Optimal Transformations of Games and Automata using Muller Conditions

Antonio Casares
LaBRI, Université de Bordeaux, France

Thomas Colcombet
CNRS, IRIF, Université de Paris, France

Nathanaël Fijalkow
CNRS, LaBRI, Université de Bordeaux, France
The Alan Turing Institute of Data Science, London, United Kingdom

Abstract

We consider the following question: given an automaton or a game with a Muller condition, how can we efficiently construct an equivalent one with a parity condition? There are several examples of such transformations in the literature, including in the determinisation of Büchi automata.

We define a new transformation called the alternating cycle decomposition, inspired and extending Zielonka's construction. Our transformation operates on transition systems, encompassing both automata and games, and preserves semantic properties through the existence of a locally bijective morphism. We show a strong optimality result: the obtained parity transition system is minimal both in number of states and number of priorities with respect to locally bijective morphisms.

We give two applications: the first is related to the determinisation of Büchi automata, and the second is to give crisp characterisations on the possibility of relabelling automata with different acceptance conditions.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects

Keywords and phrases Automata over infinite words, Omega regular languages, Determinisation of automata

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.116

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/pdf/2011.13041.pdf` [4]

Funding *Thomas Colcombet*: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No.670624) and the DeLTA ANR project (ANR-16-CE40-0007).

Acknowledgements We want to thank Philipp Meyer and Salomon Sickert for their comments and for spotting a mistake on a previous version of this paper.

1 Introduction

Games and automata form the theoretical basis for the verification and synthesis of reactive systems; we refer to the recent Handbook [5] for a broad exposition of this research area, in particular Chapters 2 and 27. A milestone objective is the synthesis of reactive systems specified in *Linear Temporal Logic* (LTL). The original approach of Pnueli and Rosner [24] using automata and games devised more than four decades ago is today at the heart of the state of the art synthesis tools [8, 16, 20, 21]. The bottleneck is the determinisation of Büchi automata: given a non-deterministic Büchi automaton, construct an equivalent parity automaton. This problem has a long history; it was originally solved by McNaughton [18], and the first asymptotically optimal construction is due to Safra [25], see also [15] for a recent exposition. Most of the recent theoretical and practical solutions of this problem are based on the construction of Piterman [23]. Schewe's [26] enlightening perspective on this construction is to decompose it into two steps: first construct a deterministic Muller automaton, and then transform it into an equivalent deterministic parity automaton. Piterman and Schewe's determinisation procedure is one of many examples of constructions using as an intermediate step (subclasses of) Muller conditions before transforming them into parity conditions, either working with automata models or games models.

The objective of this work is to focus on this particular step and study transformations from Muller to parity. We work with general transition systems to seamlessly encompass both automata and games models.

There are several existing constructions transforming subclasses of Muller conditions to parity. The first is the Latest Appearance Record (LAR) [9], which applies to all Muller conditions. It was proved to be optimal in the worst case [17]: *there exists* a family of Muller automata for which the obtained parity automata are minimal. Many refinements of the LAR have been constructed for subclasses of Muller conditions, *e.g.* [17, 13].

The starting point of our work is the notion of a Zielonka tree of a Muller condition, which was introduced in [30] and shown to capture the exact memory requirements of Muller games [7]. In the long version of [7], it implicitly appears that the Zielonka tree of a Muller condition can be used to construct a parity automaton recognising this Muller condition. Our first observation is to show a *strong* optimality result: *for all* Muller conditions, the parity automaton obtained from the Zielonka tree of a Muller condition is minimal both in the number of states and in the number of priorities. This result has also been obtained in the independent unpublished work [19]. This optimality result is much stronger than the worst case optimality result of the LAR transformation; in essence, it shows that the Zielonka tree of a Muller condition precisely captures the properties of the Muller condition, whereas for instance the LAR only depends on the number of colours.

Our first insight is to note that all existing constructions, including the one based on Zielonka trees, only consider the Muller condition but do not take into account the structure of the underlying transition system. In other words, all transformations work at the level of conditions: they transform a Muller condition into a parity condition, and ignore the interplay between the condition and the transition structure.

Our main contribution is to construct a new transformation called the alternating cycle decomposition (ACD) which captures this interplay: the ACD transforms a Muller transition system $\mathcal{T}$ into a parity transition system $\mathcal{P}_{\mathcal{ACD}(\mathcal{T})}$, extending Zielonka trees by considering the alternation of accepting and rejecting cycles in $\mathcal{T}$.

Our second insight is to introduce the notion of locally bijective morphisms to capture the notion of a "transformation", preserving many natural semantic properties (such as language equivalence, being deterministic, unambiguous, or good for game in the context of automata, and the winner for games). We use this notion to state and prove a strong optimality result for the ACD transformation: $\mathcal{P}_{\mathcal{ACD}(\mathcal{T})}$ is minimal both in the number of states and in the number of priorities amongst parity transition systems admitting a locally bijective morphism into $\mathcal{T}$.

We present two applications. The first is an improvement in the determinisation of Büchi automata: the second step of the Piterman and Schewe construction is a locally bijective transformation of some deterministic Muller automaton into a deterministic parity automaton; we show that our ACD transformation yields in all cases smaller (and in some sense minimal) automata, and in many cases strictly smaller. The second application is a set of crisp characterisations for relabelling transition systems with different classes of acceptance conditions: for instance, given a transition system with a Rabin condition, does there exist a parity condition on the same structure yielding an equivalent transition system? This unifies and extends results from [1, 30].

The outline of the paper follows the narration of this introduction. We show in Section 3 how the Zielonka tree yields a parity automaton recognising the Muller condition, inducing a transformation at the level of conditions. We then lift this transformation from conditions to transition systems: we introduce the alternating cycle decomposition and its transformation in Section 4. Our two applications are discussed in Section 5.

2 Notations and definitions

The symbol ω denotes the ordered set of non-negative integers. For $i, j \in \omega$, $i \leq j$, the notation $[i, j]$ stands for $\{i, i+1, \dots, j-1, j\}$. For a set Σ, a *word* over Σ is a sequence of elements from Σ. The length of a word u is $|u|$. The set of words of finite length (resp. of length ω) over Σ will be written Σ^* (resp. Σ^ω). We let $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. For a word $u \in \Sigma^\infty$ we write u_i to represent the i-th letter of u. If $u = v \cdot w$ for $v \in \Sigma^*, u, w \in \Sigma^\infty$, we say that v is a *prefix* of u and we write $v \sqsubseteq u$ (it induces a partial order on Σ^*). For a finite word $u \in \Sigma^*$ we write $\mathit{First}(u) = u_1$ and $\mathit{Last}(u) = u_{|u|}$. For a word $u \in \Sigma^\infty$, we let $\mathit{Inf}(u) = \{a \in \Sigma \ : \ u_i = a$ for infinitely many $i \in \omega\}$ and $\mathit{Occ}(u) = \{a \in \Sigma \ : \ \exists i \in \omega$ such that $u_i = a\}$. Given a map $\alpha : A \to B$, we implicitly extend α to words component-wise, i.e., $\alpha : A^\infty \to B^\infty$ will be defined as $\alpha(a_1 a_2 \dots) = \alpha(a_1)\alpha(a_2)\dots$. A *directed graph* is a tuple $(V, E, \mathit{Source}, \mathit{Target})$ where V is a set of vertices, E a set of edges and $\mathit{Source}, \mathit{Target} : E \to V$ are maps indicating the source and target for each edge. A *path* is a word $\varrho \in E^*$ such that $\mathit{Source}(\varrho_{i+1}) = \mathit{Target}(\varrho_i)$ for $i < |\varrho|$. A graph is *strongly connected* if there is a path connecting each pair of vertices. A *subgraph* of $(V, E, \mathit{Source}, \mathit{Target})$ is a graph $(V', E', \mathit{Source}', \mathit{Target}')$ such that $V' \subseteq V$, $E' \subseteq E$ and Source' and Target' are the restriction to E' of Source and Target, respectively. A *strongly connected component* is a maximal strongly connected subgraph. For a subset of vertices $A \subseteq V$ we write: $\mathit{In}(A) = \{e \in E \ : \ \mathit{Target}(e) \in A\}$ and $\mathit{Out}(A) = \{e \in E \ : \ \mathit{Source}(e) \in A\}$.

Transition systems. A *transition system graph* $\mathcal{T}_G = (V, E, \mathit{Source}, \mathit{Target}, I_0)$ is a directed graph with a non-empty set of initial vertices $I_0 \subseteq V$. We will also refer to vertices and edges as *states* and *transitions*, respectively. We will suppose that every vertex has at least one outgoing edge. A *transition system* $\mathcal{T}$ is obtained from a transition system graph $\mathcal{T}_G$ by adding:

- A function $\gamma : E \to \Gamma$. The set Γ will be called a *set of colours* and the function γ a *colouring function*.
- An *acceptance condition* $\mathit{Acc} \subseteq \Gamma^\omega$.

For technical convenience we use transition-labelled systems: acceptance conditions are defined over edges instead of over states. These can be easily transformed into state-labelled systems. We will usually take $\Gamma = E$ and γ the identity function. In that case we will omit γ in the description of $\mathcal{T}$. We let $|\mathcal{T}|$ denote $|V|$, for V the set of vertices.

A (finite or infinite) *run* from $q \in V$ on a transition system graph $\mathcal{T}$ is a path $\varrho = e_1 e_2 \dots \in E^\infty$ starting at q. For $A \subseteq V$ we let $\mathcal{Run}_{\mathcal{T},A}$ denote the set of runs on $\mathcal{T}$ starting from some $q \in A$, and $\mathcal{Run}_{\mathcal{T}} = \mathcal{Run}_{\mathcal{T},I_0}$ the set of runs starting from some initial vertex. A run $\varrho \in \mathcal{Run}_{\mathcal{T}}$ is *accepting* if $\gamma(\varrho) \in \mathit{Acc}$, and rejecting otherwise. In this work we suppose that only infinite runs can be accepted.

We say that a vertex $v \in V$ is *accessible* if there exists a finite run $\varrho \in \mathcal{R}un_{\mathcal{T}}$ ending in v. A set of vertices $B \subseteq V$ is accessible if every vertex $v \in B$ is accessible. The *accessible part* of a transition system is the set of accessible vertices.

We might want to add additional information to a transition system (as illustrated in the following paragraphs). For this purpose we introduce labelled transition system: a *vertex-labelled* (resp. edge-labelled) transition system is a transition system $\mathcal{T}$ with a labelling function $l_V : V \to L_V$ (resp. $l_E : E \to L_E$) from vertices (resp. edges) into a set of labels.

Automata as transition systems. An *automaton* is an edge-labelled transition system $\mathcal{A} = (V, E, Source, Target, I_0, Acc, l_E)$ where $l_E : E \to \Sigma$, for Σ a finite set called the *input alphabet* (we say that $\mathcal{A}$ is an automaton over Σ). Given a word $w \in \Sigma^\omega$, a *run over* w is an infinite run $\varrho \in \mathcal{R}un_{\mathcal{T}}$ such that $l_E(\varrho_i) = w_i$ for every $i > 0$. The word $w \in \Sigma^\omega$ is accepted by the automaton $\mathcal{A}$ if there exists an accepting run over w in $\mathcal{A}$. The *language accepted* by an automaton $\mathcal{A}$ is the set $\mathcal{L}(\mathcal{A}) := \{u \in \Sigma^\omega \ : \ u \text{ is accepted by } \mathcal{A}\}$.

We say that an automaton $\mathcal{A}$ is *deterministic* if $|I_0| = 1$ and for every $q \in V$ and every $a \in \Sigma$ there is exactly one edge $e \in Out(v)$ such that $l_E(e) = a$. In this case, we write $\delta(q, a)$ for the only state reachable from q taking the transition labelled with a. We extend the function $\delta(q, -)$ to finite words in the natural way. If $\mathcal{A}$ is deterministic then there is a single run over w for each $w \in \Sigma^\omega$, written $\mathcal{A}(w)$.

Games as transition systems. A *game* $\mathcal{G}_{v_0} = (V, E, Source, Target, v_0, Acc, l_V)$ is a vertex-labelled transition system with a single initial vertex v_0 and vertices labelled by a function $l_V : V \to \{Eve, Adam\}$ that induces a partition of V into vertices controlled by two different players. A *play* is an infinite run produced by moving a token along edges: the player controlling the current vertex chooses what transition to take. It is *winning* for Eve if it is accepting, and winning for Adam otherwise. We say that player $P \in \{Eve, Adam\}$ *wins* the game $\mathcal{G}_{v_0}$ if P can force to always produce a winning play. The *winning region* for player P is the set of vertices $v \in V$ such that P wins the game $\mathcal{G}_v$ obtained by setting the initial vertex to v.

Classes of acceptance conditions. We present the main classes of *ω-regular conditions*. Let Γ be a finite set of *colours*, it will usually be the set of edges of a transition system.

- **Büchi** A *Büchi condition* Acc_B is represented by a subset $B \subseteq \Gamma$. An infinite word $u \in \Gamma^\omega$ belongs to Acc_B if some colour from B appears infinitely often in u.
- **Rabin** A *Rabin condition* Acc_R is represented by a family of *Rabin pairs*, $R = \{(E_1, F_1), \dots, (E_r, F_r)\}$, where $E_i, F_i \subseteq \Gamma$. A word $u \in \Gamma^\omega$ belongs to Acc_R if $Inf(u) \cap E_i \neq \emptyset$ and $Inf(u) \cap F_i = \emptyset$ for some index $i \in \{1, \dots, r\}$.
- **Streett** A word $u \in \Gamma^\omega$ belongs to the *Streett condition* Acc_S associated to the family $S = \{(E_1, F_1), \dots, (E_r, F_r)\}$, $E_i, F_i \subseteq \Gamma$ if $Inf(u) \cap E_i \neq \emptyset \ \rightarrow \ Inf(u) \cap F_i \neq \emptyset$ for every $i \in \{1, \dots, r\}$.
- **Parity** To define a *parity condition* we suppose that Γ is a finite subset of $\mathbb{N}$. A word $u \in \Gamma^\omega$ belongs to the condition Acc_P if $\min Inf(u)$ is even. The elements of Γ are called *priorities* in this case. We associate to a parity condition the interval $[\mu, \eta]$, where $\mu = \min \Gamma$ and $\eta = \max \Gamma$.
- **Muller** A *Muller condition* $Acc_{\mathcal{F}}$ is given by a family $\mathcal{F} \subseteq \mathcal{P}(\Gamma)$. A word $u \in \Gamma^\omega$ is accepted if the colours appearing infinitely often in u form a set of the family $\mathcal{F}$.

Equivalent conditions. Two different acceptance conditions over a set Γ are *equivalent* if they define the same set $Acc \subseteq \Gamma^\omega$. Given a transition system graph $\mathcal{T}_G$, two representations

$\mathcal{R}_1, \mathcal{R}_2$ of acceptance conditions are equivalent over $\mathcal{T}_G$ if they define the same accepting subset of runs of $\mathcal{R}un_T$. We write $(\mathcal{T}_G, \mathcal{R}_1) \simeq (\mathcal{T}_G, \mathcal{R}_2)$ in that case.

If $\mathcal{A}$ is the transition system graph of an automaton and $\mathcal{R}_1, \mathcal{R}_2$ are two representations of acceptance conditions such that $(\mathcal{A}, \mathcal{R}_1) \simeq (\mathcal{A}, \mathcal{R}_2)$, then they recognise the same language: $\mathcal{L}(\mathcal{A}, \mathcal{R}_1) = \mathcal{L}(\mathcal{A}, \mathcal{R}_2)$. However, the converse only holds for deterministic automata.

▶ **Proposition 2.1.** *Let $\mathcal{A}$ be the the transition system graph of a deterministic automaton over the alphabet Σ and let $\mathcal{R}_1, \mathcal{R}_2$ be two representations of acceptance conditions such that $\mathcal{L}(\mathcal{A}, \mathcal{R}_1) = \mathcal{L}(\mathcal{A}, \mathcal{R}_2)$. Then, both conditions are equivalent over $\mathcal{A}$, $(\mathcal{A}, \mathcal{R}_1) \simeq (\mathcal{A}, \mathcal{R}_2)$.*

▶ Remark. A parity condition given by $\Gamma \subseteq \mathbb{N}$ is equivalent to Rabin and Streett conditions over Γ. Any of the previous conditions over a set Γ is equivalent to a Muller condition.

Trees. A *tree* is a set of sequences of non-negative integers $T \subseteq \omega^*$ that is prefix-closed: if $\tau \cdot i \in T$, for $\tau \in \omega^*, i \in \omega$, then $\tau \in T$. In this paper we will only consider finite trees.

The elements of T are called *nodes*. A *subtree* of T is a tree $T' \subseteq T$. The empty sequence ε belongs to every non-empty tree and it is called the *root* of the tree. A node of the form $\tau \cdot i$, $i \in \omega$, is called a *child* of τ, and τ is called its *parent*. We let $\mathit{Children}(\tau)$ denote the set of children of a node τ. Two different children σ_1, σ_2 of τ are called *siblings*, and we say that σ_1 is *older* than σ_2 if $\mathit{Last}(\sigma_1) < \mathit{Last}(\sigma_2)$. If two nodes τ, σ verify $\tau \sqsubseteq \sigma$, then τ is called an *ancestor* of σ, and σ a *descendant* of τ (we add the adjective "strict" if in addition they are not equal). A node is called a *leaf* of T if it is a maximal sequence of T. A *branch* of T is the set of prefixes of a leaf. The set of branches of T is denoted $\mathit{Branch}(T)$. We order the set of branches from left to right.

For a node $\tau \in T$ we define $\mathit{Subtree}_T(\tau)$ as the subtree consisting on the set of nodes that appear below τ, or above it in the same branch: $\mathit{Subtree}_T(\tau) = \{\sigma \in T \ : \ \sigma \sqsubseteq \tau \text{ or } \tau \sqsubseteq \sigma\}$.

Given a node τ of a tree T, the *depth* of τ in T is defined as the length of τ, $\mathit{Depth}(\tau) = |\tau|$. The *height of a tree* T, written $\mathit{Height}(T)$, is defined as the maximal depth of a leaf of T plus 1. The *height of the node* $\tau \in T$ is $\mathit{Height}(T) - \mathit{Depth}(\tau)$.

A *labelled tree* is a pair (T, ν), where T is a tree and $\nu : T \to \Lambda$ is a labelling function into a set of labels Λ.

3 An optimal transformation of Muller into parity conditions

In this section we show how to use the Zielonka tree of a Muller condition to construct a deterministic parity automaton recognising the Muller condition. This can be seen as an extension of the existing constructions transforming Muller conditions into parity conditions such as the LAR [9] or the Index Appearance Record (IAR) [13, 17]. We prove that for all Muller conditions, the parity automaton has a minimal number of states (Theorem 3.7) and a minimal number of priorities (Proposition 3.6).

3.1 The Zielonka tree automaton

▶ **Definition 3.1** (Zielonka tree of a Muller condition [30]). *Let Γ be a finite set of colours and $\mathcal{F} \subseteq \mathcal{P}(\Gamma)$ a Muller condition over Γ. The Zielonka tree of $\mathcal{F}$, written $T_{\mathcal{F}}$, is a tree labelled with subsets of Γ via the labelling $\nu : T_{\mathcal{F}} \to \mathcal{P}(\Gamma)$, defined inductively as:*

- $\nu(\varepsilon) = \Gamma$
- *If τ is a node already constructed labelled with $S = \nu(\tau)$, we let $S_1, \ldots, S_k$ be the maximal subsets of S verifying the property $S_i \in \mathcal{F} \iff S \notin \mathcal{F}$, for $i \in \{1, \ldots, k\}$. For each $i \in \{1, \ldots, k\}$ we add a child to τ labelled with S_i.*

We say that the condition $\mathcal{F}$ and the tree $T_{\mathcal{F}}$ are *even* (resp. *odd*) if $\Gamma \in \mathcal{F}$ (resp. $\Gamma \notin \mathcal{F}$). To each node τ of the Zielonka tree we associate the priority $p_Z(\tau) = \mathit{Depth}(\tau)$, and we add 1 to it if $T_{\mathcal{F}}$ is odd.

This way, $p_Z(\tau)$ is even if and only if $\nu(\tau) \in \mathcal{F}$. We represent nodes $\tau \in T_{\mathcal{F}}$ such that $p_Z(\tau)$ is even as a *circle* (*round nodes*), and those for which $p_Z(\tau)$ is odd as a *square*.

▶ **Example 3.2.** Let $\Gamma_1 = \{a, b, c\}$ and $\mathcal{F}_1 = \{\{a\}, \{b\}\}$. The Zielonka tree $T_{\mathcal{F}_1}$ is shown in Figure 1. It is odd.

Let $\Gamma_2 = \{a, b, c, d\}$ and $\mathcal{F}_2 = \{\{a, b, c, d\}, \{a, b, d\}, \{a, c, d\}, \{b, c, d\}, \{a, b\}, \{a, d\}, \{b, c\}, \{b, d\}, \{a\}, \{b\}, \{d\}\}$. The Zielonka tree $T_{\mathcal{F}_2}$ is even and it is shown on Figure 2.

On the right of each tree there are the priorities assigned to the nodes of the corresponding level. We have named the branches of the Zielonka trees with greek letters and we indicate the names of the nodes in violet.

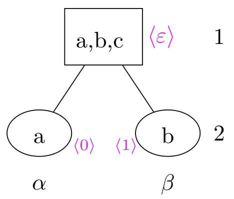


Figure 1 Zielonka tree $T_{\mathcal{F}_1}$.

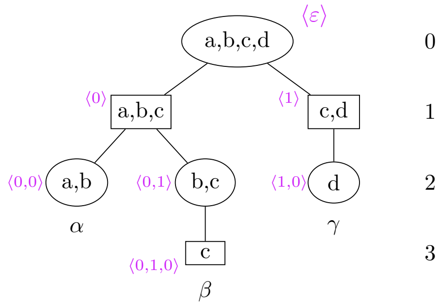


Figure 2 Zielonka tree $T_{\mathcal{F}_2}$.

We show next how to use the Zielonka tree of $\mathcal{F}$ to build a deterministic automaton recognizing the Muller condition $\mathcal{F}$. This automaton can be implicitly found in [7].

For a branch $\beta \in \mathit{Branch}(T_{\mathcal{F}})$ and a colour $a \in \Gamma$ we define $\mathit{Supp}(\beta, a) = \tau$ as the deepest node (maximal for $\sqsubseteq$) in β such that $a \in \nu(\tau)$.

Given a node $\tau \in \beta$, if τ is not a leaf then it has a unique child σ_β such that $\sigma_\beta \in \beta$. In this case, we let $\mathit{Nextchild}(\beta, \tau)$ be the next sibling of σ_β on its right, or the smallest child of τ if σ_β is the biggest one.

We define $\mathit{Nextbranch}(\beta, \tau)$ as the leftmost branch in T below $\mathit{Nextchild}(\beta, \tau)$, if τ is not a leaf, and we let $\mathit{Nextbranch}(\beta, \tau) = \beta$ if τ is a leaf of T.

▶ **Definition 3.3** (Zielonka tree automaton). *Given a Muller condition $\mathcal{F}$ over Γ with Zielonka tree $T_{\mathcal{F}}$, we define the Zielonka tree automaton $\mathcal{Z}_{\mathcal{F}}$ as a deterministic automaton over Γ using a parity acceptance condition given by $p : E \to [\mu, \eta]$, where*

- *$Q = \mathit{Branch}(T_{\mathcal{F}})$, the set of states is the set of branches of $T_{\mathcal{F}}$.*
- *The initial state q_0 is irrelevant, we pick the leftmost branch of $T_{\mathcal{F}}$.*
- *The transitions are: $\delta(\beta, a) = \mathit{Nextbranch}(\beta, \mathit{Supp}(\beta, a))$, for $\beta \in \mathit{Branch}(T_{\mathcal{F}})$ and $a \in \Gamma$.*
- *$\mu = 0$, $\eta = \mathit{Height}(T_{\mathcal{F}}) - 1$ if $\mathcal{F}$ is even; $\mu = 1$, $\eta = \mathit{Height}(T_{\mathcal{F}})$ if $\mathcal{F}$ is odd.*
- *$p(\beta, a) = p_Z(\mathit{Supp}(\beta, a))$.*

The transitions of the automaton are determined as follows: if we are in a branch β and we read a colour a, then we move up in the branch β until we reach a node τ that contains the colour a in its label. Then we pick the child of τ just on the right of the branch β (in a cyclic way) and we move to the leftmost branch below it. We produce the priority corresponding to the depth of τ.

▶ **Example 3.4.** Let us consider the conditions of Example 3.2. The Zielonka tree automaton for the Muller condition $\mathcal{F}_1$ is shown in Figure 3, and that for $\mathcal{F}_2$ in Figure 4.

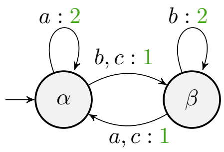


Figure 3 The Zielonka tree automaton $\mathcal{Z}_{\mathcal{F}_1}$.

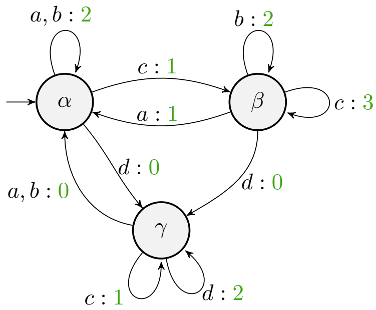


Figure 4 The Zielonka tree automaton $\mathcal{Z}_{\mathcal{F}_2}$.

▶ **Proposition 3.5** (Correctness). *Let $\mathcal{F} \subseteq \mathcal{P}(\Gamma)$ be a Muller condition over Γ. Then, a word $u \in \Gamma^\omega$ verifies $Inf(u) \in \mathcal{F}$ if and only if u is accepted by $\mathcal{Z}_\mathcal{F}$.*

3.2 Optimality of the Zielonka tree automaton

We prove in this section the strong optimality of the Zielonka tree automaton, both for the number of priorities (Proposition 3.6) and for the size (Theorem 3.7). These results have been obtained independently in a recent unpublished work by Meyer and Sickert [19].

▶ **Proposition 3.6** (Optimal number of priorities, independently proved in [19]). *The Zielonka tree $\mathcal{Z}_\mathcal{F}$ uses the optimal number of priorities for recognizing a Muller condition $\mathcal{F}$. More precisely, if $[\mu, \eta]$ are the priorities used by $\mathcal{Z}_\mathcal{F}$ and $\mathcal{P}$ is another parity automaton recognizing $\mathcal{F}$, then $\mathcal{P}$ uses at least $\eta - \mu + 1$ priorities, and in case of equality, its smallest priority has the same parity as μ.*

▶ **Theorem 3.7** (Optimal size of the Zielonka tree automaton, independently proved in [19]). *Every deterministic parity automaton $\mathcal{P}$ accepting a Muller condition $\mathcal{F}$ over Γ verifies $|\mathcal{Z}_\mathcal{F}| \leq |\mathcal{P}|$.*

The proof of both results appear in the full version of this paper [4], and Proposition 3.6 can also be deduced from the results of [22]. We sketch the proof of Theorem 3.7: for a set of letters $X \subseteq \Sigma$ we define an X-SCC of an automaton $\mathcal{A}$ over Σ as a strongly connected component of the graph obtained restricting the transitions of $\mathcal{A}$ to those labelled with letters from X. We prove that if A and B are the labels of two siblings in the Zielonka tree $T_\mathcal{F}$, and $\mathcal{P}$ is a parity automaton recognising the Muller condition $\mathcal{F}$, then A-SCCs and B-SCCs of $\mathcal{P}$ must be disjoint. Finding such disjoints X-SCC for the children of the nodes of the Zielonka tree allows us to conclude the proof by induction.

4 An optimal transformation of Muller into parity transition systems

In the previous section we have shown how the Zielonka tree yields a transformation of a Muller condition into a parity condition, through the construction of a deterministic parity automaton. This can be naturally lifted to transition systems by composing the automaton with the transition system. However this approach is oblivious to the transition system, meaning it does not consider the possibly fruitful interplay between the transition structure and the condition. All existing transformations follow this approach.

In this section we present our main contribution: an optimal transformation of Muller transition systems into parity transition systems. The key novelty is that it precisely captures the way the transition structure interacts with the condition. In the seminal work [28], Wagner introduces the *alternating chains of loops* of an automaton. This idea has been successfully applied to determine the complexity of computing the Rabin index of different types of ω-automata [2, 14, 22, 29]. Inspired by the notion of Zielonka trees and Wagner's alternating chains, we define a data structure called the alternating cycle decomposition (ACD) analysing the alternating chains of accepting and rejecting cycles of the transition system. We arrange this information in a collection of Zielonka trees obtaining a data structure, the alternating cycle decomposition, that subsumes all the structural information of the transition system necessary to determine whether a run is accepted or not.

We start in Subsection 4.1 by defining the notion of "transformations" using locally bijective morphisms. This will allow us to state the strong optimality result of Proposition 4.8 and Theorem 4.10: for all Muller transition system $\mathcal{T}$, the parity transition system $\mathcal{P}_{\mathcal{ACD}(\mathcal{T})}$ is minimal both in number of states and number of priorities amongst parity transition systems admitting a locally bijective morphism into $\mathcal{T}$.

4.1 Locally bijective morphisms as witnesses of transformations

▶ **Definition 4.1.** *Let $\mathcal{T} = (V, E, \mathit{Source}, \mathit{Target}, I_0, \mathit{Acc})$, $\mathcal{T}' = (V', E', \mathit{Source}', \mathit{Target}', I'_0, \mathit{Acc}')$ be two transition systems. A morphism of transition systems, written $\varphi : \mathcal{T} \to \mathcal{T}'$, is a pair of maps $(\varphi_V : V \to V', \varphi_E : E \to E')$ such that:*

- *$\varphi_V(v_0) \in I'_0$ for every $v_0 \in I_0$ (initial states are preserved).*
- *$\mathit{Source}'(\varphi_E(e)) = \varphi_V(\mathit{Source}(e))$ for every $e \in E$ (origins of edges are preserved).*
- *$\mathit{Target}'(\varphi_E(e)) = \varphi_V(\mathit{Target}(e))$ for every $e \in E$ (targets of edges are preserved).*
- *For every run $\varrho \in \mathcal{R}un_{\mathcal{T}}$, $\varrho \in \mathit{Acc} \Leftrightarrow \varphi_E(\varrho) \in \mathit{Acc}'$ (acceptance condition is preserved).*

For labelled transition systems, we say that φ is a morphism of labelled transition systems if it also preserves the labels.

We will denote both maps by φ whenever no confusion arises.

▶ **Definition 4.2.** *Given two transition systems $\mathcal{T}$ and $\mathcal{T}'$, a morphism of transition systems $\varphi : \mathcal{T} \to \mathcal{T}'$ is called locally bijective if for every $v \in V$ the restriction of φ_E (resp. φ_V) to $\mathit{Out}(v)$ (resp. I_0) is a bijection into $\mathit{Out}(\varphi(v))$ (resp. I'_0).*

This is a very similar concept to the usual notion of bisimulation. The main difference is that locally bijective morphisms treat the acceptance of a run as a whole, allowing us to compare transition systems using different classes of acceptance conditions.

▶ **Observation 4.3.** *If $\varphi : \mathcal{T} \to \mathcal{T}'$ is a locally bijective morphism, then φ induces a bijection between the runs in $\mathcal{R}un_{\mathcal{T}}$ and $\mathcal{R}un_{\mathcal{T}'}$ that preserves their acceptance.*

Intuitively, if we transform a transition system $\mathcal{T}_1$ into $\mathcal{T}_2$ "without adding non-determinism", we will have a locally bijective morphism $\varphi : \mathcal{T}_2 \to \mathcal{T}_1$. In particular, if we take the product $\mathcal{T}_2 = \mathcal{T}_1 \times \mathcal{B}$ of $\mathcal{T}_1$ by some deterministic automaton $\mathcal{B}$, the projection over $\mathcal{T}_1$ yields a locally bijective morphism.

The existence of a locally bijective morphism is a witness of the fact that two systems share the same semantic properties: languages recognised by automata are preserved, as well as winning regions of games. Moreover, other important semantic properties of automata, such as being *unambiguous* or *good for games* (notions studied, respectively, in [3] and [10]) are preserved too. We refer to the full version for details [4].

4.2 The alternating cycle decomposition

In the following we will consider Muller transition systems $\mathcal{T} = (V, E, \mathit{Source}, \mathit{Target}, I_0, \mathcal{F})$ with the Muller acceptance condition using edges as colours. We can always suppose this, however, the size of the representation of the condition $\mathcal{F}$ might change. Making this assumption corresponds to considering what are called *explicit Muller conditions*. In particular, solving Muller games with explicit Muller conditions is in PTIME [11], while solving general Muller games is PSPACE-complete [12].

▶ **Example 4.4.** We will use the transition system $\mathcal{T}$ in Figure 5 as a running example. Its Muller condition is given by $\mathcal{F} = \{\{c, d, e\}, \{e\}, \{g, h, i\}, \{l\}, \{h, i, j, k\}, \{j, k\}\}$.

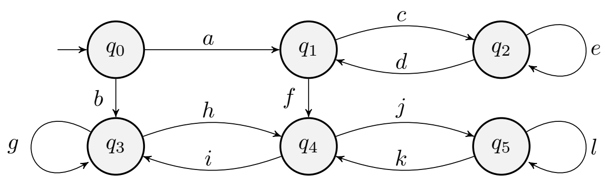


Figure 5 Transition system $\mathcal{T}$.

Given a transition system $\mathcal{T}$, a *loop* is a subset of edges $l \subseteq E$ such that exists $v \in V$ and a finite run $\varrho \in \mathcal{R}un_{T,v}$ starting and ending in v and $\mathit{Occ}(\varrho) = l$. The set of loops of $\mathcal{T}$ is denoted $\mathcal{L}oop(\mathcal{T})$. For a loop $l \in \mathcal{L}oop(\mathcal{T})$ we write $\mathit{States}(l) := \{v \in V \ : \ \exists e \in l, \ \mathit{Source}(e) = v\}$.

There is a natural partial order in the set $\mathcal{L}oop(\mathcal{T})$ given by set inclusion. The maximal loops of $\mathcal{L}oop(\mathcal{T})$ are disjoint and in one-to-one correspondence with the strongly connected components of $\mathcal{T}$.

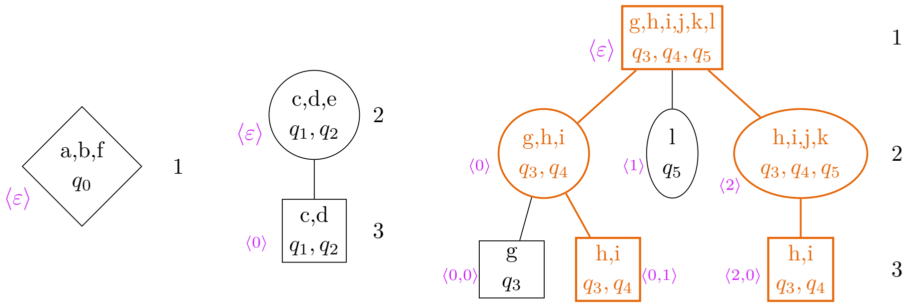
In the system $\mathcal{T}$ in Figure 5, examples of loops are $l_1 = \{c, d, e\}$ or $l_2 = \{j, k\}$, with $\mathit{States}(l_1) = \{q_1, q_2\}$ and $\mathit{States}(l_2) = \{q_4, q_5\}$. The loop l_1 is maximal.

▶ **Definition 4.5** (Alternating cycle decomposition). *Let $\mathcal{T}$ be a Muller transition system with acceptance condition given by $\mathcal{F} \subseteq \mathcal{P}(E)$. The alternating cycle decomposition of $\mathcal{T}$, noted $\mathcal{ACD}(\mathcal{T})$, is a family of labelled trees $(t_1, \nu_1), \dots, (t_r, \nu_r)$ with nodes labelled by loops in $\mathcal{L}oop(\mathcal{T})$, $\nu_i : t_i \to \mathcal{L}oop(\mathcal{T})$. We define it inductively as follows:*

- *Let $\{l_1, \dots, l_r\}$ be the set of maximal loops of $\mathcal{L}oop(\mathcal{T})$. For each $i \in \{1, \dots, r\}$ we consider a tree t_i and define $\nu_i(\varepsilon) = l_i$.*
- *Given an already defined node τ of a tree t_i we consider the maximal loops of the set $\{l \subseteq \nu_i(\tau) \ : \ l \in \mathcal{L}oop(\mathcal{T}) \text{ and } l \in \mathcal{F} \Leftrightarrow \nu_i(\tau) \notin \mathcal{F}\}$ and for each of these loops l we add a child to τ in t_i labelled by l.*

For notational convenience we add a special tree (t_0, ν_0) with a single node ε labelled with the edges not appearing in any other tree of the forest, i.e., $\nu_0(\varepsilon) = E \setminus \bigcup_{i=1}^{r} l_i$. We define $\mathit{States}(\nu_0(\varepsilon)) := V \setminus \bigcup_{i=1}^{r} \mathit{States}(l_i)$.

We call the trees $t_1, \dots, t_r$ the proper trees of the alternating cycle decomposition of $\mathcal{T}$. Given a node τ of t_i, we note $\mathit{States}_i(\tau) := \mathit{States}(\nu_i(\tau))$.



Tree t_0. Tree t_1. Tree t_2. In bold orange, the subtree t_{q_4}.

Figure 6 Alternating cycle decomposition of $\mathcal{T}$. The priority assigned to the nodes of each level of the trees is indicated on the right. Nodes with an even priority are drawn as circles and those with an odd priority as rectangles (excepting the special node forming the root of t_0). Each node τ is labelled with $\nu_i(\tau)$ and with $States_i(\tau)$. In violet the names of the nodes.

The ACD of $\mathcal{T}$ is shown in Figure 6. It consists of two proper trees, t_1 and t_2, corresponding to the strongly connected components of $\mathcal{T}$ and the tree t_0 that corresponds to the edges not appearing in the strongly connected components.

▶ Remark. The Zielonka tree for a Muller condition $\mathcal{F}$ can be seen as a special case of this construction, for an automaton with a single state.

Since each state and edge of $\mathcal{T}$ appears in exactly one of the trees of $\mathcal{ACD}(\mathcal{T})$, we can define the *index* of a state $q \in V$ (resp. of an edge $e \in E$) in $\mathcal{ACD}(\mathcal{T})$ as the only number $j \in \{0, 1, \ldots, r\}$ such that $q \in States_j(\varepsilon)$ (resp. $e \in \nu_j(\varepsilon)$).

For each state $q \in V$ of index j we define the *subtree associated to the state q* as the subtree t_q of t_j consisting in the set of nodes $\{\tau \in t_j \;:\; q \in States_j(\tau)\}$.

In Figure 6, state q_4 has index 2, and the subtree associated to q_4 is shown in bold orange.

For each proper tree t_i of $\mathcal{ACD}(\mathcal{T})$ we say that t_i is *even* if $\nu_i(\varepsilon) \in \mathcal{F}$ and that it is *odd* if $\nu_i(\varepsilon) \notin \mathcal{F}$. We say that $\mathcal{ACD}(\mathcal{T})$ is *odd* if all the trees of maximal height of $\mathcal{ACD}(\mathcal{T})$ are odd.

For each $\tau \in t_i$, $i = 1, \ldots, r$, we define the *priority* of τ in t_i as $p_i(\tau) = Depth(\tau)$, adding 1 if t_i is odd. In the case where $\mathcal{ACD}(\mathcal{T})$ is odd we add 2 to nodes on even trees in order to use an optimal number of priorities. We assign to $p_0(\varepsilon)$ the minimal priority appearing in other trees (0 or 1).

We proceed to show how to use the alternating cycle decomposition of a Muller transition system to obtain a parity one.

▶ **Definition 4.6** (ACD-transformation). *Let $\mathcal{T}$ be a Muller transition system with alternating cycle decomposition $\mathcal{ACD}(\mathcal{T}) = \{(t_0, \nu_0), (t_1, \nu_1), \ldots, (t_r, \nu_r)\}$. We define its ACD-transformation $\mathcal{P}_{\mathcal{ACD}(\mathcal{T})} = (V_P, E_P, Source_P, Target_P, I_0', p : E_P \to \mathbb{N})$ as follows:*

For each state $q \in \mathcal{T}$ we consider the subtree t_q consisting of the nodes with q in its label, and we add a state for each branch of this subtree. For each initial state in $\mathcal{T}$, we choose one of its corresponding states in $\mathcal{P}_{\mathcal{ACD}(\mathcal{T})}$ and we set it as initial (the leftmost branch of t_q).

To define transitions in $\mathcal{P}_{\mathcal{ACD}(\mathcal{T})}$ we move simultaneously in $\mathcal{T}$ and in $\mathcal{ACD}(\mathcal{T})$. When we take a transition e in $\mathcal{T}$ that goes from q to q', while being in a branch β, we climb the branch β searching the lowest node τ with e and q' in its label (the support). We produce the priority corresponding to the level reached. If no such node exists in the branch β, we jump to the root of the tree containing q', producing the priority assigned to this root. After having *reached the support τ, we move to the next child of τ on the right of β in the tree $t_{q'}$, and we pick the leftmost branch under it in $t_{q'}$. If we had jumped to the root of $t_{q'}$ from a different tree, we just pick the leftmost branch of $t_{q'}$.*

For a formal definition we refer the reader to the full version of this paper [4].

In Figure 7 we show the ACD-transformation $\mathcal{P}_{\mathcal{ACD}(\mathcal{T})}$ of $\mathcal{T}$. States are labelled with the corresponding state q_j in $\mathcal{T}$, the tree of its index and a node $\tau \in t_i$ that is a leaf in t_{q_j}.

We have tagged the edges of $\mathcal{P}_{\mathcal{ACD}(\mathcal{T})}$ with names of edges from $\mathcal{T}$, in order to indicate the image of the edges by the morphism $\varphi : \mathcal{P}_{\mathcal{ACD}(\mathcal{T})} \to \mathcal{T}$.

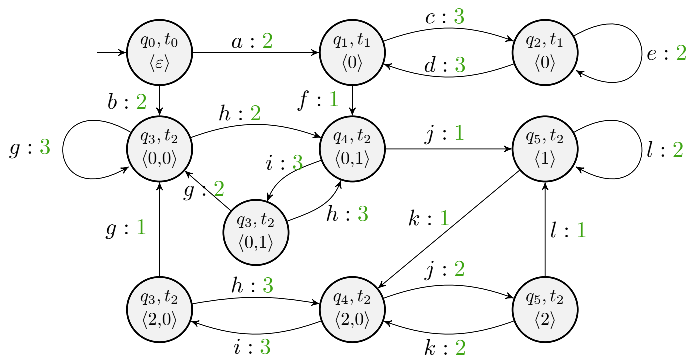


Figure 7 Transition system $\mathcal{P}_{\mathcal{ACD}(\mathcal{T})}$.

▶ **Proposition 4.7** (Correctness). *Let $\mathcal{T}$ be a (possibly labelled) Muller transition system and $\mathcal{P}_{\mathcal{ACD}(\mathcal{T})}$ its ACD-transformation. Then, there exists a locally bijective morphism (of labelled transition systems) $\varphi : \mathcal{P}_{\mathcal{ACD}(\mathcal{T})} \to \mathcal{T}$.*

4.3 Optimality of the alternating cycle decomposition transformation

▶ **Proposition 4.8** (Optimality of the number of priorities). *Let $\mathcal{T}$ be a Muller transition system and let $\mathcal{P}_{\mathcal{ACD}(\mathcal{T})}$ be its ACD-transition system. If $\mathcal{P}$ is another parity transition system such that there is a locally bijective morphism $\varphi : \mathcal{P} \to \mathcal{T}$, then $\mathcal{P}$ uses at least the same number of priorities than $\mathcal{P}_{\mathcal{ACD}(\mathcal{T})}$.*

In the case of deterministic automata, the results from [22] imply this proposition:

▶ **Proposition 4.9.** *If $\mathcal{A}$ is a deterministic Muller automaton, then $\mathcal{P}_{\mathcal{ACD}(\mathcal{A})}$ uses the optimal number of priorities to recognize $\mathcal{L}(\mathcal{A})$.*

Finally, we state the optimality of $\mathcal{P}_{\mathcal{ACD}(\mathcal{A})}$ for size.

▶ **Theorem 4.10** (Optimality of the number of states). *Let $\mathcal{T}$ be a Muller transition system and let $\mathcal{P}_{\mathcal{ACD}(\mathcal{T})}$ be its ACD-transition system. If $\mathcal{P}$ is another parity transition system such that there is a locally bijective morphism $\varphi : \mathcal{P} \to \mathcal{T}$, then $|\mathcal{P}_{\mathcal{ACD}(\mathcal{T})}| \leq |\mathcal{P}|$.*

The proof of Theorem 4.10 follows the same lines as for Theorem 3.7, we refer to the full version of this paper [4]. We note that from the hypothesis of Theorem 4.10 we cannot deduce that there is a morphism from $\mathcal{P}$ to $\mathcal{P}_{\mathcal{ACD}(\mathcal{T})}$ or vice-versa.

5 Applications

Determinisation of Büchi automata

The best theoretical bounds for the determinisation of Büchi automata are achieved by Piterman's construction [23]. In [26], Schewe revisits this construction and presents it as two consecutive steps: a first one producing a deterministic Rabin automaton $\mathcal{R}_\mathcal{B}$, and a second one transforming $\mathcal{R}_\mathcal{B}$ into a parity automaton $\mathcal{P}_\mathcal{B}$. This second step induces a locally bijective morphism from $\mathcal{P}_\mathcal{B}$ to $\mathcal{R}_\mathcal{B}$, therefore, thanks to Theorem 4.10 it is guaranteed that the ACD-transformation $\mathcal{P}_{\mathcal{ACD}(\mathcal{R}_\mathcal{B})}$ always yields a smaller deterministic parity automaton that uses less priorities. In particular, by Proposition 4.9 the number of priorities used by $\mathcal{P}_{\mathcal{ACD}(\mathcal{R}_\mathcal{B})}$ are the optimal one for recognising $\mathcal{L}(\mathcal{B})$ (that is, $\mathcal{ACD}(\mathcal{R}_\mathcal{B})$ gives the *parity index* of the language).

In many cases, the gain in both size and number of priorities is strict (we refer to the full version for one example [4]). However, both steps of Piterman Schewe's construction are already optimal in the worst case [6, 27], and applying the ACD-transformation in this worst-case example would generate the same parity automaton.

Relabelling of transition systems by acceptance conditions

We use the information provided by the alternating cycle decomposition to obtain results about the possibility of relabelling Muller transition systems with parity, Rabin and Streett conditions. The results presented here lift the seminal results of [30, Section 5] from conditions to transition systems.

Given a Zielonka tree $T_\mathcal{F}$, we say that it has *Rabin shape* (resp. *parity shape*) if every node with an even (reps. even or odd) priority assigned has at most one child. Given a Muller transition system $\mathcal{T}$, we say that its alternating cycle decomposition $\mathcal{ACD}(\mathcal{T})$ is a *Rabin ACD* (resp. *parity ACD*) if for every state $q \in V$, the tree t_q has Rabin shape (resp. parity shape).

▶ **Theorem 5.1.** *Let $\mathcal{T}$ be a Muller transition system. The following conditions are equivalent:*

1. *We can define a Rabin (resp. parity) condition that is equivalent to $\mathcal{F}$ over $\mathcal{T}$.*
2. *For every pair of loops $l_1, l_2 \in \mathit{Loop}(\mathcal{T})$, if $l_1 \notin \mathcal{F}$ and $l_2 \notin \mathcal{F}$ (resp. l_1 and l_2 are both in $\mathcal{F}$ or both in $\mathcal{P}(\Gamma) \setminus \mathcal{F}$), then $l_1 \cup l_2 \notin \mathcal{F}$ (resp. $l_1 \cup l_2 \in \mathcal{F} \Leftrightarrow l_1 \in \mathcal{F}$).*
3. *$\mathcal{ACD}(\mathcal{T})$ is a Rabin ACD (resp. parity ACD).*

By duality, a symmetric result of the Rabin case holds for Streett conditions.

Similar results can be obtained for weak automata, see the full version for details [4].

▶ **Corollary 5.2.** *Given a transition system graph $\mathcal{T}_G$ and a Muller condition $\mathcal{F} \subseteq \mathcal{P}(E)$, we can define a parity condition $p : E \to \mathbb{N}$ equivalent to $\mathcal{F}$ over $\mathcal{T}_G$ if and only if we can define both Rabin and Streett conditions over $\mathcal{T}_G$, R and S, such that $(\mathcal{T}_G, \mathcal{F}) \simeq (\mathcal{T}_G, R) \simeq (\mathcal{T}_G, S)$.*

The previous results are stated for non-labelled transition systems. We must be careful when translating these results to non-deterministic automata [1, Section 4]. However, Proposition 2.1 allows us to obtain analogous results for deterministic automata.

▶ **Corollary 5.3** (First proven in [1, Theorem 7])**.** *Let $\mathcal{A}$ be a deterministic automaton such that there are a Rabin condition R and a Streett condition S over $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}, R) = \mathcal{L}(\mathcal{A}, S)$. Then, there exists a parity condition p over $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}, p) = \mathcal{L}(\mathcal{A}, R) = \mathcal{L}(\mathcal{A}, S)$.*

6 Discussions

In this work we have introduced the alternating cycle decomposition of a transition system, uncovering the interplay between a transition system and its acceptance condition. In order to formalise the notion of a "transformation" we have introduced locally bijective morphisms, which open new lines of research concerning questions such as the complexity of minimising automata with respect to these morphisms. We formulate the following conjecture, which implies that lower bounds established for Muller, Rabin or Streett automata [6] yield lower bounds for parity automata.

▶ **Conjecture 6.1.** *If $\mathcal{A}$ is a minimal deterministic Muller (resp. Rabin) automaton recognising $\mathcal{L}(\mathcal{A})$, then $\mathcal{P}_{\mathcal{ACD}(\mathcal{A})}$ is a minimal deterministic parity automaton recognising $\mathcal{L}(\mathcal{A})$.*

References

1. Udi Boker, Orna Kupferman, and Avital Steinitz. Parityizing Rabin and Streett. In *FSTTCS*, pages 412–423, 2010. doi:10.4230/LIPIcs.FSTTCS.2010.412.
2. Olivier Carton and Ramón Maceiras. Computing the Rabin index of a parity automaton. *RAIRO: Theoretical Informatics and Applications*, pages 495–506, 1999. doi:10.1051/ita:1999129.
3. Olivier Carton and Max Michel. Unambiguous Büchi automata. *Theoretical Computer Science*, 297(1):37 – 81, 2003. doi:10.1016/S0304-3975(02)00618-7.
4. Antonio Casares, Thomas Colcombet, and Nathanaël Fijalkow. Optimal transformations of Muller conditions. *CoRR*, abs/2011.13041, 2020. arXiv:2011.13041.
5. Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors. *Handbook of Model Checking*. Springer, Cham, 2018. doi:10.1007/978-3-319-10575-8_2.
6. Thomas Colcombet and Konrad Zdanowski. A tight lower bound for determinization of transition labeled Büchi automata. In *ICALP*, pages 151–162, 2009. doi:10.1007/978-3-642-02930-1_13.
7. Stefan Dziembowski, Marcin Jurdziński, and Igor Walukiewicz. How much memory is needed to win infinite games? In *LICS*, pages 99–110, 1997. doi:10.1109/LICS.1997.614939.
8. Javier Esparza, Jan Křetínský, Jean-François Raskin, and Salomon Sickert. From LTL and limit-deterministic Büchi automata to deterministic parity automata. In *TACAS*, pages 426–442, 2017. doi:10.1007/978-3-662-54577-5_25.
9. Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *STOC*, pages 60–65, 1982. doi:10.1145/800070.802177.
10. Thomas Henzinger and Nir Piterman. Solving games without determinization. In *CSL*, pages 395–410, 2006. doi:10.1007/11874683_26.
11. Florian Horn. Explicit Muller games are PTIME. In *FSTTCS*, pages 235–243, 2008. doi:10.4230/LIPIcs.FSTTCS.2008.1756.
12. Paul Hunter and Anuj Dawar. Complexity bounds for regular games. In *MFCS*, pages 495–506, 2005. doi:10.1007/11549345_43.
13. Jan Křetínský, Tobias Meggendorfer, Clara Waldmann, and Maximilian Weininger. Index appearance record for transforming Rabin automata into parity automata. In *TACAS*, pages 443–460, 2017. doi:10.1007/978-3-662-54577-5_26.
14. Sriram C. Krishnan, Anuj Puri, and Robert K. Brayton. Structural complexity of omega-automata. In *STACS*, pages 143–156, 1995. doi:10.1007/3-540-59042-0_69.
15. Christof Löding and Anton Pirogov. Determinization of Büchi automata: Unifying the approaches of Safra and Muller-Schupp. In *ICALP*, pages 120:1–120:13, 2019. doi:10.4230/LIPIcs.ICALP.2019.120.

16 Michael Luttenberger, Philipp J. Meyer, and Salomon Sickert. Practical synthesis of reactive systems from LTL specifications via parity games. *Acta Informatica*, pages 3–36, 2020. doi:10.1007/s00236-019-00349-3.

17 Christof Löding. Optimal bounds for transformations of ω-automata. In *FSTTCS*, page 97–109, 1999. doi:10.1007/3-540-46691-6_8.

18 Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and control*, 9:521–530, 1966.

19 Philipp Meyer and Salomon Sickert. On the optimal and practical conversion of Emerson-Lei automata into parity automata. *Personal Communication*, 2021.

20 Thibaud Michaud and Maximilien Colange. Reactive synthesis from LTL specification with Spot. In *SYNT@CAV*, 2018.

21 David Müller and Salomon Sickert. LTL to deterministic Emerson-Lei automata. In *GandALF*, pages 180–194, 2017. doi:10.4204/EPTCS.256.13.

22 Damian Niwiński and Igor Walukiewicz. Relating hierarchies of word and tree automata. In *STACS*, pages 320–331, 1998. doi:10.1007/BFb0028571.

23 Nir Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *LICS*, pages 255–264, 2006. doi:10.1109/LICS.2006.28.

24 Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, page 179–190, 1989. doi:10.1145/75277.75293.

25 Schmuel Safra. On the complexity of ω-automata. In *FOCS*, page 319–327, 1988. doi:10.1109/SFCS.1988.21948.

26 Sven Schewe. Tighter bounds for the determinisation of Büchi automata. In *FoSSaCS*, pages 167–181, 2009. doi:10.1007/978-3-642-00596-1_13.

27 Sven Schewe and Thomas Varghese. Determinising parity automata. In *MFCS*, pages 486–498, 2014. doi:10.1007/978-3-662-44522-8_41.

28 Klaus Wagner. On ω-regular sets. *Information and Control*, 43(2):123–177, 1979. doi:10.1016/S0019-9958(79)90653-3.

29 Thomas Wilke and Haiseung Yoo. Computing the Rabin index of a regular language of infinite words. *Information and Computation*, pages 61–70, 1996. doi:10.1006/inco.1996.0082.

30 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:10.1016/S0304-3975(98)00009-7.