TheoretiCS

From Muller to Parity and Rabin Automata: Optimal Transformations Preserving (History) Determinism

Antonio Casares [a] ✉ ⓘ

Thomas Colcombet [b,c] ✉ ⓘ

Nathanaël Fijalkow [a,b,d] ✉ ⓘ

Karoliina Lehtinen [b,e] ✉ ⓘ

a LaBRI, Université de Bordeaux, France

b CNRS, France

c IRIF, Université Paris Cité, France

d MIMUW, University of Warsaw, Poland

e Aix-Marseille Université, LIS, France

ABSTRACT. We study transformations of automata and games using Muller conditions into equivalent ones using parity or Rabin conditions. We present two transformations, one that turns a deterministic Muller automaton into an equivalent deterministic parity automaton, and another that provides an equivalent history-deterministic Rabin automaton. We show a strong optimality result: the obtained automata are minimal amongst those that can be derived from the original automaton by duplication of states. We introduce the notions of locally bijective morphisms and history-deterministic mappings to formalise the correctness and optimality of these transformations.

The proposed transformations are based on a novel structure, called the alternating cycle decomposition, inspired by and extending Zielonka trees. In addition to providing optimal transformations of automata, the alternating cycle decomposition offers fundamental information on their structure. We use this information to give crisp characterisations on the possibility

of relabelling automata with different acceptance conditions and to perform a systematic study of a normal form for parity automata.

This document contains hyperlinks. Each occurrence of a notion 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.

1.   Introduction

Context

Games and automata for LTL synthesis. Games and automata over infinite words form the theoretical basis for the verification and synthesis of reactive systems; we refer to chapters 2, 4, and 27 of the recent Handbook of Model Checking [79, 58, 5] for a broad exposition of this research area. A milestone objective is the synthesis of reactive systems with specifications given in *Linear Temporal Logic* (LTL). The original approach of Pnueli and Rosner [80] using automata and games devised more than four decades ago is still at the heart of the state-of-the-art synthesis tools [39, 65, 70, 73]. The limiting factor in this method is the transformation of the LTL formula to a deterministic parity automaton. This automaton is then used to build a game, and a controller for the reactive system can be obtained from a winning strategy for this game. Most solutions to this problem (including the top-ranked tools in the SyntComp competitions [48], Strix [65, 69] and `ltlsynt` [70]) first construct a Muller (or Emerson-Lei) automaton, and then transform it into an equivalent parity automaton (we remark that, nevertheless, synthesis procedures avoiding the construction of deterministic automata have been proposed, for example, via the use of universal coBüchi automata [61]). 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 [64, 78, 88]. For this reason, understanding transformations of Muller automata and finding efficient procedures for them is of great importance.

Which are the simplest acceptance conditions? There exist multiple kinds of acceptance conditions that are commonly employed by ω-automata (Büchi, Rabin, Muller...). The use of parity conditions for LTL synthesis is justified by both practical and theoretical reasons. Firstly, there exist several high-performing algorithms solving parity games [33, 41, 48, 62, 92], so the last step in the LTL synthesis method described above can be carried out smoothly once the parity game is obtained. 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; it could even be argued that there is a canonical aspect to them:

— The optimal number of colours needed by a parity automaton to recognise a language L reveals a fundamental piece of 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 [74, 16] (and these hierarchies are closely related [60, 76]). In both cases, this index is a measure of the structural complexity of automata recognising L [93, 76] and of its topological complexity [3, 90]. 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 [75, 30], which is tightly related with the alternation depth of fixpoint operators in μ-calculus formulas [74]

— 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 (Proposition 6.4).

— Parity languages are bipositional [37] (in a parity game, both players can play optimally using positional strategies, that is, strategies that use no memory). Moreover, over infinite game graphs, these are the only bipositional languages [31], and over finite game graphs, these are the unique bipositional Muller languages [94].

— Solving parity games is both in NP and co-NP [38] (more precisely, the problem is in UP ∩ co-UP [49]). They can be solved in quasi-polynomial time [18], 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 [36] and PSPACE-complete [47].

However, these are not the only kind of conditions that deserve our attention. In this work, we further investigate transformations producing automata using a Rabin acceptance condition. Although in practice solvers for Rabin games are not as developed, Rabin languages are a natural choice and interesting from a theoretical point of view: they are exactly the half-positional Muller languages [94], there exists a correspondence between Rabin automata and memory structures for Muller games [21, 24], and the determinisation of Büchi automata naturally produces Rabin automata [40, 83, 88].

Transformations of games and automata. 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 [43] and is known as the Latest Appearance Record (LAR). Löding proved that the LAR is optimal in the worst case [63]: *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 [53, 63]. 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 [54, 68, 81]. 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 3.

History-deterministic automata. For the purposes of LTL synthesis and game transforma-tions, it is imperative to eliminate non-determinism from automata, since non-deterministic automata do not yield correct games. Unfortunately, deterministic automata can be exponen-tially larger than non-deterministic ones. Recently, an intermediate model of automata, named history-deterministic (also called good-for-games), has received considerable attention. The reason is that history-determinism exactly captures the features of deterministic automata that make them suitable for synthesis purposes, while being a less restrictive model. A natural question that arises is whether history-deterministic automata can be more succinct than de-terministic 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 [44, 28] that an example of an ω-regular language for which history-deterministic automata are smaller than deterministic ones was exhibited [57] (and it was even conjectured that such an automaton could not exist [27]). History-deterministic automata are the focus of several lines of research (we refer to the survey [14] for a detailed exposition). Despite this, a complete under-standing of history-deterministic automata remains elusive, and their scope of applicability is still uncertain. One key aspect that has not yet been addressed is how to design techniques as general as possible for building history-deterministic automata. To the best of our knowledge, the only existing result in this direction is a polynomial-time algorithm to minimise coBüchi history-deterministic automata [1].

The Zielonka tree and the alternating cycle decomposition. The starting point of our work is the notion of Zielonka tree, introduced by Zielonka [94] 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 [94], and, most importantly, it characterises their exact memory requirements, both in two-player games [34] and stochastic games [46].

　　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 [93], 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 work, we carry out 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 4.2). This construction implicitly appears in the long version of [34]. We show a strong optimality result: *for all* Muller language L, the parity automaton obtained from the Zielonka tree is minimal both amongst deterministic and history-deterministic parity automata recognising L (Theorem 4.15).[1] Moreover, it uses the optimal number of output colours to recognise L (Theorem 4.14). The optimality result we obtain is much stronger than the worst case optimality result of the LAR transformation [63], since it applies to every Muller language. In particular, our characterisation yields an algorithm to minimise deterministic parity automata recognising Muller languages in polynomial time (Theorem 6.32). 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 Muller language (Section 4.3), and prove that this automaton is minimal amongst history-deterministic Rabin automata (Theorem 4.51). 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 3). 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 kinds of morphisms will be of central importance: (1) locally bijective morphisms, which generalise composition with deterministic automata

---

1   The optimality of the Zielonka-tree-parity-automaton amongst deterministic automata has also been obtained in the independent unpublished work [68].

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 transition systems.** 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 these transition systems and their acceptance condition (Section 5). 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 5.34 and 5.35). 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 5.39). 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 $\mathcal{B}'$.

4. **Structural results for Muller transition systems.** The ACD does not only provide optimal transformations of games and automata, it also features 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 6.1). 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 [7, 11, 55, 94].

   In Section 6.2, we conduct a comprehensive examination of a normal form for parity automata. This normal form implicitly appears in [19], and has since proven instrumental in proofs about history-deterministic automata [1, 35, 57], positionality of ω-regular languages [15] and learning of ω-automata [6]. Similar normalisation procedures are commonly applied to parity games to speed up algorithms solving them [41]. 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.

Our model: transition systems and acceptance over edges. We want to point out a few technical details about the model used in this paper. First, we work with general transition systems 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, we define acceptance conditions over the edges of transitions systems – instead of over the vertices. This choice has been shown to yield more canonical results in theory, for instance, in the study of strategy complexity for games [15, 21, 31, 94], the determinisation of Büchi automata [32, 89], or the minimisation of history-deterministic automata [1, 35]. It has also proven to be more applicable in practical scenarios [2, 42]. We believe that the present work provides further evidence to this claim, as the minimal automaton obtained from the Zielonka tree, as well as the transformations based on the ACD, substantially rely on the use of edge-based acceptance.

Finally, we remark that in this work we are concerned with *state complexity*, that is, the efficiency of a construction is measured based on the number of states of the resulting transition system. We do not focus on the representation of the acceptance conditions; for instance, we will not differentiate between Muller or Emerson-Lei conditions, as they have the same expressive power (see also Remark 2.11).

Follow-up work. Despite its recent introduction [23], the alternating cycle decomposition has already found applications in both practical and theoretical scenarios. The ACD-parity-transform has been implemented in two open-source tools: Spot 2.10 [2] and Owl 21.0 [52], and it is used in the LTL-synthesis tools `ltlsynt` [2] and Strix [69]. These implementations were presented in the conference paper [25], where transformations based on the ACD are compared to the state-of-the-art existing paritizing methods.

The typeness results stemming from the ACD (Section 6.1) have also been proven instrumental in theoretical applications. They have been used to show a correspondence between Rabin automata and memory structures for games [21], and to provide lower bounds in the size of deterministic Rabin automata [24].

2.   Preliminaries

In this section, we introduce definitions that will be used throughout the paper.

Basic definitions. For a set A, we let $|A|$ denote its cardinality, 2^A its power set and $2^A_+ = 2^A \setminus \{\emptyset\}$. 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*. For a word $w \in \Sigma^\infty$ and $i \geq 0$ we write w_i to represent the i-th letter of w. 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 $v \in \Sigma^*$ and $u, w \in \Sigma^\infty$, we say that v is a *prefix* of u, and we write $v \sqsubseteq u$. For a word $w \in \Sigma^\omega$, we let $\mathsf{Inf}(w) = \{a \in \Sigma \mid w_i = a$ for infinitely many $i \in \mathbb{N}\}$.

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

Given a map $\alpha : A \to B$, we will extend α to words component-wise, i.e., $\alpha : A^\infty \to B^\infty$ will be defined as $\alpha(w_0 w_1 w_2 \dots) = \alpha(w_0)\alpha(w_1)\alpha(w_2)\dots$. We will use this convention throughout the paper without explicitly mentioning it. If $A' \subseteq A$, we denote $\alpha|_{A'}$ 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).

In this work, we will use the term *graph* to denote what is sometimes called a *directed multigraph*: A *graph* is a tuple $G = (V, E, \mathsf{Source}, \mathsf{Target})$, where V is a set of vertices, E a set of edges and $\mathsf{Source}: E \to V$ and $\mathsf{Target}: E \to V$ are maps indicating the source and target for each edge. A *path* is a (finite or infinite) sequence $\rho = e_0 e_1 \dots \in E^\infty$ 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 $\mathit{Path}_A^{\mathsf{fin}}(G)$ and $\mathit{Path}_A(G)$ denote, respectively, the set of finite and infinite paths on G starting from some $v \in A$ (we omit brackets if $A = \{v\}$ is a singleton). We let $\mathit{Path}_A^\infty(G) = \mathit{Path}_A^{\mathsf{fin}}(G) \cup \mathit{Path}_A(G)$. For a subset of vertices $A \subseteq V$ we write:

— $\mathsf{In}(A) = \{e \in E \mid \mathsf{Target}(e) \in A\}$,
— $\mathsf{Out}(A) = \{e \in E \mid \mathsf{Source}(e) \in A\}$.

All graphs considered in this paper will be finite.

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.

2.1 Transition systems, automata and games

Transition systems. A *pointed graph* $G = (V, E, \mathsf{Source}, \mathsf{Target}, I)$ is a graph together with a non-empty subset of *initial vertices* $I \subseteq V$. An *acceptance condition* over G is a tuple $\mathsf{Acc} = (\gamma, \Gamma, \mathbb{W})$ where Γ is a finite set of *colours*, $\gamma : E \to \Gamma \cup \{\varepsilon\}$ is an *edge-colouring* of G and $\mathbb{W} \subseteq \Gamma^\omega$ is a language of infinite words called the *acceptance set*. We allow *uncoloured edges* (ε-edges), but we impose the condition that no infinite path of G is eventually composed exclusively of ε-edges (that is, every cycle contains some edge e with $\gamma(e) \neq \varepsilon$).

A *transition system* (abbreviated *TS*) is a tuple $\mathcal{TS} = (G_{\mathcal{TS}}, \text{Acc}_{\mathcal{TS}})$, consisting in a pointed graph $G_{\mathcal{TS}} = (V, E, \text{Source}, \text{Target}, I)$, called the *underlying graph* of $\mathcal{TS}$, and an acceptance condition $\text{Acc}_{\mathcal{TS}} = (\gamma, \Gamma, \mathbb{W})$ 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 $\text{Source}(e) = v$, $\text{Target}(e) = v'$ and $\gamma(e) = c$. We will 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 $|\mathcal{TS}|$.

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, $\text{Source}(e_0) \in I$. We let $\mathcal{R}un^{\text{fin}}(\mathcal{TS})$ and $\mathcal{R}un(\mathcal{TS})$ be the set of finite and infinite runs on $\mathcal{TS}$, respectively, and we let $\mathcal{R}un^\infty(\mathcal{TS}) = \mathcal{R}un^{\text{fin}}(\mathcal{TS}) \cup \mathcal{R}un(\mathcal{TS})$. (We note that $\mathcal{R}un(\mathcal{TS}) = \mathcal{P}ath_I(G_{\mathcal{TS}})$.)

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

We say that a vertex $v \in V$ is *accessible* (or *reachable*) from a vertex v_0 if there exists a finite path from v_0 to v. We say that v 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 transition system is the set of accessible states. We define analogously the *accessible part from a vertex* v_0.

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.

REMARK 2.1. We remark that, whenever necessary, we can assume without loss of generality that in the acceptance condition $\text{Acc} = (\gamma, \Gamma, \mathbb{W})$ of a transition system, $\Gamma = E$ is the whole set of edges and γ is the identity function. Indeed, an equivalent acceptance condition can always be defined by using the acceptance set $\mathbb{W}' = \{w \in E^\omega \mid \gamma(w) \in \mathbb{W}\} \subseteq E^\omega$.

Automata. A *(non-deterministic) automaton* over Σ is an edge-labelled transition system $\mathcal{A} = (G_{\mathcal{A}}, \text{Acc}_{\mathcal{A}}, (l_\Sigma, \Sigma))$, where Σ is a finite set of *input letters*. Let $\mathcal{A}$ be an automaton with $G_{\mathcal{A}} = (Q, \Delta, \text{Source}, \text{Target}, I)$ as underlying graph and $\text{Acc}_{\mathcal{A}} = (\gamma, \Gamma, \mathbb{W})$ as acceptance condition. We write $e = q \xrightarrow{a:c} q'$ to denote that $e \in \Delta$ satisfies $l_\Sigma(e) = a$ and $\gamma(e) = c$. We can assume that $\Delta \subseteq Q \times \Sigma \times \Gamma \times Q$. We define:

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

We note that we can now recover the classical representation of an automaton as a tuple $\mathcal{A} = (Q, \Sigma, I, \Gamma, \delta, \mathbb{W})$, (or $(Q, \Sigma, I, \Gamma, \Delta, \mathbb{W})$) which we might use when working exclusively with automata.

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 assume that automata are complete without loss of generality by adding a sink state.

Given an automaton $\mathcal{A}$ and a word $w \in \Sigma^\infty$, a *run over w* in $\mathcal{A}$ is a run $\rho = e_0 e_1 \cdots \in \mathcal{R}un^\infty(\mathcal{A})$ such that $l_\Sigma(e_i) = w_i$ for all $i \geq 0$. A word $w \in \Sigma^\omega$ is *accepted* by $\mathcal{A}$ if there is a run over w that is accepting (that is, a run ρ such that $\gamma(\rho) \in \mathbb{W}$). 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*.

We remark that if $\mathcal{A}$ is deterministic (resp. complete), there is at most one (resp. at least one) run over w for each $w \in \Sigma^\infty$.

Given a subgraph G' of the underlying graph of an automaton $\mathcal{A}$ and a subset of states I' in G', the *subautomaton induced by G' with initial states I'* is the automaton having G' as underlying graph, I' as set of initial states, and whose acceptance condition and labelling with input letters are the restrictions of those of $\mathcal{A}$.

History-deterministic automata. Let $\mathcal{A}$ be a (non-deterministic) automaton over Σ with Δ as set of transitions and I as set of initial states. A *resolver* for $\mathcal{A}$ is a pair (r_0, r), consisting of a choice of an initial state,[2] $r_0 \in I$, and a function $r \colon \Delta^* \times \Sigma \to \Delta$ 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 $e_i = r(e_0 \ldots e_{i-1}, w_i)$ is actually a run over w in $\mathcal{A}$ starting from r_0. 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 $\mathcal{L}(\mathcal{A})$. An automaton $\mathcal{A}$ is called *history-deterministic* (shortened HD, also called *good-for-games* in the literature) if there is a sound resolver for it.

REMARK 2.2. Deterministic automata are history-deterministic, and they admit a unique resolver.

EXAMPLE 2.3. In Figure 1, we show an automaton $\mathcal{A}$ over $\Sigma = \{a, b, c\}$ that is not deterministic (as it has two b-transitions from q_1) but is history-deterministic. Its set of output colours is $\Gamma = \{1, 2\}$ and its acceptance set is $\mathbb{W} = \{u \in \{1, 2\}^\omega \mid u$ contains finitely many 1s$\}$ (this is a coBüchi condition, as introduced in the next section). It is easy to check that $\mathcal{A}$ recognises the language:

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

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.
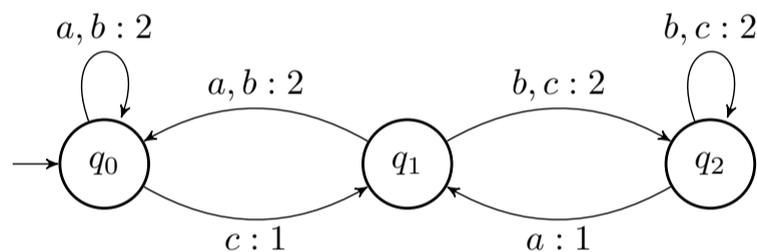

Figure 1. An example of a history-deterministic automaton that is not deterministic. The acceptance set is $\mathbb{W} = \{u \in \{1, 2\}^\omega \mid u$ contains finitely many 1s$\}$. An arrow of the form $q \xrightarrow{a,b:2} q'$ represents two different transitions with input letters a and b, respectively.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

◆

We say that a state q is *reachable using the resolver* (r_0, r) if there is a finite run $\rho = r_0 \rightsquigarrow q$ such that ρ is the run induced by r over some word $w \in \Sigma^*$.

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

REMARK 2.4. 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, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\tilde{\mathcal{A}})$.

The next lemma provides a simplification for automata recognising prefix-independent languages. Its proof can be found in Appendix C. Together with Remark 2.4, it indicates that when dealing with HD automata for this kind of languages, we can assume 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 2.5. *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, it holds that $\mathcal{A}$ recognises the same language if we fix q as initial state, that is, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_q)$. Moreover, $\mathcal{A}_q$ is also history-deterministic. In particular, if $\mathcal{A}$ is deterministic, this is the case for any reachable state q.*

Games. A *game* is a vertex-labelled transition system $\mathcal{G} = (G_\mathcal{G}, \mathrm{Acc}_\mathcal{G}, (l_{\mathrm{Players}}, \{\mathrm{Eve}, \mathrm{Adam}\}))$, with $G_\mathcal{G} = (V, E, \mathrm{Source}, \mathrm{Target}, I)$ a pointed graph, and $l_{\mathrm{Players}} \colon V \to \{\mathrm{Eve}, \mathrm{Adam}\}$ a vertex-labelling function inducing a partition of V into vertices controlled by two *players* that we refer to as *Eve* and *Adam*. We let $V_{\mathrm{Eve}} = l_{\mathrm{Players}}^{-1}(\mathrm{Eve})$ and $V_{\mathrm{Adam}} = l_{\mathrm{Players}}^{-1}(\mathrm{Adam})$.

During a play, players move a token from one vertex to another for an infinite amount of time. The player who owns the vertex v where the token is placed chooses an edge in $\mathrm{Out}(v)$ and the token travels through this edge to its target. In this way, they produce an infinite run ρ on $\mathcal{G}$ (that we also call a *play*). The objective of Eve is to produce an accepting run (a sequence of colours in $\mathbb{W}$), and Adam tries to prevent it.

A *strategy* from $v \in V$ for Eve is a (partial) function $\mathrm{strat}_v \colon \mathit{Path}_v^{\mathrm{fin}}(\mathcal{G}) \rightharpoonup E$, defined for finite paths from v ending in a vertex in V_{Eve}, that tells Eve which move to choose after any possible finite play. We say that a play $\rho \in \mathit{Path}_v^\infty(\mathcal{G})$ is *consistent with the strategy* strat_v if after each finite prefix $\rho' \sqsubseteq \rho$ ending in a vertex controlled by Eve, the next edge in ρ is $\mathrm{strat}_v(\rho')$. We say that strat_v is a *winning strategy for Eve* if all infinite plays from v consistent with strat_v are accepting. We say that Eve *wins* the game $\mathcal{G}$ from v if there is a winning strategy from v for her. Strategies for Adam are defined dually.

Given a game $\mathcal{G}$, the *winning region* of $\mathcal{G}$ for Eve, written $\mathcal{W}_{\mathrm{Eve}}(\mathcal{G})$, is the set of initial vertices $v \in I$ such that she wins the game $\mathcal{G}$ from v. The *full winning region* of $\mathcal{G}$ for Eve is her winning region in the game $\mathcal{G}_V$ where all vertices are initial, that is, the set of vertices $v \in V$ such that Eve wins the game $\mathcal{G}$ from v.

In some proofs, we will need to take a close look into the strategies used in games, for which we need to introduce *finite memory strategies*. For a set X (usually the set of edges of a game), we define a *memory skeleton* over X as an edge-labelled pointed graph $\mathcal{M} = (M, E_M, \mathrm{Source}, \mathrm{Target}, m_0)$ with a single initial state m_0 and labels $l_M \colon E_M \to X$ inducing a deterministic structure, that is, satisfying that for each $m \in M$ and $x \in X$ there is at most one transition $e \in \mathrm{Out}(m)$ labelled x. We denote $\mu \colon M \times X \rightharpoonup M$ the *update function* given by $\mu(m, x) = m'$ if $m \xrightarrow{x} m'$ is the (only) transition from m labelled x. We extend μ to $\mu \colon M \times X^* \rightharpoonup M$ by induction ($\mu(m, \varepsilon) = m$ and $\mu(m, x_1 \dots x_n) = \mu(\mu(m, x_1 \dots x_{n-1}), x_n)$). A *memory structure* (for Eve) for a game $\mathcal{G}$ is a memory skeleton over the set E of edges of $\mathcal{G}$ together with a *next-move function* $\sigma \colon V_{\mathrm{Eve}} \times M \to E$. We say that $(\mathcal{M}, \sigma)$ *implements* a strategy $\mathrm{strat}_v \colon \mathit{Path}_v^{\mathrm{fin}}(\mathcal{G}) \rightharpoonup E$ if for any finite play $\rho \in \mathit{Path}_v^{\mathrm{fin}}(\mathcal{G})$ ending in V_{Eve}, $\mathrm{strat}_v(\rho) = \sigma(\mathrm{Target}(\rho), \mu(m_0, \rho))$. We remark

that a memory structure for G implements at most one strategy from a given vertex. We say that strat_v is a *finite memory strategy* if it can be implemented by a finite memory structure.

Composition of a transition system and an automaton. 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 $\mathbb{W}_1$ to another one using a different acceptance set $\mathbb{W}_2$. 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 2.6, 2.7, and 2.8).

Let $\mathcal{TS} = (G_{\mathcal{TS}}, \text{Acc}_{\mathcal{TS}})$ be a transition system, with $G_{\mathcal{TS}} = (V, E, \text{Source}_{\mathcal{TS}}, \text{Target}_{\mathcal{TS}}, I_{\mathcal{TS}})$ and $\text{Acc}_{\mathcal{TS}} = (\gamma_{\mathcal{TS}}, \Sigma, \mathbb{W}_{\mathcal{TS}})$, and let $\mathcal{A} = (G_{\mathcal{A}}, \text{Acc}_{\mathcal{A}}, (l_\Sigma, \Sigma))$ be a complete automaton over the alphabet Σ, where $G_{\mathcal{A}} = (Q, \Delta, \text{Source}_{\mathcal{A}}, \text{Target}_{\mathcal{A}}, I_{\mathcal{A}})$ and $\text{Acc}_{\mathcal{A}} = (\gamma_{\mathcal{A}}, \Gamma, \mathbb{W}_{\mathcal{A}})$. 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,

$$E^\ltimes = \{(e_1, e_2) \in E \times \Delta \mid \gamma_{\mathcal{TS}}(e_1) = l_\Sigma(e_2)\} \ \cup \ \{e_1 \in E \mid \gamma_{\mathcal{TS}}(e_1) = \varepsilon\} \ \subseteq \ (E \times \Delta) \cup E.$$

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

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 $\mathbb{W}_{\mathcal{A}} \subseteq \Gamma^\omega$ of the automaton as the acceptance set for $\mathcal{TS} \ltimes \mathcal{A}$.

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 $\mathcal{A}$.

We could, of course, 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 ($\gamma(e) = \varepsilon$), $v' \in V_{\text{Eve}}$, and e is the only incoming edge to v' ($\text{In}(v') = \{e\}$). We remark that any game 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.

The following results are well known and constitute the main application of automata composition. They can be seen as corollaries of our results from Section 3.4, which generalise them.

PROPOSITION 2.6 (Folklore). *Let B be an automaton with acceptance set $\mathbb{W}_B$ and let A be an automaton recognising $\mathcal{L}(A) = \mathbb{W}_B$. Then, $\mathcal{L}(B \ltimes A) = \mathcal{L}(B)$. Moreover, if A and B are deterministic (resp. history-deterministic), so is $B \ltimes A$.*

PROPOSITION 2.7 ([44]). *Let G be a game that is suitable for transformations with acceptance set $\mathbb{W}_G$, and let A be a history-deterministic automaton recognising $\mathcal{L}(A) = \mathbb{W}_G$. Then, the winning region of Eve in G is the projection of her winning region in $G \ltimes A$, that is, Eve wins G from an initial vertex v if and only if she wins $G \ltimes A$ from (v, q_0), for q_0 some initial vertex of A.*

Proposition 2.7 fails if the automaton is not HD. In fact, this property characterises history-determinism, which is the reason why HD automata are also called *good-for-games* in the literature. 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 [13, 28].

PROPOSITION 2.8 ([44]). *Let A be an automaton recognising $\mathbb{W}_G \subseteq \Sigma^\omega$ satisfying that for every game G suitable for transformations with acceptance set $\mathbb{W}_G$, Eve wins the game G from an initial vertex v if and only if she wins $G \ltimes A$ from (v, q_0), for q_0 some initial vertex of A. Then, A is history-deterministic.*

2.2 Muller languages, cycles and the parity hierarchy

Languages commonly used as acceptance sets. 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:

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

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

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

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

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

Rabin. A Rabin language is represented by a family $R = \{(G_1, R_1), \ldots, (G_r, R_r)\}$ of *Rabin pairs*, where $G_j, R_j \subseteq \Gamma$. The *Rabin language associated to R* is defined as:

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

If $[\text{Inf}(w) \cap G_j \neq \emptyset$ and $\text{Inf}(w) \cap R_j = \emptyset]$, we say that w is *accepted by the Rabin pair* (G_j, R_j). 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 = $ Rabin$_\Gamma(R)$.

Streett. The *Streett language associated to* a family $S = \{(G_1, R_1), \ldots, (G_r, R_r)\}$ of Rabin pairs is defined as:

$$\text{Streett}_\Gamma(S) = \{w \in \Gamma^\omega \mid [\text{Inf}(w) \cap G_j \neq \emptyset \text{ implies } \text{Inf}(w) \cap R_j \neq \emptyset] \text{ for all indices } j\}.$$

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

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

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

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 $\phi(w) \in \text{parity}_{[d_{\min}, d_{\max}]}$. 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.

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

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

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 = $ Muller$_\Gamma(\mathcal{F})$.

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 acceptance condition (resp. transition system, automaton) is an *X condition* (resp. *X* transition system, *X* automaton), for *X* one of the classes of languages above, if its acceptance set is an *X* language. In the case of parity transition systems, we will always assume 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 discuss further classes of languages in Appendix A (generalised Büchi and coBüchi languages, as well as generalised weak acceptance). We refer to the survey [8] for a more detailed account on different types of acceptance conditions.

REMARK 2.9 (Inclusions between classes). We observe that there are many inclusions between the classes of languages that we have introduced. For example, Büchi languages are exactly $[0, 1]$-parity languages, and parity languages are Rabin languages [71]. In particular, all classes above are special cases of Muller languages. The relations between these classes of languages are outlined in Figure 2.
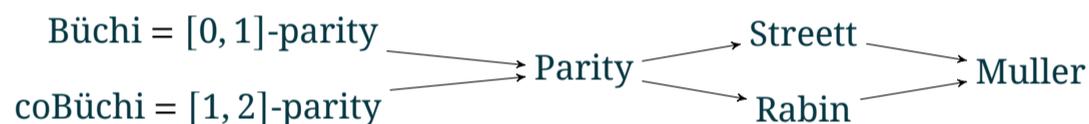
$$\text{Büchi} = [0, 1]\text{-parity} \qquad \text{Streett}$$
$$\text{Parity} \qquad \qquad \text{Muller}$$
$$\text{coBüchi} = [1, 2]\text{-parity} \qquad \text{Rabin}$$

Figure 2. 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 [94].

REMARK 2.10. A language $L \subseteq \Gamma^\omega$ is a Muller language if and only if it satisfies:

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

REMARK 2.11 (Representation of acceptance conditions). In practice, there exists a variety of ways to represent Muller languages and acceptance conditions of automata: using boolean formulas (Emerson-Lei conditions), as a list of accepting subsets of edges, etc. The complexity and practicality of algorithms manipulating automata and games may greatly differ depending on the representation of their acceptance conditions [45, 47]. However, in this work, we are mostly interested in the expressive power of acceptance conditions, and the results we present will not depend on how they are represented.

EXAMPLE 2.12. In Figure 3 we show three different types of automata over the alphabet $\Sigma = \{a, b\}$ recognising the language

$$L = \{w \in \Sigma^\omega \mid w = ub^\omega \text{ or } (w = ua^\omega \text{ and } u \text{ has an even number of 'b's })\}.$$
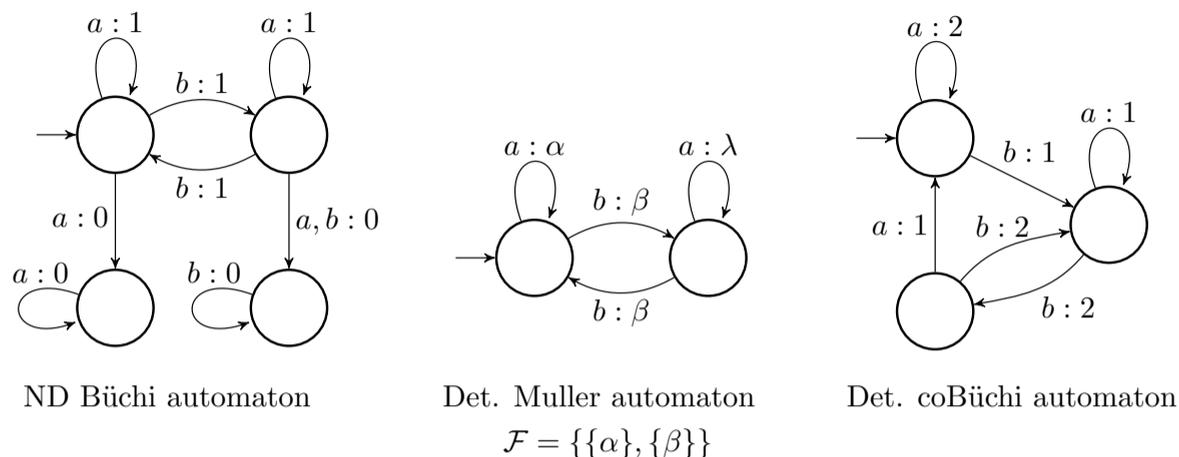
Figure 3. Different types of automata recognising the language
$L = \{w \in \Sigma^\omega \mid w = ub^\omega$ or $(w = ua^\omega$ and u has an even number of 'b's)\}.

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 2.13 ([71, 72]). *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* [17], those that can be described by using *ω-regular expressions* [67], and those that can be recognised by an *ω-semigroup* [77, Chapter 2].

Cycles. Let $\mathcal{TS}$ be a transition system with V and E as set of vertices and edges, respectively. A *cycle* of $\mathcal{TS}$ is a subset $\ell \subseteq E$ such that there is a finite path $v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} v_2 \to \ldots v_r \xrightarrow{e_r} v_0$ with $\ell = \{e_0, e_1, \ldots, e_r\}$. 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 $\mathrm{States}(\ell) = \{v_0, v_1, \ldots v_r\}$. The set of cycles of a transition system $\mathcal{TS}$ is written $\mathit{Cycles}(\mathcal{TS})$. We will consider the set of cycles ordered by inclusion. For a state $v \in V$, we denote $\mathit{Cycles}_v(\mathcal{TS})$ the subset of cycles of $\mathcal{TS}$ containing v. We remark that a vertex v is recurrent if and only if $\mathit{Cycles}_v(\mathcal{TS}) \neq \emptyset$. We note that

$Cycles_v(\mathcal{TS})$ is closed under union; moreover, the union of two cycles $\ell_1, \ell_2 \in Cycles(\mathcal{TS})$ is again a cycle if and only if there is some state v such that both ℓ_1 and ℓ_2 contain v.

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

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

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*, having its origins in the works of Wagner [93], Kaminski [50], and Mostowski [71]. We represented this hierarchy in Figure 4. This hierarchy is strict, that is, for each level of the hierarchy there are languages that do not appear in lower levels [93]. It is known that we can decide in polynomial time the parity index of an ω-regular language represented by a deterministic parity automaton [19], but this problem is NP-complete if the language is given by a deterministic Rabin or Streett automaton [56].

$$
\begin{array}{ccccc}
\vdots & & \vdots & & \vdots \\
& & \mathsf{Weak}_3 & & \\
[0,2] & & & & [1,3] \\
& & \mathsf{Weak}_2 & & \\
[0,1] & & & & [1,2] \\
& & \mathsf{Weak}_1 & & \\
[0,0] & & & & [1,1]
\end{array}
$$

Figure 4. The (history-)deterministic parity hierarchy.

DEFINITION 2.14 (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.

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 Appendix A for more details about languages of parity index Weak_d.

One of our contributions is to show that 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 6.14).

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) [67].

PROPOSITION 2.15 ([12, Theorem 19]). *Let $\mathcal{A}$ be an HD parity automaton recognising a language L, and assume 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 $\mathcal{TS}$ be a transition system using the Muller acceptance condition $(\gamma, \Gamma, \mathsf{Muller}_\Gamma(\mathcal{F}))$. A *d-flower* over a state v of $\mathcal{TS}$ is a set of d cycles $\ell_1, \ell_2, \ldots, \ell_d \in \mathit{Cycles}_v(\mathcal{TS})$ such that $\ell_i \supsetneq \ell_{i+1}$ and $\gamma(\ell_i) \in \mathcal{F} \iff \gamma(\ell_{i+1}) \notin \mathcal{F}$. We say that it is a *positive flower* if $\gamma(\ell_1) \in \mathcal{F}$ and that it is *negative* otherwise.

LEMMA 2.16 (Flower Lemma, [76, 93]). *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.

2.3   Trees

We introduce some technical notations that will be used to define automata based on the Zielonka tree (Sections 4.2 and 4.3) and the transformations based on the ACD (Sections 5.2 and 5.3).

A *tree* $T = (N, \leq)$ is a non-empty finite set of *nodes* N equipped with an order relation $\leq$ called the *ancestor relation* (we say that x is an *ancestor* of y, or that y is below x if $x \leq y$), such that (1) there is a minimal node for $\leq$, called *the root*, and (2) the ancestors of an element are totally ordered by $\leq$. The converse relation $\geq$ 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 $T = (N, \leq)$ is a tree $T' = (N', \leq')$ such that $N' \subseteq N$, $\leq'$ is the restriction of $\leq$ 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 $\leq$.

An *ordered tree* is a tree $T = (N, \leq)$ 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 $n \leq n'$, 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 $n_1 \leq n$ and $n_2 \leq n'$. 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'*.

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 5. 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 5.

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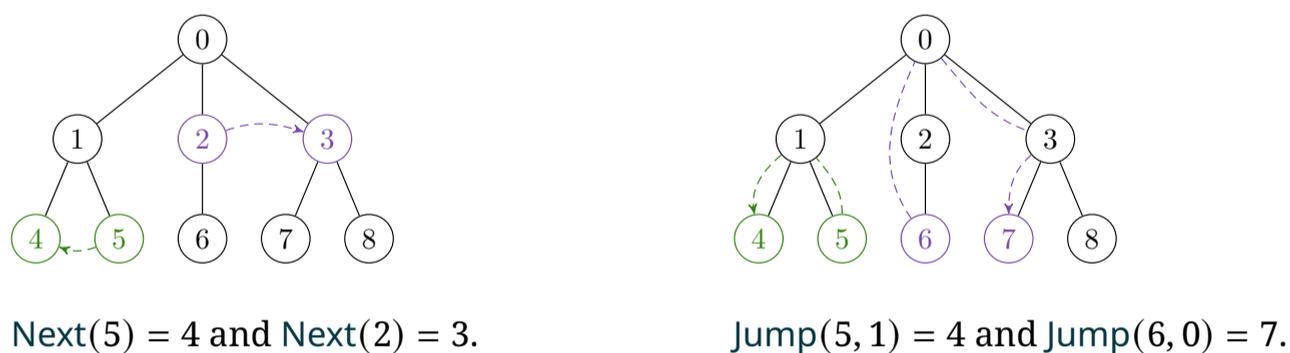
Figure 5. Illustration of the functions Next and Jump.

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 4.5 and 5.5 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 $T' = (N', \leq')$ be a subtree of $T = (N, \leq)$. Let $n' \in N'$ and $n \in N$ such that n' is a (non-strict) ancestor of n ($n' \leq n$). 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 $\mathsf{Jump}_{T'}(n, n') = l_{\mathrm{dest}} \in \mathsf{Leaves}(T')$ 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:

— $n_1 \leq n$,
— $n_2 = \mathsf{Next}_{T'}(n_1)$ (in particular, $n_2 \in N'$),
— $l_{\mathrm{dest}} \geq n_2$ 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'.

An *A-labelled (ordered) tree* is an (ordered) tree T together with a labelling function $\nu \colon N \to A$. A set of trees is called a *forest*.

3.   Morphisms as witnesses of transformations

As mentioned in the introduction, all existing methods transforming a Muller into a parity automaton 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.

We note that almost identical notions of morphisms were considered by Sakarovitch [84, Section 2] and Sakarovitch and de Souza [85, 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

---

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   We thank Géraud Sénizergues for pointing us to the works of Sakarovitch and De Souza.

history-deterministic automata: an automaton is history-deterministic if it is the homomorphic image of a (possibly infinite) deterministic automaton for the same language [27, Definition 13].

In all of this section, $\mathcal{TS} = (G, \mathrm{Acc})$ and $\mathcal{TS}' = (G', \mathrm{Acc}')$ will stand for transition systems with underlying graphs $G = (V, E, \mathrm{Source}, \mathrm{Target}, I)$ and $G' = (V', E', \mathrm{Source}', \mathrm{Target}', I')$, and acceptance conditions $\mathrm{Acc} = (\gamma, \Gamma, \mathbb{W})$ and $\mathrm{Acc}' = (\gamma', \Gamma', \mathbb{W}')$.

3.1 Morphisms of transition systems

DEFINITION 3.1. 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:
— $\mathrm{Source}'(\varphi_E(e)) = \varphi_V(\mathrm{Source}(e))$ for every $e \in E$,
— $\mathrm{Target}'(\varphi_E(e)) = \varphi_V(\mathrm{Target}(e))$ 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 $v_0 \in I$.

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 3.2. 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 $\rho \in \mathcal{R}un(\mathcal{TS})$, $\gamma(\rho) \in \mathbb{W} \iff \gamma'(\varphi_{\mathcal{R}uns}(\rho)) \in \mathbb{W}'$.

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*.

3.2   Local properties of morphisms

DEFINITION 3.3. A morphism of pointed graphs $\varphi : G \to G'$ is called:

— *Locally surjective* if it verifies:
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 verifies:
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 5.4.3). This property will allow us to show that a locally bijective morphism witnesses the semantic equivalence of $\mathcal{TS}$ and $\mathcal{TS}'$ (see Section 3.4).

We note that the notion of locally bijective morphisms of transition systems almost coincide with the usual concept of bisimulation. 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 $\gamma(e)$ to coincide with the colour $\gamma'(\varphi(e))$. This allows us to compare transition systems using different types of acceptance conditions.

REMARK 3.4. 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, the weak morphisms under consideration will be locally surjective. The next lemma ensures that we can assume that they are surjective without loss of generality.

LEMMA 3.5. *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 3.6. In Figure 6 we provide an example of a locally bijective morphism between the two rightmost transition systems from Figure 3 (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.
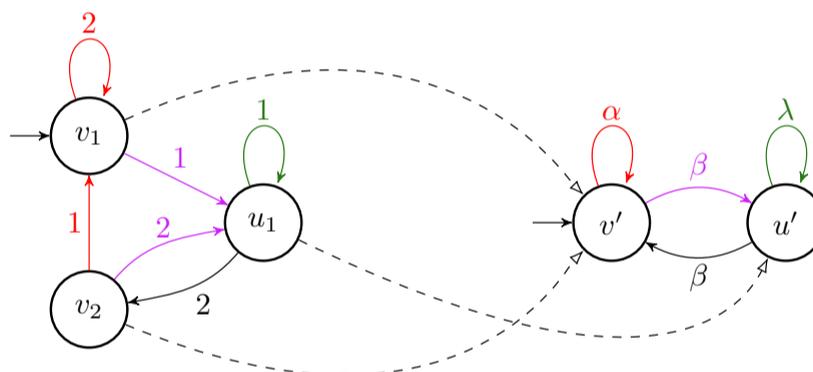

Figure 6. A locally bijective morphism from a parity TS to a Muller TS with acceptance set given by $\mathcal{F} = \{\{\alpha\}, \{\beta\}\}$. We use dashed arrows to represent the images of vertices, and colours to represent the image of edges (that can be inferred from φ_V).

◆

3.3 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 ot $\mathcal{TS}$, 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 2.7).

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 3.4).
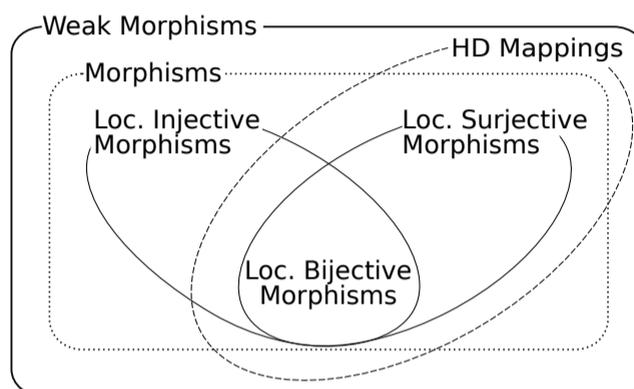

Figure 7. Different types of morphisms and the relations between them. The fact that locally surjective morphisms are HD mappings is given by Lemma 3.12. Note that HD mappings are also locally surjective weak morphisms (Remark 3.7).

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 in a pair of functions $r_{\mathsf{Init}} \colon I' \to I$ and $r \colon E^* \times E' \to E$ such that:

1. $\varphi(r_{\mathsf{Init}}(v_0')) = v_0'$ for all $v_0' \in I'$,
2. $\varphi(r(\rho, e')) = e'$, for all $\rho \in E^*$ and $e' \in E'$,
3. if $e_0' \in \mathsf{Out}(I')$, $\mathsf{Source}(r(\varepsilon, e_0')) = r_{\mathsf{Init}}(\mathsf{Source}(e_0'))$, and
4. if ρ is a finite run in $\mathcal{TS}$ ending in v and $e' \in \mathsf{Out}(\varphi(v))$, then $r(\rho, e') \in \mathsf{Out}(v)$.

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 $r_{\mathcal{R}uns}(\rho') = e_0 e_1 e_2 \cdots \in \mathcal{R}un^\infty(\mathcal{TS})$ defined by $e_i = r(e_0 \ldots e_{i-1}, e_i')$, 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 $r_{\mathcal{R}uns}(\rho')$ is accepting in $\mathcal{TS}$. Note that we do not impose $r_{\mathcal{R}uns}(\rho')$ to be rejecting if ρ' is.

REMARK 3.7. Provided that all states of $\mathcal{TS}'$ are accessible, a resolver simulating φ can only exist if φ is a locally surjective weak morphism.

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 $e_1 e_2 \ldots$ is an accepting run in $\mathcal{TS}$ from v_0 or $e'_1 e'_2 \ldots$ 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 3.8. 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 $\gamma(\rho) \in \mathbb{W} \implies \gamma'(\varphi_{\mathcal{R}uns}(\rho)) \in \mathbb{W}'$, 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 3.9. *Let $\varphi \colon \mathcal{TS} \to \mathcal{TS}'$ be an HD mapping and let (r_{Init}, r) be a sound resolver simulating it. The following holds:*

— *$\varphi_{\mathcal{R}uns} \circ r_{\mathcal{R}uns} = \mathrm{Id}_{\mathcal{R}un^\infty(\mathcal{TS}')}$.*

— *$r_{\mathcal{R}uns}$ 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 $r_{\mathcal{R}uns}(\rho')$ is accepting in $\mathcal{TS}$.*

PROOF. The first item follows from the fact that $\varphi(r(\rho, e')) = e'$ 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 $r_{\mathcal{R}uns}(\rho')$. For the other direction, if $r_{\mathcal{R}uns}(\rho')$ is accepting, then $\varphi_{\mathcal{R}uns}(r_{\mathcal{R}uns}(\rho')) = \rho'$ has to be accepting, as an HD mapping preserves accepting runs. ∎

EXAMPLE 3.10. In Figure 8 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 $\varphi(f_1) = \varphi(f_2) = f'$.
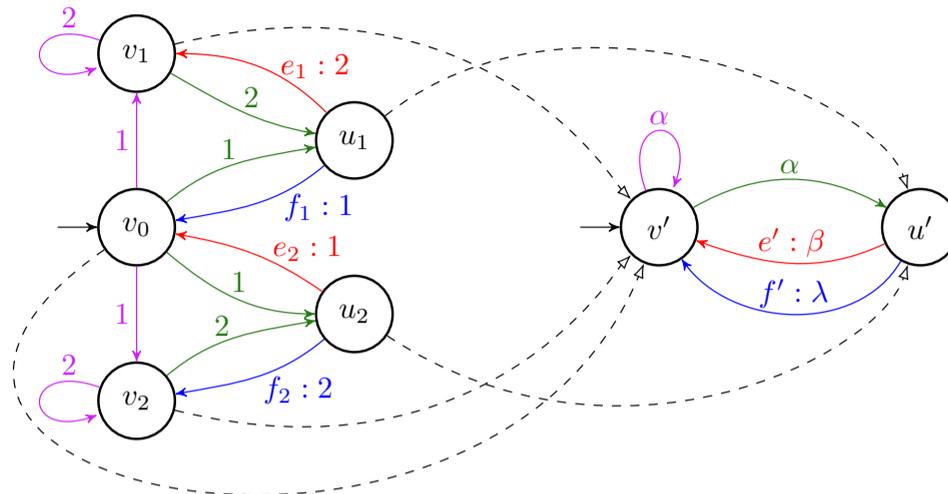
Figure 8. A history-deterministic mapping from a parity TS to a Muller TS with acceptance set given by $\mathcal{F} = \{\{\alpha\}, \{\alpha, \beta\}, \{\alpha, \lambda\}\}$. We use dashed arrows to represent the images of vertices, and colours to represent the image of edges.

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 symmetrically. 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 $\mathcal{TS}'$). ◆

History-deterministic-for-games mappings. In the case of games, we need to slightly strengthen the definition of HD mappings to guarantee that, if there is a suitable mapping $\varphi \colon \mathcal{G} \to \mathcal{G}'$, then $\mathcal{G}$ and $\mathcal{G}'$ have the same winner. 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.

Let $\mathcal{G}$ and $\mathcal{G}'$ be two games, and $\varphi : \mathcal{G} \to \mathcal{G}'$ be a weak morphism between them admitting a resolver (r_{Init}, 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* (r_{Init}, r) over ρ' if:

1. $\text{Source}(e_0) = r_{\text{Init}}(\text{Source}(e'_0))$,

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 $e_n = r(e_0 \dots e_{n-1}, e'_n)$.

We remark that there exists at least one run consistent with (r_{Init}, r) over ρ', namely $r_{\mathscr{R}uns}(\rho')$. We say that (r_{Init}, r) is *sound for $\mathcal{G}$* if it verifies that for any accepting run $\rho' \in \mathscr{R}un(\mathcal{G}')$, all runs consistent with (r_{Init}, 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 $e_n = v_{n-1} \to v_n \in \text{Out}(v_{n-1})$ such that $\varphi(e_n) = e'_n$. If v_n is controlled by Eve, it is Duplicator who chooses one such e_n.

— Duplicator wins if either $e_1 e_2 \dots$ is an accepting run in $\mathcal{G}$ from v_0 or $e'_1 e'_2 \dots$ is not an accepting run in $\mathcal{G}'$ from v'_0. Spoiler wins otherwise.

DEFINITION 3.11. 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 $l_{\text{Players}} : V \to \{\text{Eve}, \text{Adam}\}$), and that φ preserves those vertex-labellings).

In the next lemma, we prove that HD and HD-for-games 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 3.12. *If $\varphi : \mathcal{TS} \to \mathcal{TS}'$ is a locally surjective morphism, it is also an HD mapping. If $\mathcal{TS}$ and $\mathcal{TS}'$ are games, φ is moreover HD-for-games.*

PROOF. We need to define a sound resolver simulating φ. Let $r_{\text{Init}} : I' \to I$ be any function choosing initial vertices satisfying that $\varphi \circ r_{\text{Init}} = Id_{I'}$ (which exists by local surjectivity of φ). For each $v \in V$ and edge $e' \in \text{Out}(\varphi(v))$ we choose one edge $f(v, e') \in \text{Out}(v)$ 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 $r : E^* \times E' \to E$ recursively. For the base case, if $e'_0 \in \text{Out}(v')$, with $v' \in I'$, we define $r(\varepsilon, e'_0) = f(r_{\text{Init}}(v'), e'_0)$. 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 $e' \notin \text{Out}(\text{Target}(\varphi(\rho)))$, we let $r(\rho, e')$ be any edge in $\varphi^{-1}(e')$. If not, let $v = \text{Target}(\rho)$ and define $r(\rho, e')$ to be the edge $f(v, e')$.

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

Restrictions and extensions of initial sets. The following simple lemma states that reducing the number of initial vertices preserves the history-determinism of mappings.

LEMMA 3.13. *Let $\mathcal{TS}$ and $\mathcal{TS}'$ be two TS such that there is an HD (resp. HD-for-games) mapping $\varphi : \mathcal{TS} \to \mathcal{TS}'$. For any non-empty subset $\tilde{I} \subseteq I'$, φ is also an HD (resp. HD-for-games) mapping between the transition systems $\mathcal{TS}_{r_{\text{Init}}(\tilde{I})}$ and $\mathcal{TS}'_{\tilde{I}}$; that is, the transitions systems obtained by setting $r_{\text{Init}}(\tilde{I})$ 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. The proof can be found in Appendix C.

LEMMA 3.14. *Let $\mathcal{TS}$ and $\mathcal{TS}'$ be two TS such that all their states are accessible, and let $\varphi : \mathcal{TS} \to \mathcal{TS}'$ be an HD (resp. HD-for-games) mapping between them. If $\mathbb{W}$ and $\mathbb{W}'$ are prefix-independent, the mapping φ is also HD (resp. HD-for-games) 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.*

3.4   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 compositions by deterministic and history-deterministic automata, respectively (Proposition 3.15). Then, we prove that these mappings witness the semantic equivalence of the transition systems under consideration. That is, (1) if $\varphi : \mathcal{A} \to \mathcal{A}'$ is an HD mapping of automata, then $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$, and if φ is locally bijective, $\mathcal{A}$ is deterministic (or unambiguous) if and only if $\mathcal{A}'$ is (Proposition 3.16);[5] and (2) if $\varphi : \mathcal{G} \to \mathcal{G}'$ is an HD-for-games mapping, $\mathcal{G}$ and $\mathcal{G}'$ have the same winner (Proposition 3.18 and Corollary 3.19).

Morphisms generalise composition by an automaton.

---

5   The results in this section do not directly imply that if $\mathcal{A}$ is an automaton recognising the acceptance set of another automaton $\mathcal{B}$, then $\mathcal{B} \ltimes \mathcal{A}$ recognises the same language as $\mathcal{B}$, if $\mathcal{A}$ is not history-deterministic (Proposition 2.6). In that case, the equality $\mathcal{L}(\mathcal{B} \ltimes \mathcal{A}) = \mathcal{L}(\mathcal{B})$ follows from the idea that runs in $\mathcal{B}$ can be simulated in $\mathcal{B} \ltimes \mathcal{A}$ "guided by the non-deterministic choices of $\mathcal{A}$", which we do not formalise in this work.

PROPOSITION 3.15. *Let $\mathcal{A}$ be a complete automaton accepting the language $\mathcal{L}(\mathcal{A}) = \mathbb{W} \subseteq \Sigma^\omega$, and let $\mathcal{TS}$ be a (labelled) TS with acceptance set $\mathbb{W}$. Then, there exists a locally surjective weak morphism of (labelled) TS $\varphi \colon \mathcal{TS} \ltimes \mathcal{A} \to \mathcal{TS}$ 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.*
3. *If $\mathcal{A}$ is HD and $\mathcal{TS}$ is a game suitable for transformations, then φ can be chosen to be an HD-for-games 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 $\mathbb{W}_{\mathcal{A}} \subseteq \Gamma^\omega$ 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 $\rho = (v_0, q_0) \xrightarrow{c_0} (v_1, q_1) \xrightarrow{c_1} \ldots$ in $\mathcal{TS} \ltimes \mathcal{A}$, we can consider its projection over $\mathcal{TS}$, $\varphi_{\mathcal{R}uns}(\rho) = v_0 \xrightarrow{a_0} v_1 \xrightarrow{a_1} \ldots$. We note that there must exist a unique run in $\mathcal{A}$ of the form

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

(Formally, some 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, $c_0 c_1 c_2 \cdots \in \mathbb{W}_{\mathcal{A}}$, and therefore $\varphi_{\mathcal{A}}(\rho)$ is an accepting run in $\mathcal{A}$ over $a_0 a_1 a_2 \ldots$, so we conclude that $a_0 a_1 a_2 \cdots \in \mathbb{W}$ 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 $E^\ltimes_{\text{col}} \subseteq E \times \Delta$ and $E^\ltimes_\varepsilon \subseteq E$ such that $E^\ltimes = E^\ltimes_{\text{col}} \cup E^\ltimes_\varepsilon$. We remark that $\varphi|_{E^\ltimes_\varepsilon}$ is the identity function (so injective) and that $\varphi(E^\ltimes_{\text{col}}) \cap \varphi(E^\ltimes_\varepsilon) = \emptyset$ because $\varphi(E^\ltimes_{\text{col}})$ are exactly coloured transitions of $\mathcal{TS}$. Finally, let (e_1, e_2) and (e'_1, e'_2) in $\text{Out}(v, q) \cap E^\ltimes_{\text{col}}$. 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, $c_0 c_1 c_2 \cdots \notin \mathbb{W}_{\mathcal{A}}$, and therefore $\varphi_{\mathcal{A}}(\rho)$ is a rejecting run over $a_0 a_1 a_2 \ldots$. Since $\mathcal{A}$ is deterministic, this is the only run over $a_0 a_1 a_2 \ldots$, so we conclude that it does not belong to $\mathbb{W}$. We conclude that $\varphi_{\mathcal{R}uns}(\rho)$ is a rejecting run in $\mathcal{TS}$.

2. Let $(r_0, r_{\mathcal{A}})$ be a resolver for $\mathcal{A}$. We define a resolver $(r_{\text{Init}}, r_\varphi)$ simulating φ. First, we let $r_{\text{Init}}(v_0) = (v_0, r_0)$ for all $v_0 \in I_{\mathcal{TS}}$. We define $r_\varphi \colon E^{\ltimes *} \times E \to E^\ltimes$ by induction on the length of the runs. Let $e_0 = v_0 \xrightarrow{a} v_1 \in \text{Out}(v_0)$ be an edge in $\mathcal{TS}$. If e_0 is uncoloured $(a = \varepsilon)$, we let $r_\varphi(\varepsilon, e_0) = e_0 = (v_0, r_0) \xrightarrow{\varepsilon} (v_1, r_0)$. If not, we let $r_\varphi(\varepsilon, e_0) = (e_0, e_a)$, where $e_a = r(\varepsilon, a)$. Assume that r_φ has been defined for sequences of edges of $\mathcal{TS} \ltimes \mathcal{A}$ of length $< n$ and let $\rho = e_0 e_1 \ldots e_{n-1} \in E^{\ltimes *}$ 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 $r_\varphi(\rho, e_{\mathcal{TS}})$ 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 $r_\varphi(\rho, e_{\mathcal{TS}}) = e_{\mathcal{TS}}$. As noted before, ρ induces a run $\varphi_{\mathcal{A}}(\rho) = q_0 \xrightarrow{a_0 : c_0} q_1 \xrightarrow{a_1 : c_1} \ldots \to q_n$ in $\mathcal{A}$. We let $e_{\mathcal{A}} = r_{\mathcal{A}}(\varphi_{\mathcal{A}}(\rho), a_n)$ be the transition chosen by the resolver of $\mathcal{A}$ after this run, and we define $r_\varphi(\rho, e_{\mathcal{TS}}) = (e_{\mathcal{TS}}, e_{\mathcal{A}})$.

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

3. We prove that, if $\mathcal{TS}$ is a game suitable for transformations, the resolver $(r_{\text{Init}}, r_\varphi)$ defined in the previous item is sound for $\mathcal{TS}$. We claim that if ρ is a run in $\mathcal{TS}$, the only run consistent with $(r_{\text{Init}}, r_\varphi)$ over ρ is $r_{\varphi, \mathcal{R}uns}(\rho)$. This follows from the fact that if (v, q) is a vertex in $\mathcal{TS} \ltimes \mathcal{A}$ controlled by Adam and $e \in \text{Out}(v)$, then there is a unique $e' \in \text{Out}(v, q)$ such that $\varphi(e') = e$. This is indeed the case: as $\mathcal{TS}$ is suitable for transformations, if v is an Adam's vertex, every $e \in \text{Out}(v)$ 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 $\mathcal{TS}$ and $\rho^\ltimes$ is a run consistent with $(r_{\text{Init}}, r_\varphi)$ over ρ, then $\rho^\ltimes = r_{\varphi, \mathcal{R}uns}(\rho)$, which is accepting by soundness of the resolver $(r_{\text{Init}}, r_\varphi)$. ■

Morphisms witness equivalence of automata.

PROPOSITION 3.16. *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 (r_{Init}, r), if ρ is an accepting run over $w \in \Sigma^\omega$ in $\mathcal{A}'$, then $r_{\mathcal{R}uns}(\rho)$ is an accepting run over w in $\mathcal{A}$, so $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$.

Let $(r_{\text{Init}}, r_\varphi)$ 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: $r_0' = \varphi(r_0)$ and for $\rho' \in \mathcal{R}un^{\text{fin}}(\mathcal{A}')$ and $a \in \Sigma$, $r'(\rho', a) = \varphi(r(r_{\varphi, \mathcal{R}uns}(\rho'), a))$. 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 $r_0 = r_{\mathrm{Init}}(r_0')$, and for $\rho \in \mathcal{R}un^{\mathrm{fin}}(\mathcal{A})$ and $a \in \Sigma$, $r(\rho, a) = r_\varphi((\varphi_{\mathcal{R}uns}(\rho), r(\varphi_{\mathcal{R}uns}(\rho), a)))$. 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 [29, 20] 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 3.4, 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.

LEMMA 3.17. *Let G, G' be two games, such that there is a weak morphism of games $\varphi \colon G \to G'$ that is locally surjective and preserves accepting runs. If Eve wins the game G from an initial vertex v, then she wins G' from $\varphi(v)$.*

PROOF. Let $v' = \varphi(v)$, and let $\mathrm{strat}_v \colon \mathcal{P}ath_v^{\mathrm{fin}}(G) \to E$ be a strategy from v for Eve in G. Intuitively, we will define a strategy in G' as follows: for each finite run ρ' from v' in G', we pick a preimage $\rho \in \varphi^{-1}(\rho')$ in G, look at the decision made by strat_v at the end of ρ and transfer it back to G' via φ. In order to define a correct strategy, 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 $\mathrm{choice}_{st} \colon \mathcal{P}ath_{v'}^{\mathrm{fin}}(G') \to \mathcal{P}ath_v^{\mathrm{fin}}(G)$ satisfying that for any $\rho' = e_0' e_1' \ldots e_{n-1}' e_n' \in \mathcal{P}ath_{v'}^{\mathrm{fin}}(G')$:

— The run $\mathrm{choice}_{st}(\rho')$ has length $n + 1$.
— $\varphi_{\mathcal{R}uns}(\mathrm{choice}_{st}(\rho')) = \rho'$.
— Monotonicity: if $\tilde{\rho}' \sqsubseteq \rho'$ then $\mathrm{choice}_{st}(\tilde{\rho}') \sqsubseteq \mathrm{choice}_{st}(\rho')$.
— If there exists $e_n \in \varphi^{-1}(e_n')$ such that $\mathrm{choice}_{st}(e_0' e_1' \ldots e_{n-1}') e_n$ is consistent with strat_v, then $\mathrm{choice}_{st}(\rho')$ is consistent with strat_v.

Assume for now that such a function exists, and define a strategy in G' as

$$\mathrm{strat}_{v'}'(\rho') = \varphi(\mathrm{strat}_v(\mathrm{choice}_{st}(\rho'))), \quad \text{for } \rho' \in \mathcal{P}ath_{v'}^{\mathrm{fin}}(G').$$

We prove that $\mathrm{strat}_{v'}'$ is winning. Let $\rho' = e_0' e_1' \cdots \in \mathcal{R}un(G')$ be an infinite play consistent with $\mathrm{strat}_{v'}'$. For each finite prefix $\tilde{\rho}' \sqsubseteq \rho'$, $\mathrm{choice}_{st}(\tilde{\rho}')$ is a finite play in G, and by the monotonicity

assumption, we can define the limit of these runs as:

$$\vec{\rho} = e_0 e_1 e_2 \cdots \in \mathcal{R}un(\mathcal{G}), \quad \text{where } e_0 e_1 \ldots e_n = \text{choice}_{st}(e'_0 e'_1 \ldots e'_n),$$

which is indeed a run in $\mathcal{G}$. We show that $\vec{\rho}$ is consistent with strat_v by induction. Let $\rho_n = e_0 e_1 \ldots e_{n-1}$ 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 $e_n = \text{strat}_v(\rho_n)$. By definition of $\text{strat}'_{v'}$, $e'_n = \varphi(\text{strat}_v(\rho_n)) = \varphi(e_n)$, and as v_n is controlled by Eve, $\text{strat}_v(\rho_n)$ 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 $\text{strat}_v(\rho_n)$, 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 $\text{choice}_{st} \colon \mathcal{P}ath_{v'}^{\text{fin}}(\mathcal{G}') \to \mathcal{P}ath_v^{\text{fin}}(\mathcal{G})$ by induction on the length of the runs. Assume that choice_{st} has been defined for runs of length $\leq n$, and let $e'_0 e'_1 \ldots e'_n$ be a run of length $n+1$, with $\text{choice}_{st}(e'_0 e'_1 \ldots e'_{n-1}) = e_0 e_1 \ldots e_{n-1}$. If $e_0 e_1 \ldots e_{n-1}$ is not consistent with strat_v, it ends in a vertex v_n controlled by Adam, or $\text{strat}_v(e_0 e_1 \ldots e_{n-1}) \notin \varphi^{-1}(e_n)$, we let $e_n \in \varphi^{-1}(e_n) \cap \text{Out}(v_n)$ be any edge (one such edge exists by local surjectivity). On the contrary, we let $e_n = \text{strat}_v(e_0 e_1 \ldots e_{n-1})$. We define $\text{choice}_{st}(e'_0 e'_1 \ldots e'_{n-1} e'_n) = e_0 e_1 \ldots e_{n-1} e_n$. By construction, the obtained function fulfils the 4 requirements. ∎

PROPOSITION 3.18. *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 projection of her winning region in $\mathcal{G}$: $\mathcal{W}_{\text{Eve}}(\mathcal{G}') = \varphi(\mathcal{W}_{\text{Eve}}(\mathcal{G}))$.*

PROOF. If Eve wins $\mathcal{G}$ from an initial vertex v, Lemma 3.17 guarantees that she wins $\mathcal{G}'$ from $\varphi(v)$.

Assume now that Eve wins $\mathcal{G}'$ from an initial vertex v' with a strategy $\text{strat}'_{v'} \colon \mathcal{P}ath_{v'}^{\text{fin}}(\mathcal{G}') \to E'$. We need to show that she wins $\mathcal{G}$ from some initial vertex in $\varphi^{-1}(v')$. Let (r_{Init}, r) be a resolver simulating φ sound for $\mathcal{G}$ and let $v = r_{\text{Init}}(v')$. We define

$$\text{strat}_v(\rho) = r(\rho, \text{strat}'_{v'}(\varphi_{\mathcal{R}uns}(\rho))), \quad \text{for } \rho \in \mathcal{P}ath_v^{\text{fin}}(\mathcal{G}).$$

That is, strat_v is a strategy in $\mathcal{G}$ from v that, given a finite run ρ, simulates ρ in $\mathcal{G}'$, looks at the move done by the strategy $\text{strat}'_{v'}$ 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 $\mathcal{G}$. Let $\rho = e_0 e_1 \cdots \in \mathcal{P}ath_v(\mathcal{G})$ be a play consistent with strat_v. We claim that $\varphi(\rho)$ is consistent with $\text{strat}'_{v'}$ and that ρ is consistent with (r_{Init}, r) over $\varphi(\rho)$. This implies the desired result; consistency with $\text{strat}'_{v'}$ implies that $\varphi(\rho)$ is accepting, and since (r_{Init}, r) is sound for $\mathcal{G}$, ρ would be accepting in $\mathcal{G}$.

We prove that $\varphi(\rho)$ is consistent with $\text{strat}'_{v'}$. Let $e'_0 e'_1 \ldots e'_{n-1}$ 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 $e_n = r(e_0 \ldots e_{n-1}, \text{strat}'_{v'}(e'_0 \ldots e'_{n-1}))$, and by definition of a resolver (item 2), we obtain that $e'_n = \varphi(e_n) = \text{strat}'_{v'}(e'_0 \ldots e'_{n-1}))$, as we wanted.

The fact that ρ is consistent with (r_{Init}, r) over $\varphi(\rho)$ follows directly from the definition of strat_v. ∎

The next corollary follows from the previous proposition and Lemma 3.14.

COROLLARY 3.19. *Let $\mathcal{G}, \mathcal{G}'$ be two games whose states are accessible and such that their acceptance sets $\mathbb{W}_\mathcal{G}$ and $\mathbb{W}_{\mathcal{G}'}$ 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 $\mathcal{G}$: $\mathcal{W}_{\text{Eve}}(\mathcal{G}'_{V'}) = \varphi(\mathcal{W}_{\text{Eve}}(\mathcal{G}_V))$.*

4. 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 [94]. 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 4.2, we describe the construction of a minimal deterministic parity automaton $\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\text{parity}}$ for a given Muller language $\text{Muller}(\mathcal{F})$. Theorem 4.15, the main contribution of this section, states the minimality of $\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\text{parity}}$ both amongst deterministic and HD parity automata. Theorem 4.14 states the optimality on the number of colours of the acceptance condition of $\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\text{parity}}$, and implies that we can determine the parity index of a Muller language from its Zielonka tree. We will use the optimality of automaton $\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\text{parity}}$ to provide a polynomial-time algorithm minimising DPAs recognising Muller languages in Section 6.3.

In Section 4.3, we describe the construction of a minimal history-deterministic Rabin automaton $\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\text{Rabin}}$ for a Muller language $\text{Muller}(\mathcal{F})$. Its minimality amongst HD automata is shown in Theorem 4.51, by using the characterisation of the memory requirements of a Muller language in terms of its Zielonka tree [34].

On the other hand, it has been shown 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 [21]. Therefore, unless P = NP, there are Muller languages for which minimal deterministic Rabin automata are strictly larger than minimal HD Rabin automata. Some explicit such languages were shown in [24, Section 4]. A summary of the minimal automata recognising Muller languages appears in Table 1.

4.1 The Zielonka tree

DEFINITION 4.1 ([94]). 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),[6] denoted $\mathcal{Z}_\mathcal{F} = (N, \leq, \nu : N \to 2_+^\Sigma)$ 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.

Type of automata	Deterministic	History-deterministic
Parity	$\mathcal{A}^{\text{parity}}_{\mathcal{Z}_{\mathcal{F}}}$	$\mathcal{A}^{\text{parity}}_{\mathcal{Z}_{\mathcal{F}}}$
Rabin	No characteri-sation	$\mathcal{A}^{\text{Rabin}}_{\mathcal{Z}_{\mathcal{F}}}$

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

— 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.

REMARK 4.2. We note that 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 $\mathcal{F}$.

For a family of subsets $\mathcal{F} \subseteq 2^{\Sigma}$ and $\Sigma' \subseteq \Sigma$, we write $\mathcal{F}|_{\Sigma'} = \mathcal{F} \cap 2^{\Sigma'}$.

REMARK 4.3. We remark that 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 $\mathcal{F}|_{v(n)}$ over the alphabet $v(n)$, that is, for the restriction of $\mathcal{F}$ to the subsets included in the label of n.

REMARK 4.4. Let n be a node of $\mathcal{Z}_{\mathcal{F}}$ and let n_1 be a child of it. If $v(n_1) \subsetneq X \subseteq v(n)$, then $v(n_1) \in \mathcal{F} \iff X \notin \mathcal{F} \iff v(n) \notin \mathcal{F}$. In particular, if n_1, n_2 are two different children of n, then $v(n_1) \in \mathcal{F} \iff v(n_2) \in \mathcal{F} \iff v(n_1) \cup v(n_2) \notin \mathcal{F}$.

We equip Zielonka trees with an order to navigate in them. That is, we equip each set Children$_{\mathcal{Z}_{\mathcal{F}}}(n)$ with a total order, making $\mathcal{Z}_{\mathcal{F}}$ an ordered tree. The precise order considered 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 $l \in$ Leaves($\mathcal{Z}_{\mathcal{F}}$) and a letter $a \in \Sigma$ we define Supp(l, a) = n to be the deepest ancestor of l (maximal for $\leq$) such that $a \in v(n)$.

EXAMPLE 4.5. We will use the Muller language associated to the following family of subsets as a running example throughout the paper. Let $\Sigma = \{a, b, c\}$ and let $\mathcal{F}$ be:

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

---

6   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 time the alphabet will be clear from the context.

In Figure 9 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 the 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 $\{a, c\}$).   ◆
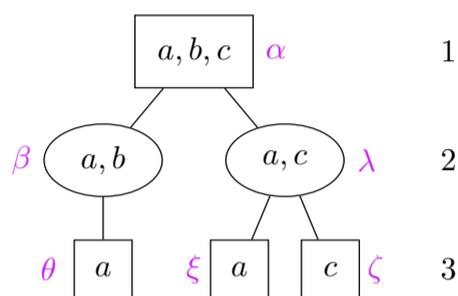

Figure 9. Zielonka tree $\mathcal{Z}_\mathcal{F}$ for $\mathcal{F} = \{\{a, b\}, \{a, c\}, \{b\}\}$.

4.2   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.

4.2.1   The Zielonka-tree-parity-automaton

We associate a non-negative integer to each level of a Zielonka tree $\mathcal{Z}_\mathcal{F} = (N, \leq, \nu)$. 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 4.6. 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 4.7. The Muller language from Example 4.5 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 9. 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 4.8 (Zielonka-tree-parity-automaton). Given a family of non-empty subsets $\mathcal{F} \subseteq 2_+^\Sigma$, we define the *ZT-parity-automaton* $\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\text{parity}} = (Q, \Sigma, q_0, [\min_\mathcal{F}, \max_\mathcal{F}], \delta, \text{parity})$ as the deterministic parity automaton given by:

— $Q = \text{Leaves}(\mathcal{Z}_\mathcal{F})$,

— q_0 is the leftmost leaf of $\mathcal{Z}_\mathcal{F}$,[7]

— The transition reading $a \in \Sigma$ from $q \in Q$ goes to $\text{Jump}(q, \text{Supp}(q, a))$ and produces $p_\mathcal{Z}(\text{Supp}(q, a))$ as output, that is,

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

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 on the right of the branch that we took before (in a cyclic way), and we move to the leftmost leaf below it. The colour produced as output is $p_\mathcal{Z}(n)$, determined by the depth of n.

EXAMPLE 4.9. In Figure 10 we show the ZT-parity-automaton $\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\text{parity}}$ of the family of subsets from Example 4.5.   ◆

Figure 10. ZT-parity-automaton recognising the Muller language associated to $\mathcal{F} = \{\{a, b\}, \{a, c\}, \{b\}\}$.

Correctness of the Zielonka-tree-parity-automaton.

PROPOSITION 4.10 (Correctness). *Let $\mathcal{F} \subseteq 2_+^\Sigma$ be a family of non-empty subsets. Then,*

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

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

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

---

7   Any state can be chosen as initial state (see Lemma 2.5).

LEMMA 4.11. *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 $a \in v(n)$, and in this case, $\mathsf{Jump}(q, \mathsf{Supp}(q, a))$ is a descendant of n too.*

PROOF OF PROPOSITION 4.10. Let $w = w_0 w_1 w_2 \cdots \in \Sigma^\omega$ be an infinite word. For $i > 0$, let q_i be the leaf of $\mathcal{Z}_{\mathcal{F}}$ reached after the (only) run over $w_0 w_1 \ldots w_{i-1}$ 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 colour of each transition, and let $c_i = p_{\mathcal{Z}}(n_i) = \gamma(q_i, w_i) \in [\min_{\mathcal{F}}, \max_{\mathcal{F}}]$ be that output colour (the output of the run over w being therefore $c_0 c_1 c_2 \cdots \in \mathbb{N}^\omega$). Let q_∞ be a node appearing infinitely often in the sequence $q_0 q_1 q_2 \ldots$, and let n_w be the deepest ancestor of q_∞ such that $\mathsf{Inf}(w) \subseteq v(n_w)$.

CLAIM 4.12. *There is $K \in \mathbb{N}$ such that for all $i \geq K$, $q_i \geq n_w$ and $\mathsf{Supp}(q_i, w_i) \geq n_w$. 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 4.11 and induction. ◆

CLAIM 4.13. *Let $n_{w,1}, \ldots, n_{w,s}$ be an enumeration of $\mathsf{Children}_{\mathcal{Z}_{\mathcal{F}}}(n_w)$ from left to right. It holds 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 $\mathsf{Inf}(w) \subseteq v(n_{w,k})$.*

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 $\mathsf{Supp}(q_i, w_i) \geq n_{w,k}$, and by Lemma 4.11, $\mathsf{Inf}(w) \subseteq v(n_{w,k})$. 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 4.12. We show that if $i \geq K$ and $q_i \geq n_{w,k}$, then there is $j > i$ such that $w_j \notin v(n_{w,k})$, $\mathsf{Supp}(q_j, w_j) = n_w$ and $q_{j+1} \geq n_{w,k+1}$ (by an abuse of notation we let $s + 1 = 1$). It suffices to consider the least $j \geq i$ such that $\mathsf{Supp}(q_j, w_j) \not\geq n_{w,k}$ (which exists by the previous remark). Since $\mathsf{Inf}(w) \subseteq v(n_w)$ we have that $\mathsf{Supp}(q_j, w_j) = n_w$, so $w_j \notin v(n_{w,j})$ (by Lemma 4.11) 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 $q_{j+1} \geq n_{w,k+1}$ 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 $w_j \notin v(n_{w,j})$ 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 colours that are produced as output infinitely often is $p_{\mathcal{Z}}(n_w)$. By Remark 4.6, $p_{\mathcal{Z}}(n_w)$ is even if and only if n_w is a round node (if $v(n_w) \in \mathcal{F}$). It remains to show that $\mathsf{Inf}(w) \in \mathcal{F}$ if and only if $v(n_w) \in \mathcal{F}$, which holds by the second item in Claim 4.13 and Remark 4.4. ∎

4.2.2   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 4.15) and number of colours of the acceptance condition (Theorem 4.14). The minimality of the ZT-parity-automaton comes in two versions. A weaker one states its minimality only amongst deterministic automata (Theorem 4.18), and a stronger one states its minimality amongst all history-deterministic automata (Theorem 4.15). 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 paper, 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 4.14 (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 colours to recognise this language.*

THEOREM 4.15 (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 4.16. *For every Muller language L, a minimal deterministic parity automaton recognising L has the same size as 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 2.3 is an HD coBüchi automaton recognising a Muller language that cannot be made deterministic just by removing transitions. We note that the (deterministic) ZT-parity-automaton for this Muller language has only 2 states.

We say that an automaton $\mathcal{A}$ is *determinisable by pruning* if there is a subset $\Delta' \subseteq \Delta$ of its transitions and an initial state q_0 such that the subautomaton induced by Δ' with initial state q_0 is deterministic and recognises $\mathcal{L}(\mathcal{A})$.

PROPOSITION 4.17. *There exists an HD parity automaton recognising a Muller language that is not determinisable by pruning.*

Optimality of the parity condition.

PROOF OF THEOREM 4.14. Let $L = \mathsf{Muller}_\Sigma(\mathcal{F})$. The ZT-parity-automaton of L is a parity automaton recognising L using colours 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 2.16. 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 $n_1 \leq n_2 \leq \ldots \leq n_d$ 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 2.16 allows us to conclude. ∎

Minimality of the ZT-parity-automaton with respect to deterministic automata. Before presenting the proof of Theorem 4.15, we prove a weaker result, namely, that the ZT-parity-automaton is minimal amongst *deterministic* parity automata recognising a Muller language.

THEOREM 4.18 (Minimality of the Zielonka Tree automaton 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 2.4 and Lemma 2.5, we can assume 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.

Let $\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta, \mathbb{W})$ be an automaton, and let $X \subseteq \Sigma$ be a subset of the input alphabet. We say that a subgraph $\mathcal{S}$ of the underlying graph of $\mathcal{A}$ is *X-closed* if for every state q in $\mathcal{S}$ and every letter $a \in X$ there is some transition $q \xrightarrow{a:c} q'$ in $\mathcal{S}$. An *X-final strongly connected component* (*X-FSCC*) of $\mathcal{A}$ 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 remark 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 4.19. *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 one. By completeness of $\mathcal{A}$, one such final SCC has to be an *X-closed subgraph*, so it is an *X-FSCC*. ∎

LEMMA 4.20. *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 $\mathcal{A} = (Q, \Sigma, q_0, \Gamma, \delta, \mathbb{W})$ (where $\mathbb{W}$ 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 $\mathbb{W}$, 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 $\mathsf{Inf}(w) \in \mathcal{F}$. ∎

The next lemma states that, in a parity automaton, the union of two accepting cycles must be accepting, and similarly for rejecting cycles. In Section 6.1, we will see that this property is actually a characterisation of parity transition systems (Proposition 6.11).

LEMMA 4.21. *Let $\mathcal{A}$ be a parity automaton. Let $\ell_1, \ell_2 \in \mathit{Cycles}(\mathcal{A})$ 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 $\gamma \colon \Delta \to \mathbb{N}$ be the colouring function of $\mathcal{A}$. The cycles ℓ_1 and ℓ_2 are accepting if and only if $d_i = \min \gamma(\ell_i)$ is even, for $i = 1, 2$. In this case, $\min \gamma(\ell_1 \cup \ell_2) = \min\{d_1, d_2\}$ 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 4.22. *Let $\mathcal{F} \subseteq 2^\Sigma_+$ be a family of subsets with Zielonka tree $\mathcal{Z}_\mathcal{F} = (N, \leq, \nu)$, 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 assume 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 $l_\Sigma \colon \Delta \to \Sigma$ 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 $l_\Sigma(\ell_i) = \nu(n_i)$. 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 $l_\Sigma(\ell_1 \cup \ell_2) = \nu(n_1) \cup \nu(n_2)$. By Lemma 4.21, 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 4.4). ∎

PROOF OF THEOREM 4.18. 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 $n_1, n_2, \ldots, n_k$ be an enumeration of the children of n_0 in $\mathcal{Z}_\mathcal{F}$. By Lemma 4.19, for each $i \in \{1, \ldots, k\}$, $\mathcal{A}$ contains some accessible $\nu(n_i)$-FSCC $\mathcal{S}_i$, and by Lemma 4.22 these must be pairwise disjoint. By Lemma 4.20, 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 $\mathcal{F}|_{v(n_i)}$. By induction hypothesis, it must be the case that $|\mathsf{Leaves}(\mathcal{Z}_i)| \leq |\mathcal{S}_i|$, so we can conclude:

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

Minimality of the ZT-parity-automaton with respect to HD automata. We intend to prove Theorem 4.15, 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, 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 $n_1, \ldots, n_k$ as children. For each child n_i we want to find an HD subautomaton $\mathcal{A}_i$ recognising the language associated to $\mathcal{F}|_{v(n_i)}$ 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 4.23. *Let n_0 be the root of the Zielonka tree of $\mathcal{F}$, and let $n_1, n_2, \ldots, n_k$ 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 $\mathcal{A}_1, \ldots, \mathcal{A}_k$ that are history-deterministic and such that $\mathcal{L}(\mathcal{A}_i) = \mathsf{Muller}_{v(n_i)}(\mathcal{F}|_{v(n_i)})$.*

The non-determinism of $\mathcal{A}$ will make this task considerably more laborious than in the previous paragraph, 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 ($\Sigma \notin \mathcal{F}$).

In order to simplify the proof, we will assume that all states are reachable using a sound resolver and that all automata have a single initial state, which can be done without loss 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: $\Sigma \notin \mathcal{F}$. Let $\mathcal{A} = (Q, \Sigma, q_0, \Gamma, \Delta, \mathbb{W})$ 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 $a \in \Sigma$, $r(\varepsilon, a) = \sigma(q_0, m_0, a)$ and for all $\rho \in \Delta^+$, $r(\rho, a) = \sigma(\mathsf{Target}(\rho), \mu(m_0, \rho), a)$, where m_0 is the initial state of $\mathcal{M}$ and $\mu \colon M \times \Delta^* \to M$ is its update function.

LEMMA 4.24 ([9]). *Every HD parity automaton admits a sound resolver implemented by a finite memory structure.*

As $\mathcal{M}$ is a pointed graph labelled with the transitions of $\mathcal{A}$, we could 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 (M, σ), we define their *composition*, which we write $\mathcal{A} \triangleleft_\sigma M = (Q \times M, \Sigma, (q_0, m_0), \Gamma, \Delta', \mathbb{W})$ 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 $\Delta \times E_M$, where E_M are the edges of the memory skeleton). We note that $\mathcal{A} \triangleleft_\sigma M$ 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 4.25. *Let $\mathcal{A}$ be an automaton and (M, σ) a memory structure for $\mathcal{A}$. The resolver implemented by (M, σ) is sound if and only if $\mathcal{A}$ and $\mathcal{A} \triangleleft_\sigma M$ recognise the same language.*

For the rest of the paragraph, we let $\mathcal{A} = (Q, \Sigma, q_0, \mathbb{N}, \Delta, \text{parity})$ 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 (M, σ). We let $\pi_\mathcal{A} : \mathcal{A} \triangleleft_\sigma M \to \mathcal{A}$ 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 4.26. *If ρ is a path in $\mathcal{A} \triangleleft_\sigma M$ 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.*

LEMMA 4.27. *Let $X \subseteq \Sigma$ and let $\mathcal{S}_X$ be an accessible X-FSCC of $\mathcal{A} \triangleleft_\sigma M$. Then, $\pi_\mathcal{A}(\mathcal{S}_X)$ induces an HD subautomaton of $\mathcal{A}$ recognising $\mathsf{Muller}_X(\mathcal{F}|_X) = \{w \in X^\omega \mid \mathrm{Inf}(w) \in \mathcal{F}\}$.*

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 M such that $(q_S, m_S) \in \mathcal{S}_X$. By Lemma 4.20, $\mathcal{S}_X$ 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

$$\mathcal{L}(\pi_\mathcal{A}(\mathcal{S}_X)) \subseteq \mathcal{L}(\mathcal{A}) \cap X^\omega = \mathsf{Muller}_X(\mathcal{F}|_X).$$

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 4.26), so

$$\mathcal{L}(\mathcal{S}_X) = \mathsf{Muller}_X(\mathcal{F}|_X) \subseteq \mathcal{L}(\pi_\mathcal{A}(\mathcal{S}_X)).$$

Moreover, a sound resolver for $\pi_\mathcal{A}(\mathcal{S}_X)$ is implemented by (M_{m_S}, σ) (the memory structure with initial state set to m_S). ∎

LEMMA 4.28. *Let $n \in N$ be a square node of the Zielonka tree of $\mathcal{F}$ ($v(n) \notin \mathcal{F}$), and let $n_1, n_2 \in \mathrm{Children}_{\mathcal{Z}_\mathcal{F}}(n)$ be two different children of n. If $\mathcal{S}_1$ and $\mathcal{S}_2$ are two accessible $v(n_1)$-FSCC and $v(n_2)$-FSCC in $\mathcal{A} \triangleleft_\sigma M$, 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 $\ell_i \in Cycles_{(q,m_i)}(\mathcal{A} \lhd_\sigma M)$ be the cycle over (q, m_i) containing all edges in $\mathcal{S}_i$. We note that $l_\Sigma(\ell_i) = v(n_i)$ and therefore $\min \gamma(\ell_i)$ has to be even (as $\mathcal{A} \lhd_\sigma M$ is deterministic), where l_Σ and γ are the labellings of $\mathcal{A} \lhd_\sigma M$ with input letters and output colours, respectively. By Remark 4.26, the $\pi_{\mathcal{A}}$-projections of ℓ_1 and ℓ_2 are cycles over q in $\mathcal{A}$ labelled with $v(n_1)$ and $v(n_2)$ 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 $\mathrm{Inf}(w) = v(n_1) \cup v(n_2)$, contradicting the fact that $v(n_1) \cup v(n_2) \notin \mathcal{F}$ (Remark 4.4).   ∎

Lemmas 4.19, 4.27 and 4.28 imply Proposition 4.23 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 4.28 is not valid if $\Sigma \in \mathcal{F}$. Indeed, even if we can find two rejecting cycles ℓ_1, ℓ_2 such that $l_\Sigma(\ell_i) = v(n_i)$, 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 $v(n_1) \cup v(n_2) \in \mathcal{F}$, 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 specific resolver for which we will be able to obtain a result analogous to Lemma 4.28. To do this, we first construct the letter game of $\mathcal{A}$, as introduced in [44], 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 [66] guided by the Zielonka tree, as presented in [34].

Let $\mathcal{A} = (Q, \Sigma, q_0, [d_{min}, d_{max}], \Delta, \mathsf{parity})$ be a parity automaton recognising $\mathsf{Muller}(\mathcal{F})$, and assume 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 $\mathcal{A}$.
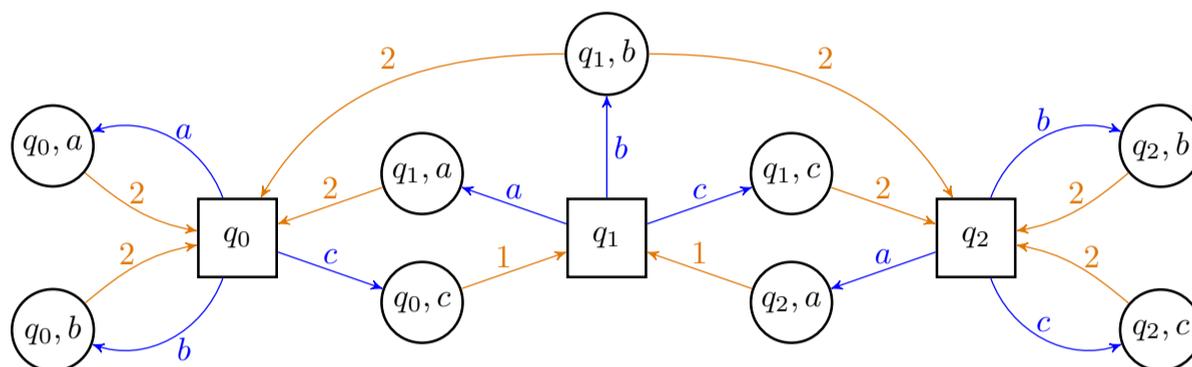

Figure 11. Letter game for the HD automaton from Figure 1, recognising the Muller language associated to $\mathcal{F} = \{\{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}\}$. Squares represent Adam's vertices (the states of the automaton) and circles Eve's ones. Blue edges correspond to input-letters, and orange 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 in order 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 2.3.

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 4.29. 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 4.30. 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 4.31. If ρ is a play in $\mathcal{G}_{\mathcal{A}}$, labelled $a_0 c_0 a_1 c_1 \cdots \in (\Sigma \cdot [d_{\min}, d_{\max}])^{\infty}$, the $\mathsf{aut}_{\mathcal{A}}$-projection of ρ is a run in $\mathcal{A}$ over $a_0 a_1 \cdots \in \Sigma^{\infty}$ producing $c_0 c_1 \cdots \in [d_{\min}, d_{\max}]^{\infty}$ as output.

LEMMA 4.32 ([44]). *A parity automaton $\mathcal{A}$ is HD if and only if Eve wins the letter game from some initial state of $\mathcal{A}$.*

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 note $\mathsf{Attr}_{\mathcal{G}}(c) = \mathsf{Attr}_{\mathcal{G}}(E_c)$, where E_c is the set of edges coloured c.

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 assume 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 assume that Eve wins the letter game from any vertex (see Lemma 2.5). We let n_0 be the root of the Zielonka tree of $\mathcal{F}$ (assumed to be round, that is $v(n_0) \in \mathcal{F}$), let $n_1, \ldots, n_k$ be its children, and let $\Sigma_i = v(n_i) \subseteq \Sigma$ (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 12. 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 $\mathcal{G}_{\mathcal{A}}$.
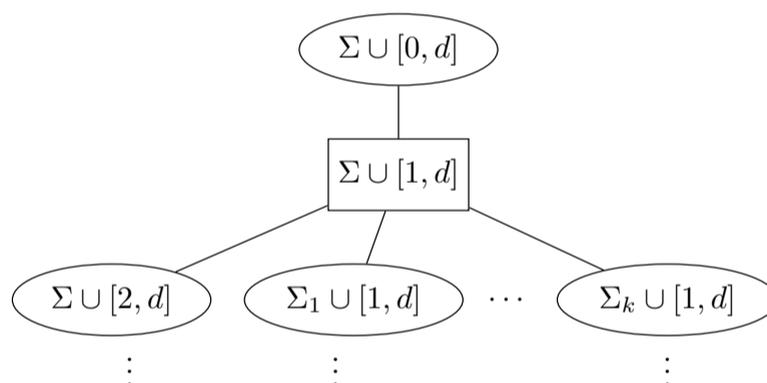

Figure 12. First levels of the Zielonka tree of the Muller condition $\mathcal{F} \to \mathsf{parity}$, which is the winning condition of the letter game $\mathcal{G}_{\mathcal{A}}$.

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

$$V' = \mathsf{Attr}_{\mathcal{G}'}(x) \sqcup V_1 \sqcup A_1 \sqcup \cdots \sqcup V_l \sqcup A_l,$$

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 $v' \in \text{Attr}_{\mathcal{G}'}(x) \sqcup V_1 \sqcup A_1 \sqcup \ldots V_{j-1} \sqcup A_{j-1}$. 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.

— $A_j = \text{Attr}_{\mathcal{G}_j}(V_j)$, where $\mathcal{G}_j$ is the subgame induced by the subset of vertices given by $V \setminus (\text{Attr}_{\mathcal{G}'}(x) \sqcup V_1 \sqcup \ldots \sqcup V_{j-1} \sqcup A_{j-1})$ (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:

$$\mathcal{D}_{\mathcal{G}'} = \langle \text{Attr}_{\mathcal{G}'}(x), (V_1, A_1, \mathcal{D}_{\mathcal{G}'_1}), (V_2, A_2, \mathcal{D}_{\mathcal{G}'_2}), \ldots, (V_l, A_l, \mathcal{D}_{\mathcal{G}'_l})\rangle,$$

where $\text{Attr}_{\mathcal{G}'}(x) \sqcup V_1 \sqcup A_1 \sqcup \cdots \sqcup V_l \sqcup A_l$ 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).

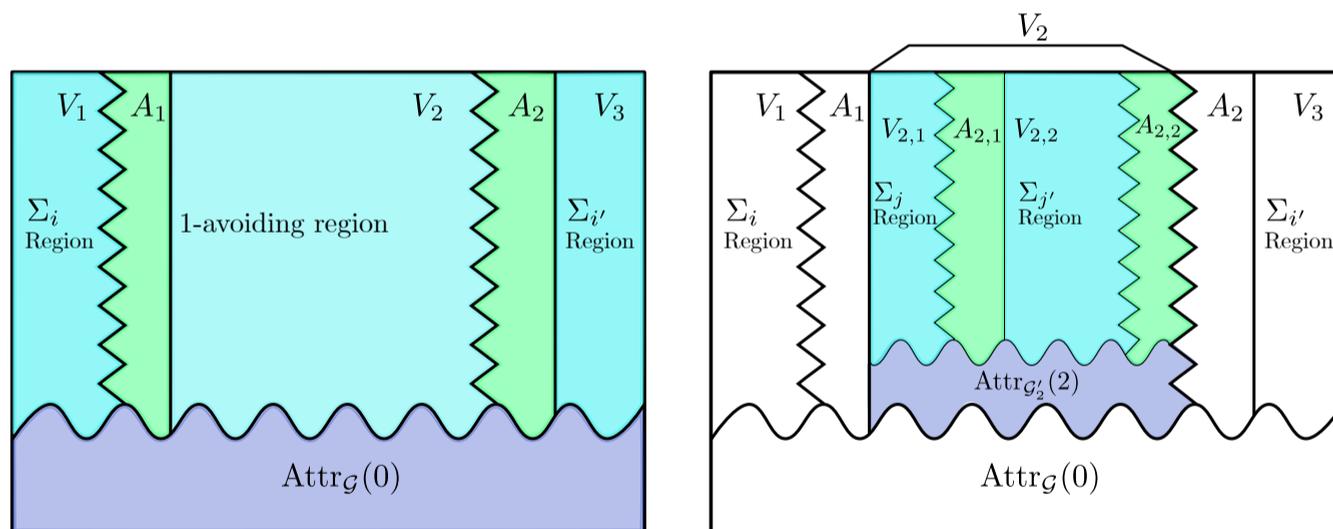A representation of an attractor decomposition appears in Figure 13.

Figure 13. 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.

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. We say that the full game 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 G' there is one and only one minimal y-avoiding region of $\mathcal{D}_{G'}$ containing S (note that y might equal -1).

REMARK 4.33. A 0-recursive attractor decomposition $\mathcal{D}_{G_{\mathcal{A}}}$ of $G_{\mathcal{A}}$ induces a partition of the vertices into

$$V = S_1 \sqcup \cdots \sqcup S_r \ \sqcup \ A_1 \sqcup \ldots A_r \ \sqcup \ B_1 \sqcup \cdots \sqcup B_s,$$

such that:

— S_j is a Σ_i-region of $\mathcal{D}_{G_{\mathcal{A}}}$, for some $i \in \{1, \ldots, k\}$,
— $A_j = \mathrm{Attr}_{G_j}(S_j)$ for some subgame G_j appearing at some level of the decomposition,
— $B_j = \mathrm{Attr}_{G'_j}(x)$ for some even integer x and some $x - 1$-avoiding region 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 $\mathcal{D}_{G_{\mathcal{A}}}$ 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 $G_{\mathcal{A}}$. We remark that, by definition of an attractor decomposition, Eve wins $G_{\mathcal{A}}(S_j)$ from every vertex for every S_j. See Figure 13 for an illustration.

The proof that $G_{\mathcal{A}}$ admits a full attractor decomposition uses the ideas appearing in [34, Section 3].

LEMMA 4.34. *Let x be an even integer. If G' is a subgame of $G_{\mathcal{A}}$ with no colour smaller than x and such that Eve can win from every vertex, then it admits an x-attractor decomposition. In particular, $G_{\mathcal{A}}$ admits a full attractor decomposition.*

PROOF. We assume without loss of generality that $x = 0$. Suppose that $V_1, A_1, \ldots V_{j-1}, A_{j-1}$ have already been defined and that they verify the desired properties. Suppose that the game G_j with vertices $V \setminus \left(\mathrm{Attr}_{G'}(0) \sqcup V_1 \sqcup \ldots V_{j-1} \sqcup A_{j-1} \right)$ is non-empty. First, note that Eve wins G_j from any position. Indeed, Eve wins G' from any vertex v in G_j (as we suppose that she can win G' starting anywhere); moreover, since $v \notin A_{j'}$ for any $j' < j$, Adam has a strategy from v forcing to remain in G_j, and Eve has to be able to win against any such strategy.

We prove that either (1) there is some $i \in \{1, \ldots, k\}$ and v vertex in 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 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 producing 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 G_j). Afterward, he continues 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 G_j from v.

We assume 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 for each of them we fix a strategy strat_v that is winning in 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_v in G_j, then $v' \in V_j$ (Eve can still win without producing colours in $\Sigma \setminus \Sigma_i$), so Adam cannot force leaving V_j. This proves that:

1. strat_v is winning in $G_{\mathcal{A}}(V_j)$ from v,
2. if $v \in V_j$ is controlled by Adam and $v \to v'$ is an edge in $G_{\mathcal{A}}$, then $v' \in \mathrm{Attr}_{G'}(0) \sqcup V_1 \sqcup A_1 \sqcup \ldots V_{j-1} \sqcup A_{j-1} \sqcup V_j$.

Also, if a vertex v controlled by Adam is in V_j, no edge $v \xrightarrow{a \notin \Sigma_i} v'$ appears in G_j, so no colour of $\Sigma \setminus \Sigma_i$ appears in $G_{\mathcal{A}}(V_j)$.

To finish the proof, we define A_j to be the attractor of V_j in G_j.

The existence of a full attractor decomposition for $G_{\mathcal{A}}$ follows from the fact that any $x + 1$-avoiding region of an x-attractor decomposition verifies the hypothesis of the lemma. ∎

LEMMA 4.35 ([66]). *Let G be a game using a Muller acceptance condition such that Eve wins G from every vertex. Then, there is a finite memory structure $(\mathcal{M}, \sigma)$ over G implementing a winning strategy uniformly, that is, for every vertex v of G there is a memory state m_v in $\mathcal{M}$ such that the memory structure $(\mathcal{M}_{m_v}, \sigma)$ implements a winning strategy from v.*

For the rest of the paragraph, we fix a 0-recursive attractor decomposition $\mathcal{D}_{G_{\mathcal{A}}}$ for $G_{\mathcal{A}}$ 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 $(\mathcal{M}_j, \sigma_j)$ uniformly implementing a winning strategy for Eve in $G_{\mathcal{A}}(S_j)$ (as given by Lemma 4.35). As in the previous paragraph, we can consider the composition $G_{\mathcal{A}}(S_j) \triangleleft_{\sigma_j} \mathcal{M}_j$ 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 $G_{\mathcal{A}}(S_j) \triangleleft_{\sigma_j} \mathcal{M}_j$, and since $(\mathcal{M}_j, \sigma_j)$ implements a winning strategy, any infinite path in $G_{\mathcal{A}}(S_j) \triangleleft_{\sigma_j} \mathcal{M}_j$ produces a set of colours in $\mathcal{F} \to$ parity. We let $\pi_{\mathcal{G}} \colon G_{\mathcal{A}}(S_j) \triangleleft_{\sigma_j} \mathcal{M}_j \to G_{\mathcal{A}}(S_j)$ be the projection into $G_{\mathcal{A}}(S_j)$.

A subgraph G_j of $G_{\mathcal{A}}(S_j) \triangleleft_{\sigma_j} \mathcal{M}_j$ 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 $G_{\mathcal{A}}(S_j) \triangleleft_{\sigma_j} \mathcal{M}_j$ to the graph where Adam's choices are restricted to letters in X that is moreover *X-Adam-closed*. We say that a subgraph G of $G_{\mathcal{A}}$ is an *X-closed subgame* (with respect to the attractor decomposition $\mathcal{D}_{G_{\mathcal{A}}}$ and a family of finite memory strategies) if $G = \pi_{\mathcal{G}}(G_j)$ for G_j some *X-Adam-closed* SCC of some product $G_{\mathcal{A}}(S_j) \triangleleft_{\sigma_j} \mathcal{M}_j$.

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 4.36. *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 4.29 we obtain:

LEMMA 4.37. *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 $\mathrm{aut}_\mathcal{A}(\mathcal{G}_X)$ is HD and recognises $\mathsf{Muller}_X(\mathcal{F}|_X)$.*

LEMMA 4.38. *If a product $\mathcal{G}_\mathcal{A}(S_j) \triangleleft_{\sigma_j} \mathcal{M}_j$ does not contain any X-Adam-closed subgraph, for $X \subseteq \Sigma$, then from any vertex (q, m) Adam can force leaving S_j while playing only letters in X. That is, there is a path $(q, m) \rightsquigarrow (q', m')$ in $\mathcal{G}_\mathcal{A}(S_j) \triangleleft_{\sigma_j} \mathcal{M}_j$ 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 $\mathcal{G}_\mathcal{A}(S_j) \triangleleft_{\sigma_j} \mathcal{M}_j$ consisting of the vertices that can be reachable from (q, m) by reading letters in X would form an *X-Adam-closed* subgraph. ∎

LEMMA 4.39. *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. Assume that the full attractor decomposition of $\mathcal{G}_\mathcal{A}$ induced by $\mathcal{D}_{\mathcal{G}_\mathcal{A}}$ is the following:

$$V = S_1 \sqcup \cdots \sqcup S_r \ \sqcup \ A_1 \sqcup \ldots A_r \ \sqcup \ B_1 \sqcup \cdots \sqcup B_s,$$

We fix the following strategy strat for Eve in the letter game:
— whenever the play lands to B_j, where $B_j = \mathrm{Attr}_{\mathcal{G}'_j}(x)$ for some even colour x, she forces producing colour x,
— whenever the play arrives to some A_j, she forces going 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 4.40. *Let ρ be a play consistent with strat (from any vertex), and let $y \geq -1$ be the maximal odd number such that $\mathrm{Inf}(\rho)$ is contained in a y-avoiding region S of $\mathcal{D}_{\mathcal{G}_\mathcal{A}}$. Then, either ρ eventually stays in a Σ_i-region S_j contained in S, or the minimal colour produced infinitely often by ρ is $y + 1$.*

Proof. Let $\mathrm{Attr}_{\mathcal{S}}(y+1) \sqcup V_1 \sqcup A_1 \sqcup \ldots \ldots, V_l \sqcup A_l$ 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 $v' \in \mathrm{Attr}_{\mathcal{S}}(y+1) \sqcup V_1 \sqcup A_1 \sqcup \ldots V_{j-1} \sqcup A_{j-1}$. 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 $\mathrm{Inf}(w) = \Sigma_i$,
   (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 4.38. By Claim 4.40, the minimal colour produced infinitely often by such a play is even. By Remark 4.31, we can project such a play in the automaton $\mathcal{A}$, obtaining an accepting run over w. This is a contradiction, since $w \notin \mathrm{Muller}(\mathcal{F}) = \mathcal{L}(\mathcal{A})$ (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 4.23 in the case in which the root of $\mathcal{Z}_{\mathcal{F}}$ is round: from Lemma 4.39, we obtain Σ_i-closed subgames in $\mathcal{G}_{\mathcal{A}}$ for each $i \in \{1, \ldots, k\}$ that are moreover contained in Σ_i-regions. Therefore, their $\mathrm{aut}_{\mathcal{A}}$-projections are disjoint (Remark 4.30), and each of these projections induces an HD-subautomaton recognising $\mathcal{F}|_{\Sigma_i}$ (Lemma 4.37).

4.3   A minimal history-deterministic Rabin automaton

In this section, we present the construction of a history-deterministic Rabin automaton $\mathcal{A}_{\mathcal{Z}_{\mathcal{F}}}^{\mathrm{Rabin}}$ for a Muller language $\mathrm{Muller}(\mathcal{F})$ using the Zielonka tree $\mathcal{Z}_{\mathcal{F}}$, and prove its minimality (Theorem 4.51). The automaton $\mathcal{A}_{\mathcal{Z}_{\mathcal{F}}}^{\mathrm{Rabin}}$ can be seen as a quotient of the ZT-parity-automaton; that is, $\mathcal{A}_{\mathcal{Z}_{\mathcal{F}}}^{\mathrm{Rabin}}$ is obtained by merging some states of $\mathcal{A}_{\mathcal{Z}_{\mathcal{F}}}^{\mathrm{parity}}$. Thus, we replace the complexity in the number of states by complexity in the acceptance condition. The size of the automaton $\mathcal{A}_{\mathcal{Z}_{\mathcal{F}}}^{\mathrm{Rabin}}$ is a well-studied parameter of Zielonka trees: its round-branching width, $\mathrm{rbw}(\mathcal{Z}_{\mathcal{F}})$. This

parameter was introduced by Dziembowski, Jurdziński and Walukiewicz [34] (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 Muller($\mathcal{F}$) as an acceptance condition (see Proposition 4.52 below). In this paper, we are not concerned with the memory of winning conditions, but we will use the result from [34] to obtain the minimality of $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_{\mathcal{F}}}$.

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 [12] for the definition of universal HD automata).

4.3.1   The Zielonka-tree-HD-Rabin-automaton

DEFINITION 4.41 ([34]). 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:

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

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

LEMMA 4.42. *Let* $T = (N = N_\bigcirc \sqcup N_\square, \leq)$ *be a tree with nodes partitioned into round and square nodes. There exists a mapping* $\eta \colon \mathsf{Leaves}(T) \to \{1, 2, \ldots, \mathsf{rbw}(T)\}$ *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 4.43. Let $\mathcal{F} = \{\{a, b\}, \{a, c\}, \{b\}\}$ be the family of subsets considered in Example 4.5. 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.   ◆

DEFINITION 4.44 (Zielonka-tree-HD-Rabin-automaton). Let $\mathcal{F} \subseteq 2^{\Sigma}_+$, let $\mathcal{Z}_{\mathcal{F}} = (N = N_\bigcirc \sqcup N_\square, \leq)$ be its Zielonka tree and $\eta \colon \mathsf{Leaves}(\mathcal{Z}_{\mathcal{F}}) \to \{1, 2, \ldots, \mathsf{rbw}(\mathcal{Z}_{\mathcal{F}})\}$ be 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:

— $Q = \{1, 2, \ldots, \mathsf{rbw}(\mathcal{Z}_{\mathcal{F}})\}$,
— $I = Q$,[8]

— $\Gamma = N$ (the colours of the acceptance condition are the nodes of the Zielonka tree),

— $\delta(q, a) = \{(\text{Jump}(l, \text{Supp}(l, a)), \text{Supp}(l, a)) \mid l \in \text{Leaves}(\mathcal{Z}_{\mathcal{F}}) \text{ such that } \eta(l) = q\}$,

— $R = \{(G_n, R_n)\}_{n \in N_{\bigcirc}}$, where G_n and R_n are defined as follows: Let n be a round node and n' be any node of $\mathcal{Z}_{\mathcal{F}}$,

$$\begin{cases} n' \in G_n & \text{if } n' = n, \\ n' \in R_n & \text{if } n' \neq n \text{ and } n \text{ is not an ancestor of } n'. \end{cases}$$

REMARK 4.45. Although we will usually say that $\mathcal{A}_{\mathcal{Z}_{\mathcal{F}}}^{\text{Rabin}}$ 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 $\mathcal{A}_{\mathcal{Z}_{\mathcal{F}}}^{\text{Rabin}}$ has $\text{rbw}(\mathcal{Z}_{\mathcal{F}})$ 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 $l \in \text{Leaves}(\mathcal{Z}_{\mathcal{F}})$ and letter $a \in \Sigma$, we identify the deepest ancestor $n = \text{Supp}(l, a)$ 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 (we note that the output colour is given by $n = \text{Supp}(l, a)$, although this node does not appear as a state of the automaton). This way, we can identify a run in the automaton $\mathcal{A}_{\mathcal{Z}_{\mathcal{F}}}^{\text{Rabin}}$ with a promenade through the nodes of the Zielonka tree in which jumps 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 4.49 below).

Another way of presenting the automaton $\mathcal{A}_{\mathcal{Z}_{\mathcal{F}}}^{\text{Rabin}}$ is as a quotient of the deterministic parity automaton $\mathcal{A}_{\mathcal{Z}_{\mathcal{F}}}^{\text{parity}}$. Indeed, the graph structure and the labelling by input letters of $\mathcal{A}_{\mathcal{Z}_{\mathcal{F}}}^{\text{Rabin}}$ is obtained by merging the states of $\mathcal{A}_{\mathcal{Z}_{\mathcal{F}}}^{\text{parity}}$ (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 $\text{Muller}(\mathcal{F})$.

EXAMPLE 4.46. The ZT-HD-Rabin-automaton $\mathcal{A}_{\mathcal{Z}_{\mathcal{F}}}^{\text{Rabin}}$ of the family $\mathcal{F} = \{\{a, b\}, \{a, c\}, \{b\}\}$ from Example 4.5 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 $\eta \colon \text{Leaves}(\mathcal{Z}_{\mathcal{F}}) \to \{1, 2\}$ is represented by the numbers below its branches.

---

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

The Rabin condition of this automaton is given by two Rabin pairs (corresponding to the round nodes of the Zielonka tree):

$$G_\beta = \{\beta\}, \quad R_\beta = \{\alpha, \lambda, \xi, \zeta\},$$
$$G_\lambda = \{\lambda\}, \quad R_\lambda = \{\alpha, \beta, \theta\}.$$

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 10, and replacing the output colours by suitable nodes from the Zielonka tree.   ◆
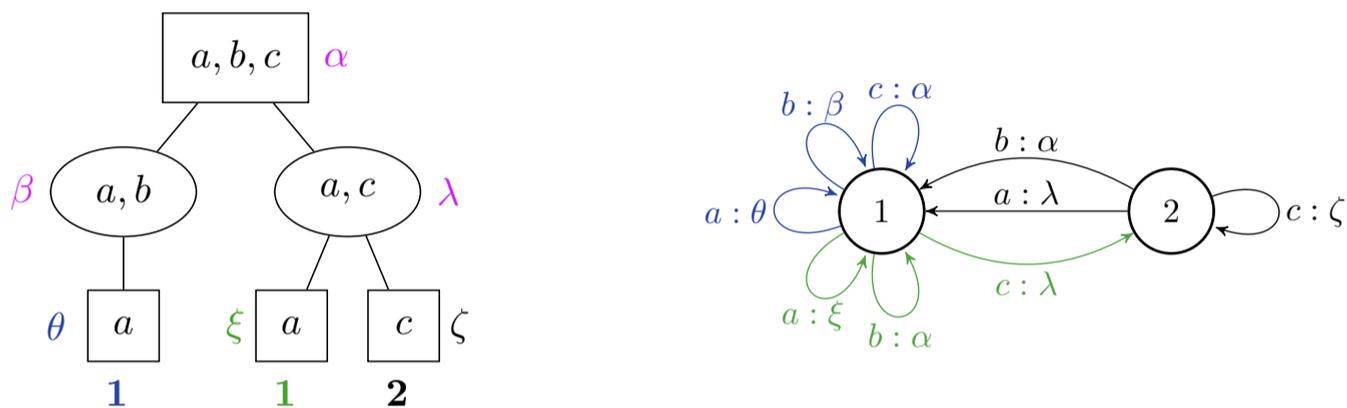

Figure 14. On the left, the Zielonka tree of $\mathcal{F} = \{\{a,b\}, \{a,c\}, \{b\}\}$. On the right, the ZT-HD-Rabin-automaton $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$. Blue transitions correspond to those coming from leaf θ, and green ones to those originating from leaf ξ.

REMARK 4.47. 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 D (Proposition D.1). For the language from the previous example, an equivalent automaton is proposed in Figure 19

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

PROPOSITION 4.48 (Correctness). *Let $\mathcal{F} \subseteq 2^\Sigma_+$ be a family of non-empty subsets. Then,*

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

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

LEMMA 4.49. *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 $\mathit{Rabin}_N(R)$, for $R = \{(G_n, R_n)\}_{n \in N_\bigcirc}$ 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 $\mathrm{Inf}(u)$ and this minimal node is round (recall that the root is the minimal element for $\leq$).*

PROOF. Assume 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 (G_n, R_n). It is clear that $\mathsf{Inf}(u) \cap G_n \neq \emptyset$, because $n \in G_n$. It suffices to show that $\mathsf{Inf}(u) \cap R_n = \emptyset$: By minimality, any other node $n' \in \mathsf{Inf}(u)$ is a descendant of n (equivalently, n is an ancestor of n'), so $n' \notin R_n$.

Conversely, assume that $u \in \mathsf{Rabin}_N(R)$. Then, there is some round node $n \in N_\bigcirc$ such that $\mathsf{Inf}(u) \cap G_n \neq \emptyset$ and $\mathsf{Inf}(u) \cap R_n = \emptyset$. Since $G_n = \{n\}$, we deduce that $n \in \mathsf{Inf}(u)$. Moreover, as $\mathsf{Inf}(u) \cap R_n = \emptyset$, 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 4.50. *There exists a morphism of automata $\varphi \colon \mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\mathsf{parity}} \to \mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\mathsf{Rabin}}$.*

PROOF. We define the morphism φ as follows:

— $\varphi_V(l) = \eta(l)$, for $l \in \mathsf{Leaves}(\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\mathsf{parity}})$,

— for a transition $e = l \xrightarrow{a:c} l'$ in $\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\mathsf{parity}}$, 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 $\rho = l_0 \xrightarrow{w_0} l_1 \xrightarrow{w_1} l_2 \xrightarrow{w_2} \cdots \in \mathcal{R}un(\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\mathsf{parity}})$ be an infinite run in $\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\mathsf{parity}}$ (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 $\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\mathsf{Rabin}}$ is $\gamma'(\rho') = n_0 n_1 n_2 \cdots \in N^\omega$. In the proof of Proposition 4.10, we proved (Claims 4.12 and 4.13) that there exists a unique node n_w appearing infinitely often in $\gamma'(\rho')$. Moreover, we proved that ρ is accepting in $\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\mathsf{parity}}$ if and only if n_w is round. Lemma 4.49 allows us to conclude that $\varphi_{\mathcal{R}uns}(\rho)$ is accepting in $\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\mathsf{Rabin}}$ if and only if ρ is accepting in $\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\mathsf{parity}}$. ∎

PROOF OF PROPOSITION 4.48. $\mathcal{L}(\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\mathsf{Rabin}}) \subseteq \mathbf{Muller}_\Sigma(\mathcal{F})$: Let $w \in \mathcal{L}(\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\mathsf{Rabin}})$ and let $u \in N^\omega$ be the sequence of nodes produced as output of an accepting run over w in $\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\mathsf{Rabin}}$. By Lemma 4.49, there is a unique minimal node n for $\leq$ appearing infinitely often in u and moreover n is round. Let $n_1, \ldots, n_k$ 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 $\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\mathsf{Rabin}}$).

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 assume 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 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 $\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\mathsf{Rabin}}$, 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 colour a go to Q_{i+1}, for $1 \leq i \leq k - 1$ (and to Q_1 if $i = k$). Also, if $a \in v(n_i)$, 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 $\mathsf{Inf}(w) \subseteq v(n_j)$. However, the only transitions from Q_j that would produce n as output are those corresponding to a colour $a \notin v(n_j)$, 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: We claim that 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 4.50) and the correctness of $\mathcal{A}^{\mathsf{parity}}_{\mathcal{Z}_\mathcal{F}}$ (Proposition 4.10) 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 $r(\rho_R, a) = \varphi(e)$, 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 $w \in \mathsf{Muller}(\mathcal{F})$. ∎

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

We devote this section to the proof of the optimality of $\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}$.

THEOREM 4.51 (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, $|\mathcal{A}^{\mathsf{Rabin}}_{\mathcal{Z}_\mathcal{F}}| \leq |\mathcal{A}|$.*

PROPOSITION 4.52 ([34]). *Let $L = \mathsf{Muller}_\Sigma(\mathcal{F})$ be a Muller language.*

1. *If Eve wins a game with L as acceptance set from a position v, there is a winning strategy from v for her implemented by a memory structure of size $\mathsf{rbw}(\mathcal{Z}_\mathcal{F})$.*
2. *There exists a game $\mathcal{G}$ using L as acceptance condition in which Eve can win from a position v, but there is no winning strategy from v for her implemented by a memory structure of size strictly smaller than $\mathsf{rbw}(\mathcal{Z}_\mathcal{F})$.*

LEMMA 4.53 ([51, 94]). *Rabin languages are positionally determined, that is, if Eve wins a game using a Rabin acceptance condition from a position v, there is a winning strategy from v for her implemented by a memory structure of size 1.*

COROLLARY 4.54. *Let $\mathcal{A}$ be a history-deterministic Rabin automaton. Then, if Eve wins a game with $\mathbb{W} = \mathcal{L}(\mathcal{A})$ as acceptance set from a position v, there is a winning strategy from v for her implemented by a memory structure of size $|\mathcal{A}|$.*

PROOF. Let $\mathcal{G}$ be a game with $\mathbb{W} = \mathcal{L}(\mathcal{A})$ as acceptance set. In order to be able to take the product by $\mathcal{A}$ and obtain an equivalent game, we transform $\mathcal{G}$ into 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. By Proposition 2.7, if Eve wins $\mathcal{G}$ from a vertex v, she wins $\tilde{\mathcal{G}} \ltimes \mathcal{A}$ from a vertex (v, q_0), where q_0 is an initial vertex of $\mathcal{A}$. Moreover, the game $\tilde{\mathcal{G}} \ltimes \mathcal{A}$ uses the acceptance set from $\mathcal{A}$, which is a Rabin language, so, by Lemma 4.53, she can win using a strategy given by a function $\sigma \colon \tilde{V}_{\mathrm{Eve}} \to \tilde{E}$, where Q is the set of states of $\mathcal{A}$ and $\tilde{V}_{\mathrm{Eve}}$ the vertices controlled by Eve in $\tilde{\mathcal{G}}$ (a subset of $(V_{\mathrm{Eve}} \sqcup (V \times E)) \times Q$). We build a memory structure $(\mathcal{M}, \sigma_{\mathcal{M}})$ of size $|Q|$ that projects the strategy implemented by σ onto $\mathcal{G}$:

— its set of states is $M = Q$,

— the initial state is q_0,

— the update function $\mu \colon M \times E \to M$ sends $\mu(q, e) = q'$ if $\sigma((v, e), q) = ((v, e), q) \to (v', q')$ is the move chosen by σ from vertex $((v, e), q)$,

— for $v \in V_{\mathrm{Eve}}, q \in M$, we let $\sigma_{\mathcal{M}}(v, q) = e$ if e is the move chosen by σ from (v, q), that is, if $\sigma(v, q) = (v, q) \to ((v, q), e)$.

Since σ implements a winning strategy in $\tilde{\mathcal{G}} \ltimes \mathcal{A}$ from (v, q_0), its projection onto $\mathcal{G}$ via the memory structure $(\mathcal{M}, \sigma_{\mathcal{M}})$ is a strategy that verifies that any play consistent with it produces as output a word in $\mathcal{L}(\mathcal{A})$, so it is winning. ∎

Theorem 4.51 is obtained by combining the fact that $|\mathcal{A}^{\mathrm{Rabin}}_{\mathcal{Z}_{\mathcal{F}}}| = \mathrm{rbw}(\mathcal{Z}_{\mathcal{F}})$ with Proposition 4.52 (second item) and Corollary 4.54.

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

In Section 4, 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 [93]. The alternating chains of loops of a DMA are known to determine the parity index of the language it recognises [93], 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 5.1. In Section 5.2, 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 5.19). In Section 5.3, 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 5.28). 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 5.35 and 5.36). 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 colours (Theorem 5.34). The optimality of these constructions is shown in Section 5.4. 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 $\mathcal{TS} = (G_{\mathcal{TS}}, \mathsf{Acc}_{\mathcal{TS}})$ be a Muller transition system with underlying graph $G_{\mathcal{TS}} = (V, E, \mathsf{Source}, \mathsf{Target}, I)$ and using a Muller acceptance condition $\mathsf{Acc}_{\mathcal{TS}} = (\gamma, \Gamma, \mathsf{Muller}_\Gamma(\mathcal{F}))$.

5.1   The alternating cycle decomposition

DEFINITION 5.1. Let $\ell_0 \in \mathit{Cycles}(\mathcal{TS})$ be a cycle. We define the *tree of alternating subcycles* of ℓ_0, denoted $\mathsf{AltTree}(\ell_0) = (N, \leq, v \colon N \to \mathit{Cycles}(\mathcal{TS}))$ 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 ($\gamma(\ell) \in \mathcal{F}$), then it is a round node, and its children are labelled exactly with the maximal subcycles $\ell' \subseteq \ell$ such that ℓ' is rejecting ($\gamma(\ell') \notin \mathcal{F}$).
— If a node is labelled $\ell \in \mathit{Cycles}(\mathcal{TS})$, and ℓ is a rejecting cycle ($\gamma(\ell) \notin \mathcal{F}$), then it is a square node, and its children are labelled exactly with the maximal subcycles $\ell' \subseteq \ell$ such that ℓ' is accepting ($\gamma(\ell') \in \mathcal{F}$).

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

REMARK 5.2. 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 $v(n_1) \subsetneq \ell' \subseteq v(n)$, then $v(n_1)$ is accepting $\iff$ ℓ' is rejecting $\iff$ $v(n)$ is rejecting.

DEFINITION 5.3 (Alternating cycle decomposition). Let $\mathcal{TS}$ be a transition system, and let $\ell_1, \ell_2, \ldots, \ell_k$ be an enumeration of its maximal cycles (that is, the edge set of its SCCs). We define the *alternating cycle decomposition* of $\mathcal{TS}$ as the forest $\mathcal{ACD}_{\mathcal{TS}} = \{\mathsf{AltTree}(\ell_1), \ldots, \mathsf{AltTree}(\ell_k)\}$.

We let N_{ℓ_i} be the set of nodes of $\mathsf{AltTree}(\ell_i)$, and n_{ℓ_i} its root. We will assume 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, from now on we equip the trees of $\mathcal{ACD}_{\mathcal{TS}}$ with an arbitrary order making them ordered trees, without explicitly mentioning it.

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 $v \in v_{\mathsf{States}}(n_{\ell_i})$. 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 $N_v = \{n \in N_{\ell_i} \mid v \in v_{\mathsf{States}}(n)\}$. 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 v, whose restriction to $\mathcal{T}_v$ has an image in $\mathcal{Cycles}_v(\mathcal{TS})$.

REMARK 5.4. Let $v \in v_{\mathsf{States}}(n_{\ell_i})$. 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 $e \in v(n')$. We remark that if $e = v \rightarrow v'$, then $\mathsf{Supp}(n, e)$ is a node in both $\mathcal{T}_v$ and $\mathcal{T}_{v'}$.

EXAMPLE 5.5. 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 $v(n)$ and the set of states of it. For example, $v(\kappa) = \{g, h, i\}$ and $v_{\mathsf{States}}(\kappa) = \{v_3, v_4\}$. 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 the next section.   ◆
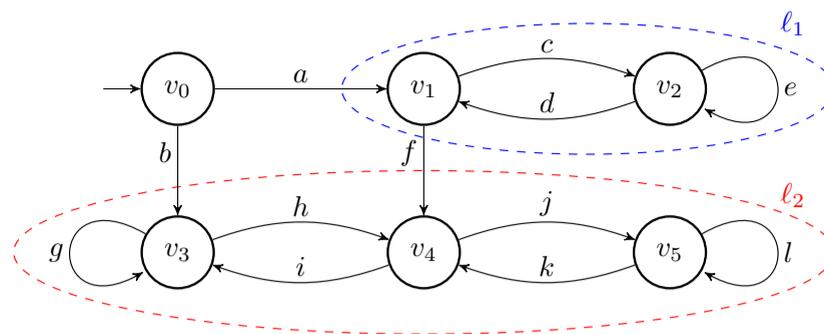
Figure 15. Transition system $\mathcal{TS}$ using a Muller acceptance condition given by $\mathcal{F} = \{\{c, d, e\}, \{e\}, \{g, h, i\},$ $\{l\}, \{h, i, j, k\}, \{j, k\}\}$. The two maximal cycles, ℓ_1 and ℓ_2, are encircled by blue and red dashed lines, respectively.
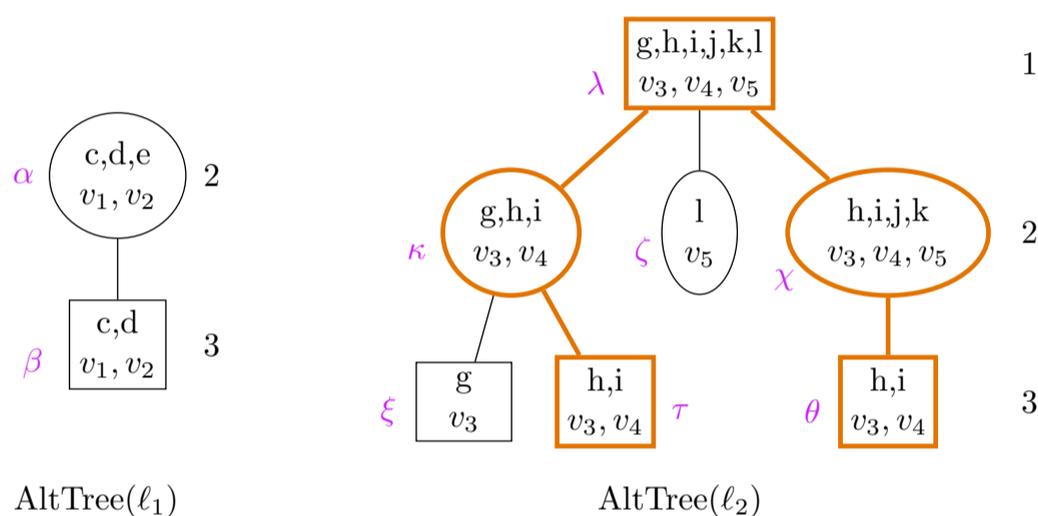

Figure 16. Alternating cycle decomposition of $\mathcal{TS}$. In bold orange, the local subtree at v_4, $\mathcal{T}_{v_4}$.

REMARK 5.6. Let $\mathcal{TS}$ be a Muller TS using as acceptance set $\mathbb{W} = \text{Muller}_\Gamma(\mathcal{F})$, and let $\overline{\mathcal{TS}}$ be the TS obtained by replacing $\mathbb{W}$ with $\overline{\mathbb{W}} = \Gamma^\omega \setminus \mathbb{W}$ (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: $\text{Nodes}_\bigcirc(\mathcal{ACD}_{\overline{\mathcal{TS}}}) = \text{Nodes}_\square(\mathcal{ACD}_{\mathcal{TS}})$ and $\text{Nodes}_\square(\mathcal{ACD}_{\overline{\mathcal{TS}}}) = \text{Nodes}_\bigcirc(\mathcal{ACD}_{\mathcal{TS}})$.

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 5.7. The Zielonka tree can be seen as a special case of the alternating cycle decomposition. Indeed, a Muller language $\text{Muller}_\Sigma(\mathcal{F})$ 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 $\mathcal{F}$.

REMARK 5.8 (Size and computation of the ACD). Let $\mathcal{TS}$ be a Muller TS and let $\mathcal{Z}_\mathcal{F}$ be the Zielonka tree of its acceptance set. It can be shown that for each vertex v of $\mathcal{TS}$ we have that

$|\mathcal{T}_v| \leq |\mathcal{Z}_{\mathcal{F}}|$, and therefore the size of $\mathcal{ACD}_{\mathcal{TS}}$ is polynomial in $\mathcal{Z}_{\mathcal{F}}$. This, and the question of the complexity of computing the ACD is the subject of an independent work [26].

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 $\Gamma_v = Cycles_v(\mathcal{TS})$ associated to:

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

We note that $\mathsf{LocalMuller}_{\mathcal{TS}}(v)$ is determined by singletons ($C \in \mathsf{LocalMuller}_{\mathcal{TS}}(v)$ if and only if $\{\bigcup_{\ell \in C} \ell\} \in \mathsf{LocalMuller}_{\mathcal{TS}}(v)$). For simplicity, and by a slight abuse of notation, we will work as if $\mathsf{LocalMuller}_{\mathcal{TS}}(v) \subseteq Cycles_v(\mathcal{TS})$. Also, 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 the 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 5.9. *The tree $\mathcal{T}_v$ is the Zielonka tree of the family $\mathsf{LocalMuller}_{\mathcal{TS}}(v)$,[9] for any recurrent vertex v.*

5.2   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 output colours, we need to pay attention to the parity of the minimal colour used in different SCCs. To incorporate this parameter in the transformation, we define positive and negative ACDs.

Let $\mathcal{TS}$ be a Muller transition system and let $\mathcal{ACD}_{\mathcal{TS}} = \{\mathsf{AltTree}(\ell_1), \ldots, \mathsf{AltTree}(\ell_k)\}$ 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,

---

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

— $p_{\mathcal{ACD}}(n) = \mathsf{Depth}(n) + 1$, 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 5.10. 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 5.11. 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 5.12. In the previous Example 5.5, $\mathsf{AltTree}(\ell_1)$ 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 5.13 (ACD-parity-transform). Let $\mathcal{TS}$ be a Muller TS with $\mathcal{ACD}_{\mathcal{TS}} = \{\mathsf{AltTree}(\ell_1)$ $,\ldots, \mathsf{AltTree}(\ell_k)\}$. We define the *ACD-parity-transform* of $\mathcal{TS}$ be the parity TS $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS}) = (G', \mathsf{Acc}')$, with $G' = (V', E', \mathsf{Source}', \mathsf{Target}', I')$, and $\mathsf{Acc}' = (\gamma', [\min_{\mathcal{TS}}, \max_{\mathcal{TS}}], \mathsf{parity})$ 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 $e_n = (v, n) \xrightarrow{\gamma'(e_n)} (v', n')$. Formally,

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

If v and v' are not in the same SCC, we let n' be the leftmost leaf in $\mathcal{T}_{v'}$ and $\gamma'(e_n) = \min_{\mathcal{TS}}$.[10] If v and v' belong to the same SCC, we let:

— $n' = \mathsf{Jump}_{\mathcal{T}_{v'}}(n, \mathsf{Supp}(n, e))$,

— $\gamma'(e_n) = p_{\mathcal{ACD}}(\mathsf{Supp}(n, e))$.

---

10   The colours associated to transitions changing of SCC are *almost* arbitrary (we could even leave them uncoloured). We define them to be the minimal colour used so that the obtained transition system is normalised in the sense of Section 6.2.

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 $ñ$ with e and v' in its label ($ñ = \mathsf{Supp}(n, e)$). We produce as output the colour 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 $ñ$, 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 5.14. 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 5.5). 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 the mappings $\varphi_V(v, l) = v$ and $\varphi_E(e_l) = e$ define a locally bijective morphism of transition systems from $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ to $\mathcal{TS}$.
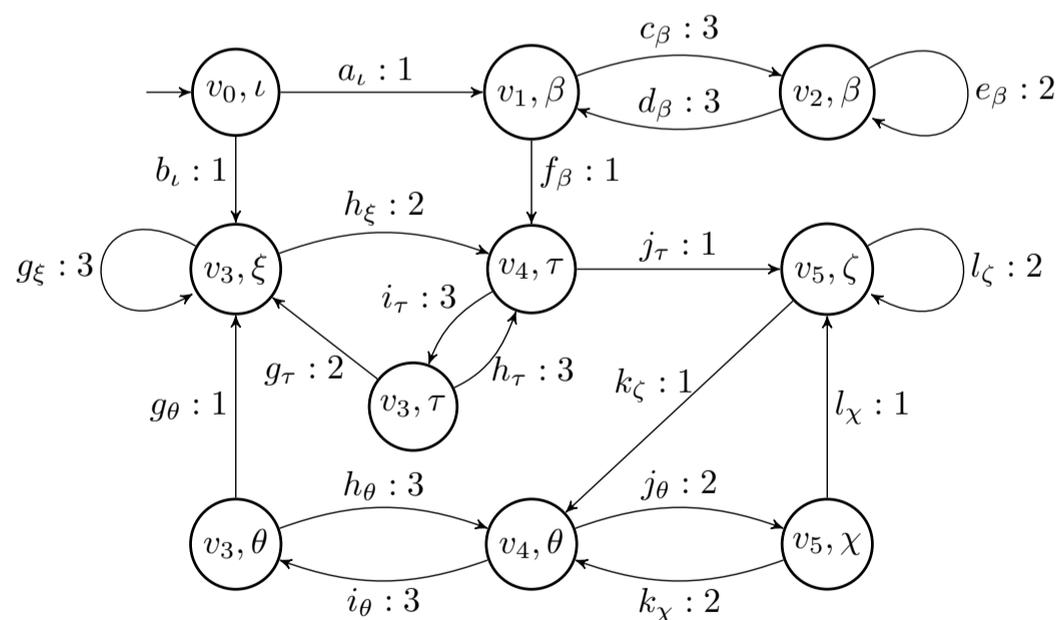

Figure 17. ACD-parity-transform $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ of the transition system $\mathcal{TS}$ from Figure 15.

Another example can be found in Figure 18. ◆

REMARK 5.15. The size of the ACD-parity-transformation of $\mathcal{TS}$ is:

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

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

REMARK 5.16. We remark that if $\mathcal{TS} = (G_{\mathcal{TS}}, \text{Acc}_{\mathcal{TS}})$ is already a parity TS, then the underlying graphs of $\text{ACD}_{\text{parity}}(\mathcal{TS})$ and $\mathcal{TS}$ are isomorphic. In fact, by Proposition 5.19, $\text{ACD}_{\text{parity}}(\mathcal{TS})$ and $\mathcal{TS}$ will also be isomorphic as transition systems. In this case, the construction of $\text{ACD}_{\text{parity}}(\mathcal{TS})$ boils down to the application of the procedure described by Carton and Maceiras [19].

REMARK 5.17. The ACD-parity-transform is oblivious to the labelling γ of the acceptance condition of $\mathcal{TS}$; the only information taken into account to define the graph of $\text{ACD}_{\text{parity}}(\mathcal{TS})$ 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 $f \colon \mathit{Cycles}(\mathcal{TS}) \to \{\text{Accept}, \text{Reject}\}$.

REMARK 5.18. The ZT-parity-automaton can be seen as a special case of the ACD-parity-transform, as $\mathcal{A}_{\mathcal{Z}_{\mathcal{F}}}^{\text{parity}}$ coincides with the DPA $\text{ACD}_{\text{parity}}(\mathcal{A})$, where $\mathcal{A}$ is the DMA with a single state recognising $\text{Muller}(\mathcal{F})$ (see Remark 5.7).

Correctness of the ACD-parity-transform.

PROPOSITION 5.19 (Correctness of the ACD-parity-transform). *Let $\mathcal{TS}$ be a (labelled) Muller TS and let $\text{ACD}_{\text{parity}}(\mathcal{TS})$ be its ACD-parity-transform. There is a locally bijective morphism of (labelled) transition systems $\varphi \colon \text{ACD}_{\text{parity}}(\mathcal{TS}) \to \mathcal{TS}$.*

The following lemma, analogous to Lemma 4.11 from Section 4.2, follows from the definition of the ACD-parity-transform.

LEMMA 5.20. *Let n be a node of $\text{AltTree}(\ell_i)$, let $\tilde{n}$ be an ancestor of n and let $e = v \to v'$ be an edge in ℓ_i. Then, $\text{Supp}(n, e)$ is a descendant of $\tilde{n}$ if and only if $e \in v(\tilde{n})$, and in this case, if $e_n = (v, n) \to (v', n')$ is an edge of $\text{ACD}_{\text{parity}}(\mathcal{TS})$, then n' is a descendant of $\tilde{n}$ too.*

PROOF OF PROPOSITION 5.19. 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 $e \in \text{Out}(v)$ of $\mathcal{TS}$, we have defined 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 4.10. We can assume w.l.o.g. (see Remarks 2.1 and 5.17) 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 assume 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 4.10), 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 ith 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 5.21. *For all $i \geq 0$, $n_i \geq n_\rho$ and $\tilde{n}_i \geq n_\rho$ (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 5.20 and induction.   ◆

CLAIM 5.22. *Let $n_{\rho,1}, \dots, n_{\rho,s}$ be an enumeration of $\mathsf{Children}_{\mathsf{AltTree}(\ell_{\max})}(n_\rho)$. It holds 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 4.13, from Proposition 4.10.   ◆

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 5.2 and 5.10.   ■

REMARK 5.23. 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:

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

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 5.9 and the correctness of the ZT-parity-automaton, the minimal

colour 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 output colour 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 colour 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 5.19 witnesses that $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ shares the same semantic properties as $\mathcal{TS}$. The next corollaries follow from Proposition 3.16 and Corollary 3.19 (and the fact that the choice of initial vertices in $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ is arbitrary).

COROLLARY 5.24. *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 5.25. *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.*

5.3 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 that from Section 4.3.

DEFINITION 5.26 (ACD-HD-Rabin-transform). Let $\mathcal{TS}$ be a Muller TS. For each vertex $v \in V$ we let $\eta_v \colon \mathsf{Leaves}(\mathcal{T}_v) \to \{1, \ldots, \mathsf{rbw}(\mathcal{T}_v)\}$ be a mapping satisfying Property $(\star)$ from Lemma 4.42.

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

Vertices. The set of vertices is

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

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

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

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 $x \in \{1, \ldots, \mathsf{rbw}(\mathcal{T}_v)\}$, 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:

— $\eta_v(l) = x, \eta_{v'}(l') = x'$,
— $l' = \text{Jump}_{\mathcal{T}_{v'}}(l, \text{Supp}(l, e))$,
— $n = \text{Supp}(l, e)$.

Rabin condition. $R = \{(G_n, R_n)\}_{n \in \text{Nodes}_\bigcirc(\mathcal{ACD}_{\mathcal{TS}})}$, where G_n and R_n are defined as follows: Let n be a round node, and let n' be any node in $\text{Nodes}(\mathcal{ACD}_{\mathcal{TS}})$,

$$\begin{cases} n' \in G_n & \text{if } n' = n, \\ n' \in R_n & \text{if } n' \neq n \text{ and } n \text{ is not an ancestor of } n'. \end{cases}$$

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

This construction generalises the ZT-HD-Rabin-automaton in the same way as the ACD-parity-transform generalises the ZT-parity-automaton. Intuitively, a run in $\text{ACD}_{\text{Rabin}}(\mathcal{TS})$ 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 5.27. The size of the ACD-HD-Rabin-transform of $\mathcal{TS}$ is:

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

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 4.48).

PROPOSITION 5.28 (Correctness of the ACD-HD-Rabin-transform). *Let $\mathcal{TS}$ be a (labelled) Muller TS and let $\text{ACD}_{\text{Rabin}}(\mathcal{TS})$ be its ACD-HD-Rabin-transform. There is an HD mapping of (labelled) transition systems $\varphi : \text{ACD}_{\text{Rabin}}(\mathcal{TS}) \to \mathcal{TS}$.*

The proof of the next two lemmas are completely analogous to those of Lemmas 4.49 and 4.50.

LEMMA 5.29. *Let $u = n_0 n_1 n_2 \cdots \in \text{Nodes}(\mathcal{ACD}_{\mathcal{TS}})^\omega$ be an infinite sequence of nodes of the ACD of $\mathcal{TS}$. The word u belongs to $\text{Rabin}(R)$, for $R = \{(G_n, R_n)\}_{n \in \text{Nodes}_\bigcirc(\mathcal{ACD}_{\mathcal{TS}})}$ the Rabin condition of $\text{ACD}_{\text{Rabin}}(\mathcal{TS})$, if and only if there is a unique minimal node for the ancestor relation in $\text{Inf}(u)$ and this minimal node is round.*

LEMMA 5.30. *There exists a morphism of transition systems $\varphi : \text{ACD}_{\text{parity}}(\mathcal{TS}) \to \text{ACD}_{\text{Rabin}}(\mathcal{TS})$.*

Using these lemmas we can prove Proposition 5.28.

PROOF OF PROPOSITION 5.28. 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 $\mathcal{L}(\mathcal{A}_{\mathcal{Z}_\mathcal{F}}^{\mathsf{Rabin}}) \subseteq \mathsf{Muller}_\Sigma(\mathcal{F})$ in Proposition 4.48 (by using Lemma 5.29).

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 5.30). Let $\tilde{\varphi} \colon \mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS}) \to \mathcal{TS}$ be the locally bijective morphism given by Proposition 5.19, and let $\hat{\varphi} \colon \mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS}) \to \mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})$ be the morphism given by Lemma 5.30. 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 $\tilde{\varphi}_{\mathcal{R}uns}^{-1}$. Composing this mapping with $\hat{\varphi}$, we obtain a way to simulate the runs from $\mathcal{TS}$ in $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})$:

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

This composition of mappings provides a sound resolver simulating φ. Formally, let (r_{Init}, r) be the resolver defined as follows. The choice of initial vertices $r_{\mathsf{Init}} \colon I \to I'$ is given by $r_{\mathsf{Init}}(v_0, x) = v_0$. 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 (r_{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. ∎

From Proposition 3.16 we obtain:

COROLLARY 5.31. *Let $\mathcal{A}$ be a Muller automaton and let $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{A})$ be its ACD-HD-Rabin-transform. Then, $\mathcal{L}(\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{A})) = \mathcal{L}(\mathcal{A})$. Moreover, $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{A})$ is history-deterministic if and only if $\mathcal{A}$ is history-deterministic.*

ACD-HD-Rabin-transform-for-games. In Section 2.1, we discussed some technical difficulties appearing when we wanted to define the composition of a game $\mathcal{G}$ and an HD automaton: as the output of such operation, we would like to obtain a game in which Eve always chooses the transitions taken in the automaton, even if it is Adam who makes a move in the game, which is not the case if $\mathcal{G}$ is an arbitrary game. Also, in Section 3.3 we had to introduce HD-for-games mappings in order to formalise correct transformations of games. A similar difficulty appears in the context of the ACD-HD-Rabin-transform; 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 current 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 in order to obtain an equivalent game 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 (we note that in Proposition 5.28 we did not claim that

there is an HD-for-games mapping $\varphi\colon \mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS}) \to \mathcal{TS}$). In order to obtain a transformation working for games, we need to slightly modify the definition of the ACD-HD-Rabin-transform.

For a Muller game $\mathcal{G}$ suitable for transformations, we define its ACD-HD-Rabin-transform-for-games, written $\mathsf{ACD}_{\mathsf{parity}}^{\mathsf{game}}(\mathcal{G})$. The idea is simply to take from Adam the power to update the $\mathcal{ACD}_{\mathcal{G}}$-component of vertices. The update of this information is delayed of one transition, so it is Eve who makes the choice of how to move in the ACD. To do this, we need to introduce some additional vertices controlled by Eve. The formal details of this construction and the proof of correctness can be found in Appendix B.

PROPOSITION 5.32 (Correctness of the ACD-HD-Rabin-transform-for-games). *Let $\mathcal{G}$ be a Muller game suitable for transformations, and let $\mathsf{ACD}_{\mathsf{parity}}^{\mathsf{game}}(\mathcal{G})$ be its ACD-HD-Rabin-transform-for-games. Then, there is an HD-for-games mapping $\varphi\colon \mathsf{ACD}_{\mathsf{parity}}^{\mathsf{game}}(\mathcal{G}) \to \mathcal{TS}$.*

COROLLARY 5.33. *Let $\mathcal{G}$ be a Muller game suitable for transformations, and let $\mathsf{ACD}_{\mathsf{parity}}^{\mathsf{game}}(\mathcal{G})$ be its ACD-HD-Rabin-transform-for-games. Then, Eve's full winning region in $\mathcal{G}$ is the projection of her full winning region in $\mathsf{ACD}_{\mathsf{parity}}^{\mathsf{game}}(\mathcal{G})$.*

5.4   Optimality of the ACD-transforms

We now state and prove the optimality of both the ACD-parity-transform (Theorems 5.34 and 5.35) and the ACD-HD-Rabin-transform (Theorem 5.36). The proofs of these results will use the optimality of the automata based on the Zielonka tree (c.f. Section 4) 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.

5.4.1   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, an hypothesis that can always be made without loss of generality. We recall that HD mappings are in particular locally bijective morphisms and HD-for-games mappings (c.f. Figure 7).

THEOREM 5.34. *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 colours as that of $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$.*

THEOREM 5.35. *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, $|\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})| \le |\widetilde{\mathcal{TS}}|$.*

THEOREM 5.36. *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, $|\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})| \leq |\widetilde{\mathcal{TS}}|$.*

We obtain an analogous optimality result 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 $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{parity}}(\mathcal{G})$ (see Appendix B for details).

COROLLARY 5.37. *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, $|\mathsf{ACD}^{\mathsf{game}}_{\mathsf{parity}}(\mathcal{G})| \leq 2|\widetilde{\mathcal{G}}|$.*

5.4.2 Discussion: Limits on the applicability of HD automata and preservation of minimality

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 [44] before finding HD automata that were actually smaller than equivalent deterministic ones [57]. As of today, we only know a handful of examples of ω-regular languages admitting succinct HD automata [1, 57, 24], 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 4.16 already sets the impossibility of the existence of small HD parity automata recognising Muller languages. Corollary 5.39 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 5.38. *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 5.39. *Let $\mathcal{A}$ be a history-deterministic parity automaton. Assume 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 5.35.

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 min-
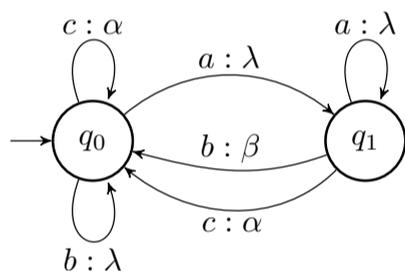
imal number of states for the language it recognises, is $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{A})$ minimal amongst DPAs recognising $\mathcal{L}(\mathcal{A})$?[11] The answer to this question is negative, as we show now.

PROPOSITION 5.40. *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.*
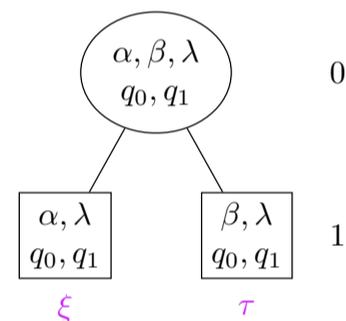
We consider the alphabet $\Sigma = \{a, b, c\}$ and the language

$$L = \{w \in \Sigma^\omega \mid c \in \mathrm{Inf}(w) \text{ and } w \text{ contains infinitely often the factor } ab\}.$$
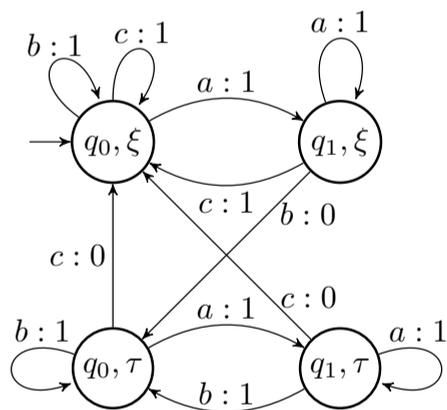
A minimal DMA for L is depicted in Figure 18a. Its minimality follows simply from the fact that, as L is not a Muller language $((abc)^\omega \in L$ but $(bac)^\omega \notin L$, c.f. Remark 2.10), a DMA with just one state cannot recognise L. In Figure 18 we show its alternating cycle decomposition and 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.

(a) A Muller automaton with acceptance set given by $\mathcal{F} = \{\{\alpha, \beta, \lambda\}\}$.

(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.

---

11   This question was left open as a conjecture in the conference version of this paper [23].

5.4.3 Optimality of the parity condition of $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$

We show next the proof of Theorem 5.34. To prove this result, we would like to use the Flower
Lemma 2.16, however, the statement of Theorem 5.34 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 $\mathcal{TS} = (G, \mathsf{Acc})$ and $\mathcal{TS}' = (G', \mathsf{Acc}')$ 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
$\mathsf{Acc} = (\gamma, \Gamma, \mathbb{W})$ and $\mathsf{Acc}' = (\gamma', \Gamma', \mathbb{W}')$. 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 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 5.41. *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 (r_{Init}, r) is a sound resolver simulating φ (where
$r_{\mathsf{Init}}(q_0') = q_0$ 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 (r_{Init}, r) be a sound resolver simulating φ. If ρ'
is an accepting run in $\mathcal{TS}'$, then $r_{\mathcal{R}uns}(\rho')$ is an accepting run over ρ' in $\mathcal{A}_\varphi$. ∎

We recall that $[\mathsf{min}_{\mathcal{TS}}, \mathsf{max}_{\mathcal{TS}}]$ are the colours 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 $\mathsf{min}_{\mathcal{TS}} = 0$ if
$\mathcal{ACD}_{\mathcal{TS}}$ is positive or equidistant, and that $\mathsf{min}_{\mathcal{TS}} = 1$ if $\mathcal{ACD}_{\mathcal{TS}}$ is negative.

LEMMA 5.42. *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 as the one used in the proof of Theorem 4.14. Let $n_1 \leq n_2 \leq \ldots \leq n_d$ be a branch of length d of AltTree(ℓ) (where n_1 is the root and n_d is a leaf of the tree). Let $v \in v_{\text{States}}(n_d)$ be a vertex appearing in the leaf. Then, the whole branch is contained in $\mathcal{T}_v$ (by Remark 5.4), that is, $v(n_i) \in \textit{Cycles}_{\mathcal{TS}}(v)$. Moreover, $v(n_1) \supsetneq v(n_2) \supsetneq \ldots v(n_d)$ is a chain that alternates accepting and rejecting cycles, so it is a d-flower that is positive if and only if $v(n_1) = \ell$ is an accepting cycle, that is, if AltTree(ℓ) is positive. ∎

LEMMA 5.43. *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,*
— *$\text{Weak}_{\max_{\mathcal{TS}}}$ if $\mathcal{ACD}_{\mathcal{TS}}$ is equidistant.*

PROOF. We consider the identity morphism $Id_{\mathcal{TS}} : \mathcal{TS} \to \mathcal{TS}$ and its automaton $\mathcal{A}_{Id_{\mathcal{TS}}}$, 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 2.16 and the fact that a tree AltTree(ℓ) $\in \mathcal{ACD}_{\mathcal{TS}}$ of height d provides a d-flower that is positive if ℓ is accepting and negative if ℓ is rejecting (Lemma 5.42). 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 5.34 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 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 5.44. 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 5.34 from the preceding lemmas.

PROOF OF THEOREM 5.34. We assume that we are in the first case of Remark 5.44 (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 (r_{Init}, r) be a sound resolver simulating it. We let $\tilde{v} = r_{\mathsf{Init}}(v)$ 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 assume that both $\widetilde{\mathcal{TS}}$ and $\mathcal{TS}$ have a single initial vertex. By Lemma 5.43 and Proposition 2.15, a parity history-deterministic automaton recognising $\mathcal{L}_{\mathcal{R}uns}(\mathcal{TS})$ uses at least $|[\min_{\mathcal{TS}}, \max_{\mathcal{TS}}]|$ colours. By Lemma 5.41, 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 $|[\min_{\mathcal{TS}}, \max_{\mathcal{TS}}]|$ colours. Since the acceptance condition of $\widetilde{\mathcal{TS}}$ is exactly the same as that of $\mathcal{A}_\varphi$, we can conclude. ∎

5.4.4   Optimality of the sizes of $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ and $\mathsf{ACD}_{\mathsf{Rabin}}(\mathcal{TS})$

We prove now Theorems 5.35 and 5.36.

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 5.45. Let $\mathcal{TS}$ and $\mathcal{TS}'$ be two transition systems, and let $(\gamma, \Gamma, \mathsf{Muller}_\Gamma(\mathcal{F}))$ 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 $\ell' \in \mathit{Cycles}_{v'}(\mathcal{TS}')$ 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 $\mathcal{A}_{(\varphi^{-1},v')} = (Q_{v'}, \mathit{Cycles}_{v'}(\mathcal{TS}'), Q_{v'}, 2_+^\Gamma, \delta, \mathsf{Muller}_{2_+^\Gamma}(\tilde{F}))$ over the input alphabet $\mathit{Cycles}_{\mathcal{TS}'}(v')$ 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 $\rho \in \mathcal{P}ath_{q_1}^{\mathsf{fin}}(\mathcal{TS})$ from q_1 to q_2 such that $\varphi(\rho) = \rho_{\ell'}$ producing as output the colours in $C \subseteq \Gamma$, that is $\gamma(\rho) = C$. If C is empty, this corresponds to an uncoloured edge $q_1 \xrightarrow{\ell':\varepsilon} q_2$. We remark that, since φ is assumed locally surjective, there is at least one such path ρ.

— $\{C_1, \ldots, C_k\} \in \tilde{F}$ 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 5.46. *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. Assume 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.

Assume now that L is a Rabin language represented by the Rabin pairs $\{(G_1, R_1), \dots, (G_r, R_r)\}$. We define a family of Rabin pairs $\tilde{R} = \{(\tilde{G}_1, \tilde{R}_1), \dots, (\tilde{G}_r, \tilde{R}_r)\}$ for $\tilde{L}$ as: $\{C_1, \dots, C_k\} \in \tilde{G}_i$ (resp. $\in \tilde{R}_i$) if $\cup_{i=1}^k C_i \in G_i$ (resp. $\in R_i$). It is immediate to see that $\tilde{L} = \mathsf{Rabin}_{2^\Gamma_+}(\tilde{R})$. ∎

LEMMA 5.47. *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 $\ell'_1 \ell'_2 \dots \in \mathit{Cycles}_v(\mathcal{TS})^\omega$ be a sequence of cycles accepted by $\mathcal{A}_{(\varphi^{-1}, v')}$. By prefix-independence of Muller languages we can assume 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 $\ell'_1 \ell'_2 \dots \in \mathsf{LocalMuller}_{v'}(\mathcal{TS})$ by definition of local Muller condition.

$\mathsf{LocalMuller}_{\mathcal{TS}'}(v') \subseteq \mathcal{L}(\mathcal{A}_{(\varphi^{-1}, v')})$ **and history-determinism:** Let $r_\varphi \colon E^* \times E' \to E$ be a sound resolver simulating φ. We will transfer the strategy given by r_φ to define a resolver $r_\mathcal{A} \colon \Delta^* \times \mathit{Cycles}_{v'}(\mathcal{TS}') \to \Delta$ for $\mathcal{A}_{(\varphi^{-1}, v')}$, where Δ is the set of transitions of the automaton. Let $\rho'_0 \in \mathit{Run}^{\mathsf{fin}}(\mathcal{TS}')$ be a finite run reaching v', and let $\rho_0 = r_{\varphi, \mathit{Runs}}(\rho'_0)$ 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 $e_1 e_2 \dots e_k \in \Delta^*$ and $\ell' \in \mathit{Cycles}_{v'}(\mathcal{TS}')$, we let

$$r_\mathcal{A}(e_1 e_2 \dots e_k, \ell') = r_\varphi(\rho'_0 \rho'_1 \dots \rho'_k, \rho_{\ell'}),^{[12]}$$

where $\rho'_j = \varphi(\mathsf{Unfold}(e_j))$ and $v' \xrightarrow{\rho_{\ell'}} v'$ 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 $e_1 e_2 \dots \in \Delta^\omega$ is the run induced by $r_\mathcal{A}$ over $\ell'_1 \ell'_2 \dots \in \mathit{Cycles}_{v'}(\mathcal{TS}')^\omega$,
>
> then $\rho_0 \mathsf{Unfold}(e_1 e_2 \dots) = r_{\varphi, \mathit{Runs}}(\rho'_0 \rho'_1 \rho'_2 \dots)$.

---

[12] 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 $r_\mathcal{A}(e_1 e_2 \dots e_k, \ell')$ is a suffix of $r_{\varphi, \mathit{Runs}}(\rho'_0 \rho'_1 \dots \rho'_k \rho_{\ell'})$.

This gives us:

$$\bigcup \mathrm{Inf}(\ell'_1, \ell'_2, \dots) \text{ is accepting cycle in } \mathcal{TS}' \iff \rho'_0\rho'_1\rho'_2\dots \text{ is accepting run in } \mathcal{TS}' \implies$$
$$\implies \rho_0\mathrm{Unfold}(e_1e_2\dots) \text{ accepting run in } \mathcal{TS} \iff e_1e_2\dots \text{ accepting run in } \mathcal{A}_{(\varphi^{-1},v')}.$$

Which allows us to conclude that the $\mathcal{A}_{(\varphi^{-1},v')}$ recognises $\mathrm{LocalMuller}_{v'}(\mathcal{TS})$ and that $r_\mathcal{A}$ is a sound resolver. ∎

COROLLARY 5.48. *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,*

$$|\varphi^{-1}(v)| \geq |\mathrm{Leaves}(\mathcal{Z}_{\mathrm{LocalMuller}_{\mathcal{TS}}(v)})| = |\mathrm{Leaves}(\mathcal{T}_v)|.$$

PROOF. By Lemma 5.47, the automaton $\mathcal{A}_{(\varphi^{-1},v)}$ is a history-deterministic automaton recognising $\mathrm{LocalMuller}_v(\mathcal{TS})$ of size $|\varphi^{-1}(v)|$, and by Lemma 5.46, it is a parity automaton. The optimality of the ZT-parity-automaton (Theorem 4.15) gives us the first inequality. The second equality follows from the fact that $\mathcal{T}_v$ is the Zielonka tree of $\mathrm{LocalMuller}_{\mathcal{TS}}(v)$ (Lemma 5.9). ∎

The next corollary admits an identical proof, using the optimality of the ZT-HD-Rabin-automaton (Theorem 4.51).

COROLLARY 5.49. *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,*

$$|\varphi^{-1}(v)| \geq |\mathrm{rbw}(\mathcal{Z}_{\mathrm{LocalMuller}_{\mathcal{TS}}(v)})| = |\mathrm{rbw}(\mathcal{T}_v)|.$$

Theorems 5.35 and 5.36 follow from these two corollaries, the formulas for the size of the ACD-transforms (Remarks 5.15 and 5.27) 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 3.5).

6. Corollaries

In this section, we discuss some further applications of the Zielonka tree and the alternating cycle decomposition. In Section 6.1, we use the insights gained from the ACD to conduct a comprehensive study of typeness results for deterministic Muller automata (that is, when can we relabel a DMA with an equivalent and simpler acceptance condition). In Section 6.2 we present a normal form for parity transition systems and prove the main properties exhibited by TS in this form. In Section 6.3, we provide a polynomial-time algorithm minimising DPA recognising Muller languages.

6.1 Typeness results

As we have seen, there are many different types of acceptance conditions for ω-regular automata. An important question is the following:

Question: Given a Muller automaton $\mathcal{A}$, can we define a simpler acceptance condition over 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 [55, 56], who showed how to determine if a DMA can be re-labelled with an equivalent Büchi condition. Their work was generalised to parity automata by Boker, Kupferman and Steinitz [11], and related questions about typeness were studied for non-deterministic automata by Kupferman, Morgenstern and Murano [59], and for history-deterministic automata by Boker, Kupferman and Skrzypczak [10].

In this section, we provide new general characterisations of typeness for Muller transition systems. The main contributions of this section appear in Propositions 6.9, 6.10 and 6.11, which 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 6.9 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 these results was already known [63], 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 [11, 55, 56]. We also show how to use the ACD to determine the parity index of the language recognised by a DMA (Proposition 6.13), which can be seen as a simplification of the results from [56, Section 3.2]. Further results concerning generalised Büchi languages and weak automata can be found in Appendix A.

6.1.1 Typeness for Muller languages

We first present some results proven by Zielonka [94, 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. A study of further types of Muller languages can be found in Appendix A.

We do not include the proofs of the results of this section in the main body of the paper, as they are known results [94, Section 5] and they are special cases of the proofs in Section 6.1.2. Nevertheless, we include them in Appendix E.

We first introduce some definitions. The terminology will be justified by the upcoming results.

DEFINITION 6.1. Let T be a tree 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.

PROPOSITION 6.2. *Let $\mathcal{F} \subseteq 2^\Gamma_+$ be a family of non-empty subsets. The following conditions are equivalent:*

1. *$Muller_\Gamma(\mathcal{F})$ is a Rabin language.*

2. *$2^\Gamma_+ \setminus \mathcal{F}$ 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.*

PROPOSITION 6.3. *Let $\mathcal{F} \subseteq 2^\Gamma_+$ be a family of non-empty subsets. The following conditions are equivalent:*

1. *$Muller_\Gamma(\mathcal{F})$ is a Streett language.*

2. *The family $\mathcal{F}$ is closed under union.*

3. *$\mathcal{Z}_\mathcal{F}$ has Streett shape.*

PROPOSITION 6.4. *Let $\mathcal{F} \subseteq 2^\Gamma_+$ be a family of non-empty subsets. The following conditions are equivalent:*

1. *$Muller_\Gamma(\mathcal{F})$ is a parity language.*

2. *Both $\mathcal{F}$ and $2^\Gamma_+ \setminus \mathcal{F}$ 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.*

Moreover, if some of these conditions is satisfied, $Muller_\Gamma(\mathcal{F})$ is a $[\min_\mathcal{F}, \max_\mathcal{F}]$-parity language.

COROLLARY 6.5. *A Muller language $L \subseteq \Gamma^\omega$ is a parity language if and only if it is both a Rabin and a Streett language.*

6.1.2   Typeness for Muller transition systems and deterministic automata

We start this subsection by introducing the necessary definitions about equivalence of acceptance conditions and typeness. Then, we state and prove our main contributions concerning typeness of transition systems.

Equivalence of acceptance conditions and typeness. Let $\mathcal{TS}_1 = (G, \mathrm{Acc}_1)$ and $\mathcal{TS}_2 = (G, \mathrm{Acc}_2)$ be two transitions systems over the same underlying graph G, with acceptance conditions $\mathrm{Acc}_i = (\gamma_i, \Gamma_i, \mathbb{W}_i)$, 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, $\gamma_1(\rho) \in \mathbb{W}_1 \iff \gamma_2(\rho) \in \mathbb{W}_2$.

We write $\mathcal{TS}_1 \simeq \mathcal{TS}_2$ if $\mathcal{TS}_1$ and $\mathcal{TS}_2$ are isomorphic. We recall that two transition systems are isomorphic if there is a morphism of transition systems $\varphi \colon \mathcal{TS}_1 \to \mathcal{TS}_2$ whose inverse is also a morphism, that is, φ and φ^{-1} preserve the acceptance of runs.

REMARK 6.6. If $\varphi \colon \mathcal{TS}_1 \to \mathcal{TS}_2$ is an isomorphism, then $(\gamma_2 \circ \varphi, \Gamma_2, \mathbb{W}_2)$ is an acceptance condition over the underlying graph of $\mathcal{TS}_1$ that is equivalent to $(\gamma_1, \Gamma_1, \mathbb{W}_1)$ 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 $\mathcal{TS}_1 = (G, \mathsf{Acc}_1)$ and $\mathcal{TS}_2 = (G, \mathsf{Acc}_2)$.

For X one of types of languages defined in Section 2.2 (Büchi, parity, Muller, etc...), we say that a transition system $\mathcal{TS}$ is *X type* if there exists an isomorphic transition system $\mathcal{TS}' \simeq \mathcal{TS}$ using an X acceptance condition. We note that, by the previous remark, in that case an X acceptance condition can be defined directly over the underlying graph of $\mathcal{TS}$.

We remark that, given a pointed graph G (whose states are accessible), the equivalence classes of Muller acceptance conditions for the relation $\simeq_G$ are given exactly by the mappings $f \colon \mathit{Cycles}(G) \to \{\mathsf{Accept}, \mathsf{Reject}\}$.

The ACD determines the type of transition systems.

DEFINITION 6.7. 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).

REMARK 6.8. $\mathcal{ACD}_{\mathcal{TS}}$ is a parity ACD if and only if it is both a Rabin and a Streett ACD.

PROPOSITION 6.9. *Let* $\mathcal{TS} = (G_{\mathcal{TS}}, \mathsf{Acc}_{\mathcal{TS}})$ *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.*

PROOF. (**1** $\Rightarrow$ **2**) Let $\mathsf{Acc}_R = (\gamma, \Gamma, \mathsf{Rabin}(R)$ be the Rabin acceptance condition equivalent to $\mathsf{Acc}_{\mathcal{TS}}$, and let $R = (G_1, R_1), \ldots, (G_r, R_r)$ 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 (G_j, R_j), so for all edges $e \in \ell_1 \cup \ell_2$, $\gamma(e) \notin R_j$,

and there is some $e_0 \in \ell_1 \cup \ell_2$ such that $\gamma(e_0) \in G_j$. If e_0 belongs to ℓ_1, then ℓ_1 is accepted by the Rabin pair (G_j, R_j), and if $e_0 \in \ell_2$, then ℓ_2 is accepted by it.

(2 ⟹ 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 $v(n_1)$ and $v(n_2)$ are rejecting cycles over v, but their union is an accepting cycle (by Remark 5.2).

(3 ⟹ 1) We observe that $\mathcal{ACD}_{\mathcal{TS}}$ is a Rabin ACD if and only if $\mathrm{rbw}(\mathcal{T}_v) = 1$ 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 $\varphi \colon \mathrm{ACD}_{\mathrm{Rabin}}(\mathcal{TS}) \to \mathcal{TS}$ given by $\varphi_V(v, x) = v$, $\varphi_E(e, l) = e$ defined in the proof of Proposition 5.28 is an isomorphism, and $\mathcal{TS}$ uses a Rabin acceptance condition. ■

PROPOSITION 6.10. *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.*

PROOF. Implications (1 ⟹ 2) and (2 ⟹ 3) are analogous to those from Proposition 6.9.

(3 ⟹ 1) We consider the transition system $\overline{\mathcal{TS}}$ obtained by complementing the acceptance set of $\mathrm{Acc}_{\mathcal{TS}}$. By Remark 5.6, 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 ACD, and by applying the previous proposition we can define a Rabin condition $\mathrm{Acc}_R = (\gamma, \Gamma, \mathrm{Rabin}_\Gamma(R))$ such that the transition system $(G_{\mathcal{TS}}, \mathrm{Acc}_R)$ is isomorphic to $\overline{\mathcal{TS}}$. Since $\mathrm{Streett}_\Gamma(R)$ is the complement language of $\mathrm{Rabin}_\Gamma(R)$, we obtain that $\mathrm{Acc}_S = (\gamma, \Gamma, \mathrm{Streett}_\Gamma(R))$ is a Streett acceptance condition equivalent to $\mathrm{Acc}_{\mathcal{TS}}$ over $G_{\mathcal{TS}}$. ■

PROPOSITION 6.11. *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.*

Moreover, if some condition is satisfied, $\mathcal{TS}$ is $[0, d-1]$ (resp. $[1, d]$)-parity type if and only if $\mathcal{ACD}_{\mathcal{TS}}$ is a $[0, d-1]$(resp. $[1, d]$)-parity ACD.

PROOF. **(1 ⟹ 2)** Proven in Lemma 4.21.

---

[13]   This property was introduced by Le Saëc under the name *cyclically closed automata* [**82**]. We point out that the "if" direction of the result stated in [**82, 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 6.13 for a correct characterisation.

(2 ⇒ 3) Admits an analogous proof to the corresponding implication in Proposition 6.9.

(3 ⇒ 1) By definition, $\mathcal{ACD}_{\mathcal{TS}}$ is a parity ACD if and only if 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 5.19 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 $[0, d-1]$ (resp. $[1, d]$)-parity condition if $\mathcal{ACD}_{\mathcal{TS}}$ is a $[0, d-1]$ (resp. $[1, d]$)-parity ACD. If $\mathcal{ACD}_{\mathcal{TS}}$ is not a $[0, d-1]$(resp. $[1, d]$)-parity ACD, then the number of colours cannot be reduced by the optimality of the number of colours of the ACD-parity-transform (Theorem 5.34). ■

COROLLARY 6.12. *A Muller transition system is parity type if and only if it is both Rabin and Streett type.*

The ACD and the parity index of ω-regular languages.

PROPOSITION 6.13. *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.*

PROOF. We prove the right-to-left implication for the case Weak$_d$. Assume 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 colours in $[0, d]$. In order to obtain a DPA for $\mathcal{L}(\mathcal{A})$ with colours 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 colours in $[1, d+1]$.

To prove that no DPA recognising L uses less than d colours, it suffices to use the Flower Lemma 2.16 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 5.42).

This is indeed a complete characterisation, since for any ACD there is a minimal d such that $\mathcal{ACD}_{\mathcal{A}}$ lies in one and only one of the classes specified in the statement of the proposition. ∎

PROPOSITION 6.14. *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 $(\gamma, \Gamma, \mathsf{Muller}_\Gamma(\mathcal{F}))$. By Proposition 6.13, 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: $\gamma_{\mathcal{ACD}}(n) = \gamma(\nu(n))$. 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 4.2. If $\mathcal{A}$ is a history-deterministic Muller automaton, it admits a sound resolver implemented by a finite memory structure $(\mathcal{M}, \sigma)$ (Lemma 4.24). Then, the composition $\mathcal{A} \triangleleft_\sigma \mathcal{M}$ 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 [19]), is refined and proven in Appendix A (Corollary A.16).

PROPOSITION 6.15. *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. $[1, d]$). Then, $\mathcal{A}$ is $[0, d-1]$ (resp. $[1, d]$)-parity type.*

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 [10, Theorems 10 and 13]. We do not know whether the result holds for languages of arbitrary parity index.

Typeness for deterministic automata. Two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $\mathcal{A}_1 \simeq \mathcal{A}_2$ recognise the same language: $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$. However, the converse only holds for deterministic automata.

LEMMA 6.16. *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:

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

COROLLARY 6.17 (First proven in [11, Theorem 7]). *Let $G_\mathcal{A}$ be the underlying graph of a deterministic automaton. Then, there are 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 [11].

PROPOSITION 6.18 (First proven in [55, Theorem 15]). *Let $\mathcal{A}$ be a deterministic Rabin (resp. Streett) automaton, and assume that $\mathcal{L}(\mathcal{A})$ can be recognised by a deterministic Büchi (resp. coBüchi) automaton; that is, the parity index of $\mathcal{L}(\mathcal{A})$ is 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 assume 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 6.13. As $\mathcal{A}$ is a Rabin automaton, its ACD has Rabin shape (Proposition 6.9), 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 6.11, $\mathcal{A}$ is Büchi type. $\qquad \blacksquare$

6.2 A normal form for parity transition systems

In this section, we propose a definition of a normal form of parity automata. This is exactly the form of automata resulting by applying the procedure defined by Carton and Maceiras [19], or, equivalently, of automata resulting from the ACD-parity-transform (Corollary 6.24). These automata satisfy that they are parity-index-tight, that is, their acceptance condition uses the minimal possible number of colours. But they offer some further convenient properties, stated in Propositions 6.25 and 6.27, which make them particularly well-suited for reasoning about deterministic parity automata.

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 [1, 35, 57], positionality of languages defined by deterministic Büchi automata [15] or learning of DPAs [6]. The normalisation of transition systems also facilitates solving parity

games in practice [41]. 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. $\mathcal{TS}$ use the smallest possible colour in each of its transitions (Definition 6.20).
2. $\mathcal{TS} = \mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ (Corollary 6.24).
3. Paths in $\mathcal{TS}$ producing colour $d > 0$ can be closed into a cycle producing d' as minimal colour, for all $d' \leq d$ (Theorem 6.28).

REMARK 6.19. If $\mathsf{Acc} = (\gamma, [d_{\min}, d_{\max}], \mathsf{parity})$ is a parity acceptance condition over a pointed graph G, 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 $(\gamma', [d_{\min} - \chi, d_{\max} - \chi], \mathsf{parity})$ defined as $\gamma'(e) = \gamma(e) - \chi$ is equivalent to Acc over G.

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 colours, 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 colour used by a parity acceptance condition using an optimal number of colours is 1.

DEFINITION 6.20 (Normal form). Let $\mathcal{TS} = (G_{\mathcal{TS}}, \mathsf{Acc}_{\mathcal{TS}})$ be a parity transition system using a colouring function γ. If $\mathcal{TS}$ is not negative, we say that $\mathcal{TS}$ is in *normal form* if any other parity acceptance condition equivalent to $\mathsf{Acc}_{\mathcal{TS}}$ over $G_{\mathcal{TS}}$ using a colouring function γ' satisfies that for every edge e:

$$\gamma(e) \leq \gamma'(e).$$

If $\mathcal{TS}$ is negative, we say that it is in *normal form* if any other equivalent colouring γ' not using colour 0 satisfies that for any edge e:

$$1 \leq \gamma(e) \leq \gamma'(e).$$

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 6.21. Parity transition systems from Figures 3, 10, 17 and 18 are all in normal form. Parity automata appearing in Figures 10 and 17 are negative (the minimal colour used by an optimal acceptance condition is odd), whereas parity automata in Figure 18 are not.

On the other hand, the automaton from Figure 1 is not in normal form (even if it uses an optimal number of colours). We can put it in normal form by assigning colour 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 6.22. *Let $\mathcal{TS} = (G_{\mathcal{TS}}, \mathrm{Acc}_{\mathcal{TS}})$ be a parity transition system with a colouring function γ. 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 6.22, we prove a useful technical lemma.

LEMMA 6.23. *Let $\mathcal{TS}$ be a parity transition system with colouring function γ. If $\ell_1 \supsetneq \ell_2 \supsetneq \cdots \supsetneq \ell_k$ is a positive (resp. negative) k-flower of $\mathcal{TS}$, then $\min \gamma(\ell_k) \geq k - 1$ (resp. $\min \gamma(\ell_k) \geq k$).*

PROOF. We show the result for negative flowers. Let $d_i = \min \gamma(\ell_i)$. 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 6.22. 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 $\gamma_{\mathcal{ACD}}$. 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 6.11). We let $n_0 \leq n_1 \leq \ldots \leq n_r$ 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, $\gamma_{\mathcal{ACD}}(e) = p_{\mathcal{ACD}}(e) = k$. Also, $\nu(n_0) \leq \nu(n_1) \leq \ldots \leq \nu(n_k)$ is a positive $k + 1$-flower (by Lemma 5.42). Lemma 6.23 implies then that any equivalent parity condition using a colouring function γ' verifies $\gamma'(e) \geq \gamma_{\mathcal{ACD}}(e) = k$.   ■

COROLLARY 6.24. *The ACD-parity-transform $\mathrm{ACD}_{\mathrm{parity}}(\mathcal{TS})$ of any Muller transition system $\mathcal{TS}$ is in normal form.*

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. Theorem 6.28 states that these properties characterise the normal form.

PROPOSITION 6.25. *Let $\mathcal{TS}$ be a parity transition system in normal form. If there is a path $v \rightsquigarrow v'$ producing d as minimal colour, 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 colour strictly smaller than d.*

PROOF. By Proposition 6.22, we know that the colouring of $\mathcal{TS}$ is the one given by its ACD-transform, that we note $\gamma_{\mathcal{ACD}}$. If v and v' are in different SCCs the result is trivial. Let v and v' be in the same SCC, that we suppose 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 $\min \gamma_{\mathcal{ACD}}(\rho) = d$ as minimal colour. 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 $\gamma_{\mathcal{ACD}}(e)$ equals the depth of the deepest node containing e. In particular, if $e \in v(n)$ for some node e, $\gamma_{\mathcal{ACD}}(e) \geq \mathsf{Depth}(n)$. Our objective is to show that a similar result holds for the path ρ as a set of edges:

CLAIM 6.26. *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, $N_\rho = \{n \in \mathsf{Nodes}(\mathcal{ACD}_{\mathcal{TS}}) \mid \{e_1, \dots, e_k\} \subseteq v(n)\}$. Then, $\min(\gamma_{\mathcal{ACD}}(\rho))$ equals the depth of a node of maximal depth of N_ρ.*[14]

This claim allows us to conclude. Indeed, let n be a node of maximal depth of N_ρ, verifying $\mathsf{Depth}(n) = d$. Then, $v(n)$ is a cycle containing the vertices v and v', and for all the edges $e \in v(n)$, $\gamma_{\mathcal{ACD}}(e) \geq \mathsf{Depth}(n) = d$. This provides the desired path from v' to v.

Proof of Claim 6.26. 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, $\gamma_{\mathcal{ACD}}(e) \geq \mathsf{Depth}(n)$. Suppose by contradiction that $\gamma_{\mathcal{ACD}}(\rho) > \mathsf{Depth}(n)$. Then, each edge e_i of ρ would appear in some strict descendant n_i of n (we can assume that n_i is a child of n). Then, $v(n_1), \dots, v(n_k)$ would be cycles such that $v(n_i)$ and $v(n_{i+1})$ 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 $v(n)$ is accepting if and only if each of the $v(n_i)$ is rejecting (see Lemma 4.21). ◆

This completes the proof of Proposition 6.25. ∎

PROPOSITION 6.27 (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 $\ell \in \mathsf{Cycles}_v(\mathcal{TS})$ be a cycle over v and let d_ℓ be the minimal colour appearing in it.*

— *If $\mathcal{TS}$ is not negative, for each $x \in [0, d_\ell]$ (resp. $x \in [1, d_\ell]$) there is a cycle $\ell_x \in \mathsf{Cycles}_v(\mathcal{TS})$ producing x as minimal colour.*

---

[14] 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 negative, for each $x \in [2, d_\ell]$ (resp. $x \in [1, d_\ell]$) there is a cycle $\ell_x \in Cycles_v(\mathcal{TS})$ producing x as minimal colour.*

PROOF. We do the proof for the case in which $\mathcal{TS}$ is not negative and v belongs to an accepting SCC. By Proposition 6.22, the colouring of $\mathcal{TS}$ is the one given by its ACD-transform, noted $\gamma_{\mathcal{ACD}}$. Consider the local subtree at v, $\mathcal{T}_v$, consisting in a single branch, as it has parity shape (Proposition 6.11). Let $n_0 \leq \ldots \leq n_k$ be that branch, and let n_i be the deepest node such that $\ell \subseteq v(n_i)$. We remark that, by definition of $\gamma_{\mathcal{ACD}}$, $d_\ell = \mathsf{Depth}(n_i) = i$. The desired cycles are obtained by taking $\ell_x = v(n_x)$, for $x \in [0, d_\ell]$. ∎

The next theorem states a simple characterisation of transition systems in normal form. It provides a useful tool to show normality of parity TS 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 an SCC of a parity TS is *positive* if the minimal colour appearing on it is even, and that it is *negative* if this minimal colour is odd.

THEOREM 6.28. *A non-negative parity transition system is in normal form if and only if:*
— *transitions changing of SCCs are coloured 0, and*
— *if v and v' belong to a same positive (resp. negative) SCC and there is a transition $v \xrightarrow{d} v'$ producing colour $d > 0$ (resp. $d > 1$), then there are two paths $v' \rightsquigarrow v$ producing as minimal colour 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 γ as colouring function. Let $e = v \xrightarrow{d} v'$ be an edge with $\gamma(e) = d$. We will show that for any other equivalent colouring γ', we have $\gamma'(e) \geq d$. 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 colour, respectively. Cycle ℓ_{d-1} can be decomposed in $v \rightarrow v' \rightsquigarrow v_1 \xrightarrow{d-1} v'_1 \rightsquigarrow v$. 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 colour, which can be merged with ℓ_{d-1} to produce a cycle ℓ_{d-2} over v producing $d - 2$ as minimal colour. 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 colour. Taking $\ell'_i = \cup^d_{j=i} \ell_i$, we obtain a positive $(d + 1)$-flower $\ell'_0 \supsetneq \ell'_1 \supsetneq \cdots \supsetneq \ell'_d$, so by Lemma 6.23 we conclude that $\gamma'(e) \geq d$. ∎

Parity index from automata in normal form. The next definition constitutes a syntactic version of the parity index, defined at the level of parity transition systems. The following

results establish the tight relation between the semantic notion of parity index and its syntactic counterpart, and state that the parity index of a language can be directly read from a DPA in normal form.

DEFINITION 6.29. 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 colours as $\mathrm{Acc}_{\mathcal{TS}}$.

We have shown in Corollary 6.24 that the ACD-parity-transform is always in normal form. Therefore, the optimality properties of the colouring of $\mathrm{ACD}_{\mathrm{parity}}(\mathcal{TS})$ (Theorem 5.34) transfer to parity transition systems in normal form.

COROLLARY 6.30. *A parity transition system in normal form is parity-index-tight.*

Moreover, the parity index of an ω-regular language can be read from any DPA in normal form recognising it.

COROLLARY 6.31. *Let $\mathcal{A}$ be a deterministic parity automaton in normal form such that all its states are accessible. If $\mathcal{A}$ uses colours in $[0, d-1]$ (resp. $[1, d]$), then the parity index of $\mathcal{L}(\mathcal{A})$ is Weak_{d-1} or $[0, d-1]$ (resp. Weak_{d-1} or $[1, d]$).*

We prove this result in Appendix A (Corollary A.15), and we provide there a refined characterisation by using generalised weak automata.

6.3   Minimisation of deterministic parity automata recognising Muller languages

The minimisation of ω-automata is a fundamental problem of an intriguing complexity. 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 [86]. However, the reduction of NP-hardness does not generalise to automata with edge-based acceptance. A surprising positive result was obtained in 2019 by Abu Radi and Kupferman: we can minimise in polynomial time HD coBüchi automata using transition-based acceptance [1]. Schewe showed that the minimisation was again NP-hard for HD automata with state-based acceptance [87]. To the best of our knowledge, the only existing hardness result applying to transition-based automata is Casares' result about the NP-completeness of the minimisation of deterministic Rabin automata [21]. In fact, in [21] a stronger result is proven: it is NP-hard to minimise deterministic Rabin automata recognising Muller languages.

In this section, we provide a polynomial-time algorithm for the minimisation of DPA recognising Muller languages (with acceptance condition over transitions). By Proposition 4.10 and Theorem 4.15, we know that a minimal (history-)deterministic parity automaton recognising a Muller language $L = \mathrm{Muller}_\Sigma(\mathcal{F})$ 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.

THEOREM 6.32. *Let $\mathcal{A}$ be a DPA recognising a Muller language $L = \text{Muller}_\Sigma(\mathcal{F})$. We can find a minimal deterministic (resp. history-deterministic) parity automaton recognising L in polynomial time in the size of the representation of $\mathcal{A}$.*[15]

Description of the algorithm. Let $\mathcal{A} = (Q, \Sigma, q_0, \Gamma, \Delta, \text{parity})$ be a DPA recognising $L = \text{Muller}_\Sigma(\mathcal{F})$. We outline a recursive algorithm building $\mathcal{Z}_\mathcal{F} = (N, \leq, \nu)$ 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 1, which we disclose next.

We assume 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 $\text{Muller}_{\nu(n)}(\mathcal{F}|_{\nu(n)})$ (see also Lemma 4.20). 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 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 11). In order to find them, we remove the transitions using the minimal colour 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 in it. Then, $\Sigma_\mathcal{S} \notin \mathcal{F}$ if and only if the minimal output colour in $\mathcal{S}$ is odd (see Lemma 6.33 below). In this case, we add $\Sigma_\mathcal{S}$ to altSets. On the contrary, we remove the minimal (even) colour 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 1. We use the following notations:

— `Letters`$(\mathcal{S})$ is the set of input letters appearing in $\mathcal{S}$,

— `MinColour`$(\mathcal{S})$ is the minimal output colour 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.

— `MaxInclusion`(lst) returns the list of the maximal subsets in lst.

---

[15]   We can assume that the representation of $\mathcal{A}$ has size polynomial in $|Q| + |\Sigma|$, where Q and Σ are the set of states and the input alphabet of $\mathcal{A}$. Indeed, as $\mathcal{A}$ is deterministic the number of transitions is at most $|Q| \cdot |\Sigma|$, and we can assume that $\mathcal{A}$ has no more output colours than transitions.

```
Input:   A strongly connected automaton 𝒜 over Σ; ℒ(𝒜) = Muller(ℱ)
Output:  Maximal subsets Σ₁,...,Σₖ ⊆ Σ such that Σᵢ ∈ ℱ ⟺ Σ ∉ ℱ
```
1: $d \leftarrow$ MinColour($\mathcal{A}$)

2: $\mathcal{A}_{>d} \leftarrow$ restriction of $\mathcal{A}$ to transitions $\Delta_{>d} = \{q \xrightarrow{a:x} q' \in \Delta \mid x > d\}$

3: $\langle \mathcal{S}_1, \ldots, \mathcal{S}_r \rangle \leftarrow$ SCC-Decomposition($\mathcal{A}_{>d}$)

4: altSets $\leftarrow \{\}$

5: **for** $i = 1, \ldots, r$ **do**

6:      **if** MinColour($\mathcal{S}_i$) is odd if and only if d is even **then**

7:          altSets $\leftarrow$ altSets $\cup$ {Letters($\mathcal{S}_i$)}

8:      **else**

9:          altSets $\leftarrow$ altSets $\cup$ AlternatingSets($\mathcal{S}_i$)

10: maxAltSets $\leftarrow$ MaxInclusion(altSets)

11: **return** maxAltSets

Algorithm 1. AlternatingSets($\mathcal{A}$): Computing the children of a node.

Correctness of the algorithm.

Let n be a node of the Zielonka tree of $\mathcal{F}$ labelled with $v(n)$, and let $\mathcal{A}_n$ be an accessible subautomaton of $\mathcal{A}$ over $v(n)$ recognising $\mathsf{Muller}_{v(n)}(\mathcal{F}|_{v(n)})$. 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 assume without loss of generality that $v(n) \in \mathcal{F}$ and therefore the minimal colour 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 colour 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 6.33. *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 in ℓ, and let d_ℓ be the minimal colour 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, `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 6.34. *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 colour 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 6.33.   ◆

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 colour is even, and we can conclude by induction.

Complexity analysis. We will show that the proposed algorithm works in time $O(|Q|^3|\Sigma|^2|\Gamma|)$, 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 `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 $O(|Q||\Sigma|)$ [91]. Therefore, the children of a node of the Zielonka tree can be computed in $O(|Q|^2|\Sigma|^2)$.

We perform this operation for each node of the Zielonka tree. By the optimality of the ZT-parity-automaton (Theorems 4.14 and 4.15), 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 time $O(|Q|^3|\Sigma|^2|\Gamma|)$.

REMARK 6.35 (State-based automata). The acceptance condition of the parity automaton obtained from the Zielonka tree appears naturally over the transitions of the automaton. In order to make it a state-based automaton, we would need to add one state per colour it uses. It turns out that, in this specific case, this is optimal, and the state-based parity automaton we obtain is minimal. Therefore, we can also minimise in polynomial time state-based parity automata recognising Muller languages. However, it is no longer possible to obtain optimal transformations towards state-based parity automata based on the ACD (see [25, Section 5.3] and [22, Section I.8] for further details).

7. Conclusion

In this work, we have carried out an extensive study of transformations of automata and games that use Muller acceptance conditions. We have proposed different types of morphisms to formalise the idea of valid transformations of transition systems, which distil the central features of existing transformations. Our main contribution resides in the introduction of a new structure, the alternating cycle decomposition, which is a succinct representation of the *alternating chains of loops* of a Muller automaton – in the sense of Wagner [93] – and provides the necessary information to understand the interplay between its acceptance condition and its underlying graph.

Optimal and practical transformations of automata. We have presented a transformation that, given a deterministic Muller automaton, provides an equivalent deterministic parity automaton, and another that provides an equivalent history-deterministic Rabin automaton. These transformations are optimal in a strong sense; the obtained automata have a minimal number of states amongst those which accept a history-deterministic mapping to the original Muller automaton. The first of these transformations has been implemented in the open-source tools Spot 2.10 [2] and Owl 21.0 [52], and it has been shown to perform extremely well in practice [25], as the natural definition of the ACD provides a fairly efficient way to compute the transformation, while its optimality guarantees to produce automata as small as possible.

Understanding the limitations of history-deterministic automata. As a corollary of our results, we have obtained that minimal deterministic and history-deterministic parity automata recognising Muller languages have the same size (Corollary 4.16). Moreover, we have shown that HD parity automata that are strictly smaller than equivalent deterministic ones cannot come from a deterministic Muller automaton (Corollary 5.39). This provides a partial explanation on the difficulty to find succinct HD parity automata, as we could argue that a simple way to conceptualise ω-regular languages is through deterministic Muller automata. Maybe most importantly, this sets a limitation in the usefulness of history-determinism in practice, as procedures that use a DMA as an intermediate step – as the ones from the tools Strix [65] and ltlsynt [70], or automata determinisation [78, 88, 64] – cannot benefit from the succinctness of HD automata.

On the other hand, we have shown that, if our objective is to obtain Rabin automata as output, the ACD-HD-Rabin-transform allows us to benefit from succinct HD automata. In this case, it has been shown that these automata can be exponentially smaller than equivalent deterministic ones [24, Theorem 21].

Disclosing the structure of ω-automata. As an application of the insights gained from the alternating cycle decomposition, we have derived results concerning typeness of automata. In

particular, we have characterised when we can define a parity, Rabin or Streett condition on top of a Muller automaton, obtaining an equivalent automaton (Propositions 6.9, 6.10 and 6.11). These characterisations have already been proven instrumental in works about the memory for games [21], and to obtain lower bounds on the size of deterministic Rabin automata [24].

We have also employed the ACD to present a normal form for parity transition systems and systematically proved the most important properties that make this form a valuable tool for manipulating parity automata. We believe that this normal form will be useful to extend existing results about Büchi and coBüchi automata (as the ones in [1, 10, 15]) to parity automata.

References

[1] **Bader Abu Radi and Orna Kupferman**. Minimization and canonization of GFG transition-based automata. *Logical Methods in Computer Science*, 18(3), 2022. DOI (4, 6, 7, 70, 83, 88, 93)

[2] **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? *International Conference on Computer-Aided Verification, CAV*, volume 13372 of *Lecture Notes in Computer Science*, pages 174–187, 2022. DOI (7, 92)

[3] **André Arnold, Jacques Duparc, Filip Murlak, and Damian Niwiński**. On the topological complexity of tree languages. *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, volume 2 of *Texts in Logic and Games*, pages 9–28, 2008. (3)

[4] **Tomáš Babiak, František Blahoudek, Alexandre Duret-Lutz, Joachim Klein, Jan Křetínský, David Müller, David Parker, and Jan Strejček**. The Hanoi omega-automata format. *International Conference on Computer-Aided Verification, CAV*, pages 479–486, 2015. DOI (107)

[5] **Roderick Bloem, Krishnendu Chatterjee, and Barbara Jobstmann**. Graph games and reactive synthesis. **Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem**, editors, *Handbook of Model Checking*, pages 921–962. Springer International Publishing, 2018. DOI (2)

[6] **León Bohn and Christof Löding**. Constructing deterministic parity automata from positive and negative examples. *CoRR*, abs/2302.11043, 2023. DOI (6, 83)

[7] **Bernard Boigelot, Sébastien Jodogne, and Pierre Wolper**. On the use of weak automata for deciding linear arithmetic with integer and real variables. *International Joint Conference on Automated Reasoning, IJCAR*, pages 611–625, 2001. DOI (6, 101)

[8] **Udi Boker**. Why these automata types? *Logic for Programming, Artificial Intelligence and Reasoning, LPAR*, volume 57 of *EPiC Series in Computing*, pages 143–163, 2018. DOI (16)

[9] **Udi Boker, Denis Kuperberg, Orna Kupferman, and Michał Skrzypczak**. Nondeterminism in the presence of a diverse or unknown future. *International Colloquium on Automata, Languages and Programming, ICALP*, pages 89–100, 2013. DOI (10, 42)

[10] **Udi Boker, Orna Kupferman, and Michal Skrzypczak**. How deterministic are good-for-games automata? *Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, volume 93, 18:1–18:14, 2017. DOI (77, 82, 93)

[11] **Udi Boker, Orna Kupferman, and Avital Steinitz**. Parityizing Rabin and Streett. *Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, volume 8 of *LIPIcs*, pages 412–423, 2010. DOI (6, 77, 83)

[12] **Udi Boker and Karoliina Lehtinen**. Good for Games Automata: From Nondeterminism to Alternation. *International Conference on Concurrency Theory, CONCUR*, volume 140, 19:1–19:16, 2019. DOI (19, 52)

[13] **Udi Boker and Karoliina Lehtinen**. History determinism vs. good for gameness in quantitative automata. *Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, volume 213, 38:1–38:20, 2021. DOI (14)

[14] **Udi Boker and Karoliina Lehtinen**. When a little nondeterminism goes a long way: an introduction to history-determinism. *ACM SIGLOG News*, 10(1):24–51, 2023. DOI (4, 10)

[15] **Patricia Bouyer, Antonio Casares, Mickael Randour, and Pierre Vandenhove**. Half-positional objectives recognized by deterministic Büchi automata. *International Conference on Concurrency Theory, CONCUR*, volume 243, 20:1–20:18, 2022. DOI (6, 7, 83, 93)

[16] Julian C. Bradfield. Simplifying the modal mu-calculus alternation hierarchy. *Symposium on Theoretical Aspects of Computer Science, STACS*, pages 39–49, 1998. DOI (3)

[17] J. Richard Büchi. On a decision method in restricted second order arithmetic. *Proceedings of the International Congress on Logic, Methodology and Philosophy of Science*:1–11, 1960. (17)

[18] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. *Symposium on Theory of Computing, STOC*, pages 252–263. ACM, 2017. DOI (3)

[19] Olivier Carton and Ramón Maceiras. Computing the Rabin index of a parity automaton. *Theoretical Informatics and Applications, RAIRO*:495–506, 1999. DOI (6, 18, 64, 82, 83)

[20] Olivier Carton and Max Michel. Unambiguous Büchi automata. *Theoretical Computer Science*, 297(1):37–81, 2003. DOI (32)

[21] Antonio Casares. On the minimisation of transition-based Rabin automata and the chromatic memory requirements of Muller conditions. *Computer Science Logic, CSL*, volume 216, 12:1–12:17, 2022. DOI (3, 7, 34, 88, 93)

[22] Antonio Casares. Structural properties of automata over infinite words and memory for games (Propriétés structurelles des automates sur les mots infinis et mémoire pour les jeux). PhD Thesis, Université de Bordeaux, France, 2023. URL (91)

[23] Antonio Casares, Thomas Colcombet, and Nathanaël Fijalkow. Optimal transformations of games and automata using Muller conditions. *International Colloquium on Automata, Languages and Programming, ICALP*, volume 198, 123:1–123:14, 2021. DOI (1, 7, 71)

[24] 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. *International Colloquium on Automata, Languages and Programming, ICALP*, volume 229, 117:1–117:20, 2022. DOI (1, 3, 7, 34, 70, 92, 93)

[25] Antonio Casares, Alexandre Duret-Lutz, Klara J. Meyer, Florian Renkin, and Salomon Sickert. Practical applications of the Alternating Cycle Decomposition. *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, volume 13244 of *Lecture Notes in Computer Science*, pages 99–117, 2022. DOI (7, 91, 92)

[26] Antonio Casares and Corto Mascle. The complexity of simplifying ω-automata through the alternating cycle decomposition. *CoRR*, abs/2401.03811, 2024. DOI (61)

[27] Thomas Colcombet. Forms of Determinism for Automata (Invited Talk). *Symposium on Theoretical Aspects of Computer Science, STACS*, volume 14, pages 1–23, 2012. DOI (4, 22)

[28] Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. *International Colloquium on Automata, Languages and Programming, ICALP*, pages 139–150, 2009. DOI (4, 14)

[29] Thomas Colcombet. Unambiguity in automata theory. *International Conference on Descriptional Complexity of Formal Systems, DFCS*, volume 9118 of *Lecture Notes in Computer Science*, pages 3–18, 2015. DOI (32)

[30] Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. *International Colloquium on Automata, Languages and Programming, ICALP*, volume 5126, pages 398–409, 2008. DOI (3)

[31] Thomas Colcombet and Damian Niwiński. On the positional determinacy of edge-labeled games. *Theoretical Computer Science*, 352(1-3):190–196, 2006. DOI (3, 7)

[32] Thomas Colcombet and Konrad Zdanowski. A tight lower bound for determinization of transition labeled Büchi automata. *International Colloquium on Automata, Languages and Programming, ICALP*, pages 151–162, 2009. DOI (7)

[33] Antonio Di Stasio, Aniello Murano, Vincenzo Prignano, and Loredana Sorrentino. Improving parity games in practice. *Annals of Mathematics and Artificial Intelligence*, 2021. DOI (2)

[34] Stefan Dziembowski, Marcin Jurdziński, and Igor Walukiewicz. How much memory is needed to win infinite games? *Symposium on Logic in Computer Science, LICS*, pages 99–110, 1997. DOI (4, 5, 34, 44, 48, 52, 56)

[35] Rüdiger Ehlers and Sven Schewe. Natural colors of infinite words. *Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, volume 250, 36:1–36:17, 2022. DOI (6, 7, 83)

[36] E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs. *SIAM Journal of Computing*, 29(1):132–158, 1999. DOI (3)

[37] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). *Symposium on Foundations of Computer Science, FOCS*, pages 368–377, 1991. DOI (3)

[38] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model-checking for fragments of μ-calculus. *International Conference on Computer-Aided Verification, CAV*, volume 697 of *Lecture Notes in Computer Science*, pages 385–396, 1993. DOI (3)

[39] Javier Esparza, Jan Křetínský, Jean-François Raskin, and Salomon Sickert. From LTL and limit-deterministic Büchi automata to deterministic parity automata. *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, pages 426–442, 2017. DOI (2)

[40] Seth Fogarty, Orna Kupferman, Moshe Y. Vardi, and Thomas Wilke. Profile trees for Büchi word automata, with application to determinization. *Information and Computation*, 245:136–151, 2015. DOI (3)

[41] Oliver Friedmann and Martin Lange. Solving parity games in practice. *International Symposium on Automated Technology for Verification and Analysis, ATVA*, pages 182–196, 2009. DOI (2, 6, 84)

[42] Dimitra Giannakopoulou and Flavio Lerda. From states to transitions: improving translation of LTL formulae to Büchi automata. *International Conference on Formal Techniques for Distributed Objects, Components, and Systems, FORTE*, pages 308–326, 2002. DOI (7)

[43] Yuri Gurevich and Leo Harrington. Trees, automata, and games. *Symposium on Theory of Computing, STOC*, pages 60–65, 1982. DOI (3)

[44] Thomas A. Henzinger and Nir Piterman. Solving games without determinization. *Computer Science Logic, CSL*, pages 395–410, 2006. DOI (4, 10, 14, 44, 45, 70)

[45] Florian Horn. Explicit Muller games are PTIME. *Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, pages 235–243, 2008. DOI (16)

[46] Florian Horn. Random fruits on the Zielonka tree. *Symposium on Theoretical Aspects of Computer Science, STACS*, volume 3, pages 541–552, 2009. DOI (4)

[47] Paul Hunter and Anuj Dawar. Complexity bounds for regular games. *International Symposium on Mathematical Foundations of Computer Science, MFCS*, pages 495–506, 2005. DOI (3, 16)

[48] 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. DOI (2)

[49] Marcin Jurdziński. Deciding the winner in parity games is in UP ∩ co-UP. *Information Processing Letters*, 68(3):119–124, 1998. DOI (3)

[50] Michael Kaminski. A classification of ω-regular languages. *Theoretical Computer Science*, 36:217–229, 1985. DOI (18)

[51] Nils Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Annals of Pure and Applied Logic*, 69(2):243–268, 1994. DOI (56)

[52] Jan Kretínský, Tobias Meggendorfer, and Salomon Sickert. Owl: A library for ω-words, automata, and LTL. *International Symposium on Automated Technology for Verification and Analysis, ATVA*, volume 11138 of *Lecture Notes in Computer Science*, pages 543–550, 2018. DOI (7, 92)

[53] Jan Křetínský, Tobias Meggendorfer, Clara Waldmann, and Maximilian Weininger. Index appearance record for transforming Rabin automata into parity automata. *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, pages 443–460, 2017. DOI (3)

[54] Jan Křetínský, Tobias Meggendorfer, Clara Waldmann, and Maximilian Weininger. Index appearance record with preorders. *Acta Informatica*, 59:585–618, 2021. DOI (4)

[55] Sriram C. Krishnan, Anuj Puri, and Robert K. Brayton. Deterministic ω-automata vis-a-vis deterministic Büchi automata. *International Symposium on Algorithms and Computation, ISAAC*, volume 834 of *Lecture Notes in Computer Science*, pages 378–386, 1994. DOI (6, 77, 83)

[56] Sriram C. Krishnan, Anuj Puri, and Robert K. Brayton. Structural complexity of omega-automata. *Symposium on Theoretical Aspects of Computer Science, STACS*, pages 143–156, 1995. DOI (18, 77)

[57] Denis Kuperberg and Michał Skrzypczak. On determinisation of good-for-games automata. *International Colloquium on Automata, Languages and Programming, ICALP*, pages 299–310, 2015. DOI (4, 6, 70, 83)

[58] Orna Kupferman. Automata theory and model checking. Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 107–151. Springer International Publishing, 2018. DOI (2)

[59] Orna Kupferman, Gila Morgenstern, and Aniello Murano. Typeness for omega-regular automata. *International Journal on Foundations of Computer Science*, 17(4):869–884, 2006. DOI (77)

[60] Orna Kupferman, Shmuel Safra, and Moshe Y. Vardi. Relating word and tree automata. *Symposium on Logic in Computer Science, LICS*, pages 322–332, 1996. DOI (3)

[61] Orna Kupferman and Moshe Y. Vardi. Safraless decision procedures. *Symposium on Foundations of Computer Science, FOCS*, pages 531–542, 2005. DOI (2)

[62] Oebele Lijzenga and Tom van Dijk. Symbolic parity game solvers that yield winning strategies. *International Symposium on Games, Automata, Logics, and Formal Verification, GandALF*, volume 326, pages 18–32, 2020. DOI (2)

[63] Christof Löding. Optimal bounds for transformations of ω-automata. *Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, pages 97–109, 1999. DOI (3, 5, 77)

[64] Christof Löding and Anton Pirogov. Determinization of Büchi automata: unifying the approaches of Safra and Muller-Schupp. *International Colloquium on Automata, Languages and Programming, ICALP*, 120:1–120:13, 2019. DOI (2, 92)

[65] **Michael Luttenberger**, **Philipp J. Meyer**, and **Salomon Sickert**. Practical synthesis of reactive systems from LTL specifications via parity games. *Acta Informatica*:3–36, 2020. DOI (2, 92)

[66] **Robert McNaughton**. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993. DOI (44, 49)

[67] **Robert McNaughton**. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):521–530, 1966. DOI (17, 19)

[68] **Philipp Meyer and Salomon Sickert**. On the optimal and practical conversion of Emerson-Lei automata into parity automata, 2021. Personal Communication (4, 5)

[69] **Philipp J. Meyer and Salomon Sickert**. Modernising strix, 2021. URL (2, 7)

[70] **Thibaud Michaud and Maximilien Colange**. Reactive synthesis from LTL specification with Spot. *Workshop on Synthesis, SYNT*, Electronic Proceedings in Theoretical Computer Science, 2018. (2, 92)

[71] **Andrzej W. Mostowski**. Regular expressions for infinite trees and a standard form of automata. *Symposium on Computation Theory, SCT*, pages 157–168, 1984. DOI (16–18)

[72] **David E. Muller**. Infinite sequences and finite machines. *Symposium on Switching Circuit Theory and Logical Design, SWCT*, pages 3–16, 1963. DOI (17)

[73] **David Müller and Salomon Sickert**. LTL to deterministic Emerson-Lei automata. *International Symposium on Games, Automata, Logics, and Formal Verification, GandALF*, pages 180–194, 2017. DOI (2)

[74] **Damian Niwiński**. On fixed-point clones (extended abstract). *International Colloquium on Automata, Languages and Programming, ICALP*, volume 226, pages 464–473, 1986. DOI (3)

[75] **Damian Niwiński and Igor Walukiewicz**. Deciding nondeterministic hierarchy of deterministic tree automata. *Workshop on Logic, Language, Information and Computation, WoLLIC*, volume 123 of *Electronic Notes in Theoretical Computer Science*, pages 195–208, 2004. DOI (3)

[76] **Damian Niwiński and Igor Walukiewicz**. Relating hierarchies of word and tree automata. *Symposium on Theoretical Aspects of Computer Science, STACS*, pages 320–331, 1998. DOI (3, 19)

[77] **Dominique Perrin and Jean-Eric Pin**. Infinite words - automata, semigroups, logic and games, volume 141 of *Pure and applied mathematics series*. Elsevier Morgan Kaufmann, 2004. (17)

[78] **Nir Piterman**. From nondeterministic Büchi and Streett automata to deterministic parity automata. *Symposium on Logic in Computer Science, LICS*, pages 255–264, 2006. DOI (2, 92)

[79] **Nir Piterman and Amir Pnueli**. Temporal logic and fair discrete systems. **Edmund M. Clarke**, **Thomas A. Henzinger**, **Helmut Veith**, and **Roderick Bloem**, editors, *Handbook of Model Checking*, pages 27–73. Springer International Publishing, 2018. DOI (2)

[80] **Amir Pnueli and Roni Rosner**. On the synthesis of a reactive module. *POPL*, pages 179–190, 1989. DOI (2)

[81] **Florian Renkin**, **Alexandre Duret-Lutz**, and **Adrien Pommellet**. Practical "paritizing" of Emerson-Lei automata. *International Symposium on Automated Technology for Verification and Analysis, ATVA*, volume 12302 of *Lecture Notes in Computer Science*, pages 127–143, 2020. DOI (4)

[82] **Bertrand Le Saëc**. Saturating right congruences. *Theoretical Informatics and Applications, RAIRO*, 24:545–559, 1990. DOI (80)

[83] **Schmuel Safra**. On the complexity of ω-automata. *Symposium on Foundations of Computer Science, FOCS*, pages 319–327, 1988. DOI (3)

[84] **Jacques Sakarovitch**. A construction on finite automata that has remained hidden. *Theoretical Computer Science*, 204(1-2):205–231, 1998. DOI (21)

[85] **Jacques Sakarovitch and Rodrigo de Souza**. Lexicographic decomposition of k-valued transducers. *Theoretical Computer Science*, 47(3):758–785, 2010. DOI (21)

[86] **Sven Schewe**. Beyond hyper-minimisation—minimising DBAs and DPAs is NP-complete. *Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, volume 8, pages 400–411, 2010. DOI (88)

[87] **Sven Schewe**. Minimising Good-For-Games automata is NP-complete. *Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, volume 182, 56:1–56:13, 2020. DOI (88)

[88] **Sven Schewe**. Tighter bounds for the determinisation of Büchi automata. *International Conference on Foundations of Software Science and Computation Structures, FoSSaCS*, pages 167–181, 2009. DOI (2, 3, 92)

[89] **Sven Schewe and Thomas Varghese**. Determinising parity automata. *International Symposium on Mathematical Foundations of Computer Science, MFCS*, pages 486–498, 2014. DOI (7)

[90] **Michał Skrzypczak**. Topological extension of parity automata. *Information and Computation*, 228-229:16–27, 2013. DOI (3)

[91] **Robert Tarjan**. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):114–121, 1972. DOI (91)

[92] **Tom van Dijk**. Oink: an implementation and evaluation of modern parity game solvers. *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, volume 10805 of *Lecture Notes in Computer Science*, pages 291–308, 2018. DOI (2)

[93] **Klaus Wagner**. On ω-regular sets. *Information and Control*, 43(2):123–177, 1979. DOI (3, 5, 18, 19, 57, 92)

[94] **Wiesław Zielonka**. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998. DOI   (3, 4, 6, 7, 16, 34, 56, 77)

A.   Generalised classes of acceptance conditions

Further acceptance conditions.

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

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

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

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

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

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

REMARK A.1. 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. $[1, 2]$).

We will also define *conditions that depend on the structure* of the transition system and not only on the set of colours.

Generalised weak transition systems. Let $\mathcal{TS} = (G_{\mathcal{TS}}, \mathsf{Acc}_{\mathcal{TS}})$ be a transition system using a parity condition $\mathsf{Acc}_{\mathcal{TS}} = (\gamma, [d_{\min}, d_{\max}], \mathsf{parity})$. We say that $\mathcal{TS}$ 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, $|\gamma(E_{\mathcal{S}})| \leq d$, where $E_{\mathcal{S}}$ is the set of edges of $\mathcal{S}$.

As for the rest of conditions, we say that a transition system $\mathcal{TS}$ is Weak_d *type* if there exists an isomorphic parity transition system $\mathcal{TS}' \simeq \mathcal{TS}$ that is Weak_d.

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

As we will show (Corollary A.11), 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.

The Zielonka tree of generalised acceptance conditions.

DEFINITION A.2. Let T be a tree T with nodes partitioned into round nodes and square nodes. We say that T has:

— *Büchi shape* if it has a single branch, height at most 2, and if it has height 2 its root is round.
— *coBüchi shape* if it has a single branch, height at most 2, and if it has height 2 its root is square.
— *Generalised Büchi shape* if it has height at most 2, and if it has height 2 its root is round.
— *Generalised coBüchi shape* if it has height at most 2, and if it has height 2 its root is square.

PROPOSITION A.3. *Let $\mathcal{F} \subseteq 2_+^\Gamma$ be a family of non-empty subsets. Then $\mathsf{Muller}_\Gamma(\mathcal{F})$ is a Büchi (resp. coBüchi) language if and only if $\mathcal{Z}_\mathcal{F}$ has Büchi (resp. coBüchi) shape.*

PROOF. This is just a special case of Proposition 6.4. ∎

PROPOSITION A.4. *Let $\mathcal{F} \subseteq 2_+^\Gamma$ be a family of non-empty subsets. Then $\mathsf{Muller}_\Gamma(\mathcal{F})$ is a generalised Büchi (resp. coBüchi) language if and only if $\mathcal{Z}_\mathcal{F}$ has generalised Büchi (resp. generalised coBüchi) shape.*

PROOF. We do the proof for the case generalised Büchi (symmetric for generalised coBüchi). Assume that $\mathsf{Muller}_\Gamma(\mathcal{F}) = \mathsf{genBüchi}_\Gamma(B)$ for some family $B = \{B_1, \ldots, B_k\}$. Then, $\Gamma \in \mathcal{F}$, as $\Gamma \cap B_i \neq \emptyset$, so the root of $\mathcal{Z}_\mathcal{F}$ is round. If $C \subseteq \Gamma$ is rejecting, $C \cap B_i = \emptyset$ for all i, then it is the same for any subset $C' \subseteq C$, so square nodes of $\mathcal{Z}_\mathcal{F}$ are leaves and $\mathcal{Z}_\mathcal{F}$ has height at most 2.

Conversely, assume that $\mathcal{Z}_\mathcal{F}$ has height 2 and that its root is round ($\Gamma \in \mathcal{F}$). Let $A_1, \ldots, A_k$ be the labels of the k leaves of $\mathcal{Z}_\mathcal{F}$ and define $B_i = A_i$. We claim that $\mathsf{Muller}_\Gamma(\mathcal{F}) = \mathsf{genBüchi}_\Gamma(B)$, for $B = \{B_1, \ldots, B_k\}$. Indeed, if $C \in \mathcal{F}$ if and only if $C \not\subseteq A_i$ for any i if and only if $C \cap B_i \neq \emptyset$ for all i. ∎

COROLLARY A.5. *Let $\mathcal{A}$ be a deterministic generalised Büchi (resp. generalised coBüchi) automaton recognising a Muller language $L = \mathsf{Muller}_\Sigma(\mathcal{F})$. There is a deterministic generalised Büchi (resp. generalised coBüchi) automaton recognising L with just one state, that can be computed in polynomial time in the size of the representation of $\mathcal{A}$.*

PROOF. We do the proof for the case generalised Büchi. By Remark A.1, the parity index of L is at most $[0, 1]$, so by Proposition 6.13, the Zielonka tree of $\mathcal{F}$ has generalised Büchi shape. Therefore, by Proposition A.4, L is a generalised Büchi that can be trivially recognised by a generalised Büchi automaton with just one state.

The acceptance condition of such automaton can be deduced in linear time from the Zielonka tree $\mathcal{Z}_\mathcal{F}$, as indicated in the proof of Proposition A.4. The Zielonka tree $\mathcal{Z}_\mathcal{F}$ can be computed from the original automaton $\mathcal{A}$ using a similar argument than in the proof of Theorem 6.32. Suppose that the generalised Büchi condition used by $\mathcal{A}$ is given by the sets $B_1, \ldots, B_k \subseteq \Gamma$. Then, for each $i \in \{1, \ldots, k\}$ we compute the restriction of $\mathcal{A}$ to the transitions using colours in $\Gamma \setminus B_i$, and perform a decomposition in SCCs of the obtained graph. If $\Sigma_S \subseteq \Sigma$ is the set of input letters appearing in one of those SCC, then $\Sigma_S \notin \mathcal{F}$. We put all the subsets of letters obtained in that way in a list *altSets*. The leaves of $\mathcal{Z}_\mathcal{F}$ correspond then to the maximal subsets of *altSets*. ∎

ACD and typeness for generalised acceptance conditions.

DEFINITION A.6. 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:

— *Büchi ACD* if it is a $[0, 1]$-parity ACD.

— *coBüchi ACD* if it is a $[1, 2]$-parity ACD.

— *Generalised Büchi ACD* if for every state $v \in V$, the tree $\mathcal{T}_v$ has generalised Büchi shape.

— *Generalised coBüchi ACD* if for every state $v \in V$, the tree $\mathcal{T}_v$ has generalised coBüchi shape.

— *Weak_d ACD* if it is a parity ACD and trees of $\mathcal{ACD}_{\mathcal{TS}}$ have height at most d.

REMARK A.7. $\mathcal{ACD}_{\mathcal{TS}}$ is a Weak_d ACD if and only if it is a $[0, d]$-parity ACD and a $[1, d+1]$-parity ACD.

PROPOSITION A.8. *A transition system $\mathcal{TS}$ is Büchi (resp. coBüchi) type if and only if $\mathcal{ACD}_{\mathcal{TS}}$ is a Büchi ACD (resp. coBüchi ACD).*

PROOF. This is a special case of Proposition 6.11.   ■

PROPOSITION A.9. *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 A.4, using as set of output colours the set of edges of $\mathcal{TS}$.   ■

PROPOSITION A.10. *A transition system $\mathcal{TS}$ is Weak_d type if and only if $\mathcal{ACD}_{\mathcal{TS}}$ is a Weak_d ACD.*

PROOF. Proposition 6.11 already provides that $\mathcal{TS}$ is parity type if and only if $\mathcal{ACD}_{\mathcal{TS}}$ is a parity ACD. As in the proof of the aforementioned proposition, we observe that $\mathcal{TS}$ and $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ are isomorphic. If $\mathcal{ACD}_{\mathcal{TS}}$ is a Weak_d ACD, then $\mathsf{ACD}_{\mathsf{parity}}(\mathcal{TS})$ is Weak_d. Conversely, if $\mathcal{ACD}_{\mathcal{TS}}$ is not a Weak_d ACD, then $\mathcal{TS}$ contains a $(d+1)$-flower, so the number of colours cannot be reduced (using the same argument as in the proof of Theorem 5.34).   ■

COROLLARY A.11. *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.*

PROOF. Let L be of parity index Weak_d. By definition, L is recognised by parity automaton $\mathcal{A}$ using colours in $[0, d]$ (it is also recognised by an automaton using colours in $[1, d+1]$; we make an arbitrary choice). We will prove that $\mathcal{A}$ is in fact Weak_d. We will show that its ACD $\mathcal{ACD}_\mathcal{A}$ is Weak_d type, which allows to conclude by Proposition A.10. Suppose that this was not the case, that is, that some tree of $\mathcal{ACD}_\mathcal{A}$ has height at least $d+1$. In this case, $\mathcal{A}$ would contain a $(d+1)$-flower (Lemma 5.42), so by the Flower Lemma 2.16, L has parity index at least $[0, d]$ or $[1, d+1]$, a contradiction.   ■

The following result generalises [7, Theorem 2].

COROLLARY A.12. *A Muller transition system is $Weak_d$ type if and only if it is both $[0, d]$ and $[1, d + 1]$-parity type.*

Deterministic automata using generalised acceptance conditions.

COROLLARY A.13. *Let $G_{\mathcal{A}}$ be the underlying graph of a deterministic automaton. There are $[0, d − 1]$ and $[1, d]$-parity conditions $Acc_{p,0}$ and $Acc_{p,1}$ such that $\mathcal{L}(G_{\mathcal{A}}, Acc_{p,0}) = \mathcal{L}(G_{\mathcal{A}}, Acc_{p,1})$ if and only if there is a $Weak_p$ condition Acc_W such that $\mathcal{L}(G_{\mathcal{A}}, Acc_W) = \mathcal{L}(G_{\mathcal{A}}, Acc_{p,0}) = \mathcal{L}(G_{\mathcal{A}}, Acc_{p,1})$.*

PROPOSITION A.14. *Let $\mathcal{A}$ be a deterministic Muller automaton, and assume that $\mathcal{L}(\mathcal{A})$ can be recognised by a deterministic Büchi (resp. coBüchi) automaton; that is, the parity index of $\mathcal{L}(\mathcal{A})$ is 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 assume 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 6.13, so it is a generalised Büchi ACD, and by Proposition A.9, $\mathcal{A}$ is generalised Büchi type. ∎

Parity index from automata in normal form.

COROLLARY A.15. *Let $\mathcal{A}$ be a deterministic parity automaton in normal form using colours 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 colours in $[0, d − 1]$ (in particular, it is not negative). If $\mathcal{A}$ is not $Weak_{d−1}$, there is an SCC containing all the colours $[0, d − 1]$. By Proposition 6.27, such SCC contains a positive d-flower, so by the Flower Lemma 2.16, the parity index of $\mathcal{L}(\mathcal{A})$ is $[0, d − 1]$.

Suppose now that $\mathcal{A}$ is $Weak_{d−1}$. Let ℓ be a cycle of $\mathcal{A}$ in which the colour $d − 1$ occurs. By Proposition 6.27, 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 2.16, the parity index of $\mathcal{L}(\mathcal{A})$ is $Weak_{d−1}$. ∎

COROLLARY A.16. *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 assume that $\mathcal{L}(\mathcal{A})$ has parity index $[0, d-1]$ (the other cases are similar). By the Flower Lemma 2.16, $\mathcal{A}$ does not contain any negative d-flower. By Proposition 6.13, 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 6.11. ∎

B.   Transformations for games

Games suitable for transformations. As we have indicated throughout the paper, 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*.
— 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 paper 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 ($\gamma(e) = \varepsilon$), $v' \in V_{\text{Eve}}$, and e is the only incoming edge to v' ($\text{In}(v') = \{e\}$).

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 G with vertices V and edges E, there exists a game $\widetilde{G}$ that is suitable for transformations, of size $|\widetilde{G}| = O(|E|)$, and equivalent to G in the following sense: there is an injective function $f\colon V \to \widetilde{V}$ such that Eve wins G from v if and only if she wins $\widetilde{G}$ from $f(v)$.*

PROOF. We define $\widetilde{G}$ 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 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 G, we add the edges $v \xrightarrow{\varepsilon} e$ and $e \xrightarrow{c} v'$ to G'. It is clear that $\widetilde{G}$ is suitable for transformations and that Eve wins G from v if and only if she wins $\widetilde{G}$ from v. ∎

ACD-HD-Rabin-transform-for-games. As discussed in Section 5.3, the ACD-HD-Rabin-transform of a game G does not always induce an HD-for-games mapping $\varphi\colon \text{ACD}_{\text{Rabin}}(G) \to G$, and G and $\text{ACD}_{\text{Rabin}}(G)$ 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 paragraph we propose a small modification on the transformation to obtain a correct transformation for games.

Let G be a game. If there is an edge $v \to v'$ with $v \in V_{\text{Adam}}$, we say that v' is an *A-successor*. We remark that if 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 G and $V_{\text{normal}} = V \setminus V_{\text{A-succ}}$. If G is suitable for transformations, for each $v \in V_{\text{A-succ}}$ we let $\text{pred}(v)$ be its unique predecessor.

The idea to define the ACD-HD-Rabin-transform-for-games $\text{ACD}_{\text{parity}}^{\text{game}}(G)$ is the following: starting from the regular ACD-HD-Rabin-transform $\text{ACD}_{\text{Rabin}}(G)$, we make some local changes to vertices that are A-successors. First, if $v \in V_{\text{Adam}}$, we replace edges of the form $(v, x) \xrightarrow{n} (v', x')$ in $\text{ACD}_{\text{Rabin}}(G)$ 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 $\text{ACD}_{\text{Rabin}}(G)$, then we add $(v', x) \xrightarrow{n} (v'', x'')$ to $\text{ACD}_{\text{parity}}^{\text{game}}(G)$ (we note that $v' \in V_{\text{Eve}}$). 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.2. Let 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 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 $n_1 \in \text{Leaves}(\mathcal{T}_v)$ and $n_2 = \text{Jump}_{\mathcal{T}_{v'}}(n_1, \text{Supp}(n_1, e_1))$. Then, $\text{Supp}(n_2, e_2))$ is a descendant of $\text{Supp}(n_1, e_1))$ in $\mathcal{T}_{v'}$.

DEFINITION B.3 (ACD-HD-Rabin-transform-for-games). Let G be a Muller game suitable for transformations. For each vertex $v \in V$ we let $\eta_v\colon \text{Leaves}(\mathcal{T}_v) \to \{1, \dots, \text{rbw}(\mathcal{T}_v)\}$ be a mapping

satisfying Property (★) from Lemma 4.42. We define the *ACD-HD-Rabin-transform-for-games* of $\mathcal{G}$ to be the Rabin transition system $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{parity}}(\mathcal{G})$ defined as follows.

Vertices. The set of vertices is

$$\widetilde{V} = \bigcup_{v \in V_{\mathsf{normal}}} \{v\} \times [1, \mathsf{rbw}(\mathcal{T}_v)] \ \cup \ \bigcup_{v \in V_{\mathsf{A\text{-}succ}}} \{v\} \times [1, \mathsf{rbw}(\mathcal{T}_{\mathsf{pred}(v)})].$$

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 $v \in V_{\mathsf{Eve}} \cap V_{\mathsf{normal}}$, we add $(v, x) \xrightarrow{n} (v', x')$ to $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{parity}}(\mathcal{G})$ exactly in the same cases as in the regular ACD-HD-Rabin-transform.

— If $v \in V_{\mathsf{Adam}}$, we let $(v, x) \xrightarrow{\varepsilon} (v', x)$ in $\widetilde{E}$ for each $x \in \{1, \dots, \mathsf{rbw}(\mathcal{T}_v)\}$.

— If $v \in V_{\mathsf{A\text{-}succ}}$, we add $(v, x) \xrightarrow{n} (v', x')$ to $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{parity}}(\mathcal{G})$ 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,

$$\widetilde{E} = \bigcup_{\substack{e = v \to v' \in E \\ v \in V_{\mathsf{normal}}}} \{e\} \times \mathsf{Leaves}(\mathcal{T}_v) \ \cup \ \bigcup_{\substack{e = v \to v' \in E \\ v \in V_{\mathsf{A\text{-}succ}}}} \{e\} \times \mathsf{Leaves}(\mathcal{T}_{\mathsf{pred}(v)}).$$

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 follows: 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 5.32 (Correctness of the ACD-HD-Rabin-transform-for-games). *Let $\mathcal{G}$ be a Muller game suitable for transformations, and let $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{parity}}(\mathcal{G})$ be its ACD-HD-Rabin-transform-for-games. Then, there is an HD-for-games mapping $\varphi \colon \mathsf{ACD}^{\mathsf{game}}_{\mathsf{parity}}(\mathcal{G}) \to \mathcal{TS}$.*

PROOF. The proof is analogous to that of the correctness of the usual ACD-HD-Rabin-transform (Proposition 5.28). We define the mapping $\varphi \colon \mathsf{ACD}^{\mathsf{game}}_{\mathsf{parity}}(\mathcal{G}) \to \mathcal{G}$ 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 5.29 and Remark B.2.

We define a resolver (r_0, r) simulating φ similarly to the proof of Proposition 5.28: 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 $\mathsf{ACD}^{\mathsf{game}}_{\mathsf{parity}}(\mathcal{G})$ 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 $\rho \colon (v_i, x_i) \xrightarrow{\varepsilon} (v_{i+1}, x_i)$. 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 $n_i = \text{Supp}(l_i, e_i)$.

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 $n_i = \text{Supp}(l_i, e_i)$. This is indeed an edge appearing in $\text{ACD}^{\text{game}}_{\text{parity}}(\mathcal{G})$, 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 $\text{ACD}_{\text{Rabin}}(\mathcal{G})$, with $x'_i = \eta_{v_i}(l_i)$.

The resolver obtained in this way is sound for $\text{ACD}^{\text{game}}_{\text{parity}}(\mathcal{G})$, 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. ∎

Optimality of the ACD-HD-Rabin-transform-for-games

COROLLARY 5.37. *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, $|\text{ACD}^{\text{game}}_{\text{parity}}(\mathcal{G})| \leq 2|\widetilde{\mathcal{G}}|$.*

PROOF. The vertices of $\text{ACD}^{\text{game}}_{\text{parity}}(\mathcal{G})$ corresponding to vertices in V_{normal} are exactly the same that those in $\text{ACD}_{\text{Rabin}}(\mathcal{G})$:

$$\{(v, x) \in \text{ACD}^{\text{game}}_{\text{parity}}(\mathcal{G}) \mid v \in V_{\text{normal}}\} = \{(v, x) \in \text{ACD}_{\text{Rabin}}(\mathcal{G}) \mid v \in V_{\text{normal}}\}.$$

Moreover, for $v \in V_{\text{A-succ}}$, there is one vertex of the form (v, x) for each vertex $(\text{pred}(v), x)$, and each $v \in V_{\text{A-succ}}$ has exactly one predecessor in V_{normal}, so we conclude that:

$$|\text{ACD}^{\text{game}}_{\text{parity}}(\mathcal{G})| \leq 2 \cdot |\{(v, x) \in \text{ACD}^{\text{game}}_{\text{parity}}(\mathcal{G}) \mid v \in V_{\text{normal}}\}| \leq 2 \cdot |\text{ACD}_{\text{Rabin}}(\mathcal{G})| \leq 2 \cdot \mathcal{G}',$$

where the last inequality follows from Theorem 5.36. ∎

C. Simplifications for prefix-independent conditions

We prove in this appendix results applying to automata recognising prefix-independent languages and games using prefix-independent winning conditions. We recall 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 $w \in L$.

LEMMA 2.5. *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, it holds that $\mathcal{A}$ recognises the same language if we fix q as initial state, that is, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_q)$. 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) (there is a word $w_0 \in \Sigma^*$ and a run $\rho_0 = r_0 \xrightarrow{w_0} q$ induced by r over w). 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 $r'(\rho, a) = r(\rho_0 \rho, a)$ 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:

$$w \in \mathcal{L}(\mathcal{A}) \iff w_0 w \text{ is accepted using } (r_0, r) \iff w_0 w \text{ is accepted using } (r_0', r'). \qquad \blacksquare$$

LEMMA 3.14. *Let TS and TS' be two TS such that all their states are accessible, and let $\varphi \colon TS \to TS'$ be an HD (resp. HD-for-games) mapping between them. If $\mathbb{W}$ and $\mathbb{W}'$ are prefix-independent, the mapping φ is also HD (resp. HD-for-games) when considered between the transition systems TS_V and $TS'_{V'}$, consisting of the transition systems TS and TS' where all the states are set to be initial.*

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

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

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

In the rest of the proof we assume that φ is an HD-for-games mapping, (which covers the HD case). Let (r_{Init}, r) be a resolver sound for TS simulating $\varphi \colon TS \to TS'$. We define a resolver $(\tilde{r}_{\text{Init}}, \tilde{r})$ for the new mapping. For every state v of TS, we fix a finite run $\rho_v \in \mathcal{P}ath_I^{\text{fin}}(TS)$ ending in v that is consistent with (r_{Init}, r) over some ρ', if such a run exists. 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 TS', one such v is $\text{Target}(r_{\mathcal{R}uns}(\rho'_{v'}))$ (that is, the vertex to which we arrive in TS when simulating $\rho'_{v'}$ via the original resolver). We let $\tilde{r}_{\text{Init}}(v')$ be this vertex. If $e' \in \text{Out}(v')$ is an edge in TS', we let

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

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

Finally, we show that $(\tilde{r}_{\mathrm{Init}}, \tilde{r})$ is sound for $\mathcal{TS}$. Let $\rho' = v' \xrightarrow{w'} \in \mathcal{R}un(\mathcal{TS}'_{V'})$ be an accepting run, and let $\rho = v \xrightarrow{w'}$ be a run consistent with $(\tilde{r}_{\mathrm{Init}}, \tilde{r})$ over ρ'. In particular, $v = \tilde{r}_{\mathrm{Init}}(v')$. Let $\rho_v = v_0 \xrightarrow{u} v$ be the chosen run reaching v and let $\rho'_v = v'_0 \xrightarrow{u'} v'$ be a finite run in $\mathcal{R}un^{\mathrm{fin}}(\mathcal{TS}')$ such that ρ_v is consistent with (r_{Init}, r) over ρ'_v. It is immediate to check that $\rho_v \rho$ is consistent with (r_{Init}, r) over $\rho'_v \rho'$. Since ρ' is accepting, we have that $w' \in \mathbb{W}'$, and by prefix-independence of the acceptance sets and the fact that $\rho_v \rho$ is accepting if $\rho'_v \rho'$ is, we have:

$$w' \in \mathbb{W}' \implies u'w' \in \mathbb{W}' \implies uw \in \mathbb{W} \implies w \in \mathbb{W},$$

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

D.   Simplifying automata with duplicated edges

Given an automaton $\mathcal{A} = (Q, \Sigma, I, \Gamma, \delta, \mathbb{W})$ we say that it has *duplicated edges* if there is 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 4.47, 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 [4] 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 D.1 (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, $\Delta' = \{(q, a, x_j, q') : (q, a, y, q') \in \Delta$ for some $y\}$. The new Rabin condition $\{(G'_1, R'_1), \ldots, (G'_r, R'_r)\}$ is defined as follows. For each transition $q \xrightarrow{a:x} q'$:

— $x \in G'_i$ if $q \xrightarrow{a:y} q' \in \Delta$ for some $y \in G_i$,
— $x \in R'_i$ if for all $q \xrightarrow{a:y} q' \in \Delta$, $y \in R_i$.

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 (G_i, R_i), then the corresponding run ρ' in $\mathcal{A}'$ over u is also accepted by the Rabin pair (G'_i, R'_i): the transitions of $\mathrm{Inf}(\rho) \cap G_i$ induce transitions of $\mathrm{Inf}(\rho') \cap G'_i$ and the fact that $\mathrm{Inf}(\rho) \cap R_i = \emptyset$ guarantees that $\mathrm{Inf}(\rho') \cap R'_i = \emptyset$.

Conversely, if $u \in \mathcal{L}(\mathcal{A}')$ as witnessed by a run ρ' and Rabin pair (G'_i, 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 $x \in G'_i$ a transition $q \xrightarrow{a:y} q' \in \Delta$ such that $y \in G_i$, which exists by definition of $\mathcal{A}'$, for each transition $q \xrightarrow{a:x} q'$ where $x \notin G_i \cup R_i$ a transition $q \xrightarrow{q,y} q' \in \Delta$ such that $y \notin R_i$, which also exists by definition of $\mathcal{A}'$, and for other transitions $q \xrightarrow{a:x} q'$ (that is, those for which $x \in R'_i$) an arbitrary transition $q \xrightarrow{a:y} q' \in \Delta$. Since ρ' is accepting, we have $\mathrm{Inf}(\rho') \cap G_i \neq \emptyset$ and $\mathrm{Inf}(\rho) \cap R_i = \emptyset$, 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 transition t that occurs infinitely often in ρ' such that $\bigcup_{t \in \mathrm{Inf}(\rho)} S_t$ is accepting in $\mathcal{A}$. 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 $t \in \mathrm{Inf}(\rho)$ each transition in S_t infinitely often, and otherwise takes an arbitrary transition. The set of transitions ρ visits infinitely often is exactly $\bigcup_{t \in \mathrm{Inf}(\rho)} S_t$, 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 D.2. The ZT-HD-Rabin-automaton from Figure 14 has duplicated transitions. In Figure 19 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:
$$G'_\beta = \{\beta, (\alpha\beta)\}, \quad R'_\beta = \{\alpha, \lambda, \xi, \zeta\},$$
$$G'_\lambda = \{\lambda\}, \qquad\quad R'_\lambda = \{\alpha, \beta, (\alpha\beta), \theta\}.$$
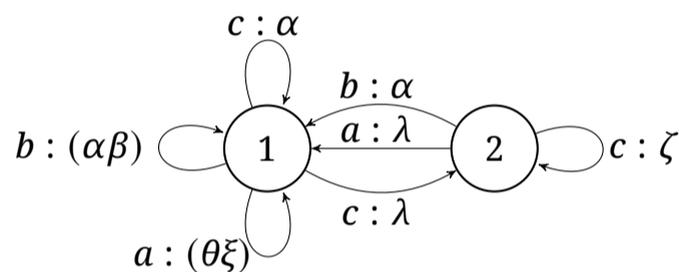
Figure 19. The simplified ZT-HD-Rabin-automaton.

E. Proofs for Section 6.1.1

PROOF OF PROPOSITIONS 6.2 AND 6.3. We prove it for the Rabin case, Streett conditions being the dual notion.

   If all round nodes of $\mathcal{Z}_{\mathcal{F}} = (N = N_\bigcirc \sqcup N_\square, \leq, \nu : N \to 2^\Gamma_+)$ have at most one child, we define a family of Rabin pairs $R = \{(G_n, R_n) \mid n \in N_\bigcirc\}$ such that $\mathsf{Rabin}(R) = \mathsf{Muller}(\mathcal{F})$ as follows: for each round node $n \in N_\bigcirc$, we add a Rabin pair (G_n, R_n). We let $G_n = \Gamma \setminus \nu(n)$. In order to define R_n, we observe that n has at most one child n', and we define $R_n = \nu(n) \setminus \nu(n')$, for n' the only child of n, if it exists, or $R_n = \nu(n)$ if n has no children at all. This is, the pair (G_n, R_n) accepts the sets of colours $A \subseteq \Gamma$ that contain some of the colours that disappear in the step $n \to n'$ and none of the colours appearing up in the tree. 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 $\leq$) containing A. It is a round node and there is some colour $c \in A$ not appearing in the only child of n. Therefore, $c \in G_n$ and $A \cap R_n = \emptyset$. Conversely, if $A \notin \mathcal{F}$, then for every round node n with a child n', either $A \subseteq \nu(n')$ (and therefore $A \cap G_n = \emptyset$) or $A \nsubseteq \nu(n)$ (and in that case $A \cap R_n \neq \emptyset$).

We remark that this construction uses more Rabin pairs than necessary, since we could reuse Rabin pairs for nodes that are in the same level and that are not siblings.

Conversely, suppose that $\mathsf{Muller}(\mathcal{F}) = \mathsf{Rabin}(R)$ for the Rabin language associated to $R = \{(G_1, R_1), \ldots, (G_r, R_r)\}$. If $n \in \mathcal{Z}_{\mathcal{F}}$ is a round node ($A = v(n) \in \mathcal{F}$), then its label A contains some colours that belongs to $G_{i_1}, \ldots, G_{i_k}$ and none belonging to $R_{i_1}, \ldots, R_{i_k}$ for some $i_1, \ldots, i_k$, $k \geq 1$. A child of n must not have these colours, so the only maximal subset of A that is not in $\mathcal{F}$ is $A \setminus (G_{i_1} \cup \cdots \cup G_{i_k})$. ∎

PROOF OF PROPOSITION 6.4. We assume $\Gamma \in \mathcal{F}$ ($\min_{\mathcal{F}} = 0$), the other case is symmetric.

Assume that $\mathcal{Z}_{\mathcal{F}}$ has a single branch of length $\max_{\mathcal{F}} + 1$. We define a mapping $\phi \colon \Gamma \to [0, \max_{\mathcal{F}}]$ as follows: for each colour $c \in \Gamma$ we let n_c be the deepest node in $\mathcal{Z}_{\mathcal{F}}$ containing c, and we define $\phi(c) = v(n_c)$. It is easy to check that for all $w \in \Gamma^\omega$, $w \in \mathsf{Muller}(\mathcal{F})$ if and only if $\phi(w) \in \mathsf{parity}$.

Conversely, assume that we can assign colours to the elements of Γ by $\phi : \Gamma \to [0, d]$, whose corresponding parity language is $\mathsf{Muller}(\mathcal{F})$. We show that any node of the Zielonka tree $\mathcal{Z}_{\mathcal{F}}$ has at most one child. Indeed, let $n \in N$ and let $c \in v(n)$ such that $\phi(c) = \min\{\phi(c) \mid c \in v(n)\}$. We suppose that $\phi(c)$ is odd (the proof is symmetric for $\phi(c)$ even). Let $p = \min\{\phi(c') \mid c' \in v(n)$ and $\phi(c')$ even$\}$. In every child of n the elements with a smaller colour than p must disappear, so the set of elements $v(n) \cap \{c \in \Gamma \mid \phi(c) \geq p\}$ is the only maximal subset of $v(n)$ belonging to $\mathcal{F}$. Moreover, in the label of the child of n there is at least one colour less, so the height of $\mathcal{Z}_{\mathcal{F}}$ will be at most $d + 1$. ∎