On the Minimisation of Transition-Based Rabin Automata and the Chromatic Memory Requirements of Muller Conditions

Antonio Casares*

LaBRI, Université de Bordeaux, France
antonio.casares-santos@labri.fr

Abstract

In this paper, we relate the problem of determining the chromatic memory requirements of Muller conditions with the minimisation of transition-based Rabin automata. Our first contribution is a proof of the NP-completeness of the minimisation of transition-based Rabin automata. Our second contribution concerns the memory requirements of games over graphs using Muller conditions. A memory structure is a finite state machine that implements a strategy and is updated after reading the edges of the game; the special case of chromatic memories being those structures whose update function only consider the colours of the edges. We prove that the minimal amount of chromatic memory required in games using a given Muller condition is exactly the size of a minimal Rabin automaton recognising this condition. Combining these two results, we deduce that finding the chromatic memory requirements of a Muller condition is NP-complete. This characterisation also allows us to prove that chromatic memories cannot be optimal in general, disproving a conjecture by Kopczyński.

1 Introduction

Games and memory. Automata on infinite words and infinite duration games over graphs are well established areas of study in Computer Science, being central tools used to solve problems such as the synthesis of reactive systems (see for example the Handbook [CHVB18]). Games over graphs are used to model the interaction between a system and the environment, and winning strategies can be used to synthesize controllers ensuring

*This work was done while the author was participating in the program *Theoretical Foundations of Computer Systems* at the Simons Institute for the Theory of Computing.

that the system satisfies some given specification. The games we will consider are played between two players (Eve and Adam), that alternatively move a pebble through the edges of a graph forming an infinite path. In order to define which paths are winning for the first player, Eve, we suppose that each transition in the game produces a colour in a set Γ, and a winning condition is defined by a subset $\mathbb{W} \subseteq \Gamma^\omega$. A fundamental parameter of the different winning conditions is the amount of memory that the players may require in order to define a winning strategy in games where they can force a victory. This parameter will influence the complexity of algorithms solving games that use a given winning condition, as well as the resources needed in a practical implementation of such a strategy as a controller for a reactive system.

A memory structure for Eve for a given game is a finite state machine that implements a strategy: for every position of the game, each state of the memory determines what move to perform next. After a transition of the game takes place, the memory state is updated according to an update function. We consider 3 types of memory structures:

- General memories.
- Chromatic memories: if the update function only takes as input the colour produced by the transition of the game.
- Arena-independent memories for a condition $\mathbb{W}$: if the memory structure can be used to implement winning strategies in any game using the condition $\mathbb{W}$.

In this work, we study these three notions of memories for Muller conditions, an important class of winning conditions that can be used to represent any ω-regular language via some deterministic automaton. Muller conditions appear naturally, for example, in the synthesis of reactive systems specified in Linear Temporal Logic [PR89, MS17].

In the seminal paper [DJW97], the authors establish the exact general memory requirements of Muller conditions, giving matching upper and lower bounds for every Muller condition in terms of its Zielonka tree. However, the memory structures giving the upper bounds are not chromatic. In his PhD thesis [Kop06, Kop08], Kopczyński raised the questions of whether minimal memory structures for a given game can be chosen to be chromatic, and whether arena-independent memories can be optimal, that is, if for each condition $\mathbb{W}$ there is a game won by Eve where the optimal amount of memory she can use is the size of a minimal arena-independent memory for $\mathbb{W}$. Another question appearing in [Kop06, Kop08] concerns the influence in the memory requirements of allowing or not ε-transitions in games (that is, transitions that do not produce any colour). In particular, Kopczyński asks whether all conditions that are half-positionally determined over transition-coloured games without ε-transitions are also half-positionally determined when allowing ε-transitions (it was already shown in [Zie98] that it is not the case in state-coloured games).

In this work, we characterise the minimal amount of chromatic memory required by Eve in games using a Muller condition as the size of a minimal deterministic transition-based Rabin automaton recognising the Muller condition, that can also be used as an arena-independent memory (Theorem 28); further motivating the study of the minimisation of transition-based Rabin automata. We prove that, in general, this quantity is strictly greater than the general memory requirements of the Muller condition, answering negatively the question by Kopczyński (Proposition 32). Moreover, we show that the general memory requirements of a Muller condition are different over ε-free games and over games with ε-transitions (Proposition 25), but that this is no longer the case when considering the chromatic memory requirements (Theorem 28). In particular, in order to obtain the lower bounds of [DJW97] we need to use ε-transitions. However, the question stated in [Kop06, Kop08] of whether allowing ε-transitions could have an impact on the half-positionality of conditions remains open, since it cannot be the case for Muller conditions (Lemma 24).

Minimisation of transition-based automata. Minimisation is a well studied problem for many classes of automata. Automata over finite words can be minimised in polynomial time [Hop71], and for every regular language there is a canonical minimal automaton recognising it. For automata over infinite words, the status of the minimisation problem for different models of ω-automata is less well understood. Traditionally, the acceptance conditions of ω-automata have been defined over the set of states; however, the use of transition-based automata is becoming common in both practical and theoretical applications (see for instance [GL02]), and there is evidence that decision problems relating to transition-based models might be easier than the corresponding problems for state-based ones. The minimisation of state-based Büchi automata has been proven to be NP-complete by Schewe (therefore implying the NP-hardness of the minimisation of state-based parity, Rabin and Streett automata), both for deterministic [Sch10] and Good-For-Games (GFG) automata [Sch20]. However, these reductions strongly use the fact that the acceptance condition is defined over the states and not over the transitions. Abu Radi and Kupferman have proven that the minimisation of GFG-transition-based co-Büchi automata can be done in polynomial time and that a canonical minimal GFG-transition-based automaton can be defined for co-Büchi languages [AK19, AK20]. This suggests that transition-based automata might be a more adequate model for ω-automata, raising many questions about the minimisation of different kinds of transition-based automata (Büchi, parity, Rabin, GFG-parity, etc). Moreover, Rabin automata are of great interest, since the determinization of Büchi automata via Safra's construction naturally provides deterministic transition-based Rabin automata [Saf88, Sch09], and, as proven in Theorem 28, these automata provide minimal arena-independent memories for Muller games.

In Section 2.2, we prove that the minimisation of transition-based Rabin automata is NP-complete (Theorem 14). The proof consists in a reduction from the chromatic number problem of graphs. This reduction uses a particularly simple family of ω-regular languages: languages $L \subseteq \Sigma^\omega$ that correspond to Muller conditions, that is, whether a word $w \in \Sigma^\omega$ belongs to L or not only depends in the set of letters appearing infinitely often in w (we called them Muller languages). A natural question is whether we can extend this reduction to prove the NP-hardness of the minimisation of other kinds of transition-based automata, like parity or generalised Büchi ones. However, we prove in Section 2.3 that the minimisation of parity and generalised Büchi automata recognising Muller languages can be done in polynomial time. This is based in the fact that the minimal parity automaton recognising a Muller condition is given by the Zielonka tree of the condition [CCF21, MS21].

These results allow us to conclude that determining the chromatic memory requirements of a Muller condition is NP-complete even if the condition is represented by its Zielonka tree (Theorem 31). This is a surprising result, since the Zielonka tree of a Muller condition allows to compute in linear time the non-chromatic memory requirements of it [DJW97].

Related work. As already mentioned, the works [DJW97, Kop06, Zie98] extensively study the memory requirements of Muller conditions. In the paper [CN06], the authors characterise parity conditions as the only prefix-independent conditions that admit positional strategies over transition-coloured infinite graphs. This characterisation does not apply to state-coloured games, which supports the idea that transition-based systems might present more canonical properties. Conditions that admit arena-independent memories are characterised in [BRO$^+$20], extending the work of [GZ05] characterising conditions that accept positional strategies over finite games. The memory requirements of generalised safety conditions have been established in [CFH14]. The use of Rabin automata as memories for games with ω-regular conditions have been fruitfully used in [CZ09] in order to obtain theoretical lower bounds on the size of deterministic Rabin automata obtained by the process of determinisation of Büchi automata.

Concerning the minimisation of automata over infinite words, beside the aforementioned results of [Sch10, Sch20, AK19], it is also known that weak automata can be minimised in $\mathcal{O}(n \log n)$ [Löd01]. The algorithm minimising a parity automaton recognising a Muller language presented in the proof of

Proposition 17 can be seen as a generalisation of the algorithm appearing in [CM99] computing the Rabin index of a parity automaton. Both of them have their roots in the work of Wagner [Wag79].

Organisation of this paper. In Section 2 we discuss the minimisation of transition-based Rabin and parity automata. We give the necessary definitions in Section 2.1, in Section 2.2 we show the NP-completeness of the minimisation of Rabin automata and in Section 2.3 we prove that we can minimise transition-based parity and generalised Büchi automata recognising Muller languages in polynomial time.

In Section 3 we introduce the definitions of games and memory structures, and we discuss the impact on the memory requirements of allowing or not ε-transitions in the games.

In Section 4, the main contributions concerning the chromatic memory requirements of Muller conditions are presented.

2 Minimising transition-based automata

In this section, we present our main contributions concerning the minimisation of transition-based automata. We start in Section 2.1 by giving some basic definitions and results related to automata used throughout the paper. In Section 2.2 we show a reduction from the problem of determining the chromatic number of a graph to the minimisation of Rabin automata, proving the NP-completeness of the latter. Moreover, the languages used in this proof are Muller languages. In Section 2.3 we prove that, on the contrary, we can minimise parity and generalised Büchi automata recognising Muller languages in polynomial time.

2.1 Automata over infinite words

General notations

The greek letter ω stands for the set $\{0, 1, 2, \dots\}$. Given a set A, we write $\mathcal{P}(A)$ to denote its power set and $|A|$ to denote its cardinality. A word over an alphabet Σ is a sequence of letters from Σ. We let Σ^* and Σ^ω be the set of finite and infinite words over Σ, respectively. For an infinite word $w \in \Sigma^\omega$, we write *$Inf(w)$* to denote the set of letters that appear infinitely often in w. We will extend functions $\gamma : A \to \Gamma$ to A^*, A^ω and $\mathcal{P}(A)$ in the natural way, without explicitly stating it.

A (directed) *graph* $G = (V, E)$ is given by a set of vertices V and a set of edges $E \subseteq V \times V$. A graph $G = (V, E)$ is *undirected* if every pair of vertices (v, u) verifies $(v, u) \in E \iff (u, v) \in E$. A graph $G = (V, E)$ is *simple* if $(v, v) \notin E$ for any $v \in V$. A subgraph $\mathcal{S}$ of G is *strongly connected* if for every pair of vertices v_1, v_2 in $\mathcal{S}$, there is a path in $\mathcal{S}$ from v_1 to v_2. A *strongly* *connected component* of G is a maximal strongly connected subgraph of G. We say that a strongly connected component $\mathcal{S}$ is *ergodic* if no vertex of $\mathcal{S}$ has outgoing edges leading to vertices not in $\mathcal{S}$.

A *coloured graph* $G = (V, E)$ is given by a set of vertices V and a set of edges $E \subseteq V \times C_1 \times \cdots \times C_k \times V$, where $C_1, \dots, C_k$ are sets of colours.

Automata

An *automaton* is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Gamma, Acc)$, where Q is a finite set of states, Σ is a finite input alphabet, $q_0 \in Q$ is an initial state, $\delta : Q \times \Sigma \to Q \times \Gamma$ is a transition function, Γ is an output alphabet and Acc is an accepting condition defining a subset $\mathbb{W} \subseteq \Gamma^\omega$ (the conditions will be defined more precisely in the next paragraph). In this paper, all automata will be deterministic and complete (that is, δ is a function) and transition-based (that is, the output letter that is produced depends on the transition, and not only on the arrival state). The *size* of an automaton is its number of states, $|Q|$.

Given an input word $w = w_0 w_1 w_2 \cdots \in \Sigma^\omega$, the *run over* w in $\mathcal{A}$ is the only sequence of pairs $(q_0, c_0), (q_1, c_1), \dots \in Q \times \Gamma$ verifying that q_0 is the initial state and $\delta(q_i, w_i) = (q_{i+1}, c_i)$. The *output* produced by w is the word $c_0 c_1 c_2 \cdots \in \Gamma^\omega$. A word $w \in \Sigma^\omega$ is *accepted by* the automaton $\mathcal{A}$ if its output belongs to the set $\mathbb{W} \subseteq \Gamma^\omega$ defined by the accepting condition. The *language accepted by* an automaton $\mathcal{A}$, written $\mathcal{L}(\mathcal{A})$, is the set of words accepted by $\mathcal{A}$. Given two automata $\mathcal{A}$ and $\mathcal{B}$ over the same input alphabet Σ, we say that they are *equivalent* if $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$.

Given an automaton $\mathcal{A}$, the *graph associated to* $\mathcal{A}$, denoted $G(\mathcal{A})$, is the coloured graph $G(\mathcal{A}) = (Q, E_{\mathcal{A}})$, whose set of vertices is Q, and the set of edges $E_{\mathcal{A}} \subset Q \times \Sigma \times \Gamma \times Q$ is given by $(q, a, c, q') \in E_{\mathcal{A}}$ if $\delta(q, a) = (q', c)$. We denote by $\iota \colon E_{\mathcal{A}} \to \Sigma$ the projection over the second component and by $\gamma \colon E_{\mathcal{A}} \to \Gamma$ the projection over the third one.

A *cycle* of an automaton $\mathcal{A}$ is a subset of edges $\ell \subseteq E_{\mathcal{A}}$ such that there is a state $q \in Q$ and a path in $G(\mathcal{A})$ starting and ending in q passing through exactly the edges in ℓ. We write $\gamma(\ell) = \bigcup_{e \in \ell} \gamma(e)$ to denote the set of colours appearing in the cycle ℓ. A state $q \in Q$ is contained in a cycle $\ell \subseteq E_{\mathcal{A}}$ if there is some edge in ℓ whose first component is q. We write $\mathit{States}(\ell)$ to denote the set of states contained in ℓ.

Acceptance conditions

Let Γ be a set of colours. We define next some of the acceptance conditions used to define subsets $\mathbb{W} \subseteq \Gamma^\omega$. All the subsequent conditions verify that the acceptance of a word $u \in \Gamma^\omega$ only depends on the set $\mathit{Inf}(u)$.

Muller. A *Muller condition* is given by a family of subsets $\mathcal{F} = \{S_1, \dots, S_k\}$, $S_i \subseteq \Gamma$. A word $u \in \Gamma^\omega$ is accepting if $\mathit{Inf}(u) \in \mathcal{F}$.

Rabin. A *Rabin condition* is represented by a family of *Rabin pairs*, $R = \{(E_1, F_1), \ldots, (E_r, F_r)\}$, where $E_i, F_i \subseteq \Gamma$. A word $u \in \Gamma^\omega$ is accepting if $\mathit{Inf}(u) \cap E_i \neq \emptyset$ and $\mathit{Inf}(u) \cap F_i = \emptyset$ for some index $i \in \{1, \ldots, r\}$.

Streett. A *Streett condition* is represented by a family of pairs $S = \{(E_1, F_1), \ldots, (E_r, F_r)\}$, $E_i, F_i \subseteq \Gamma$. A word $u \in \Gamma^\omega$ is accepting if $\mathit{Inf}(u) \cap E_i \neq \emptyset \rightarrow \mathit{Inf}(u) \cap F_i \neq \emptyset$ for every $i \in \{1, \ldots, r\}$.

Parity. To define a *parity condition* we suppose that Γ is a finite subset of $\mathbb{N}$. A word $u \in \Gamma^\omega$ is accepting if $\max \mathit{Inf}(u)$ is even. The elements of Γ are called *priorities* in this case.

Generalised Büchi. A *generalised Büchi condition* is represented by a family of subsets $\{B_1, \ldots, B_r\}$, $B_i \subseteq \Gamma$. A word $u \in \Gamma^\omega$ is accepted if $\mathit{Inf}(u) \cap B_i \neq \emptyset$ for all $i \in \{1, \ldots, r\}$.

Generalised co-Büchi. A *generalised co-Büchi condition* is represented by a family of subsets $\{B_1, \ldots, B_r\}$, $B_i \subseteq \Gamma$. A word $u \in \Gamma^\omega$ is accepted if $\mathit{Inf}(u) \cap B_i = \emptyset$ for some $i \in \{1, \ldots, r\}$.

An automaton $\mathcal{A}$ using a condition of type X will be called an X-automaton.

We remark that all the previous conditions define a family of subsets $\mathcal{F} \subseteq \mathcal{P}(\Gamma)$ and can therefore be represented as Muller conditions (in particular, all automata referred to in this paper can be regarded as Muller automata). Also, parity conditions can be represented as Rabin or Streett ones. We say that a language $L \subseteq \Gamma^\omega$ is a *Muller language* if $u_1 \in L$ and $u_2 \notin L$ implies that $\mathit{Inf}(u_1) \neq \mathit{Inf}(u_2)$. We associate to each Muller condition $\mathcal{F}$ the language $L_\mathcal{F} = \{w \in \Gamma^\omega : \mathit{Inf}(w) \in \mathcal{F}\}$.

The *parity index* (also called *Rabin index*) of an ω-regular language $L \subseteq \Sigma^\omega$ is the minimal $p \in \mathbb{N}$ such that there exists a deterministic parity automaton recognising L using p priorities in its parity condition.

Given an ω-regular language $L \subseteq \Sigma^\omega$, we write $\mathfrak{rabin}(L)$ to denote the size of a minimal Rabin automaton recognising L.

Remark 1. *Let $\mathcal{A}$ be a Rabin-automaton recognising a language $L \subseteq \Sigma^\omega$ using Rabin pairs $R = \{(E_1, F_1), \ldots, (E_r, F_r)\}$. If we consider the Streett automaton obtained by setting the pairs of R as defining a Streett condition over the structure of $\mathcal{A}$, we obtain a Streett automaton $\mathcal{A}'$ recognising the language $\Sigma^\omega \setminus L$ (and vice versa). Therefore, the size of a minimal Rabin automaton recognising L coincides with that of a minimal Streett automaton recognising $\Sigma^\omega \setminus L$, and the minimisation problem for both classes of automata is equivalent. Similarly for generalised Büchi and generalised co-Büchi automata.*

Let $\mathcal{A}$ be an automaton using some of the acceptance conditions above defining a family $\mathcal{F} \subseteq \mathcal{P}(\Gamma)$. We remark that since the acceptance of a run only depends on the set of colours produced infinitely often, we can associate to each cycle ℓ of $\mathcal{A}$ an accepting or rejecting status. We say that a cycle ℓ of $\mathcal{A}$ is *accepting* if $\gamma(\ell) \in \mathcal{F}$ and that it is *rejecting* otherwise.

We are going to be interested in simplifying the acceptance conditions of automata, while preserving their structure. We say that we can *define a condition of type X on top of a Muller automaton $\mathcal{A}$* if we can recolour the transitions of $\mathcal{A}$ with colours in a set Γ' and define a condition of type X over Γ' such that the resulting automaton is equivalent to $\mathcal{A}$. Definition 2 formalises this notion.

Definition 2. *Let X be some of the conditions defined previously and let $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Gamma, \mathcal{F})$ be a Muller automaton. We say that we can define a condition of type X on top of $\mathcal{A}$ if there is an X-condition over a set of colours Γ' and an automaton $\mathcal{A}' = (Q, \Sigma, q_0, \delta', \Gamma', X)$ verifying:*

- *$\mathcal{A}$ and $\mathcal{A}'$ have the same set of states and the same initial state.*
- *$\delta(q, a) = (p, c) \;\Rightarrow \delta'(q, a) = (p, c')$, for some $c' \in \Gamma'$, for every $q \in Q$ and $a \in \Sigma$ (that is, $\mathcal{A}$ and $\mathcal{A}'$ have the same transitions, except for the colours produced).*
- *$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

The next proposition, proven in [CCF21], characterises automata that admit Rabin conditions on top of them. It will be a key property used throughout the paper.

Proposition 3 ([CCF21])**.** *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Gamma, \mathcal{F})$ be a Muller automaton. The following properties are equivalent:*

1. *We can define a Rabin condition on top of $\mathcal{A}$.*
2. *Any pair of cycles ℓ_1 and ℓ_2 in $\mathcal{A}$ verifying $\mathit{States}(\ell_1) \cap \mathit{States}(\ell_2) \neq \emptyset$ satisfies that if both ℓ_1 and ℓ_2 are rejecting, then $\ell_1 \cup \ell_2$ is also a rejecting cycle.*

The Zielonka tree of a Muller condition

In order to study the memory requirements of Muller conditions, Zielonka introduced in [Zie98] the notion of split trees (later called Zielonka trees) of Muller conditions. The Zielonka tree of a Muller condition naturally provides a minimal parity automaton recognising this condition [CCF21, MS21]. We will use this property to show that parity automata recognising Muller languages can be minimised in polynomial time in Proposition 17. We will come back to Zielonka trees in Section 4 to discuss the characterisation of the memory requirements of Muller conditions.

Definition 4. *Let Γ be a set of labels. We give the definition of a* Γ-labelled-tree *by induction:*

- *$T = \langle A, \langle \emptyset \rangle \rangle$ is a Γ-labelled-tree for any $A \subseteq \Gamma$. In this case, we say that T is a* leaf *and A is its label.*
- *If $T_1, \dots, T_n$ are Γ-labelled-trees, then $T = \langle A, \langle T_1, \dots, T_n \rangle \rangle$ is a Γ-labelled-tree for any $A \subseteq \Gamma$. In that case, we say that A is the label of T and $T_1, \dots, T_n$ are their children.*

The set of nodes *of a tree T is defined recursively as:*

$$Nodes(T) = \{T\} \cup \bigcup_{T' \text{ child of } T} Nodes(T').$$

Definition 5 ([Zie98])**.** *Let $\mathcal{F} \subseteq \mathcal{P}(\Gamma)$ be a Muller condition. The* Zielonka tree *of $\mathcal{F}$, denoted $\mathcal{Z}_{\mathcal{F}}$, is the Γ-labelled-tree defined recursively as follows: let $A_1, \dots A_k$ be the maximal subsets of Γ such that $A_i \in \mathcal{F} \Leftrightarrow \Gamma \notin \mathcal{F}$.*

- *If no such subset $A_i \subseteq \Gamma$ exists, then $\mathcal{Z}_{\mathcal{F}} = \langle \Gamma, \langle \emptyset \rangle \rangle$.*
- *Otherwise, $\mathcal{Z}_{\mathcal{F}} = \langle \Gamma, \langle \mathcal{Z}_{\mathcal{F}_1}, \dots, \mathcal{Z}_{\mathcal{F}_k} \rangle \rangle$, where $\mathcal{Z}_{\mathcal{F}_i}$ is the Zielonka tree for the condition $\mathcal{F}_i = \mathcal{F} \cap \mathcal{P}(A_i)$ over the colours A_i.*

An example of a Zielonka tree can be found in Figure 2 (page 28).

Proposition 6 ([CCF21], [MS21])**.** *Let $\mathcal{F}$ be a Muller condition and $\mathcal{Z}_{\mathcal{F}}$ its Zielonka tree. We can build in linear time in the representation of $\mathcal{Z}_{\mathcal{F}}$ a parity automaton recognising $L_{\mathcal{F}}$ that has as set of states the leaves of $\mathcal{Z}_{\mathcal{F}}$. This automaton is minimal, that is, any other parity automaton recognising $L_{\mathcal{F}}$ has at least as many states as the number of leaves of $\mathcal{Z}_{\mathcal{F}}$.*

2.2 Minimising transition-based Rabin and Streett automata is NP-complete

This section is devoted to proving the NP-completeness of the minimisation of transition-based Rabin automata, stated in Theorem 14.

For the containment in NP, we use the fact that we can test language equivalence of Rabin automata in polynomial time.

Proposition 7 ([CDK93])**.** *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two Rabin automata over Σ. We can decide in polynomial time on the representation of the automata if $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$. (We recall that all considered automata are deterministic).*

Corollary 8. *Given a Rabin automaton $\mathcal{A}$ and a positive integer k, we can decide in non-deterministic polynomial time whether there is an equivalent Rabin automaton of size k.*

Proof. A non-deterministic Turing machine just has to guess an equivalent automaton $\mathcal{A}_k$ of size k, and by Proposition 7 it can check in polynomial time whether $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_k)$. $\square$

In order to prove the NP-hardness, we will describe a reduction from the `Chromatic-Number` problem (one of 21 Karp's NP-complete problems) to the minimisation of transition-based Rabin automata. Moreover, this reduction will only use languages that are Muller languages of Rabin index 3.

Definition 9. *Let $G = (V, E)$ be a simple undirected graph. A colouring of size k of G is a function $c : V \to [1, k]$ such that for any pair of vertices $v, v' \in V$, if $(v, v') \in E$ then $c(v) \neq c(v')$.*

The chromatic number of a simple undirected G, written $\chi(G)$, is the smallest number k such that there exists a colouring of size k of G.

Lemma 10 ([Kar72])**.** *Deciding whether a simple undirected graph has a colouring of size k is* NP*-complete.*

Let $G = (V, E)$ be a simple undirected graph, n be its number of vertices and m its number of edges. We consider the language L_G over the alphabet V given by:

$$L_G = \bigcup_{(v,u)\in E} V^*(v^+u^+)^\omega.$$

That is, a sequence $w \in V^\omega$ is in L_G if eventually it alternates between exactly two vertices connected by an edge in G.

Remark 11. *For any simple undirected graph G, L_G is a Muller language over V, that is, whether a word $w \in V^\omega$ belongs to L_G or not only depends on Inf(w). Moreover, the parity index of this condition is at most* [1-3]*.*

However, we cannot extend this reduction to show NP-hardness of the minimisation of transition-based parity automata, as we will show in Section 2.3 that we can minimise parity automata recognising Muller languages in polynomial time.

Lemma 12. *We can build a Rabin automaton $\mathcal{A}_G$ of size n over the alphabet V recognising L_G in $\mathcal{O}(mn^2)$.*

Proof. We define the automaton $\mathcal{A}_G = (Q, V, q_0, \delta, V \times V, R)$ as follows:

- $Q = V$.
- q_0 an arbitrary vertex in Q.
- $\delta(v, x) = (x, (v, x))$, for $v, x \in V$.

- $R = \{(E_{(v,u)}, F_{(v,u)}) : (v,u) \in E\}$, where we define for each $(v,u) \in E$ the sets $E_{(v,u)}, F_{(v,u)}$ as:
  - $E_{(v,u)} = \{(v,u)\}$
  - $F_{(v,u)} = V \times V \setminus \{(v,u),(u,v),(v,v),(u,u)\}$.

That is, the states of the automaton are the vertices of the graph G, and when we read a letter $u \in V$ we jump to the state u. The colours defining the Rabin condition are all pairs of vertices, and we define one Rabin pair for each edge of the graph. This Rabin pair will enable to accept words that eventually alternate between the endpoints of the edge.

We prove that $L_G = \mathcal{L}(\mathcal{A}_G)$. If $w \in L_G$, the word w is eventually of the form $(v^+u^+)^\omega$ for some $v,u \in V$ such that $(v,u) \in E$, so eventually we will only visit the states v and u of the automaton. If we are in v and read the letter v, we produce the colour (v,v) that is not contained in the set $F_{(v,u)}$. If we read the letter u, we will produce (v,u), contained in $E_{(v,u)}$ and not appearing in $F_{(v,u)}$ nor in $F_{(u,v)}$. The behaviour is symmetric from the state u. Let us denote $\alpha \in E^\omega$ the word output by the automaton reading w. We obtain that $\mathit{Inf}(\alpha) \cap E_{(v,u)} \neq \emptyset$ and $\mathit{Inf}(\alpha) \cap F_{(v,u)} = \emptyset$, so the word w is accepted by the automaton.

Conversely, if a word w is accepted by $\mathcal{A}_G$, then the output $\alpha \in (V \times V)^\omega$ must verify $\mathit{Inf}(\alpha) \cap F_{(v,u)} = \emptyset$ for some $(v,u) \in E$, so eventually α only contains the pairs $(v,u),(u,v),(v,v)$ and (u,u). If w was eventually of the form v^ω or u^ω, we would have that $\mathit{Inf}(\alpha) \cap E_{(v,u)} = \emptyset$ for all $(v,u) \in E$. We conclude that the word w eventually alternates between two vertices connected by an edge, so $w \in L_G$.

The automaton has n states, the transition function δ has size $\mathcal{O}(n^2)$ and the Rabin condition R has size $\mathcal{O}(mn^2)$. ☐

Lemma 13. *Let $G = (V,E)$ be a simple undirected graph. Then, the size of a minimal Rabin automaton recognising L_G coincides with the chromatic number of G.*

Proof. We denote $\mathfrak{rabin}(L_G)$ the size of a minimal Rabin automaton recognising L_G and $\chi(G)$ the chromatic number of G.

$\mathfrak{rabin}(L_G) \leq \chi(G)$: Let $c : V \to [1,k]$ be a colouring of size k of G. We will define a Muller automaton of size k recognising L_G and then use Proposition 3 to show that we can put a Rabin condition on top of it. Let $\mathcal{A}_c = (Q, V, q_0, \delta, V, \mathcal{F})$ be the Muller automaton defined by:

- $Q = \{1,2,\dots,k\}$.
- $q_0 = 1$.
- $\delta(q,x) = (c(x), x)$ for $q \in Q$ and $x \in V$.

- A set $C \subseteq V$ belongs to $\mathcal{F}$ if and only if $C = \{v, u\}$ for two vertices $v, u \in V$ such that $(v, u) \in E$.

The language recognised by $\mathcal{A}_c$ is clearly L_G, since the output produced by a word $w \in V^\omega$ is w itself, and the acceptance condition $\mathcal{F}$ is exactly the Muller condition defining the language L_G.

Let $G(\mathcal{A}_c) = (Q, E_{\mathcal{A}_c})$ be the graph associated to $\mathcal{A}_c$. We will prove that the union of any pair of rejecting cycles of $\mathcal{A}_c$ that have some state in common must be rejecting. By Proposition 3 this implies that we can define a Rabin condition on top of $\mathcal{A}_c$.

Let $\ell_1, \ell_2 \subseteq E_{\mathcal{A}_c}$ be two cycles such that $\gamma(\ell_i) \notin \mathcal{F}$ for $i \in \{1, 2\}$ and such that $\mathit{States}(\ell_1) \cap \mathit{States}(\ell_2) \neq \emptyset$. We distinguish 3 cases:

- $|\gamma(\ell_i)| \geq 3$ for some $i \in \{1, 2\}$. In this case, their union also has more than 3 colours, so it must be rejecting.
- $\gamma(\ell_i) = \{u, v\}$, $(u, v) \notin E$ for some $i \in \{1, 2\}$. In that case, $\gamma(\ell_1 \cup \ell_2)$ also contains two vertices that are not connected by an edge, so it must be rejecting.
- $\gamma(\ell_1) = \{v_1\}$ and $\gamma(\ell_2) = \{v_2\}$. In this case, since from every state q of $\mathcal{A}_c$ and every $v \in V$ $\delta(q, v) = (c(v), v)$, the only state in each cycle is, respectively, $c(v_1)$ and $c(v_2)$. As ℓ_1 and ℓ_2 share some state, we deduce that $c(v_1) = c(v_2)$. If $v_1 = v_2$, $\ell_1 \cup \ell_2$ is rejecting because $|\gamma(\ell_1 \cup \ell_2)| = 1$. If $v_1 \neq v_2$, it is also rejecting because $c(v_1) = c(v_2)$ and therefore $(v_1, v_2) \notin E$.

Since $\gamma(\ell_i)$ is rejecting, it does not consist on two vertices connected by some edge and we are always in some of the cases above. We conclude that we can put a Rabin condition on top of $\mathcal{A}_c$, obtaining a Rabin automaton recognising L_G of size k.

$\chi(G) \leq \mathfrak{rabin}(L_G)$: Let $\mathcal{A} = (Q, V, q_0, \delta, \Gamma, R)$ be a Rabin automaton of size k recognising L_G and $G(\mathcal{A}) = (Q, E_{\mathcal{A}})$ its graph. We will define a colouring of size k of G, $c : V \to Q$.

For each $v \in V$ we define a subset $Q_v \subseteq Q$ as:

$$Q_v = \{q \in Q \ : \ \text{there is a cycle } \ell \text{ containing } q \text{ and } \gamma(\ell) = \{v\}\}.$$

For every $v \in V$, the set Q_v is non-empty, as it must exist a (non-accepting) run over v^ω in $\mathcal{A}$. For each $v \in V$ we pick some $q_v \in Q_v$, and we define the colouring $c : V \to Q$ given by $c(v) = q_v$.

In order to prove that it is indeed a colouring, we we will show that any two vertices $v, u \in V$ such that $(v, u) \in E$ verify that $Q_v \cap Q_u = \emptyset$, and therefore they also verify $c(v) \neq c(u)$. Suppose by contradiction that there was some $q \in Q_v \cap Q_u$. Let us write ℓ_x for a cycle containing q labelled with x, for $x \in \{v, u\}$. By the definition of L_G, both cycles ℓ_v and ℓ_u have to be rejecting as $x^\omega \notin L_G$ for any $x \in V$. However, since $(u, v) \in E$, their union would be accepting, contradicting Proposition 3. $\square$

These results allow us to deduce the NP-completeness of the minimisation of Rabin automata.

Theorem 14. *Given a Rabin automaton $\mathcal{A}$ and a positive integer k, deciding whether there is an equivalent Rabin automaton of size k is* NP*-complete.*

Corollary 15. *Given a Streett automaton $\mathcal{A}$ and a positive integer k, deciding whether there is an equivalent Streett automaton of size k is* NP*-complete.*

2.3 Parity and generalised Büchi automata recognising Muller languages can be minimised in polynomial time

In Section 2.2 we have proven the NP-hardness of the minimisation of Rabin automata showing a reduction that uses a language that is a Muller language, that is, whether an infinite word w belongs to the language only depends on *Inf*(w). We may wonder whether Muller languages could be used to prove NP-hardness of minimising parity or generalised Büchi automata. We shall see now that this is not the case, as we prove in Propositions 17 and 18 that we can minimise parity and generalised Büchi automata recognising Muller languages in polynomial time. Before proving these propositions we fix some notations.

For the rest of the section, let Σ stand for an input alphabet, $\mathcal{F}$ for a Muller condition over Σ and let $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Gamma, P)$ be a parity automaton recognising $L_\mathcal{F}$, with Γ a finite subset of $\mathbb{N}$. Let $G(\mathcal{A}) = (V_\mathcal{A}, E_\mathcal{A})$ be the graph associated to $\mathcal{A}$, and ι and γ the projections over the input and output alphabet, respectively. We can suppose without loss of generality that $G(\mathcal{A})$ is strongly connected, since any ergodic strongly connected component of it must recognise the Muller language $L_\mathcal{F}$. We call a subgraph $\mathcal{S} = (V_S, E_S)$ of $G(\mathcal{A})$ an *$\mathcal{A}$-subgraph* and we write *Letters*$(\mathcal{S}) = \bigcup_{e \in E_S} \iota(e)$ and *Max-Priority*$(\mathcal{S}) = \max\limits_{e \in E_S} \gamma(e)$.

We say that $\mathcal{S}$ is *complete* if for all vertices $v \in V_S$ and letters $a \in$ *Letters*$(\mathcal{S})$, v has an outgoing edge e in $\mathcal{S}$ such that $\iota(e) = a$. Let $C \subseteq \Sigma$ be a set of letters; we say that a subgraph $\mathcal{S}$ of $G(\mathcal{A})$ is a *C-Strongly Connected Component* of $\mathcal{A}$ (C-SCC) if it is a strongly connected subgraph, complete and *Letters*$(\mathcal{S}) = C$.

Fact 1. *A C-SCC of $\mathcal{A}$ is naturally a parity automaton recognising the Muller condition $\mathcal{F}|_C = \mathcal{F} \cap \mathcal{P}(C)$ over C.*

Lemma 16. *For every subset of letters $C \subseteq \Sigma$ there is some C-SCC in $\mathcal{A}$.*

Proof. It suffices to take an ergodic component of the graph obtained by restricting the transitions of $\mathcal{A}$ to those labelled with letters in C. □

We denote `SCC-Decomposition`$(\mathcal{S})$ a procedure that returns a list of the strongly connected components of an $\mathcal{A}$-subgraph $\mathcal{S}$ (what can be done in linear time in the number of vertices and edges [Tar72],[Sha81]).

We denote `Complete-SCC`$(C, \mathcal{S})$ a procedure that takes as input a set of letters $C \subseteq \Sigma$ and an $\mathcal{A}$-subgraph that contains some C-SCC and it returns in linear time a C-SCC of $\mathcal{S}$. We can do it simply by taking the restriction of $\mathcal{S}$ to edges labelled with letters in C, computing a strongly connected component decomposition of it and choosing some ergodic component of it.

The procedure `MaxInclusion`(F) takes as input a family of subsets $F \subseteq \mathcal{P}(\Sigma)$ and returns a list containing the elements of F that are maximal with respect to inclusion. This can be done in $\mathcal{O}(|\Sigma|^2|F|^2)$.

Proposition 17. *Let $\mathcal{F} \subseteq \mathcal{P}(\Sigma)$ be a Muller condition. Given a parity automaton recognising $L_{\mathcal{F}}$, we can build in polynomial time a minimal parity automaton recognising $L_{\mathcal{F}}$.*[1]

Proof. By Proposition 6, we know that a minimal parity automaton recognising the Muller language associated to the condition $\mathcal{F} \subseteq \mathcal{P}(\Sigma)$ can be constructed in linear time from its Zielonka tree, $\mathcal{Z}_{\mathcal{F}}$. Let $\mathcal{A}$ be a parity automaton recognising $L_{\mathcal{F}}$. We give an algorithm (Algorithm 1) building $\mathcal{Z}_{\mathcal{F}}$ that works in $\mathcal{O}(d^2c^2n^4)$, where d is the number of priorities in $\mathcal{A}$, $c = |\Sigma|$ and $n = |\mathcal{A}|$. This algorithm builds $\mathcal{Z}_{\mathcal{F}}$ recursively: first it calls the sub-procedure `AlternatingSets` (Algorithm 2) that returns the labels of the children of the root of $\mathcal{Z}_{\mathcal{F}}$ and then it uses `Complete-SCC` to compute a C-SCC for each label C of the children.

Correctness of the algorithm. Let T be the labelled tree returned by the algorithm `Zielonka-Tree`$(G(\mathcal{A}))$. We prove that T is the Zielonka tree of $\mathcal{F}$ by induction. We suppose without loss of generality that $\Sigma \in \mathcal{F}$, which implies that the maximal priority p in $\mathcal{A}$ is even (as we have supposed that $\mathcal{A}$ is strongly connected).

If $\mathcal{Z}_{\mathcal{F}} = \langle \Sigma, \langle \emptyset \rangle \rangle$ is a leaf, we are going to prove that the procedure `AlternatingSets`$(G(\mathcal{A}))$ returns an empty set, and therefore Algorithm 1 enters in the conditional of line 4 and returns $\mathcal{Z}_{\mathcal{F}}$. Indeed, if $\mathcal{Z}_{\mathcal{F}}$ is a leaf it means that there is no subset of Σ not belonging to $\mathcal{F}$. Therefore, if we remove recursively the transitions labelled with the maximal priority from

[1] In a previous version of this paper, we asked if this Proposition held in the state-based scenario. The answer is yes, since we can use the Zielonka tree of the Muller condition to build a minimal state-based parity automaton recognising the Muller condition. This result was independently proven by Philipp Meyer.

Algorithm 1 `Zielonka-Tree`($\mathcal{S}$)

1: `Input` : An $\mathcal{A}$-subgraph $\mathcal{S}$
2: $C_{\mathcal{S}} = \mathit{Letters}(\mathcal{S})$
3: $\langle C_1, \dots, C_k \rangle =$ `AlternatingSets`($\mathcal{S}$)
4: **if** $k = 0$ (Empty list) **then**
5:     **return** $\mathcal{Z}_{\mathcal{F}} = \langle C_{\mathcal{S}}, \langle \emptyset \rangle \rangle$
6: **else**
7:     **for** $i = 1 \dots k$ **do**
8:         $\mathcal{S}_i =$ `Complete-SCC`($C_i, \mathcal{S}$)
9:         $T_i =$ `Zielonka-Tree`(S_i)
10:     **end for**
11:     **return** $\mathcal{Z}_{\mathcal{F}} = \langle C_{\mathcal{S}}, \langle T_1, \dots, T_k \rangle \rangle$
12: **end if**

$\mathcal{A}$ we must obtain a graph in which the maximal priority of any strongly connected component is even (if not, we would reject a word producing a run visiting all the transitions of this SCC). We conclude that the algorithm `AlternatingSets` does not add any element to the set AltSets in the cycle of line 6.

If $\mathcal{Z}_{\mathcal{F}} = \langle \Sigma, \langle \mathcal{Z}_{\mathcal{F}_1}, \dots, \mathcal{Z}_{\mathcal{F}_k} \rangle \rangle$, we prove that `AlternatingSets`($G(\mathcal{A})$) returns a list of sets $C_1, \dots, C_k$ corresponding to the labels of the children of the root of $\mathcal{Z}_{\mathcal{F}}$. Then, we know that each C_i-SCC computed in line 8 corresponds to a parity automaton recognising the Muller condition associated to the subtree under the i-th child and thus, by induction hypothesis, $T = \mathcal{Z}_{\mathcal{F}}$.

First, we remark that each set C added to AltSets verifies that $C \notin \mathcal{F}$, since we only add it if it is the set of letters labelling one strongly connected component whose maximal priority is odd. As in line 13 we only keep the maximal subsets of AltSets, it suffices to show that any maximal rejecting subset C will be included in some set added to AltSets. Let C be such a set. By Lemma 16 we know that there is some C-SCC $\mathcal{S}_C$ in $\mathcal{A}$, and it must verify that the maximal priority on it is odd. Let $\mathcal{S}'$ be a strongly connected subgraph of $G(\mathcal{A})$ containing $\mathcal{S}_C$. Since C is maximal amongst the rejecting subsets, either $\mathit{Letters}(\mathcal{S}') = C$ or the maximal priority of $\mathcal{S}'$ is even. Therefore, in line 10 we will disregard any SCC $\mathcal{S}'$ containing $\mathcal{S}_C$ that does not verify $\mathit{Letters}(\mathcal{S}') = C$ and we will recursively inspect it until finding one labelled with C. We note, that all subgraphs considered in this process contain $\mathcal{S}_C$ since the maximal priority of it is odd.

Complexity analysis. Let $n = |\mathcal{A}|$, d be the number of priorities appearing in $\mathcal{A}$, $c = |\Sigma|$, $f(n, d, c)$ be the complexity of `Zielonka-Tree`($G(\mathcal{A})$) and $g(n, d, c)$ the complexity of `AlternatingSets`($G(\mathcal{A})$). We start by obtaining

Algorithm 2 The sub-procedure `AlternatingSets`

`Input` : An $\mathcal{A}$-subgraph $\mathcal{S}$
$p =$ *Max-Priority*$(\mathcal{S})$
Compute $\mathcal{S}_p = (V_p, E_p)$, where $E_p = \{e \in E \;:\; \gamma(e) < p\}$ and V_p are the vertices that have some outgoing edge labelled with a priority smaller than p.
$L = \langle \mathcal{S}_1, \dots, \mathcal{S}_l \rangle =$ `SCC-Decomposition`$(\mathcal{S}_p)$
AltSets $= \{\}$ ▷ Initialise an empty set
for $i = 1, \dots l$ **do**
    if *Max-Priority*$(\mathcal{S}_i)$ is odd if and only if p is even **then**
        AltSets.`add`(*Letters*$(\mathcal{S}_i)$)
    else
        AltSets = AltSets $\cup$ `AlternatingSets`$(\mathcal{S}_i)$
    end if
end for
MaxAltSets = `MaxInclusion`(AltSets)
return MaxAltSets

an upper bound for $g(n, d, c)$. It is verified:

$$g(n, d, c) \leq g(n_1, d-2, c) + \cdots + g(n_l, d-2, c) + \mathcal{O}(c^2 n^2)$$

for some $1 \leq n_1, \dots, n_l \leq n$ such that $\sum_{i=1}^{l} n_i \leq n$.

We obtain a weighted-tree such that the sum of the weights of the children of some node τ is at most the weight of τ. The height of this tree is at most $\lceil d/2 \rceil$ and the weight of the root is n. One such tree has less than $\mathcal{O}(dn)$ nodes, so $g(n, d) = \mathcal{O}(dc^2 n^3)$.

For the complexity of $f(n, d, c)$ we have a similar recurrence:

$$f(n, d, c) \leq f(n_1, d-1, c) + \cdots + f(n_k, d-1, c) + k\mathcal{O}(cn) + g(n, d, c)$$

for some $1 \leq n_1, \dots, n_k \leq n-1$ such that $\sum_{i=1}^{k} n_i \leq n$.

Replacing $g(n, d, c)$ by $\mathcal{O}(dc^2 n^3)$ and using an argument similar to the above one, we obtain that $f(n, d, c) = \mathcal{O}(d^2 c^2 n^4)$. □

Proposition 18. *Let $\mathcal{F} \subseteq \mathcal{P}(\Sigma)$ be a Muller condition. If $L_{\mathcal{F}}$ can be recognised by a generalised Büchi (resp. generalised co-Büchi) automaton, then, it can be recognised by one such automaton with just one state. Moreover, this minimal automaton can be built in polynomial time from any generalised Büchi (resp. generalised co-Büchi) automaton recognising $L_{\mathcal{F}}$.*

Proof. We present the proof for generalised Büchi automata, the generalised co-Büchi case being dual. The language $L_{\mathcal{F}}$ can be recognised by a deterministic generalised Büchi automaton if and only if it can be recognised by a Büchi automaton, and this is possible if and only if the Zielonka tree $\mathcal{Z}_{\mathcal{F}}$

has at most height 2 and in the case it has exactly height 2, the set Σ is accepting [CCF21].

Suppose that $\mathcal{F}$ verifies this condition, and let its Zielonka tree be $\mathcal{Z}_{\mathcal{F}} = \langle \Sigma, \langle \langle A_1, \langle \emptyset \rangle \rangle, \langle A_2, \langle \emptyset \rangle \rangle, \dots, \langle A_k, \langle \emptyset \rangle \rangle \rangle \rangle$, verifying $\Sigma \in \mathcal{F}$ and $A_i \notin \mathcal{F}$ for $i \in \{1, \dots, k\}$. Then, it is easy to check that the following generalised Büchi automaton $\mathcal{A}_{min} = (\{q_0\}, \Sigma, q_0, \delta, \Sigma, \{B_1, \dots, B_k\})$ of size 1 recognises $L_{\mathcal{F}}$:

- $Q = \{q_0\}$.
- $\delta(q, c) = (q, c)$, for $c \in \Sigma$.
- $B_i = \Sigma \setminus A_i$ are the sets defining the generalised Büchi condition.

Finally, we prove that we can build this automaton from a given generalised Büchi automaton recognising $L_{\mathcal{F}}$ in polynomial time. We remark that in order to build $\mathcal{A}_{min}$ it suffices to identify the maximal rejecting subsets $A_1, A_2, \dots, A_k \subseteq \Sigma$. Let $\mathcal{A}' = (Q, \Sigma, q_0, \delta, \Gamma, \{B'_1, \dots, B'_r\})$ be a generalised Büchi automaton recognising $L_{\mathcal{F}}$, let $G(\mathcal{A}') = (V_{\mathcal{A}'}, E_{\mathcal{A}'})$ be the graph associated to $\mathcal{A}'$, and let ι and γ be the projections over the input and output alphabet, respectively. For each subset B'_i, $i \in \{1, \dots, r\}$ we consider the subgraph $G_i = (V_i, E_i)$, obtained as the restriction of $G(\mathcal{A}')$ to the transitions that do not produce a letter in B'_i, that is, $V_i = V_{\mathcal{A}'}$, $E_i = \{e \in E_{\mathcal{A}'} : \gamma(e) \notin B'_i\}$. We compute the strongly connected component of G_i, $\mathcal{S}_{i,1}, \dots, \mathcal{S}_{i,l_i}$, and for each of them we consider the projection over the input alphabet, $C_{i,j} = \iota(\mathcal{S}_{i,j}) \subseteq \Sigma$. In this way, we obtain a colection of subsets of Σ, $\{C_{i,j}\}_{1 \le i \le k; 1 \le j \le l_i}$, and all of them have to be rejecting. We claim that the maximal subsets amongst them are the maximal rejecting subsets of Σ. Indeed, if $A \subseteq \Sigma$ is a rejecting subset, a run over $\mathcal{A}'$ reading all the letters of A eventually does not see any transition producing a colour in B'_i, for some i, so this run is contained in a strongly connected component of G_i, and A is therefore a subset of $C_{i,j}$, for some j.

□

3 Memory in games

In this section, we introduce the definitions of games, memories and chromatic memories for games, as well as the distinction between games where we allow ε-transitions and ε-free games. We show in Section 3.4 that the memory requirements for this two latter classes of games might differ.

3.1 Games

A *game* is a tuple $\mathcal{G} = (V = V_E \uplus V_A, E, v_0, \gamma : E \to \Gamma \cup \{\varepsilon\}, Acc)$ where (V, E) is a directed graph together with a partition of the vertices $V = V_E \uplus V_A$,

v_0 is an initial vertex, γ is a colouring of the edges and Acc is a winning condition defining a subset $\mathbb{W} \subseteq \Gamma^\omega$. The letter ε is a neutral letter, and we impose that there is no cycle in $\mathcal{G}$ labelled exclusively with ε. We say that vertices in V_E belong to *Eve* (also called the *existential player*) and those in V_A to *Adam* (*universal player*). We suppose that each vertex in V has at least one outgoing edge. A game that uses a winning condition of type X (as defined in Section 2.1) is called an X-game.

A *play* in $\mathcal{G}$ is an infinite path $\varrho \in E^\omega$ produced by moving a token along edges starting in v_0: the player controlling the current vertex chooses what transition to take. Such a play produces a word $\gamma(\varrho) \in (\Gamma \cup \{\varepsilon\})^\omega$. Since no cycle in $\mathcal{G}$ consists exclusively of ε-colours, after removing the occurrences of ε from $\gamma(\varrho)$ we obtain a word in Γ^ω, that we will call the *output* of the play and we will also denote $\gamma(\varrho)$ whenever no confusion arises. The play is *winning* for Eve if the output belongs to the set $\mathbb{W}$ defined by the acceptance condition, and winning for Adam otherwise. A *strategy* for Eve in $\mathcal{G}$ is a function prescribing how Eve should play. Formally, it is a function $\sigma : E^* \to E$ that associates to each partial play ending in a vertex $v \in V_E$ some outgoing edge from v. A play $\varrho \in E^\omega$ *adheres* to the strategy σ if for each partial play $\varrho' \in E^*$ that is a prefix of ϱ and ends in some state of Eve, the next edge played coincides with $\sigma(\varrho')$. We say that Eve *wins* the game $\mathcal{G}$ if there is some strategy σ for her such that any play that adheres to σ produces a winning play for her (in this case we say that σ is a *winning strategy*).

We will also study games without ε-transitions. We say that a game $\mathcal{G}$ is *ε-free* if $\gamma(e) \neq \varepsilon$ for all edges $e \in E$.

3.2 Memory structures

We give the definitions of the following notions from the point of view of the existential player, Eve. Symmetric definitions can be given for the universal player (Adam), and all results of Section 4 can be dualised to apply to the universal player.

A *memory structure for the game $\mathcal{G}$* is a tuple $\mathcal{M}_\mathcal{G} = (M, m_0, \mu)$ where M is a set of states, $m_0 \in M$ is an initial state and $\mu : M \times E \to M$ is an update function (where E denotes the set of edges of the game). We extend the function μ to $M \times E^*$ in the natural way. We can use such a memory structure to define a strategy for Eve using a function $\texttt{next-move} : V_E \times M \to E$, verifying that $\texttt{next-move}(v, m)$ is an outgoing edge from v. After each move of a play on $\mathcal{G}$, the state of the memory $\mathcal{M}_\mathcal{G}$ is updated using μ; and when a partial play arrives to a vertex v controlled by Eve she plays the edge indicated by the function $\texttt{next-move}(v, m)$, where m is the current state of the memory. We say that the memory structure $\mathcal{M}_\mathcal{G}$ *sets a winning strategy* in $\mathcal{G}$ if there exists such a function $\texttt{next-move}$ defining a winning strategy for Eve.

We say that $\mathcal{M}_\mathcal{G}$ is a *chromatic memory* if there is some function $\mu' : M \times \Gamma \to M$ such that $\mu(m,e) = \mu'(m,\gamma(e))$ for every edge $e \in E$ such that $\gamma(e) \neq \varepsilon$, and $\mu(m,e) = m$ if $\gamma(e) = \varepsilon$. That is, the update function of $\mathcal{M}_\mathcal{G}$ only depends on the colours of the edges of the game.

Given a winning condition $\mathbb{W} \subseteq \Gamma^\omega$, we say that $\mathcal{M} = (M, m_0, \mu)$ is an *arena-independent memory* for $\mathbb{W}$ if for any $\mathbb{W}$-game $\mathcal{G}$ won by Eve, there exists some function $\texttt{next-move}_\mathcal{G} : V_E \times M \to E$ setting a winning strategy in $\mathcal{G}$. We remark that such a memory is always chromatic.

The *size* of a memory structure is its number of states.

Given a Muller condition $\mathcal{F}$, we write $\mathfrak{mem}_{gen}(\mathcal{F})$ (resp. $\mathfrak{mem}_{chrom}(\mathcal{F})$) for the least number n such that for any $\mathcal{F}$-game that is won by Eve, she can win it using a memory (resp. a chromatic memory) of size n. We call $\mathfrak{mem}_{gen}(\mathcal{F})$ (resp. $\mathfrak{mem}_{chrom}(\mathcal{F})$) the *general memory requirements (resp. chromatic memory requirements) of $\mathcal{F}$*. We write $\mathfrak{mem}_{ind}(\mathcal{F})$ for the least number n such that there exists an arena-independent memory for $\mathcal{F}$ of size n.

We define respectively all these notions for ε-free $\mathcal{F}$-games. We write $\mathfrak{mem}^{\varepsilon\text{-free}}_{gen}(\mathcal{F})$, $\mathfrak{mem}^{\varepsilon\text{-free}}_{chrom}(\mathcal{F})$ and $\mathfrak{mem}^{\varepsilon\text{-free}}_{ind}(\mathcal{F})$ to denote, respectively, the minimal general memory requirements, minimal chromatic memory requirements and minimal size of an arena-independent memory for ε-free $\mathcal{F}$-games.

Remark 19. *We remark that these quantities verify that $\mathfrak{mem}_{gen}(\mathcal{F}) \leq \mathfrak{mem}_{chrom}(\mathcal{F}) \leq \mathfrak{mem}_{ind}(\mathcal{F})$ and that $\mathfrak{mem}^{\varepsilon\text{-free}}_X(\mathcal{F}) \leq \mathfrak{mem}_X(\mathcal{F})$ for $X \in \{gen, chrom, ind\}$.*

A family of games is *half-positionally determined* if for every game in the family that is won by Eve, she can win using a strategy given by a memory structure of size 1.

Lemma 20 ([Kla94, Zie98]). *Rabin-games are half-positionally determined.*

If $\mathcal{A}$ is a Rabin automaton recognising a Muller condition $\mathcal{F}$, given an $\mathcal{F}$-game $\mathcal{G}$ we can perform a standard product construction $\mathcal{G} \ltimes \mathcal{A}$ to obtain an equivalent game using a Rabin condition that is therefore half-positionally determined. This allows us to use the automaton $\mathcal{A}$ as an arena-independent memory for $\mathcal{F}$.

Lemma 21 (Folklore). *Let $\mathcal{F}$ be a Muller condition. We can use a Rabin automaton $\mathcal{A}$ recognising $L_\mathcal{F}$ as an arena-independent memory for $\mathcal{F}$.*

3.3 The general memory requirements of Muller conditions

The Zielonka tree (see Definition 5) was introduced by Zielonka in [Zie98], and in [DJW97] it was used to characterise the general memory requirements of Muller games as we show next.

Definition 22. *Let $\mathcal{F}$ be a Muller condition and $\mathcal{Z}_{\mathcal{F}} = \langle \Gamma, \langle \mathcal{Z}_{\mathcal{F}_1}, \dots \mathcal{Z}_{\mathcal{F}_k} \rangle\rangle$ its Zielonka tree. We define the number $\mathfrak{m}_{\mathcal{Z}_{\mathcal{F}}}$ recursively as follows:*

$$\mathfrak{m}_{\mathcal{Z}_{\mathcal{F}}} = \begin{cases} 1 & \text{if } \mathcal{Z}_{\mathcal{F}} \text{ is a leaf} \\ \max\{\mathfrak{m}_{\mathcal{Z}_{\mathcal{F}_1}}, \dots, \mathfrak{m}_{\mathcal{Z}_{\mathcal{F}_k}}\} & \text{if } \Gamma \notin \mathcal{F} \text{ and } \mathcal{Z}_{\mathcal{F}} \text{ is not a leaf} \\ \sum_{i=1}^{k} \mathfrak{m}_{\mathcal{Z}_{\mathcal{F}_i}} & \text{if } \Gamma \in \mathcal{F} \text{ and } \mathcal{Z}_{\mathcal{F}} \text{ is not a leaf}. \end{cases}$$

Proposition 23 ([DJW97])**.** *For every Muller condition $\mathcal{F}$, $\mathfrak{mem}_{gen}(\mathcal{F}) = \mathfrak{m}_{\mathcal{Z}_{\mathcal{F}}}$. That is,*

1. *If Eve wins an $\mathcal{F}$-game, she can win it using a strategy given by a (general) memory structure of size at most $\mathfrak{m}_{\mathcal{Z}_{\mathcal{F}}}$.*

2. *There exists an $\mathcal{F}$-game (with ε-transitions) won by Eve such that she cannot win it using a strategy given by a memory structure of size strictly smaller than $\mathfrak{m}_{\mathcal{Z}_{\mathcal{F}}}$.*

3.4 Memory requirements of ε-free games

In [Zie98] and [Kop06] it was noticed that there can be major differences regarding the memory requirements of winning conditions depending on the way the games are coloured. We can differentiate 4 classes of games, according to whether we colour vertices or edges, and whether we allow or not the neutral colour ε:

A) State-coloured ε-free games.

B) General state-coloured games.

C) Transition-coloured ε-free games.

D) General transition-coloured games.

(In the previous sections we have only defined transition-coloured games). We remark that the memory requirements of any condition are the same for general state-coloured games and general transition-coloured games.

In [Zie98], Zielonka showed that there are Muller conditions that are half-positional over state-coloured ε-free games, but they are not half-positional over general state-coloured games, and he exactly characterises half-positional Muller conditions in both cases.

However, when considering transition-coloured games, this "bad behaviour" does not appear: in both general games and ε-free games, half-positional Muller conditions correspond exactly to Rabin conditions (Lemma 24). In particular, the characterisation of Zielonka of half-positional Muller conditions for state-coloured ε-free games does not generalise to transition-coloured ε-free games.

Nevertheless, the matching upper bounds for the memory requirements of Muller conditions appearing in [DJW97] are given by transition-labelled games using ε-transitions. An interesting question is whether we can produce upper-bound examples using ε-free games. In this section we answer this question negatively. We show in Proposition 25 that, for every $n \geq 2$, there is a Muller condition $\mathcal{F}$ such that Eve can use memories of size 2 to win ε-free $\mathcal{F}$-games where she can force a win, but that she might need n memory states to win $\mathcal{F}$-games with ε transitions. That is, $\mathfrak{mem}_{gen}^{\varepsilon\text{-free}}(\mathcal{F}) < \mathfrak{mem}_{gen}(\mathcal{F})$ and the gap can be arbitrarily large. In Section 4.1 we will see that this is not the case for chromatic memories: $\mathfrak{mem}_{chrom}(\mathcal{F}) = \mathfrak{mem}_{chrom}^{\varepsilon\text{-free}}(\mathcal{F})$ for any Muller condition $\mathcal{F}$.

Lemma 24. *For any Muller condition $\mathcal{F} \subseteq \mathcal{P}(\Gamma)$, $\mathcal{F}$ is half-positional determined over transition-coloured ε-free games if and only if $\mathcal{F}$ is half-positional determined over general transition-coloured games. That is, $\mathfrak{mem}_{gen}(\mathcal{F}) = 1$ if and only if $\mathfrak{mem}_{gen}^{\varepsilon\text{-free}}(\mathcal{F}) = 1$.*

Proof. Since $\mathfrak{mem}_{gen}^{\varepsilon\text{-free}}(\mathcal{F}) \leq \mathfrak{mem}_{gen}(\mathcal{F})$ for all conditions, it suffices to see that if $\mathcal{F}$ is a Muller condition such that $1 < \mathfrak{mem}_{gen}(\mathcal{F})$, then we can find an ε-free $\mathcal{F}$-game won by Eve where she cannot win positionally.

By the characterisation of [DJW97] (Proposition 23), we know that $\mathfrak{mem}_{gen}(\mathcal{F}) = 1$ if and only if every node of the Zielonka tree of $\mathcal{F}$ labelled with an accepting set of colours has at most one child. Therefore, if $\mathcal{F} \subseteq \mathcal{P}(\Gamma)$ is a Muller condition verifying $1 < \mathfrak{mem}_{gen}(\mathcal{F})$, we can find a node in $\mathcal{Z}_\mathcal{F}$ labelled with $A \in \mathcal{F}$ and with two different children, labelled respectively with $B_1, B_2 \subseteq \Gamma$, verifying that $B_i \notin \mathcal{F}$ for $i \in \{1, 2\}$ and such that $B_i \nsubseteq B_j$, for $i \neq j$. We consider the ε-free game consisting in one initial state v_0 controlled by Eve with two outgoing edges, one leading to a cycle that comes back to v_0 after producing the colours in B_1, and the other one leading to a cycle producing the colours in B_2. It is clear that Eve wins this game, but she needs at least 2 memory states to do so. ☐

Proposition 25. *For any integer $n \geq 2$, there is a set of colours Γ_n and a Muller condition $\mathcal{F}_n \subseteq \mathcal{P}(\Gamma_n)$ such that $\mathfrak{mem}_{gen}^{\varepsilon\text{-free}}(\mathcal{F}) = 2$ and $\mathfrak{mem}_{gen}(\mathcal{F}) = n$.*

Proof. Let us consider the set of colours $\Gamma_n = \{1, \ldots, n\}$ and the Muller condition

$$\mathcal{F}_n = \{A \subseteq \Gamma_n \,:\, |A| > 1\}.$$

The Zielonka tree of $\mathcal{F}_n$ is pictured in Figure 1, where round nodes represent nodes whose label is an accepting set, and rectangular ones, nodes whose label is a rejecting set.

By the characterisation of [DJW97] (Proposition 23), we know that $\mathfrak{mem}_{gen}(\mathcal{F}) = n$, that is, there is a game with ε-transitions won by Eve where she needs at least n memory states to force a win. We are going to

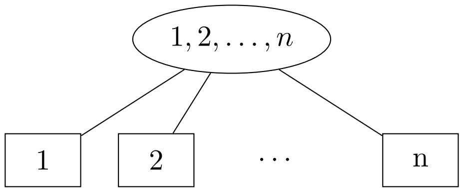

Figure 1: Zielonka tree $\mathcal{Z}_{\mathcal{F}_n}$.

prove that if Eve wins an ε-free $\mathcal{F}$-game, then she can win it using only 2 memory states.

Let $\mathcal{G} = (V = V_E \uplus V_A, E, v_0, \gamma, \mathcal{F})$ be a an ε-free $\mathcal{F}$-game. First, we can suppose that Eve wins the game where we change the initial vertex v_0 to any other vertex of V. Indeed, since $\mathcal{F}$ is a prefix-independent condition, we can restrict ourselves to the winning region of Eve (and any strategy has to ensure that she remains there). From any vertex $v \in V_E$, there is one outgoing transition coloured with some colour in Γ, since the game is ε-free. We associate to each vertex $v \in V_E$ one such colour, via a mapping $c : V_E \to \Gamma_n$, obtaining a partition $V_E = V_1 \uplus \cdots \uplus V_n$ verifying that $v \in V_x$ implies that there is some transition labelled with x leaving v, for $x \in \Gamma_n$. We denote $\sigma_0 : V_E \to E$ one application that maps each vertex $v \in V_E$ to one outgoing edge labelled with $c(v)$.

Moreover, for any $v \in V$, since Eve can win from v, there is some strategy in $\mathcal{G}$ forcing to see some colour $y \in \Gamma_n$, $y \neq c(v)$. For each colour $x \in \Gamma_n$, we consider the game $\mathcal{G}_x = (V = V_E \uplus V_A, E, v, \gamma, \mathit{Reach}(\Gamma_n \setminus \{x\}))$, where the underlying graph is the same as in $\mathcal{G}$, $v \in V$ is an arbitrarily vertex and the winning condition consists in reaching some colour different from x. Eve can win this game starting from any vertex $v \in V$, and we can fix a positional winning strategy for her that does not depend on the initial vertex (since reachability games are uniformly positional determined [GTW02]). We denote $\sigma_x : V_E \to E$ the choice of edges defining this strategy. Moreover, we can pick σ_x such that it coincides with σ_0 outside V_x.

We define a memory structure $\mathcal{M}_{\mathcal{G}} = (M = \{m_0, m_1\}, m_0, \mu)$ for $\mathcal{G}$ and a `next-move` function describing the following strategy: the state m_0 will be used to remember that we have to see the colour x corresponding to the component V_x that we are in. As soon as we arrive to a vertex controlled by Eve, we use the next transition to accomplish this and we can change to state m_1 in $\mathcal{M}_{\mathcal{G}}$. The state m_1 will serve to follow the positional strategy σ_x reaching one colour different from x. We will change to state m_0 if we arrive to some state in V_E not in V_x (this will ensure that we will see one colour different from x), or if Eve produces a colour different from x staying in the component V_x. If she does not produce this colour and we do not go to a vertex in $V_E \setminus V_x$, that means that (since σ_x ensures that we will see a colour different from x), Adam will take some transition coloured with some colour different from x. This memory structure is formally defined as follows:

- The set of states is $M = \{m_0, m_1\}$, being m_0 the initial state.
- The update function $\mu : M \times E \to M$ is defined as:

$$\begin{cases} \mu(m_i, (v, v')) = m_i & \text{if } v \in V_A, \text{ for } i \in \{1, 2\}, \\ \mu(m_0, (v, v')) = m_1 & \text{if } v \in V_E, \\ \mu(m_1, (v, v')) = m_0 & \text{if } v, v' \in V_E \text{ and } c(v) \neq c(v'), \\ \mu(m_1, (v, v')) = m_0 & \text{if } v \in V_E \text{ and } c(v) \neq \gamma((v, v')), \\ \mu(m_1, (v, v')) = m_1 & \text{in any other case.} \end{cases}$$

- The function `next-move` $: M \times V_E \to E$ is defined as:

$$\begin{cases} \texttt{next-move}(m_0, v) = \sigma_0(v), \\ \texttt{next-move}(m_1, v) = \sigma_{c(v)}(v). \end{cases} \qquad \square$$

Remark 26. *The condition of the previous proof also provides an example of a condition that is half-positional over ε-free state-coloured arenas, but for which we might need memory n in general state-coloured arenas (similar examples can be found in [Zie98, Kop06]).*

However, the question raised in [Kop06] of whether there can be conditions (that cannot be Muller ones) that are half-positional only over ε-free games remains open.

4 The chromatic memory requirements of Muller conditions

In this section we present the main contributions concerning the chromatic memory requirements of Muller conditions. In Section 4.1, we prove that the chromatic memory requirements of a Muller condition (even for ε-free games) coincide with the size of a minimal Rabin automaton recognising the Muller condition (Theorem 28). In Section 4.2 we deduce that determining the chromatic memory requirements of a Muller condition is NP-complete, for different representations of the condition. Finally, this results allow us to answer in Section 4.3 the question appearing in [Kop06, Kop08] of whether the chromatic memory requirements coincide with the general memory requirements of winning conditions.

4.1 Chromatic memory and Rabin automata

In this section we prove Theorem 28, establishing the equivalence between the chromatic memory requirements of a Muller condition (also for ε-free games) and the size of a minimal Rabin automaton recognising the Muller condition.

Lemma 27 appears in Kopczyński's PhD thesis [Kop08, Proposition 8.9] (unpublished). We present a similar proof here.

Lemma 27 ([Kop08])**.** *Let $\mathcal{F}$ be a Muller condition. Then, $\mathfrak{mem}_{chrom}(\mathcal{F}) = \mathfrak{mem}_{ind}(\mathcal{F})$. That is, there is an $\mathcal{F}$-game $\mathcal{G}$ won by Eve such that any chromatic memory for $\mathcal{G}$ setting a winning strategy has size at least $\mathfrak{mem}_{ind}(\mathcal{F})$, where $\mathfrak{mem}_{ind}(\mathcal{F})$ is the minimal size of an arena-independent memory for $\mathcal{F}$.*

The same result holds for ε-free games: $\mathfrak{mem}^{\varepsilon\text{-free}}_{chrom}(\mathcal{F}) = \mathfrak{mem}^{\varepsilon\text{-free}}_{ind}(\mathcal{F})$.

Proof. We present the proof for $\mathfrak{mem}_{chrom}(\mathcal{F}) = \mathfrak{mem}_{ind}(\mathcal{F})$, the proof for the ε-free case being identical, since we do not add any ε-transition to the games we consider.

It is clear that $\mathfrak{mem}_{chrom}(\mathcal{F}) \leq \mathfrak{mem}_{ind}(\mathcal{F})$, since any arena-independent memory for $\mathcal{F}$-games has to be chromatic. We will prove that it is not the case that $\mathfrak{mem}_{chrom}(\mathcal{F}) < \mathfrak{mem}_{ind}(\mathcal{F})$. Let $\mathcal{M}_1, \cdots, \mathcal{M}_n$ be an enumeration of all chromatic memory structures of size strictly less than $\mathfrak{mem}_{ind}(\mathcal{F})$. By definition of $\mathfrak{mem}_{ind}(\mathcal{F})$, for any of the memories $\mathcal{M}_j$ there is some $\mathcal{F}$-game $\mathcal{G}_j = (V_j, E_j, v_{0_j}, \gamma_j)$ won by Eve such that no function $\texttt{next-move}_{\mathcal{G}_j} : M_j \times V_j \to E_j$ setting a winning strategy in $\mathcal{G}_j$ exists. We define the disjoint union of these games, $\mathcal{G} = \biguplus_{i=1}^{n} \mathcal{G}_i$, as the game with an initial vertex v_0 controlled by Adam, from which he can choose to go to the initial vertex of any of the games $\mathcal{G}_i$ producing the letter $a \in \Gamma$ (for some $a \in \Gamma$ fixed arbitrarily), and such the rest of vertices and transitions of $\mathcal{G}$ is just the disjoint union of those of the games $\mathcal{G}_i$. Eve can win this game, since no matter the choice of Adam we arrive to some game where she can win. However, we show that she cannot win using a chromatic memory strictly smaller than $\mathfrak{mem}_{ind}(\mathcal{F})$. Suppose by contradiction that she wins using a chromatic memory $\mathcal{M} = (M, m_0, \mu)$, $|\mathcal{M}| < \mathfrak{mem}_{ind}(\mathcal{F})$. We let $m_0' = \mu(m_0, a)$, and we consider the memory structure $\mathcal{M}' = (M, m_0', \mu)$. Since $|\mathcal{M}'| < \mathfrak{mem}_{ind}(\mathcal{F})$, $\mathcal{M}' = \mathcal{M}_i$ for some $i \in \{1, \ldots, n\}$, and therefore Adam can choose to take the transition leading to $\mathcal{G}_i$, where Eve cannot win using this memory structure. This contradicts the fact that Eve wins $\mathcal{G}$ using $\mathcal{M}$. $\square$

Theorem 28. *Let $\mathcal{F} \subseteq \mathcal{P}(\Gamma)$ be a Muller condition. The following quantities coincide:*

1. *The size of a minimal deterministic Rabin automaton recognising $L_{\mathcal{F}}$, $\mathfrak{rabin}(L_{\mathcal{F}})$.*

2. *The size of a minimal arena-independent memory for $\mathcal{F}$, $\mathfrak{mem}_{ind}(\mathcal{F})$.*

3. *The size of a minimal arena-independent memory for ε-free $\mathcal{F}$-games, $\mathfrak{mem}_{ind}^{\varepsilon\text{-free}}(\mathcal{F})$.*

4. *The chromatic memory requirements of $\mathcal{F}$, $\mathfrak{mem}_{chrom}(\mathcal{F})$.*

5. *The chromatic memory requirements of $\mathcal{F}$ for ε-free games, $\mathfrak{mem}_{chrom}^{\varepsilon\text{-free}}(\mathcal{F})$.*

Proof. The previous Lemma 27, together with Lemma 21, prove that

$$\mathfrak{mem}_{ind}^{\varepsilon\text{-free}}(\mathcal{F}) = \mathfrak{mem}_{chrom}^{\varepsilon\text{-free}}(\mathcal{F}) \leq \mathfrak{mem}_{chrom}(\mathcal{F}) = \mathfrak{mem}_{ind}(\mathcal{F}) \leq \mathfrak{rabin}(L_{\mathcal{F}}).$$

In order to prove that $\mathfrak{rabin}(L_{\mathcal{F}}) \leq \mathfrak{mem}_{ind}^{\varepsilon\text{-free}}(\mathcal{F})$, we are going to show that we can put a Rabin condition on top of any arena-independent memory for ε-free $\mathcal{F}$-games $\mathcal{M}$, obtaining a Rabin automaton recognising $L_{\mathcal{F}}$ and having the same size than $\mathcal{M}$.

Let $\mathcal{M} = (M, m_0, \mu : M \times \Gamma \to M)$ be an arena-independent memory for ε-free $\mathcal{F}$-games. First, we remark that we can suppose that every state of $\mathcal{M}$ is accessible from m_0 by some sequence of transitions. We define a Muller automaton $\mathcal{A}_{\mathcal{M}}$ using the underlying structure of $\mathcal{M}$: $\mathcal{A}_{\mathcal{M}} = (M, \Gamma, m_0, \delta, \Gamma, \mathcal{F})$, where the transition function δ is defined as $\delta(m, a) = (\mu(m, a), a)$, for $a \in \Gamma$. Since the output produced by any word $w \in \Gamma^\omega$ is w itself and the accepting condition is $\mathcal{F}$, this automaton trivially accepts the language $L_{\mathcal{F}}$. We are going to show that the Muller automaton $\mathcal{A}_{\mathcal{M}}$ satisfies the second property in Proposition 3, that is, that for any pair of cycles in $\mathcal{A}_{\mathcal{M}}$ with some state in common, if both are rejecting then their union is also rejecting. This will prove that we can put a Rabin condition on top of $\mathcal{A}_{\mathcal{M}}$.

Let ℓ_1 and ℓ_2 be two rejecting cycles in $\mathcal{A}_{\mathcal{M}}$ such that $m \in M$ is contained in both ℓ_1 and ℓ_2. We suppose by contradiction that their union $\ell_1 \cup \ell_2$ is an accepting cycle. We will build an ε-free $\mathcal{F}$-game that is won by Eve, but where she cannot win using the memory $\mathcal{M}$, leading to a contradiction. Let $a_0 a_1 \dots a_k \in \Gamma^*$ be a word labelling a path to m from m_0 in $\mathcal{M}$, that is, $\mu(m_0, a_0 \dots a_k) = m$. We define the ε-free $\mathcal{F}$-game $\mathcal{G} = (V = V_E, E, v_0, \gamma : E \to \Gamma, \mathcal{F})$ as the game where there is a sequence of transitions labelled with $a_0 \dots a_k$ from v_0 to one vertex v_m controlled by Eve (the only vertex in the game where some player has to make a choice). From v_m, Eve can choose to see all the transitions of ℓ_1 before coming back to m (producing the corresponding colours), or to see all the transitions of ℓ_2 before coming back to m.

First, we notice that Eve can win the game $\mathcal{G}$: since $\ell_1 \cup \ell_2$ is accepting, she only has to alternate between the two choices in the state v_m. However, there is no function `next-move` $: M \times V_E \to E$ setting up a winning strategy for Eve. Indeed, for every partial play ending in v_m and labelled with $a_0 a_1 \dots a_s$, it is clear that $\mu(m_0, a_0 \dots a_s) = m$ (the memory is at state m).

If $\texttt{next-move}(m, v_m)$ is the edge leading to the cycle corresponding to ℓ_1, no matter the value $\texttt{next-move}$ takes at the other pairs, all plays will stay in ℓ_1, so the set of colours produced infinitely often would be $\gamma(\ell_1)$ which is loosing for Eve. The result is symmetric if $\texttt{next-move}(m, v_m)$ is the edge leading to the other cycle. We conclude that $\mathcal{M}$ cannot be used as a memory structure for $\mathcal{G}$, a contradiction. $\square$

4.2 The complexity of determining the chromatic memory requirements of a Muller condition

As shown in [DJW97], the Zielonka tree of a Muller condition directly gives its general memory requirements. In this section, we see that it follows from the previous results that determining the chromatic memory requirements of a Muller condition is $\texttt{NP}$-complete, even if it is represented by its Zielonka tree.

Proposition 29. *Given the Zielonka tree $\mathcal{Z}_{\mathcal{F}}$ of a Muller condition $\mathcal{F}$, we can compute in $\mathcal{O}(|\mathcal{Z}_{\mathcal{F}}|)$ the memory requirements for $\mathcal{F}$-games, $\mathfrak{mem}_{gen}(\mathcal{F})$.*

Using Algorithm 1 we can compute the Zielonka tree of a Muller condition from a parity automaton recognising that condition. We deduce the following result.

Corollary 30. *Given a parity automaton $\mathcal{P}$ recognising a Muller condition $\mathcal{F}$, we can compute in polynomial time in $|\mathcal{P}|$ the memory requirements for $\mathcal{F}$-games, $\mathfrak{mem}_{gen}(\mathcal{F})$.*

Theorem 31. *Given a positive integer $k > 0$ and a Muller condition $\mathcal{F}$ represented as either:*

a) The Zielonka tree $\mathcal{Z}_{\mathcal{F}}$.

b) A parity automaton recognising $L_{\mathcal{F}}$.

c) A Rabin automaton recognising $L_{\mathcal{F}}$.

The problem of deciding whether $\mathfrak{mem}_{chrom}(\mathcal{F}) \geq k$ (or equivalently, $\mathfrak{mem}_{ind}(\mathcal{F}) \geq k$) is $\texttt{NP}$-complete.

Proof. We first remark that by Theorem 28, the size of a minimal arena-independent memory for $\mathcal{F}$ coincides with the size of a minimal Rabin automaton for $L_{\mathcal{F}}$.

We show that for the three representations the problem is in $\texttt{NP}$. By Proposition 7 we can find in non-deterministic polynomial time a minimal Rabin automaton recognising $L_{\mathcal{F}}$, if we are given as input a Rabin automaton for $L_{\mathcal{F}}$, (therefore, also if we are given a parity automaton as input).

Since the Zielonka tree $\mathcal{Z}_{\mathcal{F}}$ allows us to produce in polynomial time a parity automaton for $L_{\mathcal{F}}$, $\mathfrak{mem}_{ind}(\mathcal{F})$ can be computed in non-deterministic polynomial time in $|\mathcal{Z}_{\mathcal{F}}|$.

The NP-hardness part has been proven in Theorem 14 if the input is a Rabin automaton. If the input is the Zielonka tree $\mathcal{Z}_{\mathcal{F}}$ or a parity automaton, we show that the reduction presented in Section 2.2 can also be applied heree. Given a simple undirected graph $G = (V, E)$, we consider the Muller condition over V given by $\mathcal{F}_G = \{\{v, u\} \subseteq V \ : \ (v, u) \in E\}$. It is verified that $L_G = L_{\mathcal{F}_G}$, and by Lemma 13 the size of a minimal Rabin automaton for L_G (and therefore the size of a minimal arena-independent memory for $\mathcal{F}$) coincides with the chromatic number of G. It is enough to show that we can build the Zielonka tree $\mathcal{Z}_{\mathcal{F}_G}$ and a parity automaton for $L_{\mathcal{F}_G}$ in polynomial time in the size of G. The Zielonka tree of $\mathcal{F}_G$ is the following one: for each pair of vertices $\{v, u\} \subseteq V$ such that $(v, u) \in E$, consider the tree $T_{\{v,u\}} = \Big\langle \{v, u\}, \langle \langle \{v\}, \langle \emptyset \rangle \rangle, \langle \{u\}, \langle \emptyset \rangle \rangle \rangle \Big\rangle$ (of height 2). Then, the Zielonka tree of $\mathcal{F}_G$ is given by:

$$\mathcal{Z}_{\mathcal{F}_G} = \langle V, \langle T_{e_1}, \dots, T_{e_k} \rangle \rangle,$$

where $e_1, \dots e_k$ is an enumeration of the unordered pairs of vertices forming an edge. We can build this tree in $\mathcal{O}(|V| + |E|)$. This Zielonka tree gives us a parity automaton for L_G in linear time (Proposition 6). $\square$

4.3 Chromatic memories require more states than general ones

In his PhD Thesis [Kop06, Kop08], Kopczyński raised the question of whether for every winning condition its general memory requirements coincide with its chromatic memory requirements. In this section we provide an example of a game that Eve can win, but she can use a strictly smaller memory structure to do so if she is allowed to employ a general memory (non-chromatic).

Proposition 32. *For each integer $n \geq 2$, there exists a set of colours Γ_n and a Muller condition $\mathcal{F}_n$ over Γ_n such that for any $\mathcal{F}_n$-game won by Eve, she can win it using a memory of size 2, but there is an $\mathcal{F}_n$-game $\mathcal{G}$ where Eve needs a chromatic memory of size n to win. Moreover, the game $\mathcal{G}$ can be chosen to be ε-free.*

Proof. Let $\Gamma_n = \{1, 2, \dots, n\}$ be a set of n colours, and let us define the Muller condition $\mathcal{F}_n$ as:

$$\mathcal{F}_n = \{A \subseteq \Gamma_n \ : \ |A| = 2\}.$$

The Zielonka tree of $\mathcal{F}_n$ is depicted in Figure 2.

The characterisation of the memory requirements of Muller conditions from Proposition 23 gives that $\mathfrak{mem}_{gen}(\mathcal{F}_n) = 2$.

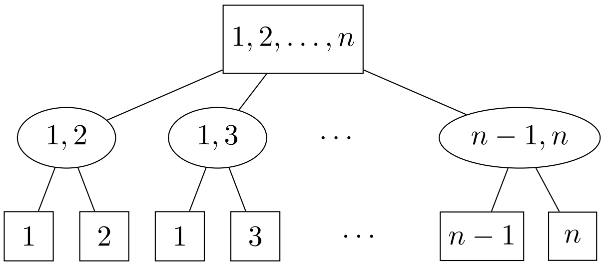

Figure 2: Zielonka tree for the condition $\mathcal{F}_n = \{A \subseteq \{1, 2, \dots, n\} : |A| = 2\}$. Square nodes are associated with rejecting sets ($A \notin \mathcal{F}_n$) and round nodes with accepting ones ($A \in \mathcal{F}_n$).

On the other hand, the language $L_{\mathcal{F}_n}$ associated to this condition coincides with the language L_G (defined in Section 2.2) associated to a graph G that is a clique of size n. By Lemma 13, the size of a minimal Rabin automaton recognising $L_{\mathcal{F}_n}$ (and therefore, by Theorem 28, the chromatic memory requirements of $\mathcal{F}_n$) coincides with the chromatic number of G. Since G is a clique of size n, its chromatic number is n. $\square$

We give next a concrete example of a game where $\mathfrak{mem}_{gen}(\mathcal{F}) < \mathfrak{mem}_{chrom}(\mathcal{F})$.

Example 33. *We consider the set of colours $\Gamma = \{a, b, c\}$ and the Muller condition from the previous proof: $\mathcal{F} = \{A \subseteq \Gamma \ : \ |A| = 2\}$. As we have discussed, $\mathfrak{mem}_{gen}(\mathcal{F}) = 2$.*

We are going to define a game $\mathcal{G}$ won by Eve for which a minimal chromatic memory providing a winning strategy cannot have less than 3 states.

Let $\mathcal{G}$ be the $\mathcal{F}$-game shown in Figure 3. Round vertices represent vertices controlled by Eve, and square ones are controlled by Adam.

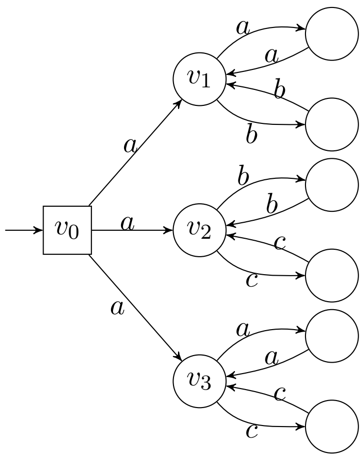

Figure 3: The game $\mathcal{G}$.

Suppose that there is a chromatic memory structure for $\mathcal{G}$ of size 2, $\mathcal{M}_{\mathcal{G}} = (M = \{m_0, m_1\}, m_0, \mu)$, together with a function `next-move` $: M \times V_E \to E$ *setting a winning strategy for Eve. For winning this game, the number of different colours produced infinitely often has to be exactly two, so the transitions of the memory must ensure that, whenever one outgoing edge from v_i is chosen, when we come back to v_i (after reading two colours in Γ) we have changed of state in the memory. Therefore, for each colour $x \in \{a, b, c\}$ we must have that $\mu(m_0, x) = \mu(m_1, x)$, what implies that there are two different colours (we will suppose for simplicity that they are a and b) such that $\mu(m_j, a) = \mu(m_j, b) = m$, for $j \in \{1, 2\}$ and a state $m \in \{m_0, m_1\}$. However, this implies that whenever a partial play ends up in v_1, the memory will be in the state m, and Eve will always choose to play the edge given by* `next-move`(m, v_1)*, producing a loosing play.*

5 Conclusions and open questions

In this work, we have fully characterised the chromatic memory requirements of Muller conditions, proving that arena-independent memory structures for a given Muller condition correspond to Rabin automata recognising that condition. We have also answered several open questions concerning the memory requirements of Muller conditions when restricting ourselves to chromatic memories or to ε-free games. We have proven the `NP`-completeness of the minimisation of transition-based Rabin automata and that we can minimise parity automata recognising Muller languages in polynomial time, advancing in our understanding on the complexity of decision problems related to transition-based automata.

The question of whether we can minimise transition-based parity or Büchi automata in polynomial time remains open. The contrast between the results of Abu Radi and Kupferman [AK19, AK20], showing that we can minimise GFG transition-based co-Büchi automata in polynomial time and those of Schewe [Sch20], showing that minimising GFG state-based co-Büchi automata is `NP`-complete; as well as the contrast between Theorem 14 and Proposition 17, make of this question a very intriguing one.

Regarding the memory requirements of games, we have shown that forbidding ε-transitions might cause a reduction in the memory requirements of Muller conditions. However, the question raised by Kopczyński in [Kop06] remains open: are there prefix-independent winning conditions that are half-positional when restricted to ε-free games, but not when allowing ε-transitions?

Acknowledgements

I would like to thank Alexandre Blanché for pointing me to the chromatic number problem. I also want to thank Bader Abu Radi, Thomas Colcombet, Nathanaël Fijalkow, Orna Kupferman, Karoliina Lehtinen and Nir Piterman for interesting discussions on the minimisation of transition-based automata, a problem introduced to us by Orna Kupferman. Finally, I warmly thank Thomas Colcombet, Nathanaël Fijalkow and Igor Walukiewicz for their help in the preparation of this paper, their insightful comments and for introducing me to the different problems concerning the memory requirements studied in this paper.

References

[AK19] Bader Abu Radi and Orna Kupferman. Minimizing GFG transition-based automata. In *ICALP*, volume 132, pages 100:1–100:16, 2019.

[AK20] Bader Abu Radi and Orna Kupferman. Canonicity in GFG and transition-based automata. In *GandALF*, volume 326, pages 199–215, 2020.

[BRO+20] Patricia Bouyer, Stéphane Le Roux, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Games where you can play optimally with arena-independent finite memory. In *CONCUR*, volume 171 of *LIPIcs*, pages 24:1–24:22, 2020.

[CCF21] Antonio Casares, Thomas Colcombet, and Nathanaël Fijalkow. Optimal transformations of games and automata using Muller conditions. In *ICALP*, volume 198, pages 123:1–123:14, 2021.

[CDK93] Edmund M. Clarke, I. A. Draghicescu, and Robert P. Kurshan. A unified approch for showing language inclusion and equivalence between various types of omega-automata. *Inf. Process. Lett.*, 46(6):301–308, 1993.

[CFH14] Thomas Colcombet, Nathanaël Fijalkow, and Florian Horn. Playing safe. In *FSTTCS*, volume 29, pages 379–390, 2014.

[CHVB18] Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors. *Handbook of Model Checking.* Springer, Cham, 2018.

[CM99] Olivier Carton and Ramón Maceiras. Computing the Rabin index of a parity automaton. *RAIRO*, pages 495–506, 1999.

[CN06] Thomas Colcombet and Damian Niwiński. On the positional determinacy of edge-labeled games. *Theoretical Computer Science*, 352(1):190–196, 2006.

[CZ09] Thomas Colcombet and Konrad Zdanowski. A tight lower bound for determinization of transition labeled Büchi automata. In *ICALP*, pages 151–162, 2009.

[DJW97] Stefan Dziembowski, Marcin Jurdziński, and Igor Walukiewicz. How much memory is needed to win infinite games? In *LICS*, pages 99–110, 1997.

[GL02] Dimitra Giannakopoulou and Flavio Lerda. From states to transitions: Improving translation of LTL formulae to Büchi automata. In *FORTE*, pages 308–326, 2002.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata Logics, and Infinite Games.* Springer, Berlin, Heidelberg, 2002.

[GZ05] Hugo Gimbert and Wieslaw Zielonka. Games where you can play optimally without any memory. In *CONCUR*, volume 3653 of *Lecture Notes in Computer Science*, pages 428–442, 2005.

[Hop71] John E. Hopcroft. An n log n algorithm for minimizing states in a finite automaton. Technical report, Stanford University, 1971.

[Kar72] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. The IBM Research Symposia Series. Springer US, 1972.

[Kla94] Nils Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Annals of Pure and Applied Logic*, 69(2):243–268, 1994.

[Kop06] Eryk Kopczyński. Half-positional determinacy of infinite games. In *ICALP*, pages 336–347, 2006.

[Kop08] Eryk Kopczyński. Half-positional determinacy of infite games. PhD Thesis, 2008.

[Löd01] Christof Löding. Efficient minimization of deterministic weak omega-automata. *Inf. Process. Lett.*, 79(3):105–109, 2001.

[MS17] David Müller and Salomon Sickert. LTL to deterministic Emerson-Lei automata. In *GandALF*, pages 180–194, 2017.

[MS21] Philipp Meyer and Salomon Sickert. On the optimal and practical conversion of Emerson-Lei automata into parity automata. *Personal Communication*, 2021.

[PR89] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, page 179–190, 1989.

[Saf88] Schmuel Safra. On the complexity of ω-automata. In *FOCS*, page 319–327, 1988.

[Sch09] Sven Schewe. Tighter bounds for the determinisation of Büchi automata. In *FoSSaCS*, pages 167–181, 2009.

[Sch10] Sven Schewe. Beyond hyper-minimisation—minimising DBAs and DPAs is NP-complete. In *FSTTCS*, volume 8 of *LIPIcs*, pages 400–411, 2010.

[Sch20] Sven Schewe. Minimising Good-For-Games automata is NP-complete. In *FSTTCS*, volume 182, pages 56:1–56:13, 2020.

[Sha81] Micha Sharir. A strong-connectivity algorithm and its applications in data flow analysis. *Computers and Mathematics with Applications*, 7(1):67–72, 1981.

[Tar72] Robert Tarjan. Depth first search and linear graph algorithms. *Siam Journal On Computing*, 1(2), 1972.

[Wag79] Klaus Wagner. On ω-regular sets. *Information and control*, 43(2):123–177, 1979.

[Zie98] Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.