum_{i=1}^{k} \mathsf{rbw}(T_i) & \text{if the root is round.} \end{cases}$$

Next lemma directly follows from the definition of $\mathsf{rbw}(T)$.

Lemma II.42.

Let $T = (N = N_{\bigcirc} \sqcup N_{\square}, \preceq)$ be a tree with nodes partitioned into round and square nodes. There exists a mapping $\eta\colon \mathsf{Leaves}(T) \to \{1, 2, \dots, \mathsf{rbw}(T)\}$ satisfying:

> If $n \in N$ is a round node with children $n_1 \neq n_2$, for any pair of leaves l_1 and l_2 below n_1 and n_2, respectively, $\eta(l_1) \neq \eta(l_2)$. $(\star)$

Example II.43.

Let $\mathcal{F} = \{\{a, b\}, \{a, c\}, \{b\}\}$ be the family of subsets considered in Example II.29. The round-branching width of $\mathcal{Z}_{\mathcal{F}}$ is $\mathsf{rbw}(\mathcal{Z}_{\mathcal{F}}) = 2$. A labelling $\eta\colon \mathsf{Leaves}(\mathcal{Z}_{\mathcal{F}}) \to \{1, 2\}$ satisfying Property $(\star)$ is given by $\eta(\theta) = \eta(\xi) = 1$ and $\eta(\zeta) = 2$. This labelling is represented in the Zielonka tree $\mathcal{Z}_{\mathcal{F}}$ on the left of Figure 14.

We give the construction of a minimal HD Rabin automaton for a Muller language.

Definition II.44 (Zielonka-tree-HD-Rabin-automaton)**.**

Let $\mathcal{F} \subseteq 2^{\Sigma}_{+}$, $\mathcal{Z}_{\mathcal{F}} = (N = N_{\bigcirc} \sqcup N_{\square}, \preceq)$ its Zielonka tree and $\eta\colon \mathsf{Leaves}(\mathcal{Z}_{\mathcal{F}}) \to \{1, 2, \dots, \mathsf{rbw}(\mathcal{Z}_{\mathcal{F}})\}$ a mapping satisfying Property $(\star)$. We define the *ZT-HD-Rabin-automaton* $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_{\mathcal{F}}} = (Q, \Sigma, I, \Gamma, \Delta, \mathsf{Rabin}_{\Gamma}(R))$ as a (non-deterministic) automaton using a Rabin acceptance condition, where:

- $Q = \{1, 2, \dots, \mathsf{rbw}(\mathcal{Z}_{\mathcal{F}})\}$,
- $I = Q$,[7]
- $\Gamma = N$ (the colours of the acceptance condition are the nodes of the Zielonka tree),
- $\delta(q, a) = \{\big(\mathsf{Jump}(l, \mathsf{Supp}(l, a)), \mathsf{Supp}(l, a)\big) \mid l \in \mathsf{Leaves}(\mathcal{Z}_{\mathcal{F}}) \text{ such that } \eta(l) = q\}$,
- $R = \{(\mathfrak{g}_n, \mathfrak{r}_n)\}_{n \in N_{\bigcirc}}$, where $\mathfrak{g}_n$ and $\mathfrak{r}_n$ are defined as follow: Let n be a round node and n' be any node of $\mathcal{Z}_{\mathcal{F}}$,

$$\begin{cases} n' \in \mathfrak{g}_n & \text{if } n' = n, \\ n' \in \mathfrak{r}_n & \text{if } n' \neq n \text{ and } n \text{ is not an ancestor of } n'. \end{cases}$$

✦ Remark II.45. *Although we will usually say that* $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ *is the ZT-HD-Rabin-automaton of* $\mathcal{F}$*, the structure of this automaton is not unique, it depends on two choices: the order over the nodes of the Zielonka tree and the mapping* η*.*

The intuition behind this definition is the following. The automaton $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ has $\mathsf{rbw}(\mathcal{Z}_\mathcal{F})$ states, and each of them can be associated to a subset of leaves of $\mathcal{Z}_\mathcal{F}$ by $\eta^{-1}(q)$. The mapping η is such that the lowest common ancestor of two leaves in $\eta^{-1}(q)$ is a square node. As for the ZT-parity-automaton, for each leaf of $l \in \mathsf{Leaves}(\mathcal{Z}_\mathcal{F})$ and letter $a \in \Sigma$, we identify the deepest ancestor $n = \mathsf{Supp}(l, a)$ containing a in its label, and, using the Jump function, pick a leaf l' below the next child of n. We add a transition $q \xrightarrow{a:n} q'$ if there are leaves $l \in \eta^{-1}(q)$ and $l' \in \eta^{-1}(q')$ giving such a path. This way, we can identify a run in the automaton $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ with a promenade through the nodes of the Zielonka tree in which shifts between leaves with the same η-image are allowed. If during this promenade a unique minimal node (for $\preceq$) is visited infinitely often, it is not difficult to see that the sequence of input colours belongs to $\mathcal{F}$ if and only if the label of this minimal node belongs to $\mathcal{F}$ (it is a round node). The Rabin condition over the set of nodes of the Zielonka tree is devised so that it accepts exactly these sequences of nodes (see Lemma II.49 below).

Another way of presenting the automaton $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ is as a quotient of the deterministic parity automaton $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$. Indeed, the graph structure and the labelling by input letters of $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ is obtained by merging the states of $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$ (which are the leaves of $\mathcal{Z}_\mathcal{F}$) with the same η-image, and keeping all the transitions between them. However, a parity acceptance condition over this smaller structure is no longer sufficient to accept $\mathsf{Muller}(\mathcal{F})$.

Example II.46.

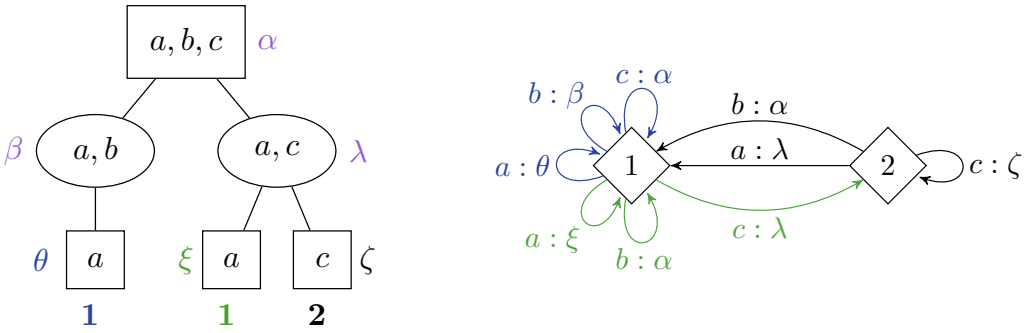
The ZT-HD-Rabin-automaton $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ of the family $\mathcal{F} = \{\{a, b\}, \{a, c\}, \{b\}\}$ from Example II.29 is shown on the right of Figure 14. The Zielonka tree $\mathcal{Z}_\mathcal{F}$ appears on the left of the figure, and the labelling $\eta\colon \mathsf{Leaves}(\mathcal{Z}_\mathcal{F}) \to \{1, 2\}$ is represented by the numbers below its branches. The Rabin condition of this automaton is given by two Rabin pairs (corresponding to the round nodes of the Zielonka tree):

$$\mathfrak{g}_\beta = \{\beta\}, \quad \mathfrak{r}_\beta = \{\alpha, \lambda, \xi, \zeta\},$$
$$\mathfrak{g}_\lambda = \{\lambda\}, \quad \mathfrak{r}_\lambda = \{\alpha, \beta, \theta\}.$$

We note that the automaton $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ is obtained by merging the states θ and ξ from the ZT-parity-automaton $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$ appearing in Figure 13, and replacing the output colours by suitable nodes from the Zielonka tree.

✦ Remark II.47. *We observe that the automaton from Figure 14 presents duplicated edges, in the sense that there are two transitions* $q \xrightarrow{a:x} q'$ *and* $q \xrightarrow{a:y} q'$ *between the same pair of states and reading the same input letter. We can always avoid this and remove duplicated edges from any automaton. We provide a proof in Appendix B.5 (Proposition B.46). For the language from the previous example, an equivalent automaton is proposed in Figure 54*

[7] Any non-empty subset of Q can be chosen as the set of initial states.



✦ **Figure 14.** On the left, the Zielonka tree of $\mathcal{F} = \{\{a, b\}, \{a, c\}, \{b\}\}$. On the right, the ZT-HD-Rabin-automaton $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$. Blue transitions correspond to those coming from leaf θ, and green ones to those originating from leaf ξ.

Correctness of the Zielonka-tree-HD-Rabin-automaton

Proposition II.48 (Correctness)**.**

Let $\mathcal{F} \subseteq 2^\Sigma_+$ be a family of non-empty subsets. Then,

$$\mathcal{L}(\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}) = \mathsf{Muller}_\Sigma(\mathcal{F}).$$

Moreover, the automaton $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ is history-deterministic.

✦ **Lemma II.49.** *Let $u = n_0 n_1 n_2 \cdots \in N^\omega$ be an infinite sequence of nodes of the Zielonka tree. The word u belongs to $\mathsf{Rabin}_N(R)$, for $R = \{(\mathfrak{g}_n, \mathfrak{r}_n)\}_{n \in N_\bigcirc}$ the Rabin condition of $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$, if and only if there is a unique minimal node for the ancestor relation in $\mathsf{Inf}(u)$ and this minimal node is round (recall that the root is the minimal element for $\preceq$).*

Proof. Suppose that there is a unique minimal node in $\mathsf{Inf}(u)$, called n, and that n is round. We claim that u is accepted by the Rabin pair $(\mathfrak{g}_n, \mathfrak{r}_n)$. It is clear that $\mathsf{Inf}(u) \cap \mathfrak{g}_n \neq \emptyset$, because $n \in \mathfrak{g}_n$. It suffices to show that $\mathsf{Inf}(u) \cap \mathfrak{r}_n = \emptyset$: By minimality, any other node $n' \in \mathsf{Inf}(u)$ is a descendant of n (equivalently, n is an ancestor of n'), so $n' \notin \mathfrak{r}_n$.

Conversely, suppose that $u \in \mathsf{Rabin}_N(R)$. Then, there is some round node $n \in N_\bigcirc$ such that $\mathsf{Inf}(u) \cap \mathfrak{g}_n \neq \emptyset$ and $\mathsf{Inf}(u) \cap \mathfrak{r}_n = \emptyset$. Since $\mathfrak{g}_n = \{n\}$, we deduce that $n \in \mathsf{Inf}(u)$. Moreover, as $\mathsf{Inf}(u) \cap \mathfrak{r}_n = \emptyset$, all nodes in $\mathsf{Inf}(u)$ are descendants of n. We conclude that n is the unique minimal node in $\mathsf{Inf}(u)$, and it is round. ◀

✦ **Lemma II.50.** *There exists a morphism of automata $\varphi \colon \mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}} \to \mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$.*

Proof. We define the morphism φ as follows:

- $\varphi_V(l) = \eta(l)$, for $l \in \mathsf{Leaves}(\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}})$,
- for a transition $e = l \xrightarrow{a:c} l'$ in $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$, we let $\varphi_E(e) = (\eta(l), a, \mathsf{Supp}(l, a), l')$.

It is clear that φ is a weak morphism. We prove that it preserve the acceptance of runs. Let $\rho = l_0 \xrightarrow{w_0} l_1 \xrightarrow{w_1} l_2 \xrightarrow{w_2} \cdots \in \mathcal{R}un(\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}})$ be an infinite run in $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$ (the only run over $w_0 w_1 w_2 \cdots \in \Sigma^\omega$), and let $n_i = \mathsf{Supp}(l_i, w_i)$. By definition of the morphism, the output of the run $\rho' = \varphi_{\mathcal{R}uns}(\rho)$ in $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ is $\mathsf{col}'(\rho') = n_0 n_1 n_2 \cdots \in N^\omega$. In the proof of

Proposition II.34, we proved (Claims II.34.1 and II.34.2) that there exists a unique node n_w appearing infinitely often in $\mathsf{col}'(\rho')$. Moreover, we proved that ρ is accepting in $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$ if and only if n_w is round. Lemma II.49 allows us to conclude that $\varphi_{\mathcal{R}uns}(\rho)$ is accepting in $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ if and only if ρ is accepting in $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$. ◀

Proof of Proposition II.48. $\mathcal{L}(\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}) \subseteq \mathsf{Muller}_\Sigma(\mathcal{F})$**:** Let $w \in \mathcal{L}(\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}})$ and let $u \in N^\omega$ be the sequence of nodes produced as output of an accepting run over w in $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$. By Lemma II.49, there is a unique minimal node n for $\preceq$ appearing infinitely often in u and moreover n is round. Let $n_1, \dots, n_k$ be an enumeration of the children of n (from left to right), with labels $\nu(n_i) \subseteq \Sigma$ (we remark that $\nu(n_i) \notin \mathcal{F}$, for $1 \leq i \leq k$). We will prove that $\mathsf{Inf}(w) \subseteq \nu(n)$ and $\mathsf{Inf}(w) \not\subseteq \nu(n_i)$ for $1 \leq i \leq k$. By definition of the Zielonka tree, as n is round, this implies that $\mathsf{Inf}(w) \in \mathcal{F}$.

Since eventually all nodes produced as output are descendants of n (by minimality), $\mathsf{Inf}(w)$ must be contained in $\nu(n)$ (by definition of the transitions of $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$).

We suppose, towards a contradiction, that $\mathsf{Inf}(w) \subseteq \nu(n_j)$ for some $1 \leq j \leq k$. Let $Q_i = \{\eta(l) : l \text{ is a leaf below } n_i\}$ be the set of states corresponding to leaves under n_i, for $1 \leq i \leq k$. We can suppose that the leaves corresponding to transitions of an accepting run over w are all below n, and therefore, transitions of such a run only visit states in $\bigcup_{i=1}^k Q_i$. Indeed, eventually this is going to be the case, because if some of the leaves l, l' corresponding to a transition (q, a, n', q') are not below n, then n' would not be a descendant of n (since n' is the least common ancestor of l and l'). Also, by Property $(\star)$, we have $Q_i \cap Q_j = \emptyset$, for all $i \neq j$. By definition of the transitions of $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$, if $a \in \Sigma$ is a letter in $\nu(n)$ but not in $\nu(n_i)$, all transitions from some state in Q_i reading the letter a go to Q_{i+1}, for $1 \leq i \leq k-1$ (and to Q_1 if $i = k$). Also, if $a \in \nu(n_i)$, transitions from states in Q_i reading a stay in Q_i. We deduce that a run over w will eventually only visit states in Q_j, for some j such that $\mathsf{Inf}(w) \subseteq \nu(n_j)$. However, the only transitions from Q_j that would produce n as output are those corresponding to a letter $a \notin \nu(n_j)$, so the node n is not produced infinitely often, a contradiction.

$\mathsf{Muller}_\Sigma(\mathcal{F}) \subseteq \mathcal{L}(\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}})$ **and history-determinism**: The existence of a morphism $\varphi\colon \mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}} \to \mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ (Lemma II.50) and the correctness of $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$ (Proposition II.34) imply that $\mathcal{L}(\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}) = \mathcal{L}(\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}) = \mathsf{Muller}(\mathcal{F})$. Indeed, if ρ is an accepting run over $w \in \Sigma^\omega$ in $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$, then $\varphi_{\mathcal{R}uns}(\rho)$ is an accepting run over w in $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$. We can moreover use $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$ and φ to define a sound resolver (r_0, r) for $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$: we let $r_0 = \varphi(q_0)$ be the image of the initial state of $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$. If $\rho_R \in \mathcal{R}un^{\mathsf{fin}}(\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}})$ is the image under $\varphi_{\mathcal{R}uns}$ of some finite run $\rho_P \in \mathcal{R}un^{\mathsf{fin}}(\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}})$, we let $\mathsf{r}(\rho_R, a) = \varphi(e)$, where e is the only a-labelled transition from $\mathsf{target}(\rho_P)$. We define r arbitrarily in other case. This way, for every $w \in \Sigma^\omega$, the run induced by r over w is the image of a run over w in $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$, which must be accepting if $w \in \mathsf{Muller}(\mathcal{F})$. ◀

3.3.2 Optimality of the Zielonka-tree-HD-Rabin-automaton

We state now the optimality of $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$.

Theorem II.4 (Optimality of the ZT-HD-Rabin-automaton)**.**

Let $\mathcal{A}$ be a history-deterministic Rabin automaton accepting a Muller language $\mathsf{Muller}_\Sigma(\mathcal{F})$. Then, $|\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}| \leq |\mathcal{A}|$.

Theorem II.4 follows easily combining the three following facts:

- $|\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}| = \mathsf{rbw}(\mathcal{Z}_\mathcal{F})$.
- There are games in which Eve cannot play optimally with less than $\mathsf{rbw}(\mathcal{Z}_\mathcal{F})$ memory states ([DJW97, Theorem 14], see also Proposition IV.1).
- Rabin objectives are half-positional ([Kla94, Lemma 9] and [Zie98, Corollary 14], see also Proposition IV.9).

Indeed, assume that $\mathcal{A}$ is an HD Rabin automaton recognising $\mathsf{Muller}(\mathcal{F})$. Since history-deterministic automata are good-for-games (Proposition II.5), we can take the composition of $\mathcal{A}$ with a given $\mathsf{Muller}(\mathcal{F})$-game, obtaining a Rabin game in which Eve can play optimally using a positional strategy. This proves that $\mathcal{A}$ can be used as a memory structure for $\mathsf{Muller}(\mathcal{F})$-games, so it cannot be smaller than $\mathsf{rbw}(\mathcal{Z}_\mathcal{F})$. Formal details are provided in Lemma IV.13 (in Chapter IV, dedicated to memory for games).

We remark that Proposition II.48 provides an alternative proof for the fact that the memory requirements of $\mathsf{Muller}(\mathcal{F})$ are at most $\mathsf{rbw}(\mathcal{Z}_\mathcal{F})$, result first established in [DJW97, Theorem 6].

Number of Rabin pairs

For the ZT-parity-automaton, we have shown the optimality of its acceptance condition (Theorem II.1). A natural question is whether the acceptance condition of the ZT-HD-Rabin-automaton is optimal, that is, if $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ uses a minimal number of Rabin pairs, or if we can minimise it. We remark that to make sense of this question, we need to fix the automaton structure, or at least, the number of states. If we allow to increase the number of states, an automaton minimising the number of Rabin pairs is $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$ with its optimal parity condition.

Therefore, we focus on the following question: What is the minimal number of Rabin pairs required to define an equivalent Rabin condition on top of $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$?

We know that the Rabin condition we have defined for $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$ is not optimal, and it admits a fairly simple improvement: we could define it in a similar way using the nodes of the Zielonka DAG instead of the nodes of the Zielonka tree (see the proof of Proposition B.33). However, this option is not optimal in general neither. We believe (see Conjecture II.2) that the problem of minimising the number of Rabin pairs for a fixed automaton structure is NP-complete.

4 The alternating cycle decomposition: An optimal approach to Muller transition systems

In Section II.3, we have provided minimal parity and Rabin automata for Muller languages, using the Zielonka tree. We can use these automata to transform Muller transition systems, by applying the product construction. However, this approach overlooks the structure of the transition system, meaning it does not take into account the relevant interplay between the underlying graph and the acceptance condition.

In this section, we present our main contributions: optimal transformations of Muller transition systems into parity and Rabin ones. The key novelty is that they precisely capture the way the transition system interacts with the acceptance condition. This is achieved by generalising Zielonka trees from Muller languages to Muller transition systems; we define the alternating cycle decomposition (ACD), consisting in a collection of Zielonka-tree-like structures subsuming all the structural information of the transition

system necessary to determine whether a run is accepting or not. More precisely, the ACD is a succinct representation of the *alternating chains of loops* of a Muller automaton, in the sense of Wagner [Wag79]. The alternating chains of loops of a DMA are known to determine the parity index of the language it recognises [Wag79], and, as we will show, they also capture the essential information to define optimal transformations of automata.

We start with the definition of the alternating cycle decomposition in Section II.4.1. In Section II.4.2, we describe the ACD-parity-transform, turning a DMA $\mathcal{A}$ into an equivalent DPA $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{A})$. Formally, the validity of this transformation is witnessed by a locally bijective morphism $\varphi\colon \mathsf{ACD}_{\mathsf{parity}}(\mathcal{A}) \to \mathcal{A}$ (Proposition II.68). In Section II.4.3, we describe the ACD-HD-Rabin-transform that turns a DMA $\mathcal{A}$ into an equivalent history-deterministic Rabin automaton $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{A})$. The validity of the transformation is witnessed by an HD mapping $\varphi\colon \mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{A}) \to \mathcal{A}$ (Proposition II.75). These constructions grant strong optimality guarantees. The automaton $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{A})$ (resp. $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{A})$) has a minimal number of states amongst parity (resp. Rabin) automata admitting an HD mapping to $\mathcal{A}$ (Theorems II.6 and II.7). We note that this implies minimality amongst automata admitting a locally bijective morphism to $\mathcal{A}$. Moreover, the acceptance condition of $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{A})$ uses an optimal number of priorities (Theorem II.5). The optimality of these constructions is shown in Section II.4.4. We are able to prove the optimality of both constructions at the same time, by reducing the problem to an application of the minimality of the ZT-parity-automaton and the ZT-HD-Rabin-automaton.

In all this section, we let $\mathcal{TS} = (G_{\mathcal{TS}}, \mathrm{Acc}_{\mathcal{TS}})$ be a Muller transition system with underlying graph $G_{\mathcal{TS}} = (V, E, \mathsf{source}, \mathsf{target}, I)$ and using a Muller acceptance condition $\mathrm{Acc}_{\mathcal{TS}} = (\mathsf{col}, \Gamma, \mathsf{Muller}_\Gamma(\mathcal{F}))$.

4.1 The alternating cycle decomposition

Tree of alternating cycles and the ACD

Definition II.51.

Let $\ell_0 \in \mathit{Cycles}(\mathcal{TS})$ be a cycle. We define the *tree of alternating subcycles* of ℓ_0, denoted $\mathsf{AltTree}(\ell_0) = (N, \preceq, \nu\colon N \to \mathit{Cycles}(\mathcal{TS}))$ as a $\mathit{Cycles}(\mathcal{TS})$-labelled tree with nodes partitioned into *round nodes* and *square nodes*, $N = N_\bigcirc \sqcup N_\square$, such that:

- The root is labelled ℓ_0.
- If a node is labelled $\ell \in \mathit{Cycles}(\mathcal{TS})$, and ℓ is an accepting cycle ($\mathsf{col}(\ell) \in \mathcal{F}$), then it is a round node, and its children are labelled exactly with the maximal subcycles $\ell' \subseteq \ell$ such that ℓ' is rejecting ($\mathsf{col}(\ell') \notin \mathcal{F}$).
- If a node is labelled $\ell \in \mathit{Cycles}(\mathcal{TS})$, and ℓ is a rejecting cycle ($\mathsf{col}(\ell) \notin \mathcal{F}$), then it is a square node, and its children are labelled exactly with the maximal subcycles $\ell' \subseteq \ell$ such that ℓ' is accepting ($\mathsf{col}(\ell') \in \mathcal{F}$).

For a $\mathit{Cycles}(\mathcal{TS})$-labelled tree $T = (N, \preceq, \nu\colon N \to \mathit{Cycles}(\mathcal{TS}))$ and $n \in N$, we let $\nu_{\mathsf{States}}(n) = \mathsf{States}(\nu(n))$ be the set of states of the cycle labelling n.

✦ Remark II.52. *Let n be a node of $\mathsf{AltTree}(\ell_0)$ and let n_1 be a child of it. If ℓ' is a cycle such that $\nu(n_1) \subsetneq \ell' \subseteq \nu(n)$, then $\nu(n_1)$ is accepting $\iff$ ℓ' is rejecting $\iff$*

$\nu(n)$ *is rejecting.*

Definition II.53 (Alternating cycle decomposition)**.**

Let $\mathcal{TS}$ be a transition system, and let $\ell_1, \ell_2, \dots, \ell_k$ be an enumeration of its maximal cycles (that is, the edges sets of its SCCs). We define the *alternating cycle decomposition* of $\mathcal{TS}$ to be the forest $\mathcal{ACD}_{\mathcal{TS}} = \{\mathsf{AltTree}(\ell_1), \dots, \mathsf{AltTree}(\ell_k)\}$.

We let N_{ℓ_i} be the set of nodes of $\mathsf{AltTree}(\ell_i)$, and n_{ℓ_i} its root. We will suppose that $N_{\ell_i} \cap N_{\ell_j} = \emptyset$ if $i \neq j$.

We define the *set of nodes of* $\mathcal{ACD}_{\mathcal{TS}}$ to be $\mathsf{Nodes}(\mathcal{ACD}_{\mathcal{TS}}) = \bigcup_{i=1}^{k} N_{\ell_i}$, and we let $\mathsf{Nodes}_{\bigcirc}(\mathcal{ACD}_{\mathcal{TS}})$ (resp. $\mathsf{Nodes}_{\square}(\mathcal{ACD}_{\mathcal{TS}})$) be the subset of round (resp. square) nodes. As for Zielonka trees, we equip the trees of $\mathcal{ACD}_{\mathcal{TS}}$ with an arbitrary order making them ordered trees. From now on, we will suppose that all trees of alternating subcycles are ordered, without explicitly mentioning it.

Local subtrees

We remark that for a recurrent vertex v of $\mathcal{TS}$, there is one and only one tree $\mathsf{AltTree}(\ell_i)$ in $\mathcal{ACD}_{\mathcal{TS}}$ such that $v \in \nu_{\mathsf{States}}(n_{\ell_i})$. On the other hand, transient vertices do not appear in the trees of $\mathcal{ACD}_{\mathcal{TS}}$.

If v is a recurrent vertex of $\mathcal{TS}$, we define the *local subtree at* v, noted $\mathcal{T}_v$, as the subtree of $\mathsf{AltTree}(\ell_i)$ containing the nodes $N_v = \{n \in N_{\ell_i} \mid v \in \nu_{\mathsf{States}}(n)\}$. If v is a transient vertex, we define $\mathcal{T}_v$ to be a tree with a single node.

For v recurrent, as N_v is a subset of the nodes of $\mathsf{AltTree}(\ell_i)$, the tree $\mathcal{T}_v$ inherits the order from $\mathsf{AltTree}(\ell_i)$, as well as its partition into round and square nodes, $N_v = N_{v,\bigcirc} \sqcup N_{v,\square}$. Also, it inherits the labelling given by the mapping ν, whose restriction to $\mathcal{T}_v$ has an image in $\mathcal{C}ycles_v(\mathcal{TS})$.

✦ Remark II.54. *Let $v \in \nu_{\mathsf{States}}(n_{\ell_i})$. If $n \in N_v$ and n' is an ancestor of n in $\mathsf{AltTree}(\ell_i)$, then $n' \in N_v$. In particular, $\mathcal{T}_v$ is indeed a subtree of $\mathsf{AltTree}(\ell_i)$. Also, we note that the root of $\mathcal{T}_v$ is n_{ℓ_i}.*

For a node $n \in N_{\ell_i}$ and an edge $e \in \ell_i$ we define $\mathsf{Supp}(n, e) = n'$ to be the deepest ancestor of n such that $e \in \nu(n')$. We remark that if $e = v \to v'$, then $\mathsf{Supp}(n, e)$ is a node in both $\mathcal{T}_v$ and $\mathcal{T}_{v'}$.

Example II.55.

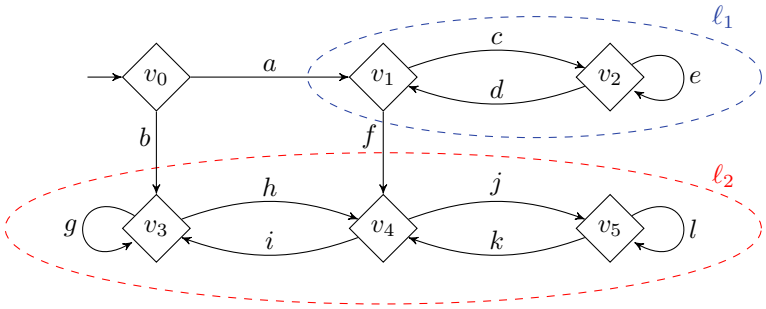
We will use the transition system $\mathcal{TS}$ from Figure 15 as a running example. We have named the edges of $\mathcal{TS}$ with letters from a to l, that are also used as the output colours of the acceptance condition. The acceptance set of $\mathcal{TS}$ is the Muller language associated to:

$$\mathcal{F} = \{\{c, d, e\}, \{e\}, \{g, h, i\}, \{l\}, \{h, i, j, k\}, \{j, k\}\}.$$

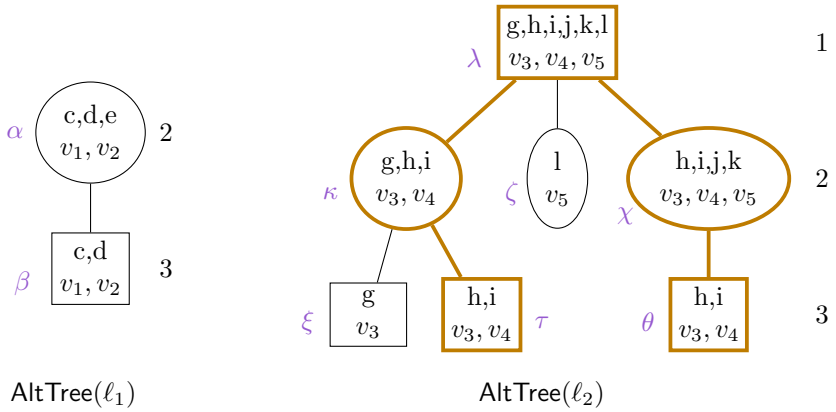
The initial vertex of $\mathcal{TS}$, v_0, is its only transient vertex, all the others vertices are recurrent. $\mathcal{TS}$ has 2 strongly connected components, corresponding to cycles ℓ_1 and ℓ_2.

The alternating cycle decomposition of $\mathcal{TS}$ is shown in Figure 16. It consists of two trees, $\mathsf{AltTree}(\ell_1)$ and $\mathsf{AltTree}(\ell_2)$. We use Greek letters (in pink) to name the nodes of the tree. Inside each node we indicate both its label $\nu(n)$ and the set of states of it. For example, $\nu(\kappa) = \{g, h, i\}$ and $\nu_{\mathsf{States}}(\kappa) = \{v_3, v_4\}$. We have that $\mathsf{Supp}(\tau, g) = \kappa$ and

$\mathsf{Supp}(\tau, j) = \lambda$. We highlight in bold orange the local subtree at v_4, $\mathcal{T}_{v_4}$. The tree $\mathcal{T}_{v_0}$, consisting in a single node, does not appear in the figure. The numbering on the right of the trees will be used in next section.



✦ **Figure 15.** Transition system $\mathcal{TS}$ using a Muller acceptance condition given by $\mathcal{F} = \{\{c,d,e\},\{e\},\{g,h,i\},\{l\},\{h,i,j,k\},\{j,k\}\}$. The two maximal cycles, ℓ_1 and ℓ_2, are encircled by blue and red dashed lines, respectively.



✦ **Figure 16.** Alternating cycle decomposition of $\mathcal{TS}$. In bold orange, the local subtree at v_4, $\mathcal{T}_{v_4}$.

✦ Remark II.56. *Let $\mathcal{TS}$ be a Muller TS using as acceptance set $W = \mathsf{Muller}_\Gamma(\mathcal{F})$, and let $\overline{\mathcal{TS}}$ be the TS obtained by replacing W with $\overline{W} = \Gamma^\omega \setminus W$ (which is a Muller language). Then, the ACD of $\overline{\mathcal{TS}}$ coincides with that of $\mathcal{TS}$, with the only difference that the partition into round and square nodes is inverted: $\mathsf{Nodes}_\bigcirc(\mathcal{ACD}_{\overline{\mathcal{TS}}}) = \mathsf{Nodes}_\square(\mathcal{ACD}_{\mathcal{TS}})$ and $\mathsf{Nodes}_\square(\mathcal{ACD}_{\overline{\mathcal{TS}}}) = \mathsf{Nodes}_\bigcirc(\mathcal{ACD}_{\mathcal{TS}})$.*

We note that if $\mathcal{A}$ is a DMA recognising $L \subseteq \Sigma^\omega$, the automaton $\overline{\mathcal{A}}$ is a DMA recognising $\Sigma^\omega \setminus L$.

✦ Remark II.57. *The Zielonka tree can be seen as the special case of the alternating cycle decomposition for transition systems with just one state. Indeed, a Muller language $\mathsf{Muller}(\mathcal{F})\Sigma$ can be trivially recognised by a DMA $\mathcal{A}$ with a single state q and self loops $q \xrightarrow{a:a} q$. The ACD of this automaton is exactly the Zielonka tree of $\mathcal{F}$.*

Local Muller languages

For a recurrent state v of $\mathcal{TS}$, we define the *local Muller language of $\mathcal{TS}$ at v* as the Muller language defined over the alphabet $\Gamma_v = \mathcal{C}ycles_v(\mathcal{TS})$ associated to:

$$\mathsf{LocalMuller}_{\mathcal{TS}}(v) = \{\mathcal{C} \subseteq \mathcal{C}ycles_v(\mathcal{TS}) \mid \bigcup_{\ell \in \mathcal{C}} \ell \text{ is an accepting cycle}\}.$$

We note that $\mathsf{LocalMuller}_{\mathcal{TS}}(v)$ is completely determined by singletons ($\mathcal{C} \in \mathsf{LocalMuller}_{\mathcal{TS}}(v)$ if and only if $\{\bigcup_{\ell\in\mathcal{C}} \ell\} \in \mathsf{LocalMuller}_{\mathcal{TS}}(v)$). For simplicity, and by a slight abuse of notation, we will work as if $\mathsf{LocalMuller}_{\mathcal{TS}}(v) \subseteq \mathcal{C}ycles_v(\mathcal{TS})$. To lighten notations, we will just write $\mathsf{LocalMuller}_{\mathcal{TS}}(v)$ to denote $\mathsf{Muller}(\mathsf{LocalMuller}_{\mathcal{TS}}(v))$ whenever no confusion arises.

The following lemma directly follows from the definition of $\mathcal{T}_v$ and that of Zielonka tree. It provides insight in the structure of the trees $\mathcal{T}_v$, and it will be a key ingredient in the proof of the optimality of the transformations based on the alternating cycle decomposition.

Lemma II.58.

Let $\mathcal{TS}$ be a Muller transition system and let v be a recurrent vertex of it. The tree $\mathcal{T}_v$ is the Zielonka tree of the family $\mathsf{LocalMuller}_{\mathcal{TS}}(v)$.[8]

The ACD-DAG

In the same way as we obtained the Zielonka DAG from the Zielonka tree, we define a DAG obtained by merging the nodes of the ACD sharing the same label. This structure will be useful for computational considerations (c.f. Section II.5).

Let $\mathcal{TS}$ be a Muller transition system. The *DAG of alternating subcycles* of a cycle ℓ_0, denoted $\mathsf{AltDAG}(\ell_0)$ is the $\mathcal{C}ycles(\mathcal{TS})$-labelled DAG obtained by merging the nodes of $\mathsf{AltTree}(\ell_0)$ with a same label.

The *ACD-DAG* of a Muller transition system $\mathcal{TS}$ is $\mathcal{ACD}\text{-}\mathsf{DAG}_{\mathcal{TS}} = \{\mathsf{AltDAG}(\ell_1), \ldots, \mathsf{AltDAG}(\ell_k)\}$, where $\ell_1, \ldots, \ell_k$ is an enumeration of the maximal cycles of $\mathcal{TS}$ (that is, of its SCCs).

For v a recurrent vertex of $\mathcal{TS}$, we define the *local subDAG at v*, noted $\mathcal{T}_v\text{-}\mathsf{DAG}$, as the DAG obtained by merging the nodes of $\mathcal{T}_v$ with a same label. We note that if ℓ_i is the maximal cycle containing v, $\mathcal{T}_v\text{-}\mathsf{DAG}$ coincides with the subDAG of $\mathsf{AltDAG}(\ell_i)$ consisting of the nodes labelled with cycles containing v.

4.2 An optimal transformation to parity transition systems

We now define the ACD-parity-transform, an optimal transformation turning a Muller TS into a parity TS while preserving determinism. In order to obtain the optimality in the number of priorities, we need to pay attention to the parity of the minimal priority used in different SCCs. To incorporate this parameter in the transformation, we define positive and negative ACDs.

[8] Formally, the labelling ν of $\mathcal{T}_v$ goes to $\mathcal{C}ycles_v(\mathcal{TS})$, and not to $2_+^{\mathcal{C}ycles_v(\mathcal{TS})}$, as required by the definition of the Zielonka tree. To obtain a proper Zielonka tree with a labelling of nodes $\nu' \colon N_v \to 2_+^{\mathcal{C}ycles_v(\mathcal{TS})}$, we would have to define $\nu'(n) = \{\ell' \in \mathcal{C}ycles_v(\mathcal{TS}) \mid \ell' \subseteq \nu(n)\}$.

Let $\mathcal{TS}$ be a Muller transition system and let $\mathcal{ACD}_{\mathcal{TS}} = \{\mathsf{AltTree}(\ell_1), \dots, \mathsf{AltTree}(\ell_k)\}$ be its alternating cycle decomposition.

We say that a tree $\mathsf{AltTree}(\ell_i) \in \mathcal{ACD}_{\mathcal{TS}}$ is *positive* if ℓ_i is an accepting cycle, and that it is *negative* otherwise. We say that the alternating cycle decomposition of $\mathcal{TS}$ is *positive* if all the trees of maximal height of $\mathcal{ACD}_{\mathcal{TS}}$ are positive, that it is *negative* if all trees of maximal height are negative, and that it is *equidistant* if there are positive and negative trees of maximal height.

As for the Zielonka tree, we associate a non-negative integer to each level of the trees of $\mathcal{ACD}_{\mathcal{TS}}$ via a function $p_{\mathcal{ACD}}(n)\colon \mathsf{Nodes}(\mathcal{ACD}_{\mathcal{TS}}) \to \mathbb{N}$. Let ℓ_i be a maximal cycle of $\mathcal{TS}$ and $n \in N_{\ell_i}$.

- If $\mathcal{ACD}_{\mathcal{TS}}$ is positive or equidistant:
  - $p_{\mathcal{ACD}}(n) = \mathsf{Depth}(n)$, if ℓ_i is accepting,
  - $p_{\mathcal{ACD}}(n) = \mathsf{Depth}(n) + 1$, if ℓ_i is rejecting.
- If $\mathcal{ACD}_{\mathcal{TS}}$ is negative:
  - $p_{\mathcal{ACD}}(n) = \mathsf{Depth}(n) + 2$, if ℓ_i is accepting,
  - $p_{\mathcal{ACD}}(n) = \mathsf{Depth}(n) + 1$, if ℓ_i is rejecting.

We let $\min_{\mathcal{TS}}$ (resp. $\max_{\mathcal{TS}}$) be the minimum (resp. maximum) value taken by the function $p_{\mathcal{ACD}}$.

✦ Remark II.59. *A node n in $\mathsf{AltTree}(\ell_i)$ verifies that $p_{\mathcal{ACD}}(n)$ is even if and only if $\nu(n)$ is an accepting cycle (that is, if n is a round node).*

✦ Remark II.60. *It is satisfied:*

- $\min_{\mathcal{TS}} = 0$ *if $\mathcal{ACD}_{\mathcal{TS}}$ is positive or equidistant,*
- $\min_{\mathcal{TS}} = 1$ *if $\mathcal{ACD}_{\mathcal{TS}}$ is negative.*

Example II.61.

In the previous Example II.55, $\mathsf{AltTree}(\ell_1)$ is a positive tree and $\mathsf{AltTree}(\ell_2)$ is negative. As $\mathsf{AltTree}(\ell_2)$ is the tree of maximal height, $\mathcal{ACD}_{\mathcal{TS}}$ is negative. The function $p_{\mathcal{ACD}}$ is represented in Figure 16 by the integers on the right of each tree. It takes values 2 and 3 over $\mathsf{AltTree}(\ell_1)$ ($p_{\mathcal{ACD}}(\alpha) = 2$ and $p_{\mathcal{ACD}}(\beta) = 3$), because $\mathcal{ACD}_{\mathcal{TS}}$ is negative. In this example, $\min_{\mathcal{TS}} = 1$ and $\max_{\mathcal{TS}} = 3$. We note that if we had associated integers 0 and 1 to the levels of $\mathsf{AltTree}(\ell_1)$, we would have used 4 integers in total, instead of just 3 of them.

Definition II.62 (ACD-parity-transform)**.**

Let $\mathcal{TS}$ be a Muller TS with $\mathcal{ACD}_{\mathcal{TS}} = \{\mathsf{AltTree}(\ell_1), \dots, \mathsf{AltTree}(\ell_k)\}$. We define the *ACD-parity-transform* of $\mathcal{TS}$ be the parity TS $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS}) = (G', \mathrm{Acc}')$, with $G' = (V', E', \mathsf{source}', \mathsf{target}', I')$, and $\mathrm{Acc}' = (\mathsf{col}', [\min_{\mathcal{TS}}, \max_{\mathcal{TS}}], \mathsf{parity})$ defined as follows.

Vertices. The set of vertices is

$$V' = \bigcup_{v \in V} (\{v\} \times \mathsf{Leaves}(\mathcal{T}_v))\,.$$

Initial vertices. $I' = \{(v_0, n) \mid v_0 \in I \text{ and } n \text{ is the leftmost leaf in } \mathcal{T}_{v_0}\}$.

Edges and output colours. For each $(v,n) \in V'$ and each edge $e = v \to v' \in \mathsf{Out}(v)$ in $\mathcal{TS}$ we define an edge $e_n = (v,n) \xrightarrow{\mathsf{col}'(e_n)} (v',n')$. Formally,

$$E' = \bigcup_{e \in E} \left(\{e\} \times \mathsf{Leaves}(\mathcal{T}_{\mathsf{source}(e)}) \right).$$

If v and v' are not in the same SCC, we let n' be the leftmost leaf in $\mathcal{T}_{v'}$ and $\mathsf{col}'(e_n) = \min_{\mathcal{TS}}$.[9] If v and v' belong to the same SCC, we let:

- $n' = \mathsf{Jump}_{\mathcal{T}_{v'}}(n, \mathsf{Supp}(n,e))$,
- $\mathsf{col}'(e_n) = p_{\mathcal{ACD}}(\mathsf{Supp}(n,e))$.

Labellings. If $\mathcal{TS}$ is a labelled transition system, with labels $l_V \colon V \to L_V$ and $l_E \colon E \to L_E$, we label $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ by $l'_{V'}(v,n) = l_V(v)$ and $l'_{E'}(e_n) = l_E(e)$.

Intuitively, a run in the transition system $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ follows a run in $\mathcal{TS}$ with some extra information, updated in the same manner as it was the case with the ZT-parity-automaton. To define transitions in $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$, we move simultaneously in $\mathcal{TS}$ and in $\mathcal{ACD}_{\mathcal{TS}}$. When we take a transition e in $\mathcal{TS}$ that goes from v to v' , while being in a node n in the ACD , we climb the branch of n searching the lowest node $\tilde{n}$ with e and v' in its label ($\tilde{n} = \mathsf{Supp}(n,e)$). We produce as output the priority corresponding to the level reached. If no such node exists in the current tree (this occurs if we change of SCC), we jump to the root of the tree containing v'. After having reached the node $\tilde{n}$, we move to its next child in the tree $\mathcal{T}_{v'}$ (in a cyclic way), and we pick the leftmost leaf under it.

Example II.63.

We show in Figure 17 the ACD-parity-transform $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ of the transition system $\mathcal{TS}$ from Figure 15 (Example II.55). For each vertex v in $\mathcal{TS}$, we make as many copies as leaves of the tree $\mathcal{T}_v$. We note that, as v_0 is transient, the tree $\mathcal{T}_{v_0}$ consists of a single node (by definition), that we name ι. Transitions are of the form (e,l), for e a transition from $\mathcal{TS}$ and l a leaf of some local subtree; these are denoted e_l in the figure for the sake of space convenience. These labels simply indicate the names of the edges, they should not be interpreted as input letters ($\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ is not an automaton).

We observe that there is a locally bijective morphism of transition systems φ from $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ to $\mathcal{TS}$ given by $\varphi_V(v,l) = v$ and $\varphi_E(e_l) = e$.

Another example can be found in Figure 18.

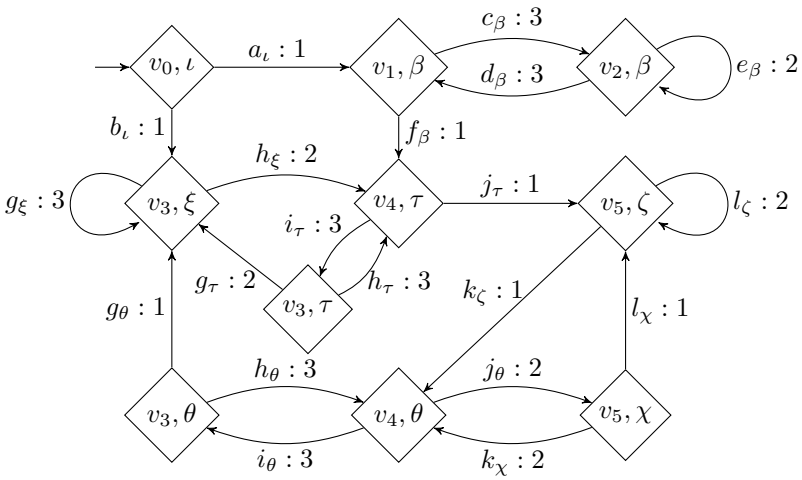
✦ Remark II.64. *The size of the ACD-parity-transformation of $\mathcal{TS}$ is:*

$$|\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})| = \sum_{v \in V} |\mathsf{Leaves}(\mathcal{T}_v)| = \sum_{v \in V_{\mathrm{rec}}} |\mathsf{Leaves}(\mathcal{T}_v)| + |V_{\mathrm{trans}}|,$$

where V_{rec} and V_{trans} are the sets of recurrent and transient vertices of $\mathcal{TS}$, respectively.

✦ Remark II.65. *We remark that if $\mathcal{TS} = (G_{\mathcal{TS}}, \mathrm{Acc}_{\mathcal{TS}})$ is already a parity TS, then the underlying graphs of $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ and $\mathcal{TS}$ are isomorphic. In fact, by Proposition II.68,*

[9] The priorities associated to transitions changing of SCC are *almost* arbitrary (we could even leave them uncoloured). We define them to be the minimal priority used so that the obtained transition system is normalised in the sense of Section II.7.



✦ **Figure 17.** ACD-parity-transform $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ of the transition system $\mathcal{TS}$ from Figure 15.

$\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ *and* $\mathcal{TS}$ *will also be isomorphic as transition systems. In this case, the construction of* $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ *boils down to the application of the procedure described by Carton and Maceiras [CM99].*

✦ Remark II.66. *The ACD-parity-transform is oblivious to the labelling* col *of the acceptance condition of* $\mathcal{TS}$*; the only information taken into account to define the graph of* $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ *and its output colours is the structure of the trees of* $\mathcal{ACD}_{\mathcal{TS}}$*. That is, the definition of this transformation is independent of the actual representation of the acceptance condition of* $\mathcal{TS}$ *(whether it is Emerson-Lei, Muller, Rabin...), and we only use that any such representation induces a mapping* $f\colon \mathcal{C}\mathit{ycles}(\mathcal{TS}) \to \{\text{Accept}, \text{Reject}\}$.

✦ Remark II.67. *The ZT-parity-automaton can be seen as a special case of the ACD-parity-transform, as* $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$ *coincides with the DPA* $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{A})$*, where* $\mathcal{A}$ *is the DMA with a single state recognising* $\mathsf{Muller}(\mathcal{F})$ *(see Remark II.57).*

Correctness of the ACD-parity-transform

Proposition II.68 (Correctness of the ACD-parity-transform)**.**

Let $\mathcal{TS}$ be a (labelled) Muller TS and let $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ be its ACD-parity-transform. There is a locally bijective morphism of (labelled) transition systems $\varphi\colon \mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS}) \to \mathcal{TS}$.

The following lemma, analogous to Lemma II.35 from Section II.3.2, follows from the definition of the ACD-parity-transform.

✦ **Lemma II.69.** *Let* n *be a node of* $\mathsf{AltTree}(\ell_i)$*, let* $\tilde{n}$ *be an ancestor of* n *and let* $e = v \to v'$ *be an edge in* ℓ_i*. Then,* $\mathsf{Supp}(n, e)$ *is a descendant of* $\tilde{n}$ *if and only if* $e \in \nu(\tilde{n})$*, and in this case, if* $e_n = (v, n) \to (v', n')$ *is an edge of* $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$*, then* n' *is a descendant of* $\tilde{n}$ *too.*

Proof of Proposition II.68. We consider the mapping $\varphi = (\varphi_V, \varphi_E)$ naturally defined by $\varphi_V(v, n) = v$ and $\varphi_E(e_n) = e$. It is immediate to check that φ is a weak morphism of transition systems (it preserves initial states and transitions). Also, it is easy to see that it

is locally bijective: for each initial state $v_0 \in I$, there is exactly one node in I' of the form (v_0, n): the node where n is the leftmost leaf of $\mathcal{T}_v$; and for each vertex (v, n) and edge $e \in \mathsf{Out}(v)$ of $\mathcal{TS}$, we have define exactly one edge outgoing from (v, n) corresponding to e.

We prove that φ preserves the acceptance of runs, following the proof scheme from Proposition II.34. We can suppose w.l.o.g. (see Remarks I.3 and II.66) that the set of output colours used by $\mathcal{TS}$ is its set of edges E. Let $\rho \in \mathcal{R}un(\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS}))$ be an infinite run in $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$. Eventually, ρ will remain in one SCC, and $\mathsf{Inf}(\rho)$ will form a cycle that is accepting if and only if ρ is an accepting run. We will suppose that all the edges in ρ appear infinitely often and belong to this cycle (we can do it by using a similar argument as the one presented in the proof of Proposition II.34), and we let:

$$\rho = (v_0, n_0) \xrightarrow{x_0} (v_1, n_1) \xrightarrow{x_1} (v_2, n_2) \xrightarrow{x_3} \dots.$$

The projection of ρ under φ is:

$$\varphi_{\mathcal{R}uns}(\rho) = v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} v_2 \xrightarrow{e_3} \dots.$$

We note that the edges $\{e_0, e_1, \dots\}$ form a cycle in $\mathcal{TS}$, that we will call ℓ_ρ. In particular, ℓ_ρ is contained in some maximal cycle $\ell_{\max}$, and all the nodes n_i belong to the same tree $\mathsf{AltTree}(\ell_{\max})$ of the ACD. Our objective is to show that ℓ_ρ is an accepting cycle in $\mathcal{TS}$ if and only if $\min\{x_0, x_1, x_2, \dots\}$ is even. We let $\tilde{n}_i = \mathsf{Supp}(n_i, e_i)$ be the node of $\mathcal{ACD}_{\mathcal{TS}}$ determining the i$^{\text{th}}$ transition of ρ, so we have that $x_i = p_{\mathcal{ACD}}(\tilde{n}_i)$. Finally, let n_ρ be the deepest ancestor of n_0 such that $\ell_\rho \subseteq \nu(n_\rho)$.

✧ Claim II.68.1. *For all $i \geq 0$, $n_i \succeq n_\rho$ and $\tilde{n}_i \succeq n_\rho$ (that is, all nodes appearing in ρ are below n_ρ). In particular, $x_i \geq p_{\mathcal{ACD}}(n_\rho)$.*

Proof. The claim follows from Lemma II.69 and induction. ◁

✧ Claim II.68.2. *Let $n_{\rho,1}, \dots, n_{\rho,s}$ be an enumeration of $\mathsf{Children}_{\mathsf{AltTree}(\ell_{\max})}(n_\rho)$. It is verified that:*

1. *$\mathsf{Supp}(n_i, e_i) = n_\rho$ infinitely often. In particular, $x_i = p_{\mathcal{ACD}}(n_\rho)$ for infinitely many i's.*
2. *There is no $n_{\rho,k} \in \mathsf{Children}(n_\rho)$ such that $\ell_\rho \subseteq \nu(n_{\rho,k})$.*

Proof. The proof is identical to that of Claim II.34.2, from Proposition II.34. ◁

We conclude that $\min\{x_0, x_1, x_2, \dots\} = p_{\mathcal{ACD}}(n_\rho)$, which is even if and only if ℓ_ρ is an accepting cycle, by Remarks II.52 and II.59. ◀

✦ Remark II.70. *We can give an alternative interpretation of the previous proof. Given a run ρ in $\mathcal{TS}$ and a vertex v appearing infinitely often in ρ, we can decompose the run into:*

$$\overset{\rho_0}{\rightsquigarrow} v \overset{\rho_1}{\rightsquigarrow} v \overset{\rho_2}{\rightsquigarrow} v \overset{\rho_3}{\rightsquigarrow} v \overset{\rho_4}{\rightsquigarrow} \dots,$$

where the finite runs ρ_i are cycles over v, for $i > 0$. Therefore, the sequence of these cycles can be processed by the ZT-parity-automaton corresponding to the local Muller condition $\mathsf{LocalMuller}_{\mathcal{TS}}(v)$. By Lemma II.58 and the correctness of the ZT-parity-automaton, the minimal priority produced by a run over this sequence of cycles in $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_{\mathsf{LocalMuller}_{\mathcal{TS}}(v)}}$ coincides with the minimal priority produced by the run $\varphi^{-1}_{\mathcal{R}uns}(\rho)$ in the ACD-parity-transform $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ (disregarding the initial path ρ_0). This priority is exactly the one corresponding to the deepest node in $\mathcal{T}_v$ above the leftmost leaf containing $\mathsf{Inf}(\rho)$.

The locally bijective morphism given by Proposition II.68 witnesses that $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ has the same semantic properties as $\mathcal{TS}$. The next corollaries follow from Propositions II.22 and II.23 (and the fact that the choice of initial vertices in $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ is arbitrary).

Corollary II.71.

Let $\mathcal{A}$ be a Muller automaton and let $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{A})$ be its ACD-parity-transform. Then, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathsf{ACD}_{\mathsf{parity}}(\mathcal{A}))$, and $\mathcal{A}$ is deterministic (resp. history-deterministic) if and only if $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{A})$ is deterministic (resp. history-deterministic).

Corollary II.72.

Let $\mathcal{G}$ be a Muller game and let $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{G})$ be its ACD-parity-transform. Eve wins $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{G})$ from a vertex of the form (v, n) if and only if she wins $\mathcal{G}$ from v.

4.3 An optimal history-deterministic transformation to Rabin transition systems

In this section we describe the ACD-HD-Rabin-transform, an optimal transformation of Muller TS to Rabin TS preserving history-determinism. This construction generalises the ZT-HD-Rabin-automaton (from Section II.3.3) in the same way as the ACD-parity-transform generalises the ZT-parity-automaton.

Definition II.73 (ACD-HD-Rabin-transform)**.**

Let $\mathcal{TS}$ be a Muller TS. For each vertex $v \in V$ we let $\eta_v \colon \mathsf{Leaves}(\mathcal{T}_v) \to \{1, \dots, \mathsf{rbw}(\mathcal{T}_v)\}$ be a mapping satisfying Property $(\star)$ from Lemma II.42.

We define the *ACD-HD-Rabin-transform* of $\mathcal{TS}$ to be the Rabin TS $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS}) = (G', \mathrm{Acc}')$, with $G' = (V', E', \mathsf{source}', \mathsf{target}', I')$, and $\mathrm{Acc}' = (\mathsf{col}', \mathsf{Nodes}(\mathcal{ACD}_{\mathcal{TS}}), \mathsf{Rabin}(R))$ defined as follows.

Vertices. The set of vertices is

$$V' = \bigcup_{v \in V} \left(\{v\} \times \{1, \dots, \mathsf{rbw}(\mathcal{T}_v)\}\right),$$

where $\mathsf{rbw}(\mathcal{T}_v)$ is the round-branching width of $\mathcal{T}_v$.

Initial vertices. $I' = \{(v_0, x) \mid v_0 \in I \text{ and } x \in \{1, \dots, \mathsf{rbw}(\mathcal{T}_{v_0})\}\}$.

Edges and output colours. We let

$$E' = \bigcup_{e \in E} \left(\{e\} \times \mathsf{Leaves}(\mathcal{T}_{\mathsf{source}(e)})\right).$$

For each edge $e = v \to v' \in E$ in $\mathcal{TS}$ and $x \in \{1, \dots, \mathsf{rbw}(\mathcal{T}_v)\}$, we will place one edge from (v, x) for each leaf l of $\mathcal{T}_v$ such that $\eta_v(l) = x$. More precisely, we let $(v, x) \xrightarrow{n} (v', x') \in E'$ if either

- v and v' are not in the same SCC (in this case the output colour n is irrelevant), or

- v and v' are in the same SCC and there are leaves l and l' of $\mathcal{T}_v$ and $\mathcal{T}_{v'}$, respectively, such that:
  - $\eta_v(l) = x$, $\eta_{v'}(l') = x'$,
  - $l' = \mathsf{Jump}_{\mathcal{T}_{v'}}(l, \mathsf{Supp}(l, e))$,
  - $n = \mathsf{Supp}(l, e)$.

Rabin condition. $R = \{(\mathfrak{g}_n, \mathfrak{r}_n)\}_{n \in \mathsf{Nodes}_{\bigcirc}(\mathcal{ACD}_{\mathcal{TS}})}$, where $\mathfrak{g}_n$ and $\mathfrak{r}_n$ are defined as follow: Let n be a round node, and let n' be any node in $\mathsf{Nodes}(\mathcal{ACD}_{\mathcal{TS}})$,

$$\begin{cases} n' \in \mathfrak{g}_n & \text{if } n' = n, \\ n' \in \mathfrak{r}_n & \text{if } n' \neq n \text{ and } n \text{ is not an ancestor of } n'. \end{cases}$$

Labellings. If $\mathcal{TS}$ is a labelled transition system, with labels $l_V \colon V \to L_V$ and $l_E \colon E \to L_E$, we label $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})$ by $l'_{V'}(v, x) = l_V(v)$ and $l'_{E'}(e') = l_E(e)$, if $e' \in E'(e)$.

Intuitively, a run in $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})$ can be identified with a promenade through the nodes of the ACD, which are used as the output colours to define the Rabin acceptance condition.

✦ Remark II.74. *The size of the ACD-HD-Rabin-transform of $\mathcal{TS}$ is:*

$$|\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})| = \sum_{v \in V} \mathsf{rbw}(\mathcal{T}_v) = \sum_{v \in V_{\mathrm{rec}}} \mathsf{rbw}(\mathcal{T}_v) + |V_{\mathrm{trans}}|,$$

where V_{rec} and V_{trans} are the sets of recurrent and transient vertices of $\mathcal{TS}$, respectively.

Correctness of the ACD-HD-Rabin-transform

To obtain the correctness of the ACD-HD-Rabin-transform, we follow the same steps as in the proof of the correctness of the ZT-HD-Rabin-automaton (Proposition II.48).

Proposition II.75 (Correctness of the ACD-HD-Rabin-transform)**.**

Let $\mathcal{TS}$ be a (labelled) Muller TS and let $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})$ be its ACD-HD-Rabin-transform. There is an HD mapping of (labelled) transition systems $\varphi \colon \mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS}) \to \mathcal{TS}$.

The proof of next two lemmas are completely analogous to those of Lemmas II.49 and II.50.

✦ **Lemma II.76.** *Let $u = n_0 n_1 n_2 \cdots \in \mathsf{Nodes}(\mathcal{ACD}_{\mathcal{TS}})^\omega$ be an infinite sequence of nodes of the ACD of $\mathcal{TS}$. The word u belongs to $\mathsf{Rabin}(R)$, for $R = \{(\mathfrak{g}_n, \mathfrak{r}_n)\}_{n \in \mathsf{Nodes}_{\bigcirc}(\mathcal{ACD}_{\mathcal{TS}})}$ the Rabin condition of $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})$, if and only if there is a unique minimal node for the ancestor relation in $\mathsf{Inf}(u)$ and this minimal node is round.*

✦ **Lemma II.77.** *There exists a morphism of transition systems $\varphi \colon \mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS}) \to \mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})$.*

Using these lemmas we can prove Proposition II.75.

Proof of Proposition II.75. We define the mapping $\varphi \colon \mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS}) \to \mathcal{TS}$ in the natural way: $\varphi_V(v, x) = v$ and $\varphi_E(e, l) = e$. It is immediate to check that φ is a weak

morphism. The fact that φ preserves accepting runs can be proven analogously to the fact that $\mathcal{L}(\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}) \subseteq \mathsf{Muller}_\Sigma(\mathcal{F})$ in Proposition II.48 (by using Lemma II.76).

Definition of a sound resolver for φ: In order to show how to simulate runs of $\mathcal{TS}$ in $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})$, we use the fact that we can see $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})$ as a quotient of $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ (Lemma II.77). Let $\tilde{\varphi}\colon \mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS}) \to \mathcal{TS}$ be the locally bijective morphism given by Proposition II.68, and let $\hat{\varphi}\colon \mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS}) \to \mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})$ be the morphism given by Lemma II.77. Since $\tilde{\varphi}$ is locally bijective, $\tilde{\varphi}_{\mathcal{R}uns}$ is a bijection between the runs of the transitions systems $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ and $\mathcal{TS}$, admitting an inverse $\tilde{\varphi}^{-1}_{\mathcal{R}uns}$. Composing this mapping with $\hat{\varphi}$, we obtain a way to simulate the runs from $\mathcal{TS}$ in $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})$:

$$\hat{\varphi}_{\mathcal{R}uns} \circ \tilde{\varphi}^{-1}_{\mathcal{R}uns}\colon \mathcal{R}un^\infty(\mathcal{TS}) \to \mathcal{R}un^\infty(\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})).$$

This composition of mappings provides a sound resolver simulating φ. Formally, let $(\mathsf{r}_{\mathsf{Init}}, r)$ be the resolver defined as follows. The choice of initial vertices $\mathsf{r}_{\mathsf{Init}}\colon I \to I'$ is given by $\mathsf{r}_{\mathsf{Init}}(v_0, x) = v_0$. The function $r\colon E'^* \times E \to E'$ associates to a finite run $\rho \in E'^*$ and $e \in E$ the last edge of the run $\hat{\varphi}(\tilde{\varphi}^{-1}(\varphi(\rho)e))$ (subscripts have been omitted for legibility). It is easy to check that $(\mathsf{r}_{\mathsf{Init}}, r)$ indeed defines a resolver simulating φ. Its soundness follows from the fact that $\tilde{\varphi}$ and $\hat{\varphi}$ preserve the acceptance of runs. ◀

As HD mappings are witnesses of the semantic equivalence of automata (Proposition II.22) we obtain:

Corollary II.78.

Let $\mathcal{A}$ be a Muller automaton and let $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{A})$ be its ACD-HD-Rabin-transform. Then, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{A}))$, and $\mathcal{A}$ is history-deterministic if and only if $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{A})$ is history-deterministic.

ACD-HD-Rabin-transform for games. As remarked in Section II.2, the existence of an HD mapping between two games does not guarantee that they have the same winner. This applies in particular to the ACD-HD-Rabin-transform, which might not preserve the winner of games. The reasons why this happens are the same as the difficulties encountered when we wanted to define the composition of a game $\mathcal{G}$ and an HD automaton $\mathcal{A}$ (see Section II.1): as the output of such operation, we would like to obtain a game in which Eve always chooses the transitions taken in $\mathcal{A}$, even if it is Adam who makes a move in the game, which is not the case if $\mathcal{G}$ is not suitable for transformations. In our case here, we can see the ACD-HD-Rabin-transform of a game $\mathcal{G}$ as a game in which, at each moment, first a move takes place in $\mathcal{G}$ and then a choice is made to update the node in $\mathcal{ACD}_\mathcal{G}$. With the current definition of $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{G})$, it is the player who makes the move in the game component who chooses how to update the node in $\mathcal{ACD}_\mathcal{G}$. This is potentially a problem, as we would like that Eve had full control to decide how to update the nodes in $\mathcal{ACD}_\mathcal{G}$, even when it was Adam who moved in the game component. In Appendix B.1, we propose a slight modification of the ACD-HD-Rabin-transform to obtain a transformation valid for games.

4.4 Optimality of the ACD-transforms

We now state and prove the optimality of both the ACD-parity-transform (Theorems II.5 and II.6) and the ACD-HD-Rabin-transform (Theorem II.7). The proofs of

these results will use the optimality of the automata based on the Zielonka tree (c.f. Section II.3) as a black-box, which will allow us to prove the optimality of both transformations at the same time. The key idea is that if $\varphi\colon \mathcal{TS} \to \mathcal{TS}'$ is an HD mapping, we can see $\mathcal{TS}$ as an HD automaton recognising the accepting runs of $\mathcal{TS}'$. We can then use local Muller conditions at vertices of $\mathcal{TS}'$ to reduce the problem to automata recognising Muller languages.

4.4.1 Statement of the optimality results

We state the optimality of the transformations based on the ACD. All the results below apply to labelled transition systems too. For technical reasons, we need to suppose that all the states of transition systems under consideration are accessible, hypothesis that can always be made without loss of generality. We recall that HD mappings are in particular locally bijective morphisms (c.f. Figure 10).

Theorem II.5.

Let $\mathcal{TS}$ be a Muller TS whose states are accessible and let $\widetilde{\mathcal{TS}}$ be a parity TS. If $\widetilde{\mathcal{TS}}$ admits an HD mapping $\varphi\colon \widetilde{\mathcal{TS}} \to \mathcal{TS}$, then, its acceptance condition uses at least as many priorities as that of $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$.

Theorem II.6.

Let $\mathcal{TS}$ be a Muller TS whose states are accessible and let $\widetilde{\mathcal{TS}}$ be a parity TS. If $\widetilde{\mathcal{TS}}$ admits an HD mapping $\varphi\colon \widetilde{\mathcal{TS}} \to \mathcal{TS}$, then, $|\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})| \leq |\widetilde{\mathcal{TS}}|$.

Theorem II.7.

Let $\mathcal{TS}$ be a Muller TS whose states are accessible and let $\widetilde{\mathcal{TS}}$ be a Rabin TS. If $\widetilde{\mathcal{TS}}$ admits an HD mapping $\varphi\colon \widetilde{\mathcal{TS}} \to \mathcal{TS}$, then, $|\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})| \leq |\widetilde{\mathcal{TS}}|$.

4.4.2 Discussion

Before presenting the proofs of the optimality theorems, we discuss some consequences and limitations of our results.

Difficulty of finding succinct history-deterministic automata

As mentioned in the introduction, several years had to pass after the introduction of history-deterministic automata [HP06] before finding HD automata that were actually smaller than equivalent deterministic ones [KS15]. As of today, we only know a handful of examples of ω-regular languages admitting succinct HD automata [AK22, KS15, CCL22], and their applicability in practice has yet to be fully determined. We assert that we can derive from our results some enlightening explanations on the difficulty of finding succinct HD parity automata, and set some limits in their usefulness in practical scenarios such as LTL synthesis.

First, Corollary II.36 already sets the impossibility of the existence of small HD parity automata recognising Muller languages. Corollary II.80 states that if an HD parity

automaton $\mathcal{A}$ has been obtained as a transformation of a DMA $\mathcal{B}$, then $\mathcal{A}$ is not strictly smaller than a minimal deterministic parity automaton for $\mathcal{L}(\mathcal{A})$.

Corollary II.79.

Let $\mathcal{TS}$ be a Muller TS. A minimal parity TS admitting an HD mapping to $\mathcal{TS}$ has the same size than a minimal parity TS admitting a locally bijective morphism to $\mathcal{TS}$.

Corollary II.80.

Let $\mathcal{A}$ be a history-deterministic parity automaton. Suppose that there exists a DMA $\mathcal{B}$ such that $\mathcal{A}$ admits an HD mapping to $\mathcal{B}$. Then, there exists a DPA $\mathcal{A}'$ recognising $\mathcal{L}(\mathcal{A})$ such that $|\mathcal{A}'| \leq |\mathcal{A}|$.

Both corollaries follow from an immediate application of Theorem II.6.

The ACD-transform does not preserve minimality

A natural question is whether the ACD-parity-transform preserves minimality of automata, that is, given a DMA $\mathcal{A}$ with a minimal number of states for the language it recognises, is $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{A})$ minimal amongst DPAs recognising $\mathcal{L}(\mathcal{A})$?[10] The answer to this question is negative, as we show now.

Proposition II.81.

There exists a DMA $\mathcal{A}$ that is minimal amongst DMAs recognising $\mathcal{L}(\mathcal{A})$, but such that its ACD-parity-transform $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{A})$ is not a minimal DPA.

We consider the alphabet $\Sigma = \{a, b, c\}$ and the language

$$L = \{w \in \Sigma^\omega \mid c \in \mathsf{Inf}(w) \text{ and } w \text{ contains infinitely often the factor } ab\}.$$

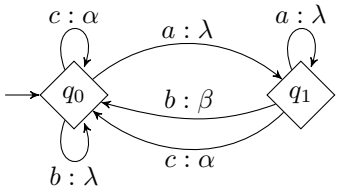
A minimal DMA for L is depicted in Figure 18a. Its minimality follows simply from the fact that, as L is not a Muller language ($(abc)^\omega \in L$ but $(bac)^\omega \notin L$, c.f. Remark I.11), a DMA with just one state cannot recognise L. In Figure 18 we show its alternating cycle decomposition and its ACD-parity-transform, that has 4 states. However, we can find a DPA with just 3 states recognising L, as shown in Figure 18d.

4.4.3 Optimality of the parity condition of $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$

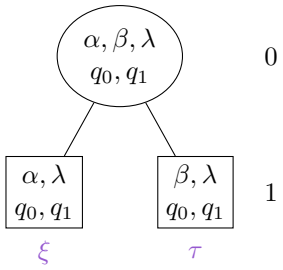
We show next the proof of Theorem II.5. To prove this result, we would like to use the Flower Lemma I.21, however, the statement of Theorem II.5 does not involve ω-regular languages. In order to set up a context in which apply the Flower Lemma, we show that, whenever we have a morphism $\varphi\colon \mathcal{TS} \to \mathcal{TS}'$, $\mathcal{TS}$ can be seen as an automaton reading the runs of $\mathcal{TS}'$.

Let $\mathcal{TS} = (G, \mathrm{Acc})$ and $\mathcal{TS}' = (G', \mathrm{Acc}')$ be transition systems with underlying graphs $G = (V, E, \mathsf{source}, \mathsf{target}, I)$ and $G' = (V', E', \mathsf{source}', \mathsf{target}', I')$, and acceptance conditions $\mathrm{Acc} = (\mathsf{col}, \Gamma, W)$ and $\mathrm{Acc}' = (\mathsf{col}', \Gamma', W')$. A weak morphism of transition systems $\varphi\colon \mathcal{TS} \to \mathcal{TS}'$ provides a labelling of the edges of $\mathcal{TS}$ by $\varphi_E\colon E \to E'$. Therefore, we

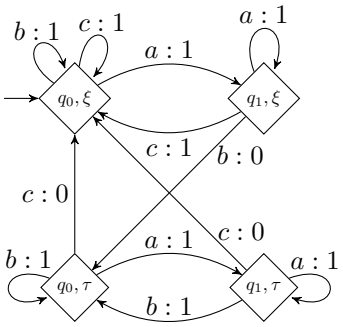
[10] This question was left open as a conjecture in [CCF21].



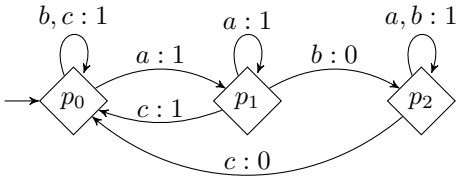
(a) A Muller automaton with acceptance set given by $\mathcal{F} = \{\{\alpha, \beta\}\}$.



(b) Alternating cycle decomposition of $\mathcal{A}$. To indicate the labels of the nodes of this ACD, we include just the colours of the corresponding edges.



(c) ACD-parity-transform of $\mathcal{A}$, $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{A})$, with 4 states.



(d) A parity automaton recognising L with only 3 states .

✦ **Figure 18.** A minimal DMA whose ACD-parity-transform is not a minimal DPA.

can see $\mathcal{TS}$ as an automaton with input alphabet E', inheriting the underlying graph and acceptance condition from $\mathcal{TS}$. We say that this is the *automaton of morphism* φ and denote it by $\mathcal{A}_\varphi$.

We define the *language of accepting runs of* a transition system $\mathcal{TS}$ as:

$$\mathcal{L}_{\mathcal{R}uns}(\mathcal{TS}) = \{\rho \in E^\omega \mid \rho \text{ is an accepting run in } \mathcal{TS}\}.$$

Lemma II.82.

Let $\mathcal{TS}$ and $\mathcal{TS}'$ be transition systems with a single initial state, let $\varphi\colon \mathcal{TS} \to \mathcal{TS}'$ be a weak morphism of transition systems, and let $\mathcal{A}_\varphi$ be its automaton. Then, φ is an HD mapping if and only if the automaton $\mathcal{A}_\varphi$ is history-deterministic, and, in this case,

$$\mathcal{L}(\mathcal{A}_\varphi) = \mathcal{L}_{\mathcal{R}uns}(\mathcal{TS}').$$

Proof. We first note that a resolver for $\mathcal{A}_\varphi$ (in the sense of HD automata) is a mapping of the form $r\colon E^* \times E' \to E$, as E' is the input alphabet of this automaton. A resolver simulating φ (in the sense of HD mappings) is a mapping of the same form. It is straightforward to check that (q_0, r) is a sound resolver for $\mathcal{A}_\varphi$ if and only if $(\mathsf{r}_{\mathsf{Init}}, r)$ is a sound resolver simulating φ (where $\mathsf{r}_{\mathsf{Init}}(q_0') = q_0$ is the only possible choice of initial vertex).

We prove that $\mathcal{L}(\mathcal{A}_\varphi) = \{\rho' \in \mathcal{R}un(\mathcal{TS}') \mid \rho' \text{ is an accepting run}\}$. First, we remark that if ρ is a run in $\mathcal{A}_\varphi$ over $\rho' \in \mathcal{R}un(\mathcal{TS}')$, then $\rho' = \varphi_{\mathcal{R}uns}(\rho)$, since the labelling of $\mathcal{A}_\varphi$ by input letters is given exactly by φ itself. Therefore, if $\rho' \in \mathcal{L}(\mathcal{A}_\varphi)$, there exists an

accepting run ρ over ρ', and since φ preserves accepting runs, $\rho' = \varphi_{\mathcal{R}uns}(\rho)$ is accepting in $\mathcal{TS}'$, proving the inclusion from left to right. For the other inclusion, we let $(\mathsf{r}_{\mathsf{Init}}, r)$ be a sound resolver simulating φ. If ρ' is an accepting run in $\mathcal{TS}'$, then $\mathsf{r}_{\mathcal{R}uns}(\rho')$ is an accepting run over ρ' in $\mathcal{A}_\varphi$. ◀

We recall that $[\min_{\mathcal{TS}}, \max_{\mathcal{TS}}]$ are the priorities used by the ACD-parity-transform of $\mathcal{TS}$, which coincides with the maximal height of a tree in $\mathcal{ACD}_{\mathcal{TS}}$. We also recall that $\min_{\mathcal{TS}} = 0$ if $\mathcal{ACD}_{\mathcal{TS}}$ is positive or equidistant, and that $\min_{\mathcal{TS}} = 1$ if $\mathcal{ACD}_{\mathcal{TS}}$ is negative.

✦ **Lemma II.83.** *Let $\mathcal{TS}$ be a Muller TS, and let $\mathsf{AltTree}(\ell) \in \mathcal{ACD}_{\mathcal{TS}}$ be a positive (resp. negative) tree of the ACD of $\mathcal{TS}$ of height d. Then, $\mathcal{TS}$ admits a positive (resp. negative) d-flower.*

Proof. We use the same argument than the one used in the proof of Theorem II.1. Let $n_1 \preceq n_2 \preceq \ldots \preceq n_d$ be a branch of length d of $\mathsf{AltTree}(\ell)$ (where n_1 is the root and n_d is a leaf of the tree). Let $v \in \nu_{\mathsf{States}}(n_d)$ be a vertex appearing in the leaf. Then, the whole branch is contained in $\mathcal{T}_v$ (by Remark II.54), that is, $\nu(n_i) \in \mathcal{C}ycles_{\mathcal{TS}}(v)$. Moreover, $\nu(n_1) \supsetneq \nu(n_2) \supsetneq \ldots \nu(n_d)$ is a chain that alternates accepting and rejecting cycles, so it is a d-flower that is positive if and only if $\nu(n_1) = \ell$ is an accepting cycle, that is, if $\mathsf{AltTree}(\ell)$ is positive. ◀

✦ **Lemma II.84.** *Let $\mathcal{TS}$ be a Muller TS with a single initial vertex and whose vertices are all accessible. Then, the parity index of $\mathcal{L}_{\mathcal{R}uns}(\mathcal{TS})$ is:*

- $[\min_{\mathcal{TS}}, \max_{\mathcal{TS}}]$ *if $\mathcal{ACD}_{\mathcal{TS}}$ is positive or negative,*
- $\mathsf{Weak}_{\max_{\mathcal{TS}}}$ *if $\mathcal{ACD}_{\mathcal{TS}}$ is equidistant.*

Proof. We consider the identity morphism $\mathsf{Id}_{\mathcal{TS}} \colon \mathcal{TS} \to \mathcal{TS}$ and its automaton $\mathcal{A}_{\mathsf{Id}_{\mathcal{TS}}}$, which is a deterministic automaton trivially recognising $\mathcal{L}_{\mathcal{R}uns}(\mathcal{TS})$ (that is, we see $\mathcal{TS}$ as an automaton reading its own edges as input letters). The result follows from the Flower Lemma I.21 and the fact that a tree $\mathsf{AltTree}(\ell) \in \mathcal{ACD}_{\mathcal{TS}}$ of height d provides a d-flower that is positive if ℓ is accepting and negative if ℓ is rejecting (Lemma II.83). These flowers are accessible as we have supposed that all the vertices of $\mathcal{TS}$ are accessible. ◀

The previous lemmas allow us to obtain Theorem II.5 for transition systems with a single initial vertex. We introduce some further notations to deal with the general case.

For a Muller TS $\mathcal{TS}$ and a vertex v, we let $\mathcal{ACD}_{(\mathcal{TS},v)}$ be the alternating cycle decomposition of the accessible part of $\mathcal{TS}$ from v. We note that the trees of $\mathcal{ACD}_{(\mathcal{TS},v)}$ are a subset of the trees of $\mathcal{ACD}_{\mathcal{TS}}$: a tree $\mathsf{AltTree}(\ell_i) \in \mathcal{ACD}_{\mathcal{TS}}$ appears in $\mathcal{ACD}_{(\mathcal{TS},v)}$ if and only if the cycle ℓ_i is accessible from v. Accordingly, for each vertex v of $\mathcal{TS}$ we let $\min_{(\mathcal{TS},v)}$ (resp. $\max_{(\mathcal{TS},v)}$) be the minimum (resp. maximum) value taken by the function $p_{\mathcal{ACD}}$ when restricted to the trees of $\mathcal{ACD}_{(\mathcal{TS},v)}$.

✦ Remark II.85. *For every transition system $\mathcal{TS}$, one of the two following statements holds:*

- *There is some vertex v such that $[\min_{\mathcal{TS}}, \max_{\mathcal{TS}}] = [\min_{(\mathcal{TS},v)}, \max_{(\mathcal{TS},v)}]$.*
- *There are two vertices v_0 and v_1 such that $\min_{(\mathcal{TS},v_0)} = 0$, $\max_{(\mathcal{TS},v_0)} = \max_{\mathcal{TS}} - 1$ and $\min_{(\mathcal{TS},v_1)} = 1$, $\max_{(\mathcal{TS},v_1)} = \max_{\mathcal{TS}}$.*

Moreover, if all the states of $\mathcal{TS}$ are accessible, we can choose v (resp. v_0 and v_1) to be an initial vertex.

We can finally deduce Theorem II.5 from the preceding lemmas.

Proof of Theorem II.5. We suppose that we are in the first case of Remark II.85 (a proof for the second case follows easily). First, we show that we can suppose that $\widetilde{\mathcal{TS}}$ and $\mathcal{TS}$ have a single initial vertex. Let v be an initial vertex of $\mathcal{TS}$ such that $[\min_{\mathcal{TS}}, \max_{\mathcal{TS}}] = [\min_{(\mathcal{TS},v)}, \max_{(\mathcal{TS},v)}]$. Let $\varphi\colon \widetilde{\mathcal{TS}} \to \mathcal{TS}$ be an HD mapping, and let $(\mathsf{r}_{\mathsf{Init}}, r)$ be a sound resolver simulating it. We let $\tilde{v} = \mathsf{r}_{\mathsf{Init}}(v)$ be the initial vertex in $\widetilde{\mathcal{TS}}$ chosen by the resolver. It suffices then to prove the result for the accessible part of $\widetilde{\mathcal{TS}}$ from $\tilde{v}$, the transition system $\mathcal{TS}_v$, and the restriction of φ to these transition systems.

From now on, we suppose that both $\widetilde{\mathcal{TS}}$ and $\mathcal{TS}$ have a single initial vertex. By Lemma II.84 and Proposition I.20, a parity history-deterministic automaton recognising $\mathcal{L}_{\mathcal{R}uns}(\mathcal{TS})$ uses at least $|[\min_{\mathcal{TS}}, \max_{\mathcal{TS}}]|$ priorities. By Lemma II.82, the automaton $\mathcal{A}_\varphi$ of the morphism φ is a parity history-deterministic automaton recognising $\mathcal{L}_{\mathcal{R}uns}(\mathcal{TS})$, and therefore uses at least $|[\min_{\mathcal{TS}}, \max_{\mathcal{TS}}]|$ priorities. Since the acceptance condition of $\widetilde{\mathcal{TS}}$ is exactly the same as that of $\mathcal{A}_\varphi$, we can conclude. ◀

4.4.4 Optimality of the sizes of $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ and $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})$

We prove now Theorems II.6 and II.7. We first give an intuitive idea of the techniques used.

Sketch of the proof. Let $\varphi\colon \widetilde{\mathcal{TS}} \to \mathcal{TS}$ be an HD mapping, and let v be a vertex in $\mathcal{TS}$. We can see the set $\varphi^{-1}(v)$ as the states of an HD automaton reading finite runs in $\mathcal{TS}$ looping around v. This allows to define an HD automaton having $\varphi^{-1}(v)$ as set of states and recognising $\mathsf{LocalMuller}_{\mathcal{TS}}(v)$. As the Zielonka tree of $\mathsf{LocalMuller}_{\mathcal{TS}}(v)$ is the tree $\mathcal{T}_v$, by optimality of the ZT-parity-automaton (resp. the ZT-HD-Rabin-automaton), we deduce that $|\varphi^{-1}(v)| \geq |\mathsf{Leaves}(\mathcal{T}_v)|$ (resp. $|\varphi^{-1}(v)| \geq \mathsf{rbw}(\mathcal{T}_v)$). ◀

Definition II.86.

Let $\mathcal{TS}$ and $\mathcal{TS}'$ be two transition systems, and let $(\mathsf{col}, \Gamma, \mathsf{Muller}_\Gamma(\mathcal{F}))$ be the acceptance condition of $\mathcal{TS}$. Let $\varphi\colon \mathcal{TS} \to \mathcal{TS}'$ be a weak morphism of transition systems that is locally surjective, and let v' be an accessible recurrent state of $\mathcal{TS}'$. For each $\ell' \in \mathcal{C}ycles_{v'}(\mathcal{TS}')$ we let $\rho_{\ell'}$ be a finite path starting and ending in v' visiting exactly the edges of ℓ'. We define the *cycle-preimage-automaton at v'* to be the Muller automaton $\mathcal{A}_{(\varphi^{-1},v')} = (Q_{v'}, \mathcal{C}ycles_{v'}(\mathcal{TS}'), Q_{v'}, 2^\Gamma_+, \delta, \mathsf{Muller}_{2^\Gamma_+}(\tilde{F}))$ over the input alphabet $\mathcal{C}ycles_{\mathcal{TS}'}(v')$ defined as:

- the set of states is $Q_{v'} = \varphi^{-1}(v')$,
- all the states are initial,
- the output colours are non-empty subsets of the colours used by $\mathcal{TS}$,
- $(q_2, C) \in \delta(q_1, \ell')$ if there is a finite path $\rho \in \mathcal{P}ath^{\mathsf{fin}}_{q_1}(\mathcal{TS})$ from q_1 to q_2 such that $\varphi(\rho) = \rho_{\ell'}$ producing as output the colours in $C \subseteq \Gamma$, that is $\mathsf{col}(\rho) = C$. If C is empty, this corresponds to an uncoloured edge $q_1 \xrightarrow{\ell':\varepsilon} q_2$. We remark that, since φ is supposed locally surjective, there is at least one such path ρ.

- $\{C_1, \dots, C_k\} \in \tilde{F}$ if and only if $\cup_{i=1}^{k} C_i \in \mathcal{F}$.

We remark that a transition $e = q_1 \xrightarrow{\ell':C} q_2$ in $\mathcal{A}_{(\varphi^{-1},v')}$ induces a finite path $\mathsf{Unfold}(e) = q_1 \overset{C}{\rightsquigarrow} q_2$ in $\mathcal{TS}$ called the *unfolding* of e, producing as output the set of colours C and such that $\varphi(\mathsf{Unfold}(e)) = \ell'$. In particular, a run ρ in $\mathcal{A}_{(\varphi^{-1},v')}$ is accepting if and only if $\mathsf{Unfold}(\rho)$ is accepting.

✦ **Lemma II.87.** *If $L = \mathsf{Muller}_\Gamma(\mathcal{F})$ is a parity (resp. Rabin) language, then, so is the language $\tilde{L} = \mathsf{Muller}_{2^\Gamma_+}(\tilde{F})$ used by the acceptance condition of $\mathcal{A}_{(\varphi^{-1},v')}$.*

Proof. Suppose that L is a parity language, that is, there are $d_{\min} \leq d_{\max}$ and $\phi\colon \Gamma \to [d_{\min}, d_{\max}]$ such that for any non-empty subset $C \subseteq \Gamma$, $C \in \mathcal{F}$ if and only if $\min \phi(C)$ is even. We define $\tilde{\phi}\colon 2^\Gamma_+ \to [d_{\min}, d_{\max}]$ as: $\tilde{\phi}(C) = \min \phi(C)$. It is immediate to see that $\{C_1, \dots, C_k\} \in \tilde{F}$ if and only if $\min \tilde{\phi}(\{C_1, \dots, C_k\})$ is even.

Suppose now that L is a Rabin language represented by the Rabin pairs $\{(\mathfrak{g}_1, \mathfrak{r}_1), \dots, (\mathfrak{g}_r, \mathfrak{r}_r)\}$. We define a family of Rabin pairs $\tilde{R} = \{(\tilde{\mathfrak{g}}_1, \tilde{\mathfrak{r}}_1), \dots, (\tilde{\mathfrak{g}}_r, \tilde{\mathfrak{r}}_r)\}$ for the language $\tilde{L}$ as: $\{C_1, \dots, C_k\} \in \tilde{\mathfrak{g}}_i$ if $\cup_{i=1}^{k} C_i \in \mathfrak{g}_i$ and $\{C_1, \dots, C_k\} \in \tilde{\mathfrak{r}}_i$ if $\cup_{i=1}^{k} C_i \in \mathfrak{r}_i$. It is immediate to check that $\tilde{L} = \mathsf{Rabin}_{2^\Gamma_+}(\tilde{R})$. ◀

✦ **Lemma II.88.** *Let $\mathcal{TS}$ and $\mathcal{TS}'$ be two Muller TS, $\varphi\colon \mathcal{TS} \to \mathcal{TS}'$ a weak morphism of TS, and v' an accessible recurrent state of $\mathcal{TS}'$. If φ is an HD mapping, then the automaton $\mathcal{A}_{(\varphi^{-1},v')}$ is history-deterministic and recognises the local Muller condition of $\mathcal{TS}'$ at v'.*

Proof. $\mathcal{L}(\mathcal{A}_{(\varphi^{-1},v')}) \subseteq \mathsf{LocalMuller}_{v'}(\mathcal{TS})$**:** Let $\ell'_1\ell'_2\cdots \in \mathit{Cycles}_{v'}(\mathcal{TS})^\omega$ be a sequence of cycles accepted by $\mathcal{A}_{(\varphi^{-1},v')}$. By prefix-independence of Muller languages we can suppose that all the cycles ℓ'_i appear infinitely often. Let $\rho = q_0 \xrightarrow{\ell'_1:C_1} q_1 \xrightarrow{\ell'_2} q_2 \to \dots$ be an accepting run in $\mathcal{A}_{(\varphi^{-1},v')}$ over $\ell'_1\ell'_2\dots$, and let $\mathsf{Unfold}(\rho)$ be its unfolding. As ρ is an accepting run, so is $\mathsf{Unfold}(\rho)$, and since φ preserves accepting runs, $\varphi(\mathsf{Unfold}(\rho))$ is an accepting run in $\mathcal{TS}'$. The edges visited by $\varphi(\mathsf{Unfold}(\rho))$ form the cycle $\cup_{i\geq 1}\ell'_i$, which is therefore an accepting cycle, so $\ell'_1\ell'_2\cdots \in \mathsf{LocalMuller}_{v'}(\mathcal{TS})$ by definition of local Muller condition.

$\mathsf{LocalMuller}_{\mathcal{TS}'}(v') \subseteq \mathcal{L}(\mathcal{A}_{(\varphi^{-1},v')})$ **and history-determinism:** Let $r_\varphi : E^* \times E' \to E$ be a sound resolver simulating φ. We will transfer the strategy given by r_φ to define a resolver $r_{\mathcal{A}} : \Delta^* \times \mathit{Cycles}_{v'}(\mathcal{TS}') \to \Delta$ for $\mathcal{A}_{(\varphi^{-1},v')}$, where Δ is the set of transitions of the automaton. Let $\rho'_0 \in \mathcal{R}un^{\mathsf{fin}}(\mathcal{TS}')$ be a finite run reaching v', and let $\rho_0 = \mathsf{r}_{\varphi,\mathcal{R}uns}(\rho'_0)$ the preimage given by the resolver, ending in some $q_0 \in Q_{v'}$ that is going to by used as initial state for $\mathcal{A}_{(\varphi^{-1},v')}$. For a sequence $e_1e_2\dots e_k \in \Delta^*$ and $\ell' \in \mathit{Cycles}_{v'}(\mathcal{TS}')$, we let

$$r_{\mathcal{A}}(e_1e_2\dots e_k, \ell') = r_\varphi(\rho'_0\rho'_1\dots\rho'_k, \rho_{\ell'}),^{11}$$

where $\rho'_j = \varphi(\mathsf{Unfold}(e_j))$ and $v' \overset{\rho_{\ell'}}{\rightsquigarrow} v'$ is the finite run corresponding to ℓ' fixed in the definition of $\mathcal{A}_{(\varphi^{-1},v')}$. By definition, the obtained resolver satisfies the following property:

> If $e_1e_2\cdots \in \Delta^\omega$ is the run induced by $r_{\mathcal{A}}$ over $\ell'_1\ell'_2\cdots \in \mathit{Cycles}_{v'}(\mathcal{TS}')^\omega$,
> then $\rho_0\mathsf{Unfold}(e_1e_2\dots)) = \mathsf{r}_{\varphi,\mathcal{R}uns}(\rho'_0\rho'_1\rho'_2\dots)$.

[11] Here we use a slight abuse of notation, since, formally, r_φ takes as input elements in $E^* \times E'$, but $\rho_{\ell'} \in E'^*$. We can naturally extend r_φ to E'^* by induction. Equivalently, we can say that $r_{\mathcal{A}}(e_1e_2\dots e_k, \ell')$ is a suffix of $\mathsf{r}_{\varphi,\mathcal{R}uns}(\rho'_0\rho'_1\dots\rho'_k\rho_{\ell'})$.

This gives us:

$$\bigcup \mathsf{Inf}(\ell'_1, \ell'_2, \dots) \text{ is accepting cycle in } \mathcal{TS}' \iff \rho'_0\rho'_1\rho'_2\dots \text{ is accepting run in } \mathcal{TS}' \implies$$
$$\implies \rho_0\mathsf{Unfold}(e_1e_2\dots) \text{ accepting run in } \mathcal{TS} \iff e_1e_2\dots \text{ accepting run in } \mathcal{A}_{(\varphi^{-1},v')}.$$

Which allows us to conclude that the $\mathcal{A}_{(\varphi^{-1},v')}$ recognises $\mathsf{LocalMuller}_{v'}(\mathcal{TS})$ and that $r_{\mathcal{A}}$ is a sound resolver. ◀

Corollary II.89.

Let $\mathcal{TS}$ and $\widetilde{\mathcal{TS}}$ be a Muller and a parity transition system, respectively, and let $\varphi\colon \widetilde{\mathcal{TS}} \to \mathcal{TS}$ be an HD mapping. Let v be an accessible recurrent state of $\mathcal{TS}$. Then,

$$|\varphi^{-1}(v)| \geq |\mathsf{Leaves}(\mathcal{Z}_{\mathsf{LocalMuller}_{\mathcal{TS}}(v)})| = |\mathsf{Leaves}(\mathcal{T}_v)|.$$

Proof. By Lemma II.88, the automaton $\mathcal{A}_{(\varphi^{-1},v)}$ is a history-deterministic automaton recognising $\mathsf{LocalMuller}_v(\mathcal{TS})$ of size $|\varphi^{-1}(v)|$, and by Lemma II.87, it is a parity automaton. The optimality of the ZT-parity-automaton (Theorem II.2) gives us the first inequality. The second equality follows from the fact that $\mathcal{T}_v$ is the Zielonka tree of $\mathsf{LocalMuller}_{\mathcal{TS}}(v)$ (Lemma II.58). ◀

The next corollary admits an identical proof, using the optimality of the ZT-HD-Rabin-automaton (Theorem II.4).

Corollary II.90.

Let $\mathcal{TS}$ and $\widetilde{\mathcal{TS}}$ be a Muller and a Rabin transition system, respectively, and let $\varphi\colon \widetilde{\mathcal{TS}} \to \mathcal{TS}$ be an HD mapping. Let v be an accessible recurrent state of $\mathcal{TS}$. Then,

$$|\varphi^{-1}(v)| \geq |\mathsf{rbw}(\mathcal{Z}_{\mathsf{LocalMuller}_{\mathcal{TS}}(v)})| = |\mathsf{rbw}(\mathcal{T}_v)|.$$

Theorems II.6 and II.7 follow from these two corollaries, the formulas for the size of the ACD-transforms (Remarks II.64 and II.74) and the fact that a locally surjective morphism $\varphi : \widetilde{\mathcal{TS}} \to \mathcal{TS}$ is surjective if all vertices of $\mathcal{TS}$ are accessible (Lemma II.12).

5 Computational aspects of the Zielonka tree and the ACD

We start in Section II.5.1 by comparing the size of different representations of Muller languages and translations between them. Then, we analyse the size of the Zielonka tree and Zielonka DAG in the worst case. By Theorems II.2 and II.4, lower bounds for the size of the Zielonka tree directly translate into lower bounds for history-deterministic parity and Rabin automata. We recover in this way some results from Löding [Löd99] and generalise them to history-deterministic automata. Missing proofs and further results are included in Appendix B.3.

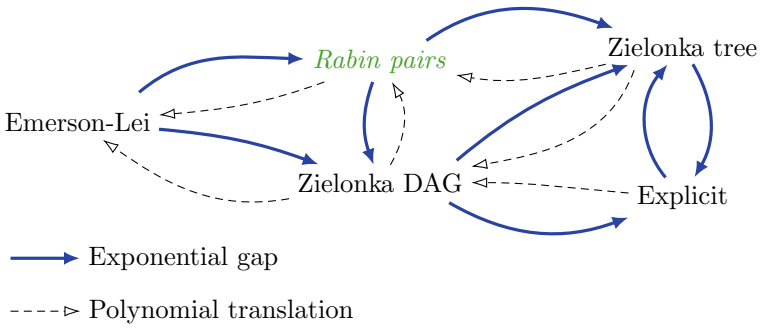
In Section II.5.2, we discuss the complexity of computing the ACD and the ACD-DAG. We show that computing the ACD is not harder than computing the Zielonka tree: given a Muller TS with its acceptance condition represented as a Zielonka tree (resp. Zielonka

DAG), we can compute its ACD (resp. ACD-DAG) in polynomial time (Theorems II.8 and II.9). These results have various implications:

- They provide support for the assertion that the transformations defined in the previous section are actually applicable in practical scenarios. Further backing for this claim is offered by the implementation of the ACD-transforms in Spot 2.10 [DL+22] and Owl 21.0 [KMS18] (see [Cas+22]).
- In Section II.8, we will show that corresponding problems for state-based models are NP-hard. This provides further evidence in favour of transition-based acceptance.
- These results will allow us to prove in next section that we can decide the typeness of Muller transition systems in polynomial time (Theorem II.10).

5.1 Size of the Zielonka tree and the ACD

5.1.1 Translations between different representations of Muller languages



✦ **Figure 19.** Comparison between the different representations of Muller conditions. A blue bold arrow from X to Y means that converting an X-representation into the form Y might require an exponential blow-up. A dashed arrow from X to Y means that it is possible to transform an X-representation into the form Y in polynomial time. We remark that the representation using Rabin pairs does only apply to the subclasses of Rabin and Streett languages. We note that dashed arrows (polynomials translations) are closed by transitivity.

In Figure 19 we represent the main relations between different representations of Muller languages used in this paper, as introduced in Section I.3.2. Detailed proofs and examples can be found in Appendix B.3, see also [Bok19, HD05, Hug23] for more details about these and other representations of Muller languages.

Emerson-Lei conditions – which represent a Muller language by a boolean formula – are the most succinct ones, and for this reason they are commonly used in practice. All problems considered in the following are trivially NP-hard when using Muller languages represented as Emerson-Lei conditions. This is the case, for example, for the problem of computing a child of a node of the Zielonka tree. For this reason, we will seldom consider Emerson-Lei conditions in the rest of the document.

The Zielonka DAG, first introduced by Hunter and Dawar [HD05], has recently been shown to offer a good trade-off between conciseness and desirable algorithmic properties [Hug23]. This makes it a sensible option for representing Muller languages and eval-

uating the complexity of decision problems related to Muller automata. Almost by definition, we can obtain a Zielonka DAG from a Zielonka tree in polynomial time. However, we show in Proposition B.32 that the Zielonka tree can be exponentially larger. Hunter and Dawar showed that we can compute a Zielonka DAG in polynomial time from an explicit representation of a Muller language [HD05, Theorem 3.3]. Zielonka trees and explicit representations are incomparable: there are families $\mathcal{F}$ for which $\mathcal{Z}_\mathcal{F}$ is exponentially larger than $|\mathcal{F}|$ and vice-versa (Propositions B.29 and B.30). We also remark that the representation of a Muller language using a Zielonka tree and a deterministic parity automaton are equivalent, by Theorem II.2 and Proposition III.18.

We remark that the representation using a family of Rabin pairs only applies to Rabin or Streett languages. Arrows involving this representation do only apply to these subclasses of languages; for example the dotted arrow from "Zielonka DAG" to "Rabin pairs" should be read: "if $\mathcal{Z}$-DAG$_\mathcal{F}$ is a Zielonka DAG such that the language associated to $\mathcal{F}$ is a Rabin language, then we can compute a representation as a Rabin condition in polynomial time".

5.1.2 Worst case analysis

We study the size of the Zielonka tree in the worst case. By Remark II.57, the given bounds apply to the ACD, as the Zielonka tree can be seen as the ACD of a Muller TS with just one state.

In all this subsection, Σ_n will stand for an alphabet of size n, for example, $\Sigma_n = \{1, \dots, n\}$.

We define the *double factorial* of a positive integer n as:

$$n!! = n(n-2)(n-4)\cdots 4 \cdot 2 \quad \text{if } n \text{ even}, \qquad n!! = n(n-2)(n-4)\cdots 3 \cdot 1 \quad \text{if } n \text{ odd}.$$

Proposition II.91 (Size of the Zielonka tree: Worst case)**.**

Let $\mathcal{F} \subseteq 2^\Sigma_+$ be a family of subsets, and let $m = |\Sigma|$ be the number of letters of the alphabet. It is satisfied:

- $|\mathcal{Z}_\mathcal{F}| \leq 1 + m + m(m-1) + \cdots + m!$,
- $|\mathsf{Leaves}(\mathcal{Z}_\mathcal{F})| \leq m!$,
- $|\mathsf{rbw}(\mathcal{Z}_\mathcal{F})| \leq m!!$, and
- the height of $\mathcal{Z}_\mathcal{F}$ is at most m.

These bound are tight: for all $m \in \mathbb{N}$, there is a family $\mathcal{F}_m \subseteq 2^{\Sigma_m}_+$ over an alphabet of m letters such that the previous relations are equalities.

Proof. We start by showing that the given bounds are tight. We suppose that m is even (the construction is symmetric if m is odd), and let $\Sigma_m = \{1, \dots, m\}$. Consider the family[12] $\mathsf{EvenLetters}_m \subseteq 2^{\Sigma_m}_+$ given by:

$$\mathsf{EvenLetters}_m = \{C \subseteq \Sigma_m \mid |C| \text{ is even}\}.$$

First, we remark that the last inequality follows from the fact that the subsets $\Sigma_m, \Sigma_m \setminus \{1\}, \dots, \Sigma_m \setminus \{1, \dots, m-1\}$ form a branch of the Zielonka tree. Let n be a node of the

[12] This family of subsets already appear in the worst-case study of parity automata recognising a Muller language in Mostowski's paper introducing the parity condition [Mos84, p.161].

Zielonka tree of $\mathsf{EvenLetters}_m$, and let $X_n = \nu(n)$ be its label. Then n has a child for each subset of X_n of size $|X_n| - 1$. A simple induction gives that the level at depth k of the Zielonka tree has $m(m-1)\cdots(m-(k-1))$ nodes. This gives the first three equalities of the statement.

We prove now the upper bounds. The last item follows from the fact that the label of a node is a set of size strictly smaller than the label of his parent. Let $\mathcal{F} \subseteq 2^{\Sigma}_{+}$, and $m = |\Sigma|$. We show by recurrence that the Zielonka tree $\mathcal{Z}_{\mathcal{F}}$ is not bigger than that of $\mathsf{EvenLetters}_m$. We remark that for all $X \subseteq \Sigma_m$, there is a node n_X in the Zielonka tree of $\mathsf{EvenLetters}_m$ labelled X, and that the subtree rooted at n_X is isomorphic to the Zielonka tree of $\mathsf{EvenLetters}_{|X|}$. Let n_0 be the root of $\mathcal{Z}_{\mathcal{F}}$, and let $n_1, \dots, n_k$ be its children. Then, we can find in the Zielonka tree of $\mathsf{EvenLetters}_m$ k incomparable nodes having as labels $\nu(n_1), \dots, \nu(n_k)$. By induction hypothesis, the subtree rooted at each of these nodes is not smaller than the subtrees rooted at $n_1, \dots, n_k$. This shows the two first items.

To prove the third item we need to be slightly more careful and pay attention to round nodes and square nodes. We show the result for the case in which $\Sigma \in \mathcal{F}$ and m even (the other three cases are analogous, by taking if necessary a symmetric condition $\mathsf{OddLetters}_m$). Let n_0 be a node of $\mathcal{Z}_{\mathcal{F}}$, let $n_1, n_2, \dots, n_k$ be its children, and let n_0' be a node in the Zielonka tree of $\mathsf{EvenLetters}_m$ such that n_0' is round if and only if n_0 is round and such that $\nu(n_0) \subseteq \nu(n_0')$. Then, we can find k descendants of n_0' in the Zielonka tree of $\mathsf{EvenLetters}_m$, $n_1', \dots, n_k'$ such that:

- nodes n_i' are round if and only if nodes n_i are round,
- $\nu(n_i) \subseteq \nu(n_i')$.

Suppose that n_0 and n_0' are round (the proof is similar if they are square). Let T_x be the subtree rooted at a node x. Then, it is satisfied:

$$\mathsf{rbw}(T_{n_0}) = \sum_{i=1}^{k} \mathsf{rbw}(T_{n_i}) \leq \sum_{i=1}^{k} \mathsf{rbw}(T_{n_i'}) \leq \mathsf{rbw}(T_{n_0}),$$

where the first inequality follows from induction hypothesis, and the last one by definition of round-branching width. We conclude by taking n_0 the root of $\mathcal{Z}_{\mathcal{F}}$. ◀

We recover results analogous to those of Löding [Löd99], and strengthen them as they apply to history-deterministic automata. These directly follow combining the previous proposition with Theorems II.2 and II.4.

Corollary II.92.

For every Muller language $L \subseteq \Sigma^\omega$ there exists a DPA recognising L of size at most $|\Sigma|!$. This bound is tight: for all n, a minimal history-deterministic parity automaton recognising the Muller language associated to $\mathsf{EvenLetters}_n$ has $n!$ states.

Corollary II.93.

Let $L \subseteq \Sigma^\omega$ be a Muller language. There exists a history-deterministic Rabin automaton recognising L of size at most $n!! = n(n-2)\cdots 2$. This bound is tight: for all n even, a minimal history-deterministic Rabin automaton recognising the Muller language associated to $\mathsf{EvenLetters}_n$ has $n!!$ states.

5.2 The ACD Computation

We discuss now how to compute the alternating cycle decomposition of a Muller transition system, and the computational cost of it. We prove that if the acceptance condition is represented as a Zielonka tree, we can compute $\mathcal{ACD}_{\mathcal{TS}}$ in polynomial time (Theorem II.8) and if it is represented as a Zielonka DAG, we can compute $\mathcal{ACD}\text{-}\mathsf{DAG}_{\mathcal{TS}}$ in polynomial time (Theorem II.9). This second result will be used to show that we can decide the typeness of Muller TS in polynomial time (Theorem II.10).

In the whole section, $\mathcal{TS}$ stands for a Muller transition system with set of states and edges V and E, respectively, and acceptance condition $\mathrm{Acc}_{\mathcal{TS}} = (\mathsf{col}, \Gamma, \mathsf{Muller}_\Gamma(\mathcal{F}))$.

5.2.1 The complexity of computing the ACD: Main results

We now state the main results concerning the complexity of the computation of the ACD. Their proof will be covered in the next section and Appendix B.4.

Theorem II.8 (Computation of the ACD)**.**

Given a Muller transition system $\mathcal{TS}$ with acceptance condition represented by a Zielonka tree $\mathcal{Z}_\mathcal{F}$, we can compute $\mathcal{ACD}_{\mathcal{TS}}$ in polynomial time in $|V| + |\mathcal{Z}_\mathcal{F}|$.

We note that the Zielonka tree of $\mathcal{F}$ can be exponentially larger than the Zielonka DAG of $\mathcal{F}$ (Proposition B.32), and (at least) super-polynomially larger than an explicit representation of $\mathcal{F}$ (Proposition B.30). Therefore, as the Zielonka tree is a special case of the ACD (c.f. Remark II.57), we cannot compute $\mathcal{ACD}_{\mathcal{TS}}$ in polynomial time in the representation of $\mathcal{TS}$ if the acceptance condition is given as a list of subsets or as a Zielonka DAG. However, in that case, we can compute in polynomial time the ACD-DAG, which is sufficient for many applications (see Section II.6).

Theorem II.9 (Computation of the ACD-DAG)**.**

Given a Muller transition system $\mathcal{TS}$ with acceptance condition represented by a Zielonka DAG $\mathcal{Z}\text{-}\mathsf{DAG}_\mathcal{F}$, we can compute $\mathcal{ACD}\text{-}\mathsf{DAG}_{\mathcal{TS}}$ in polynomial time in $|V| + |\mathcal{Z}\text{-}\mathsf{DAG}_\mathcal{F}|$.

Corollary II.94.

Given a Muller transition system $\mathcal{TS}$ with acceptance condition $\mathsf{Muller}_\Gamma(\mathcal{F})$ represented explicitly as a list of subsets, we can compute $\mathcal{ACD}\text{-}\mathsf{DAG}_{\mathcal{TS}}$ in polynomial time in $|V| + |\mathcal{F}| + |\Gamma|$.

Proof. By Proposition B.31, given an explicit representation of a Muller language $\mathsf{Muller}_\Gamma(\mathcal{F})$, we can compute its Zielonka DAG in polynomial time. It suffices then to apply Theorem II.9. ◀

Corollary II.95.

We can compute the alternating cycle decomposition of a parity transition system $\mathcal{TS}$ in polynomial time in the size of the representation of $\mathcal{TS}$.

Proof. The Zielonka tree of a parity condition can be trivially computed in polynomial time in the representation of the parity condition. It suffices then to apply Theorem II.8. Alternatively, it is possible to give a parity-TS-oriented version of Algorithm 2, which could be subject to further improvements. The algorithm obtained in such way is analogous to the algorithm of Carton and Maceiras [CM99]. ◀

We obtain similarly:

Corollary II.96.

We can compute the alternating cycle decomposition of a generalised (co)Büchi transition system $\mathcal{TS}$ in polynomial time in the size of the representation of $\mathcal{TS}$.

Proposition II.97 (Computation of the ACD-transforms)**.**

Given a Muller transition system $\mathcal{TS}$ with the acceptance condition represented as a Zielonka tree, we can compute in polynomial time $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ and $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})$.

Proposition II.98 (Degeneralisation of generalised Büchi transition systems)**.**

Given a generalised (co)Büchi transition system $\mathcal{TS}$, we can compute in polynomial time a (co)Büchi TS admitting a locally bijective morphism to $\mathcal{TS}$ with a minimal number of states.

5.2.2 A generic algorithm for computing the ACD

We present the pseudocode of a high-level algorithm computing $\mathcal{ACD}_{\mathcal{TS}}$ (Algorithm 1). This algorithm builds the ACD in a top-down fashion: first, it computes an SCC decomposition of $\mathcal{TS}$ and initialises the root of each of the trees in $\mathcal{ACD}_{\mathcal{TS}}$. Then, it successively computes the children of the already found nodes. A first naive algorithm to compute the children of a node is given in the sub-procedure `ComputeChildren`, presented in Algorithm 2. Given a node n labelled with $\nu(n) = \ell$ (suppose that ℓ is an accepting cycle), `ComputeChildren` first enumerate the maximal subsets of colours of $\mathsf{col}(\ell)$ that are rejecting. For each of these subsets D, we restrict ℓ to the edges coloured with colours in D, and compute a decomposition in SCC of the resulting graph. If the subcycles that are found in this way are rejecting, they are hold as potential children of n; if they are accepting, we continue in a recursive way. Finally, the children of n will correspond to the maximal subcycles that have been retained by this procedure.

The computation of children using Algorithm 2 will suffice to prove Theorem II.8. In order to obtain Theorem II.9, we provide an enhanced version of the algorithm in Appendix B.4 (Algorithm 4).

We use the following notations:

- `Edges`$(\mathcal{S})$ is the set of edges appearing in a subgraph $\mathcal{S}$.
- `SCC-Decomposition`$(\mathcal{S})$ outputs a list of the strongly connected components of a graph $\mathcal{S}$. If $\mathcal{S}$ is empty, it outputs an empty list.
- `pop`(stck) removes an element from the stack stck and returns it.
- `push`$(\mathsf{stck}, \mathsf{L})$ adds the elements of L to the stack stck,

- `MaxInclusion`(lst) returns the list of the maximal subsets in lst.

All the previous functions can be computed in polynomial time. We will also make use of:

- `MaxAltSubsets`$(C, \mathcal{F})$ takes a subset $C \subseteq \Gamma$ and returns the maximal subsets $D \subseteq C$ such that $D \in \mathcal{F} \iff C \notin \mathcal{F}$.

The computational cost of `MaxAltSubsets` depends on the representation of $\mathcal{F}$ (see details below).

Algorithm 1 Computation of the alternating cycle decomposition

Input: A transition system $\mathcal{TS}$
Output: $\mathcal{ACD}_{\mathcal{TS}}$

$\langle \mathcal{S}_1, \dots, \mathcal{S}_r \rangle \leftarrow$ `SCC-Decomposition`$(\mathcal{TS})$
Add $\mathcal{S}_1, \dots, \mathcal{S}_r$ as the root of r different trees of $\mathcal{ACD}_{\mathcal{TS}}$
nodesToTreat $\leftarrow \langle \mathcal{S}_1, \dots, \mathcal{S}_r \rangle$ ▷ Initialise a stack
while nodesToTreat $\neq \emptyset$ **do**
    n $\leftarrow$ `pop`(nodesToTreat)
    children $\leftarrow$ `ComputeChildren`(n)
    Add the nodes children as children of the node n in $\mathcal{ACD}_{\mathcal{TS}}$
    `push`(nodesToTreat, children)
end while
return $\mathcal{ACD}_{\mathcal{TS}}$

Algorithm 2 `ComputeChildren`$(\mathcal{S})$: Computing the children of a node of the ACD

Input: A strongly connected subgraph $\mathcal{S}$ corresponding to a node of $\mathcal{ACD}_{\mathcal{TS}}$
Output: The maximal cycles $\ell_1, \dots, \ell_k \in \mathcal{C}ycles(\mathcal{S})$ such that $\mathsf{col}(\ell_i) \in \mathcal{F} \iff \mathsf{col}(\mathcal{S}) \notin \mathcal{F}$.

$C \leftarrow \mathsf{col}(\mathcal{S})$
children $\leftarrow \{\}$
maxAltSets $\leftarrow$ `MaxAltSubsets`$(C, \mathcal{F})$
for $D \in$ maxAltSets **do**
    $\mathcal{S}_D \leftarrow$ restriction of $\mathcal{S}$ to transitions $e \in E$ such that $\mathsf{col}(e) \in D$
    $\langle \mathcal{S}_{D,1}, \dots, \mathcal{S}_{D,r} \rangle \leftarrow$ `SCC-Decomposition`$(\mathcal{S}_D)$
    for $i = 1, \dots, r$ **do**
        if $\mathsf{col}(\mathcal{S}_{D,i}) \in \mathcal{F} \iff C \notin \mathcal{F}$ **then**
            children $\leftarrow$ children $\cup \{$`Edges`$(\mathcal{S}_{D,i})\}$
        else
            children $\leftarrow$ children $\cup$ `ComputeChildren`$(\mathcal{S}_{D,i})$
        end if
    end for
end for
children $\leftarrow$ `MaxInclusion`(children)
return children

We remark that the algorithm we propose also serves to compute the ACD-DAG of $\mathcal{TS}$; when adding the children of a node in Line 7 of Algorithm 1, we just have to identify the (possibly multiple) parents of this node amongst the nodes computed so far.

Informal analysis. We provide a formal analysis of the complexity of the algorithm above in Appendix B.4. We discuss here the main ideas of it. The cost of the computation of the alternating cycle decomposition of $\mathcal{TS}$ following Algorithm 1 is proportional to the size of $\mathcal{ACD}_{\mathcal{TS}}$ and to the cost of computing the children of a node using the procedure `ComputeChildren` (Algorithm 2). The upper bound on the size of $|\mathcal{ACD}_{\mathcal{TS}}|$ is shown in Lemma B.42, and uses some non-trivial properties of the Zielonka tree. To prove that Algorithm 2 takes polynomial time in the size of $\mathcal{Z}_{\mathcal{F}}$, for each state q of $\mathcal{TS}$ we consider the tree of recursive calls of the kind `ComputeChildren`$(\mathcal{S}_{D,i})$ in which q is a state of the subgraph $\mathcal{S}$. We exhibit an injection of this tree of recursive calls into $\mathcal{Z}_{\mathcal{F}}$, obtaining the desired result.

Efficient computation of the ACD-DAG. We remark that the argument above does not suffice to show that we can compute the children of a node of the ACD in polynomial time with respect to the size of $\mathcal{Z}$-DAG$_{\mathcal{F}}$. Indeed, the tree of recursive calls mentioned above does not necessarily embed in $\mathcal{Z}$-DAG$_{\mathcal{F}}$, and the procedure `ComputeChildren` can take exponential time with respect to this measure. In Appendix B.4, we give an improved version of `ComputeChildren` (see Algorithm 4) which terminates in polynomial time in $|\mathcal{TS}|+|\mathcal{Z}\text{-DAG}_{\mathcal{F}}|$, providing a proof for Theorem II.9. Informally, this is achieved by using a kind of "memoisation guided by the Zielonka DAG": we keep a stack of subgraphs $\mathcal{S}$ that we need to inspect, and we add a new subgraph $\mathcal{S}'$ to the stack only if there is not any "similar enough" subgraph already appearing in the stack that can be merged with $\mathcal{S}'$. In this way, we can guarantee that the size of the stack will remain polynomial in $|\mathcal{TS}| \cdot |\mathcal{Z}\text{-DAG}_{\mathcal{F}}|$.

5.2.3 Implementation

The computation of the ACD and the ACD-parity-transform has been implemented in Spot 2.10 [DL+22], by Alexandre Duret-Lutz and Florian Renkin, and in Owl 21.0 [KMS18] by Klara J. Meyer and Salomon Sickert. In the tool paper [Cas+22], we compare this transformation with the state-of-the-art methods to transform Emerson-Lei automata into parity ones. The results are conclusive: in all cases, automata output by the ACD-parity-transform are smaller than automata generated by other methods. Quite surprisingly, the ACD-transform also outperforms the existing paritizing methods in computational time; this can be (partly) explained by the build-up of heuristics used by other methods, and the simplicity offered by the approach using the ACD.

Two main improvements are introduced to optimise the performance of the computation of the ACD:

- The use of memoisation to remember parts of the ACD that have already been computed, as a same subtree can appear multiple times in the ACD.
- A set of rules to simplify the booleans formulas of the Emerson-Lei condition, before passing it into a `MAX-Sat` solver.

6 Typeness results: Relabelling automata with simpler acceptance conditions

In this section, we discuss some further applications of the Zielonka tree and the alternating cycle decomposition. We use the insights gained from the ACD to conduct a comprehensive study of typeness results for deterministic Muller automata, that is, we focus on the following question:

> Given a Muller automaton $\mathcal{A}$, can we define a simpler acceptance condition on top of the underlying graph of $\mathcal{A}$ obtaining an equivalent automaton $\mathcal{A}'$?

This question was first studied (in the context of automata using state-based acceptance) by Krishnan, Puri and Brayton [KPB94, KPB95], who showed how to determine if a DMA can be relabelled with an equivalent Büchi condition. Their work was generalized to parity automata by Boker, Kupferman and Steinitz [BKS10], and related questions about typeness were studied for non-deterministic automata by Kupferman, Morgenstern and Murano [KMM06], and for history-deterministic automata by Boker, Kupferman and Skrzypczak [BKS17].

In this section, we provide new general characterisations of typeness for Muller transition systems. In Propositions II.109, II.110 and II.111, we characterise when a Muller TS can be relabelled with equivalent parity, Rabin, or Streett conditions in terms of properties of the cycles of the TS. For instance, Proposition II.109 states that a Muller TS can be relabelled with an equivalent Rabin condition if and only if its rejecting cycles are closed under union. The "only if" part of this results was already known (see for instance [Löd99]), but the fact that this is indeed a characterisation is a novel result, for which the use of the ACD is essential. These characterisations directly imply the results from [BKS10, KPB94, KPB95]. We also show how to use the ACD to determine the parity index of the language recognised by a DMA (Proposition II.115), which can be seen as a simplification of the results from [KPB95, Section 3.2]. We also give further results concerning generalised (co)Büchi languages and weak automata.

As a corollary of these results, we obtain that we can decide the typeness of a Muller transition system in polynomial time (that is, telling whether we can put a simpler equivalent acceptance condition on top of the TS, see Theorem II.10) and we can also decide the parity index of a language recognised by a deterministic Muller automaton in polynomial time (Theorem II.11).

In Section II.6.3 we study the problem of minimising the number of colours used by a Muller TS. We show that, in general, the problem is NP-hard (Theorem II.13), although we can decide the minimal number of colours necessary to define a Muller language in polynomial time (this corresponds to the case of TS with just one state, see Theorem II.12).

Structure of the section. We start in Subsection II.6.1 by studying the typeness question for Muller languages, without taking into account the structure of transition systems. The results we obtain are based in the Zielonka tree. In Subsection II.6.2 we generalise these results for transitions systems, using the ACD, and discuss several

consequences, as the decidability of typeness in polynomial time. In Subsection II.6.3 we consider the problem of the minimisation of colours for Muller TS.

6.1 Typeness for Muller languages

We present some results proven by Zielonka [Zie98, Section 5] that show how we can use the Zielonka tree to deduce if a Muller language is a Rabin, a Streett or a parity language. These results are generalised to transition systems in the next subsection.

The results of this subsection already appear in [Zie98, Section 5] (except for Proposition II.105) and they are special cases of the results appearing in Section II.6.2.

We first introduce some definitions. The terminology will be justified by the upcoming results.

Definition II.99 (Shape of Zielonka tree)**.**

Let T be a tree T with nodes partitioned into round nodes and square nodes. We say that T has:

- *Rabin shape* if every round node has at most one child.
- *Streett shape* if every square node has at most one child.
- *Parity shape* if every node has at most one child.
- *Büchi shape* (resp. *coBüchi shape*) if it has parity shape, the height of T is at most 2, and if it is exactly 2, the root is round (resp. square).
- *Generalised Büchi shape* (resp. *generalised coBüchi shape*) if the height of T is at most 2, and if it is exactly 2, the root is round (resp. square).

We define analogously the different *shapes* for the Zielonka DAG.

✦ Remark II.100. *For every family $\mathcal{F}$ and for X any of the types of conditions appearing in the previous definition, the Zielonka tree of $\mathcal{F}$ has X shape if and only if the Zielonka DAG of $\mathcal{F}$ has X shape.*

Proposition II.101 (Zielonka tree for Rabin languages)**.**

Let $\mathcal{F} \subseteq 2^{\Gamma}_{+}$ be a family of non-empty subsets. The following conditions are equivalent:

1. $\mathsf{Muller}_\Gamma(\mathcal{F})$ is a Rabin language.
2. $\overline{\mathcal{F}}$ is closed under union: If $C_1 \notin \mathcal{F}$ and $C_2 \notin \mathcal{F}$, then $C_1 \cup C_2 \notin \mathcal{F}$.
3. $\mathcal{Z}_{\mathcal{F}}$ has Rabin shape.
4. $\mathcal{Z}$-$\mathsf{DAG}_{\mathcal{F}}$ has Rabin shape.

The reader can refer to Proposition B.33 for a direct proof of the fact that if $\mathcal{Z}$-$\mathsf{DAG}_{\mathcal{F}}$ has Rabin shape, then $\mathsf{Muller}_\Gamma(\mathcal{F})$ is a Rabin language.

Proposition II.102 (Zielonka tree for Streett languages)**.**

Let $\mathcal{F} \subseteq 2^{\Gamma}_{+}$ be a family of non-empty subsets. The following conditions are equivalent:

1. $\mathsf{Muller}_\Gamma(\mathcal{F})$ is a Streett language.
2. The family $\mathcal{F}$ is closed under union.
3. $\mathcal{Z}_\mathcal{F}$ has Streett shape.
4. $\mathcal{Z}$-$\mathsf{DAG}_\mathcal{F}$ has Streett shape.

Proposition II.103 (Zielonka tree for parity languages)**.**

Let $\mathcal{F} \subseteq 2^\Gamma_+$ be a family of non-empty subsets. The following conditions are equivalent:

1. $\mathsf{Muller}_\Gamma(\mathcal{F})$ is a parity language.
2. Both $\mathcal{F}$ and $\overline{\mathcal{F}}$ are closed under union: If $C_1 \in \mathcal{F} \iff C_2 \in \mathcal{F}$, then, $C_1 \cup C_2 \in \mathcal{F} \iff C_1 \in \mathcal{F}$.
3. $\mathcal{Z}_\mathcal{F}$ has parity shape.
4. $\mathcal{Z}$-$\mathsf{DAG}_\mathcal{F}$ has parity shape.

Moreover, if some of these conditions is satisfied, $\mathsf{Muller}_\Gamma(\mathcal{F})$ is a $[\min_\mathcal{F}, \max_\mathcal{F}]$-parity language. In particular, $\mathsf{Muller}_\Gamma(\mathcal{F})$ is a Büchi (resp. coBüchi) language if and only if $\mathcal{Z}_\mathcal{F}$ has Büchi shape (resp. coBüchi shape).

Corollary II.104.

A Muller language $L \subseteq \Gamma^\omega$ is a parity language if and only if it is both a Rabin and a Streett language.

Proposition II.105 (Zielonka tree for generalised (co)Büchi languages)**.**

Let $\mathcal{F} \subseteq 2^\Gamma_+$ be a family of non-empty subsets. $\mathsf{Muller}_\Gamma(\mathcal{F})$ is a generalised Büchi language (resp. generalised coBüchi language) if and only if $\mathcal{Z}_\mathcal{F}$ has generalised Büchi shape (resp. generalised coBüchi shape).

Proof. We prove the case generalised Büchi (symmetric for generalised coBüchi). Assume first that $\mathsf{Muller}_\Gamma(\mathcal{F}) = \mathsf{genBuchi}_\Gamma(B)$ for some family $B = \{B_1, \dots, B_k\}$. Then, $\Gamma \in \mathcal{F}$, as $\Gamma \cap B_i \neq \emptyset$ for all i, so the root of $\mathcal{Z}_\mathcal{F}$ is round. If $C \subseteq \Gamma$ is rejecting, $C \cap B_i = \emptyset$ for some i, so for any subset $C' \subseteq C$ we have that $C' \cap B_i = \emptyset$ too. Therefore, square nodes of $\mathcal{Z}_\mathcal{F}$ are leaves and $\mathcal{Z}_\mathcal{F}$ has height at most 2.

Conversely, suppose that $\mathcal{Z}_\mathcal{F}$ has height 2 and that its root is round ($\Gamma \in \mathcal{F}$). Let $A_1, \dots, A_k$ be the labels of the k leaves of $\mathcal{Z}_\mathcal{F}$ and define $B_i = \Gamma \setminus A_i$. We claim that $\mathsf{Muller}_\Gamma(\mathcal{F}) = \mathsf{genBuchi}_\Gamma(B)$, for $B = \{B_1, \dots, B_k\}$. Indeed:

$$C \in \mathcal{F} \iff C \not\subseteq A_i \text{ for all } i \iff C \cap B_i \neq \emptyset \text{ for all } i. \qquad \blacktriangleleft$$

6.2 Typeness for Muller transition systems and deterministic automata

In this section, we state and prove our main contributions concerning typeness of transition systems. Results are organised in the following 4 paragraphs:

- Characterisation of typeness of transition systems using the alternating cycle decomposition.
- Utilisation of the ACD to determine the parity index of languages recognised by deterministic Muller automata.
- Semantic consequences of our typeness results for languages recognised by deterministic automata.
- Complexity of the problem of deciding typeness.

We recall that a transition system $\mathcal{TS}$ is X type (for X one of types of languages defined in Section I.3) if we can define an equivalent acceptance condition of type X on top of $\mathcal{TS}$ (or equivalently, if there exists an isomorphic transition system $\mathcal{TS}' \simeq \mathcal{TS}$ using an X acceptance condition).

The ACD determines the type of transition systems

Definition II.106 (Types of ACD)**.**

Let $\mathcal{TS}$ be a Muller transition system with a set of states V. We say that its alternating cycle decomposition $\mathcal{ACD}_{\mathcal{TS}}$ is a:

- *Rabin ACD* if for every state $v \in V$, the tree $\mathcal{T}_v$ has Rabin shape.
- *Streett ACD* if for every state $v \in V$, the tree $\mathcal{T}_v$ has Streett shape.
- *Parity ACD* if for every state $v \in V$, the tree $\mathcal{T}_v$ has parity shape.
- $[0, d-1]$*-parity ACD* (resp. $[1, d]$-parity ACD) if it is a parity ACD, trees of $\mathcal{ACD}_{\mathcal{TS}}$ have height at most d and trees of height d are positive (resp. negative).
- *Büchi ACD* (resp. *coBüchi ACD*) if it is a $[0, 1]$-parity ACD (resp. $[1, 2]$-parity ACD).
- *Generalised Büchi ACD* (resp. *generalised coBüchi ACD*) if for every state $v \in V$, the tree $\mathcal{T}_v$ has generalised Büchi shape (resp generalised coBüchi shape).
- Weak_d *ACD* if it is a parity ACD and trees of $\mathcal{ACD}_{\mathcal{TS}}$ have height at most d.

We define analogously the different *types* of ACD-DAGs.

✦ Remark II.107. *For every Muller transition system $\mathcal{TS}$, $\mathcal{ACD}_{\mathcal{TS}}$ is an X-ACD if and only if $\mathcal{ACD}$-$\mathsf{DAG}_{\mathcal{TS}}$ is an X-ACD-DAG, for any of the types of conditions considered above.*

✦ Remark II.108. *$\mathcal{ACD}_{\mathcal{TS}}$ is a parity ACD if and only if it is both a Rabin and a Streett ACD. Also, $\mathcal{ACD}_{\mathcal{TS}}$ is a Weak_d ACD if and only if it is a $[0, d]$-parity ACD and a $[1, d+1]$-parity ACD.*

Proposition II.109 (Characterisation of Rabin TS)**.**

Let $\mathcal{TS} = (G_{\mathcal{TS}}, \mathrm{Acc}_{\mathcal{TS}})$ be a Muller transition system whose states are accessible. The following conditions are equivalent:

1. $\mathcal{TS}$ is Rabin type.

2. For every pair of rejecting cycles $\ell_1, \ell_2 \in \mathit{Cycles}(\mathcal{TS})$ with some state in common, $\ell_1 \cup \ell_2$ is a rejecting cycle.
3. $\mathcal{ACD}_{\mathcal{TS}}$ is a Rabin ACD.
4. $\mathcal{ACD}$-$\mathsf{DAG}_{\mathcal{TS}}$ is a Rabin ACD-DAG.

Proof. $(1 \Rightarrow 2)$ Let $\mathrm{Acc}_R = (\mathsf{col}, \Gamma, \mathsf{Rabin}(R)$ be the Rabin acceptance condition equivalent to $\mathrm{Acc}_{\mathcal{TS}}$, and let $R = (\mathfrak{g}_1, \mathfrak{r}_1), \ldots, (\mathfrak{g}_r, \mathfrak{r}_r)$ be its Rabin pairs. Let ℓ_1 and ℓ_2 be two cycles with a state in common, and suppose that $\ell_1 \cup \ell_2$ is accepting; we show that either ℓ_1 or ℓ_2 is accepting. The cycle $\ell_1 \cup \ell_2$ is accepted by some Rabin pair $(\mathfrak{g}_j, \mathfrak{r}_j)$, so for all edges $e \in \ell_1 \cup \ell_2$, $\mathsf{col}(e) \notin \mathfrak{r}_j$, and there is some $e_0 \in \ell_1 \cup \ell_2$ such that $\mathsf{col}(e_0) \in \mathfrak{g}_j$. If e_0 belongs to ℓ_1, then ℓ_1 is accepted by the Rabin pair $(\mathfrak{g}_j, \mathfrak{r}_j)$, and if $e_0 \in \ell_2$, then ℓ_2 is accepted by it.

$(2 \Rightarrow 3)$ Let v be a vertex of $\mathcal{TS}$ and $\mathcal{T}_v$ the local subtree at v. Suppose that there is a round node $n \in \mathcal{T}_v$ with two different children n_1 and n_2. The cycles $\nu(n_1)$ and $\nu(n_2)$ are rejecting cycles over v, but their union is an accepting cycle (by Remark II.52).

$(3 \Rightarrow 1)$ We observe that $\mathcal{ACD}_{\mathcal{TS}}$ is a Rabin ACD if and only if $\mathsf{rbw}(\mathcal{T}_v) = 1$ for all vertices v of $\mathcal{TS}$. In particular, the ACD-HD-Rabin-transform of $\mathcal{TS}$ does not add any state to $\mathcal{TS}$. It is immediate to check that the morphism $\varphi\colon \mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS}) \to \mathcal{TS}$ given by $\varphi_V(v, x) = v$, $\varphi_E(e, l) = e$ defined in the proof of Proposition II.75 is an isomorphism, and $\mathcal{TS}$ uses a Rabin acceptance condition.

$(3 \iff 4)$ Given by Remark II.107. ◀

Proposition II.110 (Characterisation of Streett TS)**.**

Let $\mathcal{TS} = (G_{\mathcal{TS}}, \mathrm{Acc}_{\mathcal{TS}})$ be a Muller transition system. The following conditions are equivalent:

1. $\mathcal{TS}$ is Streett type.
2. For every pair of accepting cycles $\ell_1, \ell_2 \in \mathit{Cycles}(\mathcal{TS})$ with some state in common, $\ell_1 \cup \ell_2$ is an accepting cycle.[13]
3. $\mathcal{ACD}_{\mathcal{TS}}$ is a Streett ACD.
4. $\mathcal{ACD}$-$\mathsf{DAG}_{\mathcal{TS}}$ is a Streett ACD-DAG.

Proof. Implications $(1 \Rightarrow 2)$, $(2 \Rightarrow 3)$ and $(3 \iff 4)$ are analogous to those from Proposition II.109.

$(3 \Rightarrow 1)$ We consider the transition system $\overline{\mathcal{TS}}$ obtained by complementing the acceptance set of $\mathrm{Acc}_{\mathcal{TS}}$. By Remark II.56, the ACD of $\overline{\mathcal{TS}}$ is obtained from $\mathcal{ACD}_{\mathcal{TS}}$ by turning round nodes into square nodes and vice-versa. Thus, the ACD of $\overline{\mathcal{TS}}$ is a Rabin

[13] This property was introduced by Le Saëc under the name *cyclically closed automata* [Saë90]. We point out that the "if" direction of the result stated in [Saë90, Theorem 5.2] does not hold. That statement can be rephrased as: If a DMA $\mathcal{A}$ is cyclically closed, then the parity index of $\mathcal{A}$ is $[0, 1]$. We refer to Proposition II.115 for a correct characterisation.

ACD, and by applying the previous proposition we can define a Rabin condition $\mathrm{Acc}_R = (\mathsf{col}, \Gamma, \mathsf{Rabin}_\Gamma(R))$ such that the transition system $(G_{\overline{\mathcal{TS}}}, \mathrm{Acc}_R)$ is isomorphic to $\overline{\mathcal{TS}}$. Since $\mathsf{Streett}_\Gamma(R)$ is the complement language of $\mathsf{Rabin}_\Gamma(R)$ (Remark I.13), we obtain that $\mathrm{Acc}_S = (\mathsf{col}, \Gamma, \mathsf{Streett}_\Gamma(R))$ is a Streett acceptance condition equivalent to $\mathrm{Acc}_{\mathcal{TS}}$ over $G_{\mathcal{TS}}$. ◀

Proposition II.111 (Characterisation of parity TS)**.**

Let $\mathcal{TS} = (G_{\mathcal{TS}}, \mathrm{Acc}_{\mathcal{TS}})$ be a Muller transition system. The following conditions are equivalent:

1. $\mathcal{TS}$ is parity type.
2. For every pair of accepting (resp. rejecting) cycles $\ell_1, \ell_2 \in \mathit{Cycles}(\mathcal{TS})$ with some state in common, $\ell_1 \cup \ell_2$ is an accepting (resp. rejecting) cycle.
3. $\mathcal{ACD}_{\mathcal{TS}}$ is a parity ACD.
4. $\mathcal{ACD}\text{-}\mathsf{DAG}_{\mathcal{TS}}$ is a parity ACD-DAG.

Moreover, if some of the conditions is satisfied, $\mathcal{TS}$ is $[0, d-1]$-parity type (resp. $[1, d]$-parity type / Weak_d type) if and only if $\mathcal{ACD}_{\mathcal{TS}}$ is a $[0, d-1]$-parity ACD (resp. $[1, d]$-parity ACD / Weak_d ACD).

In particular, $\mathcal{TS}$ is Büchi (resp. coBüchi) type if and only if $\mathcal{ACD}_{\mathcal{TS}}$ is a Büchi ACD (resp. coBüchi ACD).

Proof. $(1 \Rightarrow 2)$ Proven in Lemma II.39.

$(2 \Rightarrow 3)$ Admits an analogous proof to the corresponding implication in Proposition II.109.

$(3 \Rightarrow 1)$ By definition, $\mathcal{ACD}_{\mathcal{TS}}$ is a parity ACD if and only if $\mathsf{Leaves}(\mathcal{T}_v)$ is a singleton for each vertex v of $\mathcal{TS}$. In particular, the ACD-parity-transform of $\mathcal{TS}$ does not add any state to $\mathcal{TS}$. It is immediate to check that the morphism $\varphi\colon \mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS}) \to \mathcal{TS}$ defined in the proof of Proposition II.68 is an isomorphism. Therefore, $\mathcal{TS}$ and $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ are isomorphic transition systems, and the latter uses a parity acceptance condition that is a $[0, d-1]$-parity condition if $\mathcal{ACD}_{\mathcal{TS}}$ is a $[0, d-1]$-parity ACD. If $\mathcal{ACD}_{\mathcal{TS}}$ is not a $[0, d-1]$-parity ACD, then the number of colours cannot be reduced by the optimality of the number of colours of the ACD-parity-transform (Theorem II.5). The proof is analogous for the cases $[1, d]$ and Weak_d.

$(3 \iff 4)$ Given by Remark II.107 ◀

Corollary II.112.

A Muller transition system is parity type if and only if it is both Rabin and Streett type.

Corollary II.113.

A Muller transition system is Weak_d-type if and only if it is both $[0, d]$ and $[1, d+1]$-parity type.

Proposition II.114 (Characterisation of generalised (co)Büchi TS)**.**

A transition system $\mathcal{TS}$ is generalised Büchi (resp. generalised coBüchi) type if and only if $\mathcal{ACD}_{\mathcal{TS}}$ is a generalised Büchi ACD (resp. generalised coBüchi ACD).

Proof. The result follows by applying the same argument and construction than in Proposition II.105, using as set of output colours the set of edges of $\mathcal{TS}$. ◀

The ACD and the parity index of ω-regular languages

Proposition II.115 (Reading the parity index from the ACD)**.**

Let $\mathcal{A}$ be a deterministic Muller automaton whose states are accessible. Then, the parity index of $\mathcal{L}(\mathcal{A})$ is:

- $[0, d-1]$ (resp. $[1, d]$) if and only if:
  - trees of $\mathcal{ACD}_{\mathcal{A}}$ have height at most d,
  - there is at least one tree of height d, and
  - trees of height d are positive (resp. negative).
- Weak_d if and only if:
  - trees of $\mathcal{ACD}_{\mathcal{A}}$ have height at most d,
  - there is at least one positive tree of height d, and
  - there is at least one negative tree of height d.

Moreover, all these conditions are equivalent to the corresponding ones replacing $\mathcal{ACD}_{\mathcal{TS}}$ by $\mathcal{ACD}\text{-}\mathsf{DAG}_{\mathcal{TS}}$.

Proof. The left-to-right implication follows easily from the Flower Lemma I.21 in all cases.

We prove the right-to-left implication for the case Weak_d – which poses some extra technical difficulties. Suppose that $\mathcal{ACD}_{\mathcal{A}}$ verifies the previous list of conditions (in particular, it is equidistant). Then, the ACD-parity-transform $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{A})$ is a DPA recognising $\mathcal{L}(\mathcal{A})$ using priorities in $[0, d]$. In order to obtain a DPA for $\mathcal{L}(\mathcal{A})$ with priorities in $[1, d+1]$, we need to introduce a small modification to the function $p_{\mathcal{ACD}}$. For ℓ_i a maximal cycle of $\mathcal{A}$ and $n \in N_{\ell_i}$ we define:

- $p'_{\mathcal{ACD}}(n) = \mathsf{Depth}(n) + 2$, if ℓ_i is accepting,
- $p'_{\mathcal{ACD}}(n) = \mathsf{Depth}(n) + 1$, if ℓ_i is rejecting.

It is a routine check to see that the version of the ACD-parity-transform using $p'_{\mathcal{ACD}}$ is indeed a correct parity automaton using priorities in $[1, d+1]$.

To prove that no DPA recognising L uses less than d priorities, it suffices to use the Flower Lemma I.21 and the fact that a branch of length d in a tree of the ACD induces a d-flower in $\mathcal{A}$, which is positive if and only if the corresponding tree is positive (Lemma II.83). ◀

Corollary II.116.

If $L \subseteq \Sigma^\omega$ is an ω-regular language of parity index Weak_d, then L can be recognised by a deterministic Weak_d automaton.

Proposition II.117 (Minimal number of colours coincides with parity index)**.**

Let $L \subseteq \Sigma^\omega$ be an ω-regular language of parity index at least $[0, d-1]$ (resp. $[1, d]$). Any history-deterministic Muller automaton recognising L uses an acceptance condition with at least d different output colours.

Proof. We first prove the result for deterministic automata. Let $\mathcal{A}$ be a DMA recognising L using the acceptance condition $(\mathsf{col}, \Gamma, \mathsf{Muller}_\Gamma(\mathcal{F}))$. By Proposition II.115, there is a tree $\mathsf{AltTree}(\ell_i)$ in the ACD of $\mathcal{A}$ of height at least d. We define $\gamma_{\mathcal{ACD}} \colon \mathsf{Nodes}(\mathcal{ACD}_{\mathcal{TS}}) \to \Gamma$ to be the function that assigns to each node of the ACD the colours appearing in it, that is: $\gamma_{\mathcal{ACD}}(n) = \mathsf{col}(\nu(n))$. We remark that if n' is a descendant of n then $\gamma_{\mathcal{ACD}}(n') \subseteq \gamma_{\mathcal{ACD}}(n)$, and that a node n is round if and only if $\gamma_{\mathcal{ACD}}(n) \in \mathcal{F}$. Therefore, by the alternation of round and square nodes, if n' is a strict descendent of n, $\gamma_{\mathcal{ACD}}(n') \subsetneq \gamma_{\mathcal{ACD}}(n)$. We conclude that the root of $\mathsf{AltTree}(\ell_i)$ must contain at least d different colours.

In order to obtain the result for history-deterministic automata we use finite memory resolvers, as defined in Section I.2.2. If $\mathcal{A}$ is a history-deterministic Muller automaton, it admits a sound resolver implemented by a finite memory structure $(\mathcal{M}, \sigma)$ (Lemma I.10). Then, the composition $\mathcal{A} \lhd_\sigma \mathcal{M}$ is a DMA using the same number of colours, that has to be at least d. ◀

The following result (which was already known, as it is a consequence of the construction by Carton and Maceiras [CM99]) states that if a deterministic parity automaton $\mathcal{A}$ recognises a language of parity index $[0, d-1]$, we can obtain an equivalent automaton with an acceptance condition using d colours over the same underlying graph of $\mathcal{A}$. This result can be rephrased using the normal form of parity automata (see Corollary II.136).

Proposition II.118.

Let $\mathcal{A}$ be a deterministic parity automaton such that all its states are accessible and the parity index of $\mathcal{L}(\mathcal{A})$ is $[0, d-1]$ (resp. $[1, d]$ / Weak_d). Then, $\mathcal{A}$ is $[0, d-1]$ (resp. $[1, d]$ / Weak_d)-parity type.

Proof. We suppose that $\mathcal{L}(\mathcal{A})$ has parity index $[0, d-1]$ (the other cases are similar). By the Flower Lemma I.21, $\mathcal{A}$ does not contain any negative d-flower. By Proposition II.115, the trees of the ACD of $\mathcal{A}$ have height at most d, and trees of height d are positive. That is, $\mathcal{ACD}_{\mathcal{A}}$ is a $[0, d-1]$-parity ACD and we conclude by applying Proposition II.111. ◀

The previous result does not hold for history-deterministic automata, as we could artificially add transitions augmenting the complexity of the structure of the automaton (enlarging the flowers of the automaton) without modifying the language it recognises. Nevertheless, some analogous results applying to HD automata can be obtained. Boker, Kupferman and Skrzypczak proved that any HD parity automaton recognising a language of parity index $[0, 1]$ (resp. $[1, 2]$) admits an equivalent HD subautomaton using a Büchi (resp. coBüchi) condition [BKS17, Theorems 10 and 13]. We do not know whether the result holds for languages of arbitrary parity index.

Typeness for deterministic automata

We now lift our results of typeness for transition systems to results about deterministic automata and the languages they recognise. For doing this, we use the fact that two

deterministic automata over the same underlying graph recognise the same language if and only if they are isomorphic (Lemma II.3).

Corollary II.119 (First proven in [BKS10, Theorem 7])**.**

Let $G_{\mathcal{A}}$ be the underlying graph of a deterministic automaton. There are both Rabin and Streett conditions Acc_R and Acc_S such that $\mathcal{L}(G_{\mathcal{A}}, \mathrm{Acc}_R) = \mathcal{L}(G_{\mathcal{A}}, \mathrm{Acc}_S)$, if and only if there is a parity condition Acc_p such that $\mathcal{L}(G_{\mathcal{A}}, \mathrm{Acc}_p) = \mathcal{L}(G_{\mathcal{A}}, \mathrm{Acc}_R) = \mathcal{L}(G_{\mathcal{A}}, \mathrm{Acc}_S)$.

We remark that the hypothesis of determinism in the previous corollary is necessary, as it has been shown that an analogous result does not hold for non-deterministic automata [BKS10].

The following result generalises the fact that if an automaton admits equivalent Büchi and coBüchi conditions on top of it, then it is Weak_1 (this easily follows from [Wag79] and was explicitly discussed in [MP95, p.319]).

Corollary II.120.

Let $G_{\mathcal{A}}$ be the underlying graph of a deterministic automaton. There are $[0, d-1]$ and $[1, d]$-parity conditions $\mathrm{Acc}_{p,0}$ and $\mathrm{Acc}_{p,1}$ such that $\mathcal{L}(G_{\mathcal{A}}, \mathrm{Acc}_{p,0}) = \mathcal{L}(G_{\mathcal{A}}, \mathrm{Acc}_{p,1})$ if and only if there is a Weak_p condition Acc_W such that $\mathcal{L}(G_{\mathcal{A}}, \mathrm{Acc}_W) = \mathcal{L}(G_{\mathcal{A}}, \mathrm{Acc}_{p,0}) = \mathcal{L}(G_{\mathcal{A}}, \mathrm{Acc}_{p,1})$.

Proposition II.121 (First proven in [KPB94, Theorem 15])**.**

Let $\mathcal{A}$ be a deterministic Rabin (resp. Streett) automaton, and suppose that $\mathcal{L}(\mathcal{A})$ has parity index at most $[0, 1]$ (resp. at most $[1, 2]$). Then, $\mathcal{A}$ is Büchi type (resp. coBüchi type).

Proof. We do the proof for the case Rabin-Büchi. We can suppose that all the states of $\mathcal{A}$ are accessible, as we can define a trivial acceptance condition in the part of $\mathcal{A}$ that is not accessible. Since $\mathcal{L}(\mathcal{A})$ has parity index at most $[0, 1]$, the trees of the ACD of $\mathcal{A}$ have height at most 2, and trees of height 2 are positive (the root is a round node), by Proposition II.115. As $\mathcal{A}$ is a Rabin automaton, its ACD has Rabin shape (Proposition II.109), so round nodes have at most one child. We conclude that the trees of the ACD of $\mathcal{A}$ have a single branch, so it is a $[0, 1]$-parity ACD, and by Proposition II.111, $\mathcal{A}$ is Büchi type. ◀

Proposition II.122.

Let $\mathcal{A}$ be a deterministic Muller automaton, and suppose that $\mathcal{L}(\mathcal{A})$ has parity index at most $[0, 1]$ (resp. at most $[1, 2]$). Then, $\mathcal{A}$ is generalised Büchi type (resp. generalised coBüchi type).

Proof. We prove the result for the case generalised Büchi (analogous for coBüchi). We can suppose that all the states of $\mathcal{A}$ are accessible, as we can define a trivial acceptance condition in the part of $\mathcal{A}$ that is not accessible. Since $\mathcal{L}(\mathcal{A})$ has parity index at most $[0, 1]$, the trees of the ACD of $\mathcal{A}$ have height at most 2, and trees of height 2 are positive

(the root is a round node), by Proposition II.115, so it is a generalised Büchi ACD, and by Proposition II.114, $\mathcal{A}$ is generalised Büchi type. ◀

Deciding typeness in polynomial time

Given a Muller transition system $\mathcal{TS}$, we use the expression *decide the typeness* of $\mathcal{TS}$ to refer to the family of problems consisting of deciding whether $\mathcal{TS}$ is X type for X one of the following: Rabin, Streett, parity, $[d_{\min}, d_{\max}]$-parity, Weak_d, generalised Büchi, generalised coBüchi.

Next theorem follows directly from Theorem II.9 and the results of this section.

Theorem II.10 (Deciding typeness in polynomial time)**.**

Given a Muller transition system $\mathcal{TS}$ with its acceptance condition represented explicitly, as a Zielonka tree, or as a Zielonka DAG, we can decide its typeness in polynomial time.

We remark that this implies that we can also decide the typeness of parity and generalised (co)Büchi transition systems in polynomial time (c.f. Figure 6).

Combining Theorem II.9 with Proposition II.115, we obtain that we can decide the parity index of the language recognised by a DMA in polynomial time.

Theorem II.11 (Deciding the parity index in polynomial time)**.**

Given a deterministic Muller automaton with its condition represented explicitly, as a Zielonka tree, or as a Zielonka DAG, we can decide the parity index of $\mathcal{L}(\mathcal{A})$ in polynomial time.

However, it is known that, given a deterministic Rabin (or Streett) automaton with the condition represented as a family of Rabin pairs, the problem of determining the parity index of $\mathcal{L}(\mathcal{A})$ is NP-hard [KPB95].[14,15]

6.3 Minimisation of the number of colours used by Muller conditions

We consider the problem of minimising the number of colours of a Muller transition system. That is, given $\mathcal{TS}$ using a Muller condition, what is the minimal number of colours needed to define an equivalent Muller condition on top of $\mathcal{TS}$? We first study this question for Muller languages, without taking into account the structure of the transition system. We show that given the Zielonka tree of the condition, we can minimise its number of colours in polynomial time (Theorem II.12). Then, we attack the question taking into account the structure of the transition system. Surprisingly, we show that in this case the problem is NP-hard, even if the ACD is given as input (Theorem II.13).

This subject has recently been studied by Schwarzová, Strejček and Major [SSM23]. In their approach, they use heuristics to reduce the number of colours by applying QBF-solvers. The final transition system is not guaranteed to have a minimal number of colours, though.

[14] Formally, there are two different decision problems corresponding to this question: deciding if the parity index of $\mathcal{L}(\mathcal{A})$ is *at least* or *at most* $[0, k]$. For the first version, the decision problem is NP-complete, for the second one, it is coNP-complete.

[15] This reduction was shown for state-based automata. It is easy to see that, in this case, the problem for state-based automata reduces to the problem for transition-based ones.

We also present the problem of minimising the number of Rabin pairs of a Rabin language, or of a Rabin condition on top of some fixed transition system. However, we have not been able to provide conclusive results in this case.[16]

6.3.1 Minimisation of colours to define Muller languages

We say that a Muller language $L \subseteq \Sigma^\omega$ is *k-colour type* if there is a set of k colours Γ, a Muller language $L' \subseteq \Gamma^\omega$ and a mapping $\phi\colon \Sigma \to \Gamma$ such that for all $w \in \Sigma^\omega$, $w \in L \iff \phi(w) \in L'$.

✦ Remark II.123. *A Muller language $L \subseteq \Sigma^\omega$ is k-colour type if and only if it can be recognised by a deterministic Muller automaton with one state using k output colours.*

We say that a Rabin language $L \subseteq \Sigma^\omega$ is *k-Rabin-pair type* if there is a family of k Rabin pairs R over some set of colours Γ and a mapping $\phi\colon \Sigma \to \Gamma$ such that for all $w \in \Sigma^\omega$, $w \in L \iff \phi(w) \in \mathsf{Rabin}_\Gamma(R)$.

✦ Remark II.124. *A Rabin language $L \subseteq \Sigma^\omega$ is k-Rabin-pair type if and only if it can be recognised by a deterministic Rabin automaton with one state using k Rabin pairs.*

Problem:	Colour-Minimisation-ML
Input:	A Muller language $\mathsf{Muller}_\Sigma(\mathcal{F})$ represented by the Zielonka DAG $\mathcal{Z}\text{-}\mathsf{DAG}_\mathcal{F}$ and a positive integer k.
Question:	Is $\mathsf{Muller}_\Sigma(\mathcal{F})$ k-colour type?

We could have chosen other representations of $\mathsf{Muller}_\Sigma(\mathcal{F})$ for the input of this problem (mainly, explicitly or as a Zielonka tree). We have chosen to specify the input as a Zielonka DAG, as it is more succinct than the other representations (c.f. Figure 19 and Propositions B.31, B.32). In Theorem II.12, we show that this problem can be solved in polynomial time if $\mathcal{F}$ is represented as a Zielonka DAG, which implies that it can be equally solved in polynomial time if $\mathcal{F}$ is represented explicitly or as a Zielonka tree.

Theorem II.12 (Minimisation of colours for Muller languages)**.**

The problem Colour-Minimisation-ML can be solved in polynomial time.

Proof. We say that two letters $a, b \in \Sigma$ are *$\mathcal{F}$-equivalent*, written $a \sim_\mathcal{F} b$, if, for all $C \subseteq \Sigma$:

$$C \cup \{a\} \in \mathcal{F} \iff C \cup \{b\} \in \mathcal{F} \iff C \cup \{a, b\} \in \mathcal{F}.$$

It is immediate to check that $\sim_\mathcal{F}$ is indeed an equivalence relation. We let $[a]$ denote the equivalence class of a for $\sim_\mathcal{F}$, $\Sigma/_{\sim_\mathcal{F}}$ the set of equivalence classes, and for $C \subseteq \Sigma$, we write $\pi(C) = \{[a] \mid a \in C\}$.

✧ Claim II.124.1. *For all $C \subseteq \Sigma$, $C \in \mathcal{F} \iff \bigcup_{a \in C}[a] \in \mathcal{F}$.*

Proof. For each $a \in C$, we can add all the elements in $[a]$ one by one to C without changing the acceptance status. ◁

[16] Corto Mascle has recently solved the first of these problems (unpublished). See also footnote 17.

✧ Claim II.124.2. $\mathsf{Muller}_\Sigma(\mathcal{F})$ *is* k*-colour type if and only if there are at most* k *classes for the* $\mathcal{F}$*-equivalence relation.*

Proof. Let $\mathcal{A}$ be a DMA with one state q and using as acceptance condition $\mathsf{Muller}_\Gamma(\mathcal{H})$. This defines a function $f\colon \Sigma \to \Gamma$ that sends a to α if α is the colour produced when reading a, that is, if $q \xrightarrow{a:\alpha} q$ is the corresponding transition in $\mathcal{A}$. It is immediate that if $f(a) = f(b)$, then $a \sim_{\mathcal{F}} b$, so $|\Gamma|$ is at least the number of equivalence classes.

Conversely, we can define a DMA using as output colours the equivalence classes: $q \xrightarrow{a:[a]} q$, using as acceptance set the language associated to:

$$\widetilde{\mathcal{F}} = \{\pi(C) \mid C \in \mathcal{F}\}.$$

The fact that the obtained automaton recognises $\mathsf{Muller}_\Sigma(\mathcal{F})$ follows from Claim II.124.1. ◁

✧ Claim II.124.3. *Two letters* $a, b \in \Sigma$ *are* $\mathcal{F}$*-equivalent if and only for every node* n *of the Zielonka DAG of* $\mathcal{F}$*,* $a \in \nu(n) \iff b \in \nu(n)$*.*

Proof. Suppose that there is a node n in the Zielonka DAG such that $a \in \nu(n)$ and $b \notin \nu(n)$. Take a minimal node with this property, and let n' be its parent. Then, by minimality, $a \in \nu(n)$, $b \notin \nu(n)$, but $a, b \in \nu(n')$. By Remark II.27, we obtain that $\nu(n) = \nu(n) \cup \{a\} \in \mathcal{F} \iff \nu(n) \cup \{a, b\} \notin \mathcal{F}$, so a and b are not $\mathcal{F}$-equivalent.

For the other direction, let $C \subseteq \Sigma$ and take a minimal node n such that $C \cup \{a\} \subseteq \nu(n)$. Then, $b \in \nu(n)$, so n is also minimal such that $C \cup \{a, b\} \subseteq \nu(n)$, and therefore $C \cup \{a\} \in \mathcal{F} \iff C \cup \{a, b\} \in \mathcal{F}$. The third equivalence is obtained by symmetry. ◁

Claim II.124.2 tells us that in order to minimise the number of required colours, we need to compute the classes of the $\mathcal{F}$-equivalence relation. This can be directly done by inspecting the Zielonka DAG by Claim II.124.3. ◀

We present the problem of the minimisation of Rabin pairs for Rabin languages.

Problem:	Rabin-Pair-Minimisation-ML
Input:	A family of Rabin pairs R over Σ and a positive integer k.
Question:	Is $\mathsf{Rabin}_\Sigma(R)$ k-Rabin-pair type?

Conjecture II.1 (Minimisation of Rabin pairs for Rabin languages)**.**

The problem Rabin-Pair-Minimisation-ML can be solved in polynomial time.[17]

[17] This result has recently been proven by Corto Mascle. He introduces a form of *generalised Horn formulas*, and gives a set of rules that allow to minimise the number of their clauses. The problem Rabin-Pair-Minimisation-ML reduces to the minimisation of clauses in these generalised Horn formulas. As this result is still unpublished, we state it as a conjecture here.

6.3.2 Minimisation of colours in Muller transition systems

We say that a Muller transition system $\mathcal{TS}$ is *k-colour type* if there is an equivalent Muller condition over the underlying graph of $\mathcal{TS}$ using at most k output colours. We say that a Muller transition system $\mathcal{TS}$ is *k-Rabin-pair type* if there is an equivalent Rabin condition over the underlying graph of $\mathcal{TS}$ using at most k Rabin pairs.

We consider the problem of minimising the number of colours used by a Muller condition over a fixed transition system.

Problem: Colour-Minimisation-TS
Input: A Muller transition system $\mathcal{TS}$ and a positive integer k.
Question: Is $\mathcal{TS}$ k-colour type?

We remark that we have not specified the representation of the acceptance condition of $\mathcal{TS}$, therefore, this problem admits different variants according to this representation. We will show that for the three representations that we use in this section (explicit, Zielonka tree and Zielonka DAG), the problem Colour-Minimisation-TS is NP-hard. This implies that the problem is NP-hard even if the ACD is provided as input.

Hugenroth showed[18] that, *for state-based automata*, the problem Colour-Minimisation-TS is NP-hard when the acceptance condition of $\mathcal{TS}$ is represented explicitly or as a Zielonka tree [Hug23]. However, we have not been able to generalise his reduction to *transition-based automata*. Nevertheless, we show now that both problems above are NP-hard (in the transition-based setting) by giving a different reduction. Moreover, our reduction of NP-hardness uses a transition system with only 2 disconnected states. This is quite surprising, as we could think that a generalisation of the methods from the proof of Theorem II.12 to the ACD would yield a polynomial time algorithm. However, it is not possible to properly combine the structure of the local subtrees of the ACD.

Theorem II.13 (Minimisation of colours for Muller transition systems)**.**

The problem Colour-Minimisation-TS is NP-hard, if the acceptance condition $\mathsf{Muller}_\Gamma(\mathcal{F})$ of $\mathcal{TS}$ is represented explicitly, as a Zielonka tree, as a Zielonka DAG or as the ACD of $\mathcal{TS}$.

Proof. We prove the result for the representation of the condition as an explicit list of subsets and as the Zielonka tree. The result for the other representations follows then from Theorem II.8.

We give a reduction from the problem Max-Clique. Let $G = (V, E)$ be a simple, connected undirected graph and $k \in \mathbb{N}$, $k < |V|$. We consider the transition system $\mathcal{TS}_{G,k}$ defined as:

- It has two states q_{vert} and q_k, both of them initial.
- The set of colours used by its acceptance condition is $\Gamma = V \cup A_k$, where A_k is a set of size k disjoint from V.

[18] As of today, the proof is not currently publicly available online, we got access to it by a personal communication. The statement of the theorem only express the NP-hardness for the explicit representation, but a look into the reduction works unchanged if the condition is given as a Zielonka tree.

- The set of edges is $V \cup A_k$. $\mathcal{TS}_{G,k}$ has self loops $q_{\mathsf{vert}} \xrightarrow{v} q_{\mathsf{vert}}$ for every $v \in V$ and $q_k \xrightarrow{a} q_k$ for every $a \in A_k$.
- Its acceptance set is the Muller language associated to the family:

$$\mathcal{F} = \{\{v, u\} \mid (v, u) \in E\} \cup \{\{a, a'\} \mid a, a' \in A_k,\ a \neq a'\}.$$

The representation of this transition system is polynomial in $|G| + k$, since $|\mathcal{F}| = \mathcal{O}(|E| + k^2)$. We also note that the Zielonka tree of $\mathcal{F}$ has size also $\mathcal{O}(|E| + k^2)$.

✧ Claim II.124.4. *If G admits a clique of size k, then $\mathcal{TS}_{G,k}$ is $|V|$-colour type.*

Proof. Let $V' = \{v'_1, \dots, v'_k\}$ be a clique of size k of G, and let $A_k = \{a_1, \dots, a_k\}$. We consider the Muller condition using as set of colours $\Gamma' = V$ given by the family $\mathcal{F}' = \{\{v, u\} \mid (v, u) \in E\}$. The new acceptance condition over $\mathcal{TS}_{G,k}$ is obtained by using the same colouring for the self loops over q_{vert}, and recolouring self loops $q_k \xrightarrow{a_i} q_k$ with $q_k \xrightarrow{v'_i} q_k$. It is immediate that the obtained acceptance condition is equivalent to the original one of $\mathcal{TS}_{G,k}$. ◁

In order to show the converse, we will use the following properties satisfied by $\mathcal{TS}_{G,k}$:

- Cycles consisting in a single self loop are rejecting.
- Cycles $\{a, a'\}$, with $a, a' \in A_k$, $a \neq a'$ are accepting.

✧ Claim II.124.5. *If $\mathcal{TS}_{G,k}$ is $|V|$-colour type, then G admits a clique of size k.*

Proof. If $\mathcal{TS}_{G,k}$ is $|V|$-colour type, then there is a set Γ' of $|V|$ colours and a colouring function $\mathsf{col}' \colon V \cup A_k \to \Gamma'$.

First, we show that for two different self loops over q_{vert}, named $v, v' \in V$, $\mathsf{col}'(v) \neq \mathsf{col}'(v')$. If $(v, v') \in E$, this is clear, as $\{v\}$ is a rejecting cycle, but $\{v, v'\}$ is accepting. Suppose that $(v, v') \notin E$, and let $u \in V$ such that $(v, u) \in E$ (which exists as G is connected). Then, the cycle $\{v, u\}$ is accepting and $\{v, u, v'\}$ is rejecting, so they cannot be coloured equally. Therefore, for each colour $c \in \Gamma'$ there is one self loop v such that $\mathsf{col}'(v) = c$.

Secondly, we remark that for two different self loops a, a' over q_k it is also satisfied $\mathsf{col}'(a) \neq \mathsf{col}'(a')$. On the contrary, we would have $\mathsf{col}'(\{a, a'\}) = \mathsf{col}'(\{a\})$, a contradiction. Let $\{c_1, \dots, c_k\} = \mathsf{col}'^{-1}(A_k)$ be the k different colours labelling the self loops over q_k. We obtain that the subset $\{v_1, \dots, v_k\} \subseteq V$ of vertices such $\mathsf{col}'(v_i) = c_i$ form a clique of size k in G. ◁

◀

Similarly, we consider the problem of minimising the number Rabin pairs over a fixed Rabin transition system.

Problem:	Rabin-Pair-Minimisation-TS
Input:	A Rabin transition system $\mathcal{TS}$ and a positive integer k.
Question:	Is $\mathcal{TS}$ k-Rabin-pair type?

Conjecture II.2 (Minimisation of Rabin pairs over a transition system)**.**

The problem Rabin-Pair-Minimisation-TS is NP-complete.

7 A normal form for parity automata

In this section, we propose a definition of a normal form for parity automata. This is exactly the form of automata resulting by applying (an adaptation to transition-based automata of) the procedure defined by Carton and Maceiras [CM99], or, equivalently, of automata resulting from the ACD-parity-transform (Proposition II.131). These automata satisfy that they are parity-index-tight, that is, their acceptance condition use the minimal possible number of colours. But they offer some further convenient properties, stated in Propositions II.134 and II.135, which make them particularly well-suited for reasoning about deterministic parity automata. The suitability of this normal form for theoretical purposes will be evident in Chapter V, where we rely on their properties in most of our arguments.

This normal form, or partial versions of it, have already been used in the literature to prove results about parity automata in different contexts, such as history-deterministic coBüchi automata [KS15, AK22, ES22], positionality of languages defined by deterministic Büchi automata [Bou+22] or learning of DPAs [BL23a]. This normal form also facilitates the resolution of parity games in practice [FL09]. However, the application of this normal form in the literature is limited to specific cases, and no prior works have provided a formal and systematic study of it.

From our results we obtain three equivalent ways of defining the normal form of a parity transition system $\mathcal{TS}$. Informally, they can be stated as:

1. Transitions of $\mathcal{TS}$ use the smallest possible priorities (Definition II.126).
2. $\mathcal{TS} = \mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ (Proposition II.131).
3. Paths in $\mathcal{TS}$ producing priority d can be closed into a cycle producing d' as minimal priority, for all $d' \leq d$ (Theorem II.14).

This last property is particularly useful for proving that a given parity automaton is in normal form (this will be the standard procedure to establish normality in the proofs from Section V.4.2).

Before formally defining this normal form, we remark that we can assume that parity transition system use 0 or 1 as minimal priority.

✦ Remark II.125. *We remark that if* $\mathsf{Acc} = (\mathsf{col}, [d_{\min}, d_{\max}], \mathsf{parity})$ *is the parity acceptance condition of a transition system* $\mathcal{TS}$*, we can always assume that* $d_{\min}$ *is* 0 *or* 1*. Indeed, define* $\chi = d_{\min}$ *if* $d_{\min}$ *is even, and* $\chi = d_{\min} - 1$ *if* $d_{\min}$ *is odd. The parity acceptance condition* $(\mathsf{col}', [d_{\min} - \chi, d_{\max} - \chi], \mathsf{parity})$ *defined as* $\mathsf{col}'(e) = \mathsf{col}(e) - \chi$ *is equivalent to* Acc *over* $\mathcal{TS}$*.*

7.1 Definition of the normal form

Just as in the definition of the ACD-parity-transform we had to define positive and negative ACDs to obtain an accurate optimality result in the number of priorities, we need now to take care of a small technical detail so that TS in normal form are parity-

index-tight.

We say that a transition system $\mathcal{TS}$ is *negative* if $\mathcal{ACD}_{\mathcal{TS}}$ is negative, that is, if for some d $\mathcal{TS}$ contains a negative d-flower but contains no positive d-flower. Intuitively, a parity TS is negative if and only if the minimal priority used by a parity acceptance condition using an optimal number of priorities is 1.

Definition II.126 (Normal form)**.**

Let $\mathcal{TS} = (G_{\mathcal{TS}}, \mathrm{Acc}_{\mathcal{TS}})$ be a parity transition system using a colouring function col. If $\mathcal{TS}$ is not negative, we say that $\mathcal{TS}$ is in *normal form* if any other parity acceptance condition equivalent to $\mathrm{Acc}_{\mathcal{TS}}$ over $G_{\mathcal{TS}}$ using a colouring function col' satisfies that for every edge e:

$$\mathsf{col}(e) \leq \mathsf{col}'(e).$$

If $\mathcal{TS}$ is negative, we say that it is in *normal form* if any other equivalent colouring col' not using priority 0 satisfies that for any edge e:

$$1 \leq \mathsf{col}(e) \leq \mathsf{col}'(e).$$

If $\mathcal{TS}$ is in normal form, we will also say that its acceptance condition or the colouring function it uses are in normal form.

Example II.127.

Parity transition systems from Figures 5, 13, 17 and 18 are all in normal form. Parity automata appearing in Figures 13 and 17 are negative (the minimal priority used by an optimal acceptance condition is odd), whereas parity automata in Figure 18 are not.

On the other hand, the automaton from Figure 4 is not in normal form (even if it is parity-index-tight). We can put it in normal form by assigning priority 1 to transitions $q_1 \xrightarrow{a,b} q_0$ and $q_1 \xrightarrow{b,c} q_2$. The automaton obtained in this way recognises the same language.

Proposition II.128.

Let $\mathcal{TS} = (G_{\mathcal{TS}}, \mathrm{Acc}_{\mathcal{TS}})$ be a parity transition system with a colouring function col. There is a unique parity acceptance condition equivalent to $\mathrm{Acc}_{\mathcal{TS}}$ over $G_{\mathcal{TS}}$ in normal form. Moreover, this acceptance condition is exactly the parity condition of the ACD-parity-transform of $\mathcal{TS}$.

Before showing the proof of Proposition II.128, we prove a useful technical lemma.

✦ **Lemma II.129.** *Let $\mathcal{TS}$ be a parity transition system with colouring function col. If $\ell_1 \supsetneq \ell_2 \supsetneq \cdots \supsetneq \ell_k$ is a positive (resp. negative) k-flower of $\mathcal{TS}$, then $\min \mathsf{col}(\ell_k) \geq k-1$ (resp. $\min \mathsf{col}(\ell_k) \geq k$).*

Proof. We show the result for negative flowers. Let $d_i = \min \mathsf{col}(\ell_i)$. We show that $d_i \geq i$ by induction. Since ℓ_i is an accepting cycle if and only if i is even, we have that d_i is even if and only if i is even. Clearly, $d_1 \geq 1$, as 1 is the least odd number. Also, $d_{i+1} \geq d_i$, since $\ell_{i+1} \subseteq \ell_i$, and the inequality is strict by the alternation of the parity, concluding the proof. ◀

Proof of Proposition II.128. We first remark that the uniqueness is directly implied by the definition of normal form.

We prove that the acceptance condition of the ACD-parity-transform is in normal form. We note its colouring function by $\mathsf{col}_{\mathcal{ACD}}$. The transitions not belonging to any SCC are coloured 0 if $\mathcal{TS}$ is not negative and 1 if $\mathcal{TS}$ is negative, as desired. It suffices to prove the result for edges in SCCs.

We assume that $\mathcal{TS}$ is not negative and we let $\mathcal{S}$ be an accepting SCC of $\mathcal{TS}$ (the proof is similar for $\mathcal{TS}$ negative and a rejecting SCC). Let $e = v \to v'$ be an edge in $\mathcal{S}$, and let $\mathcal{T}_v$ be the local subtree at v, which is composed of a single branch (see Proposition II.111). We let $n_0 \preceq n_1 \preceq \ldots \preceq n_r$ be that branch, where n_0 is the root and n_r the leaf . Let n_k be the deepest node of $\mathcal{T}_v$ such that $e \in \nu(n_k)$. By definition of the ACD-parity-transform, $\mathsf{col}_{\mathcal{ACD}}(e) = p_{\mathcal{ACD}}(e) = k$. Also, $\nu(n_0) \preceq \nu(n_1) \preceq \ldots \preceq \nu(n_k)$ is a positive $k+1$-flower (by Lemma II.83). Lemma II.129 implies then that any equivalent parity condition using a colouring function col' verifies $\mathsf{col}'(e) \geq \mathsf{col}_{\mathcal{ACD}}(e) = k$. ◀

We deduce the following result, that might clarify the notion of negative transition systems.

✦ Remark II.130. *A parity transition system $\mathcal{TS}$ is negative if and only if the minimal priority used by its normal form is* 1.

Corollary II.131.

The ACD-parity-transform $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ of any Muller transition system $\mathcal{TS}$ is in normal form.

We obtain as a corollary of Proposition II.97 that we can put a parity TS in normal form in polynomial time.

Proposition II.132 (Effective normalisation of parity transition systems)**.**

Given a parity transition system $\mathcal{TS}$, we can construct in polynomial time an equivalent parity condition over its underlying graph such that the obtained transition system is in normal form.

7.2 Properties of the normal form

Parity-index-tightness

We say that a parity transition system $\mathcal{TS} = (G_{\mathcal{TS}}, \mathrm{Acc}_{\mathcal{TS}})$ is *parity-index-tight* if any other parity condition Acc' over $G_{\mathcal{TS}}$ such that $\mathrm{Acc}' \simeq_{G_{\mathcal{TS}}} \mathrm{Acc}_{\mathcal{TS}}$ uses at least as many priorities as $\mathrm{Acc}_{\mathcal{TS}}$.

By Proposition II.128, the optimality of the colouring of $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ (Theorem II.5) transfers to parity transition systems in normal form.

Corollary II.133.

A parity transition system in normal form is parity-index-tight.

Fundamental properties of the normal form

We now state what we consider to be the two fundamental properties of parity transition systems in normal form (Propositions II.134 and II.135), which characterise them (Theorem II.14).

Proposition II.134 (Closing paths with large priorities)**.**

Let $\mathcal{TS}$ be a parity transition system in normal form. If there is a path $v \rightsquigarrow v'$ producing d as minimal priority, then, either:

- v and v' are in different SCCs (and in this case $d \in \{0, 1\}$), or
- there is a path $v' \rightsquigarrow v$ producing no priority strictly smaller than d.

Proof. By Proposition II.128, we know that the colouring of $\mathcal{TS}$ is the one given by its ACD-transform, that we note $\mathsf{col}_{\mathcal{ACD}}$. If v and v' are in different SCCs the result is trivial. Let v and v' be in the same SCC, that we assume to be an accepting SCC without loss of generality. Let $\rho = v \xrightarrow{e_1} \dots \xrightarrow{e_k} v'$ be a path from v to v' producing $\min \mathsf{col}_{\mathcal{ACD}}(\rho) = d$ as minimal priority. We remark that, as $\mathcal{ACD}_{\mathcal{TS}}$ is a parity ACD, each edge e appears in one and only one branch of $\mathcal{ACD}_{\mathcal{TS}}$, and that $\mathsf{col}_{\mathcal{ACD}}(e)$ equals the depth of the deepest node containing e. In particular, if $e \in \nu(n)$ for some node e, $\mathsf{col}_{\mathcal{ACD}}(e) \geq \mathsf{Depth}(n)$. Our objective is to show that a similar result holds for the path ρ as a set of edges:

✧ Claim II.134.1. *Let N_ρ be the set of nodes of $\mathcal{ACD}_{\mathcal{TS}}$ containing the edges of the path ρ in their label, that is, $N_\rho = \{n \in \mathsf{Nodes}(\mathcal{ACD}_{\mathcal{TS}}) \mid \{e_1, \dots, e_k\} \subseteq \nu(n)\}$. Then, $\min(\mathsf{col}_{\mathcal{ACD}}(\rho))$ equals the depth of a node of maximal depth of N_ρ.*[19]

This claim allows us to conclude. Indeed, let n be a node of maximal depth of N_ρ, verifying $\mathsf{Depth}(n) = d$. Then, $\nu(n)$ is a cycle containing the vertices v and v', and for all the edges $e \in \nu(n)$, $\mathsf{col}_{\mathcal{ACD}}(e) \geq \mathsf{Depth}(n) = d$. This provides the desired path from v' to v.

Proof of Claim II.134.1. First, we remark that if $\ell_1, \ell_2, \dots, \ell_k$ are cycles such that ℓ_i and ℓ_{i+1} have some state in common, then $\cup_{i=1}^{k} \ell_i$ is a cycle. Let n be a node of maximal depth in N_ρ. By the previous remarks, $\mathsf{col}_{\mathcal{ACD}}(e) \geq \mathsf{Depth}(n)$. Suppose by contradiction that $\mathsf{col}_{\mathcal{ACD}}(\rho) > \mathsf{Depth}(n)$. Then, each edge e_i of ρ would appear in some strict descendant n_i of n (we can suppose that n_i is a child of n). Then, $\nu(n_1), \dots, \nu(n_k)$ would be cycles such that $\nu(n_i)$ and $\nu(n_{i+1})$ have some state in common (namely, $\mathsf{target}(e_i) = \mathsf{source}(e_{i+1})$), so their union is a cycle. However, this is not possible in a parity transition system, as $\nu(n)$ is accepting if and only if each of the $\nu(n_i)$ is rejecting (see Lemma II.39). ◁

◀

Proposition II.135 (Normal flowers do not lack petals)**.**

Let v be a state of a parity transition system in normal form belonging to an accepting (resp. rejecting) SCC. Let $\ell \in \mathcal{C}ycles_v(\mathcal{TS})$ be a cycle over v and let d_ℓ be the minimal priority appearing on it.

[19] In fact, the nodes of N_ρ are totally ordered by the ancestor relation, so there is a unique node of maximal depth in N_ρ. This fact is not used in our proof.

- If $\mathcal{TS}$ is not negative, for each $x \in [0, d_\ell]$ (resp. $x \in [1, d_\ell]$) there is a cycle $\ell_x \in \mathit{Cycles}_v(\mathcal{TS})$ producing x as minimal priority.
- If $\mathcal{TS}$ is negative, for each $x \in [2, d_\ell]$ (resp. $x \in [1, d_\ell]$) there is a cycle $\ell_x \in \mathit{Cycles}_v(\mathcal{TS})$ producing x as minimal priority.

Proof. We do the proof for the case in which $\mathcal{TS}$ is not negative and v belongs to an accepting SCC. By Proposition II.128, the colouring of $\mathcal{TS}$ is the one given by its ACD-transform, noted $\mathsf{col}_{\mathcal{ACD}}$. Consider the local subtree at v, $\mathcal{T}_v$, consisting in a single branch, as it has parity shape (Proposition II.111). Let $n_0 \preceq \ldots \preceq n_k$ be that branch, and let n_i be the deepest node such that $\ell \subseteq \nu(n_i)$. We remark that, by definition of $\mathsf{col}_{\mathcal{ACD}}$, $d_\ell = \mathsf{Depth}(n_i) = i$. The desired cycles are obtained by taking $\ell_x = \nu(n_x)$, for $x \in [0, d_\ell]$. ◀

Next theorem provides a simple characterisation that is a useful tool to show that a parity transition system is in normal form in many proofs. In essence, it shows that the two previous propositions characterise the normal form. We state it for non-negative transition systems for simplicity; a similar characterisation for negative transition systems is immediate.

We say that a SCC of a parity TS is *positive* if the minimal priority appearing on it is even, and that it is *negative* if this minimal priority is odd.

Theorem II.14.

A non-negative parity transition system $\mathcal{TS}$ is in normal form if and only if:

- transitions changing of SCCs are coloured 0, and
- if v and v' are in a same positive (resp. negative) SCC and there is a transition $v \xrightarrow{d} v'$ producing priority $d > 0$ (resp. $d > 1$), then there are two paths $v' \rightsquigarrow v$ producing as minimal priority d and $d-1$, respectively.

Proof. The fact that a TS in normal form satisfies these properties follows from the previous propositions.

Let $\mathcal{TS}$ be a TS satisfying these properties and using col as colouring function. Let $e = v \xrightarrow{d} v'$ be an edge with $\mathsf{col}(e) = d$. We will show that for any other equivalent colouring col', we have $\mathsf{col}'(e) \geq d$. This is trivial if $d = 0$. If $d > 0$, v and v' must be in the same SCC, that we assume positive without loss of generality. By hypothesis, we can close cycles ℓ_d and ℓ_{d-1} over v producing d and $d-1$ as minimal priority, respectively. Cycle ℓ_{d-1} can be decomposed in $v \to v' \rightsquigarrow v_1 \xrightarrow{d-1} v_1' \rightsquigarrow v$. Applying the hypothesis over the edge $v_1 \xrightarrow{d-1} v_1'$ gives a path $v_1' \rightsquigarrow v_1$ producing $d-2$ as minimal priority, which can be merged with ℓ_{d-1} to produce a cycle ℓ_{d-2} over v producing $d-2$ as minimal priority. Iterating this process, we can find cycles $\ell_0 \supseteq \ell_1 \supseteq \ldots, \supseteq \ell_d$ over v such that ℓ_i produces i as minimal priority. Taking $\ell_i' = \cup_{j=i}^{d} \ell_i$, we obtain a positive $(d+1)$-flower $\ell_0' \supsetneq \ell_1' \supsetneq \cdots \supsetneq \ell_d'$, so by Lemma II.129 we conclude that $\mathsf{col}'(e) \geq d$. ◀

Parity index from automata in normal form

We show that, when applied to deterministic parity automata, the normal form directly gives the parity index of the language recognised by the automaton.

Corollary II.136.

Let $\mathcal{A}$ be a deterministic parity automaton in normal form using priorities in $[0, d-1]$ (resp. $[1, d]$) such that all its states are accessible. If $\mathcal{A}$ is Weak_{d-1}, then the parity index of $\mathcal{L}(\mathcal{A})$ is Weak_{d-1}. If not, the parity index of $\mathcal{L}(\mathcal{A})$ is $[0, d-1]$ (resp. $[1, d]$).

Proof. We assume that $\mathcal{A}$ uses priorities in $[0, d-1]$ (in particular, it is not negative, c.f. Remark II.130). The proof for $[1, d]$ is analogous.

If $\mathcal{A}$ is not Weak_{d-1}, there is a SCC containing all the priorities $[0, d-1]$. By Proposition II.135, such SCC contains a positive d-flower, so by the Flower Lemma I.21, the parity index of $\mathcal{L}(\mathcal{A})$ is $[0, d-1]$.

Assume now that $\mathcal{A}$ is Weak_{d-1}. Let ℓ be a cycle of $\mathcal{A}$ in which priority $d-1$ occurs. By Proposition II.135, ℓ contains a negative $(d-1)$-flower. As $\mathcal{A}$ is not negative, it also contains a positive $(d-1)$-flower. By the Flower Lemma I.21, the parity index of $\mathcal{L}(\mathcal{A})$ is Weak_{d-1}. ◀

8 Transformations towards state-based automata

In all the chapter, we have worked with transition systems with the acceptance condition over transitions. In this section, we discuss how the transformations presented in Section II.4 could be adapted for state-based transition systems. Our main result (Theorem II.15) is a negative answer: it is not possible to obtain transformations for state-based transition systems offering the same tight optimality guarantees. Nevertheless, in Section II.8.2 we present a suboptimal method to transform Muller TS into state-based parity TS.

8.1 NP-hardness of finding optimal transformations for state-based transition systems

We have shown that given a Muller transition system $\mathcal{TS}$ and its ACD, we can compute in polynomial time a parity TS minimal amongst those TS admitting a locally bijective morphism towards $\mathcal{TS}$ (Proposition II.97). If the input transition system is a generalised (co)Büchi one, there is no need to explicitly include the ACD in the input, as it can be computed in polynomial time (Corollary II.96). For this reason, and to simplify the statements of the results, we state the contributions of this section for generalised Büchi automata.

Problem:	Transform-GenBuchi
Input:	A generalised Büchi transition system $\mathcal{TS}$ and a positive integer k.
Question:	Is there a Büchi TS of size k admitting a locally bijective morphism to $\mathcal{TS}$?

The following result just restates Proposition II.98.

Proposition II.137.

The problem Transform-GenBuchi can be solved in polynomial time.

We can consider the analogous problem in which we ask to obtain a *state-based* Büchi TS as output.[20] We call the obtained problem Transform-GenBuchi-to-States. We will see that with just this "small" modification, the problem becomes NP-hard.

✦ Remark II.138. *The problem in which also the* input *is given as a state-based TS is easier, since we can go from state-based to transition-based models without modifying the underlying graph.*

We also consider the problem of transforming a transition-based TS into an equivalent one using a state-based acceptance.

Problem:	Transitions-to-States
Input:	A transition-based Büchi TS $\mathcal{TS}$ and a positive integer k.
Question:	Is there a state-based Büchi TS of size k admitting a locally bijective morphism to $\mathcal{TS}$?

✦ Remark II.139. Transitions-to-States *reduces to* Transform-GenBuchi-to-States *in linear time.*

We prove now that these problems are NP-complete. The reduction we provide is almost identical to the one given by Schewe to show the NP-completeness of the minimisation of state-based Büchi automata [Sch10]. We argue that this shows that the reduction proves the difficulty to go from a transition-based model to a state-based model in an optimal way. This eliminates any hope of using a similar reduction to prove the NP-hardness of the minimisation of transition-based Büchi automata.

Theorem II.15.

The problems Transform-GenBuchi-to-States and Transitions-to-States are NP-complete.

Proof. The fact these problems are in NP follows from the fact that we can guess a state-based Büchi TS of size k and a locally bijective morphism to $\mathcal{TS}$. Checking the local bijectivity can be trivially done in polynomail time. Checking that the morphism does indeed preserve the acceptance of runs is equivalent to the problem of checking the equivalence of two deterministic generalised Büchi automata, that can be done in polynomial time (Proposition I.15).

To show the NP-hardness we give a reduction from the problem Vertex Cover to Transitions-to-States. Let $G = (V, E)$ be a simple connected undirected graph. We define a transition system $\mathcal{TS}_G$ whose underlying graph is obtained from G as follows: its

[20] We note that the notion of morphism of transition system applies without any modification to state-based TS, since it does not impose any condition in the images of the colours of the edges, only in the acceptance of infinite runs.

set of states is V, and it has transitions $v \to u$ if $(v,u) \in E$ and $v \to v$ for all $v \in V$. We take an arbitrary vertex $v_0 \in V$ as initial state of $\mathcal{TS}_G$.

We define (in polynomial time) a Büchi acceptance condition over the transitions of $\mathcal{TS}_G$ as $v \xrightarrow{0} u$ if $u \neq v$ and $v \xrightarrow{1} v$.[21]

✧ Claim II.139.1. *A run of $\mathcal{TS}_G$ is accepting if and only if it visits at least two vertices infinitely often.*

Let $\mathsf{minCover}(G)$ be the minimal k such that G admits a cover of size k.

✧ Claim II.139.2. *There is a state-based Büchi TS of size $|V|+\mathsf{minCover}(G)$ admitting a locally bijective morphism to $\mathcal{TS}_G$.*

Proof. Let $V' \subseteq V$ be a cover of G. Let $\mathcal{TS}'$ be the transition system defined as:

- The set of vertices is the disjoint union of V and V', that is: $V \times \{1\} \sqcup V' \times \{0\}$.
- The transitions are given by:
  - $(v,1) \to (v,1)$ for all $v \in V$,
  - $(v',0) \to (v',1)$ for all $v' \in V$,
  - $(v,1) \to (v',0)$ if $v' \in V'$ and $(v,v') \in E$ (we note that this implies $v \neq v'$), and
  - $(v',0) \to (u',0)$ if $v',u' \in V'$ and $(v',u') \in E$. .
- The initial state is $(v_0,1)$.
- The Büchi condition (over the states) is given by $\mathsf{col}_{\mathsf{st}}((v,1)) = 1$ and $\mathsf{col}_{\mathsf{st}}((v',0)) = 0$.

That is, $\mathcal{TS}'$ is obtained by duplicating the states of V'. A run in $\mathcal{TS}'$ is accepting if and only if it visits some of the new copies of the states in V' infinitely often.

We define a morphism $\varphi\colon \mathcal{TS}' \to \mathcal{TS}_G$ by taking the projection into the first component, that is, $\varphi_V((v,x)) = v$, for $x \in \{0,1\}$, and $\varphi_E((v,x) \to (u,y)) = v \to u$. It is immediate to check that φ is a weak morphism and locally bijective. We show that it preserves the acceptance of runs. Let ρ' be an accepting run in $\mathcal{TS}'$. Then, it takes infinitely often transitions of the form $(v,x) \to (v',0)$, with $v \neq v'$ (these are the only incoming transitions to vertices of the form $(v',0)$). Therefore, $\varphi_{\mathcal{R}uns}(\rho')$ changes of state infinitely often, so it is accepting. Conversely, let ρ' be a rejecting run in $\mathcal{TS}'$. Then, it eventually does not visit any vertex $(v',0)$. We show that eventually it stays in a single vertex. Indeed, each time that rr' changes twice of state, we produce a partial run of the form: $(v_1,1) \to (v_2,1) \to (v_3,1)$, with $v_1 \neq v_2$, $v_2 \neq v_3$ and $(v_1,v_2) \in E$ and $(v_2,v_3) \in E$. Since V' is a cover, either $v_2 \in V'$ or $v_3 \in V'$. In the first case, no transition $(v_1,1) \to (v_2,1)$ exists in $\mathcal{TS}'$, in the second case, no transition $(v_2,1) \to (v_3,1)$ exists in $\mathcal{TS}'$, a contradiction. Therefore, $\varphi_{\mathcal{R}uns}(\varphi)$ eventually stays in a single vertex and it is also rejecting. ◁

✧ Claim II.139.3. *If $\mathcal{TS}'$ is a state-based Büchi TS admitting a locally bijective morphism to $\mathcal{TS}_G$, then $|\mathcal{TS}'| \geq |V| + \mathsf{minCover}(G)$.*

Proof. Let Q' be the set of states of $\mathcal{TS}'$, and let $Q' = Q'_0 \sqcup Q'_1$, where Q'_0 are the states coloured with 0 (we will call them Büchi states), and Q'_1 those coloured with 1.

[21] We could define an equivalent generalised Büchi acceptance condition over the states of $\mathcal{TS}_G$ by $B = \{B_v \mid v \in V\}$, where $B_v = \{u \in V \mid u \neq v\}$.

We prove that $|Q'_1| \geq |V|$ and $|Q'_1| \geq \mathsf{minCover}(G)$. Let $\varphi\colon \mathcal{TS}' \to \mathcal{TS}_G$ be a locally bijective morphism.

To prove that $|Q'_1| \geq |V|$, we show that for every state v of $\mathcal{TS}$, there is $q \in Q'_1$ such that $v \in \varphi^{-1}(q)$. Let ρ be a run in $\mathcal{TS}_G$ that reaches v and stays there forever (this is a rejecting run). There is a unique run ρ' in $\mathcal{TS}'$ sent to ρ under φ, and such run has to be rejecting, so eventually it stays in Q'_1. Also, since ρ eventually stays in v, the run ρ' eventually stays in q^{-1}, so $q^{-1} \cap Q'_1 \neq \emptyset$.

To prove that $|Q'_0| \geq \mathsf{minCover}(G)$, we show that $\varphi(Q'_0)$ is a cover of G. Suppose by contradiction that this is not the case, that is, there is $(v, u) \in E$ such that no $q \in Q'_0$ is sent under φ to v, and no $q \in Q'_0$ is sent to u. Consider a run ρ in $\mathcal{TS}_G$ that reaches v and then it is of the form $(v \to u \to v)^\omega$. This run is accepting, since it changes of state infinitely often. However, the run ρ does not accept as a preimage under φ an accepting run, contradicting the fact that φ is a locally bijective morphism. ◁

◀

✦ Remark II.140. *By Corollary II.96, we can compute the ACD of a generalised Büchi TS in polynomial time. This fact, combined with Theorem II.15, shows that providing optimal transformations towards state-based parity TS is* NP*-hard, even if the ACD is given as input.*

8.2 Suboptimal transformations for state-based transition systems

We now show how to use the alternating cycle decomposition to obtain transformations towards state-based transition systems. However, this transformations are not optimal. In fact, as shown in the previous section, obtaining optimal transformations towards state-based transition sytems is NP-hard even if the ACD is given as input. As we will see, we will need to add some additional states to put a correct acceptance condition over the states. There are some choices to make for adding these extra states, from where NP-hardness arises.

As no strong optimality result can be obtained from the result of this section, we keep the discussion in an informal level.

State-based version of the ZT-parity-automaton

Definition. Let $\mathcal{F} \subseteq 2^\Sigma_+$ be a family of subsets. We explain how to obtain a state-based parity automaton recognising $\mathsf{Muller}_\Sigma(\mathcal{F})$ from the Zielonka tree $\mathcal{Z}_\mathcal{F}$. We suppose for the ease of the presentation that the root of $\mathcal{Z}_\mathcal{F}$ is round, that is, $\Sigma \in \mathcal{F}$.

The automaton has a state per node in the Zielonka tree (instead of one state per leaf). We colour each state with its depth in the tree, that is:

$$\mathsf{col}_{\mathsf{st}}(q) = p_\mathcal{Z}(q).$$

The idea behind the definition of the transitions of the automaton is the same as in the ZT-parity-automaton. However, in the current case we need to visit from time to time a node that is not a leaf of the tree in order to produce the desired priority. We do so whenever we are sure that we have already tried all branches below a node n.

Assume that we are in a node n and read a letter a. To determine the successor, we go to $\mathsf{Supp}(n, a)$, and change to the next branch below it. If we were in the last children below

$\mathsf{Supp}(n, a)$, we set the target state to be $\mathsf{Supp}(n, a)$. On the contrary, we just go the the leftmost leaf below the branch, as we would have done in the classic ZT-parity-automaton.

A proof of correctness follows the same lines as the proof of Proposition II.34.

Optimality. Using a slightly more complex combinatorial argument than the one presented in the proof of Theorem II.3, one can show that this automaton is minimal amongst deterministic parity automata recognising $\mathsf{Muller}_\Sigma(\mathcal{F})$. (This fact was remarked by Klara J. Meyer.)

State-based version of the ACD-parity-transform

Definition. Let $\mathcal{TS}$ be a Muller transition system. We define a state-based parity transition system $\mathsf{ACD}^{\mathsf{st}}_{\mathsf{parity}}(\mathcal{TS})$ using the same idea as in the previous paragraph:

- Vertices are of the form (v, n), for v a vertex of $\mathcal{TS}$ and n a node in $\mathcal{T}_v$.
- We colour a vertex (v, n) by the depth of n, that is, $\mathsf{col}_{\mathsf{st}}(v, n) = p_{\mathcal{ACD}}(n)$.
- For each vertex (v, n) and transition $e = v \to v'$ in $\mathcal{TS}$, we define an edge $(v, n) \to (v', n')$. We determine n' as follows. Assume that v and v' are in the same SCC, (on the contrary, we let n' be the leftmost leaf in $\mathcal{T}_{v'}$). If $\mathsf{Jump}_{\mathcal{T}_{v'}}(n, \mathsf{Supp}(n, e))$ is not in the leftmost branch below $\mathsf{Supp}(n, e)$, then let v' be $\mathsf{Jump}_{\mathcal{T}_{v'}}(n, \mathsf{Supp}(n, e))$ (as in the transition-based case). Otherwise (a "lap" around $\mathsf{Supp}(n, e)$ is finished) we let $n' = \mathsf{Supp}(n, e)$.

A proof of correctness – that is, $\mathsf{ACD}^{\mathsf{st}}_{\mathsf{parity}}(\mathcal{TS})$ admits a locally bijective morphism towards $\mathcal{TS}$ – follows the same lines as the proof of Proposition II.68.

Non-optimality. The transition system $\mathsf{ACD}^{\mathsf{st}}_{\mathsf{parity}}(\mathcal{TS})$ is not minimal amongst those state-based parity TS admitting a locally bijective morphism to $\mathcal{TS}$. In fact, if n is not a leaf in the ACD, states of the form (v, n) are not necessarily reachable in this transition system. We only need to add those that can be reached from the initial state. However, the set of reachable states does depend on the ordering of the children in the trees of the ACD, and therefore the size of the final transition system depends on this ordering. Some heuristics to find orderings giving small transitions systems are proposed in [Cas+22].

Nevertheless, we can analyse the size of $\mathsf{ACD}^{\mathsf{st}}_{\mathsf{parity}}(\mathcal{TS})$ and deduce some (non optimal) guarantees. Let d be the maximal height of the trees of $\mathcal{ACD}_{\mathcal{TS}}$. The final TS has size:

$$|\mathsf{ACD}^{\mathsf{st}}_{\mathsf{parity}}(\mathcal{TS})| = \sum_{v \in \mathcal{TS}} |\mathcal{T}_v| \leq \sum_{v \in \mathcal{TS}} d \cdot |\mathsf{Leaves}(\mathcal{T}_v)| = d \cdot |\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})|.$$

We obtain that $\mathsf{ACD}^{\mathsf{st}}_{\mathsf{parity}}(\mathcal{TS})$ is at most d times larger than a TS admitting a locally bijective morphism to $\mathcal{TS}$ (bound that we could obtain just by applying a naive translation of $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ to a state-based model). Since the inequality above are not tight, we can slightly improve this result. In particular, for generalised (co)Büchi TS we obtain:

Proposition II.141.

Let $\mathcal{TS}$ be a generalised (co)Büchi transition system. Let $\mathcal{TS}'$ be a state-based (co)Büchi transition system admitting a locally bijective morphism to $\mathcal{TS}$. Then:

$$|\mathsf{ACD}^{\mathsf{st}}_{\mathsf{parity}}(\mathcal{TS})| \leq |\mathcal{TS}'| + |\mathcal{TS}|.$$

Proof. We can transform $\mathcal{TS}'$ into a transition-based TS admitting a locally bijective morphism to $\mathcal{TS}$ without any blow-up. By Theorem II.6, we have then that it is not smaller than $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$. The vertices of $\mathsf{ACD}^{\mathsf{st}}_{\mathsf{parity}}(\mathcal{TS})$ is the union of the set of states of $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ with a subset of nodes of the form (v, n_0), where n_0 is the root of $\mathcal{T}_v$. Therefore:

$$|\mathsf{ACD}^{\mathsf{st}}_{\mathsf{parity}}(\mathcal{TS})| \leq |\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})| + |\mathcal{TS}| \leq |\mathcal{TS}'| + |\mathcal{TS}|.$$ ◀

Minimisation of automata and succinctness results

III

Outline

Las pequeñas cosas son las que hacen más grande la vida.
Luis Piedrahita

Introduction

Automata minimisation stands as one of the most fundamental problems in automata theory, for various reasons. Firstly, for its applications: when employing algorithms that rely on automata, having the smallest possible automata is crucial for efficiency. Secondly, beneath the problematic of minimisation lies a profoundly fundamental question: What is the essential information needed to represent a formal language?

Finite words. For automata over finite words, minimal automata are well-understood. Each regular language admits a unique minimal and canonical deterministic automaton, in which states corresponds to the residuals of the language (the equivalence classes of the Myhill-Nerode congruence). Moreover, this minimal automaton can be obtained in time $\mathcal{O}(n \log n)$, where n is the size of an input deterministic automaton [Hop71].

Infinite words. In the case of ω-automata, the situation is not as simple, and the minimisation problem has an intriguing complexity status. Contrary to the case of finite words, in general, the residuals of a language are not sufficient to construct a correct deterministic automaton. For certain classes of languages, however, this suffices [SL94, MS97]; an example is that of Weak_1-languages. Using this property, Gutleben and Löding showed that Weak_1-automata can be minimised in polynomial time [Gut96, Löd01].

In 2010, Schewe showed that the minimisation of deterministic Büchi and parity automata is NP-complete, *if the acceptance condition is defined over the states* [Sch10]. However, the reduction of NP-hardness does not generalise to automata with transition-based acceptance. A surprising positive result was obtained in 2019 by Abu Radi and Kupferman: we can minimise in polynomial time history-deterministic coBüchi automata using transition-based acceptance [AK19]. Schewe showed that the minimisation was again NP-hard for HD automata with state-based acceptance [Sch20]. Abu Radi and Kupferman left open the following question, which constitutes the main motivation for the research presented in this chapter:

Question III.1 ([AK19])**.**

Can deterministic Büchi automata be minimised in polynomial time?

The corresponding questions for history-deterministic and for parity automata are also open.

Succinctness of history-deterministic automata. Since their introduction in 2006 [HP06], a pressing question regarding history-deterministic automata was whether they can be more succinct than their deterministic counterparts. This question was answered positively by Kuperberg and Skrzypczak: minimal HD coBüchi automata can be exponentially smaller than equivalent deterministic ones [KS15, Theorem 1]. In the case of Büchi automata, they showed that minimal deterministic automata have at most $\mathcal{O}(n^2)$ more states than history-deterministic ones [KS15, Theorem 8], and it is an open question whether this bound is tight. However, no much has been said for HD automata using other acceptance conditions, or for automata recognising subclasses of languages.

Contributions

Our contributions are already nearly outlined in the table of contents of this chapter:

1. **Minimisation of Rabin automata is NP-complete.** We show the NP-completeness of the minimisation of deterministic Rabin automata (Theorem III.1). Moreover, we show that this problem is already NP-hard when restricted to automata that recognise Muller languages, and even if the input Muller language is represented as a Zielonka tree. This result, together with the construction of a minimal HD Rabin automaton from the Zielonka tree (Theorem II.4), seems to indicate that HD automata are better-behaved than deterministic ones in this context.

2. **Efficient minimisation of parity automata recognising Muller languages.** Contrary to the previous case, we show that deterministic parity automata recognising Muller languages can be minimised in polynomial time (Theorem III.4). This is possible by the fact that such minimal automata are given by the Zielonka tree of the Muller language (Theorem II.2). This result exhibits the limitations of our previous NP-hardness reduction.
3. **Minimisation of generalised Büchi automata is NP-complete.** We show that the minimisation of deterministic generalised Büchi and generalised coBüchi automata is NP-complete (Theorem III.2). In this case, the use of Muller languages cannot suffice to obtain NP-hardness. Our reduction generalises the one presented for Rabin automata, and introduces ideas necessary to go beyond the limitations of the use of Muller languages in this kind of reductions.
4. **Minimisation of Muller automata is NP-hard.** As a consequence of the previous point, we also obtain the NP-harness of the minimisation of deterministic Muller automata (Theorem III.3). We find this result quite interesting because, even though Muller automata might appear more complex than other classes, they use a very natural acceptance condition. The results of Wagner [Wag79] establishing that the structure of a deterministic Muller automaton $\mathcal{A}$ determines the level in the Wagner hierarchy of $\mathcal{L}(\mathcal{A})$ could lead to the idea that this class of automata benefits from particularly good properties.
5. **Succinct HD Rabin automata.** We show that HD Rabin automata recognising Muller languages can be exponentially more succinct than equivalent deterministic ones (Theorem III.5). This result is obtained by reducing the problem to a graph theoretic question about bounds on the chromatic number of graphs. This complements our result from the previous chapter establishing that HD parity automata for Muller languages are not more succinct than deterministic ones (Corollary II.36). Our results concerning the succinctness of HD Rabin automata will be applied in Chapter IV to obtain bounds on the memory for games.

Prerequisites and links with other chapters

Besides the definitions of automata and the standard acceptance conditions introduced in the general preliminaries (Section I.3), we make extensive use of the Zielonka tree of a Muller language, as it provides an expressive representation of it (see Section II.3 for its definition). Also, we use as a black-box the fact that we can construct a minimal deterministic parity automaton (resp. minimal history-deterministic Rabin automaton) from the Zielonka tree (Theorems II.2 and II.4). To obtain Lemmas III.7 and III.20, we apply Proposition II.109, that tells us when we can relabel a Muller automaton with an equivalent Rabin condition on top of it. We include some results and discussions about history-deterministic automata.

To show the NP-hardness of some decision problems, we reduce them to Chromatic number. We introduce this problem and related vocabulary in Appendix A.

Collaborators and related publications

The contents of Sections III.1 and III.3 have been published in the conference paper [Cas22]. Section III.2 is joint work with Corto Mascle, and it is still unpublished. The result from Section III.4 is based on joint work with Thomas Colcombet and Karoliina Lehtinen and appears in the conference paper [CCL22]. We thank Marthe Bonamy and

Pierre Charbit for their help with graph theory and pointing us to the reference [MR14].

The problem of the minimisation of transition-based Büchi automata was brought to my attention by a talk of Orna Kupfermann during the program *Theoretical Foundations of Computer Systems* at the Simons Institute for the Theory of Computing, held in 2021 (online). I thank her, Bader Abu Radi, Thomas Colcombet, Nathanaël Fijalkow, Karoliina Lehtinen and Nir Piterman for interesting discussions on the subject, at the origin of this line of research.

1 NP-completeness of the minimisation of Rabin automata

This section is devoted to proving the NP-completeness of the minimisation of (transition-based) Rabin automata, stated in Theorem III.1. We start by giving the formal definition and explaining the different inputs of the decision problems under study.

1.1 Decision problems and statement of the results

We now give formal definition of the decisions problems we consider and state the main results of this section.

Problem:	Minimisation of Rabin automata
Input:	A deterministic Rabin automaton $\mathcal{A}$ and a positive integer k.
Question:	Is there a deterministic Rabin automaton equivalent to $\mathcal{A}$ of size at most k?

We define analogously the problems of the minimisation of X automata, for X any of the classes of languages introduced in Section I.3 (parity, Streett, generalised Büchi, etc.). Similarly, we define versions of this problem for history-deterministic automata. We note that for the variants of the problem Minimisation of Rabin automata, we modify "Rabin" (resp. "deterministic") by "X" (resp. "HD") in both the input and the output of the problem.

✦ Remark III.1. *We remark that the minimisation problem for deterministic automata using acceptance conditions that are dual (Rabin – Street, Büchi – coBüchi, generalised Büchi – generalised coBüchi) are polynomial time equivalent (see also Remark I.13). We note that this is no longer the case if we consider the analogous problems for history-deterministic automata.*

We will consider a particular case of this problem, namely, the one in which the language for which we want to find a minimal automaton is a Muller language. In this case, we can make the problem even easier by representing the language more explicitly by giving its Zielonka tree.

Problem: Minimal Rabin automata for Muller languages
Input: The Zielonka tree $\mathcal{Z}_\mathcal{F}$ of a family $\mathcal{F} \subseteq 2^\Sigma_+$ and a positive integer k.
Question: Is there a deterministic Rabin automaton recognising $\mathsf{Muller}_\Sigma(\mathcal{F})$ of size at most k?

✦ Remark III.2. *The problem* Minimal Rabin automata for Muller languages *reduces in polynomial time to the problem* Minimisation of Rabin automata. *Indeed, by the construction of the Zielonka-tree-parity-automaton (Section II.3.2 of Chapter II), we can build in linear time in the size of* $\mathcal{Z}_\mathcal{F}$ *a deterministic parity automaton recognising* $\mathsf{Muller}_\Sigma(\mathcal{F})$*, which is in particular a Rabin automaton.*

Theorem III.1.

The problems Minimisation of Rabin automata and Minimal Rabin automata for Muller languages are NP-complete.

✦ Remark III.3. *The reduction we give shows that the problem* Minimal Rabin automata for Muller languages *is* NP*-hard also if the input is given as the list of subsets of* $\mathcal{F}$*. However, in this case, we lack a proof of the containment in* NP.

The results of this section apply to Streett automata too by symmetry.

The reduction we propose will only make use of Muller languages of parity index $[1, 3]$. This reduction cannot be directly extended to show the NP-hardness of the minimisation of parity automata, as we will show in Section III.3 that we can minimise parity automata recognising Muller languages in polynomial time. As we will see in Corollary III.8, this reduction does not extend to history-deterministic automata neither.

1.2 Proof of NP-completeness

For the containment in NP, we use the fact that we can test equivalence of deterministic Rabin automata in polynomial time [CDK93].

Proposition III.4.

Given a deterministic Rabin automaton $\mathcal{A}$ and a positive integer k, we can decide in non-deterministic polynomial time whether there is an equivalent Rabin automaton of size k.

Proof. A non-deterministic Turing machine just has to guess an equivalent automaton $\mathcal{A}_k$ of size k, and by Corollary I.16 it can check in polynomial time whether $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_k)$. ◀

Corollary III.5.

Given the Zielonka tree $\mathcal{Z}_\mathcal{F}$ of a family $\mathcal{F} \subseteq 2^\Sigma_+$, we can decide in non-deterministic polynomial time whether there is a deterministic Rabin automaton of size k recognising $\mathsf{Muller}_\Sigma(\mathcal{F})$.

Proof. As we show in Section II.3.2, we can construct a deterministic parity automaton for $\mathsf{Muller}_\Sigma(\mathcal{F})$ in polynomial time from the Zielonka tree $\mathcal{Z}_\mathcal{F}$. This is in particular a Rabin automaton, so we can just apply the previous proposition. ◀

To show the NP-hardness, we describe a reduction from the Chromatic number problem (one of 21 Karp's NP-complete problems) to the problem Minimal Rabin automata for Muller languages. As indicated in Remark III.2, this suffices to show the NP-harness of Minimisation of Rabin automata. We refer to Appendix A for the definition of this problem and terminology on undirected graphs.

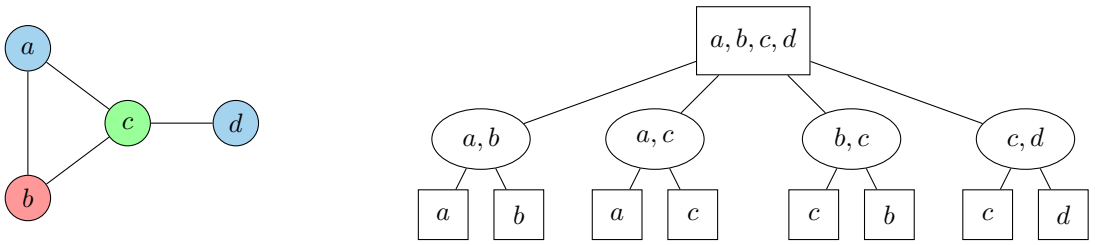
Let $G = (V, E)$ be an undirected graph and consider the family $\mathcal{F}_G \subseteq 2^V_+$ given by:

$$\mathcal{F}_G = \{\{v, u\} \mid (v, u) \in E\}.$$

That is, $\mathcal{F}_G$ contains exactly the pair of vertices joined by an edge. We remark that the Muller language associated to $\mathcal{F}_G$ is:

$$\mathsf{Muller}_V(\mathcal{F}_G) = \bigcup_{(v,u)\in E} V^*(v^+u^+)^\omega.$$

An example of the Zielonka tree of $\mathcal{F}_G$ is given in Figure 20.



✦ **Figure 20.** On the left, an undirected graph G and a colouring of it. On the right, the Zielonka tree of $\mathcal{F}_G$.

We remark that for all undirected graph G (with $|V| > 2$ and E non-empty), the Zielonka tree $\mathcal{Z}_{\mathcal{F}_G}$ has height 3 and its root is square. Therefore, the parity index of $\mathsf{Muller}_V(\mathcal{F}_G)$ is $[1, 3]$ (Proposition II.115).

✦ **Lemma III.6.** *Given a simple undirected graph $G = (V, E)$, we can build a deterministic Rabin automaton $\mathcal{A}_G$ of size $|V|$ over the alphabet V recognising $\mathsf{Muller}_V(\mathcal{F}_G)$ in polynomial time.*

Proof. We define the automaton $\mathcal{A}_G = (Q, V, q_0, \Gamma, \delta, \mathsf{Rabin}_\Gamma(R))$ as follows:

- $Q = V$.
- q_0 an arbitrary vertex in Q.
- The set of output colours is $\Gamma = V \times V$.
- $\delta(v, x) = (x, (v, x))$, for $v, x \in V$.
- $R = \{(\mathfrak{g}_{(v,u)}, \mathfrak{r}_{(v,u)}) \mid (v, u) \in E\}$, where we define for each edge $(v, u) \in E$ a Rabin pair with green and red sets given by:
  - $\mathfrak{g}_{(v,u)} = \{(v, u)\}$,
  - $\mathfrak{r}_{(v,u)} = V \times V \setminus \{(v, u), (u, v), (v, v), (u, u)\}$.

That is, the states of the automaton are the vertices of the graph G, and when we read a letter $u \in V$ we jump to the state u. The colours defining the Rabin condition are all pairs of vertices, and we define one Rabin pair for each edge of the graph. This Rabin pair will accept words that eventually alternate between the endpoints of the edge and visit no other vertex.

We prove that $\mathcal{A}_G$ recognises $\mathsf{Muller}_V(\mathcal{F}_G)$. If $w \in \mathsf{Muller}_V(\mathcal{F}_G)$, then $\mathsf{Inf}(w) = \{v, u\}$ for some $v, u \in V$, such that $(v, u) \in E$, so eventually we will only visit the states v and u of the automaton, and produce colour (v, u) when alternating between the,. Therefore, by definition of the Rabin pairs, w is accepted by the Rabin pair $(\mathfrak{g}_{(v,u)}, \mathfrak{r}_{(v,u)})$.

Conversely, suppose that a word w is accepted by $\mathcal{A}_G$, and let $\alpha \in (V \times V)^\omega$ be the output of the run of $\mathcal{A}_G$ over w. It must satisfy $\mathsf{Inf}(\alpha) \cap \mathfrak{r}_{(v,u)} = \emptyset$ for some $(v, u) \in E$, so eventually α only contains the pairs $(v, u), (u, v), (v, v)$ and (u, u). If w was eventually of the form v^ω or u^ω, we would have that $\mathsf{Inf}(\alpha) \cap \mathfrak{g}_{(v,u)} = \emptyset$ for all $(v, u) \in E$. We conclude that the word w eventually alternates between two vertices connected by an edge, so $w \in \mathsf{Muller}(\mathcal{F}_G)$.

The automaton has $|V|$ states, the transition function δ has size $\mathcal{O}(|V|^2)$ and the Rabin condition R has size $\mathcal{O}(|E||V|^2)$. ◀

Lemma III.7.

Let $G = (V, E)$ be a simple undirected graph. The size of a minimal deterministic Rabin automaton recognising $\mathsf{Muller}(\mathcal{F}_G)$ coincides with the chromatic number of G.

Proof. We denote $\mathsf{minDetRabin}(\mathcal{F}_G)$ the number of states of a minimal deterministic Rabin automaton recognising $\mathsf{Muller}_V(\mathcal{F}_G)$. $\mathsf{minDetRabin}(\mathcal{F}_G) \leq \chi(G)$**:** Let $c : V \to [1, k]$ be a colouring of size k of G. We will define a deterministic Muller automaton $\mathcal{A}_c$ with k states recognising $\mathsf{Muller}_V(\mathcal{F}_G)$ and then use Proposition II.109 to show that $\mathcal{A}_c$ is Rabin type, allowing us to conclude. Let $\mathcal{A}_c = (Q, V, q_0, \Gamma, \delta, \mathsf{Muller}_V(\mathcal{F}_G))$ be the Muller automaton defined by:

- $Q = \{1, 2, \ldots, k\}$.
- $q_0 = 1$.
- $\Gamma = V$.
- $\delta(q, x) = (c(x), x)$ for $q \in Q$ and $x \in V$.
- The acceptance set is the Muller language $\mathsf{Muller}_V(\mathcal{F}_G))$ itself.

This automaton trivially recognises $\mathsf{Muller}_V(\mathcal{F}_G))$, since the output produced by a word $w \in V^\omega$ is w itself, and the acceptance set is $\mathsf{Muller}_V(\mathcal{F}_G))$.

We will prove that the union of any pair of rejecting cycles of $\mathcal{A}_c$ with some state in common is rejecting. By Proposition II.109, this implies that we there exists a deterministic Rabin automaton over the underlying graph of $\mathcal{A}_c$ recognising $\mathsf{Muller}_V(\mathcal{F}_G)$.

Let $\ell_1, \ell_2 \subseteq \mathcal{C}ycles(\mathcal{A}_c)$ be two rejecting cycles (that is, $\mathsf{col}(\ell_i) \notin \mathcal{F}_G$ for $i \in \{1, 2\}$) and such that $\mathsf{States}(\ell_1) \cap \mathsf{States}(\ell_2) \neq \emptyset$. We note col the function associating output colours to the transitions of $\mathcal{A}_c$. We distinguish 3 cases:

- $|\mathsf{col}(\ell_i)| \geq 3$ for some $i \in \{1, 2\}$. In this case, their union also has more than 3 colours, so it must be rejecting.

- $\mathsf{col}(\ell_i) = \{u, v\}$, $(u, v) \notin E$ for some $i \in \{1, 2\}$. In that case, $\mathsf{col}(\ell_1 \cup \ell_2)$ also contains two vertices that are not connected by an edge, so it must be rejecting.
- $\mathsf{col}(\ell_1) = \{v_1\}$ and $\mathsf{col}(\ell_2) = \{v_2\}$. In this case, since from every state q of $\mathcal{A}_c$ and every $v \in V$ we have $\delta(q, v) = (c(v), v)$, the only state in each cycle is, respectively, $c(v_1)$ and $c(v_2)$. As ℓ_1 and ℓ_2 share some state, we deduce that $c(v_1) = c(v_2)$. If $v_1 = v_2$, $\ell_1 \cup \ell_2$ is rejecting because $|\mathsf{col}(\ell_1 \cup \ell_2)| = 1$. If $v_1 \neq v_2$, it is also rejecting because $c(v_1) = c(v_2)$ and therefore $(v_1, v_2) \notin E$, by definition of a colouring.

Since $\mathsf{col}(\ell_i)$ is rejecting, it does not consist on two vertices connected by some edge and we are always in some of the cases above, finishing this direction of the proof.

$\chi(G) \leq \mathsf{minDetRabin}(\mathcal{F}_G)$**:** Let $\mathcal{A} = (Q, V, q_0, \Gamma, \delta, \mathsf{Rabin}_\Gamma(R))$ be a Rabin automaton of size k recognising $\mathsf{Muller}_V(\mathcal{F}_G)$. We let col be the function assigning output colours to the transitions of $\mathcal{A}$. We will define a colouring of size k of G, $c : V \to Q$.

For each $v \in V$ we define a subset $Q_v \subseteq Q$ as:

$$Q_v = \{q \in Q \mid \text{there is a cycle } \ell \text{ containing } q \text{ and } \mathsf{col}(\ell) = \{v\}\}.$$

For every $v \in V$, the set Q_v is non-empty, as it must exist a (non-accepting) run over v^ω in $\mathcal{A}$. For each $v \in V$ we pick some $q_v \in Q_v$, and we define the colouring $c : V \to Q$ given by $c(v) = q_v$.

In order to prove that this is indeed a colouring, we we will show that any two vertices $v, u \in V$ such that $(v, u) \in E$ satisfy that $Q_v \cap Q_u = \emptyset$, and therefore they also satisfy $c(v) \neq c(u)$. Suppose by contradiction that there is some $q \in Q_v \cap Q_u$. Let ℓ_v and ℓ_u be cycles containing q labelled with v and u, respectively. By definition of $\mathcal{F}_G$, both cycles ℓ_v and ℓ_u are rejecting, as $x^\omega \notin \mathsf{Muller}_V(\mathcal{F}_G)$, for any $x \in V$. However, since $(u, v) \in E$, the union of ℓ_v and ℓ_u is accepting, contradicting Proposition II.109. ◀

1.3 Discussion: The case of history-deterministic Rabin automata

We remark that the reduction we have proposed to show the NP-hardness of the minimisation of deterministic Rabin automata does not extend to history-deterministic Rabin automata, nor to deterministic or HD parity automata. Indeed, the analogous of the problem Minimal Rabin automata for Muller languages for HD Rabin automata or for parity automata can be solved in polynomial time, as a minimal HD Rabin automaton (resp. deterministic and HD parity automata) for a language $\mathsf{Muller}_\Sigma(\mathcal{F})$ can be build in polynomial time from the Zielonka tree of $\mathcal{F}$ (Theorems II.2 and II.4).

Corollary III.8.

Given the Zielonka tree $\mathcal{Z}_\mathcal{F}$ of a family $\mathcal{F} \subseteq 2^\Sigma_+$, we can build in polynomial time in the size of $\mathcal{Z}_\mathcal{F}$:

- a minimal HD Rabin automaton recognising $\mathsf{Muller}_\Sigma(\mathcal{F})$,
- a minimal deterministic parity automaton recognising $\mathsf{Muller}_\Sigma(\mathcal{F})$, and
- a minimal HD parity automaton recognising $\mathsf{Muller}_\Sigma(\mathcal{F})$.

In Section III.3, we will show how this result can be used to minimise deterministic parity automata recognising Muller languages, as it is possible to build the Zielonka tree

of $\mathcal{F}$ from a DPA recognising $\mathsf{Muller}_\Gamma(\mathcal{F})$. However, it is not clear how to do this if $\mathsf{Muller}_\Gamma(\mathcal{F})$ is given by an HD Rabin automaton (see Question III.6).

2 NP-hardness of the minimisation of generalised Büchi and Muller automata

In this section, we show the NP-completeness of the minimisation of deterministic generalised (co)Büchi automata (Theorem III.2) and the NP-hardness of the minimisation of deterministic Muller automata (Theorem III.3). The proof of NP-hardness generalised the reduction to Chromatic number given in the previous section. However, we face a difficulty: the use of Muller languages do not suffice in this case, as generalised (co)Büchi automata recognising Muller languages can be trivially minimised (Proposition III.9). Therefore, we need to use slightly more complicated languages, which poses some technical challenges to give the required lower bounds to finish the proof.

Our results show the hardness of the minimisation of a class of automata already quite close to Büchi automata, and introduces fresh ideas for reductions that could be useful to make progress in Question III.1.

We recall that deterministic generalised Büchi (resp. generalised coBüchi) automata recognise languages of parity index at most $[0, 1]$ (resp. at most $[1, 2]$) (c.f. Remark I.22).

2.1 Statement of the results

We state the results of this section for generalised Büchi automata. They trivially apply to generalised coBüchi automata too by symmetry (c.f. Remark III.1).

Theorem III.2.

The problem Minimisation of generalised Büchi automata is NP-complete.

Theorem III.3.

The problem Minimisation of Muller automata is NP-hard, for Muller automata with the acceptance condition represented explicitly, as a Zielonka tree or as a Zielonka DAG.

Theorem III.3 can be obtained as a corollary of Theorem III.2 by using Proposition II.122 and Corollary II.96. It will also follow from the reduction we propose.

We remark that these problems are also NP-hard if the input automaton is a deterministic Büchi automaton, as given a generalised Büchi automaton we can build an equivalent Büchi one in polynomial time.

Minimisation of generalised (co)Büchi automata recognising Muller languages

Before moving to the proof of Theorem III.2, we show that, contrary to the case of Rabin automata in the previous section, we cannot restrict ourselves to the study of Muller languages to prove the NP-hardness of the minimisation of generalised Büchi automata.

The following proposition is obtained by combining Propositions II.105 and II.115.

Proposition III.9.

Let L be a Muller language of parity index at most $[0,1]$ (resp. $[1,2]$). Then, L can be recognised by a generalised Büchi (resp. generalised coBüchi) automaton with just one state.

We conclude that the decision problem Minimisation of generalised Büchi automata becomes trivial when restricted to the class of Muller languages. We remark that, moreover, the one-state minimal automaton recognising a Muller language L can be obtained from any deterministic generalised Büchi automaton recognising L in polynomial time, as it suffices to identify the maximal rejecting subset of letters, which will determine the Zielonka tree from which we can obtain the condition to put in the automaton.

2.2 Proof of NP-completeness

We now prove the NP-completeness of the minimisation of generalised Büchi automata.

As in the previous section, we obtain the inclusion in NP of Minimisation of Rabin automata as a corollary of the fact that we can check equivalence of deterministic generalised Büchi automata in polynomial time [CDK93].

Proposition III.10.

Given a deterministic generalised Büchi automaton $\mathcal{A}$ and a positive integer k, we can decide in non-deterministic polynomial time whether there is an equivalent generalised Büchi automaton of size k.

To show the NP-hardness, we will give a reduction from the problem 3-colorability to Minimisation of generalised coBüchi automata. We conjecture that this is in fact a reduction from the Chromatic number problem. In the proof, we use generalised coBüchi, as working with the complement of the languages used in the proof is slightly less natural.

Let $G = (V, E)$ be a simple undirected graph. We define the *neighbourhood* of a vertex $v \in V$ as:

$$N[v] = \{u \in V \mid (v,u) \in E\} \cup \{v\}.$$

We associate a language (of parity index $[1,2]$) over the alphabet V to each vertex $v \in V$:

$$L_v = \{w \in V^\omega \mid \mathsf{Inf}(w) \subseteq N[v] \text{ and } w \text{ does not contain the factor } vv \text{ infinitely often}\}.$$

Finally, we associate a language (of parity index $[1,2]$) over the alphabet V to the graph G:

$$L_G = \bigcup_{v \in V} L_v.$$

That is, a word $w \in V^\omega$ belongs to L_G if for some vertex v, the letters of w are eventually vertices in the neighbourhood of v, and it does not contains consecutive occurrences of v infinitely often.

✦ **Lemma III.11.** *Given a simple undirected graph $G = (V, E)$, we can build a deterministic generalised coBüchi automaton $\mathcal{A}_G$ of size $|V|$ over the alphabet V recognising L_G in polynomial time.*

Proof. We define the automaton $\mathcal{A}_G = (Q, V, q_0, \Gamma, \delta, \mathsf{genCoBuchi}_\Gamma(B))$ as follows:

- $Q = V$.
- q_0 an arbitrary vertex in Q.
- The set of output colours Γ is the whole set of transitions of the automaton. Each transition has each own colour.
- $\delta(v, x) = x$, for $v, x \in V$ (we only indicate the destination, as we use one colour per transition).
- $B = \{B_v \mid v \in V\}$, where $B_v = \{q \xrightarrow{u} u \mid u \notin N[v]\} \cup \{v \xrightarrow{v} v\}$.

That is, the states of $\mathcal{A}_G$ are V and we jump to state v when reading letter v. For each vertex, we define a subset B_v of transitions that serves to reject words $w \notin L_v$.

We prove that $\mathcal{A}_G$ recognises L_G. A word $w \in V^\omega$ belongs to L_G if and only if there is $v \in V$ such that $w \in L_v$. We prove that w belongs to L_v if and only if a run over w in $\mathcal{A}_G$ eventually does not take transitions in B_v (and therefore, it is accepted, by the semantics of generalised coBüchi conditions). Indeed, if $w \in L_v$, a run over w in $\mathcal{A}_G$ will eventually only take transitions $q \xrightarrow{u} u$ with $u \in N[v]$, and will not take the transitions $\{v \xrightarrow{v} v\}$, as this transition is only taken if v is read twice in a row. Conversely, if a run over w in $\mathcal{A}_G$ eventually does not take transitions in B_v, it eventually only takes transitions labelled with letters in $N[v]$, and, as it avoids transition $\{v \xrightarrow{v} v\}$, w does not contain consecutive occurrences of v infinitely often.

The size of the representation of this automaton is $|V|^2$, and in order to specify the acceptance condition given by B, we have to compute the neighbour $N[v]$ for each $v \in V$, which can be done in time $\mathcal{O}(|V|^3)$. ◀

Lemma III.12.

Let $G = (V, E)$ be a simple undirected graph. There exists a deterministic generalised coBüchi automaton recognising L_G of size $\chi(G)$.

Proof. Let $c : V \to [1, k]$ be a colouring of size k of G. We define an automaton $\mathcal{A}_c = (Q, V, q_0, \Gamma, \delta, \mathsf{genCoBuchi}_\Gamma(B))$ of size k and recognising L_G as follows:

- $Q = \{1, \dots, k\}$.
- q_0 an arbitrary vertex in Q.
- The set of output colours Γ is the whole set of transitions of the automaton.
- $\delta(q, x) = c(x)$, for $q \in Q$, $x \in V$ (we only indicate the destination, as we use one colour per transition).
- $B = \{B_v \mid v \in V\}$, where $B_v = \{q \xrightarrow{u} q' \mid u \notin N[v]\} \cup \{c(v) \xrightarrow{v} c(v)\}$.

That is, the states of $\mathcal{A}_c$ are the colours used by the colouring c, and we jump to the colour of v when reading letter v. As in the previous construction, for each vertex, we define a subset B_v of transitions that serves to reject words $w \notin L_v$.

As in the previous lemma, we prove that w belongs to L_v if and only if a run over w in $\mathcal{A}_c$ eventually does not take transitions in B_v. For this, we use the following claim.

✧ Claim III.12.1. *Suppose that the sequence $q \xrightarrow{u} c(v) \xrightarrow{v} c(v)$ appears in a run in $\mathcal{A}_c$. Then, either $u \notin N[v]$ or $u = v$.*

Proof. By definition of the transitions of $\mathcal{A}_c$, if the transition $q \xrightarrow{u} c(v)$ exists, $c(u) = c(v)$, so by definition of a colouring, $(u, v) \notin E$, and therefore either $u \notin N[v]$ or $u = v$. ◁

This property allows us to end the proof in the exact same way as in the previous lemma: w belongs to L_v if and only if neither transition $c(v) \xrightarrow{v} c(v)$ nor $q \xrightarrow{u} q'$ for $u \notin N[v]$ are taken infinitely often. ◀

Conjecture III.2.

Let $G = (V, E)$ be a simple undirected graph. The size of a deterministic generalised coBüchi automaton recognising L_G is at least $\chi(G)$.

Conjecture III.2 would yield a correct polynomial-time reduction from the problem Chromatic number to Minimisation of generalised coBüchi automata. Unfortunately, we have not been able to find a proof for it. However, this does not prevent us from giving a proof of NP-hardness of the problem, as we can use the remark that the problem Chromatic number is already NP-complete when k is fixed to 3. Therefore, it suffices to check Conjecture III.2 for automata of size at most 3, which can be done by examining a few cases, as the combinatorics for these automata can be greatly simplified.[1]

The following lemma provides an important family of rejecting words.

✦ **Lemma III.13.** *If $w \in V^\omega$ is such that $\mathsf{Inf}(w) = N[v]$ and for all $x \in N[v]$ the word w contains xx infinitely often, then $w \notin L_G$.*

Proof. Suppose by contradiction that $w \in L_u$ for some $u \in V$. As $\mathsf{Inf}(w) \subseteq N[u]$, we must have $(u, v) \in E$, and so $u \in N[v]$. Therefore, w contains the factor uu infinitely often, contradicting that $w \in L_u$. ◀

The prove we give applies to all Muller automata. We will make extensive use of the following remark.

✦ Remark III.14. *Let $\mathcal{A}$ be a deterministic Muller automaton. Let $w_1, w_2 \in V^\omega$ such that the set of transitions visited infinitely often by the run over w_1 and over w_2 in $\mathcal{A}$ coincide. Then, $w_1 \in \mathcal{L}(\mathcal{A})$ if and only if $w_2 \in \mathcal{L}(\mathcal{A})$.*

Lemma III.15.

Let $G = (V, E)$ be a simple undirected graph. If $\chi(G) > 3$, then no deterministic Muller automaton with 3 states recognises L_G.

Proof. We let $\mathcal{A}$ be a deterministic Muller automaton over V with 3 states recognising a language L_G for some graph G. We show that we can define a colouring of G using as colours the states of $\mathcal{A}$, so $\chi(G) \leq 3$. In all the proof, we overlook the output colours of

[1] We know... Not the most beautiful proof, but effective.

the automaton, as we will only make use of Lemma III.13 and Remark III.14 to reason about the acceptance of runs of $\mathcal{A}$.

We let $n(v) = N[v] \setminus \{v\}$ be the *open neighbourhood* of v in G. We recall that a set S of states is X-closed, for $X \subseteq V$, if no transition labelled with a letter in X leaves S. Also, a subgraph $\mathcal{S}$ of $\mathcal{A}$ is X-FSCC if it is a final SCC of the restriction of $\mathcal{A}$ to transition labelled with letters in X. By a small abuse of terminology, we say that a subset $X \subseteq V$ is an X-FSCC if it is the set of states of one, and we omit the brackets for singletons.

We will extensively use the fact that if q, q' are two states in a X-FSCC, we can go from q to q' reading a word $w \in X^*$ that contains all factors xx for $x \in X$. We can moreover suppose that such a path visits all edges of the X-FSCC.

✧ Claim III.15.1. *No $n(v)$-FSCC contains a v-closed subset. In particular, we can leave any $n(v)$-FSCC by reading v^*, and a $n(v)$-FSCC does not contain a state q with a v-self loop.*

Proof. Suppose on the contrary that a subgraph $\mathcal{S}$ of $\mathcal{A}$ is a $n(v)$-FSCC and it contains a subset S' v-closed. Let $S' = \{q_1, \dots, q_k\}$, and take k words $w_i \in n(v)^+$ such that w_i serves to label a path of the form $q_i \xrightarrow{v} \overset{w_i}{\rightsquigarrow} q_{i+1}$, while visiting all the edges of $\mathcal{S}$ (we let $q_{k+1} = q_1$). Then, $(vw_1 \dots vw_k)^\omega \in L_v$, so a run visiting all the edges in $\{\text{edges of } \mathcal{S}\} \cup \{q' \xrightarrow{v}| \ q' \in S'\}$ is accepting. However, we can build in a similar way a run that visits the same set of edges and labelled with $(vvw'_1 \dots vvw'_k)^\omega$, where w'_i is a word in $n(v)^+$ containing all factors xx for $x \in n(v)$. By Lemma III.13, such a word must be rejected, a contradiction. ◁

We obtain that the 3 states of $\mathcal{A}$ – that we name q_1, q_2 and q_3 – do not form a $n(v)$-FSCC for any $v \in V$. Therefore, for no $v \in V$ the automaton $\mathcal{A}$ contains a full cycle labelled with v ($q_1 \xrightarrow{v} q_2 \xrightarrow{v} q_3 \xrightarrow{v} q_1$). Also, it cannot be the case that, for some $v \in V$, the three states have a self-loop $q \xrightarrow{v} q$, as some $n(v)$-FSCC exists (Lemma I.4). We will show that, for each $v \in V$, one (and only one) of the following cases occurs:

(A) There are two different states q_1, q_2 such that $q_1 \xrightarrow{v} q_2 \xrightarrow{v} q_1$ and the third state admits a transition $q_3 \xrightarrow{v} q_1$.

(B) There is a unique state q_1 admitting a v-self loop $q_1 \xrightarrow{v} q_1$, and there are not both transitions $q_2 \xrightarrow{v} q_3$ and $q_3 \xrightarrow{v} q_2$.

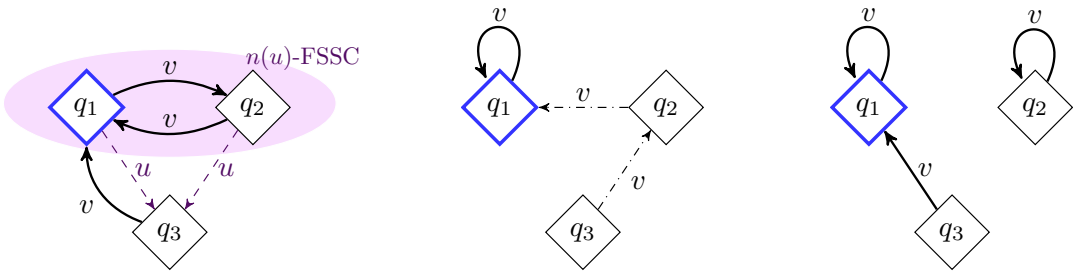
(C) There are two states q_1, q_2 admitting a v-self loop $q_i \xrightarrow{v} q_i$, and the third state admits a transition $q_3 \xrightarrow{v} q_1$.

We show the three possible configurations in Figure 21.

This will allow us to define a colouring $c\colon V \to Q$ associating a vertex v the state that corresponds to q_1, when numbering the states from the point of view of v, according to the case applying to this vertex.

✧ Claim III.15.2. *If there are two different states q_1, q_2 such that $q_1 \xrightarrow{v} q_2$, $q_2 \xrightarrow{v} q_1$, then $\{q_1, q_2\}$ is a $n(u)$-FSCC for every u such that $(v, u) \in E$.*

Proof. Let $u \in V$ be a neighbour of v and let q_3 be the third state of $\mathcal{A}$. Suppose by contradiction that $\{q_1, q_2\}$ is not a $n(u)$-FSCC. (Warning: the configurations considered in this proof do not correspond any more to that of Figure 21!) First, the three states of $\mathcal{A}$ do not form a $n(u)$-FSCCs by the previous claim, and, since the two states are connected by v-transitions, neither q_1 nor q_2 can be contained in one (as $\{q_1, q_2\}$ or all of Q would also form a $n(u)$-FSCCs). Therefore, as some $n(u)$-FSCC exists



✦ **Figure 21.** The three possible configurations of $\mathcal{A}$. In bold, the transitions that are established by the definition of each case. For Case (A), we will show that for every neighbour u of v, the states $\{q_1, q_2\}$ form a $n(u)$-FSCC and there are transitions $q_1 \xrightarrow{u} q_3$ and $q_2 \xrightarrow{u} q_3$ (dashed edges). In Case (B), the dashed-and-dotted lines represent transitions that are compatible with this case, but they are not the only possibility. In all the cases, the state q_1 (in blue) is chosen to define $c(v) = q_1$ (note that the numbering of states depends on v).

(Lemma I.4), $\{q_3\}$ is the only one. In particular, q_3 has a v-self loop $q_3 \xrightarrow{v} q_3$. Also, as $\{q_3\}$ it cannot be u-closed, there is a u-transition leaving q_3, that we suppose w.l.o.g that goes to q_1: $q_3 \xrightarrow{u} q_1$. We note that either q_1 or q_2 contains a u-self-loop. Indeed, the automaton contains a $n(v)$-FSCC that cannot contain q_3 (it has a v-self loop) nor be $\{q_1, q_2\}$ (it is v-closed). We treat the case $q_2 \xrightarrow{u} q_2$, the other case is analogous.

Let $w \in n(u)^*$ such that $q_2 \overset{w}{\rightsquigarrow} q_3$ (this word exists as $\{q_1, q_2\}$ is not $n(u)$-closed). We let $w' \in n(u)^*$ containing all factors xx for $x \in n(u)$. Then, we compare the runs:

$$q_3 \xrightarrow{u} q_1 \xrightarrow{v} q_2 \xrightarrow{u} q_2 \overset{w}{\rightsquigarrow} q_3 \overset{w'}{\rightsquigarrow} q_3, \qquad q_3 \xrightarrow{u} q_1 \xrightarrow{v} q_2 \overset{uu}{\rightsquigarrow} q_2 \overset{w}{\rightsquigarrow} q_3 \overset{w'}{\rightsquigarrow} q_3.$$

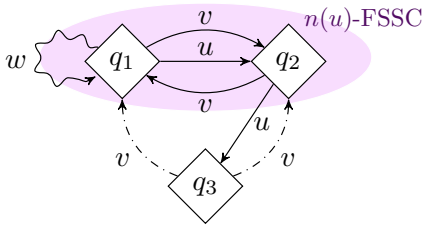
The first of these cycles is accepting (as $(uvuww')^\omega \in L_u$) while the second is rejecting (by Lemma III.13 applied to u), but they visit the same set of edges, a contradiction. ◁

✧ Claim III.15.3. *Each vertex $v \in V$ is in one and only one of the cases (A), (B) or (C).*

Proof. By definition, the three cases are pairwise incompatible. We have already shown that no v-cycle visits the three states and that there is at least one state without a v-self loop, so, to see that a vertex v is in one of the cases, it suffices to see that, if there are two states such that $q_1 \xrightarrow{v} q_2 \xrightarrow{v} q_1$, then we are in Case (A), that is, q_3 does not admit a v-self loop. By Claim III.15.1, $\{q_1, q_2\}$ is a $n(u)$-FSCC. The automaton contains some $n(v)$-FSCC, that cannot be contained in $\{q_1, q_2\}$ (as we can leave this subset reading u^* by Claim III.15.1). Therefore, q_3 is contained in such a $n(v)$-FSCC, so it cannot admit a v-self loop. ◁

✧ Claim III.15.4. *If v is in Case (A), then for every u such that $(v, u) \in E$, $q_1 \xrightarrow{u} q_3$ and $q_2 \xrightarrow{u} q_3$.*

Proof. First, we recall that there is a transition $q_3 \xrightarrow{v} \{q_1, q_2\}$ (Claim III.15.3) and that $\{q_1, q_2\}$ is a $n(u)$-FSCC (Claim III.15.2). Suppose on the contrary that there was a transition $q_1 \xrightarrow{u} q_2$ (the other case is symmetric). Then, since $\{q_1, q_2\}$ is not u-closed, there is the transition $q_2 \xrightarrow{u} q_3$. The situation is depicted in Figure 22.



✦ **Figure 22.** Situation in the proof of Claim III.15.4. The dashed transitions $q_3 \xrightarrow{v} q_1$, $q_3 \xrightarrow{v} q_2$ mean that one of these two transitions exists. The word $w \in n(u)^*$ is a word that contains all factors xx for $x \in n(u)$.

Let $w \in n(u)^*$ producing all factors xx for $x \in n(u)$ and labelling a cycle $q_1 \overset{w}{\rightsquigarrow} q_1$ (such word exists because q_1 is contained in a $n(u)$-FSCC). We compare the following cycles:

$$q_1 \overset{w}{\rightsquigarrow} q_1 \overset{uvv}{\rightsquigarrow} q_2 \xrightarrow{u} q_3 \overset{v^+}{\rightsquigarrow} q_1, \quad q_1 \overset{w}{\rightsquigarrow} q_1 \overset{uvvv}{\rightsquigarrow} q_1 \overset{uu}{\rightsquigarrow} q_3 \overset{v^+}{\rightsquigarrow} q_1.$$

The first cycle is accepting, while the second is rejecting, but they visit the same set of edges, a contradiction. ◁

As advanced earlier, we define a colouring $c\colon V \to Q$ associating a vertex v to the state that corresponds to q_1 in the (only) case applying to v. We show that this is a correct colouring. Let $u \in V$ such that $(v,u) \in E$. We enumerate the states of $\mathcal{A}$ from the point of view of v in the separation of cases above, so $Q = \{q_1, q_2, q_3\}$ and $c(v) = q_1$.

If v is in Case (A), then $c(u) = q_3 \neq q_1$: By Claim III.15.4, there are transitions $q_1 \xrightarrow{u} q_3$ and $q_2 \xrightarrow{u} q_3$. It is not possible for u to be in Case (C). If u is in Case (B), the only state that can admit a u-self loop is q_3, so $c(u) = q_3$. Finally, suppose that u is in Case (A). As $q_3 \xrightarrow{v} q_1$, we can apply Claim III.15.4 from the perspective of u, obtaining that the configuration $q_1 \xrightarrow{u} q_3 \xrightarrow{u} q_1$ is not possible. Therefore, the only remaining possibility is $q_2 \xrightarrow{u} q_3 \xrightarrow{u} q_2$ and $q_1 \xrightarrow{u} q_3$, so $c(u) = q_3$.

We suppose from now on that neither v nor u is in Case (A). In particular, there are self-loops $c(v) \xrightarrow{v} c(v)$ and $c(u) \xrightarrow{u} c(u)$.

If v is in Case (B) and u is not in Case (A), then $c(v) \neq c(u)$: We show that q_1 does not admit a u-self loop, so $c(u) \neq q_1$. As v is in Case (B), q_1 must belong to a $n(u)$-FSCC. In particular, it does not admit a u-self loop (Claim III.15.1).

If v is in Case (C), $c(v) \neq c(u)$: We show that q_1 does not admit a u-self loop, so $c(u) \neq q_1$. If v is in Case (C), the state q_3 must constitute the unique $n(v)$-FSCC of $\mathcal{A}$ (since one such FSCC must exists, and q_1 and q_2 cannot belong to it as they have v-self loops). Therefore, we can go from q_1 to q_3 reading a word $w \in n(v)^*$. Let $w' \in n(v)^*$ producing all the factors xx, for $x \in n(v)$. If we had $q_1 \xrightarrow{u} q_1$, we could build the following two cycles:

$$q_1 \overset{w}{\rightsquigarrow} q_3 \overset{w'}{\rightsquigarrow} q_3 \xrightarrow{v} q_1 \xrightarrow{u} q_1 \xrightarrow{v} q_1, \quad q_1 \overset{w}{\rightsquigarrow} q_3 \overset{w'}{\rightsquigarrow} q_3 \xrightarrow{v} q_1 \xrightarrow{u} q_1 \xrightarrow{vv} q_1.$$

The first has to be accepting and the second rejecting, but they visit the same set of edges, a contradiction.

This covers all the possible cases. ◀

Lemmas III.11, III.12 and III.15 justify that the problem 3-COLORABILITY reduces in polynomial time to the problem MINIMISATION OF GENERALISED COBÜCHI AUTOMATA.

2.3 Discussion: The case of history-deterministic generalised Büchi automata

As mentioned in the introduction, Abu Radi and Kupferman gave a procedure to minimise history-deterministic coBüchi automata in polynomial time [AK22]. Due to the semantics of non-determinism, this result does not dualise, and the problem of the minimisation of HD Büchi automata is still open (in fact, we conjecture that it is NP-complete, see Conjecture VII.2). As we have been able to establish the complexity of the minimisation of deterministic generalised (co)Büchi automata, the natural next question is what is the complexity of this problem for history-deterministic automata. We conjecture that the algorithm of Abu Radi and Kupferman can be extended to generalised coBüchi automata, whereas the minimisation of generalised Büchi automata is NP-complete.[2]

Conjecture III.3 (Tractability of minimisation of HD generalised coBüchi automata)**.**

Given a history-deterministic generalised coBüchi automaton $\mathcal{A}$, we can find in polynomial time a minimal history-deterministic generalised coBüchi automaton recognising $\mathcal{L}(\mathcal{A})$.

Conjecture III.4 (Hardness of minimisation of HD generalised Büchi automata)**.**

The problem of minimising history-deterministic generalised Büchi automata is NP-complete.

In fact, we believe that the reduction shown in Section III.2.2 can be applied to HD generalised Büchi automata by using the complement language $\overline{L_G}$. We show now that the same reduction cannot give the NP-harness of the minimisation of HD coBüchi automata, by giving an example of a small such automaton.

Let Σ be a finite alphabet, and define the language $L_{\exists\text{noRep}}$ as:

$$L_{\exists\text{noRep}} = \{w \in \Sigma^\omega \mid \text{ for some } a \in \Sigma,\ w \text{ eventually does not contain the factor } aa\}.$$

We remark that $L_{\exists\text{noRep}}$ corresponds to the language L_G for a clique G of size $|\Sigma|$. Conjecture III.2 states that a deterministic generalised coBüchi automaton for $L_{\exists\text{noRep}}$ has size at least $|\Sigma|$, and we have proved that result for $|\Sigma| = 3$ (Lemma III.15).

[2] Just before the submission of this document, we obtained together with Olivier Idir, Denis Kuperberg, Corto Mascle and Aditya Prakash that both Conjectures III.3 and III.4 hold. For reasons of time and extension of the thesis, we do not include these proofs here and still refer to the statements as conjectures.

Proposition III.16.

For all Σ, there exists a history-deterministic generalised coBüchi automaton with two states recognising $L_{\exists\text{noRep}}$.

Proof. We describe one such automaton. Let $\Sigma = \{a_1, \dots, a_n\}$ be an enumeration of the letters in Σ. We define $\mathcal{A}_c = (Q, \Sigma, q_0, \Gamma, \Delta, \mathsf{genCoBuchi}_\Gamma(B))$ as follows:

- $Q = \{q_0, q_1\}$.
- $\Gamma = \{b_1, \dots, b_n\} \cup \{\gamma\}$.
- For all $a_i \in \Sigma$, $q_0 \xrightarrow{a_i:b_i} q_0$ and $q_0 \xrightarrow{a_i:\gamma} q_1$.
- For all $a \in \Sigma$, $q_1 \xrightarrow{a_i:b_i} q_0$.
- $B = \{B_i \mid 1 \leq i \leq n\}$, where $B_i = \{b_i\}$. That is, a run is accepting if it avoids one of the colours b_i. Colour γ is a *safe colour*: it is not included in any B_i.

That is, when we are in the state q_0 and read letter a_i, we can choose to stay in q_0 and produce output b_i "as a payment", or jump to stay q_1 producing the safe colour γ.

$\mathcal{L}(\mathcal{A}) \subseteq L_{\exists\text{noRep}}$**:** Each time that a factor a_ia_i is read, the colour b_i is produced as output in $\mathcal{A}$. Therefore, if a word w is accepted by $\mathcal{A}$ by avoiding the set B_i, the word w does not contain the factor a_ia_i infinitely often.

$L_{\exists\text{noRep}} \subseteq \mathcal{L}(\mathcal{A})$ **and history-determinism:** We describe a sound resolver for $\mathcal{A}$ accepting words in $L_{\exists\text{noRep}}$. The resolver will try to build a run avoiding colour b_i, switching of b_i in a round robin fashion. The strategy of the resolver consists in n states of memory. When it is in the memory state i, it will choose to stay in q_0 paying b_j when reading $a_j \neq a_i$, and it will choose to go to q_1 when reading a_i. Whenever it reads letter a_i in the state q_1 (that is, we have just read the factor a_ia_i, the memory state updates to $i+1$ (or reinitialises to 1 if $i = n$). If $w \in L_{\exists\text{noRep}}$, this resolver will build an accepting run over w whenever we arrive to a memory state i such that a_ia_i does no longer appear in w. ◀

✦ Remark III.17. *We note that, as mentioned in Remark III.1, the previous result does not dualise to history-deterministic generalised Büchi automata. The reader can verify that the non-deterministic automaton obtained by interpreting the acceptance condition of the given automaton as a generalised Büchi one does not recognise the complement of the language* $L_{\exists\text{noRep}}$.

3 Minimisation of parity automata recognising Muller languages in polynomial time

In this section, we provide a polynomial-time algorithm for the minimisation of DPA recognising Muller languages (with acceptance condition over transitions).

Theorem III.4.

Given a DPA $\mathcal{A}$ recognising a Muller language $L = \mathsf{Muller}_\Sigma(\mathcal{F})$, we can find a minimal deterministic (resp. history-deterministic) parity automaton recognising L in polynomial time in the size of the representation of $\mathcal{A}$.

By Proposition II.34 and Theorem II.2, we know that a minimal (history-)deterministic parity automaton recognising a Muller language $L = \mathsf{Muller}_\Sigma(\mathcal{F})$ can be constructed in linear time from the Zielonka tree $\mathcal{Z}_\mathcal{F}$. We will therefore provide a polynomial-time algorithm computing this Zielonka tree from a DPA recognising L.

Proposition III.18 (Construction of the Zielonka tree from a DPA)**.**

Given a DPA $\mathcal{A}$ recognising a Muller language $\mathsf{Muller}_\Sigma(\mathcal{F})$, we can build the Zielonka tree of $\mathcal{F}$ in polynomial time in the size of the representation of $\mathcal{A}$.

The algorithm we give now is similar to the algorithm computing the alternating cycle decomposition presented in Section II.5.2, which in turn generalises the algorithm of Carton and Maceiras [CM99]. Here, the input is a parity automaton recognising $\mathsf{Muller}_\Sigma(\mathcal{F})$, so $\mathcal{A}$ has an underlying graph whose transitions are labelled with letters in Σ (as it was the case in Section II.5.2). The main difference is that, here, we do not directly have the information of which subsets of Σ belong to $\mathcal{F}$, so we need to inspect the parity acceptance condition of $\mathcal{A}$ to determine the subsets in $\mathcal{F}$.

Description of the algorithm

Let $\mathcal{A} = (Q, \Sigma, q_0, \Gamma, \Delta, \mathsf{parity})$ be a DPA recognising $L = \mathsf{Muller}_\Sigma(\mathcal{F})$. We outline a recursive algorithm building $\mathcal{Z}_\mathcal{F} = (N, \preceq, \nu)$ in a top-down fashion; it starts from the root of the tree (which is always labelled Σ), and each time that some node is added to N, we compute its children. If we have built $\mathcal{Z}_\mathcal{F}$ up to a node n, we compute the children of n by using the procedure `AlternatingSets` described in Algorithm 3, which we disclose next.

We suppose without loss of generality that n is round, that is, $\nu(n) \in \mathcal{F}$. First, we take the restriction of $\mathcal{A}$ to transitions labelled with letters in $\nu(n)$ and pick a final SCC on it. Such final SCC induces a subautomaton $\mathcal{A}'$ of $\mathcal{A}$ recognising $\mathsf{Muller}_{\nu(n)}(\mathcal{F}|_{\nu(n)})$ (see also Lemma II.38). Our objective is to find the maximal subautomata of $\mathcal{A}'$ using as input letters sets $X \subseteq \nu(n)$ such that $X \notin \mathcal{F}$. We will keep all such subsets X in a list $\mathsf{altSets}$. The labels of the children of n will then correspond to the maximal sets appearing in this list, which are returned by the algorithm `AlternatingSets` (Line 13). In order to find them, we remove the transitions using the minimal priority in $\mathcal{A}'$ (that is even, since $\nu(n) \in \mathcal{F}$) and compute a decomposition in strongly connected components of the obtained graph. Let $\mathcal{S}$ be a component of this decomposition and let $\Sigma_\mathcal{S} \subseteq \nu(n)$ be the input letters appearing on it. Then, $\Sigma_\mathcal{S} \notin \mathcal{F}$ if and only if the minimal priority in $\mathcal{S}$ is odd (see Lemma III.19 below). In this case, we add $\Sigma_\mathcal{S}$ to $\mathsf{altSets}$. On the contrary, we remove the minimal (even) priority from $\mathcal{S}$ and we start again finding a decomposition in SCCs of the obtained graph.

We include the pseudocode for the procedure `AlternatingSets` in Algorithm 3. We use the following notations:

- `Letters`$(\mathcal{S})$ is the set of input letters appearing in $\mathcal{S}$,

- `MinColour`($\mathcal{S}$) is the minimal priority appearing in $\mathcal{S}$ (which determines whether `Letters`($\mathcal{S}$) $\in \mathcal{F}$, if $\mathcal{S}$ is strongly connected),
- `SCC-Decomposition`($\mathcal{A}$) outputs a list of the strongly connected components of $\mathcal{A}$. If $\mathcal{A}$ is empty, it outputs an empty list. (Introduced in Section II.5.2.)
- `MaxInclusion`(lst) returns the list of the maximal subsets in lst. (Introduced in Section II.5.2.)

Algorithm 3 `AlternatingSets`($\mathcal{A}$): Computing the children of a node of the Zielonka tree

Input: A strongly connected automaton $\mathcal{A}$ over Σ such that $\mathcal{L}(\mathcal{A}) = \mathsf{Muller}(\mathcal{F})$
Output: The maximal subsets $\Sigma_1, \dots, \Sigma_k \subseteq \Sigma$ such that $\Sigma_i \in \mathcal{F} \iff \Sigma \notin \mathcal{F}$.

```
d ← MinColour(A)
A_{>d} ← restriction of A to transitions Δ_{>d} = {q -a:x-> q' ∈ Δ | x > d}
⟨S_1, ..., S_r⟩ ← SCC-Decomposition(A_{>d})
altSets ← {}
for i = 1, ..., r do
    if MinColour(S_i) is odd if and only if d is even then
        altSets ← altSets ∪ {Letters(S_i)}
    else
        altSets ← altSets ∪ AlternatingSets(S_i)
    end if
end for
maxAltSets ← MaxInclusion(altSets)
return maxAltSets
```

Correctness of the algorithm

Let n be a node of the Zielonka tree of $\mathcal{F}$ labelled with $\nu(n)$, and let $\mathcal{A}_n$ be an accessible subautomaton of $\mathcal{A}$ over $\nu(n)$ recognising $\mathsf{Muller}_{\nu(n)}(\mathcal{F}|_{\nu(n)})$. We prove that `AlternatingSets`($\mathcal{A}_n$) returns a list of sets corresponding to the labels of the children of n in $\mathcal{Z}_\mathcal{F}$. We suppose without loss of generality that $\nu(n) \in \mathcal{F}$ and therefore the minimal priority d in $\mathcal{A}_n$ is even.

First, we observe that if $X \subseteq \Sigma$ is added to altSets during the execution of the procedure `AlternatingSets`, then X is the set of input letters appearing in a cycle whose minimal priority is odd. Next lemma implies that in this case, $X \notin \mathcal{F}$. In particular, no subset is added if n is a leaf of $\mathcal{Z}_\mathcal{F}$.

✦ **Lemma III.19.** *Let $\mathcal{A}$ be a DPA such that $\mathcal{L}(\mathcal{A}) = \mathsf{Muller}_\Sigma(\mathcal{F})$. Let $\ell \in \mathit{Cycles}(\mathcal{A})$ be an accessible cycle of $\mathcal{A}$. Let $\Sigma_\ell \subseteq \Sigma$ be the input letters appearing on ℓ, and let d_ℓ be the minimal priority on ℓ. Then, $\Sigma_\ell \in \mathcal{F}$ if and only if d_ℓ is even.*

Proof. Since ℓ is an accessible cycle, there is a word $w \in \Sigma^\omega$ such that $\mathsf{Inf}(w) = \Sigma_\ell$ and verifying that the edges visited infinitely infinitely often by the (only) run over w in $\mathcal{A}$ are the edges of ℓ. Therefore $w \in \mathcal{L}(\mathcal{A})$ if and only if d_ℓ is even, and since $\mathcal{L}(\mathcal{A})$ is a Muller language, $w \in \mathcal{L}(\mathcal{A})$ if and only if $\mathsf{Inf}(w) = \Sigma_\ell \in \mathcal{F}$. ◀

As the final output of the algorithm consists solely on the maximal subsets in altSets, and no accepting set is added to this list, it suffices to show that each maximal rejecting subset $\Sigma_{\max} \subseteq \nu(n)$ is added to altSets at some point.

Let $\Sigma_{\max} \subseteq \nu(n)$ be one of the maximal rejecting subsets of $\nu(n)$. Let $\mathcal{S}$ be a final SCC of the restriction of $\mathcal{A}_n$ to transitions labelled with letters in $\Sigma_{\max}$ (by the previous lemma, $\texttt{MinColour}(\mathcal{S})$ is odd). We show that $\Sigma_{\max}$ will eventually be considered by the recursive procedure `AlternatingSets`, and therefore $\Sigma_{\max}$ will be added to altSets We use of the following remark:

✧ Claim III.19.1. *If $\mathcal{S}'$ is a strongly connected subautomaton of $\mathcal{A}_n$ such that $\mathcal{S} \subsetneq \mathcal{S}' \subseteq \mathcal{A}_n$, then the minimal priority in $\mathcal{S}'$ is even.*

Proof. Let Σ' be the input letters appearing in $\mathcal{S}'$. As $\mathcal{S} \subsetneq \mathcal{S}'$ and no transition labelled with a letter in $\Sigma_{\max}$ leaves $\mathcal{S}$, we must have $\Sigma_{\max} \subsetneq \Sigma'$. The claim follows from Lemma III.19. ◁

Therefore, either $\mathcal{S}$ is one of the SCCs of $\mathcal{A}_{>d}$ (in this case, $\Sigma_{\max}$ is added to altSets in Line 7), or it is contained in one SCC of $\mathcal{A}_{>d}$ whose minimal priority is even and we can conclude by induction.

Complexity analysis

We will show that the proposed algorithm works in $\mathcal{O}(|Q|^3|\Sigma|^2|\Gamma|)$, where Q, Σ and Γ are the states, set of input letters and set of output colours of the automaton, respectively. We remark that, since $\mathcal{A}$ is deterministic, $|\Delta| \leq |Q||\Sigma|$.

First, we study the complexity of the procedure $\texttt{AlternatingSets}(\mathcal{A})$. At each recursive call, at least one edge is removed from Δ, and a decomposition in strongly connected components of the automaton is performed, which can be done in $\mathcal{O}(|Q||\Sigma|)$ [Tar72]. Therefore, the children of a node of the Zielonka tree can be computed in $\mathcal{O}(|Q|^2|\Sigma|^2)$.

We perform this operation for each node of the Zielonka tree. By the optimality of the ZT-parity-automaton (Theorems II.1 and II.2), we know that $|Q| \geq |\mathsf{Leaves}(\mathcal{Z}_\mathcal{F})|$ and that the height of $\mathcal{Z}_\mathcal{F}$ is at most $|\Gamma|$. Therefore, $|\mathcal{Z}_\mathcal{F}| \leq |Q||\Gamma|$, and the procedure `AlternatingSets` is called at most $|Q||\Gamma|$ times. We conclude that the proposed algorithm works in $\mathcal{O}(|Q|^3|\Sigma|^2|\Gamma|)$.

Minimisation of history-deterministic parity and Rabin automata

In Chapter II, Section II.3, we have shown that we can directly obtain from the Zielonka tree minimal history-deterministic parity and Rabin automata recognising a Muller language. However, this does not suffice to show that we can minimise HD Rabin and parity automata recognising Muller languages in polynomial time, as this approach requires to be able to construct the Zielonka tree of $\mathcal{F}$ from a HD automaton, something that we do not currently know how to do.

Conjecture III.5.

History-deterministic parity and Rabin automata recognising Muller languages can be minimised in polynomial time.

Question III.6.

How can we build the Zielonka tree of a family $\mathcal{F} \subseteq 2^\Sigma_+$ from an HD Rabin or parity automaton recognising $\mathsf{Muller}_\Sigma(\mathcal{F})$?

4 Exponential succinctness of history-deterministic Rabin automata for Muller languages

In the previous chapter, we saw that minimal history-deterministic parity automata recognising Muller languages have the same size as minimal deterministic ones (Corollary II.36). On the contrary, for Rabin automata, complexity considerations already indicate that HD automata should be smaller: building a minimal deterministic Rabin automaton from a given Zielonka tree is NP-hard (Theorem III.1), while we can build a minimal history-deterministic one in polynomial time (Theorem II.4). A natural question is how large can be the gap on the size. In this section, we show that HD Rabin automata recognising Muller languages can be exponentially smaller than deterministic ones.

Theorem III.5.

There exists a constant $\alpha > 1$, a sequence of natural numbers $n_1 < n_2 < n_3 \dots$ and a sequence of Muller languages L_i over the alphabets $\Sigma_{n_i} = \{1, \dots, n_i\}$ such that:

- a minimal HD Rabin automaton for L_i has size $\lfloor n_i/2 \rfloor$,
- a minimal deterministic Rabin automaton for L_i has size at least α^{n_i}.

A lower bound for such a constant is $\alpha = 1.116$.

We devote the rest of this Section to proving Theorem III.5. In brief, the Muller languages in question require half the colours to be seen infinitely often. The construction of the small HD Rabin automaton follows from constructing the Zielonka tree of the languages. For the lower bound on the deterministic Rabin automaton, we reduce the problem to finding a lower bound on the chromatic number of a certain graph.

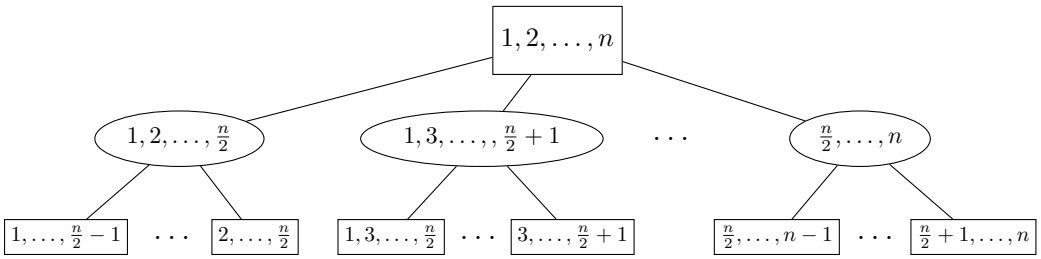
The collection of Muller languages

We define the collection of Muller languages L_i that we use to obtain Theorem III.5.

Let $n \in \mathbb{N}$ and $\Sigma_n = \{1, \dots, n\}$ be an alphabet of n letters. We let L_n be the Muller language associated to the following family:

$$\mathcal{F}_n = \{C \subseteq \Sigma_n \ : \ |C| = \lfloor n/2 \rfloor\}.$$

The Zielonka tree of $\mathcal{F}_n$ is depicted in Figure 23 (for n even).



✦ **Figure 23.** Zielonka tree $\mathcal{Z}_{\mathcal{F}_n}$, for $\mathcal{F}_n = \{C \subseteq \Sigma_n \ : \ |C| = \lfloor n/2 \rfloor\}$.

Upper bounds for history-deterministic Rabin automata

The upper bound for HD Rabin automaton recognising $L_n = \mathsf{Muller}(\mathcal{F}_n)$ follows from the construction of the ZT-HD-Rabin-automaton from Section II.3.3. We recall that the automaton $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_{\mathcal{F}_n}}$ is a (minimal) HD Rabin automaton recognising $\mathsf{Muller}(\mathcal{F}_n)$ having $\mathsf{rbw}(\mathcal{Z}_{\mathcal{F}_n})$ nodes, where $\mathsf{rbw}(\mathcal{Z}_{\mathcal{F}_n})$ is the round-branching width of its Zielonka tree (Theorem II.4).

We show that $\mathsf{rbw}(\mathcal{Z}_{\mathcal{F}_\mathcal{F}}) = \lfloor n/2 \rfloor$. Since the round nodes of the Zielonka tree of $\mathcal{F}_n$ are pairwise incomparable for the ancestor relation, the round-branching width of $\mathcal{Z}_{\mathcal{F}_n}$ is the maximum of children of a round node. A round node is labelled with a subset $C \subseteq \Sigma_n$ such that $|C| = \lfloor n/2 \rfloor$, and it has a child for each subset $X \subseteq C$ such that $|X| = \lfloor n/2 \rfloor - 1$. There are exactly $\lfloor n/2 \rfloor$ such subsets.

Lower bounds for deterministic Rabin automata

Using ideas similar to those introduced in Section III.1, we reduce the problem of searching lower bounds for deterministic Rabin automata to finding lower bounds on the chromatic number of certain graphs. By making use of known results on graph theory [MR14], we finally show that the chromatic number of these graphs is sufficiently large. We refer to Appendix A for definition about the chromatic number of undirected graphs.

The graph of rejecting subsets of a Muller condition. Let $\mathcal{F} \subseteq 2^\Sigma_+$ be a family of subsets of letters. We define the *graph of rejecting subsets associated to* $\mathcal{F}$ as the simple undirected graph $G_\mathcal{F} = (V_\mathcal{F}, E_\mathcal{F})$ defined as follows:[3]

- $V_\mathcal{F} = 2^\Sigma_+$.
- There is an edge between two subsets $X_1, X_2 \in V_\mathcal{F}$ if and only if $X_1, X_2 \notin \mathcal{F}$ and $X_1 \cup X_2 \in \mathcal{F}$.

That is, we connect two vertices if they correspond to rejecting sets but taking their union we obtain an accepting set.

Lower bounds of Rabin automata given by the chromatic number. The following result allows us to reduce the search for lower bounds for Rabin automata to lower bounds on the chromatic number of $G_\mathcal{F}$, written $\chi(G_\mathcal{F})$.

Lemma III.20 (Lower bounds for Rabin automata)**.**

Let $\mathcal{A}$ be a deterministic Rabin automaton recognising a Muller language $\mathsf{Muller}_\Sigma(\mathcal{F})$. It is satisfied:

$$\chi(G_\mathcal{F}) \leq |\mathcal{A}|.$$

Proof. Let Q be the set of states of $\mathcal{A}$. We define a colouring $c\colon V_\mathcal{F} \to Q$ of $G_\mathcal{F}$ using Q as colours. For each subset $X \subseteq \Sigma$, we let $\mathcal{S}_X$ be an accessible X-FSCC (which exists by Lemma I.4, as we can suppose that $\mathcal{A}$ is complete by prefix-independence of $\mathsf{Muller}(\mathcal{F})$) and we pick an arbitrary state $q_X \in \mathcal{S}_X$. We define $c(X) = q_X$. We prove that this is a correct colouring. Suppose that X_1 and X_2 are two vertices in $G_\mathcal{F}$ connected by some edge, that is, $X_1, X_2 \notin \mathcal{F}$ and $X_1 \cup X_2 \in \mathcal{F}$. If $q_{X_1} = q_{X_2}$, the edges of $\mathcal{S}_1$ and $\mathcal{S}_2$ form

[3] As in other places of this thesis, we do not include the dependence on the alphabet Σ in the notation.

two rejecting cycles with an state in common such that their union is an accepting cycle, contradicting Proposition II.109. ◀

The graph $G_{\mathcal{F}_n}$ of our collection of Muller languages. We describe now what are the graphs of rejecting subsets associated to $\mathcal{F}_n$ for the families $\mathcal{F}_n$ considered above, and show that the chromatic number $\chi(G_{\mathcal{F}_n})$ is exponential in n.

Applying the definition of graph of rejecting subsets in this case gives:

- $V_{\mathcal{F}_n} = 2^{\Sigma}_{+}$.
- There is an edge between two subsets $X_1, X_2 \in V_{\mathcal{F}_n}$ if and only if $|X_1| < \lfloor n/2 \rfloor$, $|X_2| < \lfloor n/2 \rfloor$ and $|X_1 \cup X_2| = \lfloor n/2 \rfloor$.

Proposition III.21.

There exists a constant $\alpha > 1$ and a sequence of natural numbers $n_1 < n_2 < n_3 \dots$ such that $\alpha^{n_i} \leq \chi(G_{\mathcal{F}_{n_i}})$.

In order to prove Proposition III.21, we introduce some further graph-theoretic notions. Let $G = (V, E)$ be an undirected graph. An *independent set* of G is a set $S \subseteq V$ such that $(v, v') \notin E$ for every pair of vertices $v, v' \in S$.

✦ **Lemma III.22.** *Let $V' \subseteq V$, and let $G' = (V', E|_{V' \times V'})$ be the subgraph of G induced by V'. Then, $\chi(V') \leq \chi(V)$.*

✦ **Lemma III.23.** *Let m be an upper bound on the size of the independent sets in G. Then*

$$\frac{|V|}{m} \leq \chi(G).$$

Proof. Let $c\colon V \to \Lambda$ be a colouring of G with $|\Lambda| = \chi(G)$. Then, by definition of a colouring, for each $x \in \Lambda$, $c^{-1}(x)$ is an independent set in G, so $|c^{-1}(x)| \leq m$. Also, $V = \bigsqcup_{x \in \Lambda} c^{-1}(x)$, so

$$|V| = \sum_{x \in \Lambda} |c^{-1}(x)| \leq \chi(G) \cdot m.$$ ◀

We will find a subgraph of $G_{\mathcal{F}_n}$ for which we can provide an upper bound on the size of its independent sets. The upper bound is provided by the following theorem (adapted from [MR14, Theorem 15]).

Proposition III.24 ([MR14, Theorem 15])**.**

Let $n > k > 2t$ such that $k - t$ is a prime number. Suppose that $\mathcal{B}$ is a family of subsets of size k of Σ_n such that $|A \cap B| \neq t$ for any pair of subsets $A, B \in \mathcal{B}$. Then,

$$|\mathcal{B}| \leq \binom{n}{k-t-1}.$$

We conclude this section with the proof of Proposition III.21.

Proof of Proposition III.21. Let p be a prime number and let $n = 5p$. We will study the subgraph of $G_{\mathcal{F}_n}$ consisting in the subsets of size exactly $k = \lfloor 3n/10 \rfloor$. We denote

this subgraph by $H_{n,k}$. Two subsets $A, B \subseteq \Sigma_n$ of size k satisfy that $|A \cup B| = \lfloor n/2 \rfloor$ if and only if $|A \cap B| = \lfloor n/10 \rfloor$. We set $t = \lfloor n/10 \rfloor$. We get $k - t = p$, so we can apply Theorem III.24 and we obtain that any independent set in $H_{n,k}$ has size at most $\binom{n}{\frac{1}{5}n-1}$. By Lemma III.23, $\chi(H_{n,k}) \geq \binom{n}{\lfloor \frac{3}{10}n \rfloor} / \binom{n}{\frac{1}{5}n-1}$. By Lemma III.22, we know that this lower bound also holds for $G_{\mathcal{F}_n}$. Using Stirling's approximation we obtain that

$$\chi(G_{\mathcal{F}_n}) \geq \binom{n}{\lfloor \frac{3}{10}n \rfloor} / \binom{n}{\frac{1}{5}n-1} = \Omega\left(\left(\frac{(1/5)^{1/5}(4/5)^{4/5}}{(3/10)^{3/10}(7/10)^{7/10}}\right)^n\right) = \Omega(1.116^n).$$

To conclude, we take an enumeration of prime numbers, $p_1 < p_2 < \dots$ and we set $n_i = 5p_i$. ◀

✦ Remark III.25 (Choices of k and t). *The choices of $k = \lfloor 3n/10 \rfloor$ and $t = \lfloor n/10 \rfloor$ in the previous proof might appear quite enigmatic. We try to explain them now.*

We want to find a number k such that there is not a big family of sets $\{A_i \subseteq \Sigma_n\}$ of size $|A_i| = k$ such that $|A_i \cup A_j| \neq n/2$, and express this fact in terms of $|A_i \cap A_j|$. Since $|A \cup B| = 2k - |A \cap B|$, if we define $t = 2k - n/2$, then $|A \cup B| \neq n/2$ if and only if $|A \cap B| \neq t$, so the value of t will be completely determined by the choice of k. Our objective is to minimise the upper bound given in Theorem III.24 (what we do by minimising $k-t$) while making sure that the hypothesis $k > 2t$ is verified. In the boundary of this condition ($k = 2t$) we obtain $k = n/3$, so we express our choices as $k = (1/3-\varepsilon)n$ and $t = (1/6-2\varepsilon)n$. Moreover, $k-t = (1/6+\varepsilon)n$ has to be a prime number (for infinitely many n). If $1/6+\varepsilon = 1/q$ for some $q \in \mathbb{N}$, we would succeed by considering n of the form $q \cdot p$, for p a prime number. We will therefore take $\varepsilon = \frac{6-q}{6q}$, for some q, $1 \leq q \leq 5$. With the optimal choice, $q = 5$, we obtain $k = 3n/10$, $t = n/10$ and $k - t = n/5$. Since k and t will not be integers for n of the form $5p$ (p a prime number) we are forced to take the integer part in the proof of Proposition III.21.

A tight correspondence between memory and automata

IV

Outline

Those who cannot remember the past are condemned to repeat it.
George Santayana

Introduction

Strategy complexity. Several parameters are relevant for solving games: their size, of course, but also their winning condition and the complexity of winning strategies. A measure of this complexity is the memory used by a strategy. A strategy uses a finite amount of memory if the information that we need to retain from the past can be summarised by a finite state machine that processes the sequence of moves played in the game. In this case, the amount of memory used by the strategy is the number of states

of this machine. Given an objective W, a fundamental question is what is the minimal quantity m such that if Eve wins a W-game, she has a winning strategy using a memory of size m (we call m the memory requirements of W). In addition to its size, a memory also has some structure, which further elucidates the game dynamics. Understanding both the size and the structure of memories for W is a crucial step to design algorithms for solving W-games.

Objective.

Give a structural description of the optimal memory in W-games.

Muller languages. In this chapter, we focus on the study of memories for games using Muller languages as winning condition. This class of languages is of special relevance: from a theoretical point of view, Muller languages can be considered one of the "building blocks" of ω-regular languages; from a practical one, the games obtained as an intermediate step from the state-of-the-art LTL-synthesis tools are Muller games [DL+22, LMS20]. Memory requirements for Muller languages have been studied in depth by Dziembowski, Jurdziński and Walukiewicz [DJW97]. They provide a "formula" for computing memory requirements of a given Muller objective, based on the Zielonka tree [Zie98] (see Section II.3.1 for its definition). Our results extend and complement this characterisation.

Different models of memory. In the study of memory for games, some choices can be made in the model used, giving rise to a few different notions. Most prominently, one may restrict to chromatic memories, meaning those that record only the colours that have appeared so far and not the exact sequence of edges, and one may include uncoloured edges (ε-edges) in the game. Chromatic memories were first introduced by Kopczyński [Kop06, Kop08], where he already pointed out a notable advantage: since a chromatic memory only depends on the set of colours, a same structure can be used in multiple games. He left open the question of whether the general and the chromatic memory requirements always coincide. It was also Kopczyński who remarked that the use of ε-edges might affect the memory requirements of winning conditions [Kop08], but he left open the question of whether this was actually the case.[1]

Contributions

1. **Separation of notions of memory.** Our first contribution is to provide examples of Muller languages that separate the different notions of memory considered in the literature. We show that the chromatic memory requirements can be exponentially larger than general ones (Proposition IV.5) and that the presence of uncoloured edges in games do affect the memory requirements of conditions (Proposition IV.7).
2. **Correspondence: Chromatic memory $\leftrightarrow$ Deterministic Rabin automata.** We show that chromatic memory structures for a Muller language W exactly correspond to deterministic *transition-based* Rabin automata recognising this language (Theorem IV.1).
3. **NP-completeness of finding the chromatic memory requirements.** As a consequence of the previous point, we obtain that computing the least chromatic

[1] In [Kop08, Section 2.5], this question is stated for half-positionality, question that is still open (see Conjecture IV.1).

memory necessary for playing optimally in all W-games is NP-complete, even if the Muller language W is represented by its Zielonka tree (Theorem IV.2). We remark that both the membership in NP and the hardness are non-trivial without the aforementioned characterisation.

4. **Correspondence: General memory ↔ Good-for-games Rabin automata.**[2] We show that the general memory requirements of a Muller language W coincide with the size of a minimal GFG Rabin automaton recognising W (Theorem IV.3). This description shows that GFG automata play a key, and, up till now, unexplored role in understanding the complexity of Muller languages and that this role is – in some respect – even more important than that of the more classical deterministic automata.

Overall, our contributions supplement our understanding of Muller languages and provide a bridge between the theory of automata and the study of strategy complexity that allows us to lift results from one area to the other. This is exactly what we have done for obtaining the NP-completeness of determining the chromatic memory requirements of a Muller language and for showing the exponential gap between deterministic and GFG Rabin automata (resp. between general and chromatic memory requirements). Moreover, our results highlight the – so far unexplored – fundamental role of GFG automata in the equation. Indeed, up to now GFG automata had mainly been studied for their succinctness, expressivity or algorithmic properties. Here, we shed light on a novel dimension of this automata class, by showing that they capture the exact memory requirements of Muller languages.

Related work

Besides the already mentioned works [Kop06, Kop08, DJW97, Zie98], many other authors have shown interest in the memory requirements of different families of objectives. The Zielonka tree was used by Horn to characterise the memory requirements of Muller conditions when randomised strategies are allowed [Hor09]. Colcombet, Fijalkow and Horn characterised the general memory requirements of open objectives [CFH14].

In the context of his PhD thesis [Van23], Vandenhove and co-authors have conducted a thorough research into the potential and limits of chromatic memory. They have characterised objectives for which both players can play optimally using finite chromatic memory both over finite games [Bou+20] and infinite games [BRV23]. They have also characterised the chromatic memory requirements of open and closed objectives, showing that for these classes of objectives, it is also NP-complete to determine this parameter [Bou+23].

The differences between chromatic and general memory over finite games have also been studied by Kozachinskiy. He provides a strong separation result [Koz22b]: there is a (non-ω-regular) objective W whose general memory requirements over finite games is 2, while its chromatic memory requirements are infinite. In a related work, he proves that, if in a game of size n Eve can win using a general memory structure of size m, then she can win with a chromatic memory of size $(m+1)^n$, and this bound is tight [Koz22d]. This improves a result from Le Roux [Rou20].

Very recently, Ohlmann and the author of this thesis have characterised both the

[2] We recall that an automaton is good-for-games if and only if it is history-deterministic (Proposition II.5). In this chapter, we employ the term good-for-games as the use we make of these automata is closer to this definition.

general[3] and the chromatic memory requirements of all objectives by means of monotone universal graphs [CO23]. This characterisation extends Ohlmann's work on half-positionality [Ohl23a].

For related work concerning positionality, see the introduction of Chapter V.

Prerequisites and links with other chapters

The main notions used in the statements of the contributions of this chapter – which refer mostly to memory requirements – are defined in the first section of the chapter. The other central notions used are those of Rabin automata and good-for-gameness (we recall that this latter notion is equivalent to history-determinism).

However, the proofs of this chapter heavily rely on previous results, as our contributions can be seen as corollaries of the work carried out previously (which we see as part of the beauty and interest of the chapter). To determine the general memory requirements of examples, we intensively use the Zielonka tree and its round-branching width (Definitions II.24 and II.41). Theorem IV.1 is a consequence of Proposition II.109 (typeness of Rabin automata), Theorem IV.2 relies in the NP-completeness of minimisation of Rabin automata (Theorem III.1), and Theorem IV.3 is a direct application of the construction presented in Section II.3.3 (minimal HD Rabin automata for Muller languages).

Collaborators and related publications

Section IV.3 is based in joint work with Thomas Colcombet and Karoliina Lehtinen. I thank Igor Walukiewicz and Nathanaël Fijalkow for interesting discussions in the subject and bringing up the question of the exponential gap between chromatic and general memory. This chapter includes material from the conference papers [Cas22] and [CCL22].

1 Different models of memory

In this section we introduce and compare various notions of memory requirements for objectives. The differences in these notions come from two factors:

- The family of games under consideration.
- Restrictions on the memory structures.

In Table 1, we summarise the notations used to represent the different types of memory requirements of objectives, based on these two factors.

1.1 Different types of game graphs

We introduce the main types of game graphs that will be studied. We notice that we do not mention the cardinality or finiteness of these games (although, in general, this is an important parameter to take into account). In this chapter we focus on Muller languages, whose memory requirements are the same over finite and infinite games [DJW97]. For this reason, we disregard this difference in the following and allow both finite and infinite game graphs.

[3] The exact result concerning general memory is a bit more subtle, as we characterise the ε-memory, as introduced in Section IV.1.

Memory / Game	General	Chromatic	Arena-Independent
Arbitrary	$\mathsf{mem}_{\mathsf{gen}}(W)$	$\mathsf{mem}_{\mathsf{chr}}(W)$	$\mathsf{mem}_{\mathsf{ArInd}}(W)$
ε-free	$\mathsf{mem}^{\varepsilon\text{-free}}_{\mathsf{gen}}(W)$	$\mathsf{mem}^{\varepsilon\text{-free}}_{\mathsf{chr}}(W)$	$\mathsf{mem}^{\varepsilon\text{-free}}_{\mathsf{ArInd}}(W)$
Vertex-coloured	$\mathsf{mem}^{\mathsf{vrt}}_{\mathsf{gen}}(W)$	$\mathsf{mem}^{\mathsf{vrt}}_{\mathsf{chr}}(W)$	$\mathsf{mem}^{\mathsf{vrt}}_{\mathsf{ArInd}}(W)$

✦ **Table 1.** Different types of memory requirements arising from the combinations of a type of game graph and a type of memory structure.

Arbitrary games. By default, games are as introduced in the general preliminaries. That is, they are represented as a tuple $\mathcal{G} = (G, V_{\text{Eve}}, V_{\text{Adam}}, \Gamma, \mathsf{col}\colon E \to \Gamma \cup \{\varepsilon\}, W)$. In particular:

- The colours of the winning condition appear on the edges of the game.
- Uncoloured edges are allowed, as long as no infinite path in $\mathcal{G}$ is composed exclusively of them.
- The underlying graph might be finite or infinite.

We sometimes call these *arbitrary games*.

ε-free games. We recall that a game $\mathcal{G}$ is ε-free if no edge is uncoloured, that is:

$$\text{For all } e \in E, \quad \mathsf{col}(e) \neq \varepsilon.$$

Vertex-coloured games. A *vertex-coloured game* is a game in which the colours of the winning condition appear on the vertices, instead of on the edges. Moreover, in this case we assume that no vertex is left uncoloured. That is, in a vertex-coloured game, the colouring function is of the form $\mathsf{col}_{\mathsf{st}}\colon V \to \Gamma$.

The reason why we do not consider vertex-coloured games with uncoloured vertices is because, for the purposes of this chapter, they are equivalent to arbitrary games [Kop08]. Indeed, an edge-coloured game can be converted into a vertex-coloured one by replacing each edge $e = v \xrightarrow{c} v'$ by the path $v \to v_e \to v'$ and letting $\mathsf{col}_{\mathsf{st}}(v) = \mathsf{col}_{\mathsf{st}}(v') = \varepsilon$ and $\mathsf{col}_{\mathsf{st}}(v_e) = c$.

1.2 Different types of memories

We recall that, as defined in the general preliminaries (Section I.1), a memory skeleton over a set X is a deterministic automaton structure over the input alphabet X. Formally, we represent it as a tuple $\mathcal{M} = (M, m_0, \mu)$, where $\mu : M \times X \to M$ is the update function, that we extend to sequences in X^* in the natural way.

In the entire section, Γ stands for a set of colours, W for an objective over Γ, and $\mathcal{G}$ a game with vertices V and edges E.

General memory

If we do not add any restriction to the definition of memory introduced in the general preliminaries, we obtain the notion of *general memory* (also called *chaotic memory* in the literature).

A *general memory structure for a game* $\mathcal{G}$ is a memory skeleton over E together with a function $\mathsf{next\text{-}move}\colon V_{\mathrm{Eve}} \times M \to E$. Such a memory structure implements a strategy $\mathsf{strat}_{\mathcal{M}} : \mathcal{R}un^{\mathsf{fin}}(\mathcal{G}) \rightharpoonup E$ as:

$$\mathsf{strat}_{\mathcal{M}}(\rho) = \mathsf{next\text{-}move}(\mathsf{target}(\rho), \mu(m_0, \rho)), \quad \text{for all } \rho \in \mathcal{R}un^{\mathsf{fin}}(\mathcal{G}) \text{ ending in } V_{\mathrm{Eve}}.$$

We say that Eve can play optimally in $\mathcal{G}$ using general memory k if she has a strategy strat_k implemented by a general memory structure of size at most k that is winning from every vertex in her winning region

We define the *general memory requirements* of an objective $W \subseteq \Gamma^\omega$, which we denote $\mathsf{mem}_{\mathsf{gen}}(W)$, as the least integer k such that Eve can play optimally in W-games using memory k. We remark that an objective W is half-positional if $\mathsf{mem}_{\mathsf{gen}}(W) = 1$.

Similarly, we define the *general memory requirements over ε-free games* (resp. *over vertex-coloured games*) of $W \subseteq \Gamma^\omega$, denoted $\mathsf{mem}^{\varepsilon\text{-}\mathsf{free}}_{\mathsf{gen}}(W)$ (resp. $\mathsf{mem}^{\mathsf{vrt}}_{\mathsf{gen}}(W)$), as the least integer k such that Eve can play optimally in ε-free (resp. vertex-coloured) W-games using memory k.

The general memory requirements of Muller languages (over arbitrary games) have been completely characterised by Dziembowski, Jurdziński and Walukiewicz [DJW97, Theorems 6 and 14] using the round-branching width of the Zielonka tree $\mathcal{Z}_{\mathcal{F}}$.

Proposition IV.1 (General memory requirements of Muller objectives [DJW97])**.**

Let $W = \mathsf{Muller}_\Gamma(\mathcal{F})$ be a Muller language. Then,

$$\mathsf{mem}_{\mathsf{gen}}(W) = \mathsf{rbw}(\mathcal{Z}_{\mathcal{F}}).$$

Chromatic memory

Chromatic memories are those that base their updates uniquely in the colour produced in the game and are oblivious to the particular edge that has been taken.

Let $\mathcal{M} = (M, m_0, \mu)$ be a memory skeleton over Γ. For every W-game $\mathcal{G}$ with edges E, the memory $\mathcal{M}$ induces a memory skeleton over E, by defining the update function $\mu_E\colon M \times E \to M$ as:

$$\mu_E(m, e) = \mu(m, \mathsf{col}(e)), \text{ if } \mathsf{col}(e) \neq \varepsilon, \quad \text{and} \quad \mu_E(m, e) = m, \text{ if } \mathsf{col}(e) = \varepsilon.$$

We say that a memory structure for $\mathcal{G}$ is a *chromatic memory* if its update function can be obtained from a memory skeleton over Γ as above.

The *chromatic memory requirements* of an objective $W \subseteq \Gamma^\omega$, denoted $\mathsf{mem}_{\mathsf{chr}}(W)$, is the least integer k such that Eve can play optimally in W-games using chromatic memory structures of size at most k. We also define the *chromatic memory requirements over ε-free games* (resp. *over vertex-coloured games*) of $W \subseteq \Gamma^\omega$, denoted $\mathsf{mem}^{\varepsilon\text{-}\mathsf{free}}_{\mathsf{chr}}(W)$ (resp. $\mathsf{mem}^{\mathsf{vrt}}_{\mathsf{chr}}(W)$), as the least integer k such that Eve can play optimally in ε-free (resp. vertex-coloured) W-games using chromatic memory k.

Arena-independent memory

We note that in the definition of chromatic memory requirements, we allow the use of a different memory skeleton over Γ for each game. As the update function only depends on Γ, we can imagine that a single structure would suffice for all W-games.

An *arena-independent memory structure for W* is a memory skeleton $\mathcal{M}$ over Γ that can implement an optimal strategy for Eve over every W-game, that is, a strategy that is winning from any vertex in her winning region.

The *arena-independent memory requirements* of an objective W is the minimal size of an arena-independent memory structure for W. We denote it $\mathsf{mem}_{\mathsf{ArInd}}(W)$.

We define analogously *arena-independent memory structures over ε-free games* (resp. *over vertex-coloured games*) and the *arena-independent memory requirements over ε-free games* (resp. vertex-coloured games) of an objective W, which we denote $\mathsf{mem}^{\varepsilon\text{-free}}_{\mathsf{ArInd}}(W)$ (resp. $\mathsf{mem}^{\mathsf{vrt}}_{\mathsf{ArInd}}(W)$).

As we will see in Section IV.2, all ω-regular objectives admit finite arena-independent memories, as a deterministic parity (or Rabin) automaton recognising an objective W is an arena-independent memory for it.

ε-memory *(digression)*

There is yet another type of memory that deserves our attention, although we will not consider it in the following.

We say that a memory structure for a game $\mathcal{G}$ is an *ε-memory* if it satisfies that, for each uncoloured edge $e \in E$ and every memory state $m \in M$, we have that $\mu(m, e) = m$. That is, the memory updates are oblivious to uncoloured edges.

The main reason we will not consider this notion further is because it coincides with general memory in the case of Muller languages [CO22, Section 4.3], and we do not have yet an example that separates both notions in the general case. However, we believe that for some objectives the ε-memory requirements are strictly larger than the general ones.

Although the restriction imposed in the definition of ε-memory might appear artificial at first sight, we argue that in many natural occurrences of ε-edges, we do indeed not want to allow the memory to be updated on them – see Appendix B.1 for such an example: games originating from logical formulas. Moreover, the ε-memory requirements of all objectives have been recently characterised by means of the existence of well-monotone universal graphs [CO23], characterisation that, a priori, does not hold for the general memory requirements.

1.3 Comparison between the different models

We now separate or establish the equality of the notions introduced above. The following inequalities are a direct consequence of the definitions:

$$\begin{array}{ccccccc} \mathsf{mem}^X_{\mathsf{gen}} & \leq & \mathsf{mem}^X_{\mathsf{chr}} & \leq & \mathsf{mem}^X_{\mathsf{ArInd}}\,, & \text{for } X \text{ any class of game graphs.} \\ \mathsf{mem}^{\mathsf{vrt}}_Y & \leq & \mathsf{mem}^{\varepsilon\text{-free}}_Y & \leq & \mathsf{mem}_Y\,, & \text{for } Y \text{ any type of memory.} \end{array}$$

In Table 2, we compile the examples included in this section to separate the different notions of memory requirements. All these examples are based on Muller languages. For

Muller languages, we will show (Theorem IV.1):

$$\mathsf{mem}_{\mathsf{chr}}(W) = \mathsf{mem}^{\varepsilon\text{-free}}_{\mathsf{chr}}(W) = \mathsf{mem}_{\mathsf{ArInd}}(W) = \mathsf{mem}^{\varepsilon\text{-free}}_{\mathsf{ArInd}}(W),$$

so we only include one column for these four notions. Letter n stands for the size of the set of colours Γ.

Memory type: **Game type:**	General Arbitrary	Chromatic Arbitrary	General ε-free	General Vertex-coloured
$\lvert\mathsf{Inf}(w)\rvert = 2$	2	n	2	1
$\lvert\mathsf{Inf}(w)\rvert = n/2$	$n/2$	exponential in n	?	?
$\lvert\mathsf{Inf}(w)\rvert > 1$	n	n	2	1
$\mathsf{Inf}(w) = \{a, b\}$	2	2	2	1

✦ **Table 2.** Summary of the different examples considered in this section and their memory requirements. Letter n stands for the size of the set of colours Γ.

Chromatic and arena-independent requirements coincide

We show next that the chromatic and the arena-independent memory requirements coincide for any objective, that is, when using chromatic memories we can suppose that a single structure suffices to play optimally in all games. A version of this result was first proven by Kopczyński in his PhD thesis [Kop08, Proposition 8.9].

Proposition IV.2 (Arena-independent and chromatic memories [Kop08])**.**

Let $W \subseteq \Gamma^\omega$ be any objective. Then:

- $\mathsf{mem}_{\mathsf{ArInd}}(W) = \mathsf{mem}_{\mathsf{chr}}(W)$,
- $\mathsf{mem}^{\varepsilon\text{-free}}_{\mathsf{ArInd}}(W) = \mathsf{mem}^{\varepsilon\text{-free}}_{\mathsf{chr}}(W)$.

Moreover, if the set of colours Γ is finite, these results also hold for the chromatic memory requirements over finite games.

Proof. The inequalities $\mathsf{mem}_{\mathsf{chr}}(W) \leq \mathsf{mem}_{\mathsf{ArInd}}(W)$ and $\mathsf{mem}^{\varepsilon\text{-free}}_{\mathsf{chr}}(W) \leq \mathsf{mem}^{\varepsilon\text{-free}}_{\mathsf{ArInd}}(W)$ directly follow from the definition. We show that these inequalities are not strict. We present here the proof for arbitrary game graphs; the proof for ε-free games is identical, as we do not add any ε-edges in our construction.

We need to show that there is a W-game $\mathcal{G}$ such that any chromatic memory implementing an optimal strategy for Eve has at least as many states as a minimal arena-independent memory for W. Let $n = \mathsf{mem}_{\mathsf{ArInd}}(W)$ be the size of such minimal arena-independent memory. Let $\mathfrak{M}^\Gamma_{<n}$ be the (possibly infinite) set of all memory skeletons over Γ of size strictly less than n (we note that if Γ is finite, this set is finite). Suppose by contradiction that $\mathsf{mem}_{\mathsf{chr}}(W) < n$. By definition of $\mathsf{mem}_{\mathsf{ArInd}}(W)$, for any memory skeleton $\mathcal{M} \in \mathfrak{M}^\Gamma_{<n}$ there is some W-game $\mathcal{G}_\mathcal{M}$ such that the structure $\mathcal{M}$ cannot implement an optimal strategy for Eve in $\mathcal{G}_\mathcal{M}$. Consider the disjoint union of all the games $\mathcal{G}_\mathcal{M}$, for

$\mathcal{M} \in \mathfrak{M}^{\Gamma}_{<n}$. As $\mathsf{mem}_{\mathsf{chr}}(W) < n$, there is some memory skeleton $\mathcal{M} \in \mathfrak{M}^{\Gamma}_{<n}$ implementing an optimal strategy in this disjoint union. However, the skeleton $\mathcal{M}$ implements an optimal strategy in $\mathcal{G}_{\mathcal{M}}$, a contradiction. ◀

General vs chromatic memory

We now show that, in general, chromatic memories are strictly less powerful than general ones: $\mathsf{mem}_{\mathsf{gen}}(W) < \mathsf{mem}_{\mathsf{chr}}(W)$. This answers a question raised by Kopczyński in his PhD thesis [Kop08, Section 8.4]. We moreover give an example proving that, already for Muller languages, the chromatic memory requirements can be exponentially larger (in the size of the alphabet) than the general ones (Proposition IV.5).

Proposition IV.3 (General vs chromatic memory: Arbitrary gap)**.**

For each integer $n \geq 2$, there exists a set of colours Γ_n of size n and a Muller language $W_n \subseteq \Gamma_n^\omega$ such that:

- $\mathsf{mem}_{\mathsf{gen}}(W_n) = 2$,
- $\mathsf{mem}_{\mathsf{chr}}(W_n) = n$.

Example IV.4 ($|\mathsf{Inf}(w)| = 2$)**.**

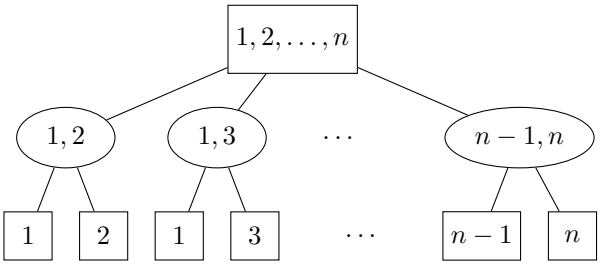
Let $\Gamma_n = \{1, \dots, n\}$ and let

$$W_n = \{w \in \Gamma_n^\omega \mid |\mathsf{Inf}(w)| = 2\}.$$

This is the Muller language associated to the family $\mathcal{F}_n = \{X \subseteq \Gamma_n \mid |X| = 2\}$.

To determine the general memory requirements of W_n we use the characterisation from Proposition IV.1, using the round-branching width of the Zielonka tree of $\mathcal{F}_n$. This Zielonka tree is pictured in Figure 24, from which we directly obtain that $\mathsf{rbw}(\mathcal{Z}_{F_n}) = \mathsf{mem}_{\mathsf{gen}}(W_n) = 2$.

For the analysis of the chromatic memory requirements of W_n we use Theorem IV.1, stating that $\mathsf{mem}_{\mathsf{chr}}(W_n)$ equals the size of a minimal deterministic Rabin automaton recognising W_n. The language W_n coincides with the language $\mathsf{Muller}(\mathcal{F}_G)$ – introduced in Section III.1 to study the minimisation of Rabin automata – for G a clique of size n. Lemma III.7 implies that a minimal deterministic Rabin automaton for this language has size n, and therefore $\mathsf{mem}_{\mathsf{chr}}(W_n) = n$.



✦ **Figure 24.** Zielonka tree of the family $\mathcal{F}_n = \{X \subseteq \{1, 2, \dots, n\} : |X| = 2\}$. The round-branching width of this tree is 2.

Proposition IV.5 (General vs chromatic memory: Exponential gap)**.**

There exists a family of Muller languages $W_n \subseteq \Gamma_n^\omega$ over alphabets of n colours such that, for a constant $\alpha > 1$:

- $\mathsf{mem}_{\mathsf{gen}}(W_n) = n/2$,
- $\mathsf{mem}_{\mathsf{chr}}(W_n) \geq \alpha^n$.

Example IV.6 ($|\mathsf{Inf}(w)| = n/2$)**.**

Let $\Gamma_n = \{1, \dots, n\}$, for n even, and let

$$W_n = \{w \in \Gamma_n^\omega \mid |\mathsf{Inf}(w)| = n/2\}.$$

That is, a word belongs to W_n if the set of colours seen infinitely often is exactly $n/2$. This is the Muller language associated to the family $\mathcal{F}_n = \{X \subseteq \Gamma_n \mid |X| = n/2\}$.

To determine the general and chromatic memory requirements of this objective, we apply Theorems IV.1 and IV.3 characterising $\mathsf{mem}_{\mathsf{gen}}(W_n)$ and $\mathsf{mem}_{\mathsf{chr}}(W_n)$ in terms of the minimal deterministic and good-for-games Rabin automata recognising W_n, respectively. Then, it suffices to note that this family of languages is the one used in the proof of Theorem III.5, stating the same exponential gap between the two models of automata.

Memory over ε-free games

We prove that the memory requirements over arbitrary and ε-free games differ. In particular, the characterisation of the memory requirements of Muller languages of Dziembowski, Jurdziński and Walukiewicz [DJW97] only holds for arbitrary games.

Proposition IV.7 (Memory over ε-free games)**.**

For each integer $n \geq 2$, there exists a set of colours Γ_n of size n and a Muller language $W_n \subseteq \Gamma_n^\omega$ such that:

- $\mathsf{mem}_{\mathsf{gen}}^{\varepsilon\text{-free}}(W_n) = 2$,
- $\mathsf{mem}_{\mathsf{gen}}(W_n) = n$.

Example IV.8 ($|\mathsf{Inf}(w)| > 1$)**.**

Let $\Gamma_n = \{1, \dots, n\}$ and let

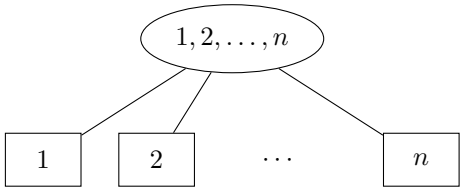
$$W_n = \{w \in \Gamma_n^\omega \mid |\mathsf{Inf}(w)| > 1\}.$$

That is, a word belongs to W_n if it does not eventually end in a^ω, for $a \in \Gamma_n$. This is the Muller language associated to the family $\mathcal{F}_n = \{X \subseteq \Gamma_n \mid |X| > 1\}$. Its Zielonka tree is depicted in Figure 25, from which we deduce that its round-branching width is n, so, by Proposition IV.1, $\mathsf{mem}_{\mathsf{gen}}(W_n) = n$.

However, we show that over ε-free games, Eve only requires 2 memory states to play optimally. Let $\mathcal{G}$ be a an ε-free W_n-game. First, by prefix-independence of W_n, we can suppose that Eve wins $\mathcal{G}$ from every vertex: it suffices to restrict ourselves to her winning

region. As the game is ε-free, from any vertex $v \in V_{\text{Eve}}$ there is one outgoing edge coloured with some colour in Γ_n. We associate to each vertex $v \in V_{\text{Eve}}$ one such colour, via a mapping $c : V_{\text{Eve}} \to \Gamma_n$, obtaining a partition $V_{\text{Eve}} = V_1 \sqcup \ldots \sqcup V_n$ satisfying that $v \in V_x$ implies that there is some edge coloured x leaving v. We denote $\mathsf{strat}_0 : V_{\text{Eve}} \to E$ one application that maps each vertex $v \in V_{\text{Eve}}$ to one outgoing edge labelled with $c(v)$. Moreover, for any $v \in V$, since Eve can win from v, there is some strategy in $\mathcal{G}$ forcing to see some colour $y \neq c(v)$. For each colour $x \in \Gamma_n$, we let $\mathsf{strat}_{\overline{x}} : V_{\text{Eve}} \to E$ be a positional strategy forcing to see a colour different from x (this positional strategy exists since reachability games are half-positional). Moreover, we can pick $\mathsf{strat}_{\overline{x}}$ such that it coincides with strat_0 outside V_x.

We define a memory structure $\mathcal{M} = (\{m_0, m_1\}, m_0, \mu)$ for $\mathcal{G}$ and a next-move function describing the following strategy: the state m_0 will be used to remember that we have to see the colour x corresponding to the component V_x that we are in. That is, when we are in m_0, we follow the strategy strat_0. As soon as we arrive to a vertex controlled by Eve, we use the next transition to accomplish this and we can change to state m_1 in $\mathcal{M}$. The state m_1 will serve to follow the positional strategy $\mathsf{strat}_{\overline{x}}$ reaching one colour different from x. We will change to state m_0 if we arrive to some state in V_{Eve} not in V_x (this will ensure that we will see one colour different from x), or if some colour different from x is produced.



✦ **Figure 25.** Zielonka tree of the family $\mathcal{F}_n = \{X \subseteq \{1, 2, \ldots, n\} \ : \ |X| > 1\}$. The round-branching width of this tree is n.

However, in the case of positionality, there is no difference for Muller languages when considering arbitrary or ε-free games: for both models, positional objectives coincide with Rabin objectives. A first proof of half-positionality for Rabin objectives was obtained by Klarlund [Kla94, Lemma 9]. A simpler proof appears implicitly in the work of Emerson [Eme85], and the fact that these are the only half-positional Muller objectives was proven by Zielonka [Zie98, Corollary 14].

Proposition IV.9 (Half-positionality of Rabin languages [Kla94, Zie98])**.**

Let W be a Muller language. The following conditions are equivalent:

- W is a Rabin language.
- W is half-positional over arbitrary games.
- W is half-positional over ε-free games.

In Chapter V (Theorem V.5) we further show that more generally, ε-edges in games do not affect the half-positionality of ω-regular languages. However, we do not know if

this is the case for any language. This question was already raised by Kopczyński [Kop08, Section 2.5].

Conjecture IV.1 (Arbitrary vs ε-free games for half-positionality [Kop08])**.**

An objective W is half-positional over arbitrary games if and only if it is half-positional over ε-free games.

Memory for vertex-coloured games

The difference between the memory requirements over vertex-coloured games without uncoloured vertices and arbitrary games was first studied by Zielonka. Using the Zielonka tree, he characterised Muller languages that are half-positional over totally coloured vertex-coloured games [Zie98, Theorem 17]. The exact memory requirements over vertex-coloured games have not been characterised in general.

Example IV.10 ($\mathsf{Inf}(w) = \{a, b\}$)**.**

Let $\Gamma = \{a, b\}$ and let $W = \mathsf{Muller}(\mathcal{F})$ be the Muller language associated to the family $\mathcal{F} = \{\{a, b\}\}$. That is, Eve wins a W-game if she produces both a and b infinitely often. It follows from Proposition IV.1 that $\mathsf{mem}_{\mathsf{gen}}(W) = 2$. However, Eve can play optimally using positional strategies in vertex-coloured games. Indeed, over a-vertices she can use a positional strategy to reach colour b, and over b-vertices a positional strategy to reach colour a.

There are many other examples that display this property. For example, the objective $W = \{w \in \Gamma^\omega \mid$ the two first letters are different$\}$ is positional only over vertex-coloured games. It is also the case for the parity objective over infinitely many priorities, as shown by Grädel and Walukiewicz [GW06].

2 Chromatic memory and deterministic Rabin automata

The main contribution of this section is to establish an exact correspondence between chromatic memory structures for a Muller language $W \subseteq \Gamma^\omega$ and deterministic Rabin automaton for W (Theorem IV.1 below). As a corollary, we obtain that it is NP-complete to determine the chromatic memory requirements of a Muller language (Theorem IV.2).

Theorem IV.1 (Chromatic memory requirements of Muller languages)**.**

Let $W \subseteq \Gamma^\omega$ be a Muller language. The following quantities coincide:

1. The number of states of a minimal deterministic Rabin automaton recognising W.
2. The arena-independent memory requirements of W, $\mathsf{mem}_{\mathsf{ArInd}}(W)$.
3. The arena-independent memory requirements over ε-free games of W, $\mathsf{mem}^{\varepsilon\text{-free}}_{\mathsf{ArInd}}(W)$.
4. The chromatic memory requirements of W, $\mathsf{mem}_{\mathsf{chr}}(W)$.
5. The chromatic memory requirements over ε-free games of W, $\mathsf{mem}^{\varepsilon\text{-free}}_{\mathsf{chr}}(W)$.

Moreover, a deterministic automaton structure over Γ is an arena-independent memory for W if and only if we can define a Rabin acceptance condition on top of it making it a

Rabin automaton recognising W.

Proof. The relations $\mathsf{mem}^{\varepsilon\text{-free}}_{\mathsf{ArInd}}(W) = \mathsf{mem}^{\varepsilon\text{-free}}_{\mathsf{chr}}(W) \leq \mathsf{mem}_{\mathsf{ArInd}}(W) = \mathsf{mem}_{\mathsf{chr}}(W)$ are given by Proposition IV.2. The final claim, as well as the missing inequalities, are given by Lemmas IV.11 and IV.12. ◀

2.1 From deterministic Rabin automata to chromatic memory

We give the first direction of Theorem IV.1: a deterministic Rabin automaton $\mathcal{A}$ is an arena-independent memory structure for $\mathcal{A}$. This result can be considered as already known, as it simply follows from the half-positionality of Rabin languages [Kla94] and the product construction of an automaton and a game [GH82].

Lemma IV.11 (Deterministic Rabin automata are arena-independent memories)**.**

Let $\mathcal{A}$ be a deterministic Rabin automaton recognising a language $W \subseteq \Gamma^\omega$. Then, $\mathcal{A}$ is an arena-independent memory structure for W.

Proof. The automaton $\mathcal{A}$ naturally gives a memory skeleton over Γ: $\mathcal{M}_\mathcal{A} = (Q, q_0, \mu)$, where Q is the set of states of $\mathcal{A}$, q_{init} its initial state, and $\mu(q, a) = q'$ if (q, a, q') is a transition in the automaton. We need to show that in every W-game the structure $\mathcal{M}_\mathcal{A}$ can implement an optimal strategy for Eve.

Let $\mathcal{G}$ be a game with $W = \mathcal{L}(\mathcal{A})$ as winning condition and having V and E as set of vertices and edges, respectively. We consider the composition $\mathcal{G} \ltimes \mathcal{A}$ of $\mathcal{G}$ with the automaton $\mathcal{A}$ as defined in Section II.1.3, that is, $\mathcal{G} \ltimes \mathcal{A}$ contains an edge $(v, q) \xrightarrow{c} (v', q')$ if there is an edge $v \xrightarrow{a} v'$ in $\mathcal{G}$, and the a-transition from q in $\mathcal{A}$ is $q \xrightarrow{a:c} q'$. By Proposition II.6, if Eve wins $\mathcal{G}$ from a vertex v, she wins $\mathcal{G} \ltimes \mathcal{A}$ from (v, q_{init}). Moreover, the game $\mathcal{G} \ltimes \mathcal{A}$ uses as winning condition the Rabin language used by $\mathcal{A}$ as acceptance set. Therefore, by Proposition IV.9, she can play optimally using a positional strategy $\mathsf{strat}_{\mathsf{pos}}$ given by a function $\mathsf{next\text{-}move} \colon V_{\mathrm{Eve}} \times Q \to E$. This is exactly a next-move function for the memory skeleton $\mathcal{M}_\mathcal{A}$, implementing a strategy $\mathsf{strat}_\mathcal{G}$ in $\mathcal{G}$. Any play consistent with $\mathsf{strat}_\mathcal{G}$ in $\mathcal{G}$ is the projection of a play consistent with $\mathsf{strat}_{\mathsf{pos}}$ in $\mathcal{G} \ltimes \mathcal{A}$. Therefore, as $\mathsf{strat}_{\mathsf{pos}}$ is optimal, so is $\mathsf{strat}_\mathcal{G}$. ◀

2.2 From chromatic memory to deterministic Rabin automata

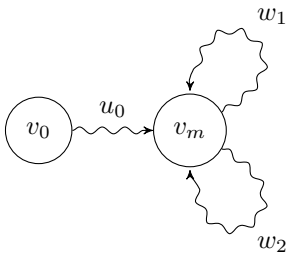
We provide the missing direction of Theorem IV.1: an arena-independent structure for a Muller language W can be equipped with the structure of a deterministic Rabin automaton recognising W. This result is easily obtained as a corollary of the typeness results offered by the alternating cycle decomposition (Proposition II.109).

Lemma IV.12 (Chromatic memories are Rabin automata)**.**

Let $W \subseteq \Gamma^\omega$ be a Muller language, and let $\mathcal{M}$ be an arena-independent memory structure for W over ε-free games. Then, we can define a Rabin acceptance condition on top of $\mathcal{M}$, obtaining a deterministic Rabin automaton recognising W.

Proof. Let $\mathcal{M} = (M, m_0, \mu)$ be an arena-independent memory for W over ε-free games. First, we remark that we can suppose that every state of $\mathcal{M}$ is accessible from m_0 by some sequence of transitions. We define a deterministic Muller automaton $\mathcal{A}_\mathcal{M}$ on top of $\mathcal{M}$ just by using the input letters as output: for each transition $e = m \xrightarrow{a} m'$ we define $\mathsf{col}(e) = a \in \Gamma$, and we let W be the acceptance set of the automaton. Since the run over a word $w \in \Gamma^\omega$ produces as output w itself and the accepting set is W, this automaton trivially accepts the language W. We show that we can also define an equivalent Rabin condition on top of $\mathcal{M}$. For this, we verify that the automaton $\mathcal{A}_\mathcal{M}$ satisfies the second property from Proposition II.109, that is, that for any pair of cycles ℓ_1, ℓ_2 in $\mathcal{A}_\mathcal{M}$ with some state in common, if both cycles are rejecting, then their union is also rejecting. This will prove that we can put a Rabin condition on top of $\mathcal{A}_\mathcal{M}$.

Let ℓ_1 and ℓ_2 be two rejecting cycles in $\mathcal{A}_\mathcal{M}$ such that $m \in M$ appears in both ℓ_1 and ℓ_2. We suppose by contradiction that their union $\ell_1 \cup \ell_2$ is an accepting cycle. We build an ε-free W-game in which Eve can win, but not using the memory skeleton $\mathcal{M}$, leading to a contradiction. Let $u_0 \in \Gamma^*$ be a word labelling a path from m_0 to m in $\mathcal{M}$, that is, $\mu(m_0, u_0) = m$. Let w_1 and w_2 in Γ^+ be two finite words labelling the cycles ℓ_1 and ℓ_2 from m; that is, the run over w_i from m visits all the edges in ℓ_i and goes back to m, without visiting any edge not appearing in ℓ_i. We note that by hypothesis $w_1^\omega \notin W$ and $w_2^\omega \notin W$. Consider the game $\mathcal{G}$ from Figure 26 (if $u_0 = \varepsilon$, we remove vertex v_0 to obtain an ε-free game). Eve can win the game $\mathcal{G}$ from v_0: as $\ell_1 \cup \ell_2$ is accepting, she only



✦ **Figure 26.** Game in which Eve cannot play optimally using the memory skeleton $\mathcal{M}$.

has to alternate between the two choices in the state v_m. However, there is no function $\mathsf{next\text{-}move} : M \times V_{\text{Eve}} \to E$ implementing a winning strategy. Indeed, for every partial play ending in v_m and labelled with $u_0 w_{i_1} \dots w_{i_k}$, with $w_{i_j} \in \{w_1, w_2\}$, it is clear that $\mu(u_0 w_{i_1} \dots w_{i_k}) = m$ (the memory is at state m). If $\mathsf{next\text{-}move}(m, v_m)$ is the edge leading to the cycle labelled w_i, all plays will stay in that cycle, which is losing for Eve. We conclude that $\mathcal{M}$ cannot be used as a memory structure for $\mathcal{G}$, a contradiction. ◀

2.3 The complexity of determining the chromatic memory

We consider next the problem of determining the chromatic memory requirements of a Muller language:

Problem:	CHROMATIC-MEMORY
Input:	A Muller language $W = \mathsf{Muller}(\mathcal{F})$ and a positive integer k.
Question:	$\mathsf{mem}_{\mathsf{chr}}(W) \leq k$?

This problem admits several variants, depending on the representation of the input Muller language. We show next that, even if the input is given as a Zielonka tree, this problem is NP-complete. This highly contrast with the fact that the Zielonka tree completely characterises the general memory requirements of a Muller language (Proposition IV.1).

Theorem IV.2 (Determining the chromatic memory is NP-complete)**.**

The problem Chromatic-Memory is NP-complete for a representation of the input Muller language as either:

- The Zielonka tree $\mathcal{Z}_{\mathcal{F}}$.
- A deterministic parity automaton recognising W.
- A deterministic Rabin automaton recognising W.

Proof. By Theorem IV.1, $\mathsf{mem}_{\mathsf{chr}}(W)$ coincides with the size of a minimal deterministic Rabin automaton recognising W. The result follows then from the NP-completeness of the problem of the minimisation of deterministic Rabin automata recognising Muller languages: for the representation as a Rabin automaton or as the Zielonka tree, this is a consequence of Theorem III.1; for the representation as a parity automaton, it suffices to remark that we can compute the Zielonka tree in polynomial time (Proposition III.18). ◀

3 General memory and good-for-games Rabin automata

In this section, we present the second main contribution of the chapter: the general memory requirements of a Muller language exactly correspond to minimal good-for-games Rabin automata recognising it. The memory requirements of a Muller language were already known, as they were characterised in [DJW97] using the Zielonka tree (Proposition IV.1); our main contribution is to provide a construction of a GFG Rabin automaton matching this lower bound. This construction was presented in Section II.3.3 (Theorem II.4).

Theorem IV.3 (General memory requirements of Muller languages as automata)**.**

Let $W \subseteq \Gamma^\omega$ be a Muller language. The following quantities coincide:

1. The number of states of a minimal good-for-games Rabin automaton recognising W.
2. The general memory requirements of W, $\mathsf{mem}_{\mathsf{gen}}(W)$.

3.1 From good-for-games Rabin automata to general memory

We show that if $\mathcal{A}$ is a good-for-games Rabin automaton, then it can be used as a memory structure for any $\mathcal{L}(\mathcal{A})$-game. This result almost follows from the definition of good-for-games automata, and can be considered as folklore.

Lemma IV.13 (GFG Rabin automata are memory structures)**.**

Let $\mathcal{A}$ be a good-for-games Rabin automaton recognising a language $W \subseteq \Gamma^\omega$. Then, Eve can play optimally in any W-game using at most $|\mathcal{A}|$ states of general memory.

Proof. Let $\mathcal{G}$ be a game with $W = \mathcal{L}(\mathcal{A})$ as winning condition, having V and E as sets of vertices and edges, respectively. Letter Q stands for the set of states of $\mathcal{A}$. We want to take the composition of $\mathcal{G}$ and $\mathcal{A}$ to obtain a Rabin game in which Eve can play optimally using a positional strategy (by Proposition IV.9), and apply a similar argument as the one used in the proof of Lemma IV.11. In order to be able to perform this composition operation – as formally presented in Section II.1.3 – we first need to ensure that $\mathcal{G}$ is a game suitable for transformations. Let $\tilde{\mathcal{G}}$ be the game obtained from $\mathcal{G}$ in the following way: for every edge $e = v \xrightarrow{a} v'$ in $\mathcal{G}$, we add a position (v, e) controlled by Eve and replace edge e by $v \xrightarrow{\varepsilon} (v, e) \xrightarrow{a} v'$. It is clear that Eve wins $\mathcal{G}$ from a vertex v if and only if she wins $\tilde{\mathcal{G}}$ from that same vertex and that $\tilde{\mathcal{G}}$ is suitable for transformations. By Proposition II.6, there is an initial state q_{init} in $\mathcal{A}$ such that if Eve wins $\mathcal{G}$ from a vertex v, she wins $\tilde{\mathcal{G}} \ltimes \mathcal{A}$ from (v, q_{init}). Moreover, the game $\tilde{\mathcal{G}} \ltimes \mathcal{A}$ uses the acceptance set from $\mathcal{A}$, which is a Rabin language, so, by Proposition IV.9, Eve can win using a positional strategy given by a function $\mathsf{strat} \colon \tilde{V}_{\mathrm{Eve}} \to \tilde{E}$, where $\tilde{V}_{\mathrm{Eve}}$ is the set of vertices controlled by Eve in $\tilde{\mathcal{G}}$ (this is a subset of $(V_{\mathrm{Eve}} \sqcup (V \times E)) \times Q$). We build a memory structure $(\mathcal{M}, \mathsf{next\text{-}move})$ of size $|Q|$ that projects the strategy strat onto $\mathcal{G}$:

- its set of states is $M = Q$,
- the initial state is q_{init},
- the update function $\mu \colon M \times E \to M$ is given by:

$$\mu(q, e) = q' \quad \text{if} \quad \mathsf{strat}((v, e), q) = ((v, e), q) \to (v', q').$$

  That is, if the strategy strat takes the transition $q \to q'$ in the "automaton component" of $\tilde{\mathcal{G}} \ltimes \mathcal{A}$ after the move $e = v \to v'$ is taken in the "game component".
- For $v \in V_{\mathrm{Eve}}$ and $q \in M$, we let $\mathsf{next\text{-}move}(v, q) = e$ if e is the move chosen by strat from (v, q), that is, if $\mathsf{strat}(v, q) = (v, q) \to ((v, q), e)$.

Since strat is a winning strategy in $\tilde{\mathcal{G}} \ltimes \mathcal{A}$ from (v, q_{init}), its projection onto $\mathcal{G}$ via the memory structure $(\mathcal{M}, \mathsf{next\text{-}move}_{\mathcal{M}})$ is a strategy that satisfies that any play consistent with it produces as output a word in $\mathcal{L}(\mathcal{A})$, so it is winning. ◀

3.2 From general memory to good-for-games Rabin automata

Lemma IV.14 (A GFG Rabin automaton of size $\mathsf{mem}_{\mathsf{gen}} W$)**.**

Let $W \subseteq \Gamma^\omega$ be a Muller language. There exists a good-for-games Rabin automaton recognising W of size $\mathsf{mem}_{\mathsf{gen}}(W)$.

Proof. Let $\mathcal{F} \subseteq 2^\Gamma_+$ be the family of subsets such that $W = \mathsf{Muller}(\mathcal{F})$. By the characterisation from [DJW97] (Proposition IV.1), the general memory requirements of W are given by the round-branching width of the Zielonka tree of $\mathcal{F}$: $\mathsf{mem}_{\mathsf{gen}}(W) = \mathsf{rbw}(\mathcal{Z}_{\mathcal{F}})$. In Section II.3.3 (Theorem II.4) we gave an explicit construction of a history-deterministic

Rabin automaton of size $\mathsf{rbw}(\mathcal{Z}_{\mathcal{F}})$ recognising W. By Proposition II.5, that automaton is good-for-games. ◀

Positionality of ω-regular languages and beyond

V

Outline

No memories. No problems.

Hank (Finding Dory)

Introduction

Positional strategies. As mentioned in the general introduction, a crucial parameter in the study of games on graphs is the complexity of strategies required by the players to play optimally. The simplest such strategies are positional ones, those that depend only on the current vertex, and not on the history of the play. In this chapter, we are interested in the following question: Given a fixed objective W, is it the case that players can play optimally using positional strategies in all games that have W as winning objective? We can ask this question just for one player (player Eve) – we say in the affirmative case that W is half-positional – or for both players – we say that W is bipositional. Also, it might be relevant to consider the question for subclasses of games, in particular, for finite games, or for 1-player games.

Bipositionality. The class of bipositional objectives, both over finite and infinite games, is already well understood. A characterisation of bipositionality over finite games was obtained by Gimbert and Zielonka [GZ05], using two properties called *monotonicity* and *selectivity*. An important and useful corollary of their result is what is commonly known as a 1-to-2-player lift: an objective W is bipositional over finite games if and only if both players can play optimally using positional strategies in finite 1-player games.

Over infinite games, a very simple and elegant characterisation of bipositionality was given by Colcombet and Niwiński for prefix-independent objectives [CN06]: a prefix-independent objective W is bipositional if and only if it is a parity language. In particular, these objectives are necessarily ω-regular. No such characterisation for non-prefix-independent objectives appears in the literature (although a generalisation of this result for finite memory without the prefix-independent assumption appears in [BRV23]).

Half-positionality. Although half-positionality is arguably more relevant than bipositionality in the context of reactive synthesis (the controller will be produced based on Eve's strategies), much less is known for this class. During the 90s, half-positionality of some central objectives was proven, notably of parity [EJ91] and Rabin languages [Kla94]. The first thorough study of half-positionality was conducted by Kopczyński in his PhD thesis [Kop08]. There, he provides some sufficient conditions for half-positionality and introduces an important set of conjectures that have greatly influenced research in the area in recent years. However, no general characterisation for half-positionality was found. One of the main contributions of Kopczyński was to show decidability of half-positionality over finite games for prefix-independent ω-regular objectives [Kop07, Theorem 2]. However, his procedure works in $\mathcal{O}(n^{\mathcal{O}(n^2)})$ time (where n is the size of a deterministic parity automaton given as input), and more importantly, does not really reveal much about the structure of automata recognising positional languages. Decidability of half-positionality over arbitrary games, or for non-prefix-independent ω-regular objectives is open.

Very recently, a significant breakthrough occurred as a result of the works in Ohlmann's PhD thesis [Ohl21, Ohl23a]. He provided a characterisation of half-positionality by means of the existence of graph-theoretical structures known as monotone universal graphs. While this characterisation is a valuable tool for proving half-positionality of many objectives, it does not directly offer methods to establish, for instance, decidability of half-positionality for ω-regular objectives. Also, Ohlmann's characterisation comes with a caveat: necessity of the existence of universal graphs for half-positionality is only obtained for objectives containing a neutral letter (a letter that does not change membership to W

after its removal). He conjectures that this hypothesis is not essential, as the addition of a neutral letter to any objective should not break half-positionality (see Conjecture V.3).

This motives the following research direction.

Objective.

Characterise the class of half-positional ω-regular languages, both over finite and infinite games.

Finite-to-infinite and 1-to-2-player lifts. As mentioned above, a consequence of Gimbert and Zielonka's result [GZ05] is that, in order to check bipositionality over finite games, it suffices to check whether players can play optimally in 1-player games. Recently, generalisations of 1-to-2-player lifts have been studied in the setting of finite memory by Kozachinskiy [Koz22c] and Vandenhove [Van23, Bou+20, BRV23]. Vandenhove states the following conjecture:

Conjecture V.1 (Version of [Van23, Conjecture 9.1.1])**.**

Let $W \subseteq \Sigma^\omega$ be an ω-regular objective. If the memory requirement of W over Eve-games (resp. over finite games) is k, then the memory requirement of W over arbitrary games is k.

He remarks that the corresponding question for $k = 1$ (lift for half-positionality) is also open.

Closure properties. The main motivation of the research done in Kopczyński's PhD thesis [Kop08] is the question whether prefix-independent half-positional objectives are closed under union. We refer to this question as Kopczyński's conjecture. Very recently, Kozachinskiy [Koz22a] disproved this conjecture, but only for half-positionality over *finite games*. Also, the counter-example he gives is not ω-regular. On the positive side, this conjecture has been proven to hold for the family of Σ^0_2 objectives (objectives recognised by infinite coBüchi automata) [Ohl23a]. Kopczyński's conjecture and this latter result have been generalised to the setting of finite memory [CO22, Section 6.3]. Solving Kopczyński's conjecture over infinite games is one of the driving open questions for the field.

One of the very few closure properties known for half-positional objectives is closure under lexicographic products, obtained as a corollary of Ohlmann's characterisation using universal graphs [Ohl23a].

Contributions

The main contribution of this chapter is a complete characterisation of half-positionality for ω-regular languages, stated in Theorem V.1. We give a syntactic description of a family of deterministic parity automata, which we call fully progress consistent signature automata, so that any such automaton recognises a half-positional language, and any half-positional language can be recognised by an automaton of this family. From this characterisation, we derive multiple corollaries that address the majority of open questions related to half-positionality in the case of ω-regular languages:

1. **Decidability in polynomial time.** Given a deterministic parity automaton $\mathcal{A}$, we can decide in polynomial time whether $\mathcal{L}(\mathcal{A})$ is half-positional or not (Theorem V.2).
2. **Finite-to-infinite and 1-to-2-players lift.** An ω-regular objective W is half-positional over arbitrary games if and only if it is half-positional over finite, ε-free Eve-games (Theorem V.3). This answers a question raised by Vandenhove [Van23, Conjecture 9.1.1].
3. **Closure under union.** The union of two ω-regular half-positional objectives is half-positional, provided that one of them is prefix-independent (Theorem V.4). This solves a stronger variant of Kopczyński's conjecture in the case of ω-regular languages.
4. **Closure under addition of a neutral letter.** If W is ω-regular and half-positional, the objective obtained by adding a neutral letter to W is half-positional too (Theorem V.5). This solves Ohlmann's conjecture in the case of ω-regular languages.

We obtain some further results pertaining to classes of objectives that are not necessarily ω-regular. We relax the ω-regularity hypothesis in two orthogonal ways.

5. **Characterisation of bipositionality of all objectives.** We extend the characterisation of bipositionality of Colcombet and Niwiński [CN06] to all objectives, getting rid of the prefix-independence assumption (Theorem V.6).
6. **Characterisation of half-positionality of closed and open objectives.** We characterise half-positionality for closed and open objectives (Theorems V.7 and V.8). We also obtain as corollaries 1-to-2 players lifts and closure under addition of a neutral letter for these classes of objectives.

Technical tools

We would like to highlight some technical tools that take primary importance in our proofs.

Universal graphs. In general, showing that a given objective is half-positional can be challenging, as we need to show that *for every game* Eve can play optimally using positional strategies. Ohlmann's characterisation using monotone universal graphs provides a painless path to prove half-positionality (see Proposition V.14). We strongly rely on this result to show that parity automata satisfying the syntactic conditions imposed in Theorem V.1 do indeed recognise half-positional languages.

History-deterministic automata. Although the statements of our results do not mention in any way history-deterministic automata, we use them in a very fundamental manner in two different parts of our proofs:

- In order to obtain the necessity of the syntactic conditions from our main characterisation (see Section V.4.2), we need to have a very fine control of the structure of automata. To do so, we put automata in some sort of canonical form, for which we need to use and generalise the methods introduced by Abu Radi and Kupferman [AK22] for the minimisation of HD coBüchi automata. This allows us to greatly simplify the structure of automata, at the cost of introducing history-determinism.
- To show the other implication of Theorem V.1, we need to build a monotone

universal graph from a signature automaton. To facilitate this process, we first "saturate" automata, adding as many transitions as possible without modifying the languages they recognise. This procedure generates non-determinism, but preserves history-determinism, the key property that allows us to prove universality of the obtained graph (see Lemma V.93).

We believe that this use of history-determinism underscores the usefulness and canonicity of the model. Our results are not aimed at understanding the succinctness of expressiveness properties of history-deterministic automata; rather, they serve as a tool. This approach reveals that history-determinism offers a natural framework, enabling us to carry out various proofs that would otherwise be unattainable using exclusively deterministic automata.

Normal form of parity automata. In our central proof (Section V.4.2), we rely on the normal form of parity automata introduced in Section II.7.1, in particular, in the properties stated in Proposition II.134 and Theorem II.14. We make consistent use of these properties in our combinatorial arguments.

Congruences for parity automata. Since the beginning of the theory of finite automata, the notion of congruence has played a fundamental role [Arn85, Saë90, MS97]. Here, we propose a notion of congruences for parity automata that make it possible to build quotient automata that are compatible with the acceptance condition (see Definition V.55). This newly introduced vocabulary allows us to formalise the details of the proof of Theorem V.1 in a simpler way.

Organisation of the chapter

The current chapter is, by far, the most technical part of the thesis. The definition of some of the central concepts appearing in the statement of the characterisation of half-positionality (Theorem V.1) is already quite involved. Therefore, the organisation of the chapter has been done with the utmost focus on clarity, aiming to help the reader understand the contributions, as well as the newly introduced definitions and techniques. This might have resulted in some not completely conventional choices.

After introducing some general definitions and terminology used throughout the chapter, we begin Section V.2 by stating the characterisation result (Theorem V.1) and its main consequences, without providing formal details about the technical concepts appearing in the statement of the Theorem. Section V.3 is a warm-up for the definitions used in the main characterisation and for the techniques used in its proof. We gradually introduce conditions that are necessary for half-positionality, obtaining partial results and providing numerous examples along the way. Section V.4 contains the most technical part of the chapter. We introduce the notions of signature automata, full progress consistency and ε-complete automata appearing in the statement of Theorem V.1, and we give a proof of it. Nevertheless, most details in the proof of necessity are relegated to Appendix C. Sections V.5 and V.6 contain, respectively, the two other contributions of the chapter: a characterisation of bipositionality for all objectives and a characterisation of half-positionality for open and closed objectives. The proofs of these latter sections are much simpler.

Prerequisites and links with other chapters

In addition to the concepts presented in the general preliminaries about parity automata, history-determinism and the parity index, we incorporate a preliminary section

that introduces various technical notions employed in our characterisation. We extensively use the normal form of parity automata introduced in Section II.7; we recall in Lemma V.13 the main properties of this form used during the chapter.

In the conclusions (Section VII.2) we comment on the relation between the results and techniques used in this chapter and the problem of the minimisation of parity automata, discussed in Chapter III.

Collaborators and related publications

The whole chapter is based on joint work with Pierre Ohlmann. We want to thank Hugo Gimbert, Damian Niwiński and Pierre Vandenhove for interesting discussions on the subject. The characterisation of half-positional objectives recognised by deterministic Büchi automata, presented in Section V.3.3, is based on joint work with Patricia Bouyer, Mickael Randour and Pierre Vandenhove. This latter characterisation was published in [Bou+22]. The rest of the contributions of the chapter have not yet been published.

1 Preliminaries

We introduce some definitions and terminology specific to this chapter.

Global hypothesis in Chapter V.

We assume in the whole chapter that automata are complete.
To simplify notations, we assume that all vertices in games are initial.

Congruences and monotone preorders over automata

Equivalence relations and preorders. We will use $\sim_X$ to denote different equivalence relations, and $[q]_X$ to denote the *equivalence class* of an element q (which is usually a state in an automaton).

A *preorder* $\leq_X$ is a binary relation that is reflexive and transitive. We say that it is *total* if every pair of elements are comparable. The *equivalence relation induced from a preorder* $\leq_X$ is the relation defined as:

$$q \sim_X q' \iff q \leq_X q' \text{ and } q' \leq_X q.$$

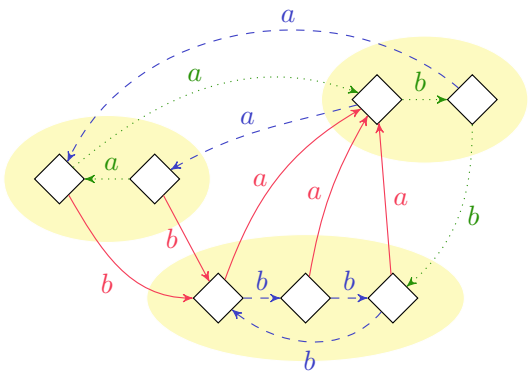
Given a preorder $\leq_X$, we always write $\sim_X$ for the induced equivalence relation, and simply write $\leq$ for the *induced order* over equivalence classes, for instance we may write $[q]_X \leq [q']_X$.

Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two binary relations over a set A (usually preorders or equivalence relations). We say that $\mathcal{R}_1$ is a *refinement* of $\mathcal{R}_2$ if for all $q, p \in A$, $q \mathcal{R}_1 p$ implies $q \mathcal{R}_2 p$. We note that if $\leq_1$ is a preorder refining $\leq_2$, then the induced equivalence relation $\sim_1$ refines $\sim_2$.

Congruences, uniformity and monotonicity. Let $\mathcal{A}$ be a (possibly non-deterministic) automaton over Σ with states Q and transitions Δ. Let $\sim$ be an equivalence relation over Q and let $\Delta' \subseteq \Delta$ be a subset of transitions (usually Δ' will be the set of transitions using a given priority in a parity automaton). We say that transitions of Δ' *are uniform over $\sim$-classes* if for all $q \sim q'$ and $a \in \Sigma$, if $q \xrightarrow{a} p \in \Delta'$ then all a-transitions $q' \xrightarrow{a}$

are in Δ'. We say that $\sim$ is a *congruence for* Δ' (or that transitions in Δ' *preserve* $\sim$) if for all $q \sim q'$ and $a \in \Sigma$, if $q \xrightarrow{a} p \in \Delta'$ then there exists $q' \xrightarrow{a} p' \in \Delta'$, and for all such transitions $p \sim p'$. If $\Delta' = \Delta$, we just say that $\sim$ is a *congruence*. We say that $\sim$ is a *strong congruence for* Δ' if, moreover, we have the equality $p = p'$ for transitions as above.

✦ Remark V.1. *If $\mathcal{A}$ is deterministic and $\sim$ is a congruence for Δ', then these transitions are uniform over $\sim$-classes.*



✦ **Figure 27.** Representation of the notions of uniformity, congruency and strong congruency. We picture an automaton with three equivalence classes, each of them represented by a yellow bubble. Green-dotted transitions are uniform over the classes, but the relation is not a congruence for them. The relation is a congruence for blue-dashed transitions, and a strong congruence for red-solid transitions.

Let $\leq$ be a preorder over Q. We say that transitions in Δ' *are monotone for* $\leq$ if for all $q \leq q'$ and $a \in \Sigma$, if $q \xrightarrow{a} p \in \Delta'$ then there exists $q' \xrightarrow{a} p' \in \Delta'$ and for all such transitions, $p \leq p'$. Transitions in Δ' are said *strictly monotone for* $\leq$ if, moreover, whenever $q < q'$, $q \xrightarrow{a} p \in \Delta'$ and $q' \xrightarrow{a} p' \in \Delta'$, we have $p < p'$. If $\Delta' = \Delta$, we simply say that $(\mathcal{A}, \leq)$ is *(strictly) monotone*.

All these properties can equivalently be stated with words $w \in \Sigma^*$ instead of letters $a \in \Sigma$.

✦ Remark V.2. *If transitions in Δ' are monotone for a preorder $\leq$, then its induced equivalence relation is a congruence for Δ'.*

Quotient by a congruence. Let $\mathcal{A}$ be an automaton and let $\sim$ be a congruence over its set of states Q. We define the *quotient of $\mathcal{A}$ by $\sim$* to be the automaton structure $\mathcal{A}/_\sim$ given by:

- The set of states are the $\sim$-classes.
- There is a transition $[q] \xrightarrow{a} [p]$ if there are $q' \in [q]$, $p' \in [p]$ such that $q' \xrightarrow{a} p'$ in $\mathcal{A}$.
- A class $[q]$ is an initial state if it contains some initial state in $\mathcal{A}$.

We note that if $\sim$ comes from a monotone preorder, the obtained automaton structure $\mathcal{A}/_\sim$ with the induced order over the classes is monotone.

✦ Remark V.3. *The quotient $\mathcal{A}/_\sim$ is a deterministic automaton structure, if all initial states of $\mathcal{A}$ are $\sim$-equivalent.*

A run over a word w in $\mathcal{A}$, $\rho = q_0 \xrightarrow{w_0} q_1 \xrightarrow{w_1} \dots$ naturally induces a run over w in $\mathcal{A}/_\sim$, $[q_0] \xrightarrow{w_0} [q_1] \xrightarrow{w_1} \dots$, that we call the *projection of ρ* in the quotient automaton.

✦ **Lemma V.4.** *Let $\sim$ be a congruence in $\mathcal{A}$. Any run in $\mathcal{A}/_\sim$ is the projection of some run in $\mathcal{A}$.*

Proof. Let $[q_0] \xrightarrow{w_0} [q_1] \xrightarrow{w_1} \dots$ be a run in $\mathcal{A}/_\sim$. We build the desired run in $\mathcal{A}$ recursively. For the base case, it suffices to take $p_0 \in [q_0]$ to be an initial state of $\mathcal{A}$ (which exists by definition of the initial states of $\mathcal{A}/_\sim$). Suppose that $p_0 \xrightarrow{w_0} p_1 \xrightarrow{w_1} \dots p_k$ has already been built, with $p_i \in [q_i]$. By definition of the quotient automaton, there are $q'_k \in [q_k]$ and $q'_{k+1} \in [q_{k+1}]$ with $q'_k \xrightarrow{w_k} q'_{k+1}$. By the definition of a congruence, there is a transition $p_k \xrightarrow{w_k} p_{k+1}$ and $p_{k+1} \in [q_{k+1}]$. ◀

Residuals and semantic determinism

Residuals of a language. Let $L \subseteq \Sigma^\omega$ be a language of infinite words and let $u \in \Sigma^*$. We define the *residual of L with respect to u* by

$$u^{-1}L = \{w \in \Sigma^\omega \mid uw \in L\}.$$

We denote $\mathsf{Res}(L)$ the set of residuals of L, which we will always order by inclusion. This induces a partial preorder $\leq_L$ over Σ^* defined by

$$u \leq_L u' \iff u^{-1}L \subseteq u'^{-1}L.$$

The induced equivalence relation $\sim_L$ is given by the equality of residuals. This yields a notion of ordered *residual classes* $[u]_L = \{u' \in \Sigma^* \mid u^{-1}L = u'^{-1}L\}$, which we sometimes write as $[u]$ when L is clear from context.

✦ Remark V.5. *A language L is prefix-independent if and only if $\mathsf{Res}(L)$ is a singleton.*

✦ Remark V.6. *If L is ω-regular, $\mathsf{Res}(L)$ is finite, and for all $u \in \Sigma^*$, $u^{-1}L$ is also ω-regular. Contrary to the case of finite words, there are non ω-regular languages with a finite set of residuals.*

We now state a key monotonicity property for residuals; its proof is a direct check.

✦ **Lemma V.7.** *For any language $L \subseteq \Sigma^\omega$ and for any finite words $u, u', w \in \Sigma^*$, if $[u] \leq [u']$ then $[uw] \leq [u'w]$. In particular, if $[u] = [u']$ then $[uw] = [u'w]$.*

Semantic determinism. We say that an automaton $\mathcal{A}$ is *semantically deterministic* if it has a single initial state and for all state $q \in Q$, letter $a \in \Sigma$ and transitions $q \xrightarrow{a} p_1$ and $q \xrightarrow{a} p_2$, it is satisfied that $\mathcal{L}(\mathcal{A}_{p_1}) = \mathcal{L}(\mathcal{A}_{p_2})$, where $\mathcal{A}_p$ is the automaton obtained by setting p as initial state.

✦ **Lemma V.8** ([KS15])**.** *Any history-deterministic automaton can be pruned into an equivalent semantically deterministic and history-deterministic automaton. Moreover, for parity automata, this can be done in polynomial time.*

Global hypothesis in Chapter V.

We assume in the whole chapter that history-deterministic automata are semantically deterministic.

We refer to [AK23] for more details on semantically deterministic automata.

Residual associated to a state. For $\mathcal{A}$ an automaton recognising L and q a state of $\mathcal{A}$, we will write $q^{-1}L$ to denote $\mathcal{L}(\mathcal{A}_q)$, and we let $q \sim_{\mathcal{A}} q'$ if $q^{-1}L = q'^{-1}L$.

If $\mathcal{A}$ is semantically deterministic and q is reachable, $q^{-1}L$ coincides with $u^{-1}L$ for any word $u \in \Sigma^*$ leading to q from the initial state of $\mathcal{A}$. In that case, we say that the *class of states* $[q]$ is *associated to* the residual class $[u]$. The inclusion of these languages induces a preorder $\leq_{\mathcal{A}}$ on the states of $\mathcal{A}$ (with its corresponding equivalence relation). By Lemma V.7, if $\mathcal{A}$ is semantically deterministic, relation $\sim_{\mathcal{A}}$ is a congruence and preorder $\leq_{\mathcal{A}}$ makes $\mathcal{A}$ a monotone automaton.

✦ Remark V.9. *An automaton $\mathcal{A}$ without unreachable states is semantically deterministic if and only if it has a single initial state and $\sim_{\mathcal{A}}$ is a congruence.*

Automaton of residuals. Let $L \subseteq \Sigma^\omega$ be a language of infinite words. The *automaton of residuals* of L, is a deterministic automaton structure $\mathcal{R}_L$ over Σ defined as follows:

- The set of states is the set of residual classes of $\mathsf{Res}(L)$: $Q = \{[u] \mid u \in \Sigma^*\}$.
- The initial state is $[\varepsilon]$.
- For each state $[u]$ and letter $a \in \Sigma$, it contains the transition $[u] \xrightarrow{a} [ua]$.

The states of $\mathcal{R}_L$ are ordered by the inclusion of residuals. By Lemma V.7, transitions of $\mathcal{R}_L$ are monotone for this order.

✦ Remark V.10. *We remark that, for any semantically deterministic automaton $\mathcal{A}$ recognising L, the automaton of residuals $\mathcal{R}_L$ coincides with the quotient of $\mathcal{A}$ by the congruence $\sim_{\mathcal{A}}$.*

Alphabets of words

As an important element of our main proof, we will need to consider automata whose transitions are labelled from an alphabet $A \subseteq \Sigma^+$ of finite words. Such an automaton defines a language $L \subseteq A^\omega$ which we would like to see as a language $L \subseteq \Sigma^\omega$; however this may pose a problem if a word $w \in \Sigma^\omega$ admits several decompositions in A^ω.

We say that a set $A \subseteq \Sigma^+$ is a *proper alphabet* if any word $w \in \Sigma^\omega$ admits a unique decomposition as elements of A: for any infinite sequences $a_1, a_2, \dots$ and $a'_1, a'_2, \dots$ of elements of A, if $a_1 a_2 \cdots = a'_1 a'_2 \dots$ then $a_i = a'_i$ for all i.

We will only consider alphabets of words $A \subseteq \Sigma^+$ which are *prefix codes*: if $a \in A$, then no proper prefix of a belongs to A. It is an easy check that these are proper alphabets, and therefore one may indeed see a language $L \subseteq A^\omega$ as $L \subseteq \Sigma^\omega$.

Further notions for parity automata

Notations on paths of parity automata. We refer to transitions of a parity automaton producing priority $x \in \mathbb{N}$ as *x-transitions*. Similarly, we refer to transitions labelled with an input letter $a \in \Sigma$ as *a-transitions*. The difference between the two uses of the term will be always clear from the context.

Letters x, y will usually stand for integers used as priorities of automata. For two states q, p of a parity automaton $\mathcal{A}$ and a finite word $w \in \Sigma^*$, we write $q \overset{w:x}{\rightsquigarrow} p$ if there exists a path from q to p labelled w such that the minimal priority appearing on it is $x \in \mathbb{N}$. We write $q \overset{w:\geq x}{\rightsquigarrow} p$ to denote that there exists such a path producing no priority strictly smaller than x. This is possibly the empty path $q \xrightarrow{\varepsilon} q$, producing no priority. We use similar notations for $\leq x$, $< x$ and $> x$. We generalise these notations for infinite paths: for an infinite word $w \in \Sigma^\omega$ we write $q \overset{w:x}{\rightsquigarrow}$ if there exists an infinite path from q labelled w such that the minimal priority seen on it is x.

We may apply this notations to non-deterministic automata – hence the use of an existential quantification – although in most cases we will work with deterministic ones.

Paths between equivalence classes. Let $\sim$ be a congruence over the states of a parity automaton $\mathcal{A}$. We write $[q] \xrightarrow{a:x} [p]$ if for all $q' \sim q$, every a-transition from q' is of the form $q' \xrightarrow{a:x} p'$ with $p' \sim p$. We extend this notations to paths $[q] \overset{w:x}{\rightsquigarrow} [p]$ and for outputs $\leq x$, $< x$, $\geq x$ and $> x$ in the natural way.

✦ Remark V.11. *If $\sim$ is a congruence for transitions producing priority x, $q \overset{w:x}{\rightsquigarrow} p$ implies $[q] \overset{w:x}{\rightsquigarrow} [p]$.*

Paths induced by a resolver. Let $\mathcal{A}$ be a history-deterministic automaton with initial state q_0 and let r be a resolver for it. We recall that we write $q_0 \overset{u:x}{\underset{\mathsf{r}}{\rightsquigarrow}} q$ to denote the run induced by r over u. We use the same conventions as above regarding outputs with the symbols $\leq x$, $< x$, $\geq x$ and $> x$.

For $u_0 \in \Sigma^*$, we write $q \overset{w:x}{\underset{u_0,\mathsf{r}}{\rightsquigarrow}} p$ if:

- $q_0 \overset{u_0}{\underset{\mathsf{r}}{\rightsquigarrow}} q$, and
- the induced run of r over $u_0 w$ ends in p and produces x as minimal priority in the part of the run corresponding to w.

We write $q \overset{w:x}{\underset{\exists,\mathsf{r}}{\rightsquigarrow}} p$ if $q \overset{w:x}{\underset{u_0,\mathsf{r}}{\rightsquigarrow}} p$ for some $u_0 \in \Sigma^*$. We write $q \overset{w:x}{\underset{\forall,\mathsf{r}}{\rightsquigarrow}} p$ if, for any word $u_0 \in \Sigma^*$ such that $q_0 \overset{u_0}{\underset{\mathsf{r}}{\rightsquigarrow}} q$, we have $q \overset{w:x}{\underset{u_0,\mathsf{r}}{\rightsquigarrow}} p$.

If $\sim$ is a congruence in $\mathcal{A}$, we write $[q] \overset{w:x}{\underset{\forall,\mathsf{r}}{\rightsquigarrow}} [p]$ if, for any word $u_0 \in \Sigma^*$ such that $q_0 \overset{u_0}{\underset{\mathsf{r}}{\rightsquigarrow}} q' \in [q]$, we have $q' \overset{w:x}{\underset{u_0,\mathsf{r}}{\rightsquigarrow}} p' \in [p]$. We avoid using this notation for paths quantified existentially, as we consider that the corresponding semantics are not as intuitive.

Restricted determinism and homogeneity. Let $\Delta' \subseteq \Delta$ be a subset of transitions of an automaton $\mathcal{A}$. We say that $\mathcal{A}$ is *deterministic over* Δ' if the restriction of $\mathcal{A}$ to Δ' is deterministic, that is, if $q \xrightarrow{a} p \in \Delta'$, then no other a-transition outgoing from q is in Δ'.

We say that a parity automaton $\mathcal{A}$ is *homogeneous* if for every state $q \in Q$ and letter $a \in \Sigma$, if $q \xrightarrow{a:x} p$ is a transition in $\mathcal{A}$, then any other a-transition from q produces priority x. For $0 \leq x \leq d$, we say that $\mathcal{A}$ is *$(\leq x)$-homogeneous* if the previous property is satisfied for priorities $\leq x$.

✦ Remark V.12. *Let $\mathcal{A}$ be an homogeneous parity automaton that is deterministic over transitions producing priority x. If $q \xrightarrow{a:x} p$ is a transition in $\mathcal{A}$, then there is no other a-transition outgoing from q in $\mathcal{A}$.*

Priority accepting a word. Let $\mathcal{A}$ be a parity automaton, and $w \in \Sigma^\omega$ an infinite word. For an even priority x, we say that w can be *accepted with priority x* in $\mathcal{A}$ if there exists a run over w such that the minimal priority produced infinitely often is x. For an odd priority x, we say that w *is rejected with priority x* if the minimal priority produced infinitely often in every run over w is x.

Automata with ε-transitions. An *automaton with ε-transitions* is defined just as an automaton over the alphabet $\Sigma \sqcup \{\varepsilon\}$, where $\varepsilon \notin \Sigma$ is a distinguished letter. The language of an automaton $\mathcal{A}$ with ε-transitions is the set of words $w \in \Sigma^\omega$ such that there exists $w' \in (\Sigma \sqcup \{\varepsilon\})^\omega$ which is accepted by $\mathcal{A}$ and such that w is obtained from w' by removing all occurrences of the letter ε.

✧ Note. *Note that ε-transitions in this context do not correspond to transitions producing no output colour, as in the ε-edges of games.*

Reminder on the normal form of parity automata. In this chapter, we will extensively use the normal form of parity automata, as presented in Section II.7. We recall now the central property that will be used repeatedly.[1]

Lemma V.13 (Fundamental property of the normal form)**.**

Let $\mathcal{A}$ be a parity automaton in normal form and $x > 0$. If there is a path $q \overset{w:x}{\rightsquigarrow} p$ producing x as minimal priority, then, for every $0 \leq y \leq x$, there is a returning path $p \overset{w':y}{\rightsquigarrow} q$ producing y as minimal priority.

That is, if $\mathcal{A}$ is in normal form, the restriction of $\mathcal{A}$ to priorities $\geq x$ consists in a disjoint union of strongly connected components. Moreover, if priority $y > x$ appears in one of these SCCs, then all priorities between x and y appear in that SCC. This two properties can be taken as the definition of the normal form (Theorem II.14). We remark that they do only refer to the colouring with priorities of the automaton, and not to the input alphabet or to the semantics of it.

Also, we can normalise a given parity automaton in polynomial time (Proposition II.132).

Universal graphs

We now introduce universal graphs, which will be our main tool for deriving positionality results. First, we need some formal definitions about graphs.

Σ-graphs. A *Σ-graph*, where Σ is a set of colours, is a (potentially infinite) graph $G = (V, E, \mathsf{source}, \mathsf{target})$ together with a map $\mathsf{col} : E \to \Sigma$.[2] We denote $v \xrightarrow{c} v'$ in G

[1] For simplicity, we omit here technical details to be addressed if priority 0 does not appear in the automaton $\mathcal{A}$ or in some of its SCCs.

[2] This definition is quite close to that of transition systems used in Chapter II. However, we will make a fundamentally different use of Σ-graphs in this chapter: we will focus on infinite graphs, and the notion of morphism will not correspond to that of morphisms of transition systems.

to refer to an edge in G with source v, target v', and colour c. This notation naturally extends to finite and infinite paths. Often, we define Σ-graphs simply by describing the set of transitions $v \xrightarrow{c} v'$ that occur, and without formally giving names to edges. The *size* of a graph G is defined to be the cardinality of V. The following assumption is useful when considering infinite paths.

Global hypothesis in Chapter V.

All Σ-graphs we consider do not admit sinks, that is, every vertex has at least one outgoing edge.

Morphisms of Σ-graphs. Given two Σ-graphs $G = (V, E, \mathsf{source}, \mathsf{target}, \mathsf{col})$ and $G' = (V', E', \mathsf{source}', \mathsf{target}', \mathsf{col}')$, a *morphism of Σ-graphs* ϕ from G to G' is a map $\phi : V \to V'$ such that for each edge $v \xrightarrow{c} v'$ in G, it holds that $\phi(v) \xrightarrow{c} \phi(v')$ defines an edge in G'. We write $\phi : G \to G'$ to denote that ϕ is a morphism.

Universality. Given a Σ-graph G, a vertex v of G and an objective $W \subseteq \Sigma^\omega$, we say that *$v$ satisfies W* in G if for any infinite path $v \overset{w}{\rightsquigarrow}$ in G, it holds that $w \in W$. Given a cardinal κ, a graph U is *(κ, W)-universal* if all graphs G of size $< \kappa$ admit a morphism $\phi : G \to U$ such that any vertex v that satisfies W in G is mapped to a vertex $\phi(v)$ that satisfies W in U.

Monotonicity. A *totally ordered graph* (resp. *well-ordered graph*) is a graph G together with a total order (resp. well-order) $\leq$ on its vertex set V. Such a graph is called *monotone* if

$$u \leq v,\ v' \leq u' \text{ and } u \xrightarrow{c} u' \text{ in } G \implies v \xrightarrow{c} v' \text{ in } G.$$

We often note the conditions on the left by $v \geq u \xrightarrow{c} u' \geq v'$.

We now state our main tool for proving half-positionality.

Proposition V.14 ([Ohl23a, Theorem 3.1])**.**

Let $W \subseteq \Sigma^\omega$ be an objective. If for all cardinals κ there exists a (κ, W)-universal well-ordered monotone graph, then W is half-positional over all games.

Universality for trees. It is often more convenient to work with trees. In this chapter, a *Σ-tree* is a Σ-graph T with a distinguished vertex t_0, called the *root*, and such that every vertex t of T admits a unique path from the root. Since graphs (and in particular trees) are assumed sinkless, trees are always infinite. We say that a tree T *satisfies* W if its root t_0 satisfies W in T.

We say that a graph U is *(κ, W)-universal for trees* if all trees T of size $< \kappa$ which satisfy W admit a morphism $\phi\colon T \to U$ mapping the root t_0 to a vertex $\phi(t_0)$ that satisfies W in U.

Given an ordered Σ-graph U, we let $U^\top$ be the Σ-graph obtained by adding a fresh vertex $\top$, maximal for the order of the graph, with transitions $\top \xrightarrow{a} v$ for every $a \in \Sigma$ and every vertex v of the graph. The following useful result follows directly from the proof of [Ohl23a, Theorem 3.1] (see also [CO23, Theorem 3]).

✦ **Lemma V.15.** *Let $W \subseteq \Sigma^\omega$ be an objective and κ a cardinal. If U is a a well-ordered monotone graph that is (κ, W)-universal for trees, then $U^\top$ is well-ordered monotone (κ, W)-universal (for graphs).*

Universal graph for the parity objective. As an important example, we give a universal graph for the parity objective; it is implicit in the works of Emerson and Jutla [EJ91] and Walukiewicz [Wal96]. In the latter, the term *signatures* was used to name tuples of ordinals ordered lexicographically (term first used in [SE89]). Such a tuple is meant to represent how many small odd priorities we can allow to see in a play while still being winning.

Example V.16 (*Universal graph for the parity objective*)**.**

Consider the parity objective over $[0, d]$, (we assume d even):

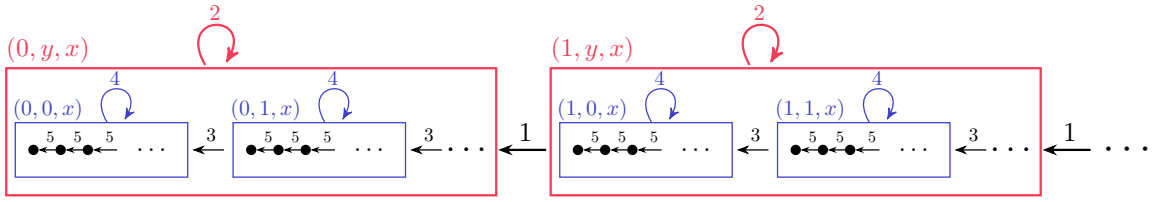
$$\mathsf{parity} = \{w \in \{0, \dots, d\}^\omega \mid \liminf w \text{ is even}\}.$$

Fix a cardinal κ. We define a graph U_{parity} having as set of vertices tuples $(\lambda_1, \lambda_3, \dots, \lambda_{d-1}) \in \kappa^{d/2}$ that we consider ordered lexicographically. This is indeed a well-order. We let its edges be:

$$(\lambda_1, \dots, \lambda_{d-1}) \xrightarrow{x} (\lambda'_1, \dots, \lambda'_{d-1}) \iff \begin{cases} (\lambda'_1, \dots, \lambda'_{x-1}) \leq (\lambda_1, \dots, \lambda_{x-1}) & \text{if } x \text{ is even,} \\ (\lambda'_1, \dots, \lambda'_x) < (\lambda_1, \dots, \lambda_x) & \text{otherwise.} \end{cases}$$

Where the order between truncated tuples as on the right is also the lexicographic one. A representation of the graph U_{parity} appears in Figure 28.

Clearly, U_{parity} is monotone. We show in Lemma V.17 below that all vertices in U_{parity} satisfy parity. Lemma V.18 states that U_{parity} is $(\kappa, \mathsf{parity})$-universal for trees, so, by Lemma V.15, $U^\top_{\mathsf{parity}}$ is a well-ordered monotone $(\kappa, \mathsf{parity})$-universal graph.



✦ **Figure 28.** Universal graph U_{parity} for the parity objective over priorities $[0, 5]$. Vertices are ordered from left to right. Edges between two boxes $B_1 \xrightarrow{x} B_2$ represent that there are edges $v_1 \xrightarrow{x} v_2$ for all $v_1 \in B_1$ and all $v_2 \in B_2$. Edges obtained by monotonicity are not represented: if $v \xrightarrow{x} v'$ and $v'' \leq v'$, then $v \xrightarrow{x} v''$ too; for example, by reading colour 5 from a vertex v one can go to any vertex strictly on the left of v. Edges coloured 0 are not depicted in the figure: they appear between every pair of vertices. The label of a box represents the forms of the names of vertices inside it.

✦ **Lemma V.17.** *Every infinite path in U_{parity} satisfies the objective* parity.

Proof. Consider an infinite path $\rho = (\lambda^1_1, \dots, \lambda^1_{d+1}) \xrightarrow{w_1} (\lambda^2_1, \dots, \lambda^2_{d+1}) \xrightarrow{w_2} \dots$ in U_{parity}. Let x be the minimal priority appearing infinitely often in ρ, which we assume odd for

contradiction. Then, from some position, no priority $> x$ is read, thus the sequence of prefixes $(\lambda_1^i, \dots, \lambda_x^i)$ is decreasing and moreover strictly decreases in infinitely many places. This contradicts well-foundedness of the lexicographical order over tuples of ordinals. ◀

✦ **Lemma V.18.** *Graph U_{parity} is $(\kappa, \mathsf{parity})$-universal for trees.*

Proof. Take a tree T of size $< \kappa$ which satisfies parity; note that by prefix-independence, all vertices in T satisfy parity. We aim to construct a morphism $\phi : T \to U_{\mathsf{parity}}$.

Fix a vertex $t \in T$ and an odd priority $y \in \{1, 3, \dots, d-1\}$. Then, in any path $t \overset{w}{\rightsquigarrow}$ in T there are only finitely many occurrences of y before a smaller priority appears. We may define an ordinal $\mathsf{rank}_y(v)$ capturing the number of such occurrences: $\mathsf{rank}_y(v)$ satisfies that, if v' is an x-sucessor of v (that is, $v \xrightarrow{x} v'$), then:

- $\mathsf{rank}_y(v') \leq \mathsf{rank}_y(v)$, if $y < x$, and
- $\mathsf{rank}_y(v') < \mathsf{rank}_y(v)$ if $x = y$.

One easily verifies that $\phi : v \mapsto (\mathsf{rank}_1(v), \mathsf{rank}_3(v), \dots, \mathsf{rank}_{d-1}(v))$ defines a morphism from T to U_{parity}. ◀

Concavity

For comparison purposes, we define concave objectives. However, this notion will play no role in our characterisation or our proofs. Concavity was introduced by Kopczyński [Kop08] as an approximation to half-positionality: prefix-independent concave objectives are half-positional over finite games [Kop08, Theorem 4.7], but the converse does not hold. This notion was further studied for non-prefix-independent objectives in [Bia+11].

We say that an objective $W \subseteq \Sigma^\omega$ is *concave* if for all pairs of sequences of finite words $u_1, u_2, \dots \in \Sigma^+$ and $w_1, w_2, \dots \in \Sigma^+$ we have:

$$u_1 u_2 \dots \notin W \text{ and } w_1 w_2 \dots \notin W \implies u_1 w_1 u_2 w_2 \dots \notin W.$$

That is, the shuffle of any pair of losing words is a losing word.

2 Half-positionality of ω-regular objectives: Statement of the results

In this section, we state the central result of the chapter and its consequences: a full characterisation of deterministic parity automata recognising half-positional ω-regular languages (Theorem V.1). The statement of the theorem uses terminology that will be formally introduced in Section V.4; here we just provide some intuitive explanations.

2.1 Characterisation of half-positionality for ω-regular objectives

We state our main characterisation theorem. Items are ordered following the sequence of implications as they will be shown in the chapter.

Theorem V.1.

Let $W \subseteq \Sigma^\omega$ be an ω-regular objective. The following are equivalent:

1. W is half-positional over finite ε-free Eve-games.
2. There is a deterministic fully progress consistent signature automaton recognising W.
3. There is a history-deterministic ε-complete automaton recognising W.
4. For all cardinals κ, there is a well-ordered monotone (κ, W)-universal graph.
5. W is half-positional over all games (potentially infinite and containing ε-edges).

The central notion of our characterisation is that of *signature automata* (name chosen because of their similarities with the structure of the universal graph U_{parity} for the parity objective). These are parity automata with a very restricted syntactic structure: for all priorities x there is a total preorder $\leq_x$ over the states, such that these refine one another and satisfy some monotonicity properties (see Section V.4.1 for the precise definition). Our first main technical contribution is to show that any half-positional ω-regular objective W can be recognised by a signature automaton (implication from (1) to (2)). This is achieved by applying a number of transformations to a given parity automaton, until obtaining an automaton with all the desired structural properties. The final automaton satisfies a further property – necessary for half-positionality – that we call full progress consistency: words making a strict progress in the automaton with respect to some of the preorders must be accepted if repeated infinitely often.

The second challenge is to prove that any such automaton indeed recognises a half-positional objective (implication from (2) to (5)). To do so, we rely on Ohlmann's characterisation of half-positionality via universal graphs [Ohl23a] (restated as Proposition V.14, and providing implication from (4) to (5)). To obtain universal graphs, we consider as an intermediate step ε-complete automata (see Section V.4.3). These are non-deterministic automata in which all pairs of states can be compared using ε-transitions of any priority. The advantage is that they admit a much simpler definition and their transitions are closed by monotonicity, which allows for a smooth transition towards monotone graphs and the implication from (3) to (4).

We now discuss consequences of Theorem V.1.

2.2 Main consequences on half-positionality

Decidability of half-positionality in polynomial time

Theorem V.2.

Given a deterministic parity automaton $\mathcal{A}$, we can decide in polynomial time whether $\mathcal{L}(\mathcal{A})$ is half-positional.

Although this result is not directly implied by Theorem V.1, it will follow from its proof. Indeed, in Section V.4.2 we provide a procedure building (when possible) a deterministic signature automaton from a deterministic parity automaton. All the transfor-

mations used in this construction can be computed in polynomial time. Existence of a deterministic signature automaton alone does not suffice to guarantee half-positionally, as we also need the signature automaton to be fully progress consistent. Lemma V.78 guarantees that, if $\mathcal{L}(\mathcal{A})$ is half-positional, the signature automaton obtained by our procedure must be fully progress consistent. We can check if this is indeed the case in polynomial time (Lemma V.80). The final decision algorithm is as follows; let $\mathcal{A}$ be a DPA recognising W:

- We apply the transformation turning a DPA into a signature automaton. If some step of the procedure fails, we conclude that W is not half-positional.
- If the procedure terminates, we obtain a signature automaton $\mathcal{A}'$ recognising W.
- We check whether $\mathcal{A}'$ is fully progress consistent.

The details of the proof of Theorem V.2 can be found at the end of Section V.4.2.

Finite-to-infinite and 1-to-2 player lifts

The following result simply restates the implication (1) $\implies$ (5) from Theorem V.1.

Theorem V.3.

If an ω-regular objective is half-positional over finite, ε-free Eve-games, then it is half-positional over all games (potentially infinite and containing ε-edges).

Closure under union of prefix-independent half-positional languages

We now show that Kopczyński's conjecture holds for ω-regular languages: prefix-independent half-positional languages are closed under union. In fact, we show a stronger result: it suffices to suppose that only one of the objectives is prefix-independent.

Theorem V.4.

Let $W_1, W_2 \subseteq \Sigma^\omega$ be two half-positional ω-regular objectives, and suppose that W_1 is prefix-independent. Then, $W_1 \cup W_2$ is half-positional.

In order to obtain this theorem, we use the 1-to-2-players lift stated in Theorem V.3. The result from Theorem V.4 can be easily obtained for Eve-games, so it suffices then to apply the lift to get the result for all types of games.

Lemma V.19.

Let $W_1, W_2 \subseteq \Sigma^\omega$ be two objectives that are half-positional over Eve-games, and suppose that W_1 is prefix-independent. Then, $W_1 \cup W_2$ is half-positional over Eve-games.

Proof. Let $\mathcal{G}$ be an Eve-game using $W_1 \cup W_2$ as winning condition. We show that Eve has a positional strategy that wins from any vertex of her winning region. We let $\mathcal{G}_1$ be the game with the same underlying graph than $\mathcal{G}$ and W_1 as winning condition. Consider Eve's winning region in this game, $\mathcal{W}in_{\text{Eve}}(\mathcal{G}_1)$. By half-positionality of W_1, she has a positional strategy strat_1 ensuring to produce paths labelled with W_1 from states in $\mathcal{W}in_{\text{Eve}}(\mathcal{G}_1)$. Moreover, by prefix-independence of W_1, there is no path leading to $\mathcal{W}in_{\text{Eve}}(\mathcal{G}_1)$ from a vertex that is not in this winning region.

We let $\mathcal{G}_2$ be the game with $\mathcal{G} \setminus \mathcal{W}in_{\mathrm{Eve}}(\mathcal{G}_1)$ as underlying graph, and using W_2 as winning condition. By half-positionality of W_2, Eve has a positional strategy strat_2 for this game that is winning from $\mathcal{W}in_{\mathrm{Eve}}(\mathcal{G}_2)$.

We consider the positional strategy strat in $\mathcal{G}$ that coincides with strat_1 over $\mathcal{W}in_{\mathrm{Eve}}(\mathcal{G}_1)$ and coincides with strat_2 over $\mathcal{G}_2$. It is clear that this strategy is winning from vertices in $\mathcal{W}in_{\mathrm{Eve}}(\mathcal{G}_1) \cup \mathcal{W}in_{\mathrm{Eve}}(\mathcal{G}_2)$. We show that these are all vertices from which Eve can win $\mathcal{G}$, so strat is an optimal strategy.

✧ Claim V.19.1. $\mathcal{W}in_{\mathrm{Eve}}(\mathcal{G}) = \mathcal{W}in_{\mathrm{Eve}}(\mathcal{G}_1) \cup \mathcal{W}in_{\mathrm{Eve}}(\mathcal{G}_2)$.

Proof. Let v be a vertex in $\mathcal{G}$ such that Eve wins from it. A strategy in an Eve-game is just an infinite path in the underlying graph. Let therefore ρ_v be an infinite path from v labelled with a word $w \in W_1 \cup W_2$. Suppose that $v \notin \mathcal{W}in_{\mathrm{Eve}}(\mathcal{G}_1)$. In particular $w \notin W_1$, so $w \in W_2$. As there is no path leading to $\mathcal{W}in_{\mathrm{Eve}}(\mathcal{G}_1)$ from a vertex that is not in this region, the path ρ_v is contained in $\mathcal{G} \setminus \mathcal{W}in_{\mathrm{Eve}}(\mathcal{G}_1)$, which is the underlying graph of $\mathcal{G}_2$. Therefore, $v \in \mathcal{W}in_{\mathrm{Eve}}(\mathcal{G}_2)$. ◁

◀

The question remains open for arbitrary objectives.

Conjecture V.2 (*Kopczyński's conjecture* – Version of [Kop08, Conjecture 7.1])**.**

Let $W_1, W_2 \subseteq \Sigma^\omega$ be two half-positional objectives (over all games), and suppose that W_1 is prefix-independent. Then, $W_1 \cup W_2$ is half-positional.

Closure of positionality under addition of neutral letters

As mentioned in the introduction, Ohlmann recently characterised half-positional objectives by means of the existence of universal graphs [Ohl23a]. One direction (stated in Proposition V.14) holds for any objective: if W admits well-ordered monotone universal graphs, then it is half-positional. To obtain the converse, the proof proposed by Ohlmann requires a further hypothesis: W has to contain a neutral letter, that is a letter that can be removed from any word without modifying the membership in W. In his work, he left open the problem of whether adding a neutral letter preserves positionality. This is a central question in the theory of positionality, as it would imply that universal graphs completely characterise half-positionality without any further hypothesis on the objectives. This question is almost[3] equivalent to the one raised by Kopczyński in his PhD thesis [Kop08, Section 2.5]: if W is half-positional over ε-free games, is it half-positional over all games?

Let $W \subseteq \Sigma^\omega$ be an objective. A letter $c \in \Sigma$ is *neutral for* W if, for all $w_1, w_2, \dots \in \Sigma^+$ and $n_1, n_2, \dots \in \mathbb{N}$:

- $c^{n_1} w_1 c^{n_2} w_2 \dots \in W \iff w_1 w_2 \dots \in W$, and
- $w_1 c^\omega \in W \iff {w_1}^{-1} W \neq \emptyset$.

Given an objective W, we let W^ε denote the unique objective obtained by adding a fresh neutral letter ε to W.

[3] The only difference is that in ε-free games we assume that there are no infinite paths composed exclusively of ε-edges, whereas these may appear in W^ε-games.

Proposition V.20 ([Ohl23a]).

Let $W \subseteq \Sigma^\omega$. Objective W^ε is half-positional if and only if for all cardinal κ there is a well-ordered monotone (κ, W)-universal graph.

Conjecture V.3 (*Neutral letter conjecture* [Ohl23a]).

For every half-positional objective W, objective W^ε is half-positional.

Our characterisation (Item (4) in Theorem V.1), together with Proposition V.20, answers this question in the case of ω-regular objectives.

Theorem V.5.

Let $W \subseteq \Sigma^\omega$ be an ω-regular objective. If W is half-positional, then W^ε is half-positional. Also, W is half-positional over ε-free games if and only if W is half-positional over all games.

3 Warm-up: Illustrating ideas on restricted classes of languages

The goal of this section is to give a gentle introduction to the techniques and ideas which are used in the proof of our main result (implication from (1) to (2) in Theorem V.1). We single out four crucial properties that a parity automaton recognising a half-positional objective should satisfy.

- Half-positional objectives have totally ordered residuals, which define a congruence over states of automata recognising them.
- This order should satisfy a semantical property called progress consistency.
- Transitions with priority 0 preserve this congruence.
- In each congruence class, states which are interreachable using paths avoiding priorities ≤ 1 have comparable (≤ 1)-safe languages.

We propose to study four restricted classes of ω-regular objectives that allow us to isolate these different points, namely, closed objectives, open objectives, and those recognised by deterministic Büchi and deterministic coBüchi automata. Considering objectives in these four classes allows us to illustrate the necessity of the four properties above, and the techniques we use to derive them. In each case, we state a characterisation of half-positionality and give a full proof of necessity, which is the more difficult direction. These characterisation and proof techniques are generalised to all ω-regular languages in our main inductive proof of necessity (Section V.4.2).

We moreover incorporate in this section many examples illustrating our results and the ideas in our proofs.

3.1 Closed objectives and total order on the residuals

We now discuss the first property announced above: residuals of half-positional objectives are totally ordered by inclusion. The necessity of this condition holds even for non-ω-regular objectives.

Residuals of half-positional objectives are totally ordered

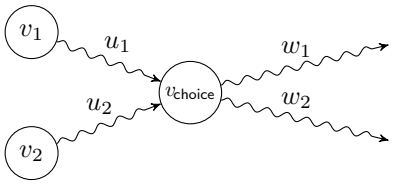
Lemma V.21.

If an objective $W \subseteq \Sigma^\omega$ is half-positional, then $\mathsf{Res}(W)$ is totally ordered by inclusion.

Proof. We show the contrapositive. Suppose that W has two incomparable residuals, ${u_1}^{-1}W$ and ${u_2}^{-1}W$. Take $w_1 \in u_1^{-1}W \setminus u_2^{-1}W$ and $w_2 \in u_2^{-1}W \setminus u_1^{-1}W$. Stated differently, we have

$$u_1w_1 \in W, \quad u_1w_2 \notin W,$$
$$u_2w_1 \notin W, \quad u_2w_2 \in W.$$

Consider the (infinite) Eve-game $\mathcal{G}$ represented in Figure 29. Eve wins $\mathcal{G}$ from v_1 and v_2: if a play starts in v_i, for $i = 1, 2$, she just has to take the path labelled w_i from v_{choice}. However, she cannot win from both v_1 and v_2 using a positional strategy. Indeed, such a positional strategy would choose one transition $v_{\mathsf{choice}} \overset{w_i}{\rightsquigarrow}$, and the play induced when starting from v_{1-i} would be losing.



✦ **Figure 29.** A game $\mathcal{G}$ in which Eve cannot play optimally using positional strategies if $\mathsf{Res}(W)$ is not totally ordered. ◀

Closed objectives

Let Σ be a set of letters and $L \subseteq \Sigma^*$ be a language of finite words. The *safety objective associated to L* is defined by

$$\mathsf{Safety}(L) = \{w \in \Sigma^\omega \mid w \text{ does not contain any prefix in } L\}.$$

An objective W is *topologically closed* if $W = \mathsf{Safety}(L)$ for some $L \subseteq \Sigma^*$. This terminology is justified since objectives of the form $\mathsf{Safety}(L)$ are exactly the closed subsets of Σ^ω for the Cantor topology (see for example [Tho91]).

✦ Remark V.22. *An objective $W = \mathsf{Safety}(L)$ is ω-regular if and only if L is a regular language of finite words if and only if $\mathsf{Res}(W)$ is finite. We refer to this class as* ω-regular closed objectives.

It turns out that for ω-regular closed objective, the converse of Lemma V.21 holds. This was first established in [CFH14].

Proposition V.23 (Half-positionality of closed objectives [CFH14])**.**

Let $W \subseteq \Sigma^\omega$ be an ω-regular closed objective. Then, W is half-positional if and only if $\mathsf{Res}(W)$ is totally ordered by inclusion.

Thus, residuals encode the information needed to decide whether an ω-regular closed objective is half-positional. We do not include a proof of sufficiency in this warm-up;

a proof for all (non-necessarily ω-regular) closed objectives is given in Theorem V.7. However, a much subtler understanding is needed for non-closed objectives, as witnessed by the example below.

Example V.24 (Non-positional open objective)**.**

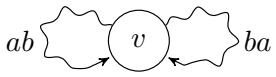
Consider the non-closed objective

$$W = \{w \in \Sigma^\omega \mid w \text{ contains the factor } aa\}.$$

Its three residuals are totally ordered by inclusion:

$$\varepsilon^{-1}W \subseteq a^{-1}W \subseteq aa^{-1}W.$$

However, it is not half-positional, as witnessed by the game in Figure 30.



✦ **Figure 30.** A game $\mathcal{G}$ in which Eve cannot produce the factor aa positionally.

3.2 Open objectives and progress consistency

We now introduce progress consistency, a semantical property of the order of residuals which is necessary for half-positionality. For ω-regular open objectives, this property, together with the total order of residuals is also sufficient.

Progress consistency

Definition V.25 (Progress consistency)**.**

An objective $W \subseteq \Sigma^\omega$ is *progress consistent* if for all $u, w \in \Sigma^*$:

$$[u]_W < [uw]_W \implies uw^\omega \in W.$$

Intuitively, a progress consistent objective satisfies that whenever we read a word that makes some strict progress with respect to the order of the residuals, by repeating this word we produce a sequence in W.

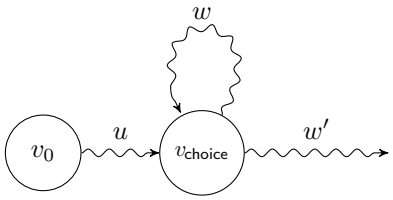
We remark that the objective $\mathsf{Reach}(\Sigma^*aa)$ from Example V.24 is not progress consistent, as the word ba makes progress from residual $\varepsilon^{-1}W$, but $(ba)^\omega \notin W$.

Let us establish necessity of progress consistency for half-positional objectives.

Lemma V.26 (Necessity of progress consistency)**.**

Any half-positional objective is progress consistent.

Proof. We show the contrapositive of the statement. Let W be an objective that is not progress consistent, that is, there are $u, w \in \Sigma^*$ such that $[u] < [uw]$ and $uw^\omega \notin W$. Let $w' \in uw^{-1}W \setminus u^{-1}W$. Consider the game $\mathcal{G}$ depicted in Figure 31.



✦ **Figure 31.** A game $\mathcal{G}$ in which Eve cannot play optimally using positional strategies if W is not progress consistent.

Eve wins game $\mathcal{G}$ from vertex v_0 by producing the play

$$v_0 \overset{u}{\rightsquigarrow} v_{\text{choice}} \overset{w}{\rightsquigarrow} v_{\text{choice}} \overset{w'}{\rightsquigarrow} .$$

However, she cannot win positionally from v_0 since positional strategies produce either uw^ω or uw', and both of these words are losing. ◀

✦ Remark V.27. *The previous lemma applies, in particular, to ω-regular closed objectives. We did not need to add progress consistency as an hypothesis in Proposition V.23, as this property is granted for closed objective by Lemma V.7.*

We are now ready to move on to the characterisation of ω-regular open objectives.

Open objectives

We now study the dual of closed objectives, namely, open ones. Let $L \subseteq \Sigma^*$. The *reachability objective associated to L* is defined by

$$\mathsf{Reach}(L) = \{w \in \Sigma^\omega \mid w \text{ contains a prefix in } L\}.$$

An objective W is *topologically open* if $W = \mathsf{Reach}(L)$ for some $L \subseteq \Sigma^*$. (These are the open subsets of Σ^ω for the Cantor topology.) Similarly to the previous subsection, we define the class of *ω-regular open objectives* as those that are both open and ω-regular.

✦ Remark V.28. *An open objective $W = \mathsf{Reach}(L)$ is ω-regular if and only if L is a regular language of finite words if and only if $\mathsf{Res}(W)$ is finite.*

Let us state a characterisation of half-positionality for ω-regular open objectives. Characterisations for the full classes of open and closed objectives (without ω-regularity assumptions) will be obtained in Section V.6.

Proposition V.29 (Half-positionality for open objectives)**.**

An ω-regular open objective W is half-positional if and only if it is progress consistent and its set of residuals $\mathsf{Res}(W)$ is totally ordered.

Necessity follows from combining Lemmas V.21 and V.26; we omit a proof of sufficiency in this warp-up. In particular, we obtain the following corollary of Propositions V.23 and V.29.

Corollary V.30.

Any half-positional ω-regular open objective is bipositional.

We now give an example of an objective that satisfies the requirement from the previous proposition.

Example V.31 (Half-positional open objective)**.**

Consider the ω-regular open objective

$$W_n = \mathsf{Reach}((a\Sigma^*)^n).$$

It was introduced (for $n = 2$) in [Bia+11, Lemma 13] as an example of a bipositional objective which is not concave. It residuals are given by

$$\varepsilon^{-1}W < a^{-1}W < aa^{-1}W < \cdots < a^{n-1}W = \Sigma^\omega,$$

which are totally ordered. Moreover, for any residual class $[a^i]$ with $i < n$, we have $[a^i] < [a^iu]$ if and only if u contains the letter a, in which case $a^iu^\omega \in W$. Therefore, W is progress consistent, so we conclude that it is bipositional.

Many natural examples of objectives are in fact prefix-independent; for those, the two conditions about the residuals above are trivially satisfied. Yet, this does not suffice to guarantee their positionality. We continue our introductory exploration with objectives recognised by deterministic Büchi automata.

3.3 Objectives recognised by Büchi automata: Uniformity of 0-transitions

Our goal in this section is to present another property of half-positional ω-regular objectives W, namely, that they can be recognised by a deterministic parity automaton $\mathcal{A}$ in which 0-transitions are uniform over each residual class:

$$\text{For any } q \sim_{\mathcal{A}} q' \text{ and } a \in \Sigma, \text{ if } q \xrightarrow{a:0} \text{ then } q' \xrightarrow{a:0}.$$

In our main induction (Section V.4.2), we will derive a similar property for all even priorities. To illustrate the technique, we now only focus on the case where W is recognised by a deterministic Büchi automaton $\mathcal{A}$ (that is, W has parity index at most $[0, 1]$), which helps alleviate some of the technicalities while preserving the important ideas behind the proof. On the way, we characterise half-positionality for these objectives, reobtaining the main result of [Bou+22].

The proof is split into two parts: first, we focus on the prefix-independent case, and then reduce to it.

Prefix-independent objectives recognized by deterministic Büchi automata

It is well-known that Büchi languages – that is, those of the form $W = \mathsf{Buchi}_\Sigma(B)$ for some $B \subseteq \Sigma$ – are half-positional [EJ91]. We prove now that these are the only half-positional prefix-independent objectives recognised by deterministic Büchi automata.

Proposition V.32 (Half-positionality for prefix-independent Büchi objectives)**.**

A prefix-independent objective W recognised by a deterministic Büchi automaton is half-positional if and only if it is a Büchi language.

In particular, Proposition V.32 tells us that there is a Büchi automaton with just one state recognising W. In this automaton, 0-transitions are trivially uniform.

We now concentrate on proving Proposition V.32. Our proof considerably simplifies that of [Bou+22, Proposition 11].

Super words and super letters. We say that $u \in \Sigma^+$ is a *super word* (for W) if, for every $w \in \Sigma^\omega$, if w contains u infinitely often as a factor, then $w \in W$. If u is a letter, we say that it is a *super letter*. Let $B_W \subseteq \Sigma$ be the set of super letters for W. It is clear that $\mathsf{Buchi}_\Sigma(B_W) \subseteq W$. We will show that, if W is half-positional, this is in fact an equality.

✦ **Lemma V.33** (Existence of super letters)**.** *A non-empty prefix-independent half-positional objective W recognised by a deterministic Büchi automaton admits a super letter.*

One may easily deduce Proposition V.32 from Lemma V.33: the restriction W' of W to non-super letters is a prefix-independent half-positional objective of parity index $[0, 1]$ which contains no super letter. Thus, Lemma V.33 tells us that $W' = \emptyset$ and therefore $W = \mathsf{Buchi}_\Sigma(B_W)$.

Proof of Lemma V.33. Fix a non-empty prefix-independent half-positional objective W recognised by a deterministic Büchi automaton $\mathcal{A}$, and assume without loss of generality that $\mathcal{A}$ is strongly connected (thus, every state can be chosen initial) and in normal form. Since W is non-empty, note that $\mathcal{A}$ must contain a transition with priority 0. We will use the following observation.

✧ Claim V.33.1 (Super words in Büchi automata). *A word $w \in \Sigma^+$ is a super word if and only if for all states q of $\mathcal{A}$, priority 0 appears on the path $q \overset{w:0}{\rightsquigarrow}$. This is in particular the case if w is a letter.*

Proof. By normality of $\mathcal{A}$ (Lemma V.13), if there is q such that $q \overset{w:1}{\rightsquigarrow} q'$ then there is a word $w' \in \Sigma^*$ labelling a returning path $q' \overset{w':1}{\rightsquigarrow} q$. Therefore, $(ww')^\omega \notin W$, so w is not a super word. The converse implication is clear, since each time word w is read, the automaton produces priority 0. ◁

We now prove existence of super words.

✧ Claim V.33.2 (Existence of super words). *There is a super word for $\mathcal{L}(\mathcal{A})$.*

Proof. We note that, as $\mathcal{A}$ is strongly connected and contains some priority 0, for each state q there is a finite word that produces priority 0 when read from q. We let $\{q_1, q_2, \dots, q_k\}$ be an enumeration of the states of $\mathcal{A}$ and recursively define k finite words $w_1, w_2, \dots, w_k \in \Sigma^*$ satisfying:

$$q_i \overset{w_1 w_2 \dots w_{i-1}}{\rightsquigarrow} q' \overset{w_i:0}{\rightsquigarrow} q''.$$

In words, for $i \in 1, \dots, k$, reading $w_1 w_2 \dots w_i$ from q_i, produces priority 0. This implies that $w_1 w_2 \dots w_i$ produces priority 0 when read from any q_j for $j \leq i$. Therefore,

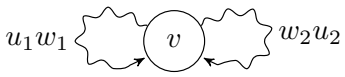
$w = w_1 w_2 \dots w_k$ produces priority 0 when read from any state of $\mathcal{A}$, so by Claim V.33.1, w is a super word. ◁

We now prove that, by half-positionality of W, super words can be chopped into smaller super words. This implies Lemma V.33 by repeatedly chopping a super word obtained from Claim V.33.2 until obtaining a super letter.

✧ Claim V.33.3 (Chopping super words). *Let $w = w_1 w_2 \in \Sigma^+$ be a super word. Then either w_1 or w_2 is a super word.*

Proof. Suppose by contradiction that neither w_1 nor w_2 are super words. Then, by Claim V.33.1, there are states q_1 and q_2 such that $q_1 \overset{w_1:1}{\rightsquigarrow} q_1'$ and $q_2 \overset{w_2:1}{\rightsquigarrow} q_2'$. By normality, we obtain returning paths $q_1' \overset{u_1:1}{\rightsquigarrow} q_1$ and $q_2' \overset{u_2:1}{\rightsquigarrow} q_2$. Therefore, $(w_1 u_1)^\omega \notin W$ and $(w_2 u_2)^\omega \notin W$. We consider the game $\mathcal{G}$ depicted in Figure 32. Eve can win this game, as alternating the two self loops she produces the word $(u_1 w_1 w_2 u_2)^\omega$, which belongs to $\mathcal{L}(\mathcal{A})$ since $w_1 w_2$ is a super word. However, positional strategies in this game produce either $(w_1 u_1)^\omega$ or $(w_2 u_2)^\omega$, both losing. This contradicts the half-positionality of $\mathcal{L}(\mathcal{A})$. ◁

◀



✦ **Figure 32.** A game $\mathcal{G}$ in which Eve can win by forming the super word $w_1 w_2$ infinitely often, but in which she cannot win using a positional strategy.

Half-positionality for objectives of parity index $[0, 1]$

We now state a characterisation of half-positionality for all objectives of parity index $[0, 1]$, without assuming prefix-independence.

Proposition V.34 (Half-positionality of Büchi languages)**.**

Let $W \subseteq \Sigma^\omega$ be a language of parity index at most $[0, 1]$. Then, W is half-positional if and only if:

- $\mathsf{Res}(W)$ is totally ordered,
- W is progress consistent, and
- W can be recognised by a Büchi automaton on top of the automaton of residuals.

Example V.35 (Half-positional objective of parity index $[0, 1]$)**.**

Over the alphabet $\Sigma = \{a, b\}$, let

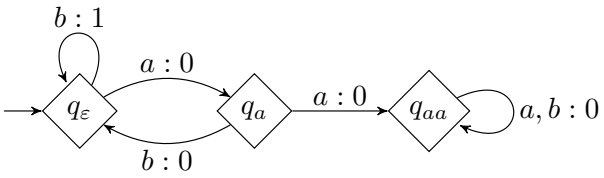
$$W = \mathsf{Buchi}(a) \cup \mathsf{Reach}(aa),$$

that is, a word $w \in \Sigma^\omega$ belongs to W if either it contains letter a infinitely often, or it contains the factor aa at some point. This objective has three different residuals,

$$[\varepsilon] < [a] < [aa] = \Sigma^\omega.$$

Figure 33 depicts a deterministic Büchi automaton defined on top of the residual automaton of W. It is easy to verify that this objective is progress consistent, so by Proposition V.34, it is a positional objective.

The half-positionality of this objective cannot be shown by applying existing half-positionality criteria [Kop08, Bia+11], nor by applying known characterisations of bipositionality [GZ05], as it is simply not bipositional.



✦ **Figure 33.** Büchi automaton recognising the objective $W = \mathsf{Buchi}(a) \cup \mathsf{Reach}(aa)$.

As earlier, we focus on explaining the necessity of the conditions from Proposition V.34, and we omit a proof of sufficiency in this warm-up. We already know that the two first conditions are necessary (Lemmas V.21 and V.26). We now present the techniques used to obtain the necessity of the third condition.

Uniformity of 0-transitions for half-positional objectives

Our objective is to derive the following result.

Lemma V.36 (Uniform behaviour of 0-transitions)**.**

Let $W \subseteq \Sigma^\omega$ be a half-positional language of parity index at most $[0, 1]$. There is a deterministic Büchi automaton $\mathcal{A}$ recognising W such that for every pair $q \sim_{\mathcal{A}} q'$ of equivalent states and for every letter a, transition $q \xrightarrow{a}$ produces priority 0 if and only if transition $q' \xrightarrow{a}$ produces priority 0.

Necessity of the conditions from Proposition V.34 easily follows. In fact, what the previous lemma tells us is that we can take the quotient automaton $\mathcal{A}/_{\sim_{\mathcal{A}}}$ and assign priorities to its transitions consistently.

The techniques we now introduce for proving Lemma V.36 will be extended in our main induction (Section V.4.2). The idea is to reduce to the prefix-independent case, captured by Proposition V.32. For this, we associate a prefix-independent language, over an ad-hoc alphabet, to each residual of the objective under consideration.

For the rest of this part of the section, we fix a half-positional objective W recognised by a deterministic Büchi automaton $\mathcal{A}$.

Localising to a residual. For each residual class $[u]$ of W, we define the *local alphabet at* $[u]$ as:

$$\Sigma_{[u]} = \{w \in \Sigma^+ \mid [uw] = [u] \text{ and for any proper prefix } w' \text{ of } w, [uw'] \neq [u]\}.$$

Note that, if it is non-empty, $\Sigma_{[u]}$ is a prefix code, and therefore it is a proper alphabet. Note that in general $\Sigma_{[u]}$ may be infinite, however this is completely harmless in this context, and we will freely allow ourselves to talk about automata over infinite alphabets.[4]

Also, $\Sigma_{[u]}$ is possibly empty; in the following definitions we assume that this is not the case.

Seeing words in $\Sigma_{[u]}^\omega$ as words in Σ^ω, define the *localisation of W to $[u]$* to be the objective

$$W_{[u]} = \{w \in \Sigma_{[u]}^\omega \mid uw \in W\}.$$

Observe that $W_{[u]}$ is prefix-independent. Moreover it is half-positional: any $W_{[u]}$-game in which Eve could not play optimally using positional strategies would provide a counterexample for the half-positionality of W.

For a state q of a Büchi automaton $\mathcal{A}$ recognising W, define $\Sigma_{[q]}$ and $W_{[q]}$ in the natural way: $\Sigma_{[q]} = \Sigma_{[u]}$ and $W_{[q]} = W_{[u]}$ for u a word that reaches q from the initial state. Observe that a word $w \in \Sigma^*$ belongs to $\Sigma_{[q]}^*$ if and only if it connects states in the class $[q]$. Elements in $\Sigma_{[q]}$ are those that do not pass twice through this class. We remark that $\Sigma_{[q]} \neq \emptyset$ if and only if q is a recurrent state (it belongs to some non-trivial SCC).

Let q be a recurrent state. The *local automaton of the residual* $[q]$ is the Büchi automaton $\mathcal{A}_{[q]}$ defined as:

- The set of states is $[q]$.
- The initial state is arbitrary.
- For $w \in \Sigma_{[q]}$, $q \xrightarrow{w:x} q'$ if $q \overset{w:x}{\rightsquigarrow} q'$ in $\mathcal{A}$.

The language of $\mathcal{A}_{[q]}$ is $W_{[u]}$, thus $W_{[u]}$ has parity index at most $[0,1]$. Therefore, Proposition V.32 yields that $W_{[u]}$ is a Büchi language: there exists a set $B_{[u]} \subseteq \Sigma_{[u]}$ such that $W_{[u]} = \mathsf{Buchi}_{\Sigma_{[u]}}(B_{[u]})$. We let $N_{[u]} = \Sigma_{[u]} \setminus B_{[u]}$ be the set of non-super letters, and extend these notations to states of $\mathcal{A}$ by putting $B_{[q]} = B_{[u]}$ and $N_{[q]} = N_{[u]}$ where u is any word leading from the initial state to q.

Polished automata. For a recurrent state q, we say that a residual class $[q]$ is *polished in $\mathcal{A}$* if:

1. For all $q_1, q_2 \in [q]$, there is a word $u \in N_{[q]}^*$ such that $q_1 \overset{u:1}{\rightsquigarrow} q_2$.
2. For every $q' \in [q]$ and every word $u \in N_{[q]}$, reading u from q' produces priority 1.

Stated differently, $[q]$ is polished if the restriction of $\mathcal{A}_{[q]}$ to transitions labelled with letters in $N_{[q]}$ is strongly connected and does not contain any transition with priority 0.

We say that the automaton $\mathcal{A}$ is *polished* if all its residual classes are polished.

✦ Remark V.37. *If $\mathcal{A}$ is polished and q is a transient state, then, it is the only state in its residual class: $[q] = \{q\}$. We will apply the term recurrent (resp. transient) to a class $[q]$ if q is recurrent (resp. transient). This is well defined by the previous comment.*

✦ **Lemma V.38** (Obtaining a polished automata)**.** *Any half-positional language W of parity index at most $[0,1]$ can be recognised by a polished deterministic Büchi automaton.*

Proof. Let $\mathcal{A}$ be a deterministic Büchi automaton recognising W. We will first polish the residual class $[q]$ of a fixed state q. Consider the restriction $\mathcal{A}'_{[q]}$ of $\mathcal{A}_{[q]}$ to transitions

[4]Note that in a finite automata over an infinite alphabet, there are finitely many classes of letters such that two letters from the same class admit exactly the same transitions. Hence one may easily turn such automata into automata over finite alphabets.

labelled with $N_{[q]}$, and take $S_{[q]}$ to be a final SCC of $\mathcal{A}'_{[q]}$; without loss of generality we assume that $q \in S_{[q]}$.

Now consider the automaton $\mathcal{A}'$ obtained from $\mathcal{A}$ by removing states in $[q] \setminus S_{[q]}$, and *redirecting* transitions that go to $[q]\setminus S_{[q]}$ in $\mathcal{A}$ to transitions towards q producing priority 0. Note that this transformation *preserves the residuals*: if $p_1 \xrightarrow{a} p_2$ in $\mathcal{A}$ and $p_1 \xrightarrow{a} p'_2$ in $\mathcal{A}'$, then $p_2 \sim_{\mathcal{A}} p'_2$. Also, either $|\mathcal{A}'| < |\mathcal{A}|$, or $\mathcal{A}$ is left unchanged. We now prove that it preserves the language.

✧ Claim V.38.1. *Automaton $\mathcal{A}'$ recognises the objective W.*

Proof. Let $w \in \Sigma^\omega$. Suppose first that the run over w in $\mathcal{A}'$ eventually does not take redirected transitions. Then, this run contains a suffix that is also a run in $\mathcal{A}$. As the transformation preserves the residuals, w is accepted by $\mathcal{A}'$ if and only if it is accepted by $\mathcal{A}'$.

Suppose now that the run over w in $\mathcal{A}'$ takes infinitely many redirected transitions. Such a run in $\mathcal{A}'$ is of the form

$$q_{\mathsf{init}} \overset{w_0}{\rightsquigarrow} q \overset{w_1}{\rightsquigarrow} p_1 \xrightarrow{a_1:0} q \overset{w_2}{\rightsquigarrow} p_2 \xrightarrow{a_2:0} q \overset{w_3}{\rightsquigarrow} p_3 \xrightarrow{a_3:0} \dots,$$

where for all i the transition $p_i \xrightarrow{a_i:0} q$ is a redirected one, meaning that in $\mathcal{A}$, reading a_i from p_i leads to $[q] \setminus S_{[q]}$. Note that w is accepted by $\mathcal{A}'$, so we should prove that $w \in \mathcal{L}(\mathcal{A}) = W$. Observe that, for $i \geq 1$, $w_i a_i \in \Sigma^*_{[q]}$ and reading $w_i a_i$ in $\mathcal{A}$ takes q to $[q] \setminus S_{[q]}$, and thus by definition of $S_{[q]}$, it holds that $w_i a_i \notin N^*_{[q]}$, so $w_i a_i \in N^*_{[q]} B^+_{[q]} N^*_{[q]}$. We conclude that $w_1 a_1 w_2 a_2 \cdots \in \mathsf{Buchi}_{\Sigma_{[q]}}(B_{[q]}) = q^{-1}W$ hence $w \in W$. ◁

It follows that the residual class of q in $\mathcal{A}'$ is $[q]_{\mathcal{A}'} = [q]_{\mathcal{A}} \cap Q' = S_{[q]}$. We now prove that it is polished.

✧ Claim V.38.2. *The residual class $[q]$ is polished in $\mathcal{A}'$.*

Proof. Let $q_1, q_2 \in S_{[q]}$. By definition of $S_{[q]}$ there is $w \in N_{[q]}$ such that $q_1 \overset{w}{\rightsquigarrow} q_2$ in $\mathcal{A}$. As this path avoids $[q] \setminus S_{[q]}$ in $\mathcal{A}$, it also belongs to $\mathcal{A}'$. We show that it produces exclusively priorities 1. By definition of $S_{[q]}$, there is a path $q_2 \overset{w'}{\rightsquigarrow} q_1$ with $w' \in N_{[q]}$. Therefore $(ww')^\omega$ does not belongs to $W_{[q]}$, so the path $q_1 \overset{w}{\rightsquigarrow} q_2$ cannot produce priority 0, which proves the first point in the definition of a polished class.

Now take $q' \in S_{[q]}$ and $u \in N^*_{[q]}$. Then by definition of $S_{[q]}$, reading u from q' in $\mathcal{A}$ leads back to $S_{[q]}$. By the previous argument, the minimal priority in this path is 1. Again, this path avoids $[q] \setminus S_{[q]}$ in $\mathcal{A}$, so it also belongs to $\mathcal{A}'$. ◁

Thus we have obtained an automaton $\mathcal{A}'$ for W in which the class $[q]$ is polished. Since there are finitely many residual classes, and the obtained automaton $\mathcal{A}'$ is strictly smaller than $\mathcal{A}$, we can repeat the process (normalising the automata after each iteration) until obtaining an automaton in which all the classes are polished. (We remark that we do not claim that classes $[p] \neq [q]$ that were polished in $\mathcal{A}$ will remain polished in $\mathcal{A}'$. Nevertheless, the process reaches a fixpoint in which all classes are polished.) ◀

We will later on use the following property, which is our main reason for introducing polished automata.

✦ **Lemma V.39** (Connection via losing words)**.** *Let $q \sim_{\mathcal{A}} q'$ be two different recurrent equivalent states of a polished automaton $\mathcal{A}$. Then there is a word $u \in \Sigma^+_{[q]}$ such that $q \overset{u}{\rightsquigarrow} q'$ and $u^\omega \notin q^{-1}\mathcal{L}(\mathcal{A})$.*

Proof. As the automaton $\mathcal{A}$ is polished, there is a word $u \in N^+_{[q]}$ such that $q \overset{u}{\rightsquigarrow} q'$. This word satisfies the desired requirement. ◀

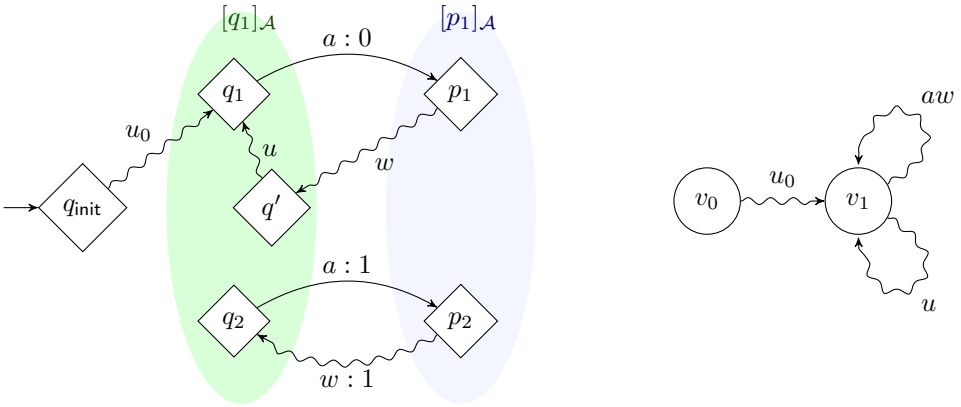
Uniform behaviour of 0-transitions. We are now ready to prove Lemma V.36. Let $\mathcal{A}$ be a polished and normalised deterministic Büchi automaton recognising W, and suppose by contradiction that there are two states $q_1 \sim_{\mathcal{A}} q_2$ and a letter $a \in \Sigma$ such that $q_1 \xrightarrow{a:0} p_1$ and $q_2 \xrightarrow{a:1} p_2$.

By normality, there is a word $w \in \Sigma^*$ such that $p_2 \overset{w:1}{\rightsquigarrow} q_2$. In particular, since $q_2 \overset{aw}{\rightsquigarrow} q_2$, the class $[q_2]$ is recurrent in $\mathcal{A}$. Note that $aw \in \Sigma^*_{[q_1]}$ and $(aw)^\omega \notin q_1^{-1}W$. Let q' be the state such that $p_1 \overset{w}{\rightsquigarrow} q'$, note that $q' \in [q_1]$. Let u_0 be such that $q_{\mathsf{init}} \overset{u_0}{\rightsquigarrow} q_1$. See Figure 34 for an illustration of the situation.

Since $(aw)^\omega \notin q_1^{-1}W$, and $q_1 \overset{aw:0}{\rightsquigarrow} q'$, it cannot be that case that $q' = q_1$. Hence, Lemma V.39 gives a word $u \in \Sigma^+_{[q]}$ such that $q' \overset{u}{\rightsquigarrow} q_1$ and $u^\omega \notin q^{-1}W$. Together, the facts that

- $(aw)^\omega \notin q^{-1}W$,
- $u^\omega \notin q^{-1}W$, and
- $(awu)^\omega \in q^{-1}W$

prove that Eve wins the game on the right of Figure 34, but not positionally.



✦ **Figure 34.** On the left, the situation in the automaton $\mathcal{A}$. We have the equivalences $q_1 \sim_{\mathcal{A}} q_2 \sim_{\mathcal{A}} q'$ and $p_1 \sim_{\mathcal{A}} p_2$. On the right, a game where Eve can win but not positionally.

Thus we proved that if W is a half-positional objective of parity index $[0, 1]$, it can be recognised by a Büchi automaton in which 0-transitions behave uniformly. By extending the technique from this section, we will show in Section V.4.2 that this property (and its generalisation to all even priorities) also holds for higher parity indices.

3.4 Objectives recognised by coBüchi automata: Total order given by safe languages

We now consider objectives of parity index $[1, 2]$, that is, those that can be recognised by deterministic coBüchi automata. Our analysis requires using history-deterministic automata and techniques from [AK22].

Most of the section is devoted to the case of prefix-independent objectives, for which we propose a characterisation of half-positionality (Proposition V.44). At the end of the section, we comment on how to extend this characterisation to any objective of parity index [1, 2] (Proposition V.52). In the spirit of this warm-up, we omit proofs of sufficiency.

Prefix-independent objectives of parity index [1, 2]

Let us start by introducing some terminology relative to coBüchi automata recognising prefix-independent languages. Our analysis requires considering automata that are not necessarily deterministic, however, we have a fine control of the non-determinism that will appear. First, all automata in this subsection will be history-deterministic. Moreover, we can suppose that they are deterministic over transitions producing priority 2 by the following result of Kuperberg et Skrzypczak [KS15] (this property is sometimes called safe determinism).

✦ **Lemma V.40** ([KS15])**.** *Every history-deterministic coBüchi automaton can be pruned in polynomial time into an equivalent one that is deterministic over* 2*-transitions.*

Safe languages and safe components. Consider a (possibly non-deterministic) coBüchi automaton $\mathcal{A}$. We define the *(<2)-safe language* of a state q (or just *safe language*) as the set of finite or infinite words such that, when read from q, priority 1 can be avoided, that is:

$$\mathsf{Safe}_{<2}(q) = \{w \in \Sigma^* \cup \Sigma^\omega \mid \text{ there is a run } q \overset{w:2}{\rightsquigarrow} \}.$$

✦ Remark V.41. *Two safe languages coincide if and only if their restrictions to finite (resp. infinite) words coincide. Indeed, an infinite word $w \in \Sigma^\omega$ belongs to $\mathsf{Safe}_{<2}(q)$ if and only all its finite prefixes do.*

We write $q \leq_2 q'$ if $\mathsf{Safe}_{<2}(q) \subseteq \mathsf{Safe}_{<2}(q')$; this defines a partial preorder on states of $\mathcal{A}$. We will sometimes use the term *safe path* to refer to paths in $\mathcal{A}$ that do not produce priority 1. The use of the notation "(< 2)-safe" will be justified by the generalisation of this notion to any parity automaton. The next lemma follows directly from the definition of safe languages.

✦ **Lemma V.42** (Monotonicity with respect to safe languages)**.** *Let $\mathcal{A}$ be a coBüchi automaton which is deterministic over* 2*-transitions, and let q, q' be two states such that $q \leq_2 q'$. Let u be a finite word in $\mathsf{Safe}_{<2}(q)$ and write $q \overset{u:2}{\rightsquigarrow} p$. There is a unique path $q' \overset{u:2}{\rightsquigarrow} p'$ and $p \leq_2 p'$.*

A *(<2)-safe component* (or just *safe component*) of $\mathcal{A}$ is a strongly connected component of the subautomaton obtained by removing from $\mathcal{A}$ all transitions labelled 1. By Lemma V.40, we can suppose that these subautomata are deterministic. Also, note that if $\mathcal{A}$ is in normal form, transitions between different safe components produce priority 1, that is, states connected by a safe path are in the same safe component.

✦ Remark V.43. *A run of a coBüchi automaton is accepting if and only if it eventually remains in a safe component. (We remark that our definition of SCC also includes the edges of the subgraph.)*

Statement of the characterisation of half-positionality. We are now ready to state a characterisation of positionality of prefix-independent objectives of parity index

at most $[1, 2]$.

Proposition V.44 (Half-positionality for prefix-independent coBüchi objectives)**.**

A prefix-independent objective of parity index at most $[1, 2]$ is half-positional if and only if it can be recognised by a deterministic coBüchi automaton satisfying that within each safe component, states are totally ordered by inclusion of safe languages.

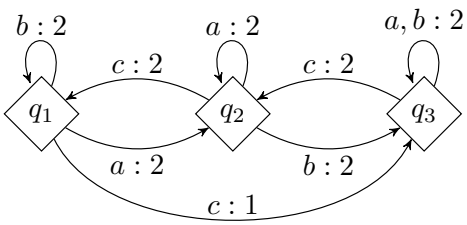
Before going on with the proof, we discuss two examples.

Example V.45.

In Figure 35 we represent a coBüchi automaton over $\Sigma = \{a, b, c\}$ recognising the following objective:

$$W = \mathsf{coBuchi}(c(a^*cb^*)^+c).$$

This is an example of an objective that is not concave, as by shuffling words $a(ccaa)^\omega \notin W$ and $(bbcc)^\omega \notin W$ we can obtain $(abcc)^\omega \in W$. However, we show that it satisfies the hypothesis of Proposition V.44, so it is half-positional. This automaton has a single safe component. The inclusions of the safe languages follows from the fact that the transitions are monotone: for every letter $\alpha \in \Sigma$, if $q_i \xrightarrow{\alpha:2} q_j$ and $i \leq i'$, then $q_{i'} \xrightarrow{\alpha:2} q_{j'}$ with $j \leq j'$.



✦ **Figure 35.** Deterministic coBüchi automaton recognising objective W from Example V.45.

Example V.46.

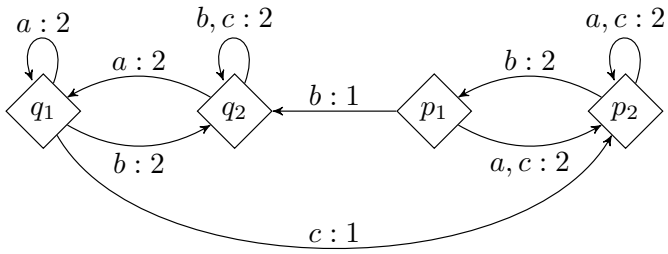
Let $\Sigma = \{a, b, c\}$ and W be the prefix-independent objective of words that eventually do not contain infinitely often both the factor ac and the factor bb, that is:

$$W = \mathsf{coBuchi}(ac) \cup \mathsf{coBuchi}(bb).$$

We give a coBüchi automaton recognising W and satisfying the hypothesis of Proposition V.44 in Figure 36. This automaton has two safe components: $S_1 = \{q_1, q_2\}$ and $S_2 = \{p_1, p_2\}$. The states of each component are totally ordered by inclusion of safe languages, as we have $q_1 <_2 q_2$ and $p_1 <_2 p_2$. Therefore, W is half-positional.

At the level of intuition, starting from a deterministic coBüchi automaton $\mathcal{A}$ recognising a half-positional objective W, our proof of the necessity in Proposition V.44 proceeds as follows:

- We turn $\mathcal{A}$ into a history-deterministic safe centralised automaton (see definition below) using the minimisation technique of Abu Radi and Kupferman [AK22].



✦ **Figure 36.** Deterministic coBüchi automaton recognising objective W from Example V.46. This automaton has two safe components: $S_1 = \{q_1, q_2\}$ and $S_2 = \{p_1, p_2\}$, and the states of each of them are totally ordered by inclusion of safe languages.

- Using half-positionality, we prove that $\leq_2$ defines a total order on each safe component.
- Exploiting the total order, we are able to re-determinise $\mathcal{A}$.

Safe centralisation and safe minimality. Let $\mathcal{A}$ be a (possibly non-deterministic) coBüchi automaton with only one residual class. We say that $\mathcal{A}$ is *safe centralised* if, for every pair of states q_1, q_2, if $q_1 \leq_2 q_2$, then q_1 and q_2 are in the same safe component. We say that $\mathcal{A}$ is *safe minimal* if there are no two different states with the same safe language.

✦ **Lemma V.47** ([AK22])**.** *Any prefix-independent language of parity index at most* $[1, 2]$ *can be recognised by a history-deterministic coBüchi automaton that is safe centralised and safe minimal.*

Let us present a proof of Lemma V.47. We say that an automaton is 1-*saturated* if for all pair of states q, q', the transition $q \xrightarrow{a:1} q'$ appears in $\mathcal{A}$. The 1-*saturation* of a coBüchi automaton is the automaton obtained by simply adding all possible transitions of the form $q \xrightarrow{a:1} q'$; note that this transformation preserves determinism over 2-transitions.

✦ **Lemma V.48.** *Let $\mathcal{A}$ be a coBüchi automaton recognising a prefix-independent language W. Then, its 1-saturation $\mathcal{A}'$ also recognises W. Moreover, if $\mathcal{A}$ is history-deterministic and deterministic over 2-transitions, then so is $\mathcal{A}'$.*

Proof. We first show $\mathcal{L}(\mathcal{A}') \subseteq W$. An accepting run over a word w in $\mathcal{A}'$ eventually only reads transitions with priority 2, so it eventually coincides with a run in $\mathcal{A}$. We conclude by prefix-independence. The fact that $W \subseteq \mathcal{L}(\mathcal{A}')$ and history-determinism are clear: one can use the same resolver in $\mathcal{A}'$ as in $\mathcal{A}$. ◀

Proof of Lemma V.47. Let $\mathcal{A}$ be a normalised, 1-saturated, history-deterministic, deterministic over 2-transitions automaton recognising W, obtained by Lemma V.48. We say that a safe component S is *redundant* if there is $q \in S$ and $q' \notin S$ such that $q \leq_2 q'$.

✧ Claim V.47.1. *Let S be redundant and consider the automaton $\mathcal{A}'$ obtained from $\mathcal{A}$ by deleting S. Then $\mathcal{A}'$ is history-deterministic and recognises W.*

Proof. Clearly $\mathcal{L}(\mathcal{A}') \subseteq W$. We will describe a sound resolver proving that $\mathcal{L}(\mathcal{A}') = W$ and that $\mathcal{A}'$ is history-deterministic. Let $q \in S$ and $q' \notin S$ such that $q \leq_2 q'$. For each $p \in S$, pick $u \in \Sigma^*$ such that $q \overset{u:2}{\rightsquigarrow} p$, and let $f(p)$ be such that $q' \overset{u:2}{\rightsquigarrow} f(p)$; this is well defined since $q \leq_2 q'$. Note that we have $p \leq_2 f(p)$ by Lemma V.42. By

normality of $\mathcal{A}$, there is a returning path $f(p) \overset{w:2}{\rightsquigarrow} q'$ and thus $f(p)$ is in the same safe component as q', so it does not belong to S. We extend f to all of Q by setting it to be the identity over $Q \setminus S$.

Take a sound resolver (q_0, r) in $\mathcal{A}$, let $w \in \Sigma^\omega$, and write

$$\rho = q_0 \xrightarrow{w_0} q_1 \xrightarrow{w_1} \dots$$

for the run in $\mathcal{A}$ induced by r over w. We will build a resolver (q'_0, r') in $\mathcal{A}'$ satisfying the property that the run induced over w, $\rho' = q'_0 \xrightarrow{w_0} q'_1 \xrightarrow{w_1} \dots$ is such that for each i, $q_i \leq_2 q'_i$. We let $q'_0 = f(q_0)$, and assume ρ' constructed up to q'_i, and $q_i \leq_2 q'_i$. If there is a state $q'_{i+1} \notin S$ such that $q'_i \xrightarrow{w_i:2} q'_{i+1}$ then we take this one, which satisfies $q_{i+1} \leq_2 q'_{i+1}$ by Lemma V.42. Otherwise, take the transition $q'_i \xrightarrow{w_i:1} f(q_{i+1})$. In particular, if $w \in \mathsf{Safe}_{<2}(q_i)$, the induced run from q_i does not produce priority 1.

We now show that if ρ is accepting, then ρ' is an accepting run too: if ρ is accepting, for some i the suffix $w_i w_{i+1} \dots \in \mathsf{Safe}_{<2}(q_i) \subseteq \mathsf{Safe}_{<2}(q'_i)$. Therefore, the run $q'_i \xrightarrow{w_i:2} q'_{i+1} \xrightarrow{w_{i+1}:2} \dots$ in $\mathcal{A}'$ is safe, and ρ' is accepting. ◁

Using Claim V.47.1, we successively remove redundant safe components until obtaining a safe centralised automaton.

Finally, to obtain a safe minimal automaton it suffices to merge states with the same safe language. That is, we define a 1-saturated automaton that has for states the classes $[q]_2$ of states of $\mathcal{A}$, and transitions $[q]_2 \xrightarrow{a:2} [p]_2$ whenever for some (or equivalently, for all) state $q' \in [q]_2$ there is transition $q' \xrightarrow{a:2} p'$ in $\mathcal{A}$, with $p' \in [p]_2$. It is not difficult to check that the obtained automaton recognises W, is history-deterministic and remains safe centralised. ◀

Total order in each safe component. The intuitive idea on why having states of a same safe component totally ordered by $\leq_2$ is necessary for positionality is the same than in the case of closed objectives (Lemma V.21): if q and q' are incomparable, there are two words w, w' that produce priority 1 from one state but not from the other. In a game, if Eve has not kept track of where we are in the automaton, she will not know what is the best option between w and w'. However, an issue arises when turning this idea into an actual proof: one needs to build two full differentiating runs from q and q'; producing priority 1 just once does not suffice. Safe centrality will come in handy for this purpose.

By definition, if $q \not\leq_2 q'$, there is a word which produces priority 1 when read from q', and stays in the corresponding safe component when read from q. The following lemma exploits safe centrality to extend those runs to synchronise them in the same state, while the run starting from q remains safe. For the purpose of the warm up, we only prove it assuming $\mathcal{A}$ is deterministic; extending it to the history-deterministic case requires some additional technicalities that will be dealt with in Section V.4.2.

✦ **Lemma V.49** (Synchronisation of separating runs)**.** *Let $\mathcal{A}$ be a normalised, safe centralised and safe minimal deterministic coBüchi automaton with a single residual class. Let q and q' be two states such that $q \not\leq_2 q'$ and p be any state in the safe component of q. There is a word $w \in \Sigma^*$ such that $q \overset{w:2}{\rightsquigarrow} p$ and $q' \overset{w:1}{\rightsquigarrow} p$.*

Proof. Since $\mathsf{Safe}_{<2}(q) \not\subseteq \mathsf{Safe}_{<2}(q')$, there is a word $w_1 \in \Sigma^*$ such that $q \overset{w_1:2}{\rightsquigarrow} q_1$ and $q' \overset{w_1:1}{\rightsquigarrow} q'_1$. By normality, note that q_1 is in the same safe component as q. If we have again

that $q_1 \not\leq_2 q'_1$, we can find a word w_2 with the same properties. While the non-inclusion of safe languages is satisfied, repeating the argument yields two runs:

$$q \xrightarrow{w_1:2} q_1 \xrightarrow{w_2:2} q_2 \xrightarrow{w_3:2} \dots$$
$$q' \xrightarrow{w_1:1} q'_1 \xrightarrow{w_2:1} q'_2 \xrightarrow{w_3:1} \dots$$

We claim that the process should stop after finite time, meaning that for some i, we have $q_i \leq_2 q'_i$. Otherwise, we would obtain two infinite runs over $w_1w_2w_3 \cdots \in \Sigma^\omega$, one of them accepting and the other rejecting, contradicting the fact that $\mathcal{A}$ has a single residual class.

Let i be the step in which $q_i \leq_2 q'_i$. First, we note that both states are in the safe component of q: state q_i is in there because there is a path $q \xrightarrow{2} q_i$, and by safe centrality, q'_i must also be in the same safe component. Let $q_{\max}$ be a state in this safe component maximal amongst states such that $q'_i \leq_2 q_{\max}$. Let $u \in \Sigma^*$ be a word labelling a path $q_i \xrightarrow{u:2} q_{\max}$ (which exists by definition of safe components). By Lemma V.42, maximality of $q_{\max}$, and safe minimality, we also have $q'_i \xrightarrow{u:2} q_{\max}$. Finally, if suffices to take a word $u' \in \Sigma^*$ labelling a path $q_{\max} \xrightarrow{u':2} p$ and define $w = w_1w_2 \dots w_iuu'$. ◀

We may derive the sought total orders.

✦ **Lemma V.50** (Total order in each safe component)**.** *Let $\mathcal{A}$ be a deterministic coBüchi automaton recognising a prefix-independent half-positional objective W. Suppose that $\mathcal{A}$ is safe centralised and safe minimal. Let q and q' be two different states in the same safe component. Then, either $q <_2 q'$ or $q' <_2 q$.*

Proof. By safe minimality, $q \not<_2 q'$ implies $q \not\leq_2 q'$. Suppose by contradiction that $q \not\leq_2 q'$ and $q' \not\leq_2 q$. Let p be a state in this safe component, and let $u, u' \in \Sigma^*$ be such that $p \xrightarrow{u:2} q$ and $p \xrightarrow{u':2} q'$. By Lemma V.49, there are $w, w' \in \Sigma^\omega$ such that:

$$q \xrightarrow{w:2} p, \quad q \xrightarrow{w':1} p,$$
$$q' \xrightarrow{w:1} p, \quad q' \xrightarrow{w':2} p.$$

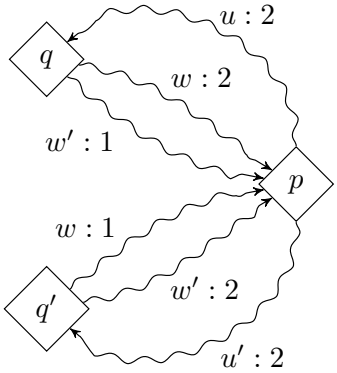
The situation is depicted in Figure 37. We obtain that:

- $(w'u)^\omega \notin W$,
- $(wu')^\omega \notin W$,
- $(wu'w'u)^\omega \in W$.

Thus consider the game in which Eve controls a vertex with two self loops labelled $w'u$ and wu'; she can win by alternating both loops, but fails to win positionally. ◀

Determinisation. We have stated last two lemmas of the previous paragraph for deterministic automata, in order to simplify the presentation. However, safe centralisation only yields history-deterministic automaton. We now explain how to use the total order from Lemma V.50 to determinise HD coBüchi automata recognising half-positional languages.

Let $\mathcal{A}$ be a normalised, history-deterministic and deterministic over 2-transitions coBüchi automaton recognising a prefix-independent half-positional language. Order $\leq_2$



✦ **Figure 37.** Situation occurring in the proof Lemma V.50.

is total over each safe component, by Lemma V.50. We show how to rearrange the 1-transitions in order to define an equivalent deterministic automaton $\mathcal{A}'$ with the same structure of safe components.

Let $\{S_1, S_2, \dots, S_k\}$ be the safe components of $\mathcal{A}$, enumerated in an arbitrary order. Recall that if a word $w \in \Sigma^\omega$ is accepted by $\mathcal{A}$, there is a run over w that eventually stays in one of the components S_i (Remark V.43). The main idea is that, when reading a word w, we can resolve the non-determinism by trying each safe component in a round-robin fashion. If for a state q and for a letter $a \in \Sigma$ there is a (unique) transition $q \xrightarrow{a:2}$, we keep it as the only a-transition from q. If there is no transition $q \xrightarrow{a:2}$, and q belongs to S_i, we define a transition $q \xrightarrow{a:1} q'$ towards some q' in S_{i+1}. The total order in S_{i+1} identifies a state in S_{i+1} which is the best to go to: we define $q \xrightarrow{a:1} q_{i+1}^{\max}$, where $q_{i+1}^{\max}$ is the unique maximal state of S_{i+1} for the total order $\leq_2$. This defines a deterministic automaton $\mathcal{A}$'.

We prove that $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$. Clearly $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$, as $\mathcal{A}'$ is a subautomaton of the 1-saturation of $\mathcal{A}$. To show the other inclusion, let $w \in \mathcal{L}(\mathcal{A})$. There is a run over w in $\mathcal{A}$ that eventually remains in a safe component, without loss of generality, we assume that it is S_1. Let $w' = w'_1 w'_2 \dots$ be a suffix of w labelling a run in S_1: $q^1 \xrightarrow{w'_1} q^2 \xrightarrow{w'_2} \dots$. Consider a run over w' in $\mathcal{A}'$. If this run never visits S_1, it must be because it eventually remains in some safe component, so w' is accepted by $\mathcal{A}'$ in that case. If the run eventually arrives to S_1 (at the i$^{\text{th}}$ step), it will arrive to the maximal element $q_1^{\max}$. At this point, the run in $\mathcal{A}$ is in $q^i \leq_2 q_1^{\max}$. Since $w'_i w'_{i+1} \dots \in \mathsf{Safe}_{<2}(q^i) \subseteq \mathsf{Safe}_{<2}(q_1^{\max})$, the run over this suffix is safe in $\mathcal{A}'$, and word w is accepted by $\mathcal{A}'$.

Example V.51.

The automaton from Figure 36 has the shape we have described: transitions producing priority 1 cycle between the two safe components, and they go to the maximal state of the other component ($p_1 \xrightarrow{b:1} q_2$ and $q_1 \xrightarrow{c:1} p_2$).

Generalisation to non prefix-independent coBüchi languages

To remove the prefix-independence assumption, we work with the localisation to residuals of the objectives, as defined in Section V.3.3. If W is a half-positional objective, for each residual $[u]$ the objective $W_{[u]}$ is half-positional. In turns out that this property, together with the hypothesis over residuals that were already necessary for open objectives,

provides a characterisation.

Proposition V.52.

Let $W \subseteq \Sigma^\omega$ be a language of parity index at most $[1, 2]$. Then, W is half-positional if and only if:

- $\mathsf{Res}(W)$ is totally ordered,
- W is progress consistent, and
- for all residual class $[u]$, objective $W_{[u]}$ is half-positional.

This result is not fully satisfying (and hard to prove directly), as it relies on the half-positionality of languages $W_{[u]}$. As these objectives are prefix-independent, we have a characterisation for their half-positionality (Proposition V.44), and we can put them together to obtain a statement using exclusively structural properties of parity automata. The statement we obtain uses a decomposition of parity automata in three layers:

- States are totally preordered by their residual class (layer 0).
- Within each residual class, states are grouped into safe components (layer 1).
- Within each safe component, states are totally ordered by inclusion of the safe languages (layer 2).

This decomposition foreshadows the definition of structured signature automata that we will use in Section V.4.2 to derive a characterisation of half-positionality for all ω-regular languages.

Proposition V.53.

Let $W \subseteq \Sigma^\omega$ be a language of parity index at most $[1, 2]$. Then, W is half-positional if and only if:

- $\mathsf{Res}(W)$ is totally ordered,
- W is progress consistent, and
- W can be recognised by a deterministic coBüchi automaton $\mathcal{A}$ such that, for all residual class $[q]$, the local automaton of the residual $\mathcal{A}_{[q]}$ satisfies that its safe components are totally ordered by inclusion of safe languages.

We do not include a proof of this proposition; it is a special case of Theorem V.1.

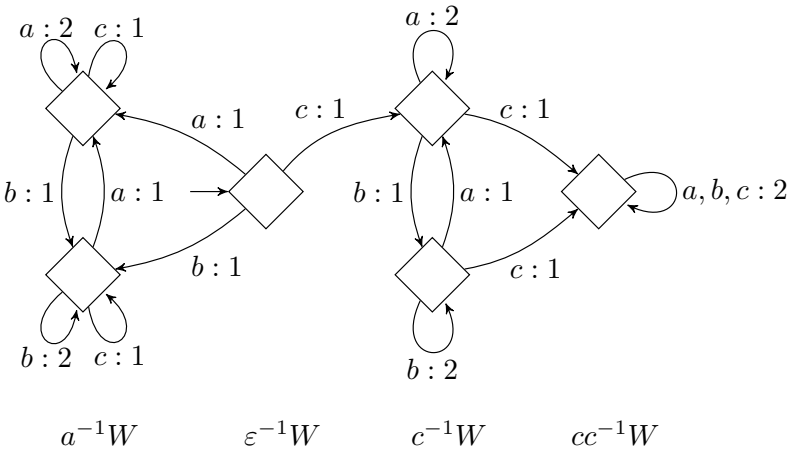
Example V.54.

We consider the following objective of parity index $[1, 2]$ over the alphabet $\Sigma = \{a, b, c\}$:

$$W = \Sigma^* a^\omega \cup \Sigma^* b^\omega \cup c\Sigma^* c\Sigma^\omega.$$

This objective was studied in [Bia+11, Lemma 12] to show that there are half-positional objectives that are not concave, nor bipositional (objectives from Examples V.35 and V.45 also have this property); their proof of half-positionality is quite involved.

A coBüchi automaton recognising objective W is depicted in Figure 38. Its residuals are totally ordered, and it is easy to check that it is progress consistent. Moreover, all its safe components are trivial. Therefore, Proposition V.53 implies that it is half-positional.

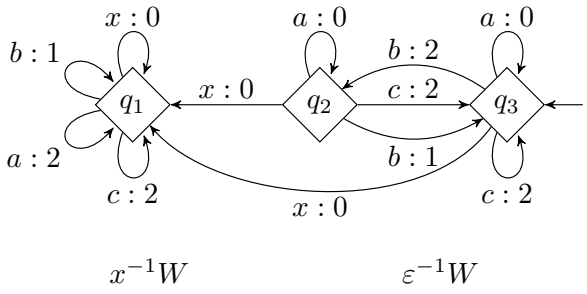


✦ **Figure 38.** Automaton recognising objective W from Example V.54.

3.5 Towards objectives of higher parity index: An example

In Figure 39, we show a deterministic parity automaton $\mathcal{A}$ over the alphabet $\Sigma = \{a, b, c, x\}$, recognising an objective W. One may describe the objective W as follows: if a word $w \in \Sigma^\omega$ does not contain the letter x, then $w \in W$ if, either it contains the letter a infinitely often, or it does not contain the factor bb infinitely often. If w contains the letter x, then its membership in W is determined by the parity condition associated to the mapping $x \mapsto 0$, $b \mapsto 1$, $a, c \mapsto 2$.

As this automaton is neither a Büchi or a coBüchi one (W has parity index $[0, 2]$), none of the characterisations of this section applies to it. However, we can combine the techniques presented above to equip $\mathcal{A}$ with a "nicely behaved" total order. In next section, we will see that this can be formalised as the fact that $\mathcal{A}$ is a deterministic fully progress consistent signature automaton, so by Theorem V.1, W is half-positional.



✦ **Figure 39.** Automaton recognising objective W.

Objective W has two residuals: $\varepsilon^{-1}W$ and $x^{-1}W$. It is easy to check that $x^{-1}W \subseteq \varepsilon^{-1}W$, and, since there is no transition $[x] \to [\varepsilon]$, it is trivially progress consistent. The

states associated to $[x]$ and $[\varepsilon]$ are respectively $\{q_1\}$ and $\{q_2, q_3\}$. The subautomaton induced by $\{q_1\}$ recognises a parity objective which is half-positional. Let us focus on the subautomaton induced by $\{q_2, q_3\}$, that, in this example, coincides with $\mathcal{A}_{[\varepsilon]}$. We observe that it satisfies the hypothesis of Lemma V.36: transitions of priority 0 act uniformly within $\mathcal{A}_{[\varepsilon]}$; indeed, these transitions are those reading letter a. Consider the restriction of $\mathcal{A}_{[\varepsilon]}$ to letters $\{b, c\}$. The obtained automaton $\mathcal{A}'_{[\varepsilon]}$ is a coBüchi automaton, satisfying the hypothesis of Lemma V.50: the states of the safe component of $\mathcal{A}'_{[\varepsilon]}$ are totally ordered by inclusion of safe languages, as $q_2 <_2 q_3$. In this way, we obtain a decomposition such that:

- Residuals are total ordered by inclusion, and are progress consistent.
- 0-transitions act uniformly within the states of each residual class.
- The safe components of the coBüchi automata obtained as the restriction of each class to transitions with priority ≥ 1 are totally ordered by inclusion of safe languages.

Generalising such decomposition to any parity automaton will be the central point of the next section.

4 Obtaining the structural characterisation of half-positionality

We now move on to the proof of Theorem V.1. The first step is to identify the common structural properties of deterministic parity automata recognising half-positional objectives. In Section V.4.1, we define a class of parity automata, which we call signature automata, that underscores this structure. We also introduce full progress consistency, a further necessary condition for automata recognising half-positional languages. To obtain the implication from (1) to (2) in Theorem V.1 we need then to show how to obtain a fully progress consistent signature automaton for a half-positional ω-regular objective. This is the most technical part of the proof, and it is the object of Section V.4.2. We then proceed to defining ε-complete automata and closing the cycle of implications in Section V.4.3.

4.1 Signature automata and full progress consistency

Before finally moving on to the crucial definition of signature automata, we need more precise concepts of congruences which we introduce now.

4.1.1 Priority-faithful congruences and quotient automata

Priority-faithful congruences. We recall that a congruence in an automaton allows us to define a quotient $\mathcal{A}/_\sim$, which is a deterministic automaton structure. However, in general, no acceptance condition can be defined on top of $\mathcal{A}/_\sim$ in a sensible way, as the congruence does not have to be compatible with the output colours of the automaton. We now strengthen the definition of congruence for parity automata so that it will be possible to define an approximation of a correct parity condition on top of the quotient automaton.

Definition V.55 ($[0,x]$-faithful congruence)**.**

Let $\mathcal{A}$ be a parity automaton and let $\sim$ be a congruence over its set of states Q. We say that $\sim$ is $[0,x]$-*faithful* if:

- for each $0 \leq y \leq x$, y-transitions are uniform over $\sim$-classes and relation $\sim$ is a congruence for y-transitions, and
- relation $\sim$ is a congruence for $(>x)$-transitions.

Stated differently, a relation $\sim$ is a $[0,x]$-faithful congruence if, whenever there is a transition $q \xrightarrow{a:y} p$ with priority $y \leq x$, then for every $q' \sim q$ a transition $q' \xrightarrow{a} p'$ produces priority y and goes to a state $p' \sim p$. If there is a transition $q \xrightarrow{a:y} p$ producing priority $y > x$, we only require that transitions $q' \xrightarrow{a:>x} p'$ produce priorities $> x$ and go to $p' \sim p$ (but the exact priority produced may differ). (We remark that, although it is not explicitly imposed, $(>x)$-transitions are uniform over $\sim$-classes by the first property. Also, we recall that we assume that automata are complete.)

✦ Remark V.56. *A $[0,x]$-faithful congruence is $[0,y]$-faithful for any $y \leq x$.*

$(\leq x)$-quotient automata. What the definition of $[0,x]$-faithful congruence tells us is that transitions producing priorities $y \leq x$ are well defined in the quotient automaton $\mathcal{A}/\!\sim$, in the sense that we can associate a priority y to these transitions reliably. For transitions producing priorities $> x$, on the other hand, we only obtain the information that the priority produced from any state of the class will be large, but we lose some precision.

Definition V.57 ($(\leq x)$-quotient automata)**.**

If $\sim$ is a $[0,x]$-faithful congruence, we can define the $(\leq x)$-*quotient of $\mathcal{A}$ by $\sim$* to be the parity automaton $\mathcal{A}/_{\leq x}^{\sim}$ given by:

- Its set of states are the $\sim$-classes of $\mathcal{A}$.
- Initial states are those of the form $[q_{\mathsf{init}}]$, where q_{init} is an initial state of $\mathcal{A}$.
- For $y \leq x$, there is a transition $[q] \xrightarrow{a:y} [p]$ if $\mathcal{A}$ has a transition $q' \xrightarrow{a:y} p'$ with $q' \in [q]$, $p' \in [p]$.
- There is a transition $[q] \xrightarrow{a:x'} [p]$ if $\mathcal{A}$ has a transition $q' \xrightarrow{a:>x} p'$ with $q' \in [q]$, $p' \in [p]$; where $x' = x+1$ if x even, and $x' = x$ if x odd.

The automaton $\mathcal{A}/_{\leq x}^{\sim}$ is defined so transitions coming from those producing a priority $> x$ in $\mathcal{A}$ are assigned the least important odd priority. This guarantees that the projection of runs that eventually only produce priorities $> x$ in $\mathcal{A}$ are rejecting in $\mathcal{A}/_{\leq x}^{\sim}$. The next lemma refines this comment.

✦ **Lemma V.58.** *Let $\mathcal{A}$ be a parity automaton and let $\sim$ be a $[0,x]$-faithful congruence over it. The $(\leq x)$-quotient of $\mathcal{A}$ by $\sim$ recognises the language:*

$$\mathcal{L}(\mathcal{A}/_{\leq x}^{\sim}) = \{w \in \Sigma^\omega \mid w \text{ is accepted with an even priority } y \leq x \text{ in } \mathcal{A}\}.$$

Moreover, if $\mathcal{A}$ is in normal form, then so is $\mathcal{A}/_{\leq x}^{\sim}$.

Proof. A run ρ in $\mathcal{A}$ produces a priority $y \leq x$ infinitely often if and only its projection in $\mathcal{A}/_{\sim_{\leq x}}$ produces priority y infinitely often, which gives the equality of the languages. The fact that $\mathcal{A}/_{\sim_{\leq x}}$ inherits the normal form is a direct application of Theorem II.14. ◀

4.1.2 Signature automata

We give the definition of signature automata, at the core of our characterisation.

We say that a sequence of total preorders $\leq_0, \leq_1, \leq_2, \ldots, \leq_k$ over Q is a *collection of nested total preorders* if $\leq_i$ refines $\leq_{i-1}$, for $i > 1$. We note that, in that case, the induced equivalence relation $\sim_i$ also refines $\sim_{i-1}$.

Definition V.59 (Signature automaton)**.**

Let $d \in \mathbb{N}$ be a priority. A *d-signature automaton* is a semantically deterministic parity automaton $\mathcal{A}$ together with a collection of nested total preorders $\leq_0, \leq_1, \leq_2, \ldots, \leq_d$ over Q such that:[5]

I) Refinements of residual inclusion. Preorder $\leq_0$ refines the preorder $\leq_{\mathcal{A}}$ given by the inclusion of residuals.

II) Faithful partitions at even layers. For $0 \leq x \leq d$, x even, the equivalence relation $\sim_x$ is a $[0, x]$-faithful congruence.

III) $(<x)$-*safe separation* at odd layers. For $2 \leq x \leq d$, x even, and $q \sim_{x-2} q'$:

$$q <_{x-1} q' \implies \text{there is no path } q \overset{w:\geq x}{\rightsquigarrow} q'.$$

IV) *Local monotonicity* of $(\geq x)$-transitions. For an even priority $x \leq d$, transitions using priorities $\geq x$ are monotone for $\leq_x$ over each $\sim_{x-1}$ class. That is, for $q \sim_{x-1} q'$, if $q \leq_x q'$:

$$q \xrightarrow{a:\geq x} p \implies q' \xrightarrow{a:\geq x} p',\ p \sim_{x-1} p' \text{ and } p \leq_x p', \text{ for all } a\text{-transitions from } q'.$$

We say that $\mathcal{A}$ is a *signature automaton* if it is a d-signature automaton, for d the maximal priority appearing in $\mathcal{A}$.

We note that even and odd preorders play a completely different role in the previous definition. In fact, the only purpose of odd preorders is to delimit the areas in which the local monotonicity property will apply. Item (III) constrains $\sim_{x-1}$-classes to be "sufficiently large".

✦ Remark V.60. *We note that, by Item (IV), for x even the equivalence relations $\sim_{x-1}$ is a congruence for $\geq x$-transitions. However, the restrictions on these odd preorders are much weaker, as we do not impose them to be faithful.*

[5] For notational convenience, we let $\sim_{-2}$ be the trivial relation over $\mathcal{A}$ throughout this definition. That is, $q \sim_{-2} p$ for all pairs of states in Q.

Example V.61.

Consider the automaton $\mathcal{A}$ from Figure 39 from the warm-up. This automaton has 3 states, q_1, q_2, and q_3. It can be equipped with the structure of a signature automaton as follows:

- Preorder $\leq_0$ is given by the inclusion of residuals: $q_1 <_0 q_2, q_3$, and $q_2 \sim_0 q_3$.
- Preorder $\leq_1$ coincides with preorder $\leq_0$.
- Preorder $\leq_2$ is a total order: $q_1 <_2 q_2 <_2 q_3$.

If $\mathcal{A}$ is a signature automaton whose maximal even priority is d, we can see the quotient automata $\mathcal{A}/_{\leq_0}^{\sim_0}, \mathcal{A}/_{\leq_2}^{\sim_2} \dots, \mathcal{A}/_{\leq_d}^{\sim_d}$ as a sequence of automata that approximates the original one. Indeed, the i^{th} automaton of this sequence coincides with $\mathcal{A}$ over the words that are accepted or rejected with priority $\leq i$. In particular, $\mathcal{A}/_{\leq_d}^{\sim_d}$ is equivalent to $\mathcal{A}$. Although we allow non-determinism in our definition, a signature automata is almost deterministic, as each one of these quotient automata is deterministic. We further formalise this remark now.

We say that a signature automaton $\mathcal{A}$ is *reduced* if its $\sim_d$-classes are singletons, where d is the maximal even priority appearing in $\mathcal{A}$.

This notion of reduction can be seen as a generalisation of the notion of safe minimality, introduced by Abu Radi and Kupferman for coBüchi automata [AK22]. In the conclusions of this chapter (Section VII.2) we further refine this notion of reduced signature automata and include a discussion about its relation with the minimisation of automata recognising positional languages.

The next lemma simply follows by faithfulness.

✦ **Lemma V.62.** *A reduced signature automaton is deterministic.*

Given a signature automaton $\mathcal{A}$, we can make it reduced just by merging $\sim_d$-classes. We can easily state this result using the quotient of automata.

✦ **Lemma V.63.** *Let $\mathcal{A}$ be a signature automaton $\mathcal{A}$ whose maximal even priority is d. Then, its $(\leq d)$-quotient by $\sim_d$, $\mathcal{A}/_{\leq_d}^{\sim_d}$, is an equivalent reduced signature automaton.*

4.1.3 Full progress consistency.

The existence of a signature automaton recognising an objective W does not suffice to ensure half-positionality of W. The problem is similar to the one we encountered when studying open objectives in Section V.3.2: there are open objectives whose residuals are totally ordered but their are not half-positional (see Example V.24). In that case, we needed to add the property of progress consistency to characterise positionality. We generalise this notion to signature automata with multiple preorders.

Definition V.64 (Full progress consistency)**.**

We say that a signature automaton $\mathcal{A}$ recognising an objective W is *fully progress consistent* if, for each preorder $\leq_x$, for x even, and every finite word $w \in \Sigma^*$:

$$q <_x p \text{ and } q \overset{w:\geq x}{\rightsquigarrow} p \implies w^\omega \in q^{-1}W.$$

✦ Remark V.65. *A fully progress consistent signature automaton is in particular progress consistent, as the $\leq_0$-preorder refines that coming from the inclusion of residuals.*

4.1.4 Structured signature automata from semantic properties of languages

To prove the implication (1) $\implies$ (2) from Theorem V.1, we build a signature automaton from a deterministic parity automaton $\mathcal{A}$ recognising W recursively. In order to be able to carry out the recursion, we will in fact obtain a signature automaton with even stronger properties. This reinforcement of signature automata is done by ensuring that the preorders $\leq_x$ come from semantic properties of the automata, for which the notion of $<x$-safe languages will play a major role. The properties that are imposed are essentially a generalisation of the ones satisfied by the canonical history-deterministic coBüchi automata defined by Abu Radi and Kupferman [AK22].

We introducing some further notation used in our semantic reinforcement of the definition of a signature automaton.

$<x$-safe languages. Let $\mathcal{A}$ be a (possibly non-deterministic) parity automaton. We define the *$(<x)$-safe language* of a state q of $\mathcal{A}$ as:

$$\mathsf{Safe}^{\mathcal{A}}_{<x}(q) = \{w \in \Sigma^* \cup \Sigma^\omega \mid \text{ there exists } q \overset{w:\geq x}{\rightsquigarrow}\}.$$

We remark that $\mathsf{Safe}^{\mathcal{A}}_{<x}(q)$ is completely determined by its finite (resp. infinite) words. We drop the superscript $\mathcal{A}$ whenever the automaton is clear from the context. A path producing no priority strictly smaller than x is called *$(<x)$-safe*.

Next lemma simply follows from the definition.

✦ **Lemma V.66** (Monotonicity of safe languages)**.** *Let $\mathcal{A}$ be a parity automaton that is deterministic over transitions using priorities $\geq x$. Let q and p be two states such that $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(p)$, and let $q \overset{u:\geq x}{\rightsquigarrow} q'$ be a $<x$-safe run over u from q. Then, there is a unique $<x$-safe run over u from p, $p \overset{u:\geq x}{\rightsquigarrow} p'$, and it leads to a state p' satisfying $\mathsf{Safe}_{<x}(q') \subseteq \mathsf{Safe}_{<x}(p')$.*

$<x$-safe components. A *$(<x)$-safe component* of $\mathcal{A}$ is a strongly connected component of the subautomaton obtained by removing all transitions producing a priority $< x$ from $\mathcal{A}$. Note that if $\mathcal{A}$ is in normal form and $x > 0$, transitions changing of $(<x)$-safe component produce a priority $< x$. That is, $q \overset{u:\geq x}{\rightsquigarrow} p$ implies that q and p are in the same $(<x)$-safe component.

✦ Remark V.67. *The partition of $\mathcal{A}$ into $<x$-safe components is a refinement of its partition into $<y$-safe components, for $y \leq x$.*

For the following, we fix a parity automaton $\mathcal{A}$ in normal form using priorities in $[0, d_{\max}]$. For each $x \in [1, d_{\max}]$, we will totally order the $<x$-safe components of $\mathcal{A}$ in such a way that these orders successively refine each other. For $x \in [1, d_{\max}]$ we let $S_1^{<x}, \dots, S_{k_x}^{<x}$ be the $<x$-safe components of $\mathcal{A}$. For $x = 1$, we let $S_1^{<1} <_1 S_2^{<1} <_1 \cdots <_1 S_{k_1}^{<1}$ be an arbitrary order over the <1-safe components. Assume that an order has already been fixed at level $x - 2$. Then, we fix an arbitrary total order for the $<x$-safe components contained in a same $<(x-1)$-safe components, which yields a total order for the set of all those safe components, that refines the previous layers. From now on, we assume that the enumerations $S_1^{<x}, \dots, S_{k_x}^{<x}$ correspond to these orders: $S_i^{<x} <_x S_j^{<x}$ if $i < j$.

Structured signature automata. The preorders of the signature automaton we plan to build will correspond to the following semantic properties:

1. **Preorder 0 given by inclusion of residuals.** Preorder $\leq_0$ corresponds to the inclusion of residuals:
$$q \leq_0 p \iff q^{-1}\mathcal{L}(\mathcal{A}) \subseteq p^{-1}\mathcal{L}(\mathcal{A}).$$
2. **Odd layers correspond to safe components.** For $x \geq 2$ even, we define $\leq_{x-1}$ by:
$$q \leq_{x-1} p \iff q <_{x-2} p \text{ or } [q \sim_{x-2} p \text{ and } q \in S_i^{<x} \text{ and } p \in S_j^{<x} \text{ with } i \leq j].$$
In particular, $q \sim_{x-1} p$ if and only if $q \sim_{x-2} p$ and there is a path $q \overset{w:\geq x}{\rightsquigarrow} p$.
3. **Even preorders given by inclusion of safe languages.** For $x \geq 2$, x even, we define $\leq_{x-1}$ by:
$$q \leq_x p \iff q <_{x-1} p \text{ or } [q \sim_{x-1} p \text{ and } \mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(p)].$$

These preorders already ensure some of the properties required to be a signature automaton; mainly, the local monotonicity of transitions using large priorities (Item (IV)), as well as the congruence for $\geq x$-transitions at $\sim_x$-classes. Note, however, that it is not clear (and will be an important part of our proof) that $\leq_x$ is total for even x.

Given an (even or odd) priority $d \in \mathbb{N}$, we say that a parity automaton $\mathcal{A}$ in normal form together with nested preorders $\leq_0, \leq_1, \ldots, \leq_d$ as above is a *d-structured signature automaton* if these preorders are total and moreover:

4. **Strong congruence of $(\leq x)$-priorities over even classes.** Let $0 \leq x \leq d$, x even. For every $y \leq x$:
$$q \sim_x q',\ q \xrightarrow{a:y} p \text{ and } q' \xrightarrow{a:z} p' \implies z = y \text{ and } p = p'.$$
5. **Classes at layer x are $(>x)$-connected.** For $0 \leq x \leq d$ and $q \sim_x q'$, we have:
$$q \neq q' \implies \text{there is a path } q \overset{u:>x}{\rightsquigarrow} q'.^{6}$$
6. **Safe centralisation.** Let $2 \leq x \leq d$ be an even priority, and let $q \sim_{x-2} p$. Then:
$$q \nsim_{x-1} p \implies \mathsf{Safe}_{<x}(q) \nsubseteq \mathsf{Safe}_{<x}(p).$$

We say that $\mathcal{A}$ is a *structured signature automaton* if it is a d-structured signature automaton, for d the maximal priority appearing in $\mathcal{A}$.

✦ Remark V.68. *We draw the reader's attention to the fact that in Item 4 we do not only require $p \sim_x p'$, but impose $p = p'$. This will be necessary to guarantee that the relations $\sim_y$ for even priorities $y > x$ are also congruences for x-transitions.*

[6] We remark that, for x odd, this property is already implied by Item 2.

Lemma V.69.

A deterministic d-structured signature automaton is a d-signature automaton.

Proof. The fact that $\leq_0$ refines the inclusion of residuals is ensured by Item 1. Also, the $(<x)$-safe separation at odd levels (Item (III)) is directly implied by the fact that odd layers correspond to safe components (Item 2).

We now show by induction on x that for each $x \leq d$, x even, $\sim_x$ is a $[0, x]$-faithful congruence. Consider two states $q \sim_x q'$, which rewrites as $q \sim_{x-1} q'$ and $\mathsf{Safe}_{<x}(q) = \mathsf{Safe}_{<x}(p)$, and pick a transition $q \xrightarrow{a:y} p$. There are two cases.

- If $y \leq x$, then by Item 4, we have $q' \xrightarrow{a:y} p$.
- If $y > x$, then by monotonicity of safe languages (Lemma V.66), we have $q' \xrightarrow{a:\geq x} p'$ with $\mathsf{Safe}_{<x}(p') = \mathsf{Safe}_{<x}(p)$, and by induction, $p' \sim_{x-2} p$. From Item 6, it follows that $p' \sim_{x-1} p$ and thus $p' \sim_x p$, as required.

We conclude that $\sim_x$ is a $[0, x]$-faithful congruence.

Finally, the local monotonicity of $(\geq x)$-transitions follows from the fact that even preorders correspond to the inclusion of safe languages (Item 3) and the monotonicity of safe languages (Lemma V.66). ◀

4.2 From half-positionality to signature automata

This section is devoted to the proof of the implication (1) $\implies$ (2) in Theorem V.1. Many of the ideas in this proof have already appeared in the warm-up section. However further technical issues stem from the fact that we manipulate general parity automata. Details for a number of proofs are relegated to Appendix C.

Global hypothesis in Subsection V.4.2.

In the whole section, W stands for an objective that is half-positional over finite, ε-free Eve-games. These hypotheses will not necessarily be recalled in the statements of propositions.

4.2.1 Outline of the induction

Given a deterministic parity automaton recognising a half-positional objective, we will recursively define the preorders and equivalence relations making $\mathcal{A}$ a structured signature automaton. The base case consists in showing that the preorder $\leq_0$ given by the inclusion of residuals is total, and ensuring Item 4 of the definition for this preorder. For the recursion step, we suppose that we have a deterministic $(x-2)$-structured signature automaton $\mathcal{A}$ recognising W, for x even, and we define preorders $\leq_{x-1}$ and $\leq_x$ over $\mathcal{A}$ as imposed by Items 2 and 3. Then, we apply a sequence of operations, after which we obtain an equivalent deterministic automaton, that is either x-structured signature, or has strictly less states than $\mathcal{A}$. In the first case, we continue to define preorders $\leq_{x+1}$ and $\leq_{x+2}$; in the second case, we restart the structuration procedure from the beginning, with a strictly smaller automaton. In both cases, we conclude by induction.

We conjecture that we can sequentially obtain all the preorders, without having to restart the construction at each step. However, we have not been able to overcome some

technical difficulties preventing us to do so. We refer to the final subsection of Appendix C for more details.

We give a more detailed account on the specific operations we apply to obtain the different items of the definition of a structured signature automaton and their order:

i) **Relation $\sim_{x-1}$ and safe centralisation.** We define $\sim_{x-1}$, as determined by Item 2. Applying a generalisation of the procedure from [AK22], we $(<x)$-safe centralise $\mathcal{A}$, obtaining an equivalent automaton satisfying Item 6. The resulting automaton is no longer deterministic, but it is history-deterministic and has a very restricted and controlled amount of non-determinism.

ii) **Total order in safe components.** We prove that the states of each $(<x)$-safe component are totally ordered by inclusion of $(<x)$-safe languages (for which we rely on the safe centralisation hypothesis). This shows that the preorder $\leq_x$ given by the inclusion of safe languages (Item 3) is total.

iii) **Re-determinisation.** We determinise automaton $\mathcal{A}$, while preserving previously obtained properties. For this, the fact that $\leq_x$ is total will be key.

iv) **Uniformity of x-transitions.** Finally, we show that either $\mathcal{A}$ already satisfies Items 4 and 5, or we can trim the automaton to an equivalent strictly smaller one.

Moreover, we show that all these transformations can be performed in polynomial time.

This establishes that an objective W that is half-positional over finite, ε-free Eve-games can be recognised by a deterministic structured signature automaton. At the end of the section, we show that such an automaton must be fully progress consistent (Lemma V.78).

4.2.2 Constructing structured signature automata for half-positional languages

Let $\mathcal{A} = (Q, \Sigma, q_{\mathsf{init}}, [0, d_{\max}], \Delta, \mathsf{parity})$ be a deterministic parity automaton recognising W, and suppose that W is half-positional over finite, ε-free Eve-games. We assume that $\mathcal{A}$ is in normal form.

Global hypothesis in Subsection 4.2.2.

In this subsection, we will apply successive transformations to the automaton $\mathcal{A}$, ensuring an increasing list of properties. At the beginning of each paragraph, we clearly state the properties that are assumed. We allow ourselves to omit these hypotheses in the statements of propositions inside the paragraphs.

Base case: Preorder $\leq_0$.

We define $q \leq_0 p$ if $q^{-1}\mathcal{L}(\mathcal{A}) \subseteq p^{-1}\mathcal{L}(\mathcal{A})$, as imposed by Item 1. In Lemma V.21, we showed that half-positionality of W implies that this order is total. However, in our proof we used infinite, and not necessarily ε-free games. It is not difficult to modify the proof to adapt to this set of minimal hypotheses, using ω-regularity of W. We give all details in Appendix C (Lemma C.8). Items 2, 3, and 6 are trivially satisfied. Therefore, it suffices to show that we can obtain an automaton such that $\sim_0$ is a strong congruence for transitions producing priority 0 (Item 4), and that $\sim_0$-equivalent states can be connected by paths producing priority $>x$ (Item 5). For this, we apply exactly the same method presented in Section V.3.3: we obtain a polished automaton and show that it satisfies the

desired properties. This proof will be covered in the recursive step; the case $x = 0$ does not present any particularity.

Moving on to the inductive step, for the rest of the subsection, we let x be an even priority such that $2 < x \leq d_{\max}$ and assume that $\mathcal{A}$ is a deterministic $(x-2)$-structured signature automaton.

Safe centrality and relation $\sim_{x-1}$.

We say that an automaton with a preorder $\leq_{x-2}$ is *$(<x)$-safe centralised* if $\sim_{x-2}$-equivalent states that are comparable for the inclusion of $(<x)$-safe languages are in the same $(<x)$-safe component.

✦ Remark V.70. *For automata in normal form $(<x)$-safe centrality can be stated as: if $q \sim_{x-2} p$ and there is no $(<x)$-safe path connecting q and p, then $\mathsf{Safe}_{<x}(q) \not\subseteq \mathsf{Safe}_{<x}(p)$.*

Lemma V.71 ($(<x)$-safe centralisation)**.**

There exists a $(x-2)$-structured signature automaton $\mathcal{A}'$ equivalent to $\mathcal{A}$ which is:

- deterministic over transitions with priority different from $x-1$,
- homogeneous,
- history-deterministic, and
- $(<x)$-safe centralised.

Moreover, $\mathcal{A}'$ can be obtained in polynomial time from $\mathcal{A}$ and $|\mathcal{A}'| \leq |\mathcal{A}|$.

The proof of this lemma is a refinement of the corresponding result for coBüchi automata presented in the warm-up (Lemma V.47): we saturate $\sim_{x-2}$-classes of the original automaton $\mathcal{A}$ with $(x-1)$-transitions, and then remove redundant $(<x)$-safe components recursively until obtaining a $(<x)$-safe centralised automaton. We include all details in Appendix C (page 302).

Lemma V.71 allows us to define $\sim_{x-1}$ satisfying all required properties: for $q \sim_{x-2} p$, we define $q \leq_{x-1} p$ if and only if $q \in S_i^{<x}$ and $p \in S_j^{<x}$ with $i \leq j$, where $S_i^{<x}$ are the $(<x)$-safe components of $\mathcal{A}$ enumerated following the order described in Section V.4.1. By definition, Item 2 is satisfied, and by $(<x)$-safe centralisation of $\mathcal{A}$, so is Item 6.

Preorder $\leq_x$: Total order given by safe languages

In all this paragraph we assume that $\mathcal{A}$ is an automaton as obtained in the previous paragraph, that is: it has nested preorders defined up to $\leq_{x-1}$ making it a $(x-2)$-structured signature automaton and satisfying Items 2 and 6 for relation $\sim_{x-1}$. Moreover, it is history-deterministic, homogeneous, and the only non-determinism of $\mathcal{A}$ appears in $(x-1)$-transitions.

We define preorder $\leq_x$ as imposed by Item 3:

$$q \leq_x p \iff q <_{x-1} p \text{ or } [q \sim_{x-1} p \text{ and } \mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(p)],$$

and recall that it follows that $q \sim_x p$ if and only if $q \sim_{x-1} p$ and there is a $(<x)$-safe path from q to p.

✦ Remark V.72. *Using Item 4 for priorities $y \leq x-2$ and Lemma V.66 for transitions with priority $\geq x$, we get that relation $\sim_x$ is a congruence for transitions with a priority different from $x-1$. Moreover, over each $\sim_{x-1}$-class, transitions with priority $\geq x$ are monotone for $\leq_x$.*

Our objective is now to show that $\leq_x$ is total over each $\sim_{x-1}$-class. The proof of this statement uses the same ideas as the corresponding result from the warm-up (Lemma V.50). In particular, the main technical point resides in proving that, for two states $q_1 \not\leq_x q_2$, we can force to produce priority $x-1$ from q_1 while remaining $<x$-safe from q_2, and then resynchronise both paths in a same $\sim_x$-class. This result, stated in Lemma V.74, is the analogue to Lemma V.49 from the warm-up. For its proof we strongly rely on the $(<x)$-safe centralisation of $\mathcal{A}$ and the fine control of its non-determinism.

In the proofs of Lemmas V.74 and V.75 we will reason at the level of $\sim_x$-classes. As we only suppose that $\mathcal{A}$ is $(x-2)$-structured, we do not have the uniformity of x-transitions over $\sim_x$-classes yet. Lemma V.73 below provides a weaker version of this uniformity that will suffice for the arguments in the upcoming lemmas.

We say that a word w *produces priority y uniformly* in a class $[q]_x$ if for every $q' \in [q]_x$ all runs from q' are of the form $q' \overset{w:y}{\rightsquigarrow}$. In that case, we write $[q]_x \overset{w:y}{\rightsquigarrow}$. We say that such a word produces priority x uniformly in $[q]_x$ *leading to* $[p]_x$ if for every $q' \in [q]_x$ we have $q' \overset{w:y}{\rightsquigarrow} p'$ with $p' \in [p]_x$. In that case, we write $[q]_x \overset{w:y}{\rightsquigarrow} [p]_x$.

We note that whenever $\mathcal{A}$ contains a path $q \overset{w:\geq x}{\rightsquigarrow} p$, a run over w is unique, as $\mathcal{A}$ is homogeneous and its restriction to transitions coloured with priorities $\geq x$ is deterministic.

The proof of the next lemma combines normality of $\mathcal{A}$ with ideas appearing in the proof of Claim V.33.1 from the warm-up; all the details can be found in Appendix C (page 314).

Lemma V.73 (Existence of uniform words)**.**

Let p and q be two states from the same $(<x)$-safe component. There is a word $w \in \Sigma^*$ producing priority x uniformly in $[q]_x$ leading to $[p]_x$.

We next state the result that allows us to synchronise runs in a same $\sim_x$-class. Its proof is analogous to that of Lemma V.49 and can be found in Appendix C.

We let r be a sound resolver for $\mathcal{A}$ implemented by a finite memory (which can be assumed by Lemma I.10), and assume that all states are reachable using this resolver. We recall that we write $q \overset{w:y}{\underset{\forall,\mathsf{r}}{\rightsquigarrow}} p$ if, for every word $u_0 \in \Sigma^*$ such that the induced run of r over u_0 arrives to q, the induced run of r over $u_0 w$ ends in p and produces y as minimal priority in the part of the run corresponding to w. Recall also that we write $[q]_x \overset{w:y}{\underset{\forall,\mathsf{r}}{\rightsquigarrow}} [p]_x$ if for any $q' \in [q]_x$ we have $q' \overset{w:y}{\underset{\forall,\mathsf{r}}{\rightsquigarrow}} p'$ for some $p' \in [p]_x$.

Lemma V.74 (Synchronisation of separating runs)**.**

Suppose that $q \sim_{x-1} q'$ and $q \not\leq_x q'$ and let $p \in [q]_{x-1}$. There is a word $w \in \Sigma^+$ such that $[q]_x \overset{w:x-1}{\underset{\forall,\mathsf{r}}{\rightsquigarrow}} [p]_x$ and $[q']_x \overset{w:x}{\underset{\forall,\mathsf{r}}{\rightsquigarrow}} [p]_x$.

We can now deduce that $\leq_x$ is total over each $\sim_{x-1}$-class.

Lemma V.75 (Total order in $(<x)$-safe components)**.**

Let $q, q' \in Q$ be two states such that $q \sim_{x-1} q'$. Then, either $q \leq_x q'$ or $q' \leq_x q$.

Proof. Suppose by contradiction that $\mathsf{Safe}_{<x}(q) \not\subseteq \mathsf{Safe}_{<x}(q')$ and $\mathsf{Safe}_{<x}(q') \not\subseteq \mathsf{Safe}_{<x}(q)$. Let p be a state in $[q]_{x-1} = [q']_{x-1}$, and apply Lemma V.73 to obtain words $u, u' \in \Sigma^*$ such that $[p]_x \overset{u:x}{\rightsquigarrow} [q]_x$ and $[p]_x \overset{u':x}{\rightsquigarrow} [q']_x$.

By Lemma V.74, there are words $w, w' \in \Sigma^\omega$ such that:

$$[q]_x \overset{w:x}{\underset{\forall,\mathsf{r}}{\rightsquigarrow}} [p]_x, \qquad [q]_x \overset{w':x-1}{\underset{\forall,\mathsf{r}}{\rightsquigarrow}} [p]_x,$$

$$[q']_x \overset{w:x-1}{\underset{\forall,\mathsf{r}}{\rightsquigarrow}} [p]_x, \quad [q']_x \overset{w':x}{\underset{\forall,\mathsf{r}}{\rightsquigarrow}} [p]_x.$$

The situation is analogous to the one depicted in Figure 37 in the warm-up. We obtain that:

- $(w'u)^\omega \notin q^{-1}W$,
- $(wu')^\omega \notin q^{-1}W$,
- $(wu'w'u)^\omega \in q^{-1}W$.

Let $u_0 \in \Sigma^*$ be a word such that the run induced by r over u_0 ends in q (it exists, as we have supposed that all states are reachable using r). It suffices to consider the game where there is path labelled u_0 leading to a vertex controlled by Eve with two self loops; one of them producing $w'u$ and the other wu'. By the previous remarks, she can win such game by alternating both loops, but she cannot win positionally. ◀

Re-obtaining determinism

In this paragraph we assume that $\mathcal{A}$ is a parity automaton recognising W equipped with nested total preorders defined up to $\leq_x$ with all properties obtained until now:

- it is a $(x-2)$-structured signature automaton,
- preorder $\leq_{x-1}$ satisfies properties from Items 2 and 6 from the definition of a structured signature automaton,
- preorder $\leq_x$ satisfies the property from Item 3 from the definition of a structured signature automaton,
- it is deterministic over transitions with priorities different from $x-1$,
- it is homogeneous, and
- it is history-deterministic.

We claim that we can obtain a deterministic equivalent automaton preserving the entire structure of total preorders. Moreover, in the obtained automaton we guarantee that relation $\sim_x$ satisfies Item 4 from the definition of a structured signature automaton for priorities $y < x$.

Lemma V.76 (Re-determinisation)**.**

There is a deterministic parity automaton $\mathcal{A}'$ equivalent to $\mathcal{A}$ with nested total preorders defined up to $\leq_x$ satisfying that:

- it is a $(x-2)$-structured signature automaton,

- preorder $\leq_{x-1}$ satisfies properties from Items 2 and 6 from the definition of a structured signature automaton, and
- preorder $\leq_x$ is a congruence and satisfies the property from Item 3 and, for priorities $y < x$, also that from Item 4 .

Moreover, automaton $\mathcal{A}'$ can be computed in polynomial time from $\mathcal{A}$ and $|\mathcal{A}'| \leq |\mathcal{A}|$.

The idea of the proof is a direct generalisation of the one presented in the warm-up for coBüchi automata (page 209): we redefine the $(x-1)$-transitions of the automaton in such a way that we ensure that a run that changes of $<x$-safe component infinitely often passes through all these components in a round-robin fashion. The total order $\leq_x$ allows us to identify a maximal state in each component, so we can make a deterministic choice. Formal details can be found in Appendix C (page 311).

Uniformity of x-transitions over $\sim_x$-classes

We assume that $\mathcal{A}$ is a deterministic parity automaton recognising W with nested total preorders defined up to $\leq_x$ satisfying all conditions stated in Lemma V.76. The objective of this paragraph is to obtain the remaining properties of a x-structured signature automaton (Items 4 and 5).

Lemma V.77 (Uniformity of x-transitions over $\sim_x$-classes)**.**

There is a deterministic parity automaton $\mathcal{A}'$ equivalent to $\mathcal{A}$ such that either:

- $\mathcal{A}'$ is an x-structured signature automaton with $|\mathcal{A}'| \leq |\mathcal{A}|$, or
- $|\mathcal{A}'| < |\mathcal{A}|$.

In both cases, such an automaton can be computed in polynomial time from $\mathcal{A}$.

The proof of this lemma generalises the techniques introduced in Section V.3.3 of the warm-up. Details can be found in Appendix C (from page 314). We introduce the local automaton of a $\sim_x$-class $[q]_x$: the automaton originated by keeping the states of $[q]_x$ and paths connecting them producing priorities $\geq x$. Using half-positionality and ideas analogous to those from Lemma V.33, we show that these local automata admit a well-defined set of super letters, that is, there are letters that, if read infinitely often in such a local automaton, must produce an accepting word. These letters are exactly the ones carrying priority x when read from $[q]_x$ in the final automaton $\mathcal{A}'$.

To obtain the uniformity of x-transitions, we might need to simplify the automaton: we introduce x-polished automata, the target form of automata that will allow us to obtain uniformity of x-transitions. Using the existence of super letters, we show that we can polish automaton $\mathcal{A}$ by removing redundant parts of it. This operation might break the normal form of automaton $\mathcal{A}$,[7] but this is not a problem, since in any case it strictly decreases the number of states of the automaton, as desired.

This ends the induction step of the proof, establishing existence of a deterministic structured signature automaton recognising W.

[7] In fact, we believe that the polishing operation does preserve normality, but we have not been able to prove it.

4.2.3 Full progress consistency

We show that a structured signature automaton recognising a half-positional objective must be fully progress consistent. Since we showed how to obtain such a structured signature automaton in the previous section, this ends the proof of the implication (1) $\implies$ (2) from Theorem V.1.

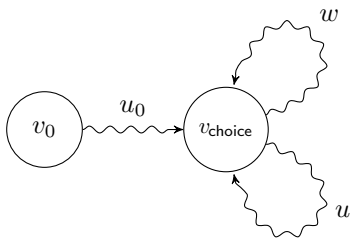
Lemma V.78 (Necessity of full progress consistency)**.**

Let $W \subseteq \Sigma^\omega$ be half-positional over finite, ε-free Eve-games. Any structured signature automaton recognising W is fully progress consistent.

Proof. Suppose by contradiction that $\mathcal{A}$ is a structured signature automaton for W that is not fully progress consistent. By definition, for some priority x even, there are $q <_x p$ and a word $w \in \Sigma^*$ such that $q \overset{w:\geq x}{\rightsquigarrow} p$, but $w^\omega \notin q^{-1}W$. As $\sim_x$ is a $[0,x]$-faithful congruence, we can work with $\sim_x$-classes and write $[q]_x \overset{w:\geq x}{\rightsquigarrow} [p]_x$. We study first the case $x > 0$. By Lemma V.74, there is a word $u \in \Sigma^+$ such that $[q]_x \overset{u:x-1}{\rightsquigarrow} [q]_x$ and $[p]_x \overset{u:x}{\rightsquigarrow} [q]_x$. Let $u_0 \in \Sigma^+$ be a word reaching q from the initial state of $\mathcal{A}$.[8] We obtain:

- $u_0 w^\omega \notin W$,
- $u_0 u^\omega \notin W$, and
- $u_0 (wu)^\omega \in W$.

We consider the game depicted in Figure 40. Eve can win from v_0 by alternating loops labelled w and u when the play arrives to v_{choice}. However, she cannot win positionally from v_0.



✦ **Figure 40.** A game $\mathcal{G}$ in which Eve cannot play optimally using positional strategies if $\mathcal{A}$ is not fully progress consistent.

For the case $x = 0$, the proof is almost identical to that of Lemma V.26; we just need to ensure that the game in Figure 31 (page 197) can be supposed finite and ε-free. Finiteness of the game can be obtained by using Lemma I.17. To guarantee that we do not include ε-transitions, if $u_0 = \varepsilon$, we remove vertex v_0 from the game. ◀

4.2.4 Complexity analysis

We now establish decidability of half-positionality of ω-regular languages in polynomial time, stated as Theorem V.2. In this subsection we assume that the equivalence between

[8] If q is initial, we omit u_0 and the state v_0 of the game from Figure 40 to ensure the use of an ε-free game.

Items (2) and (5) from Theorem V.1 holds – although at this point we have only shown necessity of Item (2).

Complexity of building a signature automata

In this section, we show how, given a deterministic parity automaton $\mathcal{A}$, one may apply the construction from the previous subsection, to decide whether $W = \mathcal{L}(\mathcal{A})$ is half-positional. The general idea is simply to go through the construction from Section V.4.2, either ending up with a failure indicating that W is not positional, or with a deterministic structured signature automaton. At this point, it suffices to check full progress consistency, which can be done in polynomial time, as explained below.

Most proofs have been already given in the previous section; we also require the following easy lemma which follows from Proposition I.15.

✦ **Lemma V.79.** *Let $\mathcal{A}$ be a parity automaton and x a priority. Assume that $\mathcal{A}$ is deterministic over $\geq x$-transitions. Given two states q, p in $\mathcal{A}$, we can decide in polynomial time whether $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(p)$.*

We now detail the polynomial-time procedure. Let $\mathcal{A}$ be a DPA recognising W.

- First, we check for each pair of states q, p whether $\mathcal{L}(\mathcal{A}_q) \subseteq \mathcal{L}(\mathcal{A}_p)$, or $\mathcal{L}(\mathcal{A}_p) \subseteq \mathcal{L}(\mathcal{A}_q)$. If for some pair of states these languages are incomparable, then residuals of W are not totally ordered, and we can conclude that W is not half-positional.

Suppose that we have defined total preorders up to $\leq_{x-2}$ making $\mathcal{A}$ a $(x-2)$-structured signature automaton.

- We $<x$-safe centralise $\mathcal{A}$, which can be done in polynomial time by Lemma V.71. The obtained automaton is deterministic over $\geq x$-transitions.
- We compute the $<x$-safe components of $\mathcal{A}$, which can be done by doing a decomposition in SCCs of $\mathcal{A}|_{\geq x}$. We check whether, for each $<x$-safe component S and state q, the states in $S \cap [q]_{x-2}$ are totally preordered by inclusion of $<x$-safe languages, which can be done in polynomial time by Lemma V.79. If this is not the case, we conclude that W is not half-positional.
- We remove the non-determinism from $\mathcal{A}$ – in polynomial time and without increasing the number of states – by applying Lemma V.76.
- We compute (in polynomial time) the automaton $\mathcal{A}'$ given by Lemma V.77 (see last subsection of Appendix C for details). We check whether $\mathcal{L}(\mathcal{A}') = W$, which can be done in polynomial time by Proposition I.15. If this is not the case, we conclude that W is not half-positional.

After these operations, if we have not yet found that W is not half-positional, we obtain an equivalent deterministic automaton $\mathcal{A}'$ that is either x-structured signature, or strictly smaller than $\mathcal{A}$. In the former case, we continue defining preorders $\leq_{x+1}$ and $\leq_{x+2}$; in the latter, we restart from the beginning. In total, we repeat at most $d \cdot |Q|$ times a sequence of operations that take polynomial time.

Checking full progress consistency

Assume that we have a deterministic structured signature automaton recognising $\mathcal{A}$. We cannot yet conclude that W is half-positional, as we do not know whether $\mathcal{A}$ is fully

progress consistent, however, by Lemma V.78, if W is half-positional this must be the case. By Theorem V.1, this condition is also sufficient. We show now that we can check full progress consistency of $\mathcal{A}$ in polynomial time, finishing the proof of Theorem V.2.

Lemma V.80.

Let $\mathcal{A}$ be a deterministic structured signature automaton. We can decide in polynomial time whether $\mathcal{A}$ is fully progress consistent.

The proof crucially relies in the following lemma.

✦ **Lemma V.81.** *A deterministic structured signature automaton $\mathcal{A}$ is fully progress consistent if and only if, for each even priority x and each pair of states q, p in $\mathcal{A}$ such that $q <_x p$ we have:*

$$q \overset{w:\geq x}{\rightsquigarrow} p \text{ and } p \overset{w:y}{\rightsquigarrow} p \implies y \text{ is even.} \tag{1}$$

Proof. It follows directly from the definition that a deterministic fully progress consistent automaton satisfies this property. To show the converse, assume that (1) holds; we aim to prove full progress consistency. Consider an even priority x and a word $w \in \Sigma^*$ such that $q <_x p$ and $q \overset{w:\geq x}{\rightsquigarrow} p$, we should prove $w^\omega \in q^{-1}W$. We take x to be minimal such that $q <_x p$, and thus we have $q \sim_{x-2} p$. For $x = 0$, the proof is identical to the one appearing in Lemma V.26, we assume that $x \geq 2$. Therefore, there is a $<x$-safe path connecting q and p, so we have $q \sim_{x-1} p$ (Item 2 from the definition of structured signature automaton), thus since $q <_x p$ we get $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(p)$ (Item 3). Consider the run over w^ω from q in $\mathcal{A}$:

$$\rho = q \overset{w:\geq x}{\rightsquigarrow} p \overset{w:y_1}{\rightsquigarrow} p_2 \overset{w:y_2}{\rightsquigarrow} p_3 \overset{w:y_3}{\rightsquigarrow} \cdots .$$

Since $w \in \mathsf{Safe}_{<x}(p)$ and $q <_x p$, Lemma V.66 yields $\mathsf{Safe}_{<x}(p) \subseteq \mathsf{Safe}_{<x}(p_2)$, and hence $y_1 \geq x$. Since $\sim_{x-2}$ is $[0, x-2]$-faithful and $q \overset{w:\geq x-2}{\rightsquigarrow} p \sim_{x-2} q$, it follows that $p_2 \sim_{x-2} p$. Then it follows from $p \overset{w:\geq x}{\rightsquigarrow} p_2$ that $p \sim_{x-1} p_2$, and thus $p \leq_x p_2$. Applying the same reasoning by induction yields $y_i \geq x$ and $p \leq_x p_i$ for all i, and thus $q <_x p_i$

Eventually, ρ closes a cycle: there are N and k such that, for every $i \geq N$, $p_i = p_{i+k}$. We let $p' = p_{kN}$ and let y denote the minimal priority produced by the cycle. Then it holds that:

$$q <_x p', \quad q \overset{w^{kN}:\geq x}{\rightsquigarrow} p', \quad \text{and} \quad p' \overset{w^{kN}:y}{\rightsquigarrow} p'.$$

Thus thanks to (1), y is even, and so $w^\omega = (w^{kN})^\omega \in q^{-1}W$. ◀

We can now deduce Lemma V.80.

Proof of Lemma V.80. For each pair of states $q, p \in Q$ and each priority x, we define the languages of finite words

$$L^x_{q \to p} = \{w \in \Sigma^* \mid q \overset{w:x}{\rightsquigarrow} p \text{ in } \mathcal{A}\}, \quad \text{and} \quad L^{\geq x}_{q \to p} = \{w \in \Sigma^* \mid q \overset{w:\geq x}{\rightsquigarrow} p \text{ in } \mathcal{A}\}.$$

By Lemma V.81, $\mathcal{A}$ is fully progress consistent if and only if, for each even priority $x \in [0, d]$ and each pair of states $q, p \in Q$ such that $q <_x p$:

$$L^{\geq x}_{q \to p} \bigcap \Big(\bigcup_{y \text{ odd}} L^y_{p \to p} \Big) = \emptyset.$$

We show that for all pair of states, languages $L^{\geq x}_{q\to p}$ and $L^{x}_{q\to p}$ are regular and we can obtain deterministic finite automata for them in polynomial time. This implies that we can check the emptiness of intersections above in polynomial time, concluding the proof.

For $L^{\geq x}_{q\to p}$ the previous claim is clear: the finite automaton obtained by taking the automaton structure of $\mathcal{A}|_{\geq x}$ and taking q and p as initial and final states, respectively, is a finite automaton recognising $L^{\geq x}_{q\to p}$.

For $L^{x}_{q\to p}$, we consider the automaton over finite words that has as states $(Q\times[0,d])\cup\{(q,\mathsf{init})\}$, and (q,init) and (p,x) as initial and final states, respectively. Transitions of the automaton are of the form $(q_1,x_1)\xrightarrow{a}(q_2,x_2)$ if the transition $q_1\xrightarrow{a:y}q_2$ in $\mathcal{A}$ is such that $x_2=\min\{x_1,y\}$. In words, this automaton keeps track of the run in $\mathcal{A}$ from q and of the minimal priority produced in the way. It accepts a word if it arrives to p while producing as minimal priority x, as we wanted. ◀

4.3 From signature automata to half-positionality

We now complete the proof of Theorem V.1 by showing the two implications (2) $\implies$ (3) and (3) $\implies$ (4). The implication (4) $\implies$ (5) follows from Proposition V.14 (taken from [Ohl23a]) and (5) $\implies$ (1) is trivial.

4.3.1 ε-complete automata

Definition V.82 (ε-complete automata)**.**

An *ε-complete* automaton $\mathcal{A}$ is a non-deterministic parity automaton (with ε-transitions), with the property that for any pair of states $q,q'\in Q$, and for any even priority $x<d$ (where d is the maximal priority in $\mathcal{A}$):

$$\text{either}\quad q\xrightarrow{\varepsilon:x}q'\quad\text{or}\quad q'\xrightarrow{\varepsilon:x+1}q.$$

On an intuitive level, $q\xrightarrow{\varepsilon:x}q'$ means that "q is much better than q'", since one may, at any point, move from q to q' and be rewarded with an even priority x on the way. On the contrary, $q'\xrightarrow{\varepsilon:x+1}q$ means that "q' is not much worse than q", since one may at any point move from q' to q at the cost of reading an odd priority $x+1$. In other words, in a ε-complete automaton, one may say that q and q' are comparable for priority x.

Observe that whenever an automaton contains a transition $q\xrightarrow{\varepsilon:x}q'$ for even x, one may also add $q\xrightarrow{\varepsilon:x+1}q'$ without increasing the language. Hence one may see ε-completeness as a (much) stronger version of totality of $\xrightarrow{\varepsilon:x+1}$.

4.3.2 From signature automata to ε-complete automata

We now prove the implication (2) $\implies$ (3) from Theorem V.1, which can be stated as follows.

Lemma V.83 (From (2) to (3) in Theorem V.1)**.**

Let W be recognised by a fully progress consistent deterministic signature automaton $\mathcal{A}$. There exists a history-deterministic ε-complete automaton $\mathcal{A}'$ recognising W.

We prove Lemma V.83. Let $\mathcal{A}$ be a fully progress consistent deterministic signature automaton with nested preorders $\leq_0, \leq_1, \ldots, \leq_d$. Consider the automaton $\mathcal{A}'$ obtained from $\mathcal{A}$ by adding, for all even priorities $x \in [0, d]$, transitions $q \xrightarrow{\varepsilon:x+1} q'$ whenever $q' \leq_x q$ and $q \xrightarrow{\varepsilon:x} q'$ whenever $q' <_x q$. Note that $\mathcal{A}'$ (potentially) has transitions with priorities up to $d+1$.

✦ Remark V.84. *Note that, for x even, $q \xrightarrow{\varepsilon:\geq x} p$ in $\mathcal{A}'$ entails $p \leq_x q$.*

Since by definition, $q' <_x q$ is the negation of $q \leq_x q'$, it follows immediately that $\mathcal{A}'$ is ε-complete. Moreover, as $\mathcal{A}$ is a subautomaton of $\mathcal{A}'$, the inclusion $W \subseteq \mathcal{L}(\mathcal{A})$ is trivial, and $\mathcal{A}'$ is determinisable by pruning. The difficulty lies in showing that $W \subseteq \mathcal{L}(\mathcal{A}')$.

✦ Remark V.85. *If $q \overset{w:x}{\rightsquigarrow} q$ is a cycle in $\mathcal{A}'$ producing an even minimal priority, then w is not composed exclusively of ε-letters.*

For a priority x (even or odd), we say that a transition $q \xrightarrow{\varepsilon} q'$ in $\mathcal{A}'$ is an *x-jump* if $q' <_x q$. We remark that if $x' \leq x$, an x'-jump is an x-jump. We start with a useful technical lemma.

✦ **Lemma V.86.** *Fix a path $q' \overset{w':\geq x}{\rightsquigarrow} p'$ in $\mathcal{A}'$, with x even, such that and consider a run $q \overset{w}{\rightsquigarrow} p$ in $\mathcal{A}$, where w is obtained from w' by removing ε-letters (where $p = q$ if w' is empty).*

a) *Assume that there is no x-jump on $q' \overset{w':\geq x}{\rightsquigarrow} p'$ and that $q \sim_x q'$. Then $p \sim_x p'$ and $q \overset{w:\geq x}{\rightsquigarrow} p$ in $\mathcal{A}$. Moreover, if $q' \overset{w':x}{\rightsquigarrow} p'$, then $q \overset{w:x}{\rightsquigarrow} p$ in $\mathcal{A}$.*

b) *Assume that there is no $(x-1)$-jump on $q' \overset{w':\geq x}{\rightsquigarrow} p'$, that $q' \sim_{x-1} q$ and $q' \leq_x q$. Then $p' \sim_{x-1} p$, $p' \leq_x p$ and $q \overset{w:\geq x}{\rightsquigarrow} p$ in $\mathcal{A}$.*

c) *We have that $p' \leq_{x-1} q'$.*

Proof. In the two first cases we deal with the case of a letter and conclude by induction.

a) There are two possibilities, depending on whether the letter is ε or not.

- Transition $q' \xrightarrow{a:\geq x} p'$ with $a \in \Sigma$. Then $[0, x]$-faithfulness of $\sim_x$ gives $p \sim_x p'$ and $q \xrightarrow{a:\geq x} p$. Moreover if $q' \xrightarrow{a:x} p'$, then by $[0, x]$-faithfulness, $q \xrightarrow{a:x} p$.
- Transition $q' \xrightarrow{\varepsilon:\geq x} p'$. Then $p' \leq_x q'$ and since there is no x-jump, $p' \sim_x q'$. Thus $p = q \sim_x q' \sim_x p'$.

b) We distinguish the two same cases.

- Transition $q' \xrightarrow{a:\geq x} p'$ with $a \in \Sigma$. Then local monotonicity of $(\geq x)$-transitions in $\mathcal{A}$ yields $q \xrightarrow{a:\geq x} p$ in $\mathcal{A}$ with $p' \leq_x p$. By Remark V.60, $p' \sim_{x-1} p$.
- Transition $q' \xrightarrow{\varepsilon:\geq x} p'$. This implies $p' \leq_x q'$ and since there is no $(x-1)$-jump, we have $p' \sim_{x-1} q'$. Thus we conclude that $p' \leq_x q' \leq_x q = p$ and $p = q \sim_{x-1} q' \sim_{x-1} p'$.

c) Suppose by contradiction that $p' >_{x-1} q'$, and let q'_1 be the first state in the run such that $q' <_{x-1} q'_1$. We have:

$$q' \overset{w'_1:\geq x}{\rightsquigarrow} q'_2 \xrightarrow{a:\geq x} q'_1 \overset{w'_2}{\rightsquigarrow} p',$$

with $q'_2 <_{x-1} q'_1$. As in particular $q'_2 <_x q'_1$, $a \neq \varepsilon$ (Remark V.84). However, this contradicts Item (III) from the definition of signature automaton. ◀

We now state the key result for deriving Lemma V.83.

✦ **Lemma V.87.** *Consider a cycle $q' \overset{w':x}{\rightsquigarrow} q'$ in $\mathcal{A}'$ with x even, and let w be obtained from w' by removing ε-letters. Then, w^ω is accepted from q' in $\mathcal{A}$.*

Proof. We note that by Remark V.85, w is not empty. Let y be minimal such that an y-jump appears on the path $q' \overset{w':x}{\rightsquigarrow} q'$.

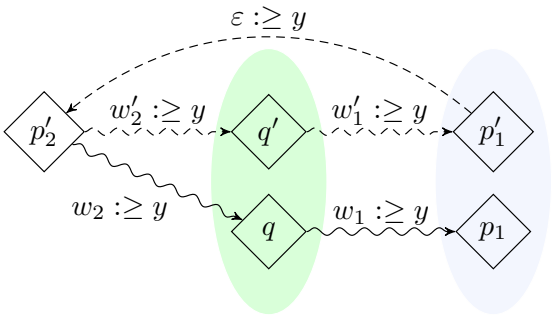
- If $y \geq x$. Then there is no $\overset{\varepsilon:x}{\longrightarrow}$ transition on the path $q' \overset{w':x}{\rightsquigarrow} q'$ (otherwise it would produce an x-jump). Thus Lemma V.86.a proves $q' \overset{w:x}{\rightsquigarrow} q_1$ in $\mathcal{A}$ with $q_1 \sim_x q'$. Then, since the $\sim_x$-class is preserved, successive applications of Lemma V.86.a give $q' \overset{w:x}{\rightsquigarrow} q_1 \overset{w:x}{\rightsquigarrow} q_2 \overset{w:x}{\rightsquigarrow} q_3 \overset{w:x}{\rightsquigarrow} \dots$ in $\mathcal{A}$, and thus w^ω is accepted from q' in $\mathcal{A}$.
- If $y < x$ and y odd. We show that this case cannot happen. Let $p'_1 \xrightarrow{\varepsilon} p'_2$ denote the first y-jump on the path $q' \overset{w}{\rightsquigarrow} q'$ in $\mathcal{A}'$, that is, we have

$$q' \overset{w'_1:>y}{\rightsquigarrow} p'_1 \xrightarrow{\varepsilon:>y} p'_2 \overset{w'_2:>y}{\rightsquigarrow} q' \text{ in } \mathcal{A}', \quad p'_2 <_y p'_1.$$

  By Lemma V.86.c, we have that $p'_1 \leq_y q'$, so $p'_2 <_y q'$. The existence of a path $p'_2 \overset{w'_2:\geq y}{\rightsquigarrow} q'$ contradicts Lemma V.86.c.
- If $y < x$ and y even. Let $p'_1 \xrightarrow{\varepsilon} p'_2$ denote the last y-jump on the path $q' \overset{w}{\rightsquigarrow} q'$ in $\mathcal{A}'$, that is, we have

$$q' \overset{w'_1:\geq y}{\rightsquigarrow} p'_1 \xrightarrow{\varepsilon:\geq y} p'_2 \overset{w'_2:\geq y}{\rightsquigarrow} q' \text{ in } \mathcal{A}', \quad p'_2 <_y p'_1,$$

  and there is no y-jump on $p'_2 \overset{w_2}{\rightsquigarrow} q'$. We let w_1, w_2 be obtained, respectively, from w'_1 and w'_2 by removing ε's. By Lemma V.86.a, we get that $p'_2 \overset{w_2:\geq y}{\rightsquigarrow} q$ in $\mathcal{A}$ for some $q \sim_y q'$. As there is no $(y-1)$-jump in the path, by Lemma V.86.b, we get that $q \overset{w_1:\geq y}{\rightsquigarrow} p_1$ in $\mathcal{A}$ for $p'_2 <_y p'_1 \leq_y p_1$. See Figure 41 for an illustration of the situation.



✦ **Figure 41.** Situation in the third case of Lemma V.87. Dashed lines represent paths in $\mathcal{A}'$ and solid lines those in $\mathcal{A}$. States that are $\sim_y$-equivalent are encircled together.

All in all, we have obtained a path

$$p'_2 \overset{w_2:\geq y}{\rightsquigarrow} q \overset{w_1:\geq y}{\rightsquigarrow} p_1 >_y p'_2 \quad \text{in } \mathcal{A}.$$

Therefore, full progress consistency yields $(w_2w_1)^\omega \in (p_2')^{-1}W$. As $(p_2')^{-1}W \subseteq q'^{-1}W = (q)^{-1}W$, we conclude that $w^\omega = (w_1w_2)^\omega(q)^{-1}W$. ◀

We are now ready conclude the proof of Lemma V.83.

Proof of Lemma V.83. As mentioned above, the inclusion $W \subseteq \mathcal{L}(\mathcal{A}')$ is trivial, as $\mathcal{A}$ is a subautomaton of $\mathcal{A}'$. This shows that, if the converse inclusion holds, $\mathcal{A}'$ is determinisable by pruning and therefore it is also history-deterministic.

We show $\mathcal{L}(\mathcal{A}') \subseteq W$. Take an accepting run in $\mathcal{A}'$ over $w' \in \Sigma^\omega$ and decompose it as:

$$q_0 \xrightarrow{w_0'} q' \xrightarrow{w_1':x} q' \xrightarrow{w_2':x} q' \xrightarrow{w_3':x} \cdots,$$

where x is even. For each i, let w_i be obtained from w_i' by removing ε's (which is non-empty by Remark V.85), and consider the corresponding run in $\mathcal{A}$:

$$q_0 \xrightarrow{w_0} q_1 \xrightarrow{w_1} q_2 \xrightarrow{w_2} q_3 \xrightarrow{w_3} \cdots,$$

It follows by induction that $q' \leq_0 q_i$, so, as order $\leq_0$ refines the order of residuals (property (I) of a signature automaton), words that are accepted from q' in $\mathcal{A}$ are also accepted from q_i. By Lemma V.86, it holds that for each pair of indices $j \leq j'$ we have $(w_jw_{j+1}\dots w_{j'})^\omega \in q^{-1}W$, so these words are also accepted from q_i, for all i.

Let $i_1, i_2, \dots$ be a sequence of indices such that $q_{i_j} = q_{i_{j+1}}$ for all j, and let $\tilde{q} = q_{i_1}$ be such recurring state. Each word $w_{i_j} \dots w_{i_{j+1}}$ forms a cycle over $\tilde{q}$, that, by the previous remark, must be accepting, so the minimal priority produced on it is even. Therefore, we have found a decomposition of the run over w in $\mathcal{A}$ of the form

$$q_0 \xrightarrow{w_0w_1\dots} \tilde{q} \xrightarrow{w_{i_1}\dots w_{i_2}:x_1} \tilde{q} \xrightarrow{w_{i_2}\dots w_{i_3}:x_2} \cdots,$$

with all x_i even. We conclude that $w_0w_1\cdots \in W$. ◀

4.3.3 Universal graphs from ε-complete automata

We now move on to implication (3) $\implies$ (4) in Theorem V.1, which is now recalled.

Proposition V.88 ((3) $\implies$ (4) in Theorem V.1)**.**

If there is a ε-complete history-deterministic automaton recognising W, then there is a well-ordered monotone (κ, W)-universal graph for each cardinal κ.

For the rest of the section, we let $\mathcal{A}$ be a ε-complete history-deterministic automaton recognising W, and we let d be even such that $\mathcal{A}$ has priorities up to $d+1$, as in the above section.

Closure of an ε-complete automaton

We define the order $\preccurlyeq$ over priorities in $[0, d+1]$ that sets $y \preccurlyeq x$ if x is "preferable" to y, that is: $1 \preccurlyeq 3 \preccurlyeq \dots d+1 \preccurlyeq d \preccurlyeq \dots \preccurlyeq 2 \preccurlyeq 0$.

✦ Remark V.89. *For any pair of infinite words $w, w' \in [0, d+1]^\omega$ satisfying that for all i $w_i \preccurlyeq w_i'$, it holds that:*

$$w \in \mathsf{parity}_{[0,d+1]} \implies w' \in \mathsf{parity}_{[0,d+1]}.$$

We say that an automaton $\mathcal{A}$ is *priority-closed* if:

- for any states q, q', priorities $y' \preccurlyeq y$, and $a \in \Sigma \cup \{\varepsilon\}$

$$q \xrightarrow{a:y} q' \implies q \xrightarrow{a:y'} q'$$

- for any states p, p', q, q' and $a \in \Sigma \cup \{\varepsilon\}$,

$$p \xrightarrow{\varepsilon:y_1} q \xrightarrow{a:y_2} q' \xrightarrow{\varepsilon:y_3} p' \implies p \xrightarrow{a:\min_{\preccurlyeq}(y_1,y_2,y_3)} p'.$$

It is easy to turn any automaton into a priority-closed one.

✦ **Lemma V.90.** *Let $\mathcal{A}$ be an automaton recognising W. There is an automaton $\mathcal{A}'$ recognising W which is priority-closed. Moreover, if $\mathcal{A}$ is history-deterministic and ε-complete, then so is $\mathcal{A}'$.*

Proof. Let $\mathcal{A}'$ be obtained by adding to $\mathcal{A}$ all transitions of the form $q \xrightarrow{a:y'} q'$, when $q \xrightarrow{a:y} q'$ is a transition in $\mathcal{A}$ and $y' \preccurlyeq y$, and all transitions of the form $p \xrightarrow{a:\min_{\preccurlyeq}(y_1,y_2,y_3)} p'$, whenever a path $p \xrightarrow{\varepsilon:y_1} q \xrightarrow{a:y_2} q' \xrightarrow{\varepsilon:y_3} p'$ appears in $\mathcal{A}$. Clearly, $\mathcal{A}'$ is priority-closed, $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$ and if $\mathcal{A}$ is ε-complete, then so is $\mathcal{A}'$. The fact that this operation preserves history-determinism is also clear, once the equality of languages is obtained. To prove that $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$, take an accepting run over $w \in \Sigma^\omega$ in $\mathcal{A}'$. We build a run over w in $\mathcal{A}$ by replacing any newly added transition $q \xrightarrow{a:y'} q'$ by $q \xrightarrow{a:y} q'$, and $p \xrightarrow{a:\min_{\preccurlyeq}(y_1,y_2,y_3)} p'$ by $p \xrightarrow{\varepsilon:y_1} q \xrightarrow{a:y_2} q' \xrightarrow{\varepsilon:y_3} p'$, respectively. By Remark V.89, the obtained run is accepting in $\mathcal{A}$. ◀

In a priority-closed automaton, for each priority y, transitions $\xrightarrow{\varepsilon:y}$ define a transitive relation. If the automaton is moreover ε-complete, then for each even priority x, transitions of the form $\xrightarrow{\varepsilon:x+1}$ define total preorders.

We define $q \geq_x q'$ if $q \xrightarrow{\varepsilon:x+1} q'$. Note that, since $\mathcal{A}$ is priority-closed, these preorders are nested: $q \geq_{x+2} q'$ implies $q \geq_x q'$. Moreover, since $\mathcal{A}$ is ε-complete, for any even x:

$$q >_x q' \implies q \xrightarrow{\varepsilon:x} q'.$$

Finally, observe that, by priority-closure, states q, q' that are $\leq_d$-equivalent have exactly the same incoming and outgoing transitions, and can thus be merged without altering the language (this transformation preserves history-determinism). Therefore, we may assume that $\leq_d$ is antisymmetric and thus defines a total order on Q.

We write $[q]_x$ to denote the equivalence class of q associated to preorder $\leq_x$. That is, $[q]_x$ contains the states q' such that $q \leq_x q'$ and $q' \leq_x q$.

Definition of the graph

For the remaining of the section, we fix a cardinal κ. Let us first recall the construction of the $(\kappa, \mathsf{parity})$-universal graph U_{parity} for the parity objective over $\{0, \dots, d+1\}$ (see Example V.16 for a proof of universality). Its vertices are of the form $(\lambda_1, \lambda_3, \dots, \lambda_{d+1}) \in \kappa^{d/2+1}$, ordered lexicographically, and its edges are given by

$$(\lambda_1, \dots, \lambda_{d+1}) \xrightarrow{x} (\lambda'_1, \dots, \lambda'_{d+1}) \iff \begin{cases} (\lambda'_1, \dots, \lambda'_{x-1}) \leq (\lambda_1, \dots, \lambda_{x-1}), & \text{if } x \text{ is even,} \\ (\lambda'_1, \dots, \lambda'_x) < (\lambda_1, \dots, \lambda_x), & \text{otherwise.} \end{cases}$$

Fix a priority-closed ε-complete and history-deterministic automaton $\mathcal{A}$ with states Q such that $\leq_d$ defines a total order on Q.

We define a Σ-graph $U_{\mathcal{A}}$ as follows. Vertices of $U_{\mathcal{A}}$ are the tuples $v = (q, \lambda_1, \lambda_3, \dots, \lambda_{d+1}) \in Q \times \kappa^{d/2+1}$. We associate to each such vertex the *extended tuple*

$$\mathsf{ext}(v) = ([q]_0, \lambda_1, [q]_2, \lambda_3, \dots, [q]_{d-1}, \lambda_{d+1}).$$

We use it to define the total order: $v \leq v'$ if $\mathsf{ext}(v)$ is smaller than $\mathsf{ext}(v')$ for the lexicographic order. This is therefore a well-order. Edges in $U_{\mathcal{A}}$ are given by:

$$(q, \lambda_1, \dots, \lambda_{d+1}) \xrightarrow{a} (q', \lambda_1, \dots, \lambda_{d+1}) \iff \exists y \begin{cases} q \xrightarrow{a:y} q' \text{ in } \mathcal{A}, & \text{and} \\ (\lambda_1, \dots, \lambda_{d+1}) \xrightarrow{y} (\lambda'_1, \dots, \lambda_{d+1}) \text{ in } U_{\mathsf{parity}}. \end{cases}$$

Paths in $U_{\mathcal{A}}$ are well-behaved with respect to W, as stated below.

✦ **Lemma V.91.** *Let $(q, \lambda_1, \dots, \lambda_{d+1}) \overset{w}{\rightsquigarrow}$ be an infinite path in $U_{\mathcal{A}}$. Then, $w \in q^{-1}W$.*

Proof. Consider a path

$$(q^0, \lambda_1^0, \dots, \lambda_{d+1}^0) \xrightarrow{w_0} (q^1, \lambda_1^1, \dots, \lambda_{d+1}^1) \xrightarrow{w_1} \dots \text{ in } U_{\mathcal{A}}.$$

By definition, there are priorities $y_0, y_1, \dots$ such that

$$q^0 \xrightarrow{w_0:y_0} q^1 \xrightarrow{w_1:y_1} \dots \text{ in } \mathcal{A}, \quad \text{and} \quad (\lambda_1^0, \dots, \lambda_{d+1}^0) \xrightarrow{y_0} (\lambda_1^1, \dots, \lambda_{d+1}^1) \xrightarrow{y_1} \dots \text{ in } U_{\mathsf{parity}}.$$

Since vertices in U_{parity} satisfy the parity objective, $\liminf(y_0 y_1 \dots)$ is even, thus the above run in $\mathcal{A}$ is accepting, and so $w_0 w_1 \dots \in (q^0)^{-1}W$. ◀

Monotonicity

Monotonicity of $U_{\mathcal{A}}$ follows from the structural assumptions over $\mathcal{A}$.

✦ **Lemma V.92.** *The graph $U_{\mathcal{A}}$ is monotone.*

Proof. Let

$$(q, \lambda_1, \dots, \lambda_{d+1}) \xrightarrow{a} (q', \lambda'_1, \dots, \lambda'_{d+1}) > (q'', \lambda''_1, \dots, \lambda''_{d+1}) \text{ in } U_{\mathcal{A}}.$$

We aim to prove that $(q, \lambda_1, \dots, \lambda_{d+1}) \xrightarrow{a} (q'', \lambda''_1, \dots, \lambda''_{d+1})$ in $U_{\mathcal{A}}$. By definition of the transitions of $U_{\mathcal{A}}$, there is a priority y such that $q \xrightarrow{a:y} q'$ in $\mathcal{A}$ and $(\lambda_0, \dots, \lambda_{d+1}) \xrightarrow{y} (\lambda'_0, \dots, \lambda'_{d+1})$ in U_{parity}. We remark that, by definition of the order in $U_{\mathcal{A}}$, we have that $([q'']_0, \lambda''_1, \dots, \lambda''_{y-1}, [q'']_y) \leq ([q']_0, \lambda'_1, \dots, \lambda'_{y-1}, [q']_y)$ (for y even, similar if y odd). We distinguish four cases:

- If y is even and $([q']_0, \lambda'_1, \dots, \lambda'_{y-1}, [q']_y) = ([q'']_0, \lambda''_1, \dots, \lambda''_{y-1}, [q'']_y)$. Then in $\mathcal{A}$, $q \xrightarrow{a:y} q' \xrightarrow{\varepsilon:y+1} q''$ thus $q \xrightarrow{a:y} q''$, and in $U_{\mathcal{A}}$, $(\lambda_1, \dots, \lambda_{y-1}) \geq (\lambda'_1, \dots, \lambda'_{y-1}) = (\lambda''_1, \dots, \lambda''_{y-1})$, which concludes.
- If y is odd and $([q']_0, \lambda'_1, \dots, [q']_{y-1}, \lambda'_y) = ([q'']_0, \lambda''_1, \dots, [q'']_{y-1}, \lambda''_y)$. Then, in $\mathcal{A}$, $q \xrightarrow{a:y} q' \xrightarrow{\varepsilon:y} q''$ thus $q \xrightarrow{a:y} q''$, and in $U_{\mathcal{A}}$, $(\lambda_1, \dots, \lambda_y) > (\lambda'_1, \dots, \lambda'_y) = (\lambda''_1, \dots, \lambda''_y)$ which concludes.

- If for some even $x \leq y$ it holds that $([q']_0, \lambda'_1, [q']_2, \ldots, \lambda'_{x-1}) = ([q'']_0, \lambda''_1, [q']_2, \ldots, \lambda''_{x-1})$ and $[q'']_x < [q']_x$. Then in $\mathcal{A}$, $q \xrightarrow{a:y} q' \xrightarrow{\varepsilon:x} q''$ thus $q \xrightarrow{a:x} q''$ and in $U_{\mathcal{A}}$, $(\lambda_1, \ldots, \lambda_{x-1}) \geq (\lambda'_1, \ldots, \lambda'_{x-1}) = (\lambda''_1, \ldots, \lambda''_{x-1})$ thus $(\lambda_1, \ldots, \lambda_{d+1}) \xrightarrow{x} (\lambda''_1, \ldots, \lambda''_{d+1})$ which concludes.
- If for some even $x < y$ it holds that $([q']_0, \lambda'_1, \ldots, [q']_x) = ([q'']_0, \lambda''_1, \ldots, [q'']_x)$ and $\lambda'_{x+1} > \lambda''_{x+1}$. Then, in $\mathcal{A}$, $q \xrightarrow{a:y} q' \xrightarrow{\varepsilon:x+1} q''$ thus $q \xrightarrow{a:x+1} q''$ and in $U_{\mathcal{A}}$, $(\lambda_1, \ldots, \lambda_{x+1}) \geq (\lambda'_1, \ldots, \lambda'_{x+1}) > (\lambda''_1, \ldots, \lambda''_{x+1})$ thus $(\lambda_1, \ldots, \lambda_{d+1}) \xrightarrow{x+1} (\lambda''_1, \ldots, \lambda''_{d+1})$ which concludes.

The other implication $v > v' \xrightarrow{a} v'' \implies v \xrightarrow{a} v''$ in $U_{\mathcal{A}}$ follows exactly the same lines. ◀

Universality of $U_{\mathcal{A}}$

To prove Proposition V.88, there remains to establish universality of $U_{\mathcal{A}}$, which follows easily from history-determinism of $\mathcal{A}$ and universality of U_{parity}.

✦ **Lemma V.93.** *The graph $U_{\mathcal{A}}^{\top}$ is (κ, W)-universal.*

Proof. We show universality for trees of $U_{\mathcal{A}}$ and conclude by Lemma V.15. Let T be a Σ-tree of size $< \kappa$ that satisfies W. Let r be a sound resolver for $\mathcal{A}$. We define in a top-down fashion a labelling $\mathsf{r}_T : T \to Q$ such that, if $w_1 w_2 \ldots w_k$ is the labelling of the path from the root to a vertex t, then $\mathsf{r}_T(t)$ is the target state of the run induced by r in $\mathcal{A}$. In particular, $t \xrightarrow{a} t'$ in T implies that $\mathsf{r}_T(t) \xrightarrow{a:x} \mathsf{r}_T(t')$ in $\mathcal{A}$ for some priority x, and, on each infinite branch $t_0 \xrightarrow{a_0} t_1 \xrightarrow{a_1} \ldots$, the run $\mathsf{r}_T(t_0) \xrightarrow{a_0:x_0} \mathsf{r}_T(t_1) \xrightarrow{a_1:x_1} \ldots$ is accepting in $\mathcal{A}$. Stated differently, the $[0, d+1]$-tree T_{parity} obtained from T by replacing each edge $t \xrightarrow{a} t'$ with the corresponding edge $t \xrightarrow{x} t'$ such that $\mathsf{r}_T(t) \xrightarrow{a:x} \mathsf{r}_T(t')$, satisfies the parity objective.

By $(\kappa, \mathsf{parity})$-universality of U_{parity}, there exists a morphism $\phi_{\mathsf{parity}} : T_{\mathsf{parity}} \to U_{\mathsf{parity}}$. As T_{parity} has the same set of vertices than T, ϕ_{parity} defines a mapping from T to U_{parity}. We consider the product mapping $\phi = \mathsf{r}_T \times \phi_{\mathsf{parity}} : T \to U_{\mathcal{A}}$ that sends $t \mapsto (\mathsf{r}_T(t), \phi_{\mathsf{parity}}(t))$. It defines a morphism, as for any edge $t \xrightarrow{a} t'$ in T it holds that, for some x, $\mathsf{r}_T(t) \xrightarrow{a:x} \mathsf{r}_T(t')$ in $\mathcal{A}$ and $\phi_{\mathsf{parity}}(t) \xrightarrow{x} \phi_{\mathsf{parity}}(t')$ in $U_{\mathcal{A}}$. ◀

This completes the proofs of the implications (2) $\implies$ (3) $\implies$ (4) from Theorem V.1, providing a characterisation of half-positionality for ω-regular languages.

5 Bipositionality of all objectives

In this section we provide a characterisation of all bipositional objectives, without ω-regularity or prefix-independence assumptions. This characterisation extends the result of Colcombet and Niwiński [CN06], who showed that the only prefix-independent bipositional objective (over all game graphs) is the parity objective. Recently, Bouyer, Randour and Vandenhove [BRV23] generalised that result in an orthogonal direction: they proved that the only objectives for which both players can play optimally using finite chromatic memory are ω-regular objectives.

5.1 Characterisation of bipositionality and consequences

We say that an objective $W \subseteq \Sigma^\omega$ is *bi-progress consistent* if both W and its complement are progress consistent, that is, if it satisfies that for all residual class $[u]$ and finite word $w \in \Sigma^*$:

- $[u] < [uw] \implies uw^\omega \in W$, and
- $[uw] < [u] \implies uw^\omega \notin W$.

Theorem V.6 (Characterisation of bipositionality)**.**

An objective $W \subseteq \Sigma^\omega$ is bipositional (over all games) if and only if:

1. W has a finite number of residuals, totally ordered by inclusion, and
2. W is bi-progress consistent, and
3. W can be recognised by a parity automaton on top of the automaton of residuals.

This characterisation only holds for infinite games, as there are non ω-regular objectives that are bipositional over finite games, as, for example, energy objectives [Bou+08] and their generalisation [Koz22a]. However, we deduce from Theorem V.3 that in the case of ω-regular objectives these conditions do also characterise bipositionality over finite games.

Corollary V.94 (Bipositionality over finite games for ω-regular objectives)**.**

An ω-regular objective $W \subseteq \Sigma^\omega$ is bipositional over finite games if and only if it satisfies the three conditions from Theorem V.6.

Consequences: Lifts and decidability

For ω-regular objectives, we can directly lift the corollaries of Theorem V.1 obtained for half-positionality to bipositionality. For non-ω-regular ones, the finite-to-infinite lift does not hold, as commented above. On the other hand, a combination with a recent result from Bouyer, Randour and Vandenhove [BRV23, Theorem 3.8] implies that the 1-to-2-player lift holds for any objective.

We say that an objective $W \subseteq \Sigma^\omega$ is *bipositional over (finite) one-player games* if W is half-positional over (finite) Eve-games and its complement $\overline{W}$ is half-positional over (finite) Adam-games.

Corollary V.95 (1-to-2 player lift of bipositionality)**.**

An objective $W \subseteq \Sigma^\omega$ is bipositional (over all games) if and only if it is bipositional over one-player games.

Corollary V.96 (Finite-to-infinite lift of bipositionality for ω-regular objectives)**.**

An ω-regular objective $W \subseteq \Sigma^\omega$ is bipositional (over all games) if and only if it is bipositional over finite one-player games.

We obtain decidability for bipositionality in polynomial time from its counterpart in the case of half-positionality (Theorem V.2). We observe that Theorem V.6 provides a more efficient way to check bipositionality.

Corollary V.97 (Decidability of bipositionality)**.**

Given a deterministic parity automaton $\mathcal{A}$, we can decide in polynomial time whether $\mathcal{L}(\mathcal{A})$ is bipositional.

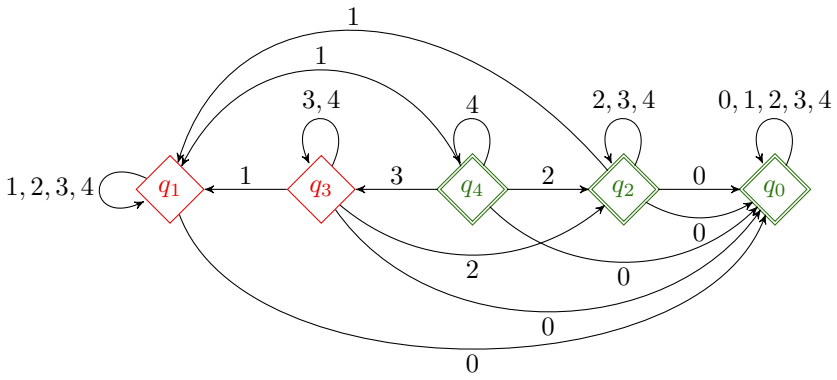
An example

Example V.98 (Parity over occurrences)**.**

We let $\Sigma = [0, d]$ and let $W_{\mathsf{OccParity}}$ be the language of words such that the minimal priority appearing on them is even:

$$W_{\mathsf{OccParity}} = \{w \in [0, d]^\omega \mid \min(w) \text{ is even}\}.$$

An automaton recognising W is depicted in Figure 42. It has one state per residual, which are totally ordered, and it is immediate to check that it is bi-progress consistent. Therefore, $W_{\mathsf{OccParity}}$ is a bipositional objective.



✦ **Figure 42.** Automaton recognising $W_{\mathsf{OccParity}}$, for $d = 4$. The initial state is q_4. This automaton is in fact a weak automaton: runs that finally end in an even state are accepting, and those ending in an odd state are rejecting.

Some more complex examples can be generated by, for example, adding some output priorities to the automaton above without breaking the bi-progress consistency condition. However, the combination of being recognisable by the automaton of residuals with bi-progress consistency greatly restricts the possibilities of generating examples of bipositional languages. We wonder whether a more precise characterisation of languages satisfying these three properties can be obtained.

5.2 Proof of the characterisation

Necessity of the conditions

The necessity of the total order over the residuals of W is given by Lemma V.21, and the necessity of bi-progress consistency is given by Lemma V.26. The following

lemma provides the necessity of the last condition of Theorem V.6. It can be obtained by instantiating the first item of [BRV23, Theorem 3.6] for the case of bipositional objectives.

Lemma V.99 ([BRV23, Theorem 3.6])**.**

If $W \subseteq \Sigma^\omega$ is bipositional over one-player games, then W is ω-regular and can be recognised by a parity automaton on top of the automaton of residuals.

Sufficiency of the conditions

The sufficiency of conditions of Theorem V.6 can be shown by providing well-ordered monotone universal graphs for W and its complement. An even simpler option – now that we have already done such construction for half-positional objectives – is to provide fully progress consistent signature automata recognising these languages, and then use the characterisation of half-positionality given by Item (2) from Theorem V.1.

We show that the parity automaton on top of the automaton of residuals is a fully progress consistent signature automata. Since by hypothesis $\mathsf{Res}(W)$ is totally ordered, the order $\leq_0$ given by inclusion of residuals satisfies the first requirement of the definition of signature automaton. As the residual classes of this automaton are trivial, we can define all relations $\sim_x$ to be trivial too, so this automaton satisfies all the requirements to be a signature automaton. Moreover, as W is progress consistent and the $\sim_x$-classes of the automaton are trivial, it is also fully progress consistent. The argument is symmetric for $\overline{W}$.

6 Half-positionality of closed and open objectives

6.1 Closed objectives

We recall that an objective W is closed if

$$W = \mathsf{Safety}(L) = \{w \mid w \text{ does not contain any prefix in } L\},$$

for some language of finite words L. Half-positional closed objectives were first characterised by Colcombet, Fijalkow and Horn [CFH14], as those that have a totally ordered set of residuals. However, as they aleady showed, this characterisation only holds for finite branching game graphs. This fact tells us that we cannot hope to have a finite-to-infinite lift for general closed objective (as the one presented for ω-regular ones in Corollary V.3).

We now give a characterisation of half-positionality over all game graphs for closed objectives. Namely, a closed objective W is half-positional if and only if $\mathsf{Res}(W)$ is well-ordered by inclusion (Theorem V.7). Moreover, we prove (Lemma V.102) that the well-foundedness of $\mathsf{Res}(W)$ is a necessary condition for finite memory determinacy of any objective.

6.1.1 Well-foundedness of residuals

Next example, taken from [CFH14], shows that total order over residuals does not suffice to ensure half-positionality of arbitrary closed objectives.

Example V.100 (Outbidding game [CFH14]: Total order does not suffice)**.**

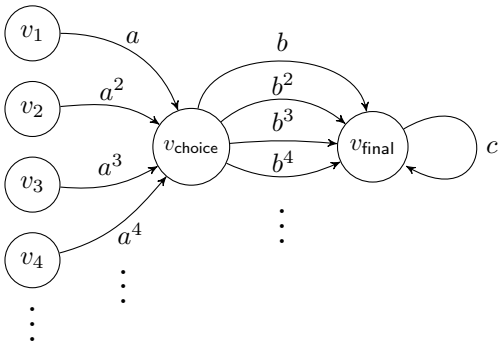
Let $\Sigma = \{a, b, c\}$ and L be the language of finite words:

$$L = \{w \in \Sigma^* \mid \text{ for some } u \in \Sigma^* \text{ with } |u|_a \leq |u|_b, uc \text{ is a prefix of } w\},$$

where $|u|_x$ is the number of occurrences of a in u. We consider the closed objective $W = \mathsf{Safety}(L)$. The residuals of W are totally ordered by inclusion:

$$\emptyset = c^{-1}W < \cdots < (a^n)^{-1}W < \cdots < a^{-1}W < \varepsilon^{-1}W < b^{-1}W < \cdots .$$

However, W is not half-positional, as witnessed by the game in Figure 43.



✦ **Figure 43.** Outbidding game. First, a sequence a^n is produced, for some $n \in \mathbb{N}$. In order to win, Eve needs to answer with b^m, with $m > n$. Therefore, she can win from any vertex in the game, but no positional strategy guarantees the victory from all states.

Lemma V.101 (Necessity of the well-order of residuals)**.**

Let $W \subseteq \Sigma^\omega$ be an objective that is half-positional over ε-free Eve-games. Then, $\mathsf{Res}(W)$ is well-ordered by inclusion.

We have already seen (Lemma V.21), that if W is half-positional, then $\mathsf{Res}(W)$ is totally ordered. We need to prove that $\mathsf{Res}(W)$ is well-founded. In fact, we show a much stronger result: if the memory requirements of W are finite, then $\mathsf{Res}(W)$ is well-founded.

Lemma V.102 (Well-foundedness of residuals necessary for finite memory)**.**

Let $W \subseteq \Sigma^\omega$ be an objective such that Eve can play optimally with finite memory over ε-free Eve-games. Then, $\mathsf{Res}(W)$ is well-founded (for the order given by inclusion of residuals).

Proof. Suppose by contradiction that there is an infinite strictly decreasing sequence of residuals:

$$u_1{}^{-1}W \supsetneq u_2{}^{-1}W \supsetneq \cdots , \; u_i \in \Sigma^*.$$

(We suppose without loss of generality that $\varepsilon \neq u_i$ for all i.) Let $w_i \in \Sigma^\omega$ such that $w_i \in u_i{}^{-1}W \setminus u_{i+1}{}^{-1}W$. We consider the game – similar to the outbidding game from Figure 43 – in which a word u_i labels a path from a vertex v_i to v_{choice}, for each i. From this latter vertex, Eve can choose a between paths labelled by $\{w_i \mid i \in \mathbb{N}\}$. Eve can

win be answering w_i to u_i. However, any finite memory strategy will only consider a finite number of responses $w_{j_1}, \dots, w_{j_n}$. Therefore, such a strategy is losing from v_K for $K > \max\{j_t \mid 1 \leq t \leq n\}$. ◀

6.1.2 Characterisation for closed objectives

Theorem V.7 (Half-positional closed objectives)**.**

Let $W \subseteq \Sigma^\omega$ be a closed objective. Then, W is half-positional (over all game graphs) if and only if $\mathsf{Res}(W)$ is well-ordered by inclusion.

Proof. We have already shown that this condition is necessary. To prove sufficiency, we give, for each cardinal κ, a (κ, W)-universal well-ordered monotone graph. We conclude by Proposition V.14.

Let U be the Σ-graph that has as vertices $\mathsf{Res}(W) \setminus \{\emptyset\}$, ordered by inclusion. By hypothesis, this is a well-order. For each $a \in \Sigma$, we let

$$u^{-1}W \xrightarrow{a} u'^{-1}W \quad \text{iff} \quad u'^{-1}W \leq (ua)^{-1}W.$$

We note that if ua is already losing ($ua^{-1}W = \emptyset$), then transition $u^{-1}W \xrightarrow{a} u'^{-1}W$ does not appear in U. By Lemma V.7, graph U is monotone. The hypothesis of closeness of W is fundamentally used in next claim.

✧ Claim V.102.1. *A vertex $u^{-1}W$ of U satisfies W if and only if $u^{-1}W \subseteq \varepsilon^{-1}W = W$.*

Proof. Let $L \subseteq \Sigma^*$ such that $W = \mathsf{Safety}(L)$. Let $u_1{}^{-1}W \xrightarrow{a_1} u_2{}^{-1}W \xrightarrow{a_2} \dots$ be a path in U from $u_1 = u$. By induction we obtain $\emptyset \neq u_i{}^{-1}W \subseteq (u_1 a_1 \dots a_{i-1})^{-1}W$. Therefore, for all i, $ua_1 \dots a_i \notin L$, so, by definition of W, the infinite word $ua_1a_2\dots$ belongs to W. ◁

We show that U is (κ, W)-universal for trees, for every cardinal κ, and conclude by Lemma V.15. Let T be a Σ-tree which satisfies W. For each node $t \in T$, let $\phi(t) = u_t{}^{-1}W$ be the minimal residual such that t satisfies $u_t{}^{-1}W$. In particular, for the root t_0, $\phi(t_0)$ satisfies W by the previous claim. We claim that ϕ is a morphism. Indeed, if $t \xrightarrow{a} t'$ in T and t satisfies $u^{-1}W$, then t' satisfies $(ua)^{-1}W$. Therefore, $u'_t{}^{-1}W \leq (u_t a)^{-1}W$, so $\phi(t) = u_t{}^{-1}W \xrightarrow{a} u_{t'}{}^{-1}W = \phi(t')$ is an edge in U. ◀

6.2 Open objectives

We recall that an objective W is open if

$$W = \mathsf{Reach}(L) = \{w \mid w \text{ contains some prefix in } L\},$$

for some $L \subseteq \Sigma^*$.

6.2.1 Reset-stability

In Section V.3.2, we showed that half-positional ω-regular open objectives are exactly those with residuals totally ordered and that are progress consistent. However, for non-ω-regular objectives, these conditions do not suffice, even if residuals are well-ordered.

Example V.103 (Progress consistency does not suffice)**.**

Let $\Sigma = \mathbb{N}$ and let W be the set of non strictly increasing sequences:

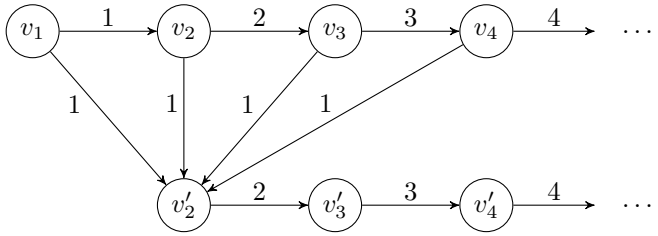
$$W = \{a_1 a_2 \cdots \in \mathbb{N}^\omega \mid a_{i+1} \leq a_i \text{ for some } i\}.$$

This objective is open, as $W = \mathsf{Reach}(\text{two consecutive non-increasing numbers})$. Its residuals are:

$$\varepsilon^{-1}W < 0^{-1}W < 1^{-1}W < 2^{-1}W < \cdots < (00)^{-1}W = \Sigma^\omega.$$

Therefore, $\mathsf{Res}(W)$ is well-ordered. Moreover, W is progress consistent: any repetition of factors induces a non-strict inequality $<$ between consecutive letters.

However, we claim that W is not half-positional. Consider the game in Figure 44. Eve can win this game: no matter what is the vertex v_i chosen by Adam, she can first move one position to the right, producing i, and then go down producing letter 1. This ensures two consecutive non-increasing numbers. However, she cannot win positionally. Indeed, if such a strategy tells her to go always to the left, the sequence produced will be strictly increasing. If she choses to go down in vertex v_i, Adam can win by initialising the play in that vertex.



✦ **Figure 44.** Game in which Eve wins if she produces a non-strictly increasing sequence of numbers. She can win from every vertex, but not positionally.

Definition V.104 (Reset-stability)**.**

We say that an objective $W \subseteq \Sigma^\omega$ is *reset-stable* if, for each sequence of finite words $u_1, u_2, u_3, \cdots \in \Sigma^+$ and each sequence of residuals ${s_0}^{-1}W, {s_1}^{-1}W, {s_2}^{-1}W, \cdots \in \mathsf{Res}(W)$:

$${s_i}^{-1}W < (s_{i-1}u_i)^{-1}W \text{ for all } i \geq 1 \implies u_1u_2u_3\cdots \in {s_0}^{-1}W.$$

An intuitive idea of reset-stability is the following. Consider the (potentially infinite) automaton of residuals of W, which inherits the order over residuals. Add to it all ε-transitions going backwards: $s^{-1}W \xrightarrow{\varepsilon} s'^{-1}W$ for $s'^{-1}W < s^{-1}W$. When a run takes an ε-transition, we say that it *makes a reset*. What reset-stability tells us is that any run making infinitely many resets must be accepting. The words $u_1, u_2, \ldots$ in the definition above correspond to fragments where no reset takes place, and ${s_i}^{-1}W$ is the residual where we land after the i^{th} reset. (See also the notion of 0-jumps in the proof of Lemma V.86).

✦ Remark V.105. *If W is reset-stable, it is progress consistent. The converse holds if $\mathsf{Res}(W)$ is finite and totally ordered.*

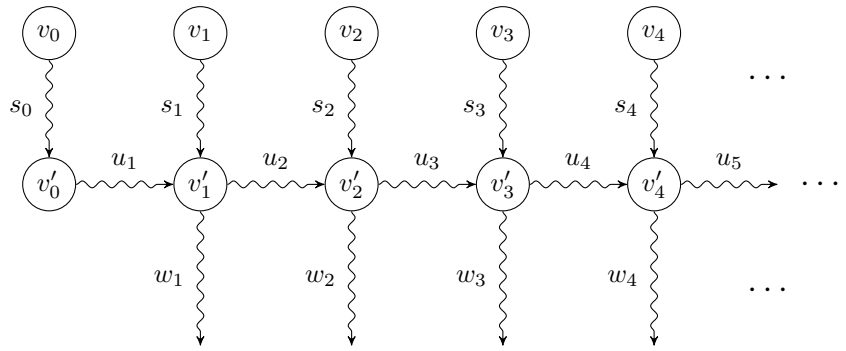
We note that all closed objectives are reset-stable.

Lemma V.106 (Necessity of reset-stability)**.**

Let $W \subseteq \Sigma^\omega$ be a half-positional objective over ε-free Eve-games. Then, W is reset-stable.

Proof. Suppose by contradiction that W is not reset-stable. That is, there are $u_1, u_2, \ldots \Sigma^+$ and ${s_0}^{-1}W, {s_1}^{-1}W \ldots$ such that ${s_i}^{-1}W < (s_{i-1}u_i)^{-1}W$, but $u_1u_2 \ldots \notin {s_0}^{-1}W$.

Let $w_i \in \Sigma^\omega$ such that $w_i \in (s_{i-1}u_i)^{-1}W \setminus {s_i}^{-1}W$. We consider the game pictured in Figure 45 (to ensure it to be ε-free, we just remove vertex v_i if $s_i = \varepsilon$).



✦ **Figure 45.** Game in which Eve wins if she produces a non-strictly increasing sequence of numbers. She can win from every v_i, but not positionally.

Eve can win $\mathcal{G}$ from any vertex v_i (and from v'_i if $s_i = \varepsilon$), as she can produce the word $s_iu_{i+1}w_{i+1}$, which belongs to W, as we have taken $w_{i+1} \in (s_iu_{i+1})^{-1}W$. However, we show that no positional strategy ensures to win from all these vertices. We distinguish two cases. If this strategy takes the path $v'_i \overset{u_{i+1}}{\rightsquigarrow} v'_{i+1}$ for all i, then it is not winning from v_0, as by hypothesis $u_1u_2 \ldots \notin {s_0}^{-1}W$. If on the contrary this strategy takes a path $v'_i \overset{w_i}{\rightsquigarrow}$, then it is not winning from v_i. ◀

6.2.2 Characterisation for open objectives

Theorem V.8 (Half-positional open objectives)**.**

Let $W \subseteq \Sigma^\omega$ be an open objective. Then, W is half-positional (over all game graphs) if and only if:

- $\mathsf{Res}(W)$ is well-ordered by inclusion, and
- W is reset-stable.

Proof. The necessity of the conditions has already been established in Lemmas V.102 and V.106. To prove the sufficiency, we give, for each cardinal κ, a well-ordered monotone graph that is (κ, W)-universal for trees, and conclude by Proposition V.14 and Lemma V.15.

We let U be the Σ-graph having as set of vertices $(\mathsf{Res}(W) \setminus \{\emptyset\}) \times \kappa$, ordered lexicographically. This graph is well-ordered, as by hypothesis so is $\mathsf{Res}(W)$. The edges are

given by:

$$(u^{-1}W, \lambda) \xrightarrow{a} (u'^{-1}W, \lambda') \quad \text{if} \quad \begin{cases} u'^{-1}W = (ua)^{-1}W \text{ and } \lambda' < \lambda, & \text{or} \\ u'^{-1}W < (ua)^{-1}W, & \text{or} \\ u^{-1}W = \Sigma^\omega. \end{cases}$$

By Lemma V.7, this graph is monotone. We show its (κ, W)-universality for trees. The key ingredient for this is next claim, which strongly relies in the reset-stability hypothesis. We let L be the language of finite words such that $W = \mathsf{Reach}(L)$.

✧ Claim V.106.1. *For each ordinal $\lambda < \kappa$ and each residual $u^{-1}W$, vertex $(u^{-1}W, \lambda)$ satisfies $u^{-1}W$ in U.*

Proof. Let $\rho = ({u_0}^{-1}W, \lambda_0) \xrightarrow{a_1} ({u_1}^{-1}W, \lambda_1) \xrightarrow{a_1} \cdots$ be an infinite path from $(u^{-1}W, \lambda)$ in U, and consider its projection over the (infinite) automaton of residuals $\mathcal{R}_W$. Whenever ρ takes a transition $({u_i}^{-1}W, \lambda_i) \xrightarrow{a}_i (ui + 1^{-1}W, \lambda_{i+1})$ with ${u_{i+1}}^{-1}W = (u_i a_i)^{-1}W$, this transition exists in $\mathcal{R}_W$. If ${u_{i+1}}^{-1}W < (u_i a_i)^{-1}W$, we say that this transition makes a reset. By induction, we obtain that ${u_i}^{-1}W \leq (ua_1 \dots a_i)^{-1}W$. We distinguish two cases: (1) If ρ makes infinitely many resets, then we conclude by reset-stability. (2) If ρ makes finitely many resets, then eventually $\lambda_{i+1} < \lambda_i$ for all i, unless ${u_i}^{-1}W = \Sigma^\omega$. We conclude that eventually ${u_i}^{-1}W = \Sigma^\omega \leq (ua_1 \dots a_i)^{-1}W$. Therefore, $ua_1 \dots a_i \in L$, so $ua_1a_2 \cdots \in W$. ◁

Let T be a Σ-tree whose root satisfies W. We give a morphism $\phi\colon T \to U$, which we decompose in $\phi_1\colon T \to \mathsf{Res}(W) \setminus \{\emptyset\}$ and $\phi_2\colon T \to \kappa$. For each $t \in T$, let u_t be the word labelling the path from the root t_0 to t. We let $\phi_1(t) = {u_t}^{-1}W$ for each t. We define ϕ_2 by transfinite induction. By hypothesis, each branch eventually contains vertices t such that $u_t \in L$ (that is, $u_t^{-1} = \Sigma^\omega$). For all these vertices, we let $\phi_2(t) = 0$. The tree obtained by removing these vertices, named T_1, does not have any infinite branch. For an ordinal $\lambda < \kappa$, let T_λ be the set of nodes for which we have not defined ϕ_2 at step λ of the induction. For each leaf t of T_λ, we let $\phi_2(t) = \lambda$.

This mapping has the two following properties:

- if $t \xrightarrow{a} t'$ in T, then $\phi_1(t') = (u_t a)^{-1}W$, and
- if $t \xrightarrow{a} t'$ in T, either $\phi_2(t') < \phi_2(t)$, or $u_{t'} \in L$.

This ensures that $\phi = (\phi_1, \phi_2)$ is a morphism, concluding the proof. ◀

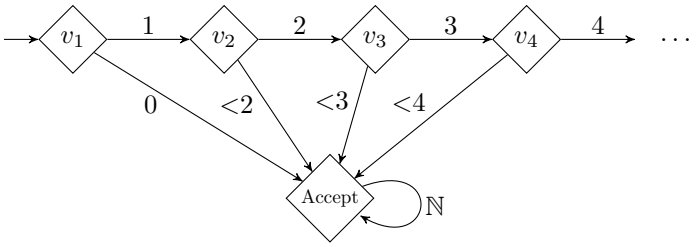
Example V.107 (Half-positional open objective)**.**

Let $\Sigma = \mathbb{N}$ and let W be the set of sequences that start by $123 \dots n$, and eventually decrease. Formally:

$$W = \{a_1 a_2 \cdots \in \mathbb{N}^\omega \mid \text{there is } j \text{ such that } a_i = i \text{ for } i < j \text{ and } a_j < j\}.$$

This objective is recognised by the infinite reachability automaton depicted in Figure 46. Its residuals are well-ordered and it is reset-stable, as after any reset we necessarily produce an word in W. Therefore, W is half-positional.

We remark that this objective is not bipositional, as $\mathsf{Res}(\overline{W})$ is not well-founded. This contrast with the case of ω-regular open objectives, for which all half-positional open objectives are bipositional (Corollary V.30).



✦ **Figure 46.** Game in which Eve wins if she produces a non-strictly increasing sequence of numbers. She can win, but not positionally.

6.3 1-to-2-player lift and addition of neutral letters

Corollary V.108 (1-to-2-player lift for open and closed objectives)**.**

Let $W \subseteq \Sigma^\omega$ be an open or closed objective. If W is half-positional over ε-free Eve-games, then W is half-positional over all game graphs.

We also obtain from our proofs that the Neutral letter conjecture (Conjecture V.3) holds for open and closed objectives.

Corollary V.109 (Closure under addition of neutral letters)**.**

Let $W \subseteq \Sigma^\omega$ be an open or closed objective. If W is half-positional, then W^{ε} is half-positional.

Proof. In the proof of Theorems V.7 and V.8, we obtained the positionality of objectives by providing well-ordered monotone (κ, W)-universal graphs. Proposition V.20 allows us to conclude. ◀

We have therefore obtained the 1-to-2-player lift for ω-regular objectives, as well as open and closed ones. However, the 1-to-2-player lift does not hold for arbitrary objectives. A counter-example is discussed in the PhD of Pierre Vandenhove [Van23, p.236].[9] The objective considered there (first appearing in [Kop08]), is:

$$\mathbf{MP}^{\mathbb{Q}} = \{w \in \{0,1\}^\omega \mid \liminf_n (\sum_{i=0}^{n} w_i)/n \quad \text{is rational}\}.$$

We provide here a different example.

Proposition V.110 (No general 1-to-2-player lift)**.**

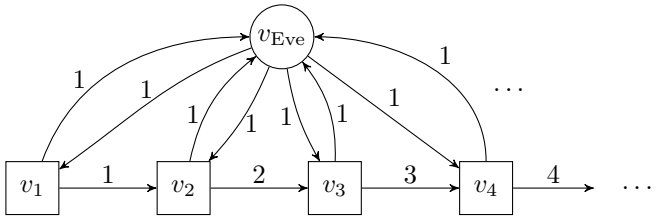
There is an objective $W \subseteq \Sigma^\omega$ that is half-positional over Eve-games, but is not half-positional over all game graphs.

Proof. Let Σ be any infinite alphabet, and let W be the following objective:

$$W_{\mathsf{fin}} = \{w \in \Sigma^\omega \mid |\mathsf{Inf}(w)| \text{ is finite}\}.$$

[9] In his PhD [Van23], Vandenhove discusses the 1-to-2-player lift for finite game graphs. The counter-example he gives also applies to infinite game graphs.

To show that it is not half-positional, we consider the game in Figure 47, in which Eve controls a single vertex, from which she can send the token to any vertex v_i controlled by Adam. We claim that Eve can win this game from every vertex by using the following



✦ **Figure 47.** Game in which Eve wins if only finitely many letters are produced infinitely often. She can win by sending the token further and further away, but she cannot win positionally.

strategy: she keeps track of the maximal index $i_{\max}$ such that the play has passed trough vertex $v_{i_{\max}}$. Whenever Adam sends back the token to v_{Eve}, she will go to vertex $v_{i_{\max}}$. This strategy ensures that only letter 1 will be produced infinitely often.

However, Eve cannot win using a finite memory strategy. Such a strategy will only consider finitely many edges $v_{\text{Eve}} \xrightarrow{1} v_i$. Let v_k be the maximal such vertex. Adam can win against such strategy by producing longer and longer paths, and then sending back the token to the vertex controlled by Eve: $v_i \xrightarrow{i(i+1)\ldots} v_{k+j} \xrightarrow{1} v_{\text{Eve}}$. In this way, all numbers greater that k will be produced infinitely often.

We show that W_{fin} is positional over Eve-games.

✧ Claim V.110.1. *For every Eve-game $\mathcal{G}$ with winning condition W_{fin} and every fixed vertex v_0, if Eve wins from v_0, she can win from v_0 using a positional strategy.*

Proof. Assume that Eve wins from v_0. A strategy from v_0 is just an infinite path from v. Consider such a path. If this path does not visit a same vertex twice, it is already a positional strategy. On the contrary, let v_k be the first vertex that repeats. Consider the first two occurrences of v_{rep}:

$$v_0 \xrightarrow{a_0} v_1 \xrightarrow{a_1} \cdots v_k \xrightarrow{a_k} \cdots v_{k+j} \xrightarrow{a_{k+j}} v_k.$$

Then, the positional strategy indicating to take the edge $v_i \xrightarrow{a_i} v_{i+1}$ for $i \leq k+j$ is winning. ◁

We use this claim to prove that W_{fin} is (uniformly) half-positional over Eve-games. Let $\mathcal{G}$ be an Eve-game with winning condition W_{fin}. By prefix-independence of W_{fin}, we can suppose without loss of generality that Eve wins from every vertex in $\mathcal{G}$. Let $v_1, v_2, \ldots$ a (potentially transfinite) enumeration of vertices in $\mathcal{G}$. We will define a positional strategy $\mathsf{strat}\colon V \to E$ by transfinite induction. At step λ, let $\mathcal{G}_\lambda$ be the game obtained by removing vertices for which strat has already been defined. Let i be the minimal index such that v_i appears in $\mathcal{G}_\lambda$. By the previous claim, Eve has a positional strategy in $\mathcal{G}_\lambda$ that wins from v_i. Let V_λ be the vertices reachable from v by using this strategy, and for $v \in V_\lambda$ let $\mathsf{strat}(v)$ be the edge indicated by such strategy. We let V'_λ be the vertices in $\mathcal{G}_\lambda \setminus V_\lambda$ from which Eve can reach V_λ, and fix a positional strategy doing so. We let $\mathsf{strat}(v')$ being given by this strategy for these vertices. It is immediate that strat is a positional strategy in $\mathcal{G}$ that wins from all vertices. ◀

Addendum: State-based vs transition-based acceptance

VI

Outline

Life is a journey, not a destination.
Ralph Waldo Emerson

Introduction

One particularity of this thesis is the use of transition-based automata and edge-coloured games throughout the entire manuscript (instead of state-based and vertex-coloured ones). The use of this formalism is not completely conventional; the vast majority of papers in the literature use games and automata with the acceptance condition defined over the states. The choice of transition-based models is far from being fortuitous: not a single result[1] of this tesis would hold when using state-based automata! I[2] strongly believe that many of these results have not been found much earlier precisely because the use of transition-based automata was not widespread.

In this addendum, we discuss the adequacy of transition-based automata and games for both the theoretical study of ω-regular languages and practical applications. We support the idea that the formalism defining acceptance conditions over transitions is preferable for all purposes. To provide evidence for this claim, we survey a collection of results that illustrate the advantages of using transition-based models.

[1] This statement might have been slightly exaggerated, but not so much really.

[2] If you have made it this far in the manuscript, we can consider ourselves almost intimates, so I will allow myself to write in a slightly more informal style and use the first person of singular whenever referring to personal thoughts and considerations.

Before diving into the list of problems, we give a brief and completely incomplete account on the use of state-based and transition-based automata in the literature.

Use of state-based and transition-based acceptance in the literature. Automata over infinite words were first introduced by Büchi in the 1960s [Bü62], using a formalism that put the acceptance condition over the states.[3] The tradition of employing state-based acceptance persisted in all subsequent classic foundational works on ω-automata: Muller's paper at the origin of the Muller condition [Mul63], Landweber's study of the complexity of ω-regular languages [Lan69], McNaughton's works on ω-regular expressions [McN66] and infinite games [McN93], Rabin's decidability result of S2S [Rab69], Wagner's paper introducing a hierarchy of complexity [Wag79], etc. Following this tradition, virtually all handbooks and surveys about automata over infinite words use state-based acceptance [Tho97, GTW02, PP04, Kup18, BCJ18, WS21, LT21, BK08]. To the best of my knowledge, the only exception to this is the recent book on games on graphs [Fij+23].

As far as I am aware of, the first occurrence of transition-based automata in the literature is due to Bertrand Le Saëc[4] [Saë90, SPW91, SL94]. He introduced transition-based Muller automata under the name of table-transition automata, and he already identified a crucial difference between state and transition-based automata: the residual automaton of a language L can recognise a richer class of languages when using transition-based acceptance. Nowadays, the use of transition-based automata is becoming more and more popular in research papers and tools – some of them are discussed in Section VI.2. The fundamental difference between the two models became undeniable following the groundbreaking results of Abu Radi and Kupferman [AK22] and Schewe [Sch20], showing, respectively, that the minimisation of history-deterministic transition-based coBüchi automata can be performed in polynomial time, and that the corresponding problem for state-based automata is NP-complete.

Why ~~is~~ was the use of state-based acceptance widespread? We may wonder why state-based automata were the ubiquitous model for more than 50 years. A first answer, and probably the most influential factor, is that ω-automata generalise automata over finite words, for which acceptance over states is indeed the natural choice. Some natural constructions of ω-automata build on automata over finite words, and for some of these, state-based acceptance may appear naturally.

One such a construction is the characterisation of languages recognised by deterministic Büchi automata as limits of languages of finite words [Lan69] (see also [Tho91, Remark 4.1]). A language $L \subseteq \Sigma^\omega$ can be recognised by a deterministic Büchi automaton if and only for some language of finite words $L_{\mathsf{fin}} \subseteq \Sigma^*$ we have:

$$L = \overrightarrow{L_{\mathsf{fin}}} = \{w \in \Sigma^\omega \mid w \text{ contains infinitely many prefixes in } L_{\mathsf{fin}}\}.$$

[3] The corroboration of this claim is reserved to a selected group of researchers capable of understanding Büchi's paper, a category I unfortunately do not fall within. Igor Walukiewicz and André Arnold confirm the use of state-based acceptance by Büchi, and point out that it can be observed, for instance, in the first line of the proof of Lemma 12 (page 8). In Büchi's 1969 paper with Landweber [BL69a] , the use of state-based acceptance is a bit simpler to appreciate in the definitions of Sup$Z$ and U, in the second page of the paper.

[4] Unfortunately, some of the results of the paper [Saë90] are incorrect. Namely, the condition from Theorem 5.1 does not characterise languages of parity index at most $[0, 1]$, as it is claimed. A correct version of this statement appears in Proposition II.110 in this thesis.

Building a state-based Büchi automaton from a deterministic automaton recognising L_{fin} is easy: we just need to interpret the final states of the automaton as the states of the Büchi condition.

Structure of the chapter. This is a short chapter, divided in three short sections. We start Section VI.1 by showing that we can easily switch between state and transition-based acceptance with at most a linear blow-up. However, we already notice a key difference: going from a transition-based automaton to a state-based one adding the minimal number of states is NP-hard. In the central section VI.2, we provide a (non-exhaustive) list of situations in which considering transition-based or state-based acceptance has a major impact, either in the complexity of decision problems or in actual theoretical results about ω-automata. The final section VI.3 is an attempt to explain the underlying reasons causing these striking differences between the two models.

Acknowledgements. I am deeply in debt with Thomas Colcombet for forcing me to use transition-based automata during my initial steps in research in my master's thesis. I also thank Géraud Sénizergues for pointing me to the works of Bertrand Le Saëc.

1 From states to transitions and vice-versa

At first sight, it seems that there is no great difference between state-based or transition-based acceptance: we can go from one model to the other with at most a linear blow-up. However, transition-based automata are always smaller, and going from a state-based automaton to a transition-based one in an optimal way is NP-hard, as stated in Proposition VI.3.[5]

Proposition VI.1 (From states to transitions)**.**

Let $\mathcal{S} = (Q, \Sigma, I, \Delta)$ be an automaton structure and let $\mathcal{A}_{\mathsf{st}} = (Q, \Sigma, I, \Gamma, \Delta, \mathsf{col}_{\mathsf{st}}, W)$ be a state-based automaton on top of $\mathcal{S}$. Then, there is a transition-based automaton on top of $\mathcal{S}$ equivalent to $\mathcal{A}_{\mathsf{st}}$, using W as acceptance set.

Proof. We define an acceptance condition $(\mathsf{col}_{\mathsf{trans}}, \Gamma, W)$ on top of $\mathcal{S}$ associating to a transition $e = q \xrightarrow{a} q'$ the colour $\mathsf{col}_{\mathsf{trans}}(e) = \mathsf{col}_{\mathsf{st}}(q)$. It is immediate to check that the obtained automaton is equivalent to $\mathcal{A}_{\mathsf{st}}$. ◀

Proposition VI.2 (From transitions to states)**.**

Let $\mathcal{A}_{\mathsf{trans}} = (Q, \Sigma, I, \Gamma, \Delta_{\mathsf{trans}}, W)$ be a transition-based automaton using a prefix-independent acceptance set $W \subseteq \Gamma^\omega$.[6] Then, there is a state-based automaton $\mathcal{A}_{\mathsf{st}}$ equivalent to $\mathcal{A}_{\mathsf{trans}}$ of size $|\mathcal{A}_{\mathsf{trans}}| \cdot |\Gamma|$ and using W as acceptance set.

[5] In this section, we only work with automata for the ease of the reader. All results can be generalised to transition systems as presented in Chapter II.

[6] There is a small annoying technical point here: the construction given in the proof only works if W is prefix-independent. Indeed, any run will start in a state (q, c_0), so it will output colour $c_0 \in \Gamma$ first no matter what is the input word. I see the fact that getting rid of this prefix-independent assumption is a hassle as further evidence that state-based acceptance is not well-suited.

Proof. Let $c_0 \in \Gamma$ be a colour picked arbitrarily. We define $\mathcal{A}_{\mathsf{st}} = (Q', \Sigma, I', \Gamma, \Delta', \mathsf{col}_{\mathsf{st}}, W)$ to be the automaton given by:

- $Q' = Q \times \Gamma$,
- the set of initial states is $I' = \{(q, c_0) \mid q \in I\}$,
- $(q, c) \xrightarrow{a} (q', c') \in \Delta'$ if $q \xrightarrow{a:c'} q' \in \Delta_{\mathsf{trans}}$,
- $\mathsf{col}_{\mathsf{st}}(q, c) = c$.

It is immediate to check that this automaton is equivalent to $\mathcal{A}_{\mathsf{trans}}$. ◀

The automaton $\mathcal{A}_{\mathsf{st}}$ from the previous proof satisfies the property that it admits a locally bijective morphism to $\mathcal{A}_{\mathsf{trans}}$, that is, it has been obtained by duplication of some states. However, in general, the factor $|\Gamma|$ is not optimal, we can obtain an equivalent automaton by duplicating only some of the states. Next proposition – which just restates Theorem II.15 – states that deciding what is the minimal number of states that we need to duplicate is NP-hard, already for the class of Büchi automata.

Proposition VI.3 (Optimal transformation from transitions to states)**.**

The following problem is NP-complete:

Input: A transition-based Büchi automaton $\mathcal{A}_{\mathsf{trans}}$ and a positive integer k.
Question: Is there a state-based Büchi automaton of size k admitting a locally bijective morphism to $\mathcal{A}_{\mathsf{trans}}$?

We recall that the reduction used to prove the previous result is the same as the one given by Schewe to show the NP-completeness of the minimisation of state-based Büchi automata [Sch10].

2 A compendium of problems

Propositions VI.1, VI.2 and VI.3 already indicate that transitions-based models are more succinct. However, the difference on the size is only linear, which might lead to the idea that there is no critical difference between these models. In this section we provide a list of problems for which the complexity greatly varies when using state-based or transition-based acceptance. We believe that this provides a solid support for our main thesis: transition-based automata are not only more succinct than state-based ones, but, most importantly, they are more canonical.

Minimisation of coBüchi automata

Consider the problem of the minimisation of coBüchi automata:

Problem:	Minimisation of coBüchi automata
Input:	A (history-)deterministic coBüchi automaton $\mathcal{A}$ and a positive integer k.
Question:	Is there a (history-)deterministic coBüchi automaton equivalent to $\mathcal{A}$ of size at most k?

This problem admits different variants; we can consider it for both deterministic or history-deterministic automata, and for state-based or transition-based ones. Surprisingly, the choice of the acceptance type has a great influence on its complexity. We remark that the importance of the choice of the model lies in the output, as we can convert the input from state-based to transition-based in polynomial time.

Proposition VI.4 ([Sch10, Sch20])**.**

The problem Minimisation of coBüchi automata is NP-complete for both deterministic and history-deterministic automata if state-based acceptance is used.

Proposition VI.5 ([AK22])**.**

The problem Minimisation of coBüchi automata can be solved in polynomial time for history-deterministic transition-based automata.

The complexity of Minimisation of coBüchi automata is open for deterministic transition-based automata.

Determinisation of Büchi automata

The complexity of the determinisation of Büchi automata is a fundamental problem in the theory of ω-automata, which has been studied since their introduction by Büchi [Bü62]. The first asymptotically optimal determinisation construction was due to Safra [Saf88], which transforms a Büchi automaton into a deterministic Rabin one. Later on, Piterman [Pit06] and Schewe [Sch09] further improved the construction, reducing the number of states of the final automaton. Schewe's construction transforms a Büchi automaton of size n into a deterministic Rabin automaton of size at most $\mathsf{sizeDet}(n)$, which is naturally equipped with a *transition-based* acceptance condition. In 2009, Colcombet and Zdanowski [CZ09] showed that the Piterman-Schewe construction is tight (up to 0 states!) as we precise now.

Proposition VI.6 (Optimality of the Piterman-Schewe construction [CZ09])**.**

There exists a family of Büchi automata $\mathcal{A}_n$ of size $|\mathcal{A}_n| = n$, such that a minimal transition-based deterministic Rabin automaton equivalent to $\mathcal{A}_n$ has size $\mathsf{sizeDet}(n)$.

We could obtain a state-based automaton from the Piterman-Schewe Rabin automaton by augmenting the number of states; but doing so we no longer have a matching lower bound. No such tight bounds are known for the determinisation of Büchi automata towards state-based automata.

The complementation and determinisation problems for Büchi and generalised Büchi automata with transition-based acceptance were further studied by Varghese in his PhD Thesis [Var14]. In the works of Schewe and Varghese [SV12, SV14], they point out the suitability of transition-based acceptance for the study of transformations of automata.

Optimal transformations of Muller automata

Chapter II was devoted to the study of transformations of Muller automata into parity and Rabin ones. In there, we obtained optimal transformations of automata: given a Muller automaton $\mathcal{A}$, we can build a minimal parity automaton admitting a locally bijective morphism to $\mathcal{A}$ (Theorem II.6). This result is based in the alternating cycle decomposition, a data structured obtained from the Muller automaton $\mathcal{A}$. Throughout the chapter, we used transition-based automata, which was not coincidental: as shown in Section II.8 (Theorem II.15) and recalled next, our results do not hold when using state-based automata.

Proposition VI.7 (Optimal transformations of Muller automata)**.**

Let $\mathcal{A}$ be a Muller automaton (transition or state-based) and let $\mathcal{ACD}_{\mathcal{A}}$ be its alternating cycle decomposition. With this input:

- We can compute in polynomial time a transition-based parity automaton admitting a locally bijective morphism to $\mathcal{A}$, and minimal amongst parity automata admitting a locally bijective morphism to $\mathcal{A}$.
- Deciding whether there is a state-based parity automaton of size $\leq k$ admitting a locally bijective morphism to $\mathcal{A}$ is NP-hard.

The problem for state-based output is already NP-hard for deterministic automata recognising languages of parity index $[0, 1]$ (that is, generalised Büchi automata). The alternating cycle decomposition of these automata can be computed in polynomial time (Proposition II.98).

Proposition VI.8 (Optimal transformations of generalised Büchi automata)**.**

Let $\mathcal{A}$ be a generalised Büchi automaton (transition or state-based):

- We can compute in polynomial time a transition-based Büchi automaton admitting a locally bijective morphism to $\mathcal{A}$, and minimal amongst Büchi automata admitting a locally bijective morphism to $\mathcal{A}$.
- Deciding whether there is a state-based Büchi automaton of size $\leq k$ admitting a locally bijective morphism to $\mathcal{A}$ is NP-hard.

Characterisations based in syntactic properties of automata

Given a class of ω-regular languages $\mathcal{C} \subseteq 2^{\Sigma^\omega}$ defined by some semantic property (e.g. half-positional languages, Muller languages, languages corresponding to a variety of ω-semigroups, etc.) a natural question is:

What is the class of deterministic parity automata recognising languages in $\mathcal{C}$?

Where "deterministic" can be replaced by other models (history-deterministic, unambiguous, etc.) and "parity" by other types of acceptance condition.

We argue that transition-based acceptance is best suited to obtain such syntactic characterisations. The evidence we provide is mainly based in the results obtained in Chapters IV and V of this thesis.

Characterisation of positional ω-regular languages. In Chapter V, we have characterised both half-positional and bipositional ω-regular languages, describing deterministic parity automata recognising them (Theorems V.1 and V.6). These characterisations use in multiple ways the fact that the automata we consider are transition-based. For instance, a necessary condition for the bipositionality of a language $L \subseteq \Sigma^\omega$ is that it has to be recognised by a parity automaton on top of the residual automaton. This condition is a completely different one if we use state-based acceptance; for instance, if L is prefix-independent, a state-based automaton on top of the residual automaton of L can only recognise the trivial languages $\emptyset$ and Σ^ω. In the case of half-positionality, the definition of a signature automaton is deeply rooted in the use of transition-based acceptance.

Correspondence memory-automata. In Chapter IV, we showed that, for a given Muller language $L \subseteq \Sigma^\omega$, chromatic memory structures for L exactly correspond to deterministic Rabin automata recognising L. Also, the general memory requirements of L correspond to the size of a minimal history-deterministic Rabin automata recognising L. These equivalences only hold if those Rabin automata are transition-based.

Positionality of objectives in vertex-coloured games

As noticed by Zielonka [Zie98] and Kopczyński [Kop08] (see also Section IV.1), whether we colour states or transitions in games modifies the memory requirements of objectives.[7] In particular, the class of half-positional objectives differs if we consider vertex-coloured or edge-coloured games. The following question arises: Is one of the two models preferable over the other? We argue that, at least for the study of memory and positionality, edge-coloured games is a more natural model. The fundamental reason behind this claim is that putting colours over the vertices gives some extra information to the player controlling the vertex that would not have otherwise.

Characterisation of bipositionality. In 2006, Colcombet and Niwiński provided an elegant characterisation of prefix-independent bipositional objectives [CN06] over infinite games. As indicated by the title of their paper, this characterisation only holds for edge-coloured games:

[7] These two notions differ uniquely if we consider ε-free games, as we can simulate an edge-coloured game by a vertex-coloured one in which we introduce intermediate uncoloured vertices (see [Kop08, Section 2.5]).

Proposition VI.9 ([CN06])**.**

A prefix-independent objective $L \subseteq \Sigma^\omega$ is bipositional over (ε-free) edge-coloured games if and only if L is a parity language.

However, there are prefix-independent objectives that are not parity languages that are bipositional over ε-free vertex-coloured games.

In Chapter V, we generalised this characterisation to non prefix-independent objectives (Theorem V.6). Similarly, our characterisation does not apply to ε-free vertex-coloured games.

Half-positionality over vertex-coloured games. As in the previous paragraph, the characterisation of half-positionality for ω-regular languages given in Chapter V (Theorem V.1) only holds for edge-coloured games: there are ω-regular objectives that are half-positional over ε-free vertex-coloured games, which cannot be recognised by a deterministic signature automaton. We refer the reader to Section IV.1 for examples.

In a recent breakthrough, Ohlmann characterised half-positionality of objectives *admitting a neutral letter* by means of universal graphs [Ohl23a]. His Neutral Letter Conjecture (Conjecture V.3) implies that this characterisation is complete, that is, it also applies to objectives without a neutral letter. Theorems V.1 and V.5 show that the Neutral Letter Conjecture holds for ω-regular languages and that the characterisation using universal graphs is complete for this class of objectives. However, Conjecture V.3 cannot hold when using vertex-coloured games, as shown by the next proposition. Moreover, the objective W of next proof does not admit any well-ordered monotone graph that is $(\aleph_0, W)$-universal, for any reasonable notion of "universality over vertex-coloured graphs".

Proposition VI.10 (No closure under addition of a neutral letter)**.**

There exists an ω-regular objective $W \subseteq \Sigma^\omega$ that is half-positional over vertex-coloured games such that the objective W^ε obtained by adding a neutral letter is not half-positional over vertex-coloured games.

Proof. Consider the alphabet $\Sigma = \{a, b\}$ and the Muller objective

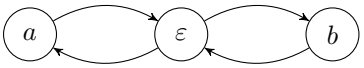
$$W = \{w \in \Sigma^\omega \mid \mathsf{Inf}(w) = \{a, b\}\}.$$

We showed in Example IV.10 that W is half-positional over vertex-coloured games (this also follows by Zielonka's characterisation [Zie98, Theorem 17]). However, by adding a neutral letter ε, we obtain the objective W^ε over $\Sigma' = \{a, b, \varepsilon\}$ given by:

$$W^\varepsilon = \{w \in \Sigma'^\omega \mid \mathsf{Inf}(w) \in \{\{\varepsilon\}, \{a, b\}, \{a, b, \varepsilon\}\}\}.$$

Objective W^ε is not half-positional over vertex-coloured games, as witnessed by the game in Figure 48. ◀

In Chapter V, we obtained a 1-to-2 player lift (Theorem V.3) and closure of half-positional prefix-independent ω-regular objectives under union (Theorem V.4). We do not know whether the analogous of these results hold for vertex-coloured games. Using Zielonka's characterisation [Zie98, Theorem 17] we can obtain a partial result: the class of Muller objectives that are half-positional over vertex-coloured games is closed under union.



✦ **Figure 48.** A vertex-coloured game $\mathcal{G}$ in which Eve can produced a and b infinitely often, but not using a positional strategy.

Practical applications

When using ω-automata for model checking or synthesis purposes, transition-based automata have been shown to perform better in practice [GL02], and the use of transition-based models is well-established in the tools used nowadays. The state-of-the-art libraries for ω-automata, Owl [KMS18] and Spot [DL+22], they both use transition-based automata by default. For instance, Owl use them internally, and only provides a converter from state-based to transition-based automata for interfacing with external tools. Regarding LTL synthesis tools, transition-based automata is the default model of tools such as Strix [LMS20, MS21b], `ltlsynt` [MC18], `ltl3tela` [Maj+19b], Rabinizer 4 [Kre+18] or `Delag` [MS17] – list which includes the top-ranked tools in the SyntComp competitions [Jac+22]. Other tools such as Acacia-Bonsai [CP23] do use state-based automata.

3 Where do all these differences come from? Some thoughts

Previous section contains a list of situation in which using transition-based acceptance is more advantageous, both for practical and theoretical reasons. The following question arises naturally:

> What are the fundamental differences between state-based and transition-based models that lead to such contrasting properties?

Role of states in automata. The fundamental question hidden behind the previous inquiry is the following:

> What is the role of a state in an ω-regular automaton?

Let me explain this in more detail. In the case of finite words, the role played by each state of a deterministic automaton recognising a language $L \subseteq \Sigma^*$ is well understood: a state q represents a residual of L. That is, when reading a word $u \in \Sigma^*$, the information we need to retain is what is the residual $u^{-1}L$. However, in the case of infinite words, this does not suffice. In general, an ω-regular language $L \subseteq \Sigma^\omega$ cannot be recognised by a Muller automaton on top of the automaton of residuals of L. This leads to one of the most fundamental questions about ω-automata: what is this "extra information" that we need to retain? In particular, this is the question at the core of the automata minimisation problem, studied in Chapter III. Of course, I do not have any answer to this question, but analysing the dichotomy state-based vs transition-based acceptance through these lenses might shed some light to help us understand the differences between the two models.

This analysis is by no means original, it is exactly what Le Saëc and Litovsky remarked 30 years ago [Saë90, SL94]: an automaton of residuals can recognise strictly more languages if the acceptance condition is put over transitions. For example, consider a prefix-independent language $L \subseteq \Sigma^\omega$. The automaton of residuals $\mathcal{R}_L$ has a single state, so L can be recognised by a Muller automaton on top of $\mathcal{R}_L$ if and only if L is trivial (it equals $\emptyset$ or Σ^ω). However, if we allow to define the acceptance over transitions, L can be recognised by a Muller automaton on top of $\mathcal{R}_L$ if and only if L is a Muller language, a much richer class.

The difference on the power of the automaton of residuals has a direct influence, for example, in our characterisations of bipositional languages (Theorem V.6), in which a necessary condition is that the language has to be recognised by a parity automaton on top of the automaton of residuals. The same remark applies to the characterisation of half-positional objectives recognised by deterministic Büchi automata (Proposition V.34).

The algebraic structure of automata. A fundamental difference between automata over finite and infinite words is that the latter ones can be seen rather as transducers reducing a target language to a simpler one. In this light, putting the output over the transitions seems more natural. One reason is that an elementary operation on automata is the composition of transitions: given two transitions $q_1 \xrightarrow{a} q_2$ and $q_2 \xrightarrow{b} q_3$, one should be able to define a transition $q_1 \xrightarrow{ab} q_3$. This new transition can only be defined in a sensible way if the output appears over transitions.

This composition operation is at the heart of the celebrated connection between automata and the algebraic theory of semigroups. In fact, one of LeSaëc's motivations for introducing transition-based automata was to obtain an algebraic proof of McNaughton's theorem for infinite words [SPW91].

Composition of coloured edges is also essential in the modern approach for solving and analysing infinite duration games based on universal graphs [Col+22, Ohl23a]. Two key concepts in the theory of universal graphs rely on composition of edges: monotonicity, and the technique of *saturation* (see [Col+22, Section 4.1], [CF18, Section 4], or [Ohl23a, Section 3.3]).

Infinite paths in automata. Another related idea (even more abstract and imprecise) is that, by placing the acceptance condition over the states, we overload them with a task that is not their primary role. In the case of finite words, to check if a word $w \in \Sigma^*$ is accepted by an automaton we look to the final state of the run over w when it ends. It makes then perfect sense to put on that state the burden of carrying the acceptance condition. However, in the case of infinite words, to check the acceptance of a run we need to look at the whole path in the automaton; the run does not end in a final state. States are intrinsically malfunctioning for identifying such paths:

> A path $v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} v_2 \xrightarrow{e_2} \dots$ in a graph is not uniquely identified by the sequence of vertices $v_0 v_1 v_2 \dots$. It is, however, identified by the sequence of edges $e_0 e_1 e_2 \dots$.

All in all, this collection of problems and considerations comes to say that, if we are solely interested in the asymptotic growth of the size of automata, the use of state-based or transition-based acceptance does not make any difference. However, if we aim

to understand their fundamental structure and obtain natural procedures to manipulate them, transition-based acceptance should be adopted.

Conclusions and perspectives

Outline

Chieftains must understand that the spirit of the law is greater than its letter.
~~*Attila the Hun*~~ *Mikołaj Bojańczyk*

1 Strategy complexity

The results presented in Section V.2 effectively address most open questions regarding half-positionality in the context of ω-regular languages. However, as in any other case, we might question the value of the characterisation presented in Theorem V.1 and how it truly contributes to our understanding of the problem.

On the one hand, Theorem V.1 fulfils many key requirements to be considered a satisfactory characterisation: it provides a polynomial-time procedure to check half-positionality, and its applicability has been demonstrated by solving many open questions.

Yet, it would be reasonable to seek a "better" characterisation. One drawback of our approach is its conceptual complexity and its exclusive focus on automata; the characterisation is based on syntactic and combinatorial properties of parity automata, rather than on intrinsic language-theoretical properties of the languages they recognise. In this respect, the insights gained about half-positionality are somewhat limited. Therefore, we believe that there is still room for improvement and for a deeper understanding of the class of half-positional ω-regular objectives.

We now discuss two research directions extending our results.

1.1 Positionality of $\mathcal{BC}(\Sigma_2^0)$ languages

A potential research direction consists in investigating half-positionality for broader classes of objectives. We initiated this path in Section V.6 by studying half-positionality of closed and open objectives. Going further in that direction, the goal is to develop characterisations for more complex objectives defined by topological properties, mainly, higher classes in the Borel hierarchy. A natural first step could be to look at objectives in Σ_2^0 and Π_2^0, which are, respectively, unions of closed objectives and intersections of open objectives. These classes admit automata-oriented definitions: Σ_2^0 are the objectives recognised by deterministic infinite Büchi automata, and Π_2^0 those recognised by infinite coBüchi automata [Skr13].

The class of objectives recognised by infinite deterministic parity automata coincides with $\mathcal{BC}(\Sigma_2^0)$: the class of boolean combinations of objectives in Σ_2^0 [Skr13] (this is a strict subclass of $\Delta_3^0 = \Sigma_3^0 \cap \Pi_3^0$). We hope to be able to give a characterisation for this class – as some of the constructions introduced in this work seems to generalise to automata with infinite states. We believe that some properties of half-positional ω-regular objectives could be lifted to $\mathcal{BC}(\Sigma_2^0)$ (or even to Δ_3). In particular, we conjecture that the 1-to-2 player lift (false in general, see Proposition V.110), holds for $\mathcal{BC}(\Sigma_2^0)$-objectives.

Conjecture VII.1 (1-to-2 player lift in $\mathcal{BC}(\Sigma_2^0)$)**.**

Let W be an objective in $\mathcal{BC}(\Sigma_2^0)$ that is half-positional over Eve-games. Then, it is half-positional over all games.

Decidability problems lose their relevance when dealing with classes of objectives defined by topological properties. Therefore, the question of what is a satisfactory characterisation becomes even harder to answer in this context – especially when taking into account that we already have Ohlmann's characterisation based on universal graphs [Ohl23a]. An important step forward would involve proving Conjecture VII.1, as well as Kopczyński's conjecture (Conjecture V.2) or Ohlmann's neutral letter conjecture (Conjecture V.3) for the class $\mathcal{BC}(\Sigma_2^0)$.

1.2 Memory requirements of ω-regular languages

An orthogonal research direction would be to maintain the focus on ω-regular languages, but attempt to characterise their memory requirements rather than just their positionality. A notable effort has already been made in this direction [DJW97, Kop08, CFH14, Bou+23, Cas22, CCL22], however, only characterisations for fairly simple classes of languages are known (Muller [DJW97] and closed languages [CFH14]). One difficulty is the multiplicity of models of memory: general, chromatic, ε-memory... (see Section IV.1 for details). Recently, Bouyer, Fijalkow, Randour and Vandenhove showed that computing the chromatic memory requirements of ω-regular open and closed objectives is NP-complete [Bou+23]. This result, together with the NP-completeness of the computation of the chromatic memory requirements of Muller objectives (Theorem IV.2), indicates that chromatic memory might be an ill-suited model for these purposes. We believe that the most promising model to focus on in order to advance on this problem is ε-memory, as discussed in [CO23]. A first step required for moving forward involves characterising memory requirements for open objectives. Nevertheless, progress in understanding the

general memory for open objectives has remained elusive for more than 10 years [Fij11, CFH14].

The generalisation of the theory of monotone universal graphs to characterise memory requirements, presented in [CO23], offers a promising tool to establish tight upper bounds on memory requirements, which could be valuable to overcome impediments in the advancement of the study of memory for ω-regular objectives.

2 The structure of ω-automata

Most contributions of this thesis are based on a fine analysis of the structural properties of ω-automata. As already mentioned, Wagner played a pioneering role in studying these properties and deriving language-theoretical results from them [Wag79]. The introduction of the alternating cycle decomposition offers a fresh perspective into these considerations. We have used it to provide optimal transformations of automata, to obtain typeness results about Muller automata and to define a normal form for parity automata. Typeness results have proven crucial in the study of the minimisation of Rabin automata (Lemma III.7) and its relation with the chromatic memory (Lemma IV.12). The normal form of parity automata has made possible the study of half-positionality in Chapter V. Moreover, our findings reinforce a shift occurring in research in automata theory nowadays, highlighting the importance of history-determinism and transition-based automata for obtaining theoretical results.

Many questions remain open. The most intriguing ones are probably those concerning the minimisation of parity automata. Our results about the minimisation of generalised (co)Büchi automata (mainly Theorem III.2 and Conjectures III.3 and III.4, which are not really conjectures, see footnote 2) make us think that the case of history-deterministic coBüchi automata might be a very special case which admits tractable minimisation, while deterministic models, or the use of Büchi acceptance might turn the problem NP-hard. For this reason, we believe that the minimisation of parity, or even Büchi automata is NP-hard, both for deterministic and history-deterministic models.

Conjecture VII.2 (Minimisation of Büchi automata (question raised in [AK23]))**.**

The minimisation of deterministic and history-deterministic Büchi automata is an NP-complete problem.

Nevertheless, we have discussed why establishing the NP-hardness of this problem may be challenging. Specifically, we have shown that minimising parity automata recognising Muller languages is tractable (Theorem III.4). We discuss another class of languages for which we believe the minimisation of parity automaton to be tractable: half-positional languages.

2.1 Minimisation of parity automata recognising positional languages

In the proof of Theorem V.1 (Section V.4.2), we have introduced many transformations of automata that aim to remove redundant states and to put automata in some form that make them suitable for our study. This kind of transformations exhibit a flavour similar to what one might expect from a minimisation algorithm for parity automata. In fact, one of the main techniques we used (safe centralisation of automata) is just a generalisation

of the minimisation algorithm for history-deterministic coBüchi automata introduced by Abu Radi and Kupferman [AK22].

From our characterisation (Theorem V.6) we obtain that we can minimise automata recognising bipositional languages in polynomial time. Indeed, a necessary condition for a language L to be bipositional is that it must be recognised by a parity automaton on top of the automaton of residuals.

Proposition VII.1.

Deterministic (resp. history-deterministic) parity automata recognising bipositional languages can be minimised in polynomial time.

By the characterisations from Propositions V.34 and V.53, we obtain that we can also minimise (history-) deterministic Büchi and coBüchi automata recognising half-positional languages in polynomial time.

Proposition VII.2.

Deterministic (resp. history-deterministic) Büchi and coBüchi automata recognising half-positional languages can be minimised in polynomial time.

We conjecture that our methods may lead to similar results in the more general case of parity automata.

Conjecture VII.3.

Deterministic and history-deterministic parity automata recognising half-positional languages can be minimised in polynomial time. Moreover, history-deterministic parity automata for this class of languages are not more succinct than deterministic ones.

We believe that we can even describe the minimal automaton that we should obtain, by refining the definition of reduced signature automaton given in Section V.4.1. We say that a signature automaton is *strongly reduced* if it is a structured signature automaton and, for every state q and even priority x, if no transition producing a priority $\geq x$ leaves q, then the $\sim_x$-class of q is trivial: $[q]_x = \{q\}$.

Conjecture VII.4.

A strongly reduced signature automaton recognising a language L has a minimal number of states amongst history-deterministic parity automata recognising L.

Moreover, if L is half-positional, it can be recognised by a deterministic strongly reduced signature automaton.

We are highly confident that this result holds. One should be able to prove that strongly reduced signature automata are minimal by generalising the methods from [AK22]. Nevertheless, such a proof runs the risk of being highly technical.

As a final consideration, we discuss the problem of the canonicity of the parity condition.

2.2 Canonicity of the parity condition

In the general introduction (page 20), we provided a list of properties satisfied by the parity condition that make it suitable in both practical and theoretical contexts. These properties serve as indicators of the canonicity and naturalness of the parity condition. Some of the results in this thesis further support this idea:

- The minimal number of colours required to recognise an ω-regular language using a deterministic Muller automaton are met with a parity automaton (Proposition II.117).
- The normal form of a parity automaton exhibits several nice properties that directly relate to the complexity of the languages they recognise (Section II.7.2).
- The obtained characterisations of half-positionality and bipositionality rely on the structure of parity automata (Theorems V.1 and V.6).

Yet, we have not been able to provide a formal statement for our claim of simplicity of the parity condition.

Question VII.5 (Canonicity of the parity condition)**.**

Formalise the statement that parity languages form the simplest ω-regular complete family of languages (that is, a family of languages that can be used to recognise all ω-regular languages with deterministic automata).

This question remains somewhat vague, and we have not yet precisely formulated the result we are seeking. Ideally, we aim to establish a statement of the form "parity languages form the only family of languages that satisfy some list of hypothesis while minimising a certain parameter". Some measures of complexity that could aid in formalising such a statement include:

- The topological complexity of languages.
- The size of ω-semigroups or a measure of their structural complexity (such as their Green relations).
- The size of automata.

Voilà voilà...

Other work

During the realisation of my PhD, I have actively engaged in various additional projects, some of which have resulted in publications, or are about to. Due to space constraints and with the objective of maintaining both the fluidity of presentation and the logical consistency of content, I have opted not to include these results in the manuscript. Below, you will find a brief summary outlining these related projects.

Universal graphs for the characterisation of memory requirements. In his PhD thesis [Ohl21, Ohl23a], Pierre Ohlmann characterised half-positionality by means of well-ordered monotone universal graphs. Together, we generalised the theory of universal graphs to capture the memory requirements of objectives (both general and chromatic). A paper presenting this characterisation, as well as its applicability and limitations, was accepted at ICALP 2023 [CO23] (for the much more comprehensive full version, see [CO22]).

Infinite lexicographic products of positional objectives. One important application of Ohlmann's characterisation of positionality is obtaining one of the very few known closure properties of half-positional objectives: they are closed under *finite lexicographic products* [Ohl23a]. He conjectured that they are moreover closed under infinite lexicographic products. In a collaboration with Pierre Ohlmann, Michał Skrzypczak and Igor Walukiewicz, we proved that this is indeed the case. This implies the positionality of very general and quite exotic objectives, such as parity objectives over infinite ordinals, setting an open question arising from the works of Grädel and Walukiewicz [GW06]. A submission is under preparation.

Fast value iteration for energy games. At the beginning of my PhD (and at the end of Pierre's), I joined Pierre Ohlmann in some of his in-depth studies of algorithms for solving parity and energy games. Together, we developed an algorithm for solving energy games – based on adding natural accelerations to value iteration algorithms – that had a very simple conceptual description and that performed very well in practice [CO21]. It turned out that (as pointed to us by Alexander Kozachinskiy) this was an alternative presentation of an already existing algorithm by Schewe [Sch08] (also presented slightly differently by Luttenberger [Lut08]). One of our contributions is to describe a symmetric variant of the algorithm with a very intriguing status: it performs extremely well in practice, but we have not been able to prove its termination in all cases.

Lower bounds for solving Rabin games. In a project originated at Autobóz 2023, jointly with Michał Pilipczuk and K. S. Thejaswini, we provided a fairly simple proof that, assuming the Exponential Time Hypothesis, there is no algorithm that solves Rabin games with n vertices and k Rabin pairs in time $2^{o(k \log k)} \cdot n^{\mathcal{O}(1)}$. This reproves

in a simpler manner a result from [Cal+22]. Our proof involves a reduction from a problem called Permutation SAT. This problem was brought to our attention by Marcin Pilipczuk and Uéverton S. Souza, who showed that it admits a similar lower bound. A conference submission is under review.

Appendices

Outline

A Some NP-complete problems

In this appendix, we define various NP-complete problems that are used in the thesis to obtain NP-hardness results.

An *undirected graph* is a pair $G = (V, E)$ consisting of a set of vertices V and a symmetric relation $E \subseteq V \times V$. We say that it is *simple* if for all $v \in V$, $(v, v) \notin E$. We say that it is *connected* if there is a path connecting any pair of vertices.

The chromatic number problem and 3-colorability

A *colouring* of a simple undirected graph is a mapping $c : V \to \Lambda$ such that $c(v) = c(v') \Rightarrow (v, v') \notin E$ for every pair of nodes $v, v' \in V$. We say that such a colouring has *size* $|\Lambda|$. The *chromatic number* of G is the minimal number k such that G has a colouring of size k. We denote it $\chi(G)$.

Problem: Chromatic number
Input: A simple, connected, undirected graph G and a positive integer k.
Question: $\chi(G) \leq k$?

Lemma A.1 ([Kar72])**.**
The problem Chromatic number is NP-complete.

This problem is already NP-complete if we only ask whether the chromatic number is no more than 3.

Problem: 3-colorability
Input: A simple, connected, undirected graph G.
Question: $\chi(G) \leq 3$?

Lemma A.2 ([Lov73])**.**
The problem 3-colorability is NP-complete.

Max Clique

A *clique* of an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ such that $(v', u) \in E$ for every $v', u' \in V'$.

Problem:	Max-Clique
Input:	A simple, connected, undirected graph G and a positive integer k.
Question:	Does G contain a clique of size k?

Lemma A.3 ([Kar72]).

The problem Max-Clique is NP-complete.

Vertex cover

A *cover* of an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ such that for every edge $(v, u) \in E$, either $v \in V'$ or $u \in V'$.

Problem:	Vertex Cover
Input:	A simple, connected, undirected graph G and a positive integer k.
Question:	Is there a cover of G of size k?

Lemma A.4 ([Kar72]).

The problem Vertex Cover is NP-complete.

B Transformations of automata: Missing details for Chapter II

B.1 Transformations for games

In this appendix, we include material showing how to adapt the concepts and constructions studied in Chapter II for the case of transformation of games, which require some technical adjustments.

B.1.1 Games suitable for transformations

As we have indicated throughout Chapter II, defining transformations not preserving determinism in the case of games poses certain formal challenges. This difficulties appear both when such transformations arise as the product $\mathcal{G} \ltimes \mathcal{A}$ of a game $\mathcal{G}$ by an non-deterministic automaton $\mathcal{A}$, or when they are witnessed by an HD mapping $\varphi\colon \mathcal{G} \to \mathcal{G}'$. The problem comes from the fact that the semantics of non-determinism in automata (or history-determinism of morphisms) are inherently asymmetric, and this asymmetry needs to be made compatible with the semantics of games. The choices we have made to overcome this technical difficulty are:

- Restrict transformations of games to games in a standard form, which we have called *games suitable for transformations*. Similar restrictions for games have already been considered in the literature [EJ91].
- Add a restriction to HD mappings in the case of games, introducing the notion of HD-for-games mapping.

The main motivation for the standard form of games that we propose comes from viewing games as *originating from logical formulas*. Indeed, an equivalent model for games can be given as follows: vertices in the game graph are not partitioned into Eve's and Adam's nodes, instead, we assign a boolean formula to each transition that determines an interaction between the two players. The outcome of this interaction is (1) the next vertex, and (2) the output colour of the acceptance condition. We can obtain a game of the kind we have defined in this thesis by unfolding the boolean formulas of the transitions. There is a natural way to standardize such games: putting the boolean formulas in disjunctive normal form (DNF). Then, the unfolding of a game with formulas in DNF yields a game in which the partition into Eve-Adam nodes induces a bipartite graph with a particular structure: first, Adam chooses an uncoloured transition leading to a vertex controlled by Eve (with only one ingoing transition), and then Eve picks a transition producing some output colour.

We recall that a game is suitable for transformations if it verifies that for every edge $e = v \to v'$, if v is controlled by Adam, then e is uncoloured ($\mathsf{col}(e) = \varepsilon$), $v' \in V_{\text{Eve}}$, and e is the only incoming edge to v' ($\mathsf{In}(v') = \{e\}$).

Games in this form have an asymmetric structure that makes them suitable for any type of transformation. As any pair of consecutive transitions are of the form $v \xrightarrow{\varepsilon} \tilde{v} \xrightarrow{c} v'$, with $\tilde{v} \in V_{\text{Eve}}$, we can force it so that if a decision needs to be made in a product, Eve is the one who makes it.

Lemma B.1.

For every game $\mathcal{G}$ with vertices V and edges E, there exists a game $\widetilde{\mathcal{G}}$ that is suitable

for transformations, of size $|\widetilde{\mathcal{G}}| = \mathcal{O}(|E|)$, and equivalent to $\mathcal{G}$ in the following sense: there is an injective function $f\colon V \to \widetilde{V}$ such that Eve wins $\mathcal{G}$ from v if and only if she wins $\widetilde{\mathcal{G}}$ from $f(v)$.

Proof. We define $\widetilde{\mathcal{G}}$ as follows. We let its set of vertices be $\widetilde{V} = V \cup E$. Vertices of the form $v \in V$ will correspond to vertices coming from $\mathcal{G}$, and vertices $e \in E$ will be intermediate vertices added to force the suitability for transformations property. We let $\widetilde{V}_{\text{Adam}} = V_{\text{Adam}}$ and $\widetilde{V}_{\text{Eve}} = V_{\text{Eve}} \cup E$. If $e = v \xrightarrow{c} v'$ is an edge in $\mathcal{G}$, we add the edges $v \xrightarrow{\varepsilon} e$ and $e \xrightarrow{c} v'$ to $\mathcal{G}'$. It is clear that $\widetilde{\mathcal{G}}$ is suitable for transformations and that Eve wins $\mathcal{G}$ from v if and only if she wins $\widetilde{\mathcal{G}}$ from v. ◀

B.1.2 History-deterministic-for-games mappings

In Section II.2 we remarked that the existence of an HD mappings between two games $\varphi\colon \mathcal{G} \to \mathcal{G}'$ do not suffice to guarantee that they have the same winner. In this section, we strengthen the notion of HD mappings in the case of games adding a minimal set of hypothesis to ensure this property.

In order to show that if Eve wins $\mathcal{G}'$ then she wins $\mathcal{G}$, we need a method to transfer strategies in $\mathcal{G}'$ to $\mathcal{G}$. A regular resolver simulating φ does not suffice to do this, as it does not take into account the partition into Eve and Adam vertices. We need to be able to simulate a play of $\mathcal{G}'$ in $\mathcal{G}$ in a two-players-game fashion: Adam's moves will be simulated by Adam, and Eve's moves by Eve. This idea leads to the notion of HD-for-games mapping.

History-determinism-for-games

Let $\mathcal{G}$ and $\mathcal{G}'$ be two games,[1] and $\varphi : \mathcal{G} \to \mathcal{G}'$ be a weak morphism between them admitting a resolver $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r})$ simulating φ. Given runs $\rho' = e'_0 e'_1 \cdots \in \mathcal{R}un(\mathcal{G}')$ and $\rho = e_0 e_1 \cdots \in \mathcal{R}un(\mathcal{G})$, we say that ρ is *consistent with* $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r})$ over ρ' if:

1. $\mathsf{source}(e_0) = \mathsf{r}_{\mathsf{Init}}(\mathsf{source}(e'_0))$,
2. $\varphi(e_i) = e'_i$, and
3. for every finite prefix $e_0 e_1 \dots e_{n-1} \sqsubseteq \rho$ ending in a vertex controlled by Eve, the next edge in ρ is $e_n = \mathsf{r}(e_0 \dots e_{n-1}, e'_n)$.

We remark that there exists at least one run consistent with $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r})$ over ρ', namely $\mathsf{r}_{\mathcal{R}uns}(\rho')$. We say that $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r})$ is *sound for* $\mathcal{G}$ if it verifies that for any accepting run $\rho' \in \mathcal{R}un(\mathcal{G}')$, all runs consistent with $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r})$ over ρ' are accepting in $\mathcal{G}$.

Said differently, a resolver sound for $\mathcal{G}$ is a winning strategy for Duplicator in the following game:

- In round 0, Spoiler picks an initial vertex v'_0 in $\mathcal{G}'$. Duplicator responds by picking an initial vertex v_0 in $\mathcal{G}$ such that $\varphi(v_0) = v'_0$.
- In round $n > 0$, Spoiler picks an edge e'_n in $\mathcal{G}'$. If v_{n-1} is controlled by Adam, Spoiler chooses an edge $e_n = v_{n-1} \to v_n \in \mathsf{Out}(v_{n-1})$ such that $\varphi(e_n) = e'_n$. If v_n is controlled by Eve, it is Duplicator who chooses one such e_n.

[1] For the following definitions, it suffices in fact to suppose that the codomain $\mathcal{G}'$ is a game, but the domain $\mathcal{G}$ can be any transition system.

- Duplicator wins if either $e_1e_2\dots$ is an accepting run in $\mathcal{G}$ from v_0 or $e'_1e'_2\dots$ is not an accepting run in $\mathcal{G}'$ from v'_0. Spoiler wins otherwise.

Definition B.2.

An HD mapping of games $\varphi : \mathcal{G} \to \mathcal{G}'$ is called *history-deterministic-for-games* if it admits a resolver sound for $\mathcal{G}$.

Whenever we apply the term HD-for-games to a map $\varphi : \mathcal{TS} \to \mathcal{TS}'$, it will implicitly imply that $\mathcal{TS}$ and $\mathcal{TS}'$ are games (that is, they have a fixed vertex-labelling $l_{\mathsf{Players}} : V \to \{\text{Eve}, \text{Adam}\}$), and that φ preserves those vertex-labellings).

The proof of next lemma is identical to that of Lemma II.18.

Lemma B.3.

If $\varphi : \mathcal{G} \to \mathcal{G}'$ is a locally surjective morphism between games, it is also an HD-for-games.

Also, Lemmas II.19 and II.20 about the reduction and extension of initial set of vertices hold similarly by replacing HD mapping by HD-for-games mapping.

HD-for-games mappings generalise the product by an HD automaton

We first show that HD-for-games mappings generalise the product of a game by an HD automaton.

Proposition B.4.

Let $\mathcal{A}$ be a complete automaton accepting the language $\mathcal{L}(\mathcal{A}) = W \subseteq \Sigma^\omega$, and let $\mathcal{G}$ be a game suitable for transformations using as winning condition W. Then, there exists an HD-for-games mapping $\varphi\colon \mathcal{G} \ltimes \mathcal{A} \to \mathcal{G}$.

Proof. The fact that there is a locally surjective weak morphism $\varphi\colon \mathcal{G} \ltimes \mathcal{A} \to \mathcal{G}$ is given by Proposition II.21. We define a resolver $(\mathsf{r}_{\mathsf{Init}}, r_\varphi)$ as in the proof of that Proposition II.21: it follows the edges indicated by a resolver for $\mathcal{A}$. We prove that this resolver is sound for $\mathcal{G}$. We claim that if ρ is a run in $\mathcal{G}$, the only run consistent with $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r}_\varphi)$ over ρ is $\mathsf{r}_{\varphi,\mathcal{R}uns}(\rho)$. This follows from the fact that if (v, q) is a vertex in $\mathcal{G} \ltimes \mathcal{A}$ controlled by Adam and $e \in \mathsf{Out}(v)$, then there is a unique $e' \in \mathsf{Out}(v, q)$ such that $\varphi(e') = e$. This is indeed the case: as $\mathcal{G}$ is suitable for transformations, if v is an Adam's vertex, every $e \in \mathsf{Out}(v)$ is uncoloured, so by definition of φ we have that $\varphi(e') = e \implies e' = e$. (This can be seen as that φ is locally bijective in Adam's vertices). We conclude that if ρ is an accepting run in $\mathcal{G}$ and $\rho^\ltimes$ is a run consistent with $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r}_\varphi)$ over ρ, then $\rho^\ltimes = \mathsf{r}_{\varphi,\mathcal{R}uns}(\rho)$, which is accepting by soundness of the resolver $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r}_\varphi)$. ◀

Preservation of the winning regions in games

Lemma B.5.

Let $\mathcal{G}, \mathcal{G}'$ be two games, such that there is a weak morphism of games $\varphi\colon \mathcal{G} \to \mathcal{G}'$ that is locally surjective and preserves accepting runs. If Eve wins the game $\mathcal{G}$ from an initial vertex v, then she wins $\mathcal{G}'$ from $\varphi(v)$.

Proof. Let $v' = \varphi(v)$, and let $\mathsf{strat}_v \colon \mathcal{P}\mathit{ath}^{\mathsf{fin}}(\mathcal{G}) \to E$ be a strategy for Eve in G that is winning from v. Intuitively, we will define a strategy in $\mathcal{G}'$ as follows: for each finite run ρ' from v' in $\mathcal{G}'$, we pick a preimage $\rho \in \varphi^{-1}(\rho')$ in $\mathcal{G}$, look at the decision made by strat_v at the end of ρ and transfer it back to $\mathcal{G}'$ via φ. In order to define a correct strategy, that suggests valid edges, the choices of the preimages have to be made in a coherent manner. We formalise this idea next.

We will make use of a function $\mathsf{choice}_{st} \colon \mathcal{P}\mathit{ath}^{\mathsf{fin}}_{v'}(\mathcal{G}') \to \mathcal{P}\mathit{ath}^{\mathsf{fin}}_{v}(\mathcal{G})$ satisfying that for any $\rho' = e'_0 e'_1 \dots e'_{n-1} e'_n \in \mathcal{P}\mathit{ath}^{\mathsf{fin}}_{v'}(\mathcal{G}')$:

- The run $\mathsf{choice}_{st}(\rho')$ has length $n+1$.
- $\varphi_{\mathcal{R}uns}(\mathsf{choice}_{st}(\rho')) = \rho'$.
- Monotonicity: if $\tilde{\rho}' \sqsubseteq \rho'$ then $\mathsf{choice}_{st}(\tilde{\rho}') \sqsubseteq \mathsf{choice}_{st}(\rho')$.
- If there exists $e_n \in \varphi^{-1}(e'_n)$ such that $\mathsf{choice}_{st}(e'_0 e'_1 \dots e'_{n-1}) e_n$ is consistent with strat_v, then $\mathsf{choice}_{st}(\rho')$ is consistent with strat_v.

Assume for now that such a function exists, and define a strategy in $\mathcal{G}'$ as

$$\mathsf{strat}'_{v'}(\rho') = \varphi(\mathsf{strat}_v(\mathsf{choice}_{st}(\rho'))), \quad \text{for } \rho' \in \mathcal{P}\mathit{ath}^{\mathsf{fin}}_{v'}(\mathcal{G}'),$$

and defined arbitrarily for paths that do not start in v'.

We prove that $\mathsf{strat}'_{v'}$ is winning from v'. Let $\rho' = e'_0 e'_1 \cdots \in \mathcal{P}\mathit{ath}_{\mathcal{G}'}(v')$ be an infinite play consistent with $\mathsf{strat}'_{v'}$. For each finite prefix $\tilde{\rho}' \sqsubseteq \rho'$, $\mathsf{choice}_{st}(\tilde{\rho}')$ is a finite play in $\mathcal{G}$, and by the monotonicity assumption, we can define the limit of these runs as:

$$\vec{\rho} = e_0 e_1 e_2 \cdots \in \mathcal{R}un(\mathcal{G}), \quad \text{where } e_0 e_1 \dots e_n = \mathsf{choice}_{st}(e'_0 e'_1 \dots e'_n),$$

which is indeed a run in $\mathcal{G}$. We show that $\vec{\rho}$ is consistent with strat_v by induction. Let $\rho_n = e_0 e_1 \dots e_{n-1}$ be the prefix of size n of $\vec{\rho}$, and suppose that it ends in a vertex v_n controlled by Eve. We want to show that $e_n = \mathsf{strat}_v(\rho_n)$. By definition of $\mathsf{strat}'_{v'}$, $e'_n = \varphi(\mathsf{strat}_v(\rho_n)) = \varphi(e_n)$, and as v_n is controlled by Eve, $\mathsf{strat}_v(\rho_n)$ is the only continuation of ρ_n consistent with strat_v, so by the last property of choice_{st}, e_n has to coincide with $\mathsf{strat}_v(\rho_n)$, as we wanted. As $\vec{\rho}$ is consistent with the winning strategy strat_v, it is an accepting run in $\mathcal{G}$, and since φ preserves accepting runs, $\rho' = \varphi_{\mathcal{R}uns}(\vec{\rho})$ is also an accepting run.

Finally, we show how to build a function $\mathsf{choice}_{st} \colon \mathcal{P}\mathit{ath}^{\mathsf{fin}}_{v'}(\mathcal{G}') \to \mathcal{P}\mathit{ath}^{\mathsf{fin}}_{v}(\mathcal{G})$ by induction on the length of the runs. Suppose that choice_{st} has been defined for runs of length $\leq n$, and let $e'_0 e'_1 \dots e'_n$ be a run of length $n+1$, with $\mathsf{choice}_{st}(e'_0 e'_1 \dots e'_{n-1}) = e_0 e_1 \dots e_{n-1}$. If $e_0 e_1 \dots e_{n-1}$ is not consistent with strat_v, it ends in a vertex v_n controlled by Adam, or $\mathsf{strat}_v(e_0 e_1 \dots e_{n-1}) \notin \varphi^{-1}(e_n)$, we let $e_n \in \varphi^{-1}(e_n) \cap \mathsf{Out}(v_n)$ be any edge (one such edge exists by local surjectivity). On the contrary, we let $e_n = \mathsf{strat}_v(e_0 e_1 \dots e_{n-1})$. We define $\mathsf{choice}_{st}(e'_0 e'_1 \dots e'_{n-1} e'_n) = e_0 e_1 \dots e_{n-1} e_n$. By construction, the obtained function fulfils the 4 requirements. ◀

Proposition B.6.

Let $\mathcal{G}, \mathcal{G}'$ be two games such that there is an HD-for-games mapping $\varphi \colon \mathcal{G} \to \mathcal{G}'$. Eve's winning region in $\mathcal{G}'$ is the image of her winning region in $\mathcal{G}$: $\mathcal{W}in_{\mathrm{Eve}}(\mathcal{G}') = \varphi(\mathcal{W}in_{\mathrm{Eve}}(\mathcal{G}))$.

Proof. If Eve wins $\mathcal{G}$ from an initial vertex v, Lemma B.5 guarantees that she wins $\mathcal{G}'$ from $\varphi(v)$.

Suppose now that Eve wins $\mathcal{G}'$ from an initial vertex v' with a strategy $\mathsf{strat}' : \mathcal{P}\mathit{ath}^{\mathsf{fin}}(\mathcal{G}') \to E'$. We need to show that she wins $\mathcal{G}$ from some initial vertex in $\varphi^{-1}(v')$. Let $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r})$ be a resolver simulating φ sound for $\mathcal{G}$ and let $v = \mathsf{r}_{\mathsf{Init}}(v')$. We define

$$\mathsf{strat}_v(\rho) = \mathsf{r}(\rho, \mathsf{strat}'_{v'}(\varphi_{\mathcal{R}\mathit{uns}}(\rho))), \quad \text{for } \rho \in \mathcal{P}\mathit{ath}^{\mathsf{fin}}(\mathcal{G}).$$

That is, strat_v is a strategy in $\mathcal{G}$ that, given a finite run ρ, simulates ρ in $\mathcal{G}'$, looks at the move done by the strategy $\mathsf{strat}'_{v'}$ in there, and transfers this choice back to $\mathcal{G}'$ by using the resolver r. We prove that strat_v is winning for Eve in $\mathcal{G}$ from v. Let $\rho = e_0 e_1 \cdots \in \mathcal{P}\mathit{ath}_v(\mathcal{G})$ be a play consistent with strat_v. We claim that $\varphi(\rho)$ is consistent with $\mathsf{strat}'_{v'}$ and that ρ is consistent with $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r})$ over $\varphi(\rho)$. This implies the desired result; consistency with $\mathsf{strat}'_{v'}$ implies that $\varphi(\rho)$ is accepting, and since $(\mathsf{r}_{\mathsf{Init}}, \mathsf{r})$ is sound for $\mathcal{G}$, ρ would be accepting in $\mathcal{G}$.

We prove that $\varphi(\rho)$ is consistent with $\mathsf{strat}'_{v'}$. Let $e'_0 e'_1 \dots e'_{n-1}$ be a subplay of $\varphi(\rho)$ ending in a vertex v_n controlled by Eve. By definition of the strategy strat_v, we have that $e_n = \mathsf{r}(e_0 \dots e_{n-1}, \mathsf{strat}'_{v'}(e'_0 \dots e'_{n-1}))$, and by definition of a resolver (item 2), $e'_n = \varphi(e_n) = \mathsf{strat}'_{v'}(e'_0 \dots e'_{n-1}))$, as we wanted.

The fact that ρ is consistent with $(\mathsf{r}_{\mathsf{Init}}, r)$ over $\varphi(\rho)$ follows directly from the definition of strat_v. ◀

The next corollary follows from the previous proposition and Lemma II.20.

Corollary B.7.

Let $\mathcal{G}, \mathcal{G}'$ be two games whose states are accessible and such that their acceptance sets $W_{\mathcal{G}}$ and $W_{\mathcal{G}'}$ are prefix-independent. If there is an HD-for-games mapping $\varphi\colon \mathcal{G} \to \mathcal{G}'$, then Eve's full winning region in $\mathcal{G}'$ is the projection of her full winning region in $\mathcal{G}$: $\mathcal{W}\mathit{in}_{\mathrm{Eve}}(\mathcal{G}'_{V'}) = \varphi(\mathcal{W}\mathit{in}_{\mathrm{Eve}}(\mathcal{G}_V))$.

B.1.3 ACD-HD-Rabin-transform-for-games

As discussed in Section II.4.3, the ACD-HD-Rabin-transform of a game $\mathcal{G}$ does not always induce an HD-for-games mapping $\varphi\colon \mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{G}) \to \mathcal{G}$, and $\mathcal{G}$ and $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{G})$ do not necessarily have the same winner. This is to be expected, as the ACD-HD-Rabin-transform does not take into account the partition into Eve and Adam nodes. In this section, we propose a small modification on the transformation to obtain a correct transformation for games.

Let $\mathcal{G}$ be a game. If there is an edge $v \to v'$ with $v \in V_{\mathrm{Adam}}$, we say that v' is an *A-successor*. We remark that if $\mathcal{G}$ is suitable for transformations, an A-successor is controlled by Eve and has a unique predecessor. We let $V_{\text{A-succ}}$ be the set of A-successor of $\mathcal{G}$ and $V_{\mathrm{normal}} = V \setminus V_{\text{A-succ}}$. If $\mathcal{G}$ is suitable for transformations, for each $v \in V_{\text{A-succ}}$ we let $\mathsf{pred}(v)$ be its unique predecessor.

The idea to define the ACD-HD-Rabin-transform-for-games $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G})$ is the following: starting from the regular ACD-HD-Rabin-transform $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{G})$, we make some local changes to vertices that are A-successors. First, if $v \in V_{\mathrm{Adam}}$, we replace edges of the form $(v, x) \xrightarrow{n} (v', x')$ in $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{G})$ by $(v, x) \xrightarrow{\varepsilon} (v', x)$ (we forbid Adam to choose how to update the ACD-component). If such an edge is followed by $(v', x') \xrightarrow{n'} (v'', x'')$ in $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{G})$, then we add $(v', x) \xrightarrow{n} (v'', x'')$ to $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G})$ (we note that $v' \in V_{\mathrm{Eve}}$). That is, Eve chooses retroactively how to update the ACD-components performing two consecutive updates. We note that the node n' is not output in the new game; this is not a problem, since n must be an ancestor of n' (we could say that n contains more information regarding the acceptance condition).

✦ Remark B.8. *Let $\mathcal{G}$ be a game suitable for transformations, let $v \in V_{\text{Adam}}$ and $v \xrightarrow{e_1} v' \xrightarrow{e_2} v''$ be a path of size 2 in $\mathcal{G}$ from v. It holds:*

- *If some cycle ℓ contains e_2, it also contains e_1.*
- *$\mathcal{T}_{v'}$ is a subtree of $\mathcal{T}_v$.*
- *Let $n_1 \in \mathsf{Leaves}(\mathcal{T}_v)$ and $n_2 = \mathsf{Jump}_{\mathcal{T}_{v'}}(n_1, \mathsf{Supp}(n_1, e_1))$. Then, $\mathsf{Supp}(n_2, e_2))$ is a descendant of $\mathsf{Supp}(n_1, e_1))$ in $\mathcal{T}_{v'}$.*

Definition B.9 (ACD-HD-Rabin-transform-for-games)**.**

Let $\mathcal{G}$ be a Muller game suitable for transformations. For each vertex $v \in V$ we let $\eta_v \colon \mathsf{Leaves}(\mathcal{T}_v) \to \{1, \dots, \mathsf{rbw}(\mathcal{T}_v)\}$ be a mapping satisfying Property $(\star)$ from Lemma II.42. We define the *ACD-HD-Rabin-transform-for-games* of $\mathcal{G}$ to be the Rabin transition system $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G})$ defined as follows.

Vertices. The set of vertices is

$$\widetilde{V} = \bigcup_{v \in V_{\text{normal}}} \{v\} \times [1, \mathsf{rbw}(\mathcal{T}_v)] \cup \bigcup_{v \in V_{\text{A-succ}}} \{v\} \times [1, \mathsf{rbw}(\mathcal{T}_{\mathsf{pred}(v)})].$$

Players partition. A vertex (v, x) belongs to Eve if and only if v belongs to Eve in $\mathcal{G}$.

Initial vertices. $\widetilde{I} = \{(v_0, x) \mid v_0 \in I \text{ and } x \in \{1, \dots, \mathsf{rbw}(\mathcal{T}_{v_0})\}\}$.

Edges and output colours. Let $e = v \to v'$ in $\mathcal{G}$.

- If $v \in V_{\text{Eve}} \cap V_{\text{normal}}$, we add $(v, x) \xrightarrow{n} (v', x')$ to $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G})$ exactly in the same cases as in the regular ACD-HD-Rabin-transform.
- If $v \in V_{\text{Adam}}$, we let $(v, x) \xrightarrow{\varepsilon} (v', x)$ in $\widetilde{E}$ for each $x \in \{1, \dots, \mathsf{rbw}(\mathcal{T}_v)\}$.
- If $v \in V_{\text{A-succ}}$, we add $(v, x) \xrightarrow{n} (v', x')$ to $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G})$ if in the regular ACD-HD-Rabin-transform there is a path of size 2 of the form

$$(\mathsf{pred}(v), x) \xrightarrow{n} (v, \tilde{x}) \xrightarrow{\tilde{n}} (v', x').$$

Formally,

$$\widetilde{E} = \bigcup_{\substack{e = v \to v' \in E \\ v \in V_{\text{normal}}}} \{e\} \times \mathsf{Leaves}(\mathcal{T}_v) \cup \bigcup_{\substack{e = v \to v' \in E \\ v \in V_{\text{A-succ}}}} \{e\} \times \mathsf{Leaves}(\mathcal{T}_{\mathsf{pred}(v)}).$$

Rabin condition. $R = \{(G_n, R_n)\}_{n \in \mathsf{Nodes}_{\bigcirc}(\mathcal{ACD}_{\mathcal{TS}})}$, where G_n and R_n are defined as follow: Let n be a round node, and let n' be any node in $\mathsf{Nodes}(\mathcal{ACD}_{\mathcal{TS}})$,

$$\begin{cases} n' \in G_n & \text{if } n' = n, \\ n' \in R_n & \text{if } n' \neq n \text{ and } n \text{ is not an ancestor of } n'. \end{cases}$$

Correctness of the ACD-HD-Rabin-transform-for-games

Proposition B.10 (Correctness of the ACD-HD-Rabin-transform-for-games)**.**

Let $\mathcal{G}$ be a Muller game suitable for transformations. There is an HD-for-games mapping $\varphi\colon \mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G}) \to \mathcal{G}$.

Proof. The proof is analogous to that of the correctness of the regular ACD-HD-Rabin-transform (Proposition II.75). We define the mapping $\varphi\colon \mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G}) \to \mathcal{G}$ as $\varphi_V(v,x) = v$ and $\varphi_E(e,l) = e$. It is clear that it is a weak morphism, and it preserve accepting runs by Lemma II.76 and Remark B.8.

We define a resolver (r_0, r) simulating φ in a similar way as in the proof of Proposition II.75: We use $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{G})$ to guide the resolver. Let $\rho = v_0 \to v_1 r e \dots$ be a run in $\mathcal{G}$, and let $(v_0, l_0) \to (v_1, l_1) \to \dots$ be the preimage of this run in $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{G})$. We simulate ρ in $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G})$ as follows: We ensure that at every moment i, if $v_i \notin V_{\text{A-succ}}$, the current vertex (v_i, x_i) is such that $x_i = \eta_{v_i}(l_i)$. There distinguish two cases to simulate the edge $e_i = v_i \to v_{i+1}$:

- If $v_i \in V_{\text{Adam}}$, there is a single outgoing edge from (v_i, x_i) mapped to the edge $v_i \to v_{i+1}$ in ρ: $(v_i, x_i) \xrightarrow{\varepsilon} (v_{i+1}, x_i)$. This must be the edge picked by the resolver
- If $v_i \in V_{\text{Eve}}$, we pick the edge $(v_i, x_i) \xrightarrow{n_i} (v_{i+1}, x_{x+1})$ such that $x_{i+1} = \eta_{l_{i+1}}$ and $n_i = \mathsf{Supp}(l_i, e_i)$.

If $v_i \in V_{\text{A-succ}}$, the vertex (v_i, x_i) will verify $x_i = x_{i-1} = \eta_{v_i}(l_i)$. In this case, we pick the edge $(v_i, x_i) \xrightarrow{n_i} (v_{i+1}, x_{x+1})$ such that $x_{i+1} = \eta_{l_{i+1}}$ and $n_i = \mathsf{Supp}(l_i, e_i)$. This is indeed an edge appearing in $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G})$, as the path $(v_{i-1}, x_{i-1}) \xrightarrow{n'_{i-1}} (v_i, x'_i) \xrightarrow{n_i} (v_{i+1}, x_{x+1})$ exists in the regular $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{G})$, with $x'_i = \eta_{v_i}(l_i)$.

The resolver obtained in this way is sound for $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G})$, as there is a unique way to simulate edges issued from Adam vertices, and the rest of the edges are simulated in the same way as the resolver defined for the regular ACD-HD-Rabin-transform, which we proved to be sound. ◀

Corollary B.11.

Let $\mathcal{G}$ be a Muller game suitable for transformations. Eve's full winning region in $\mathcal{G}$ is the projection of her full winning region in $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G})$.

Optimality of the ACD-HD-Rabin-transform-for-games

We obtain an analogous optimality result as the one given in Theorem II.7 for the ACD-HD-Rabin-transform-for-games. In this case, the bound is not tight due to the additional vertices that are added to $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G})$.

Corollary B.12.

Let $\mathcal{G}$ be a Muller game suitable for transformations whose states are accessible and let $\widetilde{\mathcal{G}}$ be a Rabin game. If $\widetilde{\mathcal{G}}$ admits an HD-for-games mapping $\varphi\colon \widetilde{\mathcal{G}} \to \mathcal{G}$, then, $|\mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G})| \leq 2|\widetilde{\mathcal{G}}|$.

Proof. The vertices of $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G})$ corresponding to vertices in V_{normal} are exactly the same that those in $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{G})$:

$$\{(v,x) \in \mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G}) \mid v \in V_{\text{normal}}\} = \{(v,x) \in \mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{G}) \mid v \in V_{\text{normal}}\}.$$

Moreover, for $v \in V_{\text{A-succ}}$, there is one vertex of the form (v,x) for each vertex $(\mathsf{pred}(v),x)$, and each $v \in V_{\text{A-succ}}$ has exactly one predecessor in V_{normal}, so we conclude that:

$$|\mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G})| \leq 2 \cdot |\{(v,x) \in \mathsf{ACD}^{\mathsf{game}}_{\mathsf{Rabin}}(\mathcal{G}) \mid v \in V_{\text{normal}}\}| \leq 2 \cdot |\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{G})| \leq 2 \cdot \mathcal{G}',$$

where the last inequality follows from Theorem II.7. ◀

B.2 Minimality of the ZT-parity-automaton with respect to HD automata: Proof of Theorem II.2

We intend to prove Theorem II.2, that is, that for any $\mathcal{F} \subseteq 2^{\Sigma}_{+}$, the automaton $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_{\mathcal{F}}}$ is minimal amongst HD parity automata recognising $\mathsf{Muller}(\mathcal{F})$. We will follow the same proof scheme than in the deterministic case (Theorem II.2), performing an induction over the height of the Zielonka tree. Assume that $\mathcal{A}$ is an HD parity automaton for $\mathsf{Muller}(\mathcal{F})$ and that n_0 is the root of $\mathcal{Z}_{\mathcal{F}}$ having $n_1,\dots,n_k$ as children. For each child n_i we want to find an HD subautomaton $\mathcal{A}_i$ recognising the language associated to $\mathcal{F}|_{\nu(n_i)}$ in such a way that the automata $\mathcal{A}_i$ are pairwise disjoint, which would allow us to carry out the induction and obtain that $|\mathcal{A}| \geq |\mathsf{Leaves}(\mathcal{Z}_{\mathcal{F}})| = |\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_{\mathcal{F}}}|$. Our objective will be therefore to prove:

Proposition B.13.

Let n_0 be the root of the Zielonka tree of $\mathcal{F}$, and let $n_1, n_2, \dots, n_k$ be an enumeration of the children of n_0. If $\mathcal{A}$ is an HD automaton recognising $\mathsf{Muller}_{\Sigma}(\mathcal{F})$, then, $\mathcal{A}$ contains k pairwise disjoint subautomata $\mathcal{A}_1,\dots,\mathcal{A}_k$ that are history-deterministic and such that $\mathcal{L}(\mathcal{A}_i) = \mathsf{Muller}_{\nu(n_i)}(\mathcal{F}|_{\nu(n_i)})$.

The non-determinism of $\mathcal{A}$ will make this task considerably more laborious than in the proof of Theorem II.3, and we will have to thoroughly examine the strategies used by the resolvers for $\mathcal{A}$. By the inherently asymmetric semantics of non-deterministic automata, there are two well-differentiated cases to consider, depending on whether the root of the Zielonka tree is round ($\Sigma \in \mathcal{F}$) or square ($\Sigma \notin \mathcal{F}$).

Global hypothesis in Section B.2.

In order to simplify the proof, we will suppose that all states are reachable using a sound resolver and that all automata have a single initial state, which can be done without lost of generality since a resolver for an HD automaton fixes such initial state in advance.

Case 1: The root of the Zielonka tree is a square node: $\Sigma \notin \mathcal{F}$

For the rest of the paragraph, we let $\mathcal{A} = (Q, \Sigma, q_0, \mathbb{N}, \Delta, \mathsf{parity})$ be a complete history-deterministic automaton recognising the Muller language $\mathsf{Muller}_{\mathcal{F}}(\Sigma)$ admitting a sound resolver (q_0, r) implemented by a memory structure $(\mathcal{M}, \sigma)$.

Composition of an automata and a memory implementing a resolver. As $\mathcal{M}$ is a pointed graph labelled with the transitions of $\mathcal{A}$, we can consider the product automaton $\mathcal{A} \ltimes \mathcal{M}$. We want to furthermore restrict the transitions of this automaton to those that are indicated by the next-move function σ.

Given an automaton $\mathcal{A}$ and a memory structure $(\mathcal{M}, \sigma)$, we define their *composition*, which we write $\mathcal{A} \lhd_\sigma \mathcal{M} = (Q \times M, \Sigma, (q_0, m_0), \Gamma, \Delta', W)$ as the automaton having transitions $(q, m) \xrightarrow{a:c} (q', m')$ if $\sigma(q, m, a) = e = q \xrightarrow{a:c} q'$ and $\mu(m, e) = m'$ (formally, Δ' is a subset of $\Delta \times E_\mathcal{M}$, where $E_\mathcal{M}$ are the edges of the memory skeleton). We note that $\mathcal{A} \lhd_\sigma \mathcal{M}$ is deterministic, and it is complete if $\mathcal{A}$ is.

The following lemma follows directly from the definition of soundness of a resolver and the definition of composition of an automaton and a memory structure.

Lemma B.14.

Let $\mathcal{A}$ be an automaton and $(\mathcal{M}, \sigma)$ a memory structure for $\mathcal{A}$. The resolver implemented by $(\mathcal{M}, \sigma)$ is sound if and only if $\mathcal{A}$ and $\mathcal{A} \lhd_\sigma \mathcal{M}$ recognise the same language.

We let $\pi_\mathcal{A} \colon \mathcal{A} \lhd_\sigma \mathcal{M} \to \mathcal{A}$ be the morphism of automata given by the projection into the first component: $\pi_{\mathcal{A},V}(q, m) = q$ and $\pi_{\mathcal{A},E}(e_1, e_2) = e_1$.

✦ Remark B.15. *If ρ is a path in $\mathcal{A} \lhd_\sigma \mathcal{M}$ that is labelled by input letters $a_0 a_1 \cdots \in \Sigma^\infty$ and producing output $c_0 c_1 \cdots \in \mathbb{N}^\infty$, then the $\pi_\mathcal{A}$-projection of ρ is a path in $\mathcal{A}$ labelled by $a_0 a_1 \cdots \in \Sigma^\infty$ and producing $c_0 c_1 \cdots \in \mathbb{N}^\infty$ as output.*

Disjoint projections of X-SCCs from the product automata.

✦ **Lemma B.16.** *Let $X \subseteq \Sigma$ and let $\mathcal{S}_X$ be an accessible X-FSCC of $\mathcal{A} \lhd_\sigma \mathcal{M}$. Then, $\pi_\mathcal{A}(\mathcal{S}_X)$ induces an HD subautomaton of $\mathcal{A}$ recognising $\mathsf{Muller}_X(\mathcal{F}|_X) = \{w \in X^\omega \mid \mathsf{Inf}(w) \in \mathcal{F}\}$.*

Proof. Let q_S be a state in $\pi_\mathcal{A}(\mathcal{S}_X)$ chosen to be initial. Let m_S be a state in $\mathcal{M}$ such that $(q_S, m_S) \in \mathcal{S}_X$. By Lemma II.38, $\mathcal{S}_X$ induces a deterministic subautomaton with initial state (q_S, m_S) recognising $\mathsf{Muller}_X(\mathcal{F}|_X)$. On the one hand, since $\pi_\mathcal{A}(\mathcal{S}_X)$ is an accessible subautomaton of $\mathcal{A}$ having only transitions labelled by X and by prefix-independence of $\mathcal{L}(\mathcal{A})$, we have that

$$\mathcal{L}(\pi_\mathcal{A}(\mathcal{S}_X)) \subseteq \mathcal{L}(\mathcal{A}) \cap X^\omega = \mathsf{Muller}_X(\mathcal{F}\big|_X).$$

On the other hand, the projection of any accepting run in $\mathcal{S}_X$ provides an accepting run in $\pi_\mathcal{A}(\mathcal{S}_X)$ (by Remark B.15), so

$$\mathcal{L}(\mathcal{S}_X) = \mathsf{Muller}_X(\mathcal{F}\big|_X) \subseteq \mathcal{L}(\pi_\mathcal{A}(\mathcal{S}_X)).$$

Moreover, a sound resolver for $\pi_\mathcal{A}(\mathcal{S}_X)$ is implemented by $(\mathcal{M}_{m_S}, \sigma)$ (the memory structure with initial state set to m_S). ◀

✦ **Lemma B.17.** *Let $n \in N$ be a square node of the Zielonka tree of $\mathcal{F}$ ($\nu(n) \notin \mathcal{F}$), and let $n_1, n_2 \in \mathsf{Children}_{\mathcal{Z}_\mathcal{F}}(n)$ be two different children of n. If $\mathcal{S}_1$ and $\mathcal{S}_2$ are two accessible $\nu(n_1)$-FSCC and $\nu(n_2)$-FSCC in $\mathcal{A} \lhd_\sigma \mathcal{M}$, respectively, then $\pi_\mathcal{A}(\mathcal{S}_1) \cap \pi_\mathcal{A}(\mathcal{S}_2) = \emptyset$.*

Proof. Suppose by contradiction that there is some state q in $\pi_{\mathcal{A}}(\mathcal{S}_1) \cap \pi_{\mathcal{A}}(\mathcal{S}_2)$, and let $m_1, m_2 \in M$ be such that (q, m_1) and (q, m_2) are states in $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. For $i = 1, 2$, let $\ell_i \in \mathit{Cycles}_{(q,m_i)}(\mathcal{A} \lhd_\sigma \mathcal{M})$ be the cycle over (q, m_i) containing all edges in $\mathcal{S}_i$. We note that $\mathsf{let}(\ell_i) = \nu(n_i)$ and therefore $\min \mathsf{col}(\ell_i)$ has to be even (as $\mathcal{A} \lhd_\sigma \mathcal{M}$ is deterministic), where let and col are the labellings of $\mathcal{A} \lhd_\sigma \mathcal{M}$ with input letters and output colours, respectively. By Remark B.15, the $\pi_{\mathcal{A}}$-projections of ℓ_1 and ℓ_2 are cycles over q in $\mathcal{A}$ labelled with $\nu(n_1)$ and $\nu(n_2)$ and in which the minimal colour appearing is even. By alternating these two cycles, we can build an accepting run in $\mathcal{A}$ over a word $w \in \Sigma^\omega$ with $\mathsf{Inf}(w) = \nu(n_1) \cup \nu(n_2)$, contradicting the fact that $\nu(n_1) \cup \nu(n_2) \notin \mathcal{F}$ (Remark II.27). ◀

Lemmas I.4, B.16 and B.17 imply Proposition B.13 in the case in which the root of the Zielonka tree is a square node.

Case 2: The root of the Zielonka tree is a round node: $\Sigma \in \mathcal{F}$

Before presenting the formal proof, let us discuss why considering these two cases separately is necessary. A first idea to obtain the desired result would be to follow the same steps as in Case 1. However, this approach encounters a major difficulty: the argument used in the proof of Lemma B.17 is not valid if $\Sigma \in \mathcal{F}$. Indeed, even if we can find two rejecting cycles ℓ_1, ℓ_2 such that $\mathsf{let}(\ell_i) = \nu(n_i)$, their $\pi_{\mathcal{A}}$-projections could a priori have a state in common; this would imply the *existence* of a rejecting run over the set of letters $\nu(n_1) \cup \nu(n_2) \in \mathcal{F}$, which is not enough to conclude, as the non-determinism of $\mathcal{A}$ leaves room for the existence of other accepting runs over this set of letters. To circumvent this difficulty, we need to take a closer look at the strategies used by the resolver. Rather than considering any finite memory strategy resolving the non-determinism of $\mathcal{A}$, we will show that we can choose a very precise resolver for which we will be able to obtain a result analogous to Lemma B.17. To do this, we first construct the letter game of $\mathcal{A}$, as introduced in [HP06], which is a Muller game satisfying that a strategy for it yields a resolver for $\mathcal{A}$. The strategy that we will use in this game is the one obtained by applying McNaughton's algorithm to solve Muller games [McN93] guided by the Zielonka tree, as presented in [DJW97].

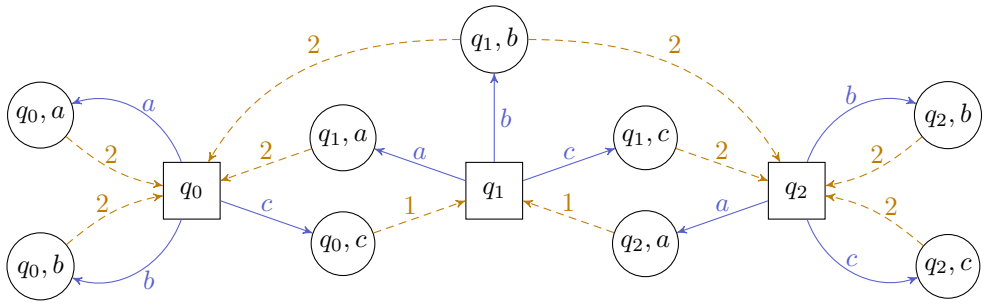
Letter game. Let $\mathcal{A} = (Q, \Sigma, q_0, [d_{\min}, d_{\max}], \Delta, \mathsf{parity})$ be a parity automaton recognising $\mathsf{Muller}(\mathcal{F})$, and suppose that $\Sigma \cap [d_{\min}, d_{\max}] = \emptyset$. The *letter game* for $\mathcal{A}$ is the game $\mathcal{G}_{\mathcal{A}}$ defined as follows:

- The set of vertices is $V = Q \sqcup (Q \times \Sigma)$. Adam controls vertices in Q, and Eve controls vertices in $Q \times \Sigma$.
- For each letter $a \in \Sigma$ and each $q \in Q$, there is an edge $q \xrightarrow{a} (q, a)$.
- For each position $(q, a) \in Q \times \Sigma$, and for each transition $q \xrightarrow{a:c} q'$ in $\mathcal{A}$, there is an edge $(q, a) \xrightarrow{c} q'$.
- The set of colours is $\Gamma = \Sigma \sqcup [d_{\min}, d_{\max}]$, and the acceptance set is the Muller language associated to

$$\mathcal{F} \to \mathsf{parity} = \{C \subseteq \Gamma \mid [C \cap \Sigma \in \mathcal{F}] \implies [\min(C \cap [d_{\min}, d_{\max}]) \text{ is even}]\}.$$

That is, in the letter game, Adam provides input letters one by one, and Eve chooses transitions corresponding to those letters in the automaton $\mathcal{A}$. Eve wins this game if she

manages to build an accepting run every time that Adam gives as input an infinite word in the language recognised by $\mathcal{A}$.



✦ **Figure 49.** Letter game for the HD automaton from Figure 4, recognising the Muller language associated to $\mathcal{F} = \{\{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}\}$. Squares represent Adam's vertices (the states of the automaton) and circles Eve's ones. Solid-blue edges correspond to input-letters, and dashed-orange edges to the choices of transitions in the automaton after each input letter. The only vertex where Eve has a non-trivial choice to make to resolve the non-determinism of the automaton is (q_1, b). In this example, Eve has a winning strategy corresponding to the resolver described in Example I.7.

We remark that a subgraph of $\mathcal{G}_\mathcal{A}$ induces a subautomaton of $\mathcal{A}$ via the (partial) mapping $\mathsf{aut}_\mathcal{A} \colon \mathcal{G}_\mathcal{A} \rightharpoonup \mathcal{A}$ that sends states of the form $q \in Q$ to q and edges of the form $(q, a) \xrightarrow{c} q'$ to $q \xrightarrow{a:c} q'$.

✦ Remark B.18. *A strategy for Eve in $\mathcal{G}_\mathcal{A}$ induces a resolver in $\mathcal{A}$, which is sound if and only if the strategy is winning.*

✦ Remark B.19. *If two subsets of vertices of the letter game $S_1, S_2 \subseteq V$ are disjoint, then $\mathsf{aut}_\mathcal{A}(S_1) \cap \mathsf{aut}_\mathcal{A}(S_2) = \emptyset$.*

✦ Remark B.20. *If ρ is a play in $\mathcal{G}_\mathcal{A}$, labelled $a_0c_0a_1c_1 \cdots \in (\Sigma \cdot [d_{\min}, d_{\max}])^\infty$, the $\mathsf{aut}_\mathcal{A}$-projection of ρ is a run in $\mathcal{A}$ over $a_0a_1 \cdots \in \Sigma^\infty$ producing $c_0c_1 \cdots \in [d_{\min}, d_{\max}]^\infty$ as output.*

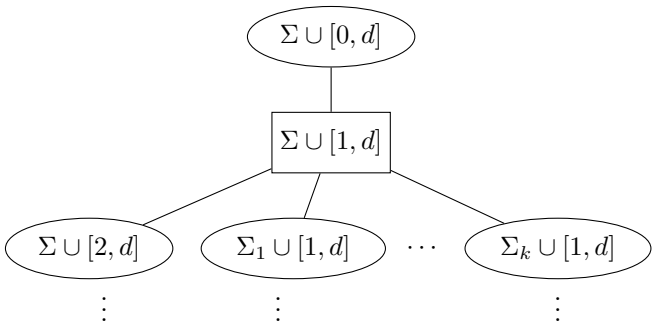
Lemma B.21 ([HP06])**.**

A parity automaton $\mathcal{A}$ is HD if and only if Eve wins the letter game from some initial state of $\mathcal{A}$.

For the rest of the paragraph, let $\mathcal{A} = (Q, \Sigma, q_0, [0, d], \Delta, \mathsf{parity})$ be a complete history-deterministic parity automaton recognising $\mathsf{Muller}(\mathcal{F})$. We can suppose without loss of generality that the minimal colour that it uses is 0. We let V and E denote the sets of vertices and edges, respectively, of the letter game and $\Gamma = \Sigma \sqcup [0, d]$ its set of colours. Whenever we use expressions like "the minimal colour appearing in a play", it will refer to the restriction of Γ to $[0, d]$. From the prefix-independence of $\mathsf{Muller}(\mathcal{F})$ we can moreover suppose that Eve wins the letter game from any vertex (see Lemma I.9). We let n_0 be the root of the Zielonka tree of $\mathcal{F}$ (supposed to be round, that is $\nu(n_0) \in \mathcal{F}$), let $n_1, \ldots, n_k$ be its children, and let $\Sigma_i = \nu(n_i) \subseteq \Sigma$ (note that $\Sigma_i \notin \mathcal{F}$ for $i \geq 1$).

Let us examine the condition $\mathcal{F} \to \mathsf{parity}$ used in the letter game a bit closer. The first levels of the Zielonka tree of this condition are depicted in Figure 50. It is clear that a strategy in $\mathcal{G}_\mathcal{A}$ ensuring to produce colour 0 infinitely often is winning. It might be the

case that Adam can prevent Eve from doing this, however, since Eve wins $\mathcal{G}_\mathcal{A}$, in that case she could ensure to produce infinitely often a set of colours included in some of the round nodes below the root, that is, to either avoid colour 1, or to produce letters included in some Σ_i. We use this idea to define next *attractor decompositions* for $\mathcal{G}_\mathcal{A}$.



✦ **Figure 50.** First levels of the Zielonka tree of the Muller condition $\mathcal{F} \to \mathsf{parity}$, which is the winning condition of the letter game $\mathcal{G}_\mathcal{A}$.

Attractor decompositions. For a subset X of vertices or edges of a game $\mathcal{G}$, we define Eve's *attractor to* X as:

$$\mathsf{Attr}_\mathcal{G}(X) = \{v \in V \mid \text{ there is a strategy for Eve ensuring to eventually visit } X \text{ from } v\}.$$

For a colour $c \in \Gamma$ we write $\mathsf{Attr}_\mathcal{G}(c) = \mathsf{Attr}_\mathcal{G}(E_c)$, where E_c is the set of edges coloured c. Given a subset of vertices $V' \subseteq V$ we write $\mathcal{G}_\mathcal{A}(V')$ to denote the subgame of $\mathcal{G}_\mathcal{A}$ containing the vertices of V' and the edges between them.

Let x be an even integer. For a subgame $\mathcal{G}' = \mathcal{G}_\mathcal{A}(V')$ of $\mathcal{G}_\mathcal{A}$ with no colour strictly smaller than x, we define an *x-attractor decomposition* of $\mathcal{G}'$ as a partition of V' into

$$V' = \mathsf{Attr}_{\mathcal{G}'}(x) \sqcup V_1 \sqcup A_1 \sqcup \ldots \sqcup V_l \sqcup A_l,$$

satisfying:

- $\mathsf{Attr}_{\mathcal{G}'}(x)$ is Eve's attractor to x in $\mathcal{G}'$.
- For each V_j, either (1) there is some $i \in \{1, \ldots, k\}$ such that no colour of $\Sigma \setminus \Sigma_i$ appears in $\mathcal{G}_\mathcal{A}(V_j)$, or (2) Eve has a winning strategy for $\mathcal{G}_\mathcal{A}(V_j)$ (from any vertex) avoiding colour $x+1$; and in both cases, if Adam can leave V_j taking an edge $v \xrightarrow{a} v'$ ($v \in V_j$, $v' \notin V_j$), then $v' \in \mathsf{Attr}_{\mathcal{G}'}(x) \sqcup V_1 \sqcup A_1 \sqcup \ldots V_{j-1} \sqcup A_{j-1}$. In case (1) we say that V_j is a *Σ_i-region of the attractor decomposition* and in case (2) that V_j is an *$x+1$-avoiding region*.
- Eve wins $\mathcal{G}_\mathcal{A}(V_j)$ from every vertex for all j.
- $A_j = \mathsf{Attr}_{\mathcal{G}_j}(V_j)$, where $\mathcal{G}_j$ is the subgame induced by the subset of vertices given by $V \setminus (\mathsf{Attr}_{\mathcal{G}'}(x) \sqcup V_1 \sqcup \ldots \sqcup V_{j-1} \sqcup A_{j-1})$ (we note that this game does not contain edges coloured with x).

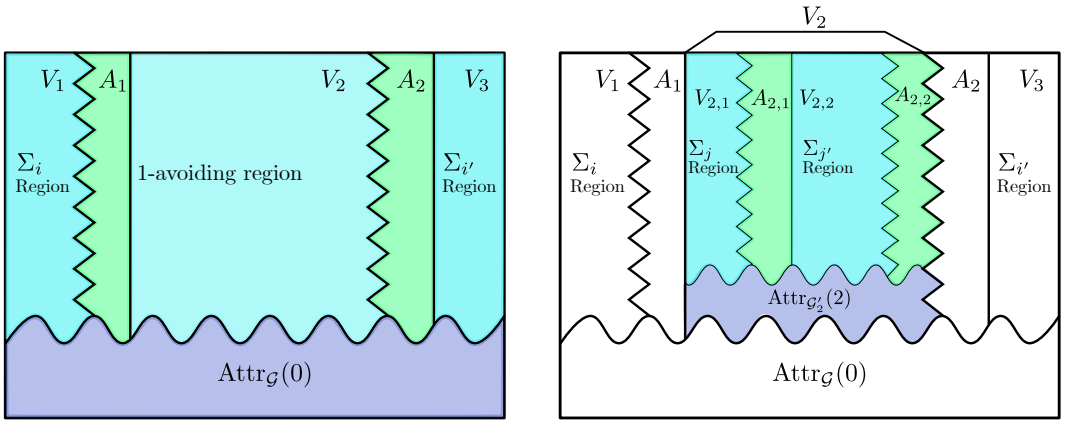
If V_j is an $x+1$-avoiding region, we let $\mathcal{G}'_j$ be the subgame obtained from $\mathcal{G}_\mathcal{A}(V_j)$ by removing the transitions labelled $x+1$.

An *x-recursive attractor decomposition* of $\mathcal{G}'$ is:

$$\mathcal{D}_{\mathcal{G}'} = \langle \mathsf{Attr}_{\mathcal{G}'}(x), (V_1, A_1, \mathcal{D}_{\mathcal{G}'_1}), (V_2, A_2, \mathcal{D}_{\mathcal{G}'_2}), \ldots, (V_l, A_l, \mathcal{D}_{\mathcal{G}'_l}) \rangle,$$

where $\mathsf{Attr}_{\mathcal{G}'}(x) \sqcup V_1 \sqcup A_1 \sqcup \ldots \sqcup V_l \sqcup A_l$ is an x-attractor decomposition of $\mathcal{G}'$, and, if V_j is an $x+1$-avoiding region, then $\mathcal{D}_{\mathcal{G}'_j}$ is an $x+2$-recursive attractor decomposition of $\mathcal{G}'_j$. (If V_j is an Σ_i-region, $\mathcal{D}_{\mathcal{G}'_j}$ can be disregarded).

✧ Note. *A representation of an attractor decomposition appears in Figure 51.*



✦ **Figure 51.** On the left, a 0-attractor decomposition of a game $\mathcal{G}$. On the right, the coloured part represents a 2-attractor decomposition of the subgame $\mathcal{G}'_2$ induced by the 1-avoiding region V_2. Since no 3-avoiding region appears on it, this is a full attractor decomposition of the game $\mathcal{G}$, inducing a partition into three different kinds of regions. The order over the Σ_i-regions is given by $V_1 <_{\mathcal{D}} V_{2,1} <_{\mathcal{D}} V_{2,2} <_{\mathcal{D}} V_3$. Adam can only force to decrease with respect to this order, that is, at each sublevel of the decomposition, Adam cannot force to go to the right.

Let $\mathcal{G}'$ be a subgame of $\mathcal{G}_{\mathcal{A}}$ with no colour strictly smaller than x and $\mathcal{D}_{\mathcal{G}'}$ an x-recursive attractor decomposition of it. We say that a subgame $\mathcal{S}$ of $\mathcal{G}'$ is a *Σ_i-region of $\mathcal{D}_{\mathcal{G}'}$* if it is a Σ_i-region of some of the recursively defined attractor decompositions. Similarly, for $y > x$ an odd integer, we say that $\mathcal{S}$ is a *y-avoiding region of $\mathcal{D}_{\mathcal{G}'}$* if it is a y-avoiding region of some of the recursively defined attractor decompositions. By convention, $\mathcal{G}'$ is an $x-1$-avoiding region (note that x might take the value 0). We remark that for any subset S of vertices of $\mathcal{G}'$ there is one and only one minimal y-avoiding region of $\mathcal{D}_{\mathcal{G}'}$ containing S (note that y might equal -1).

✦ Remark B.22. *A 0-recursive attractor decomposition $\mathcal{D}_{\mathcal{G}_{\mathcal{A}}}$ of $\mathcal{G}_{\mathcal{A}}$ induces a partition of the vertices into*

$$V = S_1 \sqcup \ldots \sqcup S_r \;\sqcup\; A_1 \sqcup \ldots A_r \;\sqcup\; B_1 \sqcup \ldots \sqcup B_s,$$

such that:

- *S_j is a Σ_i-region of $\mathcal{D}_{\mathcal{G}_{\mathcal{A}}}$, for some $i \in \{1, \ldots, k\}$,*
- *$A_j = \mathsf{Attr}_{\mathcal{G}_j}(S_j)$ for some subgame $\mathcal{G}_j$ appearing at some level of the decomposition,*
- *$B_j = \mathsf{Attr}_{\mathcal{G}'_j}(x)$ for some even integer x and some $x-1$-avoiding region $\mathcal{G}'_j$ appearing at some level of the decomposition.*

Moreover, such a decomposition induces a total order over the Σ_i-regions: for two sets S_t, $S_{t'}$, we write $S_t <_{\mathcal{D}} S_{t'}$ if there are two regions V_j, $V_{j'}$ belonging to the same attractor decomposition in $\mathcal{D}_{\mathcal{G}_{\mathcal{A}}}$ such that $j < j'$, $S_t \subseteq V_j$ and $S_{t'} \subseteq V_{k'}$.

We call such a partition a full attractor decomposition *of* $\mathcal{G}_{\mathcal{A}}$*. We remark that, by definition of an attractor decomposition, Eve wins* $\mathcal{G}_{\mathcal{A}}(S_j)$ *from every vertex for every* S_j*. See Figure 51 for an illustration.*

The proof that $\mathcal{G}_{\mathcal{A}}$ admits a full attractor decomposition uses the ideas appearing in [DJW97, Section 3].

Lemma B.23.

Let x be an even integer. If $\mathcal{G}'$ is a subgame of $\mathcal{G}_{\mathcal{A}}$ with no colour smaller than x and such that Eve can win from every vertex, then it admits an x-attractor decomposition. In particular, $\mathcal{G}_{\mathcal{A}}$ admits a full attractor decomposition.

Proof. We suppose without loss of generality that $x = 0$. Assume that $V_1, A_1, \dots V_{j-1}, A_{j-1}$ have already been defined and that they satisfy the desired properties. Assume that the game $\mathcal{G}_j$ with vertices $V \setminus (\mathsf{Attr}_{\mathcal{G}'}(0) \sqcup V_1 \sqcup \dots V_{j-1} \sqcup A_{j-1})$ is non-empty. First, note that Eve wins $\mathcal{G}_j$ from any position. Indeed, Eve wins $\mathcal{G}'$ from any vertex v in $\mathcal{G}_j$ (as we assume that she can win $\mathcal{G}'$ starting anywhere); moreover, since $v \notin A_{j'}$ for any $j' < j$, Adam has a strategy from v to force to remain in $\mathcal{G}_j$, and Eve has to be able to win against any such strategy.

We prove that either (1) there is some $i \in \{1, \dots, k\}$ and v vertex in $\mathcal{G}_j$ such that Eve has a winning strategy from v forcing to produce no colour in $\Sigma \setminus \Sigma_i$, or (2) there is some vertex v in $\mathcal{G}_j$ such that Eve has a winning strategy from v avoiding colour 1. Suppose by contradiction that this was not the case. Then, Adam can use the following strategy: first, he forces to produce colour 1, then, a colour not in Σ_1, followed by a colour not in Σ_2, and continues this pattern until a colour not in Σ_l is produced (and this without producing colour 0, since no 0-edge appears in $\mathcal{G}_j$). Afterward, he continue repeating these steps in a round-robin fashion. This allows him to produce a play winning for him (the word produced is in $\mathsf{Muller}(\mathcal{F})$ while the minimal number produced is 1), contradicting the fact that Eve wins $\mathcal{G}_j$ from v.

We suppose that we are in the case (1) (case (2) is identical), so from some vertices Eve can win producing no colour in $\Sigma \setminus \Sigma_i$. We let V_j be the set of such vertices, and we fix a strategy strat_j that is winning in $\mathcal{G}_j$ and avoids colours in $\Sigma \setminus \Sigma_i$. By definition of V_j, if $\rho = v \rightsquigarrow v'$ is a finite play consistent with strat_j in $\mathcal{G}_j$, then $v' \in V_j$ (Eve can still win without producing colours in $\Sigma \setminus \Sigma_i$), so Adam cannot force to leave V_j. This proves that:

1. strat_j is winning in $\mathcal{G}_{\mathcal{A}}(V_j)$ from every vertex,
2. if $v \in V_j$ is controlled by Adam and $v \to v'$ is an edge in $\mathcal{G}_{\mathcal{A}}$, then $v' \in \mathsf{Attr}_{\mathcal{G}'}(0) \sqcup V_1 \sqcup A_1 \sqcup \dots V_{j-1} \sqcup A_{j-1} \sqcup V_j$.

Also, if a vertex v controlled by Adam is in V_j, no edge $v \xrightarrow{a \notin \Sigma_i} v'$ appears in $\mathcal{G}_j$, so no colour of $\Sigma \setminus \Sigma_i$ appears in $\mathcal{G}_{\mathcal{A}}(V_j)$.

To finish the proof, we define A_j to be the attractor of V_j in $\mathcal{G}_j$.

The existence of a full attractor decomposition for $\mathcal{G}_{\mathcal{A}}$ follows from the fact that any $x+1$-avoiding region of an x-attractor decomposition satisfies the hypothesis of the lemma. ◀

Lemma B.24 ([McN93])**.**

Eve can play optimally in Muller games using finite memory. That is, if $\mathcal{G}$ is a game using a Muller winning condition, there is a strategy strat for Eve implemented by a finite memory structure winning from all her winning region.

Finding disjoint HD subautomata from X-closed subgames. For the rest of the paragraph, we fix a 0-recursive attractor decomposition $\mathcal{D}_{\mathcal{G}_\mathcal{A}}$ for $\mathcal{G}_\mathcal{A}$ and let $S_1 <_\mathcal{D} S_2 \ldots <_\mathcal{D} S_r$ be the Σ_i-regions of the induced full attractor decomposition. For each region S_j we fix a memory structure $(\mathcal{M}_j, \sigma_j)$ implementing a winning strategy for Eve in $\mathcal{G}_\mathcal{A}(S_j)$ (as given by Lemma B.24). As in the previous paragraph, we can consider the composition $\mathcal{G}_\mathcal{A}(S_j) \lhd_{\sigma_j} \mathcal{M}_j$ consisting of the product game in which the choices for Eve are restricted to those of the form $(v,m) \xrightarrow{c} (v',m')$ if $\sigma_j(v,m) = e = v \xrightarrow{c} v'$ and $\mu_j(m,e) = m'$. By definition, Eve does not have any choice in $\mathcal{G}_\mathcal{A}(S_j) \lhd_{\sigma_j} \mathcal{M}_j$, and since $(\mathcal{M}_j, \sigma_j)$ implements a winning strategy, any infinite path in $\mathcal{G}_\mathcal{A}(S_j) \lhd_{\sigma_j} \mathcal{M}_j$ produces a set of colours in $\mathcal{F} \to \mathsf{parity}$. We let $\pi_\mathcal{G} \colon \mathcal{G}_\mathcal{A}(S_j) \lhd_{\sigma_j} \mathcal{M}_j \to \mathcal{G}_\mathcal{A}(S_j)$ be the projection into $\mathcal{G}_\mathcal{A}(S_j)$.

A subgraph G_j of $\mathcal{G}_\mathcal{A}(S_j) \lhd_{\sigma_j} \mathcal{M}_j$ is *X-Adam-closed*, for a subset $X \subseteq \Sigma$, if for every vertex (q,m) controlled by Adam and every $a \in X$, the transition $(q,m) \xrightarrow{a} ((q,a),m')$ remains in G_j. We say that G_j is an *X-FSCC* if it is a final SCC of the restriction of $\mathcal{G}_\mathcal{A}(S_j) \lhd_{\sigma_j} \mathcal{M}_j$ to the graph where Adam's choices are restricted to letters in X that is moreover X-Adam-closed. We say that a subgraph $\mathcal{G}$ of $\mathcal{G}_\mathcal{A}$ is an *X-closed subgame* (with respect to the attractor decomposition $\mathcal{D}_{\mathcal{G}_\mathcal{A}}$ and a family of finite memory strategies) if $\mathcal{G} = \pi_\mathcal{G}(G_j)$ for G_j some X-Adam-closed SCC of some product $\mathcal{G}_\mathcal{A}(S_j) \lhd_{\sigma_j} \mathcal{M}_j$.

Intuitively, an X-closed subgame $\mathcal{G}$ of the letter game is a subgame included in a region S_j of the full attractor decomposition such that, if Adam only provides letters in X and Eve plays according to the strategy defined by the memory structures $\mathcal{M}_j$, the play will never leave $\mathcal{G}$.

✦ **Lemma B.25.** *Eve wins any X-closed subgame of $\mathcal{G}_\mathcal{A}$ (from any vertex).*

Proof. In an X-closed subgame included in a region $\mathcal{G}_\mathcal{A}(S_j)$, Adam's moves have been restricted; however, all Eve's moves coming from the strategy implemented by (M_j, σ_j) are available. Therefore, this strategy is also winning in such a subgame, since it is winning in the full $\mathcal{G}_\mathcal{A}(S_j)$. ◀

Putting this lemma together with Remark B.18 we obtain:

✦ **Lemma B.26.** *Let $X \subseteq \Sigma$, and let $\mathcal{G}_X \subseteq \mathcal{G}_\mathcal{A}$ be an X-closed subgame of $\mathcal{G}_\mathcal{A}$. The subautomaton of $\mathcal{A}$ induced by $\mathsf{aut}_\mathcal{A}(\mathcal{G}_X)$ is HD and recognises $\mathsf{Muller}_X(\mathcal{F}|_X)$.*

✦ **Lemma B.27.** *If a product $\mathcal{G}_\mathcal{A}(S_j) \lhd_{\sigma_j} \mathcal{M}_j$ does not contain any X-Adam-closed subgraph, for $X \subseteq \Sigma$, then from any vertex (q,m) Adam can force to leave S_j while playing only letters in X. That is, there is a path $(q,m) \rightsquigarrow (q',m')$ in $\mathcal{G}_\mathcal{A}(S_j) \lhd_{\sigma_j} \mathcal{M}_j$ producing exclusively letters in X such that, for some $a \in X$, the edge $q' \xrightarrow{a} (q',a)$ does not belong to $\mathcal{G}_\mathcal{A}(S_j)$.*

Proof. If this was not the case, the subgraph of $\mathcal{G}_\mathcal{A}(S_j) \lhd_{\sigma_j} \mathcal{M}_j$ consisting of the vertices that can be reachable from (q,m) by reading letters in X would form an X-Adam-closed subgraph. ◀

✦ **Lemma B.28.** *For each label Σ_i of the children of the root of $\mathcal{Z}_\mathcal{F}$, $\mathcal{G}_\mathcal{A}$ admits some Σ_i-closed subgame contained in a Σ_i-region of $\mathcal{D}_{\mathcal{G}_\mathcal{A}}$.*

Proof. Suppose that the full attractor decomposition of $\mathcal{G}_\mathcal{A}$ induced by $\mathcal{D}_{\mathcal{G}_\mathcal{A}}$ is the following:

$$V = S_1 \sqcup \ldots \sqcup S_r \ \sqcup \ A_1 \sqcup \ldots A_r \ \sqcup \ B_1 \sqcup \ldots \sqcup B_s,$$

We fix the following strategy strat for Eve in the letter game:

- whenever the play lands to B_j, where $B_j = \mathsf{Attr}_{\mathcal{G}'_j}(x)$ for some even colour x, she forces to produce colour x,
- whenever the play arrives to some A_j, she forces to go to S_j,
- in regions S_j she uses the strategy $(\mathcal{M}_j, \sigma_j)$. More precisely, let m_v be the state of $\mathcal{M}_j$ such that $(\mathcal{M}_{j,v}, \sigma_j)$ implements a winning strategy for $\mathcal{G}_\mathcal{A}(V_j)$ from (v, m_v). Each time that the play arrives to a vertex v in V_j from a different region, Eve uses $(\mathcal{M}_{j,v}, \sigma_j)$.

✧ Claim B.28.1. *Let ρ be a play consistent with* strat *(from any vertex), and let $y \geq -1$ be the maximal odd number such that $\mathsf{Inf}(\rho)$ is contained in a y-avoiding region $\mathcal{S}$ of $\mathcal{D}_{\mathcal{G}_\mathcal{A}}$. Then, either ρ eventually stays in a Σ_i-region S_j contained in $\mathcal{S}$, or the minimal colour produced infinitely often by ρ is $y+1$.*

Proof. Let $\mathsf{Attr}_\mathcal{S}(y+1) \sqcup V_1 \sqcup A_1 \sqcup \ldots\ldots, V_l \sqcup A_l$ be the attractor decomposition of $\mathcal{S}$ appearing in $\mathcal{D}_{\mathcal{G}_\mathcal{A}}$. By definition of an attractor decomposition, each time that the play leaves a V_j region, the next vertex is in $v' \in \mathsf{Attr}_\mathcal{S}(y+1) \sqcup V_1 \sqcup A_1 \sqcup \ldots V_{j-1} \sqcup A_{j-1}$. First, if V_j is a $y+2$-avoiding region, ρ cannot stay in it (by maximality of y). Thus, if ρ does not eventually stay in a Σ_i-region, it leaves regions V_j infinitely often, so it must produce $y+1$ infinitely often too. Since $\mathcal{S}$ is a y-avoiding region, no colour smaller than $y+1$ is produced. ◁

We obtain as a consequence that strat is winning for Eve from any initial position: any play staying in a y-avoiding region and producing infinitely many $y+1$'s is winning, and if a play eventually stays in a Σ_i-region S_j, it has to be winning since the strategy implemented by $(\mathcal{M}_j, \sigma_j)$ is winning in there.

We remark that we can extract a Σ_i-FSCC from any Σ_i-Adam-closed subgraph of $\mathcal{G}_\mathcal{A}(S_j) \lhd_{\sigma_j} \mathcal{M}_j$, that will be contained in the Σ_i-region S_j, so it suffices to prove the existence of such Σ_i-Adam-closed subgraphs. We also recall that in $\mathcal{G}_\mathcal{A}(S_j) \lhd_{\sigma_j} \mathcal{M}_j$ all choices are left to Adam, so he can choose to produce any path in this product whenever the play arrives to a vertex v in S_j.

Suppose by contradiction that no accessible Σ_i-Adam-closed subgraph exists in any of the products. We consider a play in which Adam does the following:

(a) the letters that he gives form a word $w \in \Sigma^\omega$ such that $\mathsf{Inf}(w) = \Sigma_i$,

(b) each time that the play arrives to a region S_j, he exists this region in a finite number of steps.

Indeed, he can ensure to exit regions S_j while only producing letters in Σ_i by Lemma B.27. By Claim B.28.1, the minimal colour produced infinitely often by such a play is even. By Remark B.20, we can project such a play in the automaton $\mathcal{A}$, obtaining an accepting run over w. This is a contradiction, since $w \notin \mathsf{Muller}(\mathcal{F}) = \mathcal{L}(\mathcal{A})$ (because $\Sigma_i \notin \mathcal{F}$). We

conclude that some $\mathcal{G}_{\mathcal{A}}(S_j) \lhd_{\sigma_j} \mathcal{M}_j$ admits a Σ_i-FSCC, and therefore $\mathcal{G}_{\mathcal{A}}$ admits some Σ_i-closed subgame. ◀

We can now infer Proposition B.13 in the case in which the root of $\mathcal{Z}_{\mathcal{F}}$ is round: from Lemma B.28, we obtain Σ_i-closed subgames in $\mathcal{G}_{\mathcal{A}}$ for each $i \in \{1, \dots, k\}$ that are moreover contained in Σ_i-regions. Therefore, their $\mathsf{aut}_{\mathcal{A}}$-projections are disjoint (Remark B.19), and each of these projections induces an HD-subautomaton recognising $\mathcal{F}|_{\Sigma_i}$ (Lemma B.26).

B.3 Size of the Zielonka tree and the ACD

B.3.1 Comparison of different representations of Muller languages

Explicit Muller vs Zielonka trees. First, we remark that an explicit representation of a family $\mathcal{F}$ can be exponentially larger than $\mathcal{Z}_{\mathcal{F}}$.

Proposition B.29.

For all $n \in \mathbb{N}$, there is a family of subsets $\mathcal{F}_n \subseteq 2^{\Sigma_n}_+$ over $\Sigma_n = \{1, \dots, n\}$ such that $|\mathcal{F}_n| = 2^n - 1$ and $|\mathcal{Z}_{\mathcal{F}}| = 1$.

Proof. It suffices to take $\mathcal{F} = 2^{\Sigma_n}_+$. ◀

Even if the family $\mathcal{F}$ is represented explicitly as a list of subsets, we cannot compute its Zielonka tree in polynomial time, as $\mathcal{Z}_{\mathcal{F}}$ can be super-polynomially larger than $|\mathcal{F}|$.

Proposition B.30.

For all $n \in \mathbb{N}$, there is a family of subsets $\mathcal{F}_n \subseteq 2^{\Sigma_n}_+$ over $\Sigma_n = \{1, \dots, n\}$ such that:

$$|\mathcal{Z}_{\mathcal{F}_n}| \geq |\mathcal{F}_n|^{\log(n)}.$$

More precisely, the family $\mathcal{F}_n$ satisfies $|\mathcal{Z}_{\mathcal{F}_n}| = n!$ and $|\mathcal{F}_n| = 2^n - 1$.

Proof. The condition $\mathsf{EvenLetters}_n$ from Proposition II.91 satisfies $|\mathcal{Z}_{\mathcal{F}_n}| = n!$ and $|\mathcal{F}_n| = 2^n - 1$. The inequality $|\mathcal{Z}_{\mathcal{F}_n}| \geq |\mathcal{F}_n|^{\log(n)}$ is obtained by applying Stirling's formula. ◀

There is no apparent reason why the lower bound in the previous proposition should be optimal.

Question B.6.

Find tight bounds for the comparison of the size of an explicit representation of a Muller condition and the size of its Zielonka tree.

Explicit Muller vs Zielonka DAGs. Hunter and Dawar showed that we can compute the Zielonka DAG of $\mathcal{F}$ in polynomial time if $\mathcal{F}$ is given as a list of subsets [HD08, Theorem 3.17].

Proposition B.31 ([HD08, Theorem 3.17])**.**

Given a family of subsets $\mathcal{F} \subseteq 2^{\Sigma}_{+}$, we can compute the Zielonka DAG of $\mathcal{F}$ in polynomial time in $|\mathcal{F}| + |\Sigma|$. In particular, $\mathcal{Z}$-DAG$_{\mathcal{F}}$ has polynomial size in $|\mathcal{F}| + |\Sigma|$.

By Proposition B.29, the Zielonka DAG can be exponentially smaller that an explicit representation of $\mathcal{F}$.

Zielonka trees vs Zielonka DAGs. It is clear that, given a Zielonka tree $\mathcal{Z}_{\mathcal{F}}$, we can compute the corresponding Zielonka DAG $\mathcal{Z}$-DAG$_{\mathcal{F}}$ in polynomial time. The converse is not possible.

Proposition B.32.

For all $n \in \mathbb{N}$, there is a family of subsets $\mathcal{F}_n \subseteq 2^{\Sigma_n}_{+}$ over $\Sigma_n = \{1, \dots, n\}$ such that:

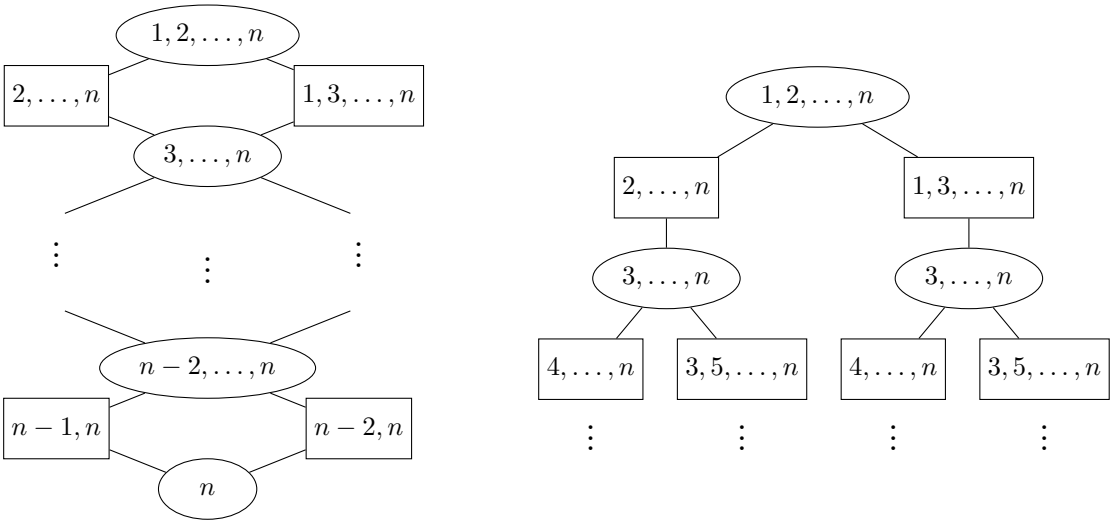
- the size of the Zielonka DAG of $\mathcal{F}_n$ is at most $2n$,
- the size of the Zielonka tree of $\mathcal{F}_n$ is at least $2^{\lfloor n/2 \rfloor}$.

Proof. Consider the family defined as follows:

$$\mathsf{MinOddAndSucc}_n = \{C = \{c_1 < c_2 < \cdots < c_k\} \subseteq \Sigma_n \mid c_1 \text{ is odd and } c_2 = c_1 + 1\}.$$

Equivalently, we can describe this family as

$$\bigcup_{\substack{i=1,\\ i \text{ odd}}}^{n} X_i, \text{ where } X_i = \{C \subseteq \Sigma_n \mid i \in C \text{ and } i+1 \in C \text{ and } c > i \text{ for all } c \in C\}.$$



✦ **Figure 52.** On the left, the Zielonka DAG of the condition $\mathsf{MinOddAndSucc}_n$ (for n odd), of size $\mathcal{O}(n)$. On the right, its Zielonka tree, of exponential size.

We show the Zielonka DAG and the Zielonka tree of $\mathsf{MinOddAndSucc}_n$ (for n odd) in Figure 52. We observe that the Zielonka DAG has height n; even levels consist in a single node, and odd levels have two nodes. Therefore, its size is $\lceil n/2 \rceil + n$. On the other hand, the Zielonka tree (with height also n), has $2^{\lfloor k/2 \rfloor}$ nodes at the level of depth k. ◀

Moreover, we note that for the Zielonka tree of $\mathsf{MinOddAndSucc}_n$, $\mathsf{rbw}(\mathcal{Z}_n)$ is also $2^{\lfloor n/2 \rfloor}$, and that the Muller language associated to this condition is a Streett language (the Zielonka tree in Figure 52 has Streett shape). We can construct a condition giving a Rabin language dually.

Rabin vs Zielonka trees and Zielonka DAGs. If the Muller language associated to a family $\mathcal{F}$ is a Rabin language, then the Zielonka DAG $\mathcal{Z}\text{-}\mathsf{DAG}_{\mathcal{F}}$ has Rabin shape (see Section II.6.1). In this case, we can compute a family of Rabin pairs R such that $\mathsf{Rabin}(R) = \mathsf{Muller}(\mathcal{F})$ in polynomial time. The converse is not possible, we cannot compute the Zielonka DAG in polynomial time, since it can be of exponential size in the number of Rabin pairs.

Proposition B.33.

Let $\mathcal{F} \subseteq 2^{\Sigma}_{+}$ be a family of subsets, and assume that its Zielonka DAG (resp. Zielonka tree) has Rabin shape. Then, we can compute in polynomial time in $|\mathcal{Z}\text{-}\mathsf{DAG}_{\mathcal{F}}|$ a family of Rabin pairs R over Σ such that $\mathsf{Rabin}_{\Sigma}(R) = \mathsf{Muller}_{\Sigma}(\mathcal{F})$.

Proof. The Rabin condition we define is almost the same as the one used by the ZT-HD-Rabin-automaton. Let $N = N_{\bigcirc} \sqcup N_{\square}$ be the nodes of the Zielonka DAG, partitioned into round and square nodes. By definition of Rabin shape, all round nodes have at most one child. We define a Rabin pair for each round node of $\mathcal{Z}\text{-}\mathsf{DAG}_{\mathcal{F}}$, $R = \{(\mathfrak{g}_n, \mathfrak{r}_n)\}_{n \in N_{\bigcirc}}$, where $\mathfrak{g}_n$ and $\mathfrak{r}_n$ are defined as follows:

$$\begin{cases} \mathfrak{g}_n = \Sigma \setminus \nu(n), \\ \mathfrak{r}_n = \nu(n) \setminus \nu(n'), \text{ for } n' \text{ the only child of } n, \text{ if it exists.} \\ \mathfrak{r}_n = \nu(n) \text{ if } n \text{ has no children.} \end{cases}$$

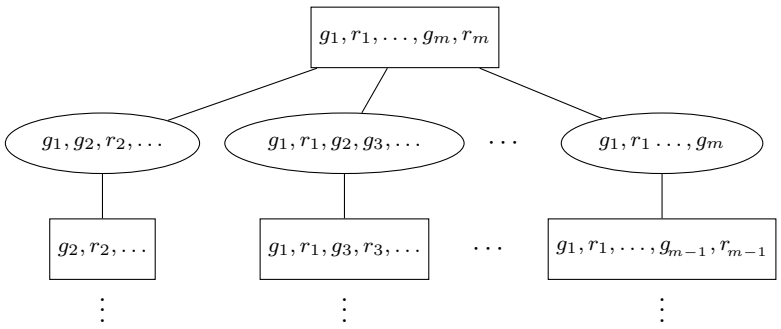
That is, the pair $(\mathfrak{g}_n, \mathfrak{r}_n)$ accepts the sets of colours $A \subseteq \Sigma$ that contain some of the colours that disappear in the child of n and none of the colours appearing above n in the Zielonka DAG. We show that $\mathsf{Rabin}(R) = \mathsf{Muller}(\mathcal{F})$. Let A be a set of colours. If $A \in \mathcal{F}$, let n be a maximal node (for $\preceq$) containing A. It is a round node and there is some colour $c \in A$ not appearing in the only child of n. Therefore, $c \in \mathfrak{g}_n$ and $A \cap \mathfrak{r}_n = \emptyset$. Conversely, if $A \notin \mathcal{F}$, then for every round node n with a child n', either $A \subseteq \nu(n')$ (and therefore $A \cap \mathfrak{g}_n = \emptyset$) or $A \not\subseteq \nu(n)$ (and in that case $A \cap \mathfrak{r}_n \neq \emptyset$). ◀

Proposition B.34.

For all $m \in \mathbb{N}$, there is a family R of m Rabin pairs over an alphabet Σ of size $2m$, such that $|\mathcal{Z}_{\mathcal{F}_R}| \geq m!$ and $|\mathcal{Z}\text{-}\mathsf{DAG}_{\mathcal{F}_R}| \geq 2^m$, where $\mathcal{F}_R \subseteq 2^{\Sigma}_{+}$ is the (only) family such that $\mathsf{Muller}(\mathcal{F}_R) = \mathsf{Rabin}(R)$.

Proof. Let $\Sigma = \{g_1, r_1, g_2, r_2, \ldots, g_m, r_m\}$ and define the Rabin pairs of R as $\mathfrak{g}_i = \{g_i\}$ and $\mathfrak{r}_i = \{r_i\}$. We depict the Zielonka tree of the corresponding family of subsets in Figure 53.

The Zielonka tree $\mathcal{Z}_{\mathcal{F}_R}$ satisfies that the levels at depth k and $k+1$ have $m(m-1)\ldots k$ nodes, which shows that $|\mathsf{Leaves}(\mathcal{Z}_{\mathcal{F}_R})| = m!$. For the bound on the size of the Zielonka DAG, we observe that for each subset $X \subseteq \{1, \ldots, m\}$ there is at least one subset appearing as the label of some nodes of the Zielonka tree, namely, $\{g_i, r_i \mid i \in X\}$. ◀



✦ **Figure 53.** The Zielonka tree $\mathcal{Z}_{\mathcal{F}_R}$ of the Rabin language from the proof of Proposition B.34.

Deterministic parity automata vs Zielonka trees. In Section II.3.2, we have seen that given a Zielonka tree $\mathcal{Z}_{\mathcal{F}}$, we can build a DPA recognising $\mathsf{Muller}(\mathcal{F})$ in polynomial time (DPA which is moreover minimal). Conversely, in Section III.3 (Proposition III.18), we show that given a DPA $\mathcal{A}$ recognising a Muller language $\mathsf{Muller}_\Sigma(\mathcal{F})$, we can compute the Zielonka tree $\mathcal{Z}_{\mathcal{F}}$ in polynomial time in the size of $\mathcal{A}$. This fact is used to prove that parity automata recognising Muller languages can be minimised in polynomial time (Theorem III.4).

Emerson-Lei vs Rabin. We recall that an Emerson-Lei condition represents a family $\mathcal{F} \subseteq 2^\Sigma_+$ as a positive boolean formula over the primitives $\mathtt{Inf}(c)$ and $\mathtt{Fin}(c)$. We show the missing implication in Figure 19.

Proposition B.35 (Folklore)**.**

Given a family of m Rabin pairs R over an alphabet Σ of size n, we can build an equivalent Emerson-Lei condition of size $\mathcal{O}(nm)$.

Proof. Let $R = \{\mathfrak{g}_1, \mathfrak{r}_1, \ldots, \mathfrak{g}_m, \mathfrak{r}_m\}$. It suffices to consider the formula

$$\bigvee_{1 \leq i \leq m} (\mathtt{Inf}(\mathfrak{g}_i) \wedge \mathtt{Fin}(\mathfrak{r}_i)),$$

where $\mathtt{Inf}(X)$ stands for $\bigvee_{c \in X} c$ and $\mathtt{Fin}(X)$ stands for $\bigwedge_{c \in X} c$. ◀

Proposition B.36 (Folklore)**.**

For each $n \in \mathbb{N}$, there exists an Emerson-Lei condition of size $\mathcal{O}(n)$ over an alphabet Σ_{2n} of size $2n$, such that it defines a Rabin language but any equivalent Rabin condition uses at least 2^n Rabin pairs.

Proof. Let $\Sigma_{2n} = \{a_1, \ldots, a_n, b_1, \ldots, b_n\}$, and consider the formula:

$$\bigwedge_{1 \leq i \leq n} (\mathtt{Fin}(a_i) \vee \mathtt{Fin}(b_i)).$$

It is easy to check that the corresponding Muller language is indeed a Rabin language. Let $R = \{\mathfrak{g}_1, \mathfrak{r}_1, \ldots, \mathfrak{g}_m, \mathfrak{r}_m\}$ be a family of Rabin pairs defining the same Muller language.

We first remark that each subset of red colours $\mathfrak{r}_i$ must have cardinality at least n, on the contrary $\Sigma_{2n}\backslash\mathfrak{r}_i$ would be accepting, which is not possible. For each subset $X \subseteq \{1,\dots,n\}$, there must be a Rabin pair accepting the set:

$$\{a_i \mid i \in X\} \cup \{b_j \mid j \notin X\}.$$

We conclude that $m \geq 2^n$. ◀

B.3.2 Size of the ACD by type

In Section II.5.1 we provided a worst-case analysis of the size of the Zielonka tree for any Muller language. Those results directly generalise to the alternating cycle decomposition. Here we refine those results by considering only subclasses of languages (parity, Rabin, generalised Büchi, etc...). We state the results directly for the alternating cycle decomposition. The proofs follow the same ideas of that of Proposition II.91, using the characterisation of typeness from Section II.6.2.

Proposition B.37.

Let $\mathcal{TS}$ be a strongly connected Muller transition system that is parity type and using m output colours. Then,

$$|\mathcal{ACD}_{\mathcal{TS}}| \leq m.$$

This bound is tight: for all $m \in \mathbb{N}$, there is a transition system $\mathcal{TS}_m$ that is parity type with one state, m edges and using m colours such that $|\mathcal{ACD}_{\mathcal{TS}}| = m$.

Proposition B.38.

Let $\mathcal{TS}$ be a strongly connected Rabin TS (resp. Streett TS), using r Rabin pairs and m output colours . Then:

- $|\mathcal{ACD}_{\mathcal{TS}}| \leq r!!$,
- $|\mathcal{ACD}_{\mathcal{TS}}| \leq m!!$.

These bound are tight: for all $r \in \mathbb{N}$, there is a Rabin transition system $\mathcal{TS}_r$ (resp. Streett TS) with one state, $2r$ edges and using r Rabin pairs such that $|\mathcal{ACD}_{\mathcal{TS}}| = r!!$.

Proposition B.39.

Let $\mathcal{TS}$ be a strongly connected transition system that is generalised Büchi type (resp. generalised coBüchi type) and using m output colours. Then,

$$|\mathcal{ACD}_{\mathcal{TS}}| \leq \binom{m}{\lfloor m/2 \rfloor}.$$

This bound is almost tight: for all $m \in \mathbb{N}$, there is a transition system $\mathcal{TS}_m$ that is generalised Büchi type (resp. generalised coBüchi type) with one state, m edges and using m colours such that $|\mathcal{ACD}_{\mathcal{TS}}| = \binom{m}{\lfloor m/2 \rfloor}$.

This result follows from Sperner's Theorem [Spe28], stating that the maximal size of an antichain of a set of m elements is $\binom{m}{\lfloor m/2 \rfloor}$.

B.4 Computation of the ACD and the ACD-DAG in polynomial time

We give here the proofs of Theorems II.8 and II.9.

B.4.1 Complexity of the ACD computation

We analyse now the complexity of Algorithm 1 proposed in Section II.5.2, depending on the representation of the acceptance condition, providing a proof for Theorem II.8. The analysis presented in the next subsection, using a different version for computing the children of a node of the ACD (function `ComputeChildren2`, Algorithm 4), supersedes the ongoing one. However, we show this analysis as the algorithms involved are the one presented in Section II.5.2, and an analysis of this first naive approach will help to understand which modifications in the algorithm are necessary to obtain the computation of the ACD-DAG in polynomial time in $|\mathcal{Z}\text{-DAG}_{\mathcal{F}}|$.

We observe that the cost of the computation of the alternating cycle decomposition of $\mathcal{TS}$ following Algorithm 1 is $\mathcal{O}(|\mathcal{ACD}_{\mathcal{TS}}| \cdot \alpha_{\mathcal{TS}})$, where $|\mathcal{ACD}_{\mathcal{TS}}|$ is the number of nodes of the ACD, and $\alpha_{\mathcal{TS}}$ denotes the cost of computing the children of a node using the procedure `ComputeChildren` (Algorithm 2). The complexity of this latter procedure depends, in turn, on the cost of $\texttt{MaxAltSubsets}(C, \mathcal{F})$, which greatly varies depending on the representation of the Muller condition.

If $\mathcal{F}$ is represented as an Emerson-Lei condition (that is, it is given by a boolean formula), the problem SAT reduces to deciding whether the output of $\texttt{MaxAltSubsets}(C, \mathcal{F})$ is empty, so this problem is therefore NP-complete. On the other hand, if $\mathcal{F}$ is given explicitly, as a Zielonka tree, or as a Zielonka DAG, we can compute $\texttt{MaxAltSubsets}(C, \mathcal{F})$ in polynomial time (Lemma B.40).

✦ **Lemma B.40.** *Given the Zielonka tree $\mathcal{Z}_{\mathcal{F}}$ (resp. Zielonka DAG $\mathcal{Z}\text{-DAG}_{\mathcal{F}}$) and a subset $C \subseteq \Gamma$, we can compute $\texttt{MaxAltSubsets}(C, \mathcal{F})$ in polynomial time. In particular, the output of $\texttt{MaxAltSubsets}(C, \mathcal{F})$ is polynomial in the size of $\mathcal{Z}\text{-DAG}_{\mathcal{F}}$.*

Proof. We show the result for the Zielonka DAG (which is a stronger result). We suppose w.l.o.g. that $C \in \mathcal{F}$. We will compute a (polynomial-sized) list altSets containing rejecting subsets of C, and amongst which lie the maximal ones. To compute $\texttt{MaxAltSubsets}(C, \mathcal{F})$ we just have to extract the maximal subsets of altSets.

To compute altSets, we first spot a node in $\mathcal{Z}\text{-DAG}_{\mathcal{F}}$ that is maximal containing C in its label. This can be done simply by inspecting the nodes in the DAG in a top-down fashion, and taking a child containing C greedily, if such child exists, until arriving at a node without children containing C. We let n_C be this node (which is round), and let $n_1, \ldots, n_k$ be its children. We let $D_i = C \cap \nu(n_i)$. For each D_i, we repeat the process looking into the nodes below n_i until finding a maximal node n_{D_i} containing D_i. If this node is square, we add D_i to altSets; if not, we repeat the process with the intersection of C with each child of n_{D_i}. By Lemma II.28, all sets added to altSets are rejecting. Moreover, if $D \subseteq C$ is a maximal rejecting subset, D is of the form $C \cap \nu(n)$, for n some square node below n_C, so all maximal rejecting subsets of C appear in altSets. ◀

We give now a technical lemma that will be useful for the subsequent analysis.

✦ **Lemma B.41.** *Let $C \subseteq \Gamma$ and let n_C be a node in $\mathcal{Z}_{\mathcal{F}}$ such that $C \subseteq \nu(n_C)$. Let $D_1, \ldots, D_k$ be k subsets of C such that, for all $i \neq j$, $C \in \mathcal{F} \iff D_i \notin \mathcal{F} \iff D_i \cup D_j \in \mathcal{F}$. Then, there are k strict descendants of n_C, $n_1, \ldots, n_k$, such that $D_i \subseteq \nu(n_i)$,*

$\nu(n_i) \in \mathcal{F} \iff D_i \in \mathcal{F}$ and such that nodes n_i are pairwise incomparable for the ancestor relation. Moreover, these nodes can be computed in polynomial time in $|\mathcal{Z}_\mathcal{F}|$.

Proof. To simplify notations we suppose that $C \in \mathcal{F}$ and $D_i \notin \mathcal{F}$ (the proof is symmetric in the other case). For each D_i we pick a node n_i which is a descendant of n_C, such that $D_i \subseteq \nu(n_i)$ and maximal for $\preceq$ with this property. In particular, n_i is square and a strict descendant (Lemma II.28). We prove that, for $j \neq i$, $D_j \nsubseteq \nu(n_i)$, implying that n_i and n_j are incomparable. Suppose by contradiction that for some $j \neq i$, $D_j \subseteq \nu(n_i)$. Then, $D_j \subseteq D_i \cup D_j \subseteq \nu(n_i)$, so, by Lemma II.28, $D_i \cup D_j \notin \mathcal{F}$, contradicting the hypothesis. ◀

As we will show (Lemma B.44), the following result also holds if we replace Zielonka tree by Zielonka DAG. Quite surprisingly, the arguments we need to use in each case are radically different.

✦ **Lemma B.42.** *For every state v, the tree $\mathcal{T}_v$ has size at most $|\mathcal{Z}_\mathcal{F}|$.*

Proof. We define in a top-down fashion an injective function $f\colon \mathcal{T}_v \to \mathcal{Z}_\mathcal{F}$. For the base case, we send the root of $\mathcal{T}_v$ to the root of $\mathcal{Z}_\mathcal{F}$. Let n be a node in $\mathcal{T}_v$ such that $f(n)$ has been defined, and let $n_1, \dots, n_k$ be its children. We let $C_n = \mathsf{col}(\nu(n))$ and $D_i = \mathsf{col}(\nu(n_i))$ be the colours labelling the cycles of these nodes. These sets satisfy that for $i \neq j$, $C_n \in \mathcal{F} \iff D_i \notin \mathcal{F} \iff D_i \cup D_j \in \mathcal{F}$. Indeed, if the union of D_i and D_j does not change the acceptance, we could take the union of the corresponding cycles, contradicting maximality. Lemma B.41 provides k descendants of $f(n)$ such that the subtrees rooted at them are pairwise disjoint. This allows to define the $f(n_i)$ and carry out the induction. ◀

We conclude that the size of $\mathcal{ACD}_\mathcal{TS}$ is polynomial in $|\mathcal{TS}| + |\mathcal{Z}_\mathcal{F}|$:

$$|\mathcal{ACD}_\mathcal{TS}| \leq \sum_{v \in V} |\mathcal{T}_v| \leq |V| \cdot |\mathcal{Z}_\mathcal{F}|.$$

To obtain Theorem II.8, it suffices to show that the computation of the children of a given node of the ACD can be done in polynomial time in $|\mathcal{TS}| + |\mathcal{Z}_\mathcal{F}|$. Indeed, this will imply that Algorithm 1 terminates in polynomial time, as each node of $\mathcal{ACD}_\mathcal{TS}$ is added to nodesToTreat only once.

✦ **Lemma B.43.** *Algorithm 2 computes the output of* `ComputeChildren`$(\mathcal{S})$ *in polynomial time in $|\mathcal{TS}| + |\mathcal{Z}_\mathcal{F}|$.*

Proof. In order to facilitate the analyse of the algorithm, we will suppose that the function takes two extra dummy arguments: a node of the Zielonka tree $n_\mathcal{S}$ and a vertex $v_\mathcal{S}$ of $\mathcal{TS}$, `ComputeChildren`$(\mathcal{S}, n_\mathcal{S}, v_\mathcal{S})$. We show that in a call of the function, we can pick these extra arguments in such a way that each pair (n, v) is used at most once, for $n \in \mathcal{Z}_\mathcal{F}$ and $v \in \mathcal{TS}$. This will finish the proof, as in each individual recursive call we perform polynomially many operations. Indeed, in each recursive call, the algorithm performs:

- a call to `MaxAltSubsets`$(C, \mathcal{F})$, which takes polynomial time (Lemma B.40),
- a `for`-loop (Line 4) over the elements in maxAltSets. By Lemma B.40, this set contains polynomially many elements. For each of them, we perform a decomposition into strongly connected components, which can be done in polynomial time [Tar72].

We show how to choose the parameters $n_{\mathcal{S}}, v_{\mathcal{S}}$ in the successive calls so that each pair is used at most once. We choose them so they satisfy the invariant $\mathsf{col}(\mathcal{S}) \subseteq n_{\mathcal{S}}$ and $v_{\mathcal{S}} \in \mathcal{S}$. Suppose that we are executing $\texttt{ComputeChildren}(\mathcal{S}, n_{\mathcal{S}}, v_{\mathcal{S}})$, and let $D_1, \dots, D_k$ be an enumeration of $\texttt{MaxAltSubsets}(C, \mathcal{F})$, where $C = \mathsf{col}(\mathcal{S})$. By Lemma B.41, there are k incomparable descendants of $n_{\mathcal{S}}$ in $\mathcal{Z}_{\mathcal{F}}$, $n_1, \dots, n_k$, such that $D_i \subseteq \nu(n_i)$. Whenever we arrive to Line 11 in a loop iteration corresponding to a set D_i, and we have to launch a new recursive call for a subgraph $\mathcal{S}' \subseteq \mathcal{S}$, we call the function with the parameters $\texttt{ComputeChildren}(\mathcal{S}', n_i, v')$, where v' is any vertex in $\mathcal{S}'$. In the subsequent recursive calls done by $\texttt{ComputeChildren}(\mathcal{S}', n_i, v')$, only strict descendants of n_i and vertices in the SCC $\mathcal{S}'$ will be used as dummy arguments. As a consequence, we obtain that no pair (n, v) is used more than once, as the subtrees rooted at nodes n_i's are pairwise disjoint.

✧ Note. *We note that this last argument does not apply to the Zielonka DAG: even if $n_1, \dots, n_k$ are pairwise incomparable nodes in $\mathcal{Z}$-$\mathsf{DAG}_{\mathcal{F}}$, the subDAGs rooted at these nodes can have a non-empty intersection.*

◀

B.4.2 Efficient computation of the ACD-DAG

We show now the proof of Theorem II.9, that is, that we can compute $\mathcal{ACD}$-$\mathsf{DAG}_{\mathcal{TS}}$ in polynomial time in the representation of $\mathcal{TS}$ if the acceptance condition is given as a Zielonka DAG $\mathcal{Z}$-$\mathsf{DAG}_{\mathcal{F}}$. For the high-level description of the algorithm we use a straightforward adaptation of Algorithm 1 to the ACD-DAG: we compute the DAG corresponding to each SCC of $\mathcal{TS}$ recursively in a top-down fashion; before adding a new child, we check whether it already appears in the part of the DAG previously computed.

As before, in order to show that this can be done in polynomial time, we need to show: (1) the size of the output, i.e. $|\mathcal{ACD}\text{-}\mathsf{DAG}_{\mathcal{TS}}|$ is polynomial in $|\mathcal{TS}| + |\mathcal{Z}\text{-}\mathsf{DAG}_{\mathcal{F}}|$, and (2) we can compute the children of a node in polynomial time in this measure.

Upper bound on the size of the ACD-DAG

We start by showing that $|\mathcal{ACD}\text{-}\mathsf{DAG}_{\mathcal{TS}}| \leq |\mathcal{TS}| \cdot |\mathcal{Z}\text{-}\mathsf{DAG}_{\mathcal{F}}|$. For that, we show that all $\mathcal{T}_v$-DAG can be embedded in $\mathcal{Z}$-$\mathsf{DAG}_{\mathcal{F}}$.

✦ **Lemma B.44.** *For every state v, the DAG $\mathcal{T}_v$-DAG has size at most $|\mathcal{Z}\text{-}\mathsf{DAG}_{\mathcal{F}}|$.*

Proof. We will define an injective function $f\colon \mathcal{T}_v\text{-}\mathsf{DAG} \to \mathcal{Z}\text{-}\mathsf{DAG}_{\mathcal{F}}$. For a node n in $\mathcal{T}_v$-DAG, we let $C_n = \mathsf{col}(\nu(n))$ be the set of colours appearing in the label of n. If n is not the root, we define $\mathsf{pred}(n)$ to be an immediate ancestor of n (that is, n is a child of $\mathsf{pred}(n)$). We let $\mathsf{pred}^*(n)$ be the sub-branch of nodes above n obtained by successive applications of pred, that is, $\mathsf{pred}^*(n) = \{n' \in \mathcal{T}_v\text{-}\mathsf{DAG} \mid n' = \mathsf{pred}^k(n) \text{ for some } k\}$. We note that the elements of $\mathsf{pred}^*(n)$ are totally ordered by $\preceq$ (n being the maximal node and the root the minimal one).

We define f recursively: For the root n_0 of $\mathcal{T}_v$-DAG, we let $f(n_0)$ be a maximal node (for $\preceq$) in $\mathcal{Z}$-$\mathsf{DAG}_{\mathcal{F}}$ containing C_{n_0} in its label. For n a node such that we have define f for all its ancestors, we let $f(n)$ be a maximal node (for $\preceq$) in the subDAG rooted at $f(\mathsf{pred}(n))$ containing C_n in its label. We remark that $f(n)$ is a round node if and only if n is a round node (by Lemma II.28). Also, if n' is an ancestor of n in $\mathsf{pred}^*(n)$, then $f(n')$ is an ancestor of $f(n)$ in $\mathcal{Z}$-$\mathsf{DAG}_{\mathcal{F}}$.

We prove now the injectivity of f. Let n_1, n_2 be two different nodes in $\mathcal{T}_v$-DAG (that is, $\nu(n_1) \neq \nu(n_2)$). First, we show that the colours appearing in their labels must differ.

✧ Claim B.44.1. *It is satisfied that* $C_{n_1} \neq C_{n_2}$.

Proof. Suppose by contradiction that $C_{n_1} = C_{n_2}$. Then, any node n containing $\nu(n_1)$ in its label satisfies that $\nu(n)$ is an accepting cycle if and only if $\nu(n) \cup \nu(n_2)$ is an accepting cycle. Let n be a node of minimal depth such that $\nu(n_1) \subseteq \nu(n)$ and $\nu(n_2) \not\subseteq \nu(n)$. The label of an immediate predecessor of n contains $\nu(n_1) \cup \nu(n_2)$ by minimality. This leads to a contradiction, as $\nu(n) \subsetneq \nu(n) \cup \nu(n_2)$, so $\nu(n)$ would not be a maximal subcycle producing an alternation in the acceptance status. ◁

We suppose w.l.o.g. that n_1 is round (that is, $C_{n_1} \in \mathcal{F}$). Suppose by contradiction that $f(n_1) = f(n_2)$. Then, n_2 is also round, and it is satisfied that $C_{n_1} \cup C_{n_2} \subseteq f(n_1)$, by definition of f. Again by definition of f, no child of $f(n_1)$ contains $C_1 \cup C_2$, so, by Lemma II.28, $C_1 \cup C_2 \in \mathcal{F}$. Let n' be the minimal node in $\mathsf{pred}^*(n_1)$ such that $\nu(n_2) \subseteq \nu(n')$. We do the prove for the case in which n' is round, the other case is symmetric. Let $\tilde{n}$ be the child of n' in $\mathsf{pred}^*(n')$, which is a square node. We claim that the following three properties hold:

i) $C_{\tilde{n}} \cup C_{n_2} \in \mathcal{F}$,

ii) $C_{\tilde{n}} \cup C_{n_2} \subseteq f(\tilde{n})$, and

iii) no child of $f(\tilde{n})$ contains $C_{\tilde{n}} \cup C_{n_2}$.

This leads to a contradiction, as the second and third items, combined with Lemma II.28 and the fact that $f(\tilde{n})$ is a square node, imply that $C_{\tilde{n}} \cup C_{n_2} \notin \mathcal{F}$. We prove the three items:

i) Follows from the fact that $\nu(\tilde{n})$ is a maximal rejecting cycle of $\nu(n')$, but $\nu(n')$ contains $\nu(\tilde{n}) \cup \nu(n_2)$.

ii) By definition of f, $C_{\tilde{n}} \subseteq f(\tilde{n})$. Also, the node $f(\tilde{n})$ is an ancestor of $f(n_2)$, so $C_{n_2} \subseteq f(n_2) \subseteq f(\tilde{n})$.

iii) By definition of f, no child of $f(\tilde{n})$ contains $C_{\tilde{n}}$ in its label. ◀

We conclude that the size of $\mathcal{ACD}\text{-}\mathsf{DAG}_{\mathcal{TS}}$ is polynomial in $|\mathcal{TS}| + |\mathcal{Z}\text{-}\mathsf{DAG}_{\mathcal{F}}|$:

$$|\mathcal{ACD}\text{-}\mathsf{DAG}_{\mathcal{TS}}| \leq \sum_{v \in V} |\mathcal{T}_v\text{-}\mathsf{DAG}| \leq |V| \cdot |\mathcal{Z}\text{-}\mathsf{DAG}_{\mathcal{F}}|.$$

A refined algorithm for computing the children of a node

We show now that we can compute the children of a node in $\mathcal{ACD}\text{-}\mathsf{DAG}_{\mathcal{TS}}$ in polynomial time in $|\mathcal{TS}| + |\mathcal{Z}\text{-}\mathsf{DAG}_{\mathcal{F}}|$. Unfortunately, the procedure `ComputeChildren` proposed in Algorithm 2 does not achieve this result. The reason is that `ComputeChildren`$(\mathcal{S})$ inspects subgraphs of $\mathcal{S}$ recursively, and we can possibly inspect many times subgraphs with a non-empty intersection, so there is no guarantee that the number of operations that we make is less than $|\mathcal{TS}|$. In Lemma B.43 we get to bound the number of times that a subgraph containing a state q is inspected by building a injection of the set of recursive calls done by the algorithm to $V \times \mathcal{Z}_{\mathcal{F}}$. However, this is not possible if we replace the Zielonka tree by the Zielonka DAG. In order to be able to employ a similar argument, we propose a different function to compute the children of a node: `ComputeChildren2` (Algorithm 4). The idea of the algorithm is to keep a stack of subgraphs $\mathcal{S}$ that we need to inspect, as cycles labelling children of the node under consideration might appear as subcycles of $\mathcal{S}$. Each time that we generate new subgraphs to inspect, before adding them to the stack,

we check whether there is a "similar enough" subgraph already appearing in the stack, and in that case we merge them. In this way, we can guarantee that the size of the stack will remain polynomial in $|\mathcal{TS}| \cdot |\mathcal{Z}\text{-DAG}_{\mathcal{F}}|$.

Next lemma follows directly from Lemma II.28. It will be useful to understand Algorithm 5 and to prove its correctness.

✦ **Lemma B.45.** *Let ℓ_1 and ℓ_2 be two cycles in $\mathcal{TS}$ with some state in common. For $i = 1, 2$, let n_i be a node in $\mathcal{Z}$-DAG$_{\mathcal{F}}$ containing $\mathsf{col}(\ell_i)$ and maximal amongst nodes containing $\mathsf{col}(\ell_i)$. If n_1 is an ancestor of n_2 ($n_1 \preceq n_2$), then*

$$\ell_1 \text{ is accepting} \iff \ell_2 \text{ is accepting} \iff \ell_1 \cup \ell_2 \text{ is accepting.}$$

Algorithm 4 `ComputeChildren2`($\mathcal{S}$): Computing the children of a node of the ACD

Input: A strongly connected subgraph $\mathcal{S}$ corresponding to a node of $\mathcal{ACD}$-DAG$_{\mathcal{TS}}$
Output: The maximal cycles $\ell_1, \dots, \ell_k \in \mathcal{C}ycles(\mathcal{S})$ such that $\mathsf{col}(\ell_i) \in \mathcal{F} \iff \mathsf{col}(\mathcal{S}) \notin \mathcal{F}$.

1: $C \leftarrow \mathsf{col}(\mathcal{S})$
2: $n_C \leftarrow$ maximal node in $\mathcal{Z}$-DAG$_{\mathcal{F}}$ containing C
3: children $\leftarrow \{\}$
4: sameAccList $\leftarrow \langle(\mathcal{S}, n_C)\rangle$
5: **while** sameAccList $\neq \emptyset$ **do**
6: $\quad\langle$alternatingList, sameAccList$\rangle \leftarrow$ `FindInterestingSubcycles`(sameAccList)
7: $\quad$children $\leftarrow$ children $\cup$ alternatingList
8: **end while**
9: children $\leftarrow$ `MaxInclusion`(children)
10: **return** children

The subprocedure `FindInterestingSubcycles` is presented in Algorithm 5. We present it in an asymmetric way (we suppose that the input subsets are accepting). This is done exclusively in order to simplify the presentation and facilitate its understanding. The generalisation to an algorithm taking a list of either accepting or rejecting cycles is straightforward.

Correctness of the algorithm. Let $\mathcal{S}$ be a strongly connected subgraph of $\mathcal{TS}$. We suppose w.l.o.g. that $\mathcal{S}$ is accepting. We show that, if the function `ComputeChildren2`($\mathcal{S}$) terminates, then it outputs the maximal rejecting subcycles of $\mathcal{S}$. Termination will be shown in the next paragraph. Suppose that the output $\langle$alternatingList, sameAccList$\rangle$ of the subprocedure `FindInterestingSubcycles` satisfies the conditions claimed at the beginning of Algorithm 5. Then, at each iteration of the `while`-loop in Line 5, the following invariants are preserved:

- alternatingList only contains rejecting subcycles,
- if ℓ is a rejecting subcycle of $\mathcal{S}$, then ℓ is either contained in some cycle of alternatingList or in some cycle of sameAccList.

Therefore, at the end of the `while`-loop, alternatingList contains the maximal rejecting subcycles of $\mathcal{S}$, and children is indeed the set of maximal rejecting subcycles of $\mathcal{S}$.

We prove now that the output of the subprocedure `FindInterestingSubcycles` (Algorithm 5) satisfies all the properties stated above Line 1. First, it is clear that all cycles

Algorithm 5 `FindInterestingSubcycles`$(\langle(\mathcal{S}_1, n_1), \dots, (\mathcal{S}_k, n_k)\rangle)$: Classifying subcycles guided by the Zielonka DAG

Input: A list of pairs $(\mathcal{S}_i, n_i)$ of a strongly connected subgraph and a node of the Zielonka DAG such that

- all the subgraphs $\mathcal{S}_i$ form accepting cycles,
- $\mathsf{col}(\mathcal{S}_i) \subseteq n_i$, and n_i is maximal with this property,
- for $i \neq j$, $\mathcal{S}_i \cap \mathcal{S}_j \neq \emptyset$ implies n_i and n_j incomparable (in particular, $n_i \neq n_j$).

Output:

1. A list alternatingList of rejecting subcycles of the subgraphs $\mathcal{S}_i$.
2. A list sameAccList of pairs $(\mathcal{S}'_j, n'_j)$ such that $\mathcal{S}'_j \subsetneq \mathcal{S}_i$ for some cycle of the input, and n'_j is a strict descendant of n_i. Moreover, the list sameAccList satisfies the same three hypothesis as the input.

Furthermore, if ℓ is a subcycle of some $\mathcal{S}_i$ that is rejecting, then ℓ is contained in some cycle in alternatingList or in sameAccList.

1: $C_i \leftarrow \mathsf{col}(\mathcal{S}_i)$, for $i = 1, \dots, k$ ▷ $C_i \in \mathcal{F}$
2: sameAccList $\leftarrow \langle(\mathcal{S}_1, n_1), \dots, (\mathcal{S}_k, n_k)\rangle$
3: maxAltSets$_i \leftarrow$ `MaxAltSubsets`$(C_i, \mathcal{F})$, for $i = 1, \dots, k$
4: **for** $i = 1, \dots, k$ **do**
5: sameAccList $\leftarrow$ sameAccList $\setminus \{(\mathcal{S}_i, n_i)\}$ ▷ We will replace $(\mathcal{S}_i, n_i)$ by some strict subcycles
6: **for** $D \in$ maxAltSets$_i$ **do**
7: $\mathcal{S}_{i,D} \leftarrow$ restriction of $\mathcal{S}_i$ to transitions $e \in E$ such that $\mathsf{col}(e) \in D$
8: $\langle \mathcal{S}_{i,D,1}, \dots, \mathcal{S}_{i,D,r} \rangle \leftarrow$ `SCC-Decomposition`$(\mathcal{S}_{i,D})$
9: **for** $j = 1, \dots, r$ **do**
10: **if** $\mathsf{col}(\mathcal{S}_{i,D,j}) \notin \mathcal{F}$ **then** ▷ An alternating subcycle has been found
11: alternatingList $\leftarrow$ alternatingList $\cup \{\mathcal{S}_{i,D,j}\}$
12: **else** ▷ $\mathcal{S}_{i,D,j}$ needs to be re-treated
13: $n_{i,D,j} \leftarrow$ a maximal descendant of n_i containing $\mathsf{col}(\mathcal{S}_{i,D,j})$
14: **if** for some $(\mathcal{S}', n') \in$ sameAccList, $\mathcal{S}' \cap \mathcal{S}_{i,D,j} \neq \emptyset$ and $n' \preceq n_{i,D,j}$ **then**
15: sameAccList $\leftarrow$ sameAccList $\cup \{(\mathcal{S}' \cup \mathcal{S}_{i,D,j}, n')\} \setminus \{(\mathcal{S}', n')\}$
▷ It is not possible that $(\mathcal{S}', n') \in$ sameAccList, $\mathcal{S}' \cap \mathcal{S}_{i,D,j} \neq \emptyset$ and $n_{i,D,j} \prec n'$
16: **else**
17: sameAccList $\leftarrow$ sameAccList $\cup \{(\mathcal{S}_{i,D,j}, n_{i,D,j})\}$
18: **end if**
19: **end if**
20: **end for**
21: **end for**
22: **end for**
23: **return** $\langle$alternatingList, sameAccList$\rangle$

added to alternatingList in Line 11 are rejecting, as this is tested in the conditional. We add elements to sameAccList in Lines 15 and 17. Cycles added in Line 17 are clearly accepting by the conditional, and those added in Line 15 are accepting by Lemma B.45. In Line 15, no new node of the Zielonka DAG is added to sameAccList. In Line 15, if a pair $(S_{i,D,j}, n_{i,D,j})$ is added to sameAccList, then the node $n_{i,D,j}$ is a strict descendant of n_i, and incomparable to all nodes corresponding to cycles in sameAccList with some

state in common to $S_{i,D,j}$. Therefore, the list sameAccList satisfies the same conditions as the input. Finally, we show that any rejecting subcycle ℓ of some $\mathcal{S}_i$ is contained in some cycle in alternatingList or in sameAccList. Let ℓ be such a cycle. Then, $\mathsf{col}(\ell) \subseteq D$ for some $D \in \mathsf{maxAltSets}_i$, so ℓ is contained in a subgraph $\mathcal{S}_{i,D,j}$ of the decomposition in SCCs of $\mathcal{S}_{i,D}$. All these SCCs are added to either alternatingList or sameAccList.

Complexity analysis. First, we look to the number of iterations done by the `while`-loop of Line 5 in the function `ComputeChildren2` (Algorithm 4). Let $\mathsf{sameAccList}_0$, $\mathsf{sameAccList}_1, \ldots$ be the sequence of lists produced at each iteration of the `while`-loop. For each state q of $\mathcal{TS}$ and each $i \geq 0$, we consider the sets

$$N_{i,q} = \{n \text{ node of } \mathcal{Z}\text{-}\mathsf{DAG}_{\mathcal{F}} \mid (\mathcal{S}, n) \in \mathsf{sameAccList}_i \text{ and } q \in \mathcal{S}\}.$$

By the properties satisfied by the output of `FindInterestingSubcycles` (Algorithm 5), each $N_{i,q}$ is an antichain for the ancestor relation, and if $n' \in N_{i+1,q}$, then n' is a strict descendant of some node in $N_{i,q}$. We obtain:

- The number of iterations of the `while`-loop of Line 5 is at most the height of $\mathcal{Z}\text{-}\mathsf{DAG}_{\mathcal{F}}$ times $|\mathcal{TS}|$.
- The size of the input passed to `FindInterestingSubcycles` at each call is at most $|\mathcal{Z}\text{-}\mathsf{DAG}_{\mathcal{F}}| \times |\mathcal{TS}|$.

We study now the complexity of the procedure `FindInterestingSubcycles`. It uses three nested `for`-loops, doing, respectively, the following number of iterations:

i) k, for k the size of the input,

ii) the number of subsets in $\texttt{MaxAltSubsets}(C, \mathcal{F})$, for some $C \subseteq \Gamma$,

iii) the number of strongly connected components of decomposition into SCCs of a subgraph of $\mathcal{TS}$.

By Lemma B.40, the number of subsets in $\texttt{MaxAltSubsets}(C, \mathcal{F})$ is polynomial in the size of the Zielonka DAG. The number of SCCs of a subgraph of $\mathcal{TS}$ is at most $|\mathcal{TS}|$.

Inside each `for`-loop it performs:

- Some unions, variable assignments and other basic operations.
- A decomposition into SCCs (Line 8).
- A computation of a maximal node in $\mathcal{Z}\text{-}\mathsf{DAG}_{\mathcal{F}}$ containing a given subset (Line 13).
- In Line 14, we inspect the elements $(\mathcal{S}', n') \in \mathsf{sameAccList}$ of $\mathcal{S}' \cap \mathcal{S}_{i,D,j} \neq \emptyset$. This requires a further `for`-loop, which performs polynomially many operations, as the size of sameAccList remains polynomial during the entire execution.

We conclude that `FindInterestingSubcycles` terminates in polynomial time in the size of the input.

B.5 Simplifying automata with duplicated edges

Given an automaton $\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta, W)$ we say that it has *duplicated edges* if there are some pair of states $q, q' \in Q$ and two different transitions between them labelled with the same input letter: $q \xrightarrow{a:\alpha} q'$, $q \xrightarrow{a:\beta} q'$.

As commented in Remark II.47, the construction of the ZT-HD-Rabin-automaton we have presented potentially introduces duplicated edges, which can be seen as an undesirable property (even if some automata models such as the HOA format [Bab+15] allow them). We show next that we can always derive an equivalent automaton without duplicated edges. Intuitively, in the Rabin case, if we want to merge two transitions having as output letters α and β, we add a fresh letter $(\alpha\beta)$ to label the new transition. For each Rabin pair, this new letter will simulate the best of either α or β depending upon the situation.

Proposition B.46 (Simplification of automata)**.**

Let $\mathcal{A}$ be a Muller (resp. Rabin) automaton presenting duplicated edges. There exists a Muller (resp. Rabin) automaton $\mathcal{A}'$ on the same set of states without duplicated edges such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$. Moreover, if $\mathcal{A}$ is history-deterministic, $\mathcal{A}'$ can be chosen history-deterministic. In the Rabin case, the number of Rabin pairs is also preserved.

Proof. For the Rabin case, let $\mathcal{A}'$ be an automaton that is otherwise as $\mathcal{A}$ except that instead of the transitions Δ of $\mathcal{A}$ it only has one a-transition $q \xrightarrow{a:x} q' \in \Delta'$ (with a fresh colour x per transition) per state-pair q, q' and letter $a \in \Sigma$. That is, $\Delta' = \{(q, a, x_j, q') : (q, a, y, q') \in \Delta \text{ for some } y\}$. The new Rabin condition $\{(\mathfrak{g}'_1, \mathfrak{r}'_1), \dots, (\mathfrak{g}'_r, \mathfrak{r}'_r)\}$ is defined as follows. For each transition $q \xrightarrow{a:x} q'$:

- $x \in \mathfrak{g}'_i$ if $q \xrightarrow{a:y} q' \in \Delta$ for some $y \in \mathfrak{g}_i$,
- $x \in R'_i$ if for all $q \xrightarrow{a:y} q' \in \Delta$, $y \in \mathfrak{r}_i$.

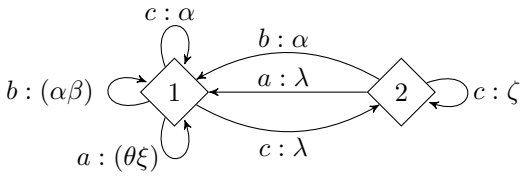
We claim that $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$. Indeed, if $u \in \mathcal{L}(\mathcal{A})$, as witnessed by some run ρ and a Rabin pair $(\mathfrak{g}_i, \mathfrak{r}_i)$, then the corresponding run ρ' in $\mathcal{A}'$ over u is also accepted by the Rabin pair $(\mathfrak{g}'_i, \mathfrak{r}'_i)$: the transitions of $\mathsf{Inf}(\rho) \cap \mathfrak{g}_i$ induce transitions of $\mathsf{Inf}(\rho') \cap \mathfrak{g}'_i$ and the fact that $\mathsf{Inf}(\rho) \cap \mathfrak{r}_i = \emptyset$ guarantees that $\mathsf{Inf}(\rho') \cap \mathfrak{r}'_i = \emptyset$.

Conversely, if $u \in \mathcal{L}(\mathcal{A}')$ as witnessed by a run ρ' and Rabin pair $(\mathfrak{g}'_i, \mathfrak{r}'_i)$, then there is an accepting run ρ over u in $\mathcal{A}$: such a run can be obtained by choosing for each transition $q \xrightarrow{a:x} q'$ of ρ' where $x \in \mathfrak{g}'_i$ a transition $q \xrightarrow{a:y} q' \in \Delta$ such that $y \in \mathfrak{g}_i$, which exists by definition of $\mathcal{A}'$, for each transition $q \xrightarrow{a:x} q'$ where $x \notin \mathfrak{g}_i \cup \mathfrak{r}_i$ a transition $q \xrightarrow{q,y} q' \in \Delta$ such that $y \notin \mathfrak{r}_i$, which also exists by definition of $\mathcal{A}'$, and for other transitions $q \xrightarrow{a:x} q'$ (that is, those for which $x \in \mathfrak{r}'_i$) an arbitrary transition $q \xrightarrow{a:y} q' \in \Delta$. Since ρ' is accepting, we have $\mathsf{Inf}(\rho') \cap \mathfrak{g}_i \neq \emptyset$ and $\mathsf{Inf}(\rho) \cap \mathfrak{r}_i = \emptyset$, that is, ρ is also accepting.

For the Muller case, the argument is even simpler. As above, we consider $\mathcal{A}'$ that is otherwise like $\mathcal{A}$ except that instead of the transitions Δ of $\mathcal{A}$, it only has one a-transition $q \xrightarrow{a:x} q' \in \Delta'$ (with a fresh colour per transition) per state-pair q, q' and the accepting condition is defined as follows. A set of transitions T is accepting if and only if for each $t = q \xrightarrow{a:x} q' \in T$ there is a non-empty set $S_t \subseteq \{q \xrightarrow{a:y} q' \in \Delta\}$ such that $\bigcup_{t \in T} S_t$ is accepting in $\mathcal{A}$. In other words, a set of transitions in $\mathcal{A}'$ is accepting if for each transition we can choose a non-empty subset of the original transitions in $\mathcal{A}$ that form an accepting run in $\mathcal{A}$.

We claim that $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$. Indeed if $u \in \mathcal{L}(\mathcal{A})$, as witnessed by some run ρ, the run ρ' that visits the same sequence of states in $\mathcal{A}'$ is accepting as witnessed by the transitions that occur infinitely often in ρ.

Conversely, assume $u \in \mathcal{L}(\mathcal{A}')$, as witnessed by a run ρ' and a non-empty subset S_t for each transitions t that occurs infinitely often in ρ' such that $\bigcup_{t \in \mathsf{Inf}(\rho)} S_t$ is accepting in



✦ **Figure 54.** The simplified ZT-HD-Rabin-automaton.

$\mathcal{A}$. Then there is an accepting run ρ over u in $\mathcal{A}$ that visits the same sequence of states as ρ' and chooses instead of a transition $t \in \mathsf{Inf}(\rho)$ each transition in S_t infinitely often, and otherwise takes an arbitrary transition. The set of transitions ρ visits infinitely often is exactly $\bigcup_{t\in\mathsf{Inf}(\rho)} S_t$, and is therefore accepting.

Finally, observe that in both cases, if $\mathcal{A}$ is HD, then the automaton $\mathcal{A}'$ without duplicate edges is also HD since $\mathcal{A}'$ is obtained from $\mathcal{A}$ by merging transitions. Indeed, the resolver r of $\mathcal{A}$ induces a resolver r' for $\mathcal{A}'$ by outputting the unique transition with the same letter and state-pair as r. By the same argument as above, the run induced by r' is accepting if and only if the run induced by r is. ◀

Example B.47.

The ZT-HD-Rabin-automaton from Figure 14 has duplicated transitions. In Figure 54 we present an equivalent HD Rabin automaton without duplicates. For this, we have merged the self loops in state 1 labelled with a and b respectively. We have added the output colours $(\alpha\beta)$ and $(\theta\xi)$. The new Rabin pairs are given by:

$$\mathfrak{g}'_\beta = \{\beta, (\alpha\beta)\}, \quad \mathfrak{r}'_\beta = \{\alpha, \lambda, \xi, \zeta\},$$
$$\mathfrak{g}'_\lambda = \{\lambda\}, \quad \mathfrak{r}'_\lambda = \{\alpha, \beta, (\alpha\beta), \theta\}.$$

C Half-positionality of ω-regular languages: Full proofs for Section V.4.2

In this Appendix, we include full proofs for all the propositions and lemmas appearing in Section V.4.2. In order to be able to tackle these proofs and formalise them in the simplest and cleanest possible way, we first introduce some technical artillery that will come in handy to deal with the structure of layered total preorders of signature automata. We begin by introducing nice transformations of automata equipped with a priority-faithful relation in Section C.1. Then, in Section C.2 we give the full details of the induction constructing a structured signature automaton for a half-positional language.

C.1 Nice transformations of automata

To recursively build a structured signature automaton, we will apply a sequence of transformations to a given d-signature automaton $\mathcal{A}$, by removing states and adding or redirecting edges in such a way that relations $\sim_x$ are preserved in a strong sense formalised in this section. Before presenting the proofs of the results from Section V.4.2, we introduce some notations and prove technical lemmas that will come in handy to reason about (structured) signature automata.

Global hypothesis in Subsection C.1.

We suppose that all automata are complete and semantically deterministic. In particular, we suppose that they have a single initial state.

Automata with a common subautomaton

Let $\mathcal{A}$ be a semantically deterministic automaton with states $Q_{\mathcal{A}}$, and let $\sim_{\mathcal{A}}$ be the congruence given by the equality of residuals. We note that if $\mathcal{B}$ is an automaton with states $Q_{\mathcal{B}} \subseteq Q_{\mathcal{A}}$, relation $\sim_{\mathcal{A}}$ induces an equivalent relation over $Q_{\mathcal{B}}$ (which, in general, is not a congruence nor coincides with the equality of residuals of $\mathcal{B}$).

✦ **Lemma C.1** (Automata preserving the structure of residuals)**.** *Let $\mathcal{A}$ be a semantically deterministic parity automaton with states $Q_{\mathcal{A}}$ and let $\mathcal{B}$ be a parity automaton with states $Q_{\mathcal{B}} \subseteq Q_{\mathcal{A}}$. Assume that $\sim_{\mathcal{A}}$ is a congruence over $\mathcal{B}$ and that $\mathcal{B}/_{\sim_{\mathcal{A}}} = \mathcal{A}/_{\sim_{\mathcal{A}}}$. Let $\mathcal{A}'$ be a subautomaton of both $\mathcal{A}$ and $\mathcal{B}$. If a word $w \in \Sigma^\omega$ admits an accepting run in $\mathcal{B}$ that eventually remains in $\mathcal{A}'$, then w is also accepted by $\mathcal{A}$.*

Proof. Let

$$q_0 \xrightarrow{w_0} q_1 \xrightarrow{w_1} q_2 \xrightarrow{w_2} \dots \xrightarrow{w_{k-1}} q_k \xrightarrow{w_k} q_{k+1} \xrightarrow{w_{k+1}} \dots$$

be an accepting run over w in $\mathcal{B}$ such that the suffix from q_k is contained in $\mathcal{A}'$ (meaning that both the states and the transitions used are part of $\mathcal{A}'$). We consider the projection of the prefix of size k of the run in the quotient automaton $\mathcal{B}/_{\sim_{\mathcal{A}}} = \mathcal{A}/_{\sim_{\mathcal{A}}}$. By Lemma V.4, there is a run over $w_0 \dots w_{k-1}$ in $\mathcal{A}$, $p_0 \xrightarrow{w_0} p_1 \xrightarrow{w_1} \dots p_k$ whose projection over the quotient automaton coincides with the previous one. Therefore, $p_k \sim_{\mathcal{A}} q_k$, that is, $\mathcal{L}(\mathcal{A}_{p_k}) = \mathcal{L}(\mathcal{A}_{q_k})$. Since $w_{k+1}w_{k+2}\dots$ admits an accepting run from q_k in $\mathcal{A}$, it also admits an accepting run from p_k, and w is accepted by $\mathcal{A}$. ◀

Nice transformations of automata

For $x \in \mathbb{N}$, we denote by $\mathcal{A}|_{\geq x}$ the subautomaton of $\mathcal{A}$ induced by the set of transitions using a priority $\geq x$.

Definition C.2 (Nice transformation at level x)**.**

Let $\mathcal{A}$ be a semantically deterministic parity automaton over Σ with states Q, let x be a priority, and let $\sim$ be a $[0, x-1]$-faithful congruence over $\mathcal{A}$. Let $\mathcal{A}'$ be a parity automaton over Σ with states $Q' \subseteq Q$. We denote $\sim$ the induced relation over Q'. We say that $\mathcal{A}'$ is a *$\sim$-nice transformation of $\mathcal{A}$ at level x* if:

- $\sim$ is a $[0, x-1]$-faithful congruence over $\mathcal{A}'$ and $\mathcal{A}/_{\sim_{\leq x-1}} = \mathcal{A}'/_{\sim_{\leq x-1}}$,
- $\sim_{\mathcal{A}}$ is a congruence over $\mathcal{A}'$ and $\mathcal{A}'/_{\sim_{\mathcal{A}}} = \mathcal{A}/_{\sim_{\mathcal{A}}}$, and
- $\mathcal{A}'|_{\geq x+1}$ coincides with the subautomaton of $\mathcal{A}|_{\geq x+1}$ induced by the states in Q'.

Intuitively, if $\mathcal{A}'$ is a nice transformation of $\mathcal{A}$ at level x, it means that the only relevant modifications applied to $\mathcal{A}$ concern x-transitions. The structure of the quotient automaton for priorities $<x$ is left unchanged, and so is the acceptance of runs that eventually only produce priorities $>x$.

✦ Remark C.3. *We note that if $\sim$ is an equivalence relation that refines $\sim_{\mathcal{A}}$, then the second item of Definition C.2 is implied by the first one. This is in particular the case if $\sim$ is an equivalent relation $\sim_x$ of a d-signature automaton, for $0 \leq x \leq d$.*

Lemma C.4 (Preservation of classes and priorities in nice transformations)**.**

Let $\mathcal{A}$ be a semantically deterministic parity automaton equipped with a $[0, x-1]$-faithful congruence $\sim$. Suppose that $\mathcal{A}$ is deterministic over $>x$-transitions. Let $\mathcal{A}'$ be a $\sim$-nice transformation of $\mathcal{A}$ at level x. If $q \sim q'$ are two states of $\mathcal{A}'$ such that there is path $q \overset{w:y}{\rightsquigarrow} p$ in $\mathcal{A}$, then a path $q' \overset{w:y'}{\rightsquigarrow} p'$ in $\mathcal{A}'$ satisfies:

- $p \sim p'$,
- if $y < x$, then $y' = y$, and
- if $y \geq x$, then $y' \geq x$.

Proof. The equivalence $p \sim p'$ follows from the fact that $\sim$ is a congruence in $\mathcal{A}'$ and $\mathcal{A}'/_{\sim_{\mathcal{A}}} = \mathcal{A}/_{\sim_{\mathcal{A}}}$. For $y \leq x-1$ or $y' \leq x-1$, the equality $y' = y$ follows from the equality of the $\leq(x-1)$-quotient automata. This directly implies the third item. ◀

✦ **Lemma C.5.** *Let $\mathcal{A}$ be a parity automaton that admits a $[0, x]$-faithful congruence $\sim$. If a word $w \in \Sigma^\omega$ admits a run such that the minimal priority produced infinitely often is $y \leq x$, then the minimal priority produced infinitely often by any run over w is y. In particular, if y is odd, w is rejected with priority y.*

Proof. Let $q_0 \xrightarrow{w_1:y_1} q_1 \xrightarrow{w_2:y_2} \dots$ and $q'_0 = q_0 \xrightarrow{w_1:y'_1} q'_1 \xrightarrow{w_2:y'_2} \dots$ be two runs over the same word in $\mathcal{A}$ ($q_0 = q'_0$ being the initial state of $\mathcal{A}$). Since $\sim$ is a congruence, we obtain by induction that $q_i \sim q'_i$ for every i. Moreover, as, for $y \leq x$, y-transitions act uniformly over $\sim$-classes, each time that $y_i \leq x$, we have that $y'_i = x_i$. ◀

Lemma C.6 (Nice transformations preserve acceptance of most words)**.**

Let $\mathcal{A}$ be a semantically deterministic parity automaton equipped with a $[0, x-1]$-faithful congruence $\sim$. Let $\mathcal{A}'$ be a $\sim$-nice transformation of $\mathcal{A}$ at level x. We have:

- A word $w \in \Sigma^\omega$ can be accepted with an even priority $y < x$ in $\mathcal{A}'$ if and only if w can be accepted with priority y in $\mathcal{A}$.
- A word $w \in \Sigma^\omega$ is rejected with an odd priority $y < x$ in $\mathcal{A}'$ if and only if w is rejected with priority y in $\mathcal{A}$.
- If a word $w \in \Sigma^\omega$ can be accepted with an even priority $y > x$ in $\mathcal{A}'$, then it is accepted by $\mathcal{A}$.

If moreover $\mathcal{A}$ is homogeneous and deterministic over transitions using priorities $> x$, we have:

- If there is a rejecting run over $w \in \Sigma^\omega$ in $\mathcal{A}'$ producing as minimal priority $y > x$, then w is rejected by $\mathcal{A}$.

Proof. Let $w \in \Sigma^\omega$ be a word accepted with a priority $y < x$ in $\mathcal{A}$ (resp. $\mathcal{A}'$). Then, by Lemma V.58, w is accepted by the quotient automaton $\mathcal{A}\big/_{\sim_{\leq x-1}} = \mathcal{A}'\big/_{\sim_{\leq x-1}}$. Again by Lemma V.58, w is accepted by $\mathcal{A}'$ (resp. $\mathcal{A}$). The second item is obtained using the same argument, combined with Lemma C.5.

The third item is directly implied by Lemma C.1, as $\mathcal{A}'|_{\geq x+1}$ is a subautomaton of $\mathcal{A}|_{\geq x+1}$.

For the last item, let $q_0 \overset{w_0}{\rightsquigarrow} q' \overset{w'}{\rightsquigarrow}$ be a rejecting run over w in $\mathcal{A}'$ such that the suffix $q' \overset{w'}{\rightsquigarrow}$ does not produce any priority $\leq x$ (that is, it is contained in $\mathcal{A}'|_{\geq x+1}$). By determinism and homogeneity, this is the only run over w' from q' in $\mathcal{A}$, and therefore $w' \notin \mathcal{L}(\mathcal{A}_{q'})$. We conclude using the equality $\mathcal{A}'/_{\sim_{\mathcal{A}}} = \mathcal{A}/_{\sim_{\mathcal{A}}}$. ◀

C.2 From half-positionality to structured signature automata: Full proofs for Section V.4.2

We provide all the technical details for the proofs of the propositions appearing in Section V.4.2. We first state a useful simple lemma.

✦ **Lemma C.7.** *Let $W \subseteq \Sigma^\omega$ be half-positional over finite, ε-free Eve-games. Then, for every word $u \in \Sigma^*$, objective $u^{-1}W$ is half-positional over finite, ε-free Eve-games.*

Proof. Any game with vertices V witnessing non-half-positionality of $u^{-1}W$ can be turned into a game witnessing non-half-positionality of W by adding, for every $v \in V$, a fresh vertex v_u and a path $v_u \overset{u}{\rightsquigarrow} v$. ◀

C.2.1 Base case: Total order of residual classes

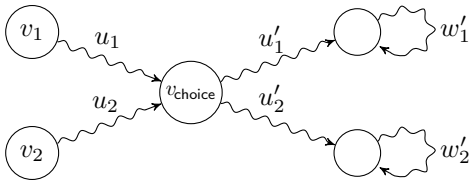
Lemma C.8 (Total order of residual classes)**.**

Let $W \subseteq \Sigma^\omega$ be an ω-regular objective that is half-positional over finite, ε-free Eve-games. Then, $\mathsf{Res}(W)$ is totally ordered by inclusion.

Proof. The proof is almost identical to that of Lemma V.21. We show the contrapositive. Assume that W has two incomparable residuals, ${u_1}^{-1}W$ and ${u_2}^{-1}W$. We consider first the case $u_1 \neq \varepsilon$ and $u_2 \neq \varepsilon$. Take $w_1 \in {u_1}^{-1}W \setminus {u_2}^{-1}W$ and $w_2 \in {u_2}^{-1}W \setminus {u_1}^{-1}W$. By Lemma I.17, we can take these words of the form $w_i = u_i'(w_i')^\omega$, with $u_i', w_i' \in \Sigma^+$, for $i = 1, 2$. We have

$$u_1 w_1 \in W, \quad u_1 w_2 \notin W,$$
$$u_2 w_1 \notin W, \quad u_2 w_2 \in W.$$

Consider the ε-free, finite, Eve-game $\mathcal{G}$ represented in Figure 55. Eve wins $\mathcal{G}$ from v_1 and v_2: if a play starts in v_i, for $i = 1, 2$, she just has to take the path labelled $u_i'(w_i')^\omega$ from v_{choice}. However, she cannot win from both v_1 and v_2 using a positional strategy. Indeed, such positional strategy would choose one path $v_{\mathsf{choice}} \overset{u_i(w_i')}{\rightsquigarrow}$, and the play induced when starting from v_{1-i} would be losing.



✦ **Figure 55.** A game $\mathcal{G}$ in which Eve cannot play optimally using positional strategies if $\mathsf{Res}(W)$ is not totally ordered.

Finally, we take care of the case in which $[u_1] = \{\varepsilon\}$ (symmetric for u_2). In that case, we cannot take $u_1 \neq \varepsilon$. We remark that, since $[u_1] \neq [u_2]$, we can take $u_2 \neq \varepsilon$. We consider the game from Figure 55 in which we simply remove vertex v_1. This game is ε-free, and Eve can win from both v_2 and v_{choice}, but not positionally. ◀

Global hypothesis in Subsection C.1.

In all the rest of the subsection, we assume that $x \geq 2$.

C.2.2 Safe centrality and relation $\sim_{x-1}$. Proof of Lemma V.71

In this paragraph we give a proof of Lemma V.71. We assume that $x \geq 2$ is an even priority and $\mathcal{A}$ is a deterministic $(x-2)$-structured signature automaton with initial state q_{init}.

Lemma V.71 ($(<x)$-safe centralisation)**.**

There exists a $(x-2)$-structured signature automaton $\mathcal{A}'$ equivalent to $\mathcal{A}$ which is:

- deterministic over transitions with priority different from $x-1$,
- homogeneous,
- history-deterministic, and
- $(<x)$-safe centralised.

Moreover, $\mathcal{A}'$ can be obtained in polynomial time from $\mathcal{A}$ and $|\mathcal{A}'| \leq |\mathcal{A}|$.

Hypothesis. During the proof, we will lose the determinism of $\mathcal{A}$. However, in all the subsection we will maintain the three first required properties. In the statements of all lemmas, $\mathcal{A}$ will stand for an $(x-2)$-structured signature automaton that is:

- deterministic over transitions with priority different from $x-1$,
- homogeneous, and
- history-deterministic.

Saturation. We say that an automaton $\mathcal{A}'$ is $(x-1)$-*saturated* if for every state q and letter $a \in \Sigma$, if a transition $q \xrightarrow{a:x-1} p$ exists in $\mathcal{A}'$, then $q \xrightarrow{a:x-1} p'$ appears in $\mathcal{A}'$ for all $p' \sim_{x-2} p$.[2] The $(x-1)$-saturation of $\mathcal{A}$ is the automaton obtained by adding all those transitions.

✦ Remark C.9. *The $(x-1)$-saturation of $\mathcal{A}$ is homogeneous and deterministic over transitions with priority different from $x-1$.*

✦ **Lemma C.10.** *The $(x-1)$-saturation of $\mathcal{A}$ is in normal form.*

Proof. We use the characterisation of normal form given in Theorem II.14. Let $\mathcal{A}'$ be the $(x-1)$-saturation of $\mathcal{A}$. The property of Theorem II.14 is satisfied for transitions already appearing in $\mathcal{A}$, as $\mathcal{A}$ is assumed to be in normal form. Let $q \xrightarrow{a:x-1} p$ be a transition added by the saturation process, and let $q \xrightarrow{a:x-1} p'$ be a transition in $\mathcal{A}$ with $p \sim_{x-2} p'$. By Item 5 of the definition of structured signature automaton, there is a path $p \overset{u:>x-2}{\rightsquigarrow} p'$ in $\mathcal{A}$. By normality of $\mathcal{A}$, there are also paths $p' \overset{u_1:x-1}{\rightsquigarrow} q$ and $p' \overset{u_2:x-2}{\rightsquigarrow} q$. We obtain the two desired paths in $\mathcal{A}'$:

$$p \overset{u:>x-2}{\rightsquigarrow} p' \overset{u_1:x-1}{\rightsquigarrow} q\,, \quad \text{and} \quad p \overset{u:>x-2}{\rightsquigarrow} p' \overset{u_2:x-2}{\rightsquigarrow} q. \qquad \blacktriangleleft$$

The following lemma states that $(x-1)$-saturation is a $\sim_{x-2}$-nice transformation at level $x-1$, so Lemmas C.6 and C.4 can be applied. We recall that $\sim_{x-2}$ refines $\sim_{\mathcal{A}}$, so congruence of $\sim_{\mathcal{A}}$ is implied by that of $\sim_{x-2}$.

✦ **Lemma C.11.** *The $(x-1)$-saturation of $\mathcal{A}$ is a $\sim_{x-2}$-nice transformation of $\mathcal{A}$ at level $x-1$.*

Proof. Let $\mathcal{A}'$ be the $(x-1)$-saturation of $\mathcal{A}$. It is immediate that $\mathcal{A}|_{\geq x} = \mathcal{A}'|_{\geq x}$. Moreover, the restriction of $\mathcal{A}$ and $\mathcal{A}'$ to transitions using priorities $\leq x-2$ coincides, so for each $0 \leq y \leq x-2$, relation $\sim_{x-2}$ is a congruence for y-transitions in $\mathcal{A}'$ (and therefore these transitions act uniformly by determinism). The congruence for transitions using priority $>(x-2)$ is preserved as we have only added $(x-1)$-transitions that go to the same $\sim_{x-2}$-class. As $\sim_{x-2}$ refines $\sim_{\mathcal{A}}$, the latter relation is also a congruence in $\mathcal{A}'$ and $\mathcal{A}/_{\sim_{\mathcal{A}}} = \mathcal{A}'/_{\sim_{\mathcal{A}}}$. $\blacktriangleleft$

✦ **Lemma C.12.** *The $(x-1)$-saturation of $\mathcal{A}$ recognises $\mathcal{L}(\mathcal{A})$. Moreover, it is history-deterministic, homogeneous and deterministic over transitions using priorities different from $x-1$.*

[2] We note that this definition slightly differs from the definition of 1-saturation used in the warm-up (Section V.3.4). In particular, the definition of the warm-up does not preserve homogeneity. We allow ourselves these small disagreements of definitions for the sake of clarity in the presentation in each respective subsection.

Proof. Let $\mathcal{A}'$ be the $(x-1)$-saturation of $\mathcal{A}$. We have already noted that it is homogeneous and deterministic over transitions using priority different from $x-1$ (Remark C.9). If $w \in \Sigma^\omega$ is accepted by $\mathcal{A}'$ (resp. by $\mathcal{A}$), it is either accepted with an even priority $y < x-1$ or $y > x-1$. In the first case, since $\mathcal{A}'$ is a $\sim_{x-2}$-nice transformation at level $x-1$, Lemma C.6 allows us to conclude. In the second case, it suffices to apply Lemma C.1.

History-determinism of $\mathcal{A}'$ is clear: one can use a resolver for $\mathcal{A}$. ◀

Redundant safe components. From now on, we suppose that $\mathcal{A}$ is $(x-1)$-saturated.

We say that a $(<x)$-safe component S of $\mathcal{A}$ is *redundant* if there is $q \in S$ and $q' \sim_{x-2} q$, $q' \notin S$, such that $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(q')$. We note that, by normality of $\mathcal{A}$, there are no $(\geq x)$-transitions entering in S; that is, there are no transitions $p \xrightarrow{a:\geq x} q$ with $p \notin S$ and $q \in S$.

✦ Remark C.13. *Automaton $\mathcal{A}$ is $(<x)$-safe centralised if and only if it does not contain any redundant $(<x)$-safe component.*

✦ **Lemma C.14.** *If $\mathcal{A}$ contains some redundant $(<x)$-safe component, we can find one of them in polynomial time.*

Proof. The computation of the $(<x)$-safe components of $\mathcal{A}$ can be done by simply a decomposition in SCC of $\mathcal{A}|_{\geq x}$. For each pair of states $q \sim_{x-2} q'$ in different $(<x)$-safe components we just need to check the inclusion $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(q')$, which can be done in polynomial time (Proposition I.15). ◀

✦ **Lemma C.15.** *Let S be a redundant $(<x)$-safe component of $\mathcal{A}$, and let S' be a different $(<x)$-safe component such that there are $q_0 \in S$ and $q_0' \in S'$, with $q_0 \sim_{x-2} q_0'$ and $\mathsf{Safe}_{<x}(q_0) \subseteq \mathsf{Safe}_{<x}(q_0')$. Then, for each $q \in S$ there is $q' \in S'$ such that $q \sim_{x-2} q'$ and $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(q')$.*

Proof. For each $q \in S$, pick $u \in \Sigma^*$ such that $q_0 \overset{u:\geq x}{\rightsquigarrow} q$. We let q' be such that $q_0' \overset{u}{\rightsquigarrow} q'$. Since $u \in \mathsf{Safe}_{<x}(q_0) \subseteq \mathsf{Safe}_{<x}(q_0')$, this latter path produces priority $\geq x$ and, by normality, q' is in S'. As $\sim_{x-2}$ is a congruence for $(\geq x-2)$-transitions, $q' \sim_{x-2} q$. By monotonicity for safe languages (Lemma V.66), we also have $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(q')$. ◀

Removing redundant safe components. For now on, fix S to be a redundant $(<x)$-safe component of $\mathcal{A}$, and S' a different $(<x)$-safe component as in the previous lemma. For each $q \in S$, we let $f(q) \in S'$ such that $q \sim_{x-2} f(q)$ and $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(f(q))$. We extend f to all of Q by setting it to be the identity over $Q \setminus S$.

We define the automaton $\mathcal{A}'$ as follows:

- The set of states is $Q' = Q \setminus S$.
- The initial state is $f(q_{\mathsf{init}})$.
- For each $p \in Q'$, if $p \xrightarrow{a:y} q$ is a transition in $\mathcal{A}$, we let $p \xrightarrow{a:y} f(q)$ in $\mathcal{A}'$.

We note that, if $q \notin S$, all transitions $p \xrightarrow{a:y} q$ in $\mathcal{A}$ are left unchanged. In particular, the $(\geq x)$-transitions of $\mathcal{A}'$ are the restriction of those appearing in $\mathcal{A}$, $\mathcal{A}'$ is $(x-1)$-saturated, and $(<x)$-transitions in $\mathcal{A}$ not entering in S appear in $\mathcal{A}'$ too. We say that transitions $p \xrightarrow{a:y} q$ of $\mathcal{A}$ such that $q \in S$ have been *redirected in $\mathcal{A}'$*.

✦ **Lemma C.16.** *Automaton $\mathcal{A}'$ is a $\sim_{x-2}$-nice transformation of $\mathcal{A}$ at level $x-1$.*

Proof. We have that $\mathcal{A}'|_{\geq x}$ is the subautomaton of $\mathcal{A}|_{\geq x}$ induced by states in Q'. We show that $\sim_{x-2}$ is $[0, x-2]$-faithful in $\mathcal{A}'$. Transitions that have not been redirected satisfy the congruence requirements, as they satisfy them in $\mathcal{A}$. Let $p \xrightarrow{a:y} q$ and $p' \xrightarrow{a:y'} q'_0$ be two transitions in $\mathcal{A}'$ such that $y \leq x-2$, $p \sim_{x-2} p'$ and such that the second transition has been redirected from $p' \xrightarrow{a:y'} q'$ (the first transition is possibly a redirected one too). By the congruence property in $\mathcal{A}$, we have that $y' = y$ and $q \sim_{x-2} q'$. Since $q' \sim_{x-2} q'_0$, we conclude by transitivity. The equality $\mathcal{A}'\big/_{\leq x-1}^{\sim_{x-2}} = \mathcal{A}\big/_{\leq x-1}^{\sim_{x-2}}$ simply follows from the fact that redirected transitions have been defined preserving the $\sim_{x-2}$-classes.

As $\sim_{x-2}$ refines $\sim_{\mathcal{A}}$, the latter relation is also a congruence in $\mathcal{A}'$ and $\mathcal{A}/_{\sim_{\mathcal{A}}} = \mathcal{A}'/_{\sim_{\mathcal{A}}}$. ◀

Lemma C.17 (Correctness of the removal of redundant components)**.**

For every state $q' \in Q'$, we have $\mathcal{L}(\mathcal{A}'_{q'}) = \mathcal{L}(\mathcal{A}_{q'})$. In particular, these automata are equivalent. Moreover, automaton $\mathcal{A}'$ is deterministic over transitions with priority different from $x-1$, homogeneous and history-deterministic.

Proof. The fact that $\mathcal{A}'$ is deterministic over transitions with priority different from $x-1$ and homogeneous is immediate from its definition. We show the equality of languages for the initial state. The proof is identical for a different state.

The inclusion $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$ directly follows from Lemma C.6.

We describe a sound resolver witnessing $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$ and history-determinism. Take a sound resolver r in $\mathcal{A}$, let $w \in \Sigma^\omega$, and write

$$\rho = p_0 \xrightarrow{w_0} p_1 \xrightarrow{w_1} \dots$$

for the run in $\mathcal{A}$ induced by r over w. We will build a resolver r' in $\mathcal{A}'$ satisfying the property that the run induced over w, $\rho' = p'_0 \xrightarrow{w_0} p'_1 \xrightarrow{w_1} \dots$ is in one of the following (non-excluding) cases:

- **a)** produces priorities $< x-1$ infinitely often,
- **b)** eventually produces only priorities $\geq x$,
- **c)** $p_i \sim_{x-2} p'_i$ and $\mathsf{Safe}_{<x}(p_i) \subseteq \mathsf{Safe}_{<x}(p'_i)$ for every i sufficiently large.

✧ Claim C.17.1. *A resolver r' satisfying the property above accepts all words in $\mathcal{L}(\mathcal{A})$.*

Proof. Suppose that $w \in \mathcal{L}(\mathcal{A})$, that is, the run ρ induced by r is accepting. Let ρ' be the run induced over w by r' in $\mathcal{A}'$. We distinguish two cases, according to the priorities produced by the run ρ in $\mathcal{A}$:

If ρ produces priorities $< x-1$ infinitely often. Then Lemma C.6 allows us to conclude.

If ρ eventually only produces priorities $\geq x-1$. Then, the two first items of Lemma C.6 tells us that ρ' eventually only produces priorities $\geq x-1$ too (so we are not in Case a). We show that ρ' eventually only produces priorities $> x-1$; the last item of Lemma C.6 allows us to conclude (we recall that $\mathcal{A}'$ is a $\sim_{x-2}$-nice transformation at level $x-1$). If we are in Case b, this property is trivially satisfied. Suppose that we are in Case c, and let $k > 0$ be such that the suffix of ρ from

q_k only produces priorities $\geq x$ and such that $\mathsf{Safe}_{<x}(p_i) \subseteq \mathsf{Safe}_{<x}(p'_i)$ for $i \geq k$. Therefore, there is a run over $w_k w_{k+1} \dots$ from p'_k producing exclusively priorities $\geq x$. By determinism over transitions with priority $\geq x$, this run is the one induced by r'. ◁

We finally show how to construct a resolver with this property. We let $p'_0 = f(p_0)$, and assume ρ' constructed up to p'_i satisfying that $p'_i \sim_{x-2} p_i$.

If there is a transition $p'_i \xrightarrow{w_i:y} p'_{i+1}$ with $y \neq x-1$, then we take this one (there is no other option), which satisfies $p_{i+1} \sim_{x-2} p'_{i+1}$ by Lemma C.16. Moreover, if $y \geq x$ and $\mathsf{Safe}_{<x}(p_i) \subseteq \mathsf{Safe}_{<x}(p'_i)$, then $\mathsf{Safe}_{<x}(p_{i+1}) \subseteq \mathsf{Safe}_{<x}(p'_{i+1})$ by Lemma V.66. If there is a transition $p'_i \xrightarrow{w_i:x-1}$, we take $p'_{i+1} = f(p_{i+1})$ (this transition exists in $\mathcal{A}'$ by $x-1$-saturation, as $p'_i \sim_{x-2} p_i$).

We show that this resolver satisfies the desired property. Suppose that we are not in the two first cases, that is, ρ' eventually only produces priorities $\geq x-1$, and it produces priority $x-1$ infinitely often. Take a suffix $p'_k \xrightarrow{w_k:y_k} p'_{k+1} \xrightarrow{w_{k+1}:y_{k+1}} \dots$ of ρ' such that no priority $< x-1$ is produced and such that $y_k = x-1$. Then, by definition of the transitions using $x-1$ chosen by the resolver, $p'_{k+1} = f(p_{k+1})$, so $\mathsf{Safe}_{<x}(p_{k+1}) \subseteq \mathsf{Safe}_{<x}(p'_{k+1})$. We conclude by induction, as transitions taken by the resolver using priorities $\geq x-1$ preserve the inclusion of $(<x)$-safe languages. ◀

Transformation preserves being a structured signature automaton. To be able to finish the proof of Lemma 5, we just need to show that $\mathcal{A}'$ is a $(x-2)$-structured signature automaton. We give some technical lemmas that will help us show this.

✦ **Lemma C.18.** *Let q and p be two states of $\mathcal{A}'$. There is a path $q \overset{w:x-1}{\rightsquigarrow} p$ in $\mathcal{A}$ if and only if there is a path $q \overset{w:x-1}{\rightsquigarrow} p$ in $\mathcal{A}'$.*

Proof. If a path $q \overset{w:x-1}{\rightsquigarrow} p$ appears in $\mathcal{A}'$, the very same path also exists in $\mathcal{A}$.

Suppose now that a path $\rho = q \overset{w:x-1}{\rightsquigarrow} p$ exists in $\mathcal{A}$. Let S be the $<x$-safe component that has been removed from $\mathcal{A}$. If the path ρ does not cross S, then it also appears in $\mathcal{A}$. Suppose that it enters in S. We remark that, by normality and by the definition of safe component, each time that ρ enters or exists S, it produces priority $x-1$. We consider the last time that ρ enters and exists S:

$$q \overset{u_1:\geq x-1}{\rightsquigarrow} q_1 \overset{u_2:\geq x}{\rightsquigarrow} q_2 \xrightarrow{a:x-1} q_3 \overset{u_3:\geq x-1}{\rightsquigarrow} p,$$

with $w = u_1 u_2 a u_3$, $q_3 \notin S$, and the path $q_3 \overset{u_3}{\rightsquigarrow} p$ does not enter S (so it also appears in $\mathcal{A}'$). Consider any run over $u_1 u_2$ from q in $\mathcal{A}'$:

$$q \overset{u_1:\geq x-1}{\rightsquigarrow} q'_1 \overset{u_2:\geq x-1}{\rightsquigarrow} q'_2.$$

As $\sim_{x-2}$ is a $[0, x-2]$-faithful congruence, we have that $q_2 \sim_{x-2} q'_2$. As $\mathcal{A}'$ is $x-1$-saturated, there is a transition $q'_2 \xrightarrow{a:x-1} q_3$. Therefore, we obtain in $\mathcal{A}'$ the path:

$$q \overset{u_1:\geq x-1}{\rightsquigarrow} q'_1 \overset{u_2:\geq x-1}{\rightsquigarrow} q'_2 \xrightarrow{a:x-1} q_3 \overset{u_3:\geq x-1}{\rightsquigarrow} p. \quad ◀$$

✦ **Lemma C.19.** *Let q and p be two states of $\mathcal{A}'$ and let y be any priority. There is a path $q \overset{w:y}{\rightsquigarrow} p$ in $\mathcal{A}$ if and only if there is a path $q \overset{w':y}{\rightsquigarrow} p$ in $\mathcal{A}'$.*

Proof. If $y \geq x$, q and p are in the same $(<x-1)$-safe component. Since the $(<x-1)$-safe components in $\mathcal{A}'$ are safe components in $\mathcal{A}$, the result is clear in this case.

If $y = x-1$, the result is assured by the previous Lemma C.18.

Assume $y < x-1$. Suppose that there is a path $q \overset{w:y}{\rightsquigarrow} p$ in $\mathcal{A}$ (the proof is analogous if we take this path in $\mathcal{A}'$), and let $q \overset{w:y'}{\rightsquigarrow} p'$ be the run over w from q in $\mathcal{A}'$. As $\mathcal{A}'$ is a $\sim_{x-2}$-nice transformation, by Lemma C.4, we have that $y' = y$ and $p \sim_{x-2} p'$. As $\mathcal{A}$ satisfies Item 5 from the definition of a structured signature automaton, there is a path $p' \overset{u:\geq x-1}{\rightsquigarrow} p$ in $\mathcal{A}$. By Lemma C.18, such a path also exists in $\mathcal{A}'$, so we can take $w' = wu$, giving us a path $q \overset{w:y}{\rightsquigarrow} p' \overset{u:>y}{\rightsquigarrow} p$. ◀

Previous lemma tells us, in particular, that for every $(<)y$-safe component $S_i^{<y}$ of $\mathcal{A}$, the intersection of $S_i^{<y}$ with Q' constitute the states of a $(<)y$-safe component in $\mathcal{A}'$. Therefore, automaton $\mathcal{A}'$ inherits the decomposition in $(<)y$-safe component $S_1^{<y}, \ldots, S_{k_y}^{<y}$ from $\mathcal{A}$, for each y; we simply remove those components whose intersection with Q' is empty.

Lemma C.20.

Automaton $\mathcal{A}'$ is a $(x-2)$-structured signature automaton.

Proof. We go through all the conditions of the definition of a $(x-2)$-structured signature automaton. We recall that, by Lemma C.16, the relation $\sim_{x-2}$ is $[0, x-2]$-faithful congruence in $\mathcal{A}'$.

Normal form. We check that $\mathcal{A}'$ satisfies the hypothesis of the characterisation from Theorem II.14. We let $q' \xrightarrow{a:y} p'$ be a transition in $\mathcal{A}'$. If it is not a redirected transition, it exists in $\mathcal{A}$, so we can conclude by normalisation of $\mathcal{A}$ and Lemma C.19. Assume that $q' \xrightarrow{a:y} p'$ is a transition that has been redirected from $q' \xrightarrow{a:y} p$. In particular, $p \sim_{x-2} p'$ and $y < x$. By Item 5 of the definition of structured signature automaton applied to $\mathcal{A}$, there is a path $p' \overset{w:>y}{\rightsquigarrow} p$ in $\mathcal{A}$, and by normality, there is a returning paths $p \overset{u_1:y}{\rightsquigarrow} q$ and $p \overset{u_2:y-1}{\rightsquigarrow} q$. Again, Lemma C.19 allows us to find the desired returning paths in $\mathcal{A}'$.

Item 1. By Lemma C.17, the residuals in $\mathcal{A}'$ correspond to those in $\mathcal{A}$, so preorder $\leq_0$ correspond to their inclusion in $\mathcal{A}'$ too.

Item 2. By the previous remarks, for all y, the $(<y)$-safe components of $\mathcal{A}'$ are obtaining by taking the intersection with those in $\mathcal{A}$. Therefore, odd preorders $\leq_{y-1}$ correspond to the order of $(<y)$-safe components on $\mathcal{A}'$.

Item 3. As $\mathcal{A}'$ is a $\sim_{x-2}$-nice transformation at level $x-1$, for every $y < x-1$ and state q' in $\mathcal{A}'$, $\mathsf{Safe}^{\mathcal{A}'}_{<y}(q') = \mathsf{Safe}^{\mathcal{A}}_{<y}(q')$. Therefore, the preorders at even levels $\leq_y$ correspond to the inclusion of safe languages in $\mathcal{A}'$, as they do in $\mathcal{A}$.

Item 4. Let q and q' be two states in $\mathcal{A}'$ such that $q \sim_y q'$, for $y \leq x-2$ even, and let $q \xrightarrow{a:y'} p$ for $y' \leq y$ and $q' \xrightarrow{a:z} p'$. As $\mathcal{A}'$ is a nice transformation, $y' = z$. If the first of these transitions is not a redirected one, then, by strong congruence of $\leq(x-2)$-priorities in $\mathcal{A}$, neither is the second one, and $p = p'$. Assume that these transitions have been redirected from, $q \xrightarrow{a:y'} p_1$ and $q' \xrightarrow{a:y'} p_1'$. Then, as Item 4 is satisfied in $\mathcal{A}$, $p_1 = p_1'$, so $p = f(p_1) = f(p_1') = p'$.

Item 5. Directly follows from Lemma C.19 and the fact that $\mathcal{A}$ satisfies this property.

Item 6. Follows from the fact that if $q \nsim_{y-1} p$ in $\mathcal{A}'$, then $q \nsim_{y-1} p$ in $\mathcal{A}$; and the equality $\mathsf{Safe}^{\mathcal{A}'}_{<y}(q) = \mathsf{Safe}^{\mathcal{A}}_{<y}(q)$. ◀

Obtaining Lemma 5. We have all the necessary elements to deduce Lemma 5. Using Lemma C.14, we can decide whether $\mathcal{A}$ contains a redundant $(<x)$-safe component in polynomial time. If it contains none, $\mathcal{A}$ is already $(<x)$-safe centralised. While we can find redundant safe components, we remove them applying the transformation described above. This transformation can clearly be done in polynomial time, and by Lemmas C.17 and C.20, the obtained automaton recognises the correct language and preserves all the hypothesis assumed in the induction.

C.2.3 Existence of uniform words and synchronising separating runs

We now provide proofs for Lemmas V.73 and V.74. In Section V.4.2, we derived totality of the order $\leq_x$ in each $\sim_{x-1}$-component from these lemmas (c.f. Lemma V.75).

Hypothesis. In all the subsection we assume that x is an even priority and $\mathcal{A}$ is a $(x-2)$-structured signature automaton with initial state q_{init} that is moreover:

- deterministic over transitions with priority different from $x-1$,
- homogeneous,
- history-deterministic, and
- $(<x)$-safe centralised.

Lemma V.73 (Existence of uniform words)**.**

Let p and q be two states from the same $(<x)$-safe component. There is a word $w \in \Sigma^*$ producing priority x uniformly in $[q]_x$ leading to $[p]_x$.

Words producing priority x uniformly.

Proof. The fact that for such paths we must have $p_1 \sim_x p_2$ follows from the monotonicity of safe languages (Lemma V.66) and the fact that $(\geq x)$-transitions preserve $\sim_{x-1}$-classes.

We let $\{q_1, q_2, \dots, q_k\}$ be an enumeration of the states of $[q]_x$.

✧ Claim 5.1. *For each $q_i \in [q]_x$ there is a word $u_i \in \Sigma^*$ such that $q_i \overset{u_i:x}{\rightsquigarrow} q_i$.*

Proof. Since q_i and q are in the same $(<x)$-safe component, there is a word u_i' such that $q_i \overset{u_i':\geq x}{\rightsquigarrow} q$. By normality, there is a word u_i'' such that $q \overset{u_i'':x}{\rightsquigarrow} q_i$. We just take $u_i = u_i' u_i''$. ◁

We will define k finite words $w_1, w_2 \dots, w_k \in \Sigma^*$ satisfying:

- For $q' \in [q]_x$ and for every $i \leq k$, $q' \overset{w_1 w_2 \dots w_i:\geq x}{\rightsquigarrow} [q]_x$.
- $q_j \overset{w_1 w_2 \dots w_i:x}{\rightsquigarrow} [q]_x$ for every $j \leq i$.

In order to obtain these properties, we just define recursively $w_1 = u_1$ and $w_i = u_j$, for u_j as given by the previous claim, if:

$$q_i \xrightarrow{w_1 w_2 \dots w_{i-1} : \geq x} q_j.$$

Finally, we let $w = w_1 w_2 \dots w_k w$, which first produces priority x when read from any state of $[q]_x$, and then goes to the $\sim_x$-class of p producing priorities $\geq x$. ◀

Synchronising separating runs. We recall (Lemma I.10 from Section I.2) that any history-deterministic parity automaton admits a sound resolver implemented by a finite memory. In the rest of the subsection we fix a sound resolver r for $\mathcal{A}$ implemented by a memory structure $\mathcal{M} = (M, m_0, \mu)$. For simplicity, we will assume that every pair of a state and a memory state (q, m) is reachable using r. It is easy to see that we can get rid of this assumption in the upcoming proof just by ignoring pairs (q, m) that are not reachable.

For $(q, m) \in Q \times M$, we let $(q, m) \xrightarrow{w}_{\mathsf{r}} (q', m')$ be the (unique) run induced by r from q when the memory structure is in state m. We extend notations of the form $(q, m) \xrightarrow[\exists, \mathsf{r}]{w:x} q'$ in the natural way; the previous one means that there exists $u_0 \in \Sigma^*$ such that the induced run of r is $\rho = q_0 \xrightarrow{u_0}_{\mathsf{r}} q$, $\mu(m_0, \rho) = m$ and $q_0 \xrightarrow{u_0 w}_{\mathsf{r}} q'$, producing priority x in the second part of this run.

As $\mathcal{A}$ is deterministic over transitions using priorities $\geq x$, we may omit the subscript r in paths producing no priority $< x$.

Lemma V.74 (Synchronisation of separating runs)**.**

Suppose that $q \sim_{x-1} q'$ and $q \not\leq_x q'$ and let $p \in [q]_{x-1}$. There is a word $w \in \Sigma^+$ such that $[q]_x \xrightarrow[\forall, \mathsf{r}]{w:x-1} [p]_x$ and $[q']_x \xrightarrow[\forall, \mathsf{r}]{w:x} [p]_x$.

Proof. We first show that we can force to produce priority $x-1$ from $[q]_x$, while remaining safe from $[q]_x$.

✧ Claim 5.1. *There is a word $u \in \Sigma^+$ such that for all $s \in [q]_x$:*

$$s \xrightarrow[\forall, \mathsf{r}]{u:x-1} \quad , \qquad \text{and} \qquad [q']_x \xrightarrow[\forall, \mathsf{r}]{u:x} [p]_x.$$

Proof. By definition of the preorder $\leq_x$, there is a word $u_1 \in \Sigma^+$ such that for all $s \in [q]_x$ and $s' \in [q']_x$, $s \xrightarrow[\forall, \mathsf{r}]{u_1:x-1}$ and $s' \xrightarrow{u_1:\geq x}_{\mathsf{r}}$. By normality, we can extend this word so that $q' \xrightarrow{u_1':\geq x}_{\mathsf{r}} q'$; by monotonicity of safe languages all runs from $[q']_x$ reading u_1' go back to $[q']_x$. Applying Lemma 5, we obtain u_2 that produces priority x uniformly in $[q']_x$ and goes to $[p]_x$. We take $u = u_1' u_2$, which satisfies $[q']_x \xrightarrow{u:x}_{\mathsf{r}} [p]_x$, and it produces at least one occurrence of priority $x - 1$ from every state $s \in [q]_x$. By $[0, x-2]$-faithfulness, a run $s \xrightarrow{u}$ only produces priorities $\geq x - 1$, which concludes. ◁

✧ Claim 5.2. *Let $p' \sim_{x-2} q$ and let $m \in M$ be a memory state. Then there is a word $w_{q,m}$ such that:*

$$(q, m) \xrightarrow{w_{q,m}:\geq x-1}_{\mathsf{r}} [p']_x, \quad \text{and} \quad [p']_x \xrightarrow[\forall, \mathsf{r}]{w_{q,m}:x} [p']_x.$$

Proof. We distinguish two cases. First, assume that $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(p')$. In this case, by $<x$-safe centrality of $\mathcal{A}$, q and p' are in the same $<x$-safe component so $q \sim_{x-1} p'$ (and $q \leq_x p'$). Let $p'_{\max}$ be a state in $[p']_{x-1}$ such that $p' \leq_x p'_{\max}$, and maximal with this property. Let $w_1 \in \Sigma^*$ be a word such that $q \xrightarrow{w_1:\geq x} p'_{\max}$ (which exists because these two states are in the same $<x$-safe component). By monotonicity of safe languages, $p' \xrightarrow{w_1:\geq x} p''$ with $p'_{\max} \leq_x p''$. By maximality of $p'_{\max}$, we must have $p'' \sim_x p'_{\max}$. Finally, let $w_2 \in \Sigma^*$ such that $p'' \xrightarrow{w_2:x} p'$ producing priority x uniformly in the class $[p'']_x$ (which exists by Lemma 5). We obtain $q \xrightarrow{w_1:\geq x} p'_{\max} \xrightarrow{w_2:x} [p']_x$ and $[p']_x \xrightarrow{w_1:\geq x} [p'_{\max}]_x \xrightarrow{w_2:x} [p']_x$ as required.

Assume now that $\mathsf{Safe}_{<x}(q) \not\subseteq \mathsf{Safe}_{<x}(p')$. In that case, we can find a word $w_1 \in \mathsf{Safe}_{<x}(p') \setminus \mathsf{Safe}_{<x}(q)$. By Lemma 5, we may assume that it produces priority x uniformly from $[p']_x$ and comes back to this class. Moreover, by faithfulness, it cannot produce priorities $\leq x-2$ and respects the $\sim_{x-2}$-classes. Thus:

$$(q,m) \xrightarrow[\mathsf{r}]{w_1:x-1} (q_1,m_1), \text{ with } q_1 \sim_{x-2} p' \text{, and } \quad [p']_x \xrightarrow{w_1:x} [p']_x.$$

If $\mathsf{Safe}_{<x}(q_1) \subseteq \mathsf{Safe}_{<x}(p')$, we can conclude by using the first case. While we do not have this inclusion, we build, using the argument above, a sequence of words $w_1, w_2, \ldots$ such that:

$$(q_i,m_i) \xrightarrow[\mathsf{r}]{w_i:x-1} (q_{i+1},m_{i+1}) \text{, and } [p']_x \xrightarrow{w_i:x} [p']_x.$$

This sequence cannot be infinite. If it were the case, resolver r would induce a rejecting run over $w_1w_2\ldots$ from (q,m), and an accepting from p'. This is a contradiction, as the equivalence $q \sim_{x-2} p$ implies $q^{-1}W = p'^{-1}W$ (since $\sim_{x-2}$ refines $\sim_{\mathcal{A}}$). Therefore, for some k we must have $\mathsf{Safe}_{<x}(q_k) \subseteq \mathsf{Safe}_{<x}(p')$ and we can extend the path as wanted using the first case. ◁

We may finally deduce the result of Lemma 5 from the two previous claims. First, we read the word u from Claim 5.1, which forces to produce priority $x-1$ from any state in $[q]_x$. We now show how to use Claim 5.2 to redirect each state, one by one, to the class $[p]_x$.

We let $(q_1,m_1),\ldots,(q_k,m_k)$ be an enumeration of all states such that there exists $s \in [q]_x$ with $s \xrightarrow[\exists,\mathsf{r}]{u:x-1} (q_i,m_i)$. We note that, by $[0,x-2]$-faithfulness, $q_i \sim_{x-2} q \sim_{x-2} p$. We recursively build a sequence of k words, $w_1,\ldots,w_k \in \Sigma^*$ by setting:

$$(q_i,m_i) \xrightarrow[\mathsf{r}]{w_1\ldots w_{i-1}:\geq x-1} (q'_i,m'_i) \xrightarrow[\mathsf{r}]{w_i:\geq x-1} [p]_x \text{ and } \quad [p]_x \xrightarrow[\forall,\mathsf{r}]{w_i:x} [p]_x.$$

We can indeed do this by letting $w_i = w_{q'_i,m'_i}$ as given by Claim 5.2, as by $[0,x-2]$-faithfulness $q'_i \sim_{x-2} p$.

The word $w_1\ldots w_i$ satisfies that, for $j \leq i$, $(q_j,m_j) \xrightarrow[\mathsf{r}]{w_1\ldots w_i:\geq x-1} [p]_x$. We conclude the proof of the lemma by putting $w' = uw_1\ldots w_k$. ◀

C.2.4 Re-determinisation

We give the proof of Lemma V.76, that is, we show that we can obtain an equivalent deterministic automaton from $\mathcal{A}$ while preserving all the obtained structure of total preorders satisfying the conditions of a structured signature automaton.

Hypothesis. We assume that $\mathcal{A}$ is a parity automaton recognising W with nested total preorders defined up to $\leq_x$ such that:

- it is a $(x-2)$-structured signature automaton,
- preorder $\leq_{x-1}$ satisfies properties from Items 2 and 6 from the definition of a structured signature automaton,
- preorder $\leq_x$ satisfies the property from Item 3 from the definition of a structured signature automaton,
- it is deterministic over transitions with priorities different from $x-1$,
- it is homogeneous, and
- it is history-deterministic.

Lemma V.76 (Re-determinisation)**.**

There is a deterministic parity automaton $\mathcal{A}'$ equivalent to $\mathcal{A}$ with nested total preorders defined up to $\leq_x$ satisfying that:

- it is a $(x-2)$-structured signature automaton,
- preorder $\leq_{x-1}$ satisfies properties from Items 2 and 6 from the definition of a structured signature automaton, and
- preorder $\leq_x$ is a congruence and satisfies the property from Item 3 and, for priorities $y < x$, also that from Item 4 .

Moreover, automaton $\mathcal{A}'$ can be computed in polynomial time from $\mathcal{A}$ and $|\mathcal{A}'| \leq |\mathcal{A}|$.

Obtaining a deterministic automaton. An intuitive idea for the construction of $\mathcal{A}'$ was given in Section V.4.2. We formalise it and prove its correctness now.

Automaton $\mathcal{A}'$ is obtained by keeping all the structure of $\mathcal{A}$, except for $(x-1)$-transitions; for each state q and letter a such that some transition $q \xrightarrow{a:x-1} p$ appears in $\mathcal{A}$, we will redefine this transition as $q \xrightarrow{a:x-1} p'$ for some $p' \sim_{x-2} p$ as determined next.

For each $\sim_{x-1}$-class $[q]_{x-1}$ of $\mathcal{A}$, we pick a state in the class that is maximal for $\leq_x$. We let $f(q)$ be that state. That is, for two states $q_1 \sim_{x-1} q_2$:

- $f(q_1) = f(q_2)$,
- $f(q_1) \in [q_1]_{x-1}$, and
- $q_1 \leq_x f(q_1)$.

We recall that we have a total order over the $<x$-safe components of $\mathcal{A}$ given by $S_1^{<x}, S_2^{<x}, \ldots, S_{k_x}^{<x}$. Let $q \in S_i^{<x}$ and $a \in \Sigma$ such that $q \xrightarrow{a:x-1} p$ appears in $\mathcal{A}$. If it exists, we let i_{next} be the maximal $0 \leq i_{\mathsf{next}} < i$ such that there is some $p' \in [p]_{x-2}$, $p' \in S_{i_{\mathsf{next}}}^{<x}$. If index i_{next} is not defined, we let it be the maximal index $i \leq i_{\mathsf{next}} \leq k_x$ with the previous property. We fix a state $p_{q,a} \in S_{i_{\mathsf{next}}}^{<x}$ with $p_{q,a} \in [p]_{x-2}$. We let the a-transition from q in $\mathcal{A}'$ be $q \xrightarrow{a:x-1} f(p_{q,a})$. This completes the description of $\mathcal{A}'$. It is indeed deterministic, as $\mathcal{A}$ is homogeneous and deterministic over transitions with priorities different from $x-1$.

We can find a maximal state $f(q)$ for $\leq_x$ in the class $[q]_{x-1}$ in polynomial time, as the comparison of $<x$-safe languages can be done in polynomial time (Proposition I.15).

Therefore, automaton $\mathcal{A}'$ can be built in polynomial time.

✦ **Lemma C.21.** *Automaton $\mathcal{A}'$ is a $\sim_{x-2}$-nice transformation of $\mathcal{A}$ at level $x-1$.*

Proof. This is clear, as the restriction of $\mathcal{A}'$ to transitions using a priority different from $(x-1)$ coincides with that of $\mathcal{A}$, and every transition $q \xrightarrow{a:x-1} p'$ in $\mathcal{A}'$ comes from a transition $q \xrightarrow{a:x-1} p$ in $\mathcal{A}$ with $p \sim_{x-2} p'$. ◀

✦ **Lemma C.22.** *For every state $q \in Q$, we have $\mathcal{L}(\mathcal{A}'_q) = \mathcal{L}(\mathcal{A}_q)$. In particular, automaton $\mathcal{A}'$ recognises the language $\mathcal{L}(\mathcal{A})$.*

Proof. For simplicity, we give the proof just for the initial state; the proof being identical for any other state.

Let $w \in \Sigma^\omega$. If the minimal priority produced infinitely often by the run over w in $\mathcal{A}'$ is $y < x-1$ or $y > x-1$, then w is accepted by $\mathcal{A}'$ if and only if w is accepted by $\mathcal{A}$, by Lemma C.6 and the fact that $\mathcal{A}'$ is a $\sim_{x-2}$-nice transformation of $\mathcal{A}$ at level $x-1$.

Assume that the minimal priority produced infinitely often by the run over w in $\mathcal{A}'$ is $x-1$ (so it is rejecting), and suppose by contradiction that w is accepted by $\mathcal{A}$. By Lemma C.6, an accepting run over w in $\mathcal{A}$ cannot produce a priority $y < x$ infinitely often. Therefore, it eventually remains in a $<x$-safe component $S^{<x}_{i_\mathcal{A}}$. Let ρ be such an accepting run, and let ρ' be the run over w in $\mathcal{A}'$. We represent them as:

$$\rho = q_0 \overset{u}{\rightsquigarrow} q_N \xrightarrow{w_N:\geq x} q_{N+1} \xrightarrow{w_{N+1}:\geq x} \dots\ , \quad \rho' = q'_0 \overset{u}{\rightsquigarrow} q'_N \overset{w_N:x'_N}{\rightsquigarrow} q'_{N+1} \xrightarrow{w_{N+1}:x'_{N+1}} \dots,$$

where u is the prefix of size N of w, $q_0 = q'_0 = q_{\mathsf{init}}$, and we suppose that $q_k \in S^{<x}_{i_\mathcal{A}}$ for all $k \geq N$. As $\mathcal{A}'$ is a $\sim_{x-2}$-nice transformation at level $x-1$, we have that $q_k \sim_{x-2} q'_k$ for all k. Let $k_1 < k_2 < k_3 \dots$ be the positions greater than N where $x'_{k_j} = x-1$, and let $i_1, i_2, \dots$ be the indices of the $<x$-safe components such that $q_{k_j+1} \in S^{<x}_{i_j}$, that is, when taking the transition $q_{k_j} \xrightarrow{w_{k_j}:x-1}$ we land in $S^{<x}_{i_j}$.

✧ Claim 5.1. *Eventually, $i_j = i_\mathcal{A}$.*

Proof. Consider a transition $q'_{k_j} \xrightarrow{w_{k_j}:x-1} q'_{k_j+1}$, and suppose first that $i_\mathcal{A} < i_{j-1}$. We claim that $i_\mathcal{A} \leq i_j < i_{j-1}$. This would end the proof, as we obtain a strictly decreasing sequence of indices bounded by $i_\mathcal{A}$. In order to determine i_j, we need to look at the definition of i_{next}. As $q_k \sim_{x-2} q'_k$ for all k, there are always states in $[q'_{k_j+1}]_{x-2}$ in some $<x$-safe components with an index $i_\mathcal{A} \leq i < i_{j-1}$. Thus, we obtain the desired result by definition of i_{next}.

If $i_{j-1} \leq i_\mathcal{A}$, by definition of i_{next}, there is be a sequence of decreasing indices $i_{j-1} > i_j > i_{j+1} > \dots$ until no $\sim_{x-2}$-equivalent state appears in a strictly smaller safe component. By the same argument as before, there is always a $\sim_{x-2}$-equivalent state in $S^{<x}_{i_\mathcal{A}}$, so eventually $i_\mathcal{A} \leq i_{j'}$, and either this is an equality, or we reduce to the previous case. ◁

Let j be the first position such that $i_j = i_\mathcal{A}$, and consider transitions $q_{k_j} \xrightarrow{w_{k_j}:\geq x} q_{k_j+1}$ and $q'_{k_j} \xrightarrow{w_{k_j}:x-1} q'_{k_j+1}$ in ρ and ρ', respectively. By definition of $x-1$-transitions of $\mathcal{A}'$, the state we go to in ρ' is $q'_{k_j+1} = f(q_{k_j+1})$. As we have chosen $f(q_{k_j+1})$ maximal in its $\sim_{x-1}$-class, we have $q_{k_j+1} \leq_x q'_{k_j+1}$, so we have the inclusion of $<x$-safe languages between these states. Therefore, if there is a $<x$-safe run over w' from q_{k_j+1} in $\mathcal{A}$, there

is also such a safe run over w' from q'_{k_j+1} in $\mathcal{A}'$. This contradicts the fact that the run ρ' produces priority $x-1$ infinitely often, while the run ρ is $<x$-safe from q_{k_j+1}, concluding the proof. ◀

To finish the proof of Lemma 5, we just need to show that $\mathcal{A}'$ preserves all the properties of the preorders induced by $\mathcal{A}$. As in the previous section, to obtain normality of the automaton and Item 5, we rely on a technical lemma that tells us that we can connect states in the same $\sim_{x-2}$-component as desired.

✦ **Lemma C.23.** *Let $q \sim_{x-2} p$ be two different states in Q. There is a word $w \in \Sigma^*$ and a path $q \overset{w:\geq x-2}{\rightsquigarrow} p$ in $\mathcal{A}'$*

Proof. Let i_p be the index such that $p \in S_{i_p}^{<x}$. If q belongs to this same safe component, we can connect both states by a path producing priorities $\geq x$. If not, by $<x$-safe centrality of $\mathcal{A}$, there is a word $w_1 \in \mathsf{Safe}_{<x}(p) \setminus \mathsf{Safe}_{<x}(q)$. We let $q \overset{w_1:x-1}{\rightsquigarrow} q_1$ and $p \overset{w_1:\geq x}{\rightsquigarrow} p_1$. We have that $q_1 \sim_{x-2} p_1$ and $p_1 \in S_{i_p}^{<x}$. While $q_j \notin S_{i_p}^{<x}$, we extend this run in a similar way. Using the same argument as in the proof of the previous lemma, by definition of i_{next} each time that the run from q sees a priority $x-1$ it decreases the index of its safe component, so eventually it must land in $S_{i_p}^{<x}$. ◀

✦ **Lemma C.24.** *Let $q \overset{w:y}{\rightsquigarrow} p$ be a path in $\mathcal{A}$. There is a path $q \overset{w':y}{\rightsquigarrow} p$ in $\mathcal{A}'$ connecting the same states and producing the same minimal priority.*

Proof. If $y \geq x$ we can just take $w = w'$. Suppose $y < x$ and consider the run $q \overset{w:y'}{\rightsquigarrow} p'$ in $\mathcal{A}'$. By Lemma C.4, $p \sim_{x-2} p'$. Also, $y = y'$ (if $y < x-1$, this is given by Lemma C.4, if $y = x-1$, by the definition of $\mathcal{A}'$). By the previous lemma, we can extend this run to $p' \overset{w_2:\geq x-2}{\rightsquigarrow} p$, and take $w' = ww_2$. ◀

✦ **Lemma C.25.** *Automaton $\mathcal{A}'$, with the preorders $\leq_0, \ldots, \leq_x$ inherited from $\mathcal{A}$ satisfies:*

- *it is a $(x-2)$-structured signature automaton,*
- *preorder $\leq_{x-1}$ satisfies properties from Items 2 and 6 from the definition of a structured signature automaton, and*
- *preorder $\leq_x$ satisfies the property from Item 3 and, for priorities $y < x$, also that from Item 4.*

Proof. We start verifying the properties for the preorders $\leq_{x-1}$ and $\leq_x$. We note that the $<x$-safe components of $\mathcal{A}'$ exactly correspond to those in $\mathcal{A}$, and that for every $q \in Q$, $\mathsf{Safe}^{\mathcal{A}}_{<x}(q) = \mathsf{Safe}^{\mathcal{A}'}_{<x}(q)$. The fact that $\leq_{x-1}$ satisfies Items 2 and 6, and that $\leq_x$ satisfies Item 3 follows immediately. We check that relation $\sim_x$ satisfies Item 4 for priorities $y < x$. For $y \leq x-2$, this follows from the fact that $\sim_x$ refines $\sim_{x-2}$, and the latter relation satisfies Item 4. For $y = x-1$, if $q \xrightarrow{a:x-1} p$ and $q' \xrightarrow{a:x-1} p'$, with $q \sim_x q'$, by definition of the $x-1$-transitions in $\mathcal{A}'$, $p = f(p) = f(p') = p'$.

Checking that $\mathcal{A}'$ is a $(x-2)$-structured signature automaton poses no difficulty. It can be done in an analogous way as it was done in the proof of Lemma C.20; by applying Lemma C.24 to obtain normality of $\mathcal{A}'$ and property from Item 5. ◀

C.2.5 Uniformity of x-transitions over $\sim_x$-classes

We finally show how to transform $\mathcal{A}$ into an equivalent automaton that is either x-structured signature, or strictly smaller. The techniques presented here generalise those applying to Büchi automata appearing in Section V.3.3 of the warm-up.

Hypothesis. In all this subsection we suppose that $\mathcal{A}$ is deterministic parity automaton recognising W with nested total preorders defined up to $\leq_x$ such that:

- it is a $(x-2)$-structured signature automaton,
- preorder $\leq_{x-1}$ satisfies properties from Items 2 and 6 from the definition of a structured signature automaton, and
- preorder $\leq_x$ is a congruence and satisfies the property from Item 3 and, for priorities $y < x$, also that from Item 4 .

Our objective is to prove, under this list of hypothesis, that we can either obtain an equivalent deterministic x-structured signature automaton, or reduce the number of states of $\mathcal{A}$.

Lemma V.77 (Uniformity of x-transitions over $\sim_x$-classes)**.**

There is a deterministic parity automaton $\mathcal{A}'$ equivalent to $\mathcal{A}$ such that either:

- $\mathcal{A}'$ is an x-structured signature automaton with $|\mathcal{A}'| \leq |\mathcal{A}|$, or
- $|\mathcal{A}'| < |\mathcal{A}|$.

In both cases, such an automaton can be computed in polynomial time from $\mathcal{A}$.

We remark that $\sim_x$ already satisfies most desired properties of monotonicity; only the uniformity for x-transitions is missing.

✦ **Lemma C.26.** *The relation $\sim_x$ is a $[0, x-1]$-faithful congruence. Moreover, over each $\sim_{x-1}$-class, transitions using priorities $\geq x$ are monotone for $\leq_x$.*

Proof. The $[0, x-2]$-faithfulness follows from the fact that $\sim_x$ refines $\sim_{x-2}$ and $\mathcal{A}$ is $(x-2)$-structured signature. The uniformity of $(x-1)$-transitions over $\sim_x$-classes is given by the fact that $\sim_x$-equivalent states have the same $<x$-safe language, combined with the uniformity of $<(x-1)$-transitions.

The fact that $\sim_x$ is a congruence for $(x-1)$-transitions follows from Item 4 of the definition of a structured signature automaton (we recall that $\sim_x$ satisfies this property by hypothesis). The congruence for $\geq x$-transitions and the monotonicity of $\geq x$-transitions for $\leq_x$ at each $\sim_{x-1}$-class follow from the monotonicity of $<x$-safe languages (Lemma V.66). ◀

Polished automata. We generalise the notion of polished automata from Section V.3.3 to our current setting.

Definition C.27 (Polished classes and automata)**.**

We say that the class $[q]_x$ is *x-polished* if:

- Words producing priority x act uniformly in $[q]_x$. That is, if $q_1, q_2 \in [q]_x$ and $q_1 \overset{w:x}{\rightsquigarrow}$, then $q_2 \overset{w:x}{\rightsquigarrow}$.
- For every $q_1, q_2 \in [q]_x$, $q_1 \neq q_2$, there is a path $q_1 \overset{w:>x}{\rightsquigarrow} q_2$ producing exclusively priorities $> x$ joining q_1 and q_2.

We say that the automaton $\mathcal{A}$ is *x-polished* if all its $\sim_x$-classes are x-polished.

✦ Remark C.28. *We remark that, as $x \geq 2$ and we assume that the automaton $\mathcal{A}$ is in normal form, all non-trivial $\sim_x$-classes are recurrent: if a $\sim_x$-class is not trivial, there is a cycle visiting all the states of the class. Therefore, we do not need to take care of transient classes (as it was the case in Lemma V.38 from the warm-up).*

✦ **Lemma C.29.** *We can decide whether $\mathcal{A}$ is x-polished in polynomial time.*

Proof. As $\sim_x$ is a $[0, x-1]$-faithful congruence (Lemma C.26), we just need to check the first property for letters, which can be done in linear time in $|\Sigma||\mathcal{A}|$.

For the second property, we just need to check whether, for each $q \in Q$, the subautomaton induced by $[q]_x$ and transitions with priority $>x$ is strongly connected. ◀

Case 1: $\mathcal{A}$ is already x-polished

Assume that $\mathcal{A}$ is x-polished. In this case, it is almost an x-structured signature automaton. We just need to ensure that if $q \sim_x q'$, two transitions $q \xrightarrow{a:x} p$ and $q' \xrightarrow{a:x} p'$ go to a same state $p = p'$.

We remark that $\sim_x$ already satisfies most desired properties of monotonicity; only the uniformity for x-transitions is missing.

In order to obtain the strong congruence of x-transitions (Item 4), we redirect some x-transitions of $\mathcal{A}$. For each $\sim_x$-class $[q]_x$, pick an arbitrary state $f(q) \in [q]_x$. (Formally, $f\colon Q \to Q$ such that $f(q) = f(q')$ if $q \sim_x q'$). We let $\mathcal{A}'$ be the automaton obtained as follows:

- The states of $\mathcal{A}'$ are the same than those in $\mathcal{A}$.
- Transitions using priorities different from x are those in $\mathcal{A}$.
- If $q \xrightarrow{a:x} p$, we let $q \xrightarrow{a:x} f(p)$ in $\mathcal{A}'$.

It is immediate to check that $\mathcal{A}'$ is a $\sim_x$-nice transformation of $\mathcal{A}$ at level x. Moreover, $\mathcal{A}|_{\geq x+1} = \mathcal{A}'|_{\geq x+1}$. These remarks directly give:

✦ **Lemma C.30.** *$\mathcal{A}'$ is x-polished.*

✦ **Lemma C.31.** *There is a path $q \overset{w:y}{\rightsquigarrow} p$ in $\mathcal{A}$ if and only if there is a path $q \overset{w':y}{\rightsquigarrow} p$ in $\mathcal{A}'$.*

Proof. We suppose that there is $q \overset{w:y}{\rightsquigarrow} p$ in $\mathcal{A}$ (the converse proof is symmetric). If $y > x$, we have that $q \overset{w:y}{\rightsquigarrow} p$, as $\mathcal{A}|_{\geq x+1} = \mathcal{A}'|_{\geq x+1}$. If $y \leq x$, as $\mathcal{A}'$ is a nice transformation at level x, we have that $q \overset{w:y}{\rightsquigarrow} p'$ in $\mathcal{A}'$, with $p \sim_x p'$. As $\mathcal{A}'$ is x-polished, there is a path $p' \overset{w_2:>x}{\rightsquigarrow} p$. We conclude by taking $w' = ww_2$. ◀

✦ **Lemma C.32.** *Automaton $\mathcal{A}'$ is equivalent to $\mathcal{A}$, and it is an x-structured signature automaton.*

Proof. The fact that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$ follows easily using that $\mathcal{A}'$ is a $\sim_x$-nice transformation of $\mathcal{A}$ at level x and applying Lemma C.6.

Lemma C.31, combined with Theorem II.14, implies that $\mathcal{A}'$ is in normal form.

Verifying that $\mathcal{A}'$ is an x-structured signature automaton is just a routine check, using that $\mathcal{A}'$ is x-polished and Lemma C.31. ◀

Case 2: Polishing a class

Assume now that there is a class $[q]_x$ that is not x-polished in $\mathcal{A}$. We show that we can remove some states from this class, obtaining an strictly smaller equivalent automaton.

Local languages and local automata. We define the *x-local alphabet at $[q]_x$* by

$$\Sigma_{[q]_x} = \{w \in \Sigma^+ \mid [q]_x \overset{w:\geq x}{\rightsquigarrow} [q]_x \text{ and for any proper prefix } w' \text{ of } w, [q]_x \overset{w:\geq x}{\rightsquigarrow} [p]_x \neq [q]_x\}.$$

We remark that, as $\sim_x$ is a congruence for $\geq x$-transitions (Lemma C.26), $\Sigma_{[q]_x}$ is well-defined and the notation $[q]_x \overset{w:\geq x}{\rightsquigarrow}$ can be used. A word $w \in \Sigma^*$ belongs to $\Sigma^*_{[q]_x}$ if and only if it connects states in the class $[q]_x$. Elements in $\Sigma_{[q]_x}$ are those that do not pass twice through this class. Note that $\Sigma_{[q]_x}$ is a prefix code, and therefore it is a proper alphabet (even if, in general, it is infinite).

Seeing words in $\Sigma^\omega_{[q]_x}$ as words in Σ^ω, define the *localisation of W to $[q]_x$* to be the objective

$$W_{[q]_x} = \{w \in \Sigma^\omega_{[q]_x} \mid w \in q^{-1}W\}.$$

Observe that, as $\sim_x$ refines $\sim_{\mathcal{A}}$, this last definition does not depend on the choice of q and $W_{[q]_x}$ is prefix-independent. Moreover, $W_{[q]_x}$ is half-positional over finite, ε-free Eve-games: any $W_{[q]_x}$-game in which Eve could not play optimally using positional strategies would provide a counterexample for the half-positionality of $q^{-1}W$, which is half-positional if W is (Lemma C.7).

The *local automaton of the class* $[q]_x$ is the automaton $\mathcal{A}_{[q]_x}$ defined as:

- The set of states is $[q]_x$.
- The initial state is arbitrary.
- For $w \in \Sigma_{[q]_x}$, $q_1 \xrightarrow{w:y} q_2$ if $q_1 \overset{w:y}{\rightsquigarrow} q_2$ in $\mathcal{A}$ (we must have $y \geq x$).

Super words and super letters for local languages. We recall some terminology introduced in the warm-up. Assume that L is a prefix-independent language. We say that $u \in \Sigma^+$ is a *super word* for L if, for every $w \in \Sigma^\omega$, if w contains u infinitely often as a factor, then $w \in L$. If s is a letter, we say that it is a *super letter*.

For q a state and x an even priority, we let $B_{[q]_x} \subseteq \Sigma_{[q]_x}$ be the set of super letters for $W_{[q]_x}$, and we write $N_{[q]_x} = \Sigma_{[q]_x} \setminus B_{[q]_x}$. We refer to $N_{[q]_x}$ as the set of *neutral letters* of $\Sigma_{[q]_x}$ (for $W_{[q]_x}$).

✦ **Lemma C.33** (Super words and uniformity)**.** *A word $w \in \Sigma^\omega_{[q]_x}$ is a super word for $W_{[q]_x}$ if and only if w produces priority x uniformly in $[q]_x$, that is, for all $q' \in [q]_x$, $q' \overset{w:x}{\rightsquigarrow} [q]_x$.*

Proof. By normality of $\mathcal{A}$ (Lemma V.13), if there is $q_1 \in [q]$ such that $q_1 \overset{w:>x}{\rightsquigarrow} q_2$, there is a word $w' \in \Sigma^*$ labelling a returning path $q_2 \overset{w':x+1}{\rightsquigarrow} q_1$. Therefore, $(ww')^\omega \notin q^{-1}W$, so

w is not a super word for $W_{[q]_x}$. The converse implication is clear, since each time word w is read, the minimal priority produced by the automaton is x. ◀

In particular, using previous lemma, we can detect the set of super letters $B_{[q]_x}$ in polynomial time.

Super words of positional languages. The use of the hypothesis of half-positionality of W for proving Lemma 5 resides in the next fundamental result.

Lemma C.34 (Neutral letters do not form super words)**.**

Let $w \in \Sigma^+_{[q]_x}$ be a super word for $W_{[q]_x}$. Then, w contains some super letter.

Proof. If w is already a letter in $\Sigma_{[q]_x}$, we are done. If not, let $w = w_1w_2$ be any non-trivial decomposition into smaller words $w_1, w_2 \in \Sigma^+_{[q]_x}$. We show that either w_1 or w_2 are super words for $W_{[q]_x}$. This allows us to finish the proof, as we can recursively chop w into strictly smaller super words until obtaining a super letter.

Suppose by contradiction that neither w_1 or w_2 are super words. Then, by Lemma C.33, there are states q_1 and q_2 such that $q_1 \overset{w_1:>x}{\rightsquigarrow} q_1'$ and $q_2 \overset{w_2:>x}{\rightsquigarrow} q_2'$. By normality (Lemma V.13), we obtain returning paths $q_1' \overset{u_1:x+1}{\rightsquigarrow} q_1$ and $q_2' \overset{u_2:x+1}{\rightsquigarrow} q_2$. Therefore, $(w_1u_1)^\omega \notin W$ and $(w_2u_2)^\omega \notin W$. We consider the game $\mathcal{G}$ with winning condition $q^{-1}W$ consisting in a vertex v with self loops u_1w_1 and u_2w_2 (see Figure 32 from the warm-up). Eve can win game $\mathcal{G}$, as alternating the two self loops she produces the word $(u_1w_1w_2u_2)^\omega$, which belongs to $q^{-1}W$ since w_1w_2 is a super word. However, positional strategies in this game produce either $(w_1u_1)^\omega$ or $(w_2u_2)^\omega$, both losing. This contradicts the half-positionality of $q^{-1}W$, and therefore, that of W (Lemma C.7). ◀

Polishing a $\sim_x$-class of $\mathcal{A}$. We show how to polish the class $[q]_x$ of $\mathcal{A}$. This process has the property that, either $[q]_x$ is already polished, or the obtained automaton $\mathcal{A}'$ has strictly less states than $\mathcal{A}$, as desired.

Assume that the class $[q]_x$ is not x-polished. Consider the restriction of $\mathcal{A}_{[q]_x}$ to transitions labelled with $N_{[q]_x}$, which we denote $\mathcal{A}'_{[q]_x}$. Take $S_{[q]_x}$ to be a final SCC of $\mathcal{A}'_{[q]_x}$ (by a small abuse of notation, we also denote $S_{[q]_x}$ the set of states of this SCC).

Fix a state $q_0 \in [q]_x$. Consider the automaton $\mathcal{A}'$ obtained from $\mathcal{A}$ by removing states in $[q]_x \setminus S_{[q]_x}$, and redirecting transition that go to $[q]_x \setminus S_{[q]_x}$ in $\mathcal{A}$ to transitions towards q_0. For these redirected transitions, we keep the same priority if it is $\leq x$, and set it to x otherwise. Formally:

- The set of states of $\mathcal{A}'$ is $Q' = Q \setminus \left([q]_x \setminus S_{[q]_x}\right)$.
- The initial state is q_{init}, or q_0 if $q_{\mathsf{init}} \in [q]_x \setminus S_{[q]_x}$.

For $q' \in Q'$:

- If $q' \xrightarrow{a:y} p$ in $\mathcal{A}$ and $p \notin [q]_x$, then $q \xrightarrow{a:y} p$ in $\mathcal{A}'$.
- If $q' \xrightarrow{a:y} p$ in $\mathcal{A}$, $p \in [q]_x \setminus S_{[q]_x}$, and $y \leq x$, then $q' \xrightarrow{a:y} q_0$ in $\mathcal{A}'$.
- If $q' \xrightarrow{a:y} p$ in $\mathcal{A}$, $p \in [q]_x \setminus S_{[q]_x}$, and $y > x$, then $q' \xrightarrow{a:x} q_0$ in $\mathcal{A}'$.

For transitions in the two latter cases, we say that $q' \xrightarrow{a:y} q_0$ has been *redirected* from $q' \xrightarrow{a:y} p$.

✦ Remark C.35. *If* $[q]_x = S_{[q]_x}$, *then* $\mathcal{A}' = \mathcal{A}$.

The following lemma will be use to show that we can compute $\mathcal{A}'$ in polynomial time, to prove the correctness of $\mathcal{A}'$, and to obtain that $[q]_x$ is x-polished in $\mathcal{A}'$.

✦ **Lemma C.36.** *Let* $q_1, q_2 \in S_{[q]_x}$, *and let* $w \in N^*_{[q]_x}$ *labelling a path* $q_1 \overset{w:y}{\rightsquigarrow} q_2$ *in* $\mathcal{A}$. *Then,* $y > x$.

Proof. The fact that $y \geq x$ simply follows from the fact that $N_{[q]_x} \subseteq \Sigma_{[q]_x}$, which, by definition, contains words connecting the states in $[q]_x$ producing no priority $<x$.

Suppose by contradiction that $y = x$. Then, by the same argument as in the proof of Lemma 5 (see also Claim V.33.2), there is $w' \in N^*_{[q]_x}$ producing priority x uniformly in $[q]_x$ and coming back to this class; that is, for every $q' \in [q]_x$, $q' \overset{w':x}{\rightsquigarrow} [q]_x$. Therefore, by Lemma C.33, w' is a super word. By Lemma C.34, w' must contain a super letter, a contradiction, as $w' \in N^*_{[q]_x}$ and $\Sigma_{[q]_x}$ is a proper alphabet. ◀

✦ **Lemma C.37.** *Automaton* $\mathcal{A}'$ *can be computed in polynomial time from* $\mathcal{A}$.

Proof. To obtain $S_{[q]_x}$, we first build a finite representation of the underlying graph of the restriction of $\mathcal{A}_{[q]_x}$ to neutral letters (we recall that, in general, $\mathcal{A}_{[q]_x}$ might have an infinite number of transitions). One way of doing that is to build the following graph G: for each pair of states $q_1, q_2 \in [q]_x$ and each $y > x$, we put an edge $q_1 \xrightarrow{y} q_2$ if there is a path from q_1 to q_2 producing y as minimal priority and not passing trough another state in $[q]_x$. By Lemma C.36, $S_{[q]_x}$ is a subgraph of G. To obtain the states in $S_{[q]_x}$ we just need to perform a decomposition in SCCs of G and take a final SCC of it.[3] ◀

We consider $\mathcal{A}'$ equipped with the preorders $\leq_0, \ldots, \leq_x$ inherited from $\mathcal{A}$.

✦ **Lemma C.38.** *Automaton* $\mathcal{A}'$ *is a* $\sim_x$*-nice transformation of* $\mathcal{A}$ *at level* x.

Proof. We first note that, by Lemma C.26, $\sim_x$ is $[0, x-1]$-faithful in $\mathcal{A}$, so it makes sense to speak of a $\sim_x$-nice transformation at level x.

Automaton $\mathcal{A}'|_{\geq x+1}$ coincides with the subautomaton of $\mathcal{A}|_{\geq x+1}$ induced by states in Q'. Indeed, let $q'_1, q'_2 \in Q'$ and $q'_1 \xrightarrow{a:>x} q'_2$ in $\mathcal{A}$. As these states are in Q', $q'_2 \notin [q]_x \setminus S_{[q]_x}$, so the transition has not been redirected, and it appears in $\mathcal{A}'$. Conversely, all transitions producing a priority $> x$ in $\mathcal{A}'$ appear in $\mathcal{A}$.

We show that $\sim_x$ is $[0, x-1]$-faithful in $\mathcal{A}'$ and $\mathcal{A}\big/_{\leq x-1}^{\sim_{x-1}} = \mathcal{A}\big/_{\leq x-1}^{\sim_{x-1}}$. Let $p_1, p_2 \in Q'$ such that $p_1 \sim_x p_2$, and let $p_1 \xrightarrow{a:y'_1} q'_1$ and $p_2 \xrightarrow{a:y'_2} q'_2$ be two transitions in $\mathcal{A}'$. Transitions that have not been redirected satisfy the congruence requirements, as they satisfy them in $\mathcal{A}$. Assume that the first of these transitions have been redirected from $p_1 \xrightarrow{a:y_1} q_1$ in $\mathcal{A}$. We have that $q'_1 = q_0 \sim_x q_1$, so, $q'_1 \sim_x q'_2$ by the congruence property in $\mathcal{A}$. If $y'_1 < x$, then $y_1 = y'_1$ and the y_1-uniformity of transitions in $\mathcal{A}$ yields $y'_1 = y'_2$. Therefore, we also

[3] If W is not half-positional, the procedure described here does provide a set of states $S_{[q]_x}$, but it might lead to an incorrect automaton $\mathcal{A}'$. If our objective is to decide the half-positionality of W, at the end of the procedure we need to check the equality $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$; if it does not hold, we can conclude that W is not half-positional.

have that $y_1' \geq x$ if and only if $y_2' \geq x$ This gives both the $[0, x-1]$-faithfulness in $\mathcal{A}'$ and the equality of the quotient automata.

As $\sim_x$ refines $\sim_{\mathcal{A}}$, the latter relation is also a congruence in $\mathcal{A}'$ and $\mathcal{A}/_{\sim_{\mathcal{A}}} = \mathcal{A}'/_{\sim_{\mathcal{A}}}$. ◀

✦ **Lemma C.39** (Correctness of the polishing operation)**.** *Automaton $\mathcal{A}'$ recognises $\mathcal{L}(\mathcal{A})$*

Proof. Let $w \in \Sigma^\omega$. If w is accepted or rejected with a priority $y < x$ or $y > x$ in $\mathcal{A}'$, by Lemma C.6, $w \in \mathcal{L}(\mathcal{A})$ if and only if $w \in \mathcal{L}(\mathcal{A}')$. Suppose then that w is accepted with priority x in $\mathcal{A}'$. Let ρ' be the run over w in $\mathcal{A}'$. If ρ' eventually does not take any redirected transition, then it is eventually a run in $\mathcal{A}$, and we can conclude by Lemma C.1. Suppose that ρ' takes redirected transitions infinitely often; moreover, eventually all such transitions produce priority x. We decompose ρ' as follows:

$$\rho' = q_{\mathsf{init}} \overset{w_0}{\rightsquigarrow} p_0' \xrightarrow{a_0:x} q_0 \overset{w_1:\geq x}{\rightsquigarrow} p_1' \xrightarrow{a_1:x} q_0 \overset{w_1:\geq x}{\rightsquigarrow} p_2' \xrightarrow{a_2:x} q_0 \rightsquigarrow \dots,$$

where no priority $< x$ appears after p_0', each transition $p_i' \xrightarrow{a_1:x} q_0$ is a redirected one, and no redirected transition appears in paths $q_0 \overset{w_i:\geq x}{\rightsquigarrow}$, in particular, these paths appear in $\mathcal{A}$.

✧ Claim C.39.1. *For each $i \geq 1$, the word $w_i a_i$ belongs to $\Sigma^+_{[q]_x}$ and is a super word for $W_{[q]_x}$.*

Proof. The word $w_i a_i$ connects two states in $[q]_x$ in $\mathcal{A}'$ producing no priority $< x$. Since $\mathcal{A}'$ is a $\sim_x$-nice transformation at level x, word $w_i a_i$ also connects states in $[q]_x$ in $\mathcal{A}$, without producing priorities $< x$. Therefore, it belongs to $\Sigma^+_{[q]_x}$.

Consider the path $q_0 \overset{w_i}{\rightsquigarrow} p_i' \xrightarrow{a_i} q'$ in $\mathcal{A}$. As we suppose that transition $p_i' \xrightarrow{a_i} q_0$ has been redirected in $\mathcal{A}'$, $q' \notin S_{[q]_x}$. Then, since $S_{[q]_x}$ is a final SCC of the restriction of $\mathcal{A}_{[q]_x}$ to $N_{[q]_x}$-transitions, $w_i a_i$ contains some factor that is a letter in $\Sigma_{[q]_x} \setminus N_{[q]_x}$. Such a factor is a super letter, so $w_i a_i$ is a super word. ◁

Consider the run over w in $\mathcal{A}$, that we divide following the decomposition of ρ':

$$\rho = q_{\mathsf{init}} \overset{w_0}{\rightsquigarrow} p_0 \xrightarrow{a_0:\geq x} q_1 \overset{w_1:\geq x}{\rightsquigarrow} p_1 \xrightarrow{a_1:\geq x} q_2 \overset{w_2:\geq x}{\rightsquigarrow} p_2 \xrightarrow{a_2:\geq x} q_3 \rightsquigarrow \dots.$$

As $\mathcal{A}'$ is a $\sim_x$-nice transformation at level x, $q_i \sim_x q_0$ for all i, and ρ does not produce any priority $< x$ from p_0. By Lemma C.33, as $w_i a_i$ is a super word, the path $q_i \overset{w_i a_i}{\rightsquigarrow} q_{i+1}$ produces priority x. Therefore, w is accepted by $\mathcal{A}$. ◀

✦ **Lemma C.40** (Polishing polishes)**.** *The class $[q]_x$ is x-polished in $\mathcal{A}'$.*

Proof. Let $q_1, q_2 \in Q'$ be two states in the class $[q]_x$. Assume that, for a word $w \in \Sigma^*$, the path $q_1 \overset{w:x}{\rightsquigarrow} p_1$ produces priority x. As $\sim_x$ is $[0, x-1]$-faithful, $q_2 \overset{w:\geq x}{\rightsquigarrow} p_2$. Suppose by contradiction that this latter path produces exclusively priorities $> x$. Then, this path also exists in $\mathcal{A}$, and by normality (of $\mathcal{A}$), there is a returning path $p_2 \overset{w':x+1}{\rightsquigarrow} q_2$. We obtain therefore a path $q_1 \overset{ww':x}{\rightsquigarrow} [q]_x$. However, in $\mathcal{A}'$, $[q]_x = S_{[q]_x}$, so, by Lemma C.36, ww' contains a super letter, so $(ww')^\omega \in q^{-1}W$, contradicting the fact that there is a cycle $q_2 \overset{ww':x+1}{\rightsquigarrow} q_2$.

The second property of the definition of an x-polished class is satisfied in $\mathcal{A}'$, as we have redirected all x-transitions entering in $[q]_x$ to the state q_0.

We show the third item. Let $q_1, q_2 \in [q]_x$. Since $[q]_x = S_{[q]_x}$ in $\mathcal{A}'$, there is a path $q_1 \overset{w}{\rightsquigarrow} q_2$ for some $w \in N^*_{[q]_x}$. By Lemma C.36, this path produces exclusively priorities $>x$. ◀

This lemma allows us to conclude. We have obtained a deterministic automaton $\mathcal{A}'$ that is equivalent to $\mathcal{A}$. We claim that $|\mathcal{A}'| < |\mathcal{A}|$. Indeed, if this was not the case, we would have that $[q]_x = S_{[q]_x}$, so, by Remark C.35, $\mathcal{A} = \mathcal{A}'$. By the previous Lemma C.40 this implies that $[q]_x$ was already x-polished in $\mathcal{A}$, a contradiction.

Discussion: Why not just continue polishing? We have just showed a method to x-polish a given class of $\mathcal{A}$. The natural continuation would be to polish the rest of classes, until obtaining an x-polished automaton, and then apply the first case. The main difficulty is that the polishing operation we have presented might break the normality of $\mathcal{A}$. Normality of automata is key in all the process (see for example Lemma C.33), so we cannot guarantee to be able to continue polishing the classes of $\mathcal{A}'$. We would need to be able to either show that $\mathcal{A}'$ is in normal form (for example, by having an analogous to Lemma C.31), or to show that we can normalise $\mathcal{A}'$ while maintaining the properties of being an $(x-2)$-structured signature automaton. We have not succeeded in ensuring these properties, although we believe that it should be possible to do so.

Bibliography

Personal references

[Bou+22] Patricia Bouyer, Antonio Casares, Mickael Randour, and Pierre Vandenhove. "Half-positional objectives recognized by deterministic Büchi automata". In: CONCUR, vol. 243. 2022, 20:1–20:18. DOI: 10.4230/LIPIcs.CONCUR.2022.20 (cited on pp. 53, 124, 182, 198, 199).

[Cas22] Antonio Casares. "On the minimisation of transition-based Rabin automata and the chromatic memory requirements of Muller conditions". In: CSL, vol. 216. 2022, 12:1–12:17. DOI: 10.4230/LIPIcs.CSL.2022.12 (cited on pp. 137, 162, 258).

[Cas+22] Antonio Casares, Alexandre Duret-Lutz, Klara J. Meyer, Florian Renkin, and Salomon Sickert. "Practical applications of the Alternating Cycle Decomposition". In: TACAS, vol. 13244. Lecture Notes in Computer Science. 2022, pp. 99–117. DOI: 10.1007/978-3-030-99527-0_6 (cited on pp. 20, 54, 103, 109, 133).

[Cas+23] Antonio Casares, Thomas Colcombet, Nathanaël Fijalkow, and Karoliina Lehtinen. "From Muller to parity and Rabin automata: Optimal transformations preserving (history-)determinism". In: *CoRR* abs/2305.04323 (2023). DOI: 10.48550/arXiv.2305.04323 (cited on p. 53).

[CCF21] Antonio Casares, Thomas Colcombet, and Nathanaël Fijalkow. "Optimal transformations of games and automata using Muller conditions". In: ICALP, vol. 198. 2021, 123:1–123:14. DOI: 10.4230/LIPIcs.ICALP.2021.123 (cited on pp. 53, 97).

[CCL22] Antonio Casares, Thomas Colcombet, and Karoliina Lehtinen. "On the size of good-for-games Rabin automata and its link with the memory in Muller games". In: ICALP, vol. 229. 2022, 117:1–117:20. DOI: 10.4230/LIPIcs.ICALP.2022.117 (cited on pp. 53, 96, 137, 162, 258).

[CO21] Antonio Casares and Pierre Ohlmann. "Fast value iteration for energy games". In: *CoRR* abs/2110.07346 (2021). DOI: 10.48550/arXiv.2110.07346 (cited on p. 263).

[CO22] Antonio Casares and Pierre Ohlmann. "Characterising memory in infinite games". In: *CoRR* abs/2209.12044 (2022). DOI: 10.48550/arXiv.2209.12044 (cited on pp. 165, 179, 263).

[CO23] Antonio Casares and Pierre Ohlmann. "Characterising memory in infinite games". In: ICALP, vol. 261. LIPIcs. 2023, 122:1–122:18. DOI: 10.4230/LIPIcs.ICALP.2023.122 (cited on pp. 162, 165, 188, 258, 259, 263).

General references

[ABF18] Dana Angluin, Udi Boker, and Dana Fisman. "Families of DFAs as acceptors of ω-regular languages". In: *Log. Methods Comput. Sci.* 14.1 (2018). DOI: 10.23638/LMCS-14(1:15)2018 (cited on p. 19).

[AK19] Bader Abu Radi and Orna Kupferman. "Minimizing GFG transition-based automata". In: ICALP, vol. 132. LIPIcs. 2019, 100:1–100:16. DOI: 10.4230/LIPIcs.ICALP.2019.100 (cited on pp. 10, 19, 136).

[AK22] Bader Abu Radi and Orna Kupferman. "Minimization and canonization of GFG transition-based automata". In: *Log. Methods Comput. Sci.* 18.3 (2022). DOI: 10.46298/lmcs-18(3:16)2022 (cited on pp. 12, 19, 27, 53, 96, 124, 150, 180, 204, 206, 207, 216, 217, 220, 246, 249, 260).

[AK23] Bader Abu Radi and Orna Kupferman. "On semantically-deterministic automata". In: *CoRR* abs/2305.15489 (2023). DOI: 10.48550/arXiv.2305.15489 (cited on pp. 185, 259).

[ALW89] Martín Abadi, Leslie Lamport, and Pierre Wolper. "Realizable and unrealizable specifications of reactive systems". In: ICALP, vol. 372. 1989, pp. 1–17. DOI: 10.1007/BFb0035748 (cited on p. 18).

[Arn+08] André Arnold, Jacques Duparc, Filip Murlak, and Damian Niwinski. "On the topological complexity of tree languages". In: Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas], vol. 2. Texts in Logic and Games. 2008, pp. 9–28 (cited on p. 20).

[Arn85] André Arnold. "A syntactic congruence for rational omega-language". In: *Theor. Comput. Sci.* 39 (1985), pp. 333–335. DOI: 10.1016/0304-3975(85)90148-3 (cited on pp. 18, 181).

[Bab+15] Tomáš Babiak, František Blahoudek, Alexandre Duret-Lutz, Joachim Klein, Jan Křetínský, David Müller, David Parker, and Jan Strejček. "The Hanoi omega-automata format". In: CAV, 2015, pp. 479–486 (cited on pp. 19, 43, 297).

[BCJ18] Roderick Bloem, Krishnendu Chatterjee, and Barbara Jobstmann. "Graph games and reactive synthesis". In: Handbook of Model Checking, ed. by Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. Springer International Publishing, 2018, pp. 921–962. DOI: 10.1007/978-3-319-10575-8_27 (cited on pp. 17, 246).

[BDL15] Souheib Baarir and Alexandre Duret-Lutz. "Sat-based minimization of deterministic ω-automata". In: LPAR, ed. by Martin Davis, Ansgar Fehnker, Annabelle McIver, and Andrei Voronkov. 2015, pp. 79–87 (cited on p. 20).

[Bia+11] Alessandro Bianco, Marco Faella, Fabio Mogavero, and Aniello Murano. "Exploring the boundary of half-positionality". In: *Ann. math. artif. intell.* 62.1-2 (2011), pp. 55–77. DOI: 10.1007/s10472-011-9250-1 (cited on pp. 24, 190, 198, 201, 211).

[BJW01] Bernard Boigelot, Sébastien Jodogne, and Pierre Wolper. "On the use of weak automata for deciding linear arithmetic with integer and real variables". In: Automated Reasoning, 2001, pp. 611–625 (cited on p. 53).

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008 (cited on p. 246).

[BK12] Udi Boker and Orna Kupferman. "Translating to co-büchi made tight, unified, and useful". In: *ACM Trans. Comput. Log.* 13.4 (2012), 29:1–29:26. DOI: 10.1145/2362355.2362357 (cited on p. 20).

[BKS10] Udi Boker, Orna Kupferman, and Avital Steinitz. "Parityizing Rabin and Streett". In: FSTTCS, vol. 8. LIPIcs. 2010, pp. 412–423. DOI: 10.4230/LIPIcs.FSTTCS.2010.412 (cited on pp. 53, 110, 118).

[BKS17] Udi Boker, Orna Kupferman, and Michal Skrzypczak. "How deterministic are good-for-games automata?" In: FSTTCS, vol. 93. 2017, 18:1–18:14. DOI: 10.4230/LIPIcs.FSTTCS.2017.18 (cited on pp. 110, 117).

[BL19] Udi Boker and Karoliina Lehtinen. "Good for Games Automata: From Nondeterminism to Alternation". In: CONCUR, vol. 140. 2019, 19:1–19:16. DOI: 10.4230/LIPIcs.CONCUR.2019.19 (cited on pp. 46, 79).

[BL21] Udi Boker and Karoliina Lehtinen. "History determinism vs. good for gameness in quantitative automata". In: FSTTCS, vol. 213. 2021, 38:1–38:20. DOI: 10.4230/LIPIcs.FSTTCS.2021.38 (cited on p. 58).

[BL23a] León Bohn and Christof Löding. "Constructing deterministic parity automata from positive and negative examples". In: *CoRR* abs/2302.11043 (2023). DOI: 10.48550/arXiv.2302.11043 (cited on pp. 19, 53, 124).

[BL23b] Udi Boker and Karoliina Lehtinen. "When a little nondeterminism goes a long way: an introduction to history-determinism". In: *ACM SIGLOG news* 10.1 (2023), pp. 24–51. DOI: 10.1145/3584676.3584682 (cited on pp. 12, 21, 38).

[BL69a] J. Richard Büchi and Lawrence H. Landweber. "Definability in the monadic second-order theory of successor". In: *J. Symb. Log.* 34.2 (1969), pp. 166–170. DOI: 10.2307/2271090 (cited on p. 246).

[BL69b] J. Richard Büchi and Lawrence H. Landweber. "Solving sequential conditions by finite-state strategies". In: *Transactions of the American Mathematical Society* 138 (1969), pp. 295–311 (cited on pp. 10, 17, 22, 24).

[Bok+13] Udi Boker, Denis Kuperberg, Orna Kupferman, and Michał Skrzypczak. "Nondeterminism in the presence of a diverse or unknown future". In: ICALP, 2013, pp. 89–100. DOI: 10.1007/978-3-642-39212-2_11 (cited on pp. 38, 40).

[Bok17] Udi Boker. "On the (in)succinctness of Muller automata". In: CSL, vol. 82. 2017, 12:1–12:16. DOI: 10.4230/LIPIcs.CSL.2017.12 (cited on p. 20).

[Bok18] Udi Boker. "Why these automata types?" In: LPAR, vol. 57. EPiC Series in Computing. 2018, pp. 143–163. DOI: 10.29007/c3bj (cited on pp. 20, 41).

[Bok19] Udi Boker. "Inherent size blowup in ω-automata". In: DLT, vol. 11647. 2019, pp. 3–17. DOI: 10.1007/978-3-030-24886-4_1 (cited on pp. 20, 43, 103).

[Bou+08] Patricia Bouyer, Ulrich Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Jirí Srba. "Infinite runs in weighted timed automata with energy constraints". In: FORMATS, vol. 5215. Lecture Notes in Computer Science. 2008, pp. 33–47. DOI: 10.1007/978-3-540-85778-5_4 (cited on pp. 23, 235).

[Bou+20] Patricia Bouyer, Stéphane Le Roux, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. "Games where you can play optimally with arena-independent finite memory". In: CONCUR, vol. 171. 2020, 24:1–24:22. DOI: 10.4230/LIPIcs.CONCUR.2020.24 (cited on pp. 24, 161, 179).

[Bou+23] Patricia Bouyer, Nathanaël Fijalkow, Mickael Randour, and Pierre Vandenhove. "How to play optimally for regular objectives?" In: ICALP, vol. 261. LIPIcs. 2023, 118:1–118:18. DOI: 10.4230/LIPIcs.ICALP.2023.118 (cited on pp. 24, 161, 258).

[Bra98] J. C. Bradfield. "Simplifying the modal mu-calculus alternation hierarchy". In: STACS, 1998, pp. 39–49 (cited on p. 20).

[BRT22] Patricia Bouyer, Stéphane Le Roux, and Nathan Thomasset. "Finite-memory strategies in two-player infinite games". In: CSL, vol. 216. 2022, 8:1–8:16. DOI: 10.4230/LIPIcs.CSL.2022.8 (cited on p. 25).

[BRV23] Patricia Bouyer, Mickael Randour, and Pierre Vandenhove. "Characterizing omega-regularity through finite-memory determinacy of games on infinite graphs". In: *Theoretics* 2 (2023). DOI: 10.46298/theoretics.23.1 (cited on pp. 24, 161, 178, 179, 234, 235, 237).

[Büc60] J. Richakd Büchi. "Weak second-order arithmetic and finite automata". In: *Zeitschrift für Math. Log. und Grundl. der Math.* 6 (1960), pp. 66–92. DOI: 10.1007/978-1-4613-8928-6_22 (cited on p. 16).

[Büc77] J. Richard Büchi. "Using determinancy of games to eliminate quantifiers". In: FCT, vol. 56. Lecture Notes in Computer Science. Springer, 1977, pp. 367–378. DOI: 10.1007/3-540-08442-8_104 (cited on p. 17).

[Büc83] J. Richard Büchi. "State-strategies for games in $F_{\sigma\delta} \bigcap G_{\delta\sigma}$". In: *The Journal of Symbolic Logic* 48.4 (1983), pp. 1171–1198 (cited on pp. 16, 17, 24).

[Bü62] J. Richard Büchi. "On a decision method in restricted second order arithmetic". In: *Proc. Internat. Congr. on Logic, Methodology and Philosophy of Science* (1962), pp. 1–11 (cited on pp. 10, 16, 45, 246, 249).

[Cal+17] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. "Deciding parity games in quasipolynomial time". In: STOC, ACM, 2017, pp. 252–263. DOI: 10.1145/3055399.3055409 (cited on p. 21).

[Cal+22] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. "Deciding parity games in quasi-polynomial time". In: *SIAM Journal on Computing* 51.2 (2022), STOC17–152–STOC17–188. DOI: 10.1137/17M1145288 (cited on p. 264).

[CDK93] Edmund M. Clarke, I. A. Draghicescu, and Robert P. Kurshan. "A unified approch for showing language inclusion and equivalence between various types of omega-automata". In: *Inf. Process. Lett.* 46.6 (1993), pp. 301–308. DOI: 10.1016/0020-0190(93)90069-L (cited on pp. 20, 44, 139, 144).

[CF13] Krishnendu Chatterjee and Nathanaël Fijalkow. "Infinite-state games with finitary conditions". In: CSL, vol. 23. 2013, pp. 181–196. DOI: 10.4230/LIPIcs.CSL.2013.181 (cited on p. 24).

[CF18] Thomas Colcombet and Nathanaël Fijalkow. "Parity games and universal graphs". In: *Corr* abs/1810.05106 (2018) (cited on pp. 24, 254).

[CF19] Thomas Colcombet and Nathanaël Fijalkow. "Universal graphs and good for games automata: new tools for infinite duration games". In: FOSSACS, vol. 11425. 2019, pp. 1–26. DOI: 10.1007/978-3-030-17127-8_1 (cited on p. 24).

[CFH14] Thomas Colcombet, Nathanaël Fijalkow, and Florian Horn. "Playing safe". In: FSTTCS, vol. 29. 2014, pp. 379–390. DOI: 10.4230/LIPIcs.FSTTCS.2014.379 (cited on pp. 24, 161, 195, 237, 238, 258, 259).

[Chu57] Alonzo Church. "Application of recursive arithmetic to the problem of circuit synthesis". In: Summaries of talks presented at the Summer Institute for Symbolic Logic, Cornell University, 1957 (cited on pp. 9, 16).

[CL08] Thomas Colcombet and Christof Löding. "The non-deterministic Mostowski hierarchy and distance-parity automata". In: ICALP, vol. 5126. 2008, pp. 398–409. DOI: 10.1007/978-3-540-70583-3_33 (cited on pp. 20, 25, 45).

[CM03] Olivier Carton and Max Michel. "Unambiguous Büchi automata". In: *Theoretical computer science* 297.1 (2003), pp. 37 –81. DOI: 10.1016/S0304-3975(02)00618-7 (cited on p. 70).

[CM99] Olivier Carton and Ramón Maceiras. "Computing the Rabin index of a parity automaton". In: *RAIRO* (1999), pp. 495–506. DOI: 10.1051/ita:1999129 (cited on pp. 20, 46, 53, 91, 107, 117, 124, 152).

[CN06] Thomas Colcombet and Damian Niwiński. "On the positional determinacy of edge-labeled games". In: *Theor. Comput. Sci.* 352.1-3 (2006), pp. 190–196. DOI: 10.1016/j.tcs.2005.10.046 (cited on pp. 21, 23, 178, 180, 234, 251, 252).

[Col09] Thomas Colcombet. "The theory of stabilisation monoids and regular cost functions". In: ICALP, 2009, pp. 139–150. DOI: 10.1007/978-3-642-02930-1_12 (cited on pp. 21, 58).

[Col12] Thomas Colcombet. "Forms of Determinism for Automata (Invited Talk)". In: STACS, vol. 14. 2012, pp. 1–23. DOI: 10.4230/LIPIcs.STACS.2012.1 (cited on p. 60).

[Col13] Thomas Colcombet. "Fonctions régulières de coût". habilitation. Université Paris Diderot – Paris 7, 2013 (cited on p. 25).

[Col15] Thomas Colcombet. "Unambiguity in automata theory". In: DCFS, vol. 9118. Lecture Notes in Computer Science. 2015, pp. 3–18. DOI: 10.1007/978-3-319-19225-3_1 (cited on p. 70).

[Col+22] Thomas Colcombet, Nathanaël Fijalkow, Pawel Gawrychowski, and Pierre Ohlmann. "The theory of universal graphs for infinite duration games". In: *Log. Methods Comput. Sci.* 18.3 (2022). DOI: 10.46298/lmcs-18(3:29)2022 (cited on p. 254).

[CP23] Michaël Cadilhac and Guillermo A. Pérez. "Acacia-bonsai: A modern implementation of downset-based LTL realizability". In: TACAS, vol. 13994. Lecture Notes in Computer Science. 2023, pp. 192–207. DOI: 10.1007/978-3-031-30820-8_14 (cited on p. 253).

[CZ09] Thomas Colcombet and Konrad Zdanowski. "A tight lower bound for determinization of transition labeled Büchi automata". In: ICALP, 2009, pp. 151–162. DOI: 10.1007/978-3-642-02930-1_13 (cited on pp. 22, 249).

[Dav64] Morton Davis. "Infinite games of perfect information". In: Advances in Game Theory. (AM-52), vol. 52. 1964, pp. 85–102. DOI: doi:10.1515/9781400882014-008 (cited on p. 22).

[Dij18] Tom van Dijk. "Oink: an implementation and evaluation of modern parity game solvers". In: TACAS, vol. 10805. Lecture Notes in Computer Science. 2018, pp. 291–308. DOI: 10.1007/978-3-319-89960-2_16 (cited on p. 18).

[DJW97] Stefan Dziembowski, Marcin Jurdziński, and Igor Walukiewicz. "How much memory is needed to win infinite games?" In: LICS, 1997, pp. 99–110. DOI: 10.1109/LICS.1997.614939 (cited on pp. 24, 51, 71, 79, 84, 160–162, 164, 168, 173, 174, 258, 278, 282).

[DL+22] Alexandre Duret-Lutz, Etienne Renault, Maximilien Colange, Florian Renkin, Alexandre Gbaguidi Aisse, Philipp Schlehuber-Caissier, Thomas Medioni, Antoine Martin, Jérôme Dubois, Clément Gillard, and Henrich Lauko. "From Spot 2.0 to Spot 2.10: what's new?" In: CAV, vol. 13372. Lecture Notes in Computer Science. 2022, pp. 174–187. DOI: 10.1007/978-3-031-13188-2_9 (cited on pp. 20, 53, 103, 109, 160, 253).

[EJ91] E. Allen Emerson and Charanjit S. Jutla. "Tree automata, mu-calculus and determinacy (extended abstract)". In: FOCS, 1991, pp. 368–377. DOI: 10.1109/SFCS.1991.185392 (cited on pp. 16, 17, 19–21, 23, 178, 189, 198, 269).

[EJ99] E. Allen Emerson and Charanjit S. Jutla. "The complexity of tree automata and logics of programs". In: *Siam j. comput.* 29.1 (1999), 132–158. DOI: 10.1137/S0097539793304741 (cited on p. 21).

[EJS93] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. "On model-checking for fragments of μ-calculus". In: CAV, vol. 697. Lecture Notes in Computer Science. 1993, pp. 385–396. DOI: 10.1007/3-540-56922-7_32 (cited on pp. 17, 21).

[EKS18] Javier Esparza, Jan Kretínský, and Salomon Sickert. "One theorem to rule them all: A unified translation of LTL into ω-automata". In: LICS, 2018, pp. 384–393. DOI: 10.1145/3209108.3209161 (cited on p. 18).

[EL85] E. Allen Emerson and Chin-Laung Lei. "Modalities for model checking (extended abstract): branching time strikes back". In: POPL, ACM. 1985, pp. 84–96. DOI: 10.1145/318593.318620 (cited on p. 19).

[Elg61] Calvin C. Elgot. "Decision problems of finite automata design and related arithmetics". In: *Transactions of the American Mathematical Society* 98.1 (1961), pp. 21–51 (cited on p. 16).

[EM79] Andrzej Ehrenfeucht and Jan Mycielski. "Positional strategies for mean payoff games". In: *International Journal of Game Theory* 8.2 (1979), pp. 109–113. DOI: 10.1007/BF01768705 (cited on p. 23).

[Eme85] E. Allen Emerson. “Automata, tableaux, and temporal logics”. In: Logics of Programs, 1985, pp. 79–88 (cited on p. 169).

[ES22] Rüdiger Ehlers and Sven Schewe. “Natural colors of infinite words”. In: FSTTCS, vol. 250. 2022, 36:1–36:17. DOI: 10.4230/LIPIcs.FSTTCS.2022.36 (cited on pp. 19, 53, 124).

[Esp+17] Javier Esparza, Jan Křetínský, Jean-François Raskin, and Salomon Sickert. “From LTL and limit-deterministic Büchi automata to deterministic parity automata”. In: TACAS, 2017, pp. 426–442. DOI: 10.1007/978-3-662-54577-5_25 (cited on p. 18).

[Fij11] Nathanaël Fijalkow. “Exercises in reachability style”. MA thesis. LIAFA, Paris, 2011 (cited on p. 259).

[Fij+23] Nathanaël Fijalkow, Nathalie Bertrand, Patricia Bouyer-Decitre, Romain Brenguier, Arnaud Carayol, John Fearnley, Hugo Gimbert, Florian Horn, Rasmus Ibsen-Jensen, Nicolas Markey, Benjamin Monmege, Petr Novotný, Mickael Randour, Ocan Sankur, Sylvain Schmitz, Olivier Serre, and Mateusz Skomra. *Games on graphs*. Ed. by Nathanaël Fijalkow. Online, 2023 (cited on pp. 23, 246).

[FL09] Oliver Friedmann and Martin Lange. “Solving parity games in practice”. In: ATVA, 2009, pp. 182–196 (cited on pp. 53, 124).

[FZ14] Nathanaël Fijalkow and Martin Zimmermann. “Parity and Streett games with costs”. In: *Log. Methods Comput. Sci.* 10.2 (2014). DOI: 10.2168/LMCS-10(2:14)2014 (cited on p. 24).

[GH82] Yuri Gurevich and Leo Harrington. “Trees, automata, and games”. In: STOC, 1982, pp. 60–65. DOI: 10.1145/800070.802177 (cited on pp. 16, 17, 24, 50, 171).

[GL02] Dimitra Giannakopoulou and Flavio Lerda. “From states to transitions: improving translation of LTL formulae to Büchi automata”. In: FORTE, 2002, pp. 308–326 (cited on p. 253).

[Göd31] Kurt Gödel. “Über formal unentscheidbare sätze der principia mathematica und verwandter systeme I”. In: *Monatshefte für Mathematik und Physik* 38.1 (1931), pp. 173–198. DOI: 10.1007/BF01700692 (cited on p. 15).

[GS53] David Gale and F. M. Stewart. “Infinite games with perfect information”. In: Contributions to the Theory of Games (AM-28), Volume II, Princeton University Press, 1953, pp. 245–266. DOI: doi:10.1515/9781400881970-014 (cited on p. 17).

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, eds. *Automata logics, and infinite games*. Springer, Berlin, Heidelberg, 2002. DOI: 10.1007/3-540-36387-4 (cited on p. 246).

[Gut96] Norbert Gutleben. “On the minimization of Muller-automata”. In: *Naturwissenschaften und informatik*. Rep. Reihe Informatik I-9 (1996) (cited on pp. 19, 136).

[GW06] Erich Grädel and Igor Walukiewicz. “Positional determinacy of games with infinitely many priorities”. In: *Log. methods comput. sci.* 2.4 (2006). DOI: 10.2168/LMCS-2(4:6)2006 (cited on pp. 170, 263).

[GZ05] Hugo Gimbert and Wieslaw Zielonka. "Games where you can play optimally without any memory". In: CONCUR, vol. 3653. 2005, pp. 428–442. DOI: 10.1007/11539452_33 (cited on pp. 23, 178, 179, 201).

[HD05] Paul Hunter and Anuj Dawar. "Complexity bounds for regular games". In: MFCS, 2005, pp. 495–506. DOI: 10.1007/11549345_43 (cited on pp. 21, 43, 44, 103, 104).

[HD08] Paul Hunter and Anuj Dawar. "Complexity bounds for Muller games". In: *Theoretical Computer Science (TCS)* (2008) (cited on pp. 285, 286).

[Hop71] John E. Hopcroft. *An n log n algorithm for minimizing states in a finite automaton*. Tech. rep. Stanford University, 1971. DOI: 10.5555/891883 (cited on p. 136).

[Hor08] Florian Horn. "Explicit Muller games are PTIME". In: FSTTCS, 2008, pp. 235–243. DOI: 10.4230/LIPIcs.FSTTCS.2008.1756 (cited on p. 44).

[Hor09] Florian Horn. "Random fruits on the Zielonka tree". In: STACS, vol. 3. 2009, pp. 541–552. DOI: 10.4230/LIPIcs.STACS.2009.1848 (cited on pp. 24, 51, 161).

[HP06] Thomas A. Henzinger and Nir Piterman. "Solving games without determinization". In: Computer Science Logic, 2006, pp. 395–410. DOI: 10.1007/11874683_26 (cited on pp. 21, 38, 58, 96, 136, 278, 279).

[HR72] Robert Hossley and Charles Rackoff. "The emptiness problem for automata on infinite trees". In: Annual Symposium on Switching and Automata Theory, 1972, pp. 121–124. DOI: 10.1109/SWAT.1972.28 (cited on p. 16).

[Hug23] Christopher Hugenroth. "Zielonka DAG acceptance, regular languages over infinite words". In: DLT, 2023 (cited on pp. 20, 43, 103, 122).

[Jac+22] Swen Jacobs, Guillermo A. Perez, Remco Abraham, Veronique Bruyere, Michael Cadilhac, Maximilien Colange, Charly Delfosse, Tom van Dijk, Alexandre Duret-Lutz, Peter Faymonville, Bernd Finkbeiner, Ayrat Khalimov, Felix Klein, Michael Luttenberger, Klara Meyer, Thibaud Michaud, Adrien Pommellet, Florian Renkin, Philipp Schlehuber-Caissier, Mouhammad Sakr, Salomon Sickert, Gaetan Staquet, Clement Tamines, Leander Tentrup, and Adam Walker. *The reactive synthesis competition (SYNTCOMP): 2018-2021*. 2022 (cited on pp. 18, 253).

[Jur98] Marcin Jurdziński. "Deciding the winner in parity games is in UP ∩ co-UP". In: *Inf. Process. Lett.* 68.3 (1998), pp. 119–124. DOI: 10.1016/S0020-0190(98)00150-1 (cited on p. 21).

[Kam85] Michael Kaminski. "A classification of ω-regular languages". In: *Theoretical Computer Science* 36 (1985), pp. 217–229. DOI: 10.1016/0304-3975(85)90043-X (cited on p. 45).

[Kar72] Richard M. Karp. "Reducibility among combinatorial problems". In: Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, The IBM Research Symposia Series. Springer US, 1972, pp. 85–103. DOI: 10.1007/978-1-4684-2001-2_9 (cited on pp. 267, 268).

[Kla94] Nils Klarlund. "Progress measures, immediate determinacy, and a subset construction for tree automata". In: *Annals of pure and applied logic* 69.2 (1994), pp. 243–268. DOI: 10.1016/0168-0072(94)90086-8 (cited on pp. 24, 84, 169, 171, 178).

[KMM06] Orna Kupferman, Gila Morgenstern, and Aniello Murano. "Typeness for omega-regular automata". In: *Int. J. Found. Comput. Sci.* 17.4 (2006), pp. 869–884. DOI: 10.1142/S0129054106004157 (cited on pp. 19, 110).

[KMS18] Jan Kretínský, Tobias Meggendorfer, and Salomon Sickert. "Owl: A library for ω-words, automata, and LTL". In: ATVA, vol. 11138. Lecture Notes in Computer Science. 2018, pp. 543–550. DOI: 10.1007/978-3-030-01090-4_34 (cited on pp. 53, 103, 109, 253).

[Kop06] Eryk Kopczyński. "Half-positional determinacy of infinite games". In: ICALP, 2006, pp. 336–347. DOI: 10.1007/11787006_29 (cited on pp. 160, 161).

[Kop07] Eryk Kopczyński. "Omega-regular half-positional winning conditions". In: CSL, vol. 4646. Lecture Notes in Computer Science. 2007, pp. 41–53. DOI: 10.1007/978-3-540-74915-8_7 (cited on p. 178).

[Kop08] Eryk Kopczyński. "Half-positional determinacy of infinite games". PhD thesis. University of Warsaw, 2008 (cited on pp. 24, 160, 161, 163, 166, 167, 170, 178, 179, 190, 193, 201, 243, 251, 258).

[Koz22a] Alexander Kozachinskiy. "Energy games over totally ordered groups". In: *CoRR* abs/2205.04508 (2022). DOI: 10.48550/arXiv.2205.04508 (cited on pp. 24, 179, 235).

[Koz22b] Alexander Kozachinskiy. "Infinite separation between general and chromatic memory". In: *CoRR* abs/2208.02691 (2022). DOI: 10.48550/arXiv.2208.02691 (cited on pp. 25, 161).

[Koz22c] Alexander Kozachinskiy. "One-to-two-player lifting for mildly growing memory". In: STACS, vol. 219. LIPIcs. 2022, 43:1–43:21. DOI: 10.4230/LIPIcs.STACS.2022.43 (cited on pp. 25, 179).

[Koz22d] Alexander Kozachinskiy. "State complexity of chromatic memory in infinite-duration games". In: *CoRR* abs/2201.09297 (2022) (cited on pp. 25, 161).

[Koz83] Dexter Kozen. "Results on the propositional mu-calculus". In: *Theor. Comput. Sci.* 27 (1983), pp. 333–354. DOI: 10.1016/0304-3975(82)90125-6 (cited on p. 17).

[KP09] Orna Kupferman and Nir Piterman. "Lower bounds on witnesses for nonemptiness of universal co-büchi automata". In: FOSSACS, vol. 5504. 2009, pp. 182–196. DOI: 10.1007/978-3-642-00596-1_14 (cited on p. 20).

[KPB94] Sriram C. Krishnan, Anuj Puri, and Robert K. Brayton. "Deterministic ω-automata vis-a-vis deterministic Büchi automata". In: ISAAC, vol. 834. Lecture Notes in Computer Science. 1994, pp. 378–386. DOI: 10.1007/3-540-58325-4_202 (cited on pp. 19, 53, 110, 118).

[KPB95] Sriram C. Krishnan, Anuj Puri, and Robert K. Brayton. "Structural complexity of omega-automata". In: STACS, 1995, pp. 143–156. DOI: 10.1007/3-540-59042-0_69 (cited on pp. 19, 20, 46, 110, 119).

[Kře+17] Jan Křetínský, Tobias Meggendorfer, Clara Waldmann, and Maximilian Weininger. "Index appearance record for transforming Rabin automata into parity automata". In: TACAS, 2017, pp. 443–460. DOI: 10.1007/978-3-662-54577-5_26 (cited on p. 51).

[Kre+18] Jan Kretínský, Tobias Meggendorfer, Salomon Sickert, and Christopher Ziegler. "Rabinizer 4: from LTL to your favourite deterministic automaton". In: CAV, vol. 10981. Lecture Notes in Computer Science. 2018, pp. 567–577. DOI: 10.1007/978-3-319-96145-3_30 (cited on p. 253).

[Kře+21] Jan Křetínský, Tobias Meggendorfer, Clara Waldmann, and Maximilian Weininger. "Index appearance record with preorders". In: *Acta informatica* (2021). DOI: 10.1007/s00236-021-00412-y (cited on p. 51).

[KS15] Denis Kuperberg and Michał Skrzypczak. "On determinisation of good-for-games automata". In: ICALP, 2015, pp. 299–310. DOI: 10.1007/978-3-662-47666-6_24 (cited on pp. 21, 53, 96, 124, 136, 184, 205).

[KSV96] Orna Kupferman, Shmuel Safra, and Moshe Y. Vardi. "Relating word and tree automata". In: LICS, 1996, pp. 322–332. DOI: 10.1109/LICS.1996.561360 (cited on pp. 20, 21).

[Kup12] Orna Kupferman. "Recent challenges and ideas in temporal synthesis". In: SOFSEM, 2012, pp. 88–98 (cited on p. 18).

[Kup18] Orna Kupferman. "Automata theory and model checking". In: Handbook of Model Checking, ed. by Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. Springer International Publishing, 2018, pp. 107–151. DOI: 10.1007/978-3-319-10575-8_4 (cited on p. 246).

[Kup22] Orna Kupferman. "Using the past for resolving the future". In: *Frontiers Comput. Sci.* 4 (2022). DOI: 10.3389/fcomp.2022.1114625 (cited on pp. 12, 21).

[KV05] Orna Kupferman and Moshe Y. Vardi. "Safraless decision procedures". In: FOCS, 2005, pp. 531–542. DOI: 10.1109/SFCS.2005.66 (cited on p. 18).

[Lan69] Lawrence H. Landweber. "Decision problems for omega-automata". In: *Math. Syst. Theory* 3.4 (1969), pp. 376–384. DOI: 10.1007/BF01691063 (cited on pp. 18, 246).

[LMS20] Michael Luttenberger, Philipp J. Meyer, and Salomon Sickert. "Practical synthesis of reactive systems from LTL specifications via parity games". In: *Acta informatica* (2020), pp. 3–36. DOI: 10.1007/s00236-019-00349-3 (cited on pp. 18, 20, 50, 160, 253).

[Löd01] Christof Löding. "Efficient minimization of deterministic weak omega-automata". In: *Inf. process. lett.* 79.3 (2001), pp. 105–109. DOI: 10.1016/S0020-0190(00)00183-6 (cited on pp. 19, 136).

[Löd98] Christof Löding. "Methods for the transformation of omega-automata: Complexity and connection to second order logic". MA thesis. Christian-Albrechts-Universität of Kiel, 1998 (cited on p. 20).

[Löd99] Christof Löding. "Optimal bounds for transformations of ω-automata". In: FSTTCS, 1999, 97–109. DOI: 10.1007/3-540-46691-6_8 (cited on pp. 20, 51, 102, 105, 110).

[Lov73] László Lovász. "Coverings and coloring of hypergraphs". In: Proceedings of the Fourth Southeastern Conference on Combinatorics, Graph Theory, and Computing, 1973, pp. 3–12 (cited on p. 267).

[LP19] Christof Löding and Anton Pirogov. "Determinization of Büchi automata: unifying the approaches of Safra and Muller-Schupp". In: ICALP, 2019, 120:1–120:13. DOI: 10.4230/LIPIcs.ICALP.2019.120 (cited on p. 50).

[LT21] Christof Löding and Wolfgang Thomas. "Automata on finite trees". In: Handbook of Automata Theory, ed. by Jean-Éric Pin. European Mathematical Society Publishing House, Zürich, Switzerland, 2021, pp. 235–264. DOI: 10.4171/Automata-1/7 (cited on p. 246).

[Lut08] Michael Luttenberger. "Strategy iteration using non-deterministic strategies for solving parity games". In: *Corr* abs/0806.2923 (2008) (cited on p. 263).

[Maj+19a] Juraj Major, Frantisek Blahoudek, Jan Strejcek, Miriama Sasaráková, and Tatiana Zboncáková. "Ltl3tela: LTL to small deterministic or nondeterministic Emerson-Lei automata". In: ATVA, vol. 11781. 2019, pp. 357–365. DOI: 10.1007/978-3-030-31784-3_21 (cited on p. 20).

[Maj+19b] Juraj Major, Frantisek Blahoudek, Jan Strejcek, Miriama Sasaráková, and Tatiana Zboncáková. "Ltl3tela: LTL to small deterministic or nondeterministic Emerson-Lei automata". In: ATVA, vol. 11781. Lecture Notes in Computer Science. 2019, pp. 357–365. DOI: 10.1007/978-3-030-31784-3_21 (cited on p. 253).

[Mar75] Donald A. Martin. "Borel determinacy". In: *Annals of Mathematics* 102.2 (1975), pp. 363–371 (cited on pp. 22, 34).

[MC18] Thibaud Michaud and Maximilien Colange. "Reactive synthesis from LTL specification with Spot". In: SYNT@CAV, Electronic Proceedings in Theoretical Computer Science. 2018 (cited on pp. 18, 20, 50, 253).

[McN66] Robert McNaughton. "Testing and generating infinite sequences by a finite automaton". In: *Information and control* 9.5 (1966), pp. 521–530. DOI: 10.1016/S0019-9958(66)80013-X (cited on pp. 18, 19, 24, 44–46, 246).

[McN93] Robert McNaughton. "Infinite games played on finite graphs". In: *Annals of Pure and Applied Logic* (1993). DOI: 10.1016/0168-0072(93)90036-D (cited on pp. 24, 246, 278, 283).

[Mos80] Yiannis N. Moschovakis. *Descriptive set theory*. Elsevier Science Limited, 1980 (cited on p. 17).

[Mos84] Andrzej W. Mostowski. "Regular expressions for infinite trees and a standard form of automata". In: SCT, 1984, pp. 157–168. DOI: 10.1007/3-540-16066-3_15 (cited on pp. 19, 20, 44, 45, 104).

[Mos91] Andrzej W. Mostowski. *Games with forbidden positions*. Tech. rep. 78. University of Gdansk, 1991 (cited on pp. 20, 23).

[MP95] Oded Maler and Amir Pnueli. "On the learnability of infinitary regular sets". In: *Inf. Comput.* 118.2 (1995), pp. 316–326. DOI: 10.1006/inco.1995.1070 (cited on pp. 19, 118).

[MPR21] Benjamin Monmege, Julie Parreaux, and Pierre-Alain Reynier. "Playing stochastically in weighted timed games to emulate memory". In: ICALP, vol. 198. 2021, 137:1–137:17. DOI: 10.4230/LIPIcs.ICALP.2021.137 (cited on p. 25).

[MR14] Dhruv Mubayi and Vojtech Rödl. "Specified intersections". In: *Transactions of the American Mathematical Society* 366.1 (2014), pp. 491–504 (cited on pp. 138, 156, 157).

[MS17] David Müller and Salomon Sickert. "LTL to deterministic Emerson-Lei automata". In: GandALF, 2017, pp. 180–194. DOI: 10.4204/EPTCS.256.13 (cited on pp. 18, 20, 253).

[MS21a] Philipp Meyer and Salomon Sickert. *On the optimal and practical conversion of Emerson-Lei automata into parity automata*. Personal Communication. 2021 (cited on p. 51).

[MS21b] Philipp J. Meyer and Salomon Sickert. "Modernising strix". In: *Synt workshop* (2021) (cited on pp. 50, 253).

[MS84] David E. Muller and Paul E. Schupp. "Alternating automata on infinite objects, determinacy and Rabin's theorem". In: Automata on Infinite Words, Ecole de Printemps d'Informatique Théorique, vol. 192. Lecture Notes in Computer Science. 1984, pp. 100–107. DOI: 10.1007/3-540-15641-0_27 (cited on p. 16).

[MS97] Oded Maler and Ludwig Staiger. "On syntactic congruences for ω-languages". In: *Theoretical Computer Science* 183.1 (1997). Formal Language Theory, pp. 93–112. DOI: 10.1016/S0304-3975(96)00312-X (cited on pp. 18, 19, 136, 181).

[MSS86] David E. Muller, Ahmed Saoudi, and Paul E. Schupp. "Alternating automata, the weak monadic theory of the tree, and its complexity". In: ICALP, 1986, pp. 275–283 (cited on p. 55).

[Muc84] Andrei A. Muchnik. "Games on infinite trees and automata with dead-ends. A new proof for the decidability of the monadic second-order theory of two successors". In: Semiotics and Information, vol. 24. (English version in Bull. EATCS 48, 1992). 1984, pp. 17–40 (cited on p. 16).

[Mul63] David E. Muller. "Infinite sequences and finite machines". In: Symposium on Switching Circuit Theory and Logical Design, 1963, 3–16. DOI: 10.1109/SWCT.1963.8 (cited on pp. 17, 19, 246).

[Niw86] Damian Niwinski. "On fixed-point clones (extended abstract)". In: ICALP, vol. 226. 1986, pp. 464–473. DOI: 10.1007/3-540-16761-7_96 (cited on p. 20).

[NW04] Damian Niwinski and Igor Walukiewicz. "Deciding nondeterministic hierarchy of deterministic tree automata". In: WoLLIC, vol. 123. Electronic Notes in Theoretical Computer Science. 2004, pp. 195–208. DOI: 10.1016/j.entcs.2004.05.015 (cited on p. 20).

[NW98] Damian Niwiński and Igor Walukiewicz. "Relating hierarchies of word and tree automata". In: STACS, 1998, pp. 320–331. DOI: 10.1007/BFb0028571 (cited on pp. 20, 47).

[Ohl21] Pierre Ohlmann. "Monotonic graphs for parity and mean-payoff games. (graphes monotones pour jeux de parité et à paiement moyen)". PhD thesis. Université Paris Cité, France, 2021 (cited on pp. 178, 263).

[Ohl23a] Pierre Ohlmann. "Characterizing positionality in games of infinite duration over infinite graphs". In: *TheoretiCS* 2 (2023). DOI: 10.46298/theoretics.23.3 (cited on pp. 24, 162, 178, 179, 188, 191, 193, 194, 228, 252, 254, 258, 263).

[Ohl23b] Pierre Ohlmann. "Positionality of mean-payoff games on infinite graphs". In: *Corr* abs/2305.00347 (2023). DOI: 10.48550/arXiv.2305.00347 (cited on p. 24).

[Pit06] Nir Piterman. "From nondeterministic Büchi and Streett automata to deterministic parity automata". In: LICS, 2006, pp. 255–264. DOI: 10.1109/LICS.2006.28 (cited on pp. 50, 249).

[Pnu77] Amir Pnueli. "The temporal logic of programs". In: FOCS, 1977, pp. 46–57. DOI: 10.1109/SFCS.1977.32 (cited on p. 17).

[PP04] Dominique Perrin and Jean-Eric Pin. *Infinite words - automata, semigroups, logic and games*. Vol. 141. Pure and applied mathematics series. Elsevier Morgan Kaufmann, 2004 (cited on pp. 45, 246).

[PR89] Amir Pnueli and Roni Rosner. "On the synthesis of a reactive module". In: POPL, 1989, 179–190. DOI: 10.1145/75277.75293 (cited on pp. 17, 18).

[Pre29] Mojżesz Presburger. "Über die vollständigkeit eines gewissen systems der arithmetik ganzer zahlen, in welchem die addition als einzige operation hervortritt". In: Comptes Rendus du I congres de Mathe- maticiens des Pays Slaves, 1929, pp. 92–101 (cited on p. 15).

[Pur95] Anuj Puri. "Theory of hybrid systems and discrete event systems". PhD thesis. EECS Department, University of California, Berkeley, 1995 (cited on p. 23).

[Rab58] Michael O. Rabin. "Effective computability of winning strategies". In: *Journal of Symbolic Logic* 23.2 (1958), pp. 224–224. DOI: 10.2307/2964422 (cited on p. 22).

[Rab69] Michael O. Rabin. "Decidability of second-order theories and automata on infinite trees". In: *Transactions of the American Mathematical Society* 141 (1969), pp. 1–35 (cited on pp. 10, 16, 18, 19, 246).

[RDLP20] Florian Renkin, Alexandre Duret-Lutz, and Adrien Pommellet. "Practical "paritizing" of Emerson-Lei automata". In: ATVA, vol. 12302. Lecture Notes in Computer Science. 2020, pp. 127–143. DOI: 10.1007/978-3-030-59152-6_7 (cited on pp. 20, 51).

[Rou20] Stéphane Le Roux. "Time-aware uniformization of winning strategies". In: CiE, vol. 12098. Lecture Notes in Computer Science. 2020, pp. 193–204. DOI: 10.1007/978-3-030-51466-2_17 (cited on p. 161).

[Saë90] Bertrand Le Saëc. "Saturating right congruences". In: *RAIRO* 24 (1990), pp. 545–559. DOI: 10.1051/ita/1990240605451 (cited on pp. 18, 114, 181, 246, 254).

[Saf88] Schmuel Safra. "On the complexity of ω-automata". In: FOCS, 1988, 319–327. DOI: 10.1109/SFCS.1988.21948 (cited on pp. 18, 249).

[Sak98] Jacques Sakarovitch. "A construction on finite automata that has remained hidden". In: *Theor. Comput. Sci.* 204.1-2 (1998), pp. 205–231. DOI: 10.1016/S0304-3975(98)00040-1 (cited on pp. 52, 60).

[Sch08] Sven Schewe. "An optimal strategy improvement algorithm for solving parity and payoff games". In: CSL, vol. 5213. Lecture Notes in Computer Science. 2008, pp. 369–384. DOI: 10.1007/978-3-540-87531-4_27 (cited on p. 263).

[Sch09] Sven Schewe. "Tighter bounds for the determinisation of Büchi automata". In: FOSSACS, 2009, pp. 167–181. DOI: 10.1007/978-3-642-00596-1_13 (cited on pp. 50, 249).

[Sch10] Sven Schewe. "Beyond hyper-minimisation—minimising DBAs and DPAs is NP-complete". In: FSTTCS, vol. 8. 2010, pp. 400–411. DOI: 10.4230/LIPIcs.FSTTCS.2010.400 (cited on pp. 10, 19, 130, 136, 248, 249).

[Sch20] Sven Schewe. "Minimising Good-For-Games automata is NP-complete". In: FSTTCS, vol. 182. 2020, 56:1–56:13. DOI: 10.4230/LIPIcs.FSTTCS.2020.56 (cited on pp. 136, 246, 249).

[SE84] Robert S. Streett and E. Allen Emerson. "The propositional mu-calculus is elementary". In: ICALP, vol. 172. Lecture Notes in Computer Science. 1984, pp. 465–472. DOI: 10.1007/3-540-13345-3_43 (cited on p. 17).

[SE89] Robert S. Streett and E. Allen Emerson. "An automata theoretic decision procedure for the propositional mu-calculus". In: *Inf. comput.* 81 (1989), pp. 249–264 (cited on pp. 17, 189).

[Sha53] L. S. Shapley. "Stochastic games". In: *Proceedings of the National Academy of Sciences* 39.10 (1953), pp. 1095–1100. DOI: 10.1073/pnas.39.10.1095 (cited on p. 23).

[Skr13] Michał Skrzypczak. "Topological extension of parity automata". In: *Information and Computation* 228-229 (2013), pp. 16–27. DOI: 10.1016/j.ic.2013.06.004 (cited on pp. 20, 258).

[SL94] Bertrand Le Saëc and Igor Litovsky. "On the minimization problem for omega-automata". In: MFCS, vol. 841. Lecture Notes in Computer Science. 1994, pp. 504–514. DOI: 10.1007/3-540-58338-6_97 (cited on pp. 18, 136, 246, 254).

[Spe28] Emanuel Sperner. "Ein satz über untermengen einer endlichen menge". In: *Mathematische Zeitschrift* 27.1 (1928), pp. 544–548. DOI: 10.1007/BF01171114 (cited on p. 289).

[SPW91] Bertrand Le Saëc, Jean-Eric Pin, and Pascal Weil. "Semigroups with idempotent stabilizers and applications to automata theory". In: *Int. J. Algebra Comput.* 1.3 (1991), pp. 291–314. DOI: 10.1142/S0218196791000195 (cited on pp. 18, 246, 254).

[SS10] Jacques Sakarovitch and Rodrigo de Souza. "Lexicographic decomposition of k-valued transducers". In: *Theory Comput. Syst.* 47.3 (2010), pp. 758–785. DOI: 10.1007/s00224-009-9206-6 (cited on pp. 52, 60).

[SSM23] Tereza Schwarzová, Jan Strejček, and Juraj Major. "Reducing acceptance marks in Emerson-Lei automata by QBF solving". In: SAT, vol. 271. 2023, 23:1–23:20. DOI: 10.4230/LIPIcs.SAT.2023.23 (cited on pp. 20, 119).

[Sta83] Ludwig Staiger. "Finite-state ω-languages". In: *Journal of Computer and System Sciences* 27.3 (1983), pp. 434–448. DOI: 10.1016/0022-0000(83)90051-X (cited on pp. 18, 19).

[SV12] Sven Schewe and Thomas Varghese. "Tight bounds for the determinisation and complementation of generalised Büchi automata". In: ATVA, 2012, pp. 42–56. DOI: 10.1007/978-3-642-33386-6_5 (cited on p. 250).

[SV14] Sven Schewe and Thomas Varghese. "Determinising parity automata". In: MFCS, 2014, pp. 486–498. DOI: 10.1007/978-3-662-44522-8_41 (cited on p. 250).

[SW01] Ulrich Schwalbe and Paul Walker. "Zermelo and the early history of game theory". In: *Games and Economic Behavior* 34.1 (2001), pp. 123–137. DOI: 10.1006/game.2000.0794 (cited on p. 17).

[Tar49] Alfred Tarski. "A decision method for elementary algebra and geometry". In: *Journal of Symbolic Logic* 14.3 (1949), pp. 188–188. DOI: 10.2307/2267068 (cited on p. 15).

[Tar72] Robert Tarjan. "Depth first search and linear graph algorithms". In: *Siam Journal On Computing* 1.2 (1972). DOI: 10.1137/0201010 (cited on pp. 154, 291).

[Tho91] Wolfgang Thomas. "Automata on infinite objects". In: Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics, 1991 (cited on pp. 18, 195, 246).

[Tho95] Wolfgang Thomas. "On the synthesis of strategies in infinite games". In: STACS, 1995, pp. 1–13. DOI: 10.1007/3-540-59042-0_57 (cited on pp. 17, 22).

[Tho97] Wolfgang Thomas. "Languages, automata, and logic". In: Handbook of Formal Languages, Volume 3: Beyond Words, 1997, pp. 389–455. DOI: 10.1007/978-3-642-59126-6_7 (cited on p. 246).

[Tra58] Boris A. Trakhtenbrot. "The synthesis of logical nets whose operators are described in terms of monadic predicates". In: *Doklady AN SSR* 118 (1958). (In Russian), pp. 646–649 (cited on p. 16).

[Tra62] Boris A. Trakhtenbrot. "Finite automata and the monadic predicate calculus". In: *Siberian Math. Journal* 3 (1962). (In Russian), pp. 103–131 (cited on p. 16).

[Van23] Pierre Vandenhove. "Strategy complexity of zero-sum games on graphs. (Complexité des stratégies des jeux sur graphes à somme nulle)". PhD thesis. University of Mons, Belgium, 2023 (cited on pp. 24, 161, 179, 180, 243).

[Var14] Thomas Varghese. "Parity and generalised Büchi automata. Determinisation and complementation". PhD Thesis. University of Liverpool, 2014 (cited on p. 250).

[Wag79] Klaus Wagner. "On ω-regular sets". In: *Information and control* 43.2 (1979), pp. 123–177. DOI: 10.1016/S0019-9958(79)90653-3 (cited on pp. 18, 20, 26, 45, 47, 51, 85, 118, 137, 246, 259).

[Wal96] Igor Walukiewicz. "Pushdown processes: games and model checking". In: CAV, vol. 1102. Lecture Notes in Computer Science. 1996, pp. 62–74. DOI: 10.1007/3-540-61474-5_58 (cited on pp. 17, 189).

[Wil91] Thomas Wilke. "An Eilenberg theorem for infinity-languages". In: ICALP, vol. 510. 1991, pp. 588–599. DOI: 10.1007/3-540-54233-7_166 (cited on p. 18).

[WS21] Thomas Wilke and Sven Schewe. "ω-automata". In: Handbook of Automata Theory, ed. by Jean-Éric Pin. European Mathematical Society Publishing House, Zürich, Switzerland, 2021, pp. 189–234. DOI: 10.4171/Automata-1/6 (cited on p. 246).

[Zer13] Ernst Zermelo. "Über eine anwendung der mengenlehre auf die theorie des schachspiels". In: Proc. Fifth Congress Mathematicians (Cambridge 1912), Cambridge University Press, 1913, pp. 501–504 (cited on p. 17).

[Zie98] Wiesław Zielonka. "Infinite games on finitely coloured graphs with applications to automata on infinite trees". In: *Theoretical Computer Science* 200.1-2 (1998), pp. 135–183. DOI: 10.1016/S0304-3975(98)00009-7 (cited on pp. 21, 24, 42, 51, 53, 71, 72, 84, 111, 160, 161, 169, 170, 251, 252).

[ZP96] Uri Zwick and Mike Paterson. "The complexity of mean payoff games on graphs". In: *Theor. Comput. Sci.* 158.1&2 (1996), pp. 343–359. DOI: 10.1016/0304-3975(95)00188-3 (cited on p. 23).