Deciding Regular Games: a Playground for Exponential Time Algorithms

Zihui Liang[1]

University of Electronic Science and Technology of China, Chengdu, China

Bakh Khoussainov[1]

University of Electronic Science and Technology of China, Chengdu, China

Mingyu Xiao

University of Electronic Science and Technology of China, Chengdu, China

Abstract

Regular games form a well-established class of games for analysis and synthesis of reactive systems. They include colored Muller games, McNaughton games, Muller games, Rabin games, and Streett games. These games are played on directed graphs $\mathcal{G}$ where Player 0 and Player 1 play by generating an infinite path ρ through the graph. The winner is determined by specifications put on the set X of vertices in ρ that occur infinitely often. These games are determined, enabling the partitioning of $\mathcal{G}$ into two sets Win_0 and Win_1 of winning positions for Player 0 and Player 1, respectively. Numerous algorithms exist that decide instances of regular games, e.g., Muller games, by computing Win_0 and Win_1. In this paper we aim to find general principles for designing uniform algorithms that decide all regular games. For this we utilize various recursive and dynamic programming algorithms that leverage standard notions such as subgames and traps. Importantly, we show that our techniques improve or match the performances of existing algorithms for many instances of regular games.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory; Theory of computation → Logic and verification

Keywords and phrases Regular games, colored Muller games, Rabin games, McNaughton games, Muller games, deciding games

Digital Object Identifier 10.4230/LIPIcs.MFCS.2025.66

Funding This work was supported by the National Science Foundation of China under grant No.62172077, No.62350710215, and No.62372095, and the Natural Science Foundation of Sichuan Province of China under grant No.2025HJPJ0006.

1 Introduction

In verification of reactive systems, model checking, and logic, studying games played on finite graphs is a key research topic [18, 26]. Colored Muller games, Rabin games, Streett games, Muller games, and McNaughton games constitute such classes of games. The recent work [16] serves as an excellent reference for the state-of-the-art in this area. Interest in these games arises from their role in modeling and verifying reactive systems as games on graphs.

1.1 Arenas, regular games, subarenas, and traps

All games that we listed above are played in arenas:

▶ **Definition 1.** *An **arena** $\mathcal{A}$ is a bipartite directed graph (V_0, V_1, E), where*

1. *$V_0 \cap V_1 = \emptyset$, and $V = V_0 \cup V_1$ is the set of nodes, also called **positions**.*
2. *$E \subseteq V_0 \times V_1 \cup V_1 \times V_0$ is the edge set where each node has an outgoing edge.*
3. *V_0 and V_1 are sets of positions for Player 0 and Player 1, respectively.*

[1] Corresponding authors.

For each $v \in V$, let $E(v) = \{u \mid (v, u) \in E\}$ be the set of successors of v.

Players play the game in a given arena $\mathcal{A}$ by taking turns and moving a token along the edges of the arena. Initially, the token is placed on a node $v_0 \in V$. If $v_0 \in V_0$, then Player 0 moves first. If $v_0 \in V_1$, then Player 1 moves first. In each round of play, if the token is positioned on a Player σ's position v, then Player σ chooses $u \in E(v)$, moves the token to u along the edge (v, u), and the play continues on to the next round. Note that condition 2 on the arena guarantees that the players can always make a move at any round of the play.

▶ **Definition 2.** *A **play**, in a given arena $\mathcal{A}$, starting at v_0, is an infinite sequence $\rho = v_0, v_1, v_2, \ldots$ such that $v_{i+1} \in E(v_i)$ for all $i \in \mathbb{N}$.*

Given a play $\rho = v_0, v_1, \ldots$, the set $\mathsf{Inf}(\rho) = \{v \in V \mid \exists^\omega i(v_i = v)\}$ is called the **infinity set** of ρ. The winner of this play is determined by a condition put on $\mathsf{Inf}(\rho)$. We list several of these conditions that are well-established in the area.

▶ **Definition 3.** *The following games played on a given arena $\mathcal{A} = (V_0, V_1, E)$ are known as **regular games**:*

1. *A **colored Muller game** is $\mathcal{G} = (\mathcal{A}, c, (\mathcal{F}_0, \mathcal{F}_1))$, where $c : V \to C$ is a mapping from V into the set C of colors, $\mathcal{F}_0 \cup \mathcal{F}_1 = 2^C$ and $\mathcal{F}_0 \cap \mathcal{F}_1 = \emptyset$. Player σ **wins** the play ρ if $c(\mathsf{Inf}(\rho)) \in \mathcal{F}_\sigma$, where $\sigma = 0, 1$.*
2. *A **McNaughton game** is the tuple $\mathcal{G} = (\mathcal{A}, W, (\mathcal{F}_0, \mathcal{F}_1))$, where $W \subseteq V$, $\mathcal{F}_0 \cup \mathcal{F}_1 = 2^W$ and $\mathcal{F}_0 \cap \mathcal{F}_1 = \emptyset$. Player σ **wins** the play ρ if $\mathsf{Inf}(\rho) \cap W \in \mathcal{F}_\sigma$.*
3. *A **Muller game** is the tuple $\mathcal{G} = (\mathcal{A}, (\mathcal{F}_0, \mathcal{F}_1))$, where $\mathcal{F}_0 \cup \mathcal{F}_1 = 2^V$ and $\mathcal{F}_0 \cap \mathcal{F}_1 = \emptyset$. Player σ **wins** the play ρ if $\mathsf{Inf}(\rho) \in \mathcal{F}_\sigma$.*
4. *A **Rabin game** is the tuple $\mathcal{G} = (\mathcal{A}, (U_1, V_1), \ldots, (U_k, V_k))$, where $U_i, V_i \subseteq V$, (U_i, V_i) is a **winning pair**, and k is the **index**. Player 0 **wins** ρ if there is a pair (U_i, V_i) with $\mathsf{Inf}(\rho) \cap U_i \neq \emptyset$ and $\mathsf{Inf}(\rho) \cap V_i = \emptyset$. Else, Player 1 wins.*
5. *A **Streett game** is the tuple $\mathcal{G} = (\mathcal{A}, (U_1, V_1), \ldots, (U_k, V_k))$, where U_i, V_i are as in Rabin game. Player 0 **wins** the play ρ if for all $i \in \{1, \ldots, k\}$ if $\mathsf{Inf}(\rho) \cap U_i \neq \emptyset$ then $\mathsf{Inf}(\rho) \cap V_i \neq \emptyset$. Otherwise, Player 1 wins.*
6. *A **KL game** is the tuple $\mathcal{G} = (\mathcal{A}, (u_1, S_1), \ldots, (u_t, S_t))$, where $u_i \in V$, $S_i \subseteq V$, (u_i, S_i) is a **winning pair**, and t is the **index**. Player 0 **wins** ρ if there is a pair (u_i, S_i) such that $u_i \in \mathsf{Inf}(\rho)$ and $\mathsf{Inf}(\rho) \subseteq S_i$. Else, Player 1 wins.*

The first three games are symmetric. Rabin games can be considered as dual to Streett games. The first five winning conditions are well-established conditions. The last condition is new. The motivation behind this new winning condition lies in the transformation of Rabin and Streett games into Muller games via the KL winning condition. In a precise sense, as will be seen in Section 4.5 via Lemma 26, the KL condition is a compressed Rabin condition.

▶ **Definition 4.** *Let $\mathcal{A}$ be an arena. A **pseudo-arena** of $\mathcal{A}$ determined by X is the tuple $\mathcal{A}(X) = (X_0, X_1, E_X)$ where $X_0 = V_0 \cap X$, $X_1 = V_1 \cap X$, $E_X = E \cap (X \times X)$. If this pseudo-arena is an arena, then we call it the **subarena** determined by X.*

The opponent of Player σ, where $\sigma \in \{0, 1\}$, is denoted by Player $\bar{\sigma}$. Traps are subarenas in games where one of the players has no choice but stay:

▶ **Definition 5** (σ-trap)**.** *A subarena $\mathcal{A}(X)$ is a **σ-trap** for Player σ if each of the following two conditions are satisfied: (1) For all $x \in X_{\bar{\sigma}}$ there is a $y \in X_\sigma$ such that $(x, y) \in E$. (2) For all $x \in X_\sigma$ it is the case that $E(x) \subseteq X$.*

If $\mathcal{A}(X)$ is a σ-trap, then Player $\bar{\sigma}$ can stay in $\mathcal{A}(X)$ forever if the player wishes so.

Let T be a subset of the arena $\mathcal{A} = (V_0, V_1, E)$. The attractor of Player σ to the set $T \subseteq V$, denoted $Attr_\sigma(T, \mathcal{A})$, is the set of positions from where Player σ can force the plays into T. The attractor $Attr_\sigma(T, \mathcal{A})$ is computed as follows:

$$W_0 = T, \quad W_{i+1} = W_i \cup \{u \in V_\sigma \mid E(u) \cap W_i \neq \emptyset\} \cup \{u \in V_{\bar{\sigma}} \mid E(u) \subseteq W_i\},$$
$$\text{and then set} \quad Attr_\sigma(T, \mathcal{A}) = \textstyle\bigcup_{i \geq 0} W_i.$$

The set $Attr_\sigma(T, \mathcal{A})$ can be computed in $O(|E|)$. We call $Attr_\sigma$ the attractor operator. Note that the set $V \setminus Attr_\sigma(T, \mathcal{A})$, the complement of the σ-attractor of T, is a σ-trap for all T. This set is the emptyset if and only if $V = Attr_\sigma(T, \mathcal{A})$.

A *strategy* for Player σ is a function that receives as input initial segments of plays $v_0, v_1, \ldots, v_k$ where $v_k \in V_\sigma$ and outputs some v_{k+1} such that $v_{k+1} \in E(v_k)$. An important class of strategies are finite-state strategies. R. McNaughton in [33] proved that the winner in McNaughton games has a finite state winning strategy. W. Zielonka proves that the winners of regular games have finite state winning strategies [39]. S. Dziembowski, M. Jurdzinski, and I. Walukiewicz in [11] investigate the memory needed for the winners of colored Muller games. They show that the memory $|V|!$ is a sharp bound for finite state winning strategies.

In the study of games, solving a given game entails two key objectives. First, one aims to devise an algorithm that, when provided with a game $\mathcal{G}$, partitions the set V into two sets $Win_0(\mathcal{G})$ and $Win_1(\mathcal{G})$ such that $v \in Win_\sigma(\mathcal{G})$ if and only if Player σ wins the game starting at v, where $\sigma \in \{0, 1\}$. This is called the **decision problem** where one wants to find out the winner of the game. Second, one would like to design an algorithm that, given a game, extracts a winning strategy for the winner. This is known as the **synthesis problem**.

Traditionally, research on regular games specifically selects an instance of regular games, e.g., Muller games, Rabin games or Streett games, and studies the decision and synthesis problems for these instances. This paper however, instead of focusing on instances of regular games, aims at finding uniform algorithms and general principles for deciding all regular games. Importantly, we show that our techniques based on general principles improve or match the performances of existing decision algorithms for many instances of regular games.

1.2 Our contributions in light of known algorithms

We provide two types of algorithms for deciding regular games. The first is recursion based, and the second is dynamic programming based. Recursive algorithms have been exploited in the area significantly. To the best of our knowledge, dynamic programming techniques have not been much used in the area. We utilize these techniques and improve known algorithms for deciding all regular games defined above.

The performances of algorithms for games $\mathcal{G}$ can be measured in two ways. One is when the input sizes are $|V| + |E|$. The other is when $\mathcal{G}$ is presented *explicitly* by listing V, E, and the corresponding winning conditions. In these explicit representations the sizes of Muller, McNaughton, and colored Muller games are bounded by $|V| + |E| + 2^{|V|} \cdot |V|$. For these games, explicit representations list the collection $\mathcal{F}_0$ instead of listing both $\mathcal{F}_0$ and $\mathcal{F}_1$. The sizes of Rabin and Streett games are bounded by $|V| + |E| + 4^{|V|} \cdot |V|$. The sizes of the KL games are bounded by $|V| + |E| + 2^{|V|} \cdot |V|^2$. We use the notation $|\mathcal{G}|$ for these representations of games $\mathcal{G}$. These two ways of representing inputs, together with the parameters $|C|$, $|W|$, k, and t, should be taken into account in our discussion below.

▶ **Definition 6.** *Call $|C|$ and $|W|$ **small** parameters, k and t **(potentially) large** parameters.*

A: Colored Muller games. The folklore algorithms deciding colored Muller games use recursion on $|C|$ with running time $O(|C||E|(|C||V|)^{|C|-1})$ and space $O(|\mathcal{G}| + |C||V|)$ [16]. Using the breakthrough quasi-polynomial time algorithm for parity games, C. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan were able to decide colored Muller games with the running time $O(|C|^{5|C|} \cdot |V|^5)$ and space $O((|C|!|V|)^{O(1)})$ [4]. Björklund et al. showed that under the ETH it is impossible to decide colored Muller games in time $O(2^{o(|C|)} \cdot |V|^a)$, where $a > 0$ [1]. C. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan in [4] improved this by showing that under the ETH it is impossible to decide colored Muller games in time $2^{o(|C| \cdot \log(|C|))} Poly(|V|)$. Their proof implies that this impossibility result holds when $|C| \leq \sqrt{|V|}$. The table below now compares these results with our algorithms.

Best known (running time, space)	Our algorithm (s)
$(O(\lvert C\rvert^{5\lvert C\rvert} \cdot \lvert V\rvert^5),\ O((\lvert C\rvert!\lvert V\rvert)^{O(1)}))$ [4]	$(O(2^{\lvert V\rvert}\lvert C\rvert\lvert E\rvert),\ O(\lvert\mathcal{G}\rvert + 2^{\lvert V\rvert}\lvert V\rvert))$ Theorem 15 (DP)
$(O(\lvert C\rvert\lvert E\rvert(\lvert C\rvert\lvert V\rvert)^{\lvert C\rvert-1}),\ O(\lvert\mathcal{G}\rvert + \lvert C\rvert\lvert V\rvert))$ folklore, e.g., see[16]	$(O(2^{\lvert V\rvert}\lvert V\rvert\lvert E\rvert),\ O(\lvert\mathcal{G}\rvert + 2^{\lvert V\rvert}))$ Theorem 19 (DP)
	$(O(\lvert C\rvert!\binom{\lvert V\rvert}{\lvert C\rvert}\lvert V\rvert\lvert E\rvert),\ O(\lvert\mathcal{G}\rvert + \lvert C\rvert\lvert V\rvert))$ Theorem 12 (recursion)

The algorithms from Theorems 15 and 19 are dynamic programming (DP) algorithms. One can verify that if $|V|/\log\log(|V|) \leq |C|$, e.g., $|V|/a < C$ where $a > 1$ is fixed, then:

1. Running times of both of these algorithms are better than $O(|C|^{5|C|} \cdot |V|^5)$,
2. Moreover, these two algorithms run in $2^{o(|C| \cdot \log(|C|))} Poly(|V|)$. This obviously refines and strengthens the impossibility result that under the ETH no algorithm decides colored Muller games in $2^{o(|C| \cdot \log(|C|))} Poly(|V|)$ [4].
3. The spaces of both of these algorithms are also better than $O((|C|!|V|)^{O(1)})$.
4. All of the previously known algorithms have superexponential running times. Our algorithms run in exponential time.

When the parameter C satisfies $|C| \leq \log(|V|)$, the algorithms from [4] and [16] outperform our algorithms. When $|V|/\log\log(|V|) \leq |C|$, as we stated above, our algorithms are better for any value of $|C|$ with $C \geq |V|/a$. Furthermore, our algorithms run in exponential time thus matching the bound of the impossibility result of Björklund et al. [1].

Our algorithm in Theorem 12 is a recast of standard recursive algorithms. However, as shown in the table, a careful running time analysis implies that our recursive algorithm has a better running time and it matches the space bounds of the previous recursive algorithms.

B: Rabin and Streett games. Historically, M. Rabin was the first who built the parameter independent exponential time algorithm that solves the emptiness problem for ω-tree automata with Rabin acceptance condition (which is equivalent to solving Rabin games). Namely, given an ω-tree automata $A = (S, M, s_0, \Omega)$ with Rabin acceptance condition, Rabin's algorithm runs in time $O(|\Sigma|4^{4|\Sigma||S|})$ and space $O(2^{2|\Sigma||S|} \cdot |\Sigma||S|)$, where Σ is the size of the alphabet [38]. Note that $|\Sigma|$ appears in the exponent[1].

E. A. Emerson and C. S. Jutla show that deciding Rabin games is NP complete [12, 14]. Hence, deciding Streett games is co-NP complete. Horn's algorithm for deciding Rabin games has the running time $O(k!|V|^{2k})$ [19]. N. Piterman and A. Pnueli show that Rabin and Streett games can be decided in time $O(|E||V|^{k+1}kk!)$ and space $O(nk)$ [37]. C. Calude, S.

[1] The authors thank M. Vardi for informing us of this Rabin's algorithm.

Jain, B. Khoussainov, W. Li, and F. Stephan gave a FPT algorithm for Rabin games on k colors by converting it to a parity game and using the quasi-polynomial algorithm [4]. A Rabin game with n vertices, m edges and k colors, can be reduced to a parity game with $N = nk^2k!$ vertices, $M = nk^2k!m$ edges and $K = 2k + 1$ colors [13]. By reducing Rabin games to parity games and using the state-of-the-art algorithms for parity games [7, 8, 15, 23] in a "space-efficient" manner, one solves Rabin games in time $O(\max\{MN^{2.38}, 2^{O(K \log K)}\})$ with exponential space, see Jurdziński and Lazić [23]. Using the values of M and N, the algorithm of Jurdziński and Lazić takes time at least proportional to $m(nk^2k!)^{3.38}$. The parity games obtained from Rabin games are such that the number of vertices N is much larger than the number of colors K, namely $K \in o(\log(N))$. For cases where the number of vertices of the resulting parity game is much larger than the number of priorities, say the number of colors $(2k + 1)$ is $o(\log(N))$, Jurdziński and Lazić also give an analysis of their algorithm that would solve Rabin games in time $O(nmk!^{2+o(1)})$. Closely matching this are the run times in the work of Fearnley et al. [15] who provide, among other bounds, a quasi-bi-linear bound of $O(MN\mathfrak{a}(N)^{\log\log N})$, where $\mathfrak{a}$ is the inverse-Ackermann function. In either case above, this best-known algorithm has at least a $(k!)^{2+o(1)}$ dependence in its run time, and takes the space proportional to $(nk^2k!)\log(nk^2k!)$; so we still have a $k!$ dependence. R. Majumdar et al. in [32] recently provided an algorithm that decides Rabin games in $\tilde{O}(|E||V|(k!)^{1+o(1)})$ time and $O(|V|k \log k \log |V|)$ space. This breaks through the $2 + o(1)$ barrier. Importantly, A. Casares et al. prove that under the ETH it is impossible to decide Rabin games in $2^{o(k \log k)}Poly(|V|)$ [6]. Just like for colored Muller games, this impossibility result holds true when $k \leq \sqrt{|V|}$. The next table compares these results with our findings.

Best known (running time, space)	Our algorithm (s)
$(O(\|E\|\|V\|^{k+1}kk!), O(\|\mathcal{G}\| + k\|V\|))$ [37]	$(O((k\|V\| + 2^{\|V\|}\|E\|)\|V\|), O(\|\mathcal{G}\| + 2^{\|V\|}\|V\|))$ Theorem 28 (DP)
$(\tilde{O}(\|E\|\|V\|(k!)^{1+o(1)}), O(\|\mathcal{G}\| + k\|V\| \log k \log \|V\|))$ [32]	$(O(\|V\|!\|V\|(\|E\| + k\|V\|)), O(\|\mathcal{G}\| + \|V\|^2))$ Theorem 14 (recursion)
$(O(\|\Sigma\|4^{4\|\Sigma\|\|S\|}), O(2^{2\|\Sigma\|\|S\|} \cdot \|\Sigma\|\|S\|))$ Rabin's algorithm [38]	

We single out four key parts of both of our algorithms:

1. In terms of time, both our dynamic and recursive algorithms outperform the known algorithms when the parameter k ranges in $[|V|, 4^{|V|}]$. In particular, when k is polynomial on $|V|$ (which is a practical consideration), then our algorithms have better running times.
2. Just as for colored Muller games we refine the impossibility result of A. Casares et al. under the assumption of the ETH [6]. Namely, when the parameter $k \geq |V| \log |V|$, both of our algorithms run in $2^{o(k \log k)}Poly(|V|)$. The condition $k \geq |V| \log |V|$ is clearly reasonable and practically feasible.
3. Our DP algorithm from Theorem 28 is the first exponential time algorithm that decides Rabin games. The previously known algorithms run in superexponential times.
4. When k falls into the range $[|V|, 4^{|V|}]$, then the recursive algorithm from Theorem 14 performs the best in terms of space against other algorithms.

If Player 0 wins Rabin games, then the player wins with a memoryless strategy [14]. Hence, one might suggest the following way of finding the winner. Enumerate all memoryless strategies and select the winning one. Even when the arena is sparse, e.g., positions have a fixed bounded out-degree, this process does not lead to exponential running time as the opponent might have a winning strategy with a large memory.

C: Muller games. Nerode, Remmel, and Yakhnis were the first who designed a competitive algorithm that decides Muller games [35]. The running time of their algorithm is $O(|V|! \cdot |V||E|)$. W. Zielonka [39] examines Muller games through Zielonka trees. The size of Zielonka tree is $O(2^{|V|})$ in the worst case. S. Dziembowski, M. Jurdzinski, and I. Walukiewicz in [11] show that deciding Muller games with Zielonka trees as part of the input is in NP ∩ co-NP. D. Neider, R. Rabinovich, and M. Zimmermann reduce Muller games to safety games with $O((|V|!)^3)$ vertices; safety games can be solved in linear time [34]. F. Horn in [20] provides the first polynomial time decision algorithm for explicitly given Muller games, which operates by solving update network games [2, 9, 10, 22] iteratively on transformed subgames. The running time of Horn's algorithm is $O(|V| \cdot |\mathcal{F}_0| \cdot (|V| + |\mathcal{F}_0|)^2)$. F. Horn's correctness proof has a non-trivial flaw. B. Khoussainov, Z. Liang, and M. Xiao in [28] provide a correct proof of Horn's algorithm through new methods; their methods improve the running time to $O(|\mathcal{F}_0| \cdot (|V| + |\mathcal{F}_0|) \cdot |V_0| \log |V_0|)$. All the known algorithms that decide Muller games are either recursive algorithms or reductions to other known classes of games. Our algorithm is a dynamic programming algorithm, and to the best of our knowledge, the first such algorithm that decides Muller games. The table below compares the best of these results for Muller games, in terms of time and space, with our algorithm from this paper:

Best known (running time, space)	Our algorithm
$(O(\|\mathcal{F}_0\| \cdot (\|V\| + \|\mathcal{F}_0\|) \cdot \|V_0\| \log \|V_0\|), O(\|\mathcal{G}\| + \|\mathcal{F}_0\|(\|V\| + \|\mathcal{F}_0\|)))$ [28]	$(O(2^{\|V\|}\|V\|\|E\|), O(\|\mathcal{G}\| + 2^{\|V\|}))$ Theorem 20 (DP)

One can see that the algorithm from [28], in terms of running time and space, is better than our algorithm when $|\mathcal{F}_0| \leq \sqrt{2^{|V|}}$. However, our algorithm becomes competitive (or better) than the algorithm in [28] when $|\mathcal{F}_0| > \sqrt{2^{|V|}}$. Also, note that by running our algorithm and the algorithm in [28] in parallel, we get the best performing polynomial time algorithm that solves explicitly given Muller games.

D: McNaughton games. R. McNaughton [33] provided the first algorithm that decides McNaughton games in time $O(a^{|W|} \cdot |W|! \cdot |V|^3)$, for a constant $a > 1$. Nerode, Remmel, and Yakhnis in [35] improved the bound to $O(|W|!|W||E|)$. A. Dawar and P. Hunter proved that deciding McNaughton games is a PSPACE-complete problem [21], while the one-player version can be decided in linear time [24]. The table below compares our algorithms with currently the best algorithm that runs in time $O(|W|!|W||E|)$:

Best known (running time, space)	Our algorithm (s)
$(O(\|W\|\|E\|\|W\|!), O(\|\mathcal{G}\| + Poly(\|V\|)))$ [35]	$(O(2^{\|V\|}\|W\|\|E\|), O(\|\mathcal{G}\| + 2^{\|V\|}\|V\|))$ Theorem 21 (DP)
	$(O(2^{\|V\|}\|V\|\|E\|), O(\|\mathcal{G}\| + 2^{\|V\|}))$ Theorem 21 (DP)

It is not too hard to see that when $|W|$ exceeds $|V|/\log\log(|V|)$, e.g., $|W| \geq |V|/a$ where $a > 1$ is fixed, then our algorithm has better running time than previous algorithms.

Important comments. All the previously known algorithms, apart from the exponential time algorithm of Rabin, involve the parameters $|C|$, k, and $|W|$. When these parameters move from very small to reasonable sizes, such as from $\log(|V|)$ to over $|V|/\log\log(|V|)$ (and above) for $|C|$ and $|W|$, and from $\log(|V|)$ to $|V|$ (and above) for k, the running times become unreasonable as they involve the multiplicative factors $|V|^k$, $k!$, $|C|!$, $|C|^{|C|}$, and $|W|!$. In all of these cases, our algorithms perform better in order of magnitude. Also, our algorithms are based on a new approach where we trade the parameters for $|V|$. One can argue that in some situations, the parameters are small with respect to $|V|$. For instance, one can invoke the reduction of non-deterministic Buchi automata to deterministic automata. In this reduction the index k of Rabin accepting condition is bounded by $\log(|S|)$, where S is the states of the deterministic automata. The following examples address these types of arguments:

- The article [5] provides a natural transformation of Muller games $(\mathcal{A}, \mathcal{F}_0)$ on arena $\mathcal{A}$ to Rabin games $(\mathcal{A}', (U_1, V_1), \ldots, (U_k, V_k))$. This natural transformation can produce examples of Rabin games where k is exponential on the number of positions in $\mathcal{A}'$.
- Boker in [3] shows examples of deterministic Muller automata M_1 and M_2 that have n states with index n, that is $|\mathcal{F}_0| = n$, such that every non-deterministic Muller automata accepting $L(M_1) \cap L(M_2)$ has index 2^n. This is an example where a natural algebraic and set-theoretic operation produces a large Muller acceptance condition.
- Although parity games can be solved in quasipolynomial time [4, 15, 36, 27], it is worth to mention when one translates parity games to Rabin games the parameter k equals the number of priorities, e.g., k can equal $|V|$.
- Random Muller games are obtained as follows. In an arena of size n, select a subset X at random. Declare $\mathcal{F}_0 = 2^X$. The expected size of $\mathcal{F}_0$ is $2^{n/2}$.

Our algorithms and their correctness look deceivingly simple. This is achieved due to Definition 7, Lemmas 8–10, Lemmas 17–18, The Enumeration Lemma 22, and The Transformation Lemma 26 each of which is non-trivial on its own. Furthermore, our algorithms remove existing superexponential bounds from the running times of the known algorithms, an important issue in regular games and ω-automata.

2 The notion of full win and characterization of winning regions

In this section we develop a few concepts and techniques used throughout the paper. We first define the notion of *full win*. This will be used in designing dynamic programming algorithms for deciding regular games. Then we provide Lemma 8 that characterizes winning regions. This lemma is used for designing recursive algorithms for solving regular games. The last result of this section is Lemma 10. We call the lemma trichotomy lemma as it characterizes three cases: (1) Player 0 fully wins the game, (2) Player 1 fully wins the game, and (3) none of the players fully wins the game. This lemma will be the basis of our dynamic algorithms.

▶ **Definition 7.** *If $Win_\sigma(\mathcal{G}) = V$, then player σ* ***fully wins*** *$\mathcal{G}$. Else, the player* ***does not fully win*** *$\mathcal{G}$. If $Win_\sigma(\mathcal{G}) \neq V$ and $Win_{\bar{\sigma}}(\mathcal{G}) \neq V$, then* ***no player fully wins*** *$\mathcal{G}$.*

We now provide two lemmas that characterize winning regions in colored Muller games. Later we algorithmically implement the lemmas and analyze them. We start with the first lemma. The statement of the lemma and its equivalent forms have been known and used in various forms [33] [16]. Later we will utilize the lemma in our recursive algorithms through their detailed exposition and analysis.

▶ **Lemma 8.** *Let $\sigma \in \{0, 1\}$ such that $c(V) \in \mathcal{F}_\sigma$. Then we have the following:*

1. *If for all $c' \in c(V)$, $Attr_\sigma(c^{-1}(c'), \mathcal{A}) = V$ or Player σ fully wins $\mathcal{G}(V \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}))$, then Player σ fully wins $\mathcal{G}$.*
2. *Otherwise, let c' be a color in C such that $Attr_\sigma(c^{-1}(c'), \mathcal{A}) \neq V$ and Player σ doesn't fully win $\mathcal{G}(V \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}))$. Then we have $Win_\sigma(\mathcal{G}) = Win_\sigma(\mathcal{G}(V \setminus X))$, where $X = Attr_{\bar{\sigma}}(Win_{\bar{\sigma}}(\mathcal{G}(V \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}))), \mathcal{A})$.*

Proof. Part 1: Assume that for all $c' \in c(V)$, Player σ fully wins $\mathcal{G}(V \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}))$ or $V = Attr_\sigma(c^{-1}(c'), \mathcal{A})$. Construct the following winning strategy for Player σ in $\mathcal{G}$. Let $c(V) = \{c_0, \ldots, c_{k-1}\}$ and let i initially be 0.

- If the token is in $Attr_\sigma(c^{-1}(c_i), \mathcal{A})$, then Player σ forces the token to a vertex in $c^{-1}(c_i)$ and once the token arrives at the vertex, sets $i = i + 1 \mod k$.
- Otherwise, Player σ uses a winning strategy in $\mathcal{G}(V \setminus Attr_\sigma(c^{-1}(c_i), \mathcal{A}))$.

Consider a play consistent with the strategy. If there is an i such that the token finally stays in $\mathcal{G}(V \setminus Attr_\sigma(c^{-1}(c_i), \mathcal{A}))$, then Player σ wins the game. Otherwise, $c(\mathsf{Inf}(\rho)) = c(V)$. So Player σ wins as $c(V) \in \mathcal{F}_\sigma$. Thus, Player σ fully wins $\mathcal{G}$.

Part 2: Let $c' \in C$ such that $Attr_\sigma(c^{-1}(c'), \mathcal{A}) \neq V$ and Player σ doesn't fully win $\mathcal{G}(V \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}))$. Let $V' = Win_{\bar\sigma}(\mathcal{G}(V \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A})))$. Consider the set $X = Attr_{\bar\sigma}(V', \mathcal{A})$ as defined in the statement of the lemma. Note that $\mathcal{A}(V')$ is a σ-trap in $\mathcal{A}(V \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}))$; furthermore, $\mathcal{A}(V \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}))$ is a σ-trap in $\mathcal{A}$. This implies that $\mathcal{A}(V')$ is a σ-trap in $\mathcal{A}$. Now we want to construct a winning strategy for Player $\bar\sigma$ in the arena $\mathcal{A}$ when the token is placed on $v \in X \cup Win_{\bar\sigma}(\mathcal{G}(V \setminus X))$. The winning strategy for Player $\bar\sigma$ in this case is the following:

- If $v \in X$, Player $\bar\sigma$ wins by forcing the token into V' and following the winning strategy in σ-trap $\mathcal{A}(V')$.
- If $v \in Win_{\bar\sigma}(\mathcal{G}(V \setminus X))$, Player $\bar\sigma$ in game $\mathcal{G}(Win_{\bar\sigma}(\mathcal{G}(V \setminus X)))$ follows a winning strategy until the opponent moves the token into X.

Note that $\mathcal{A}(Win_\sigma(\mathcal{G}(V \setminus X)))$ is a $\bar\sigma$-trap in $\mathcal{A}(V \setminus X)$ and $\mathcal{A}(V \setminus X)$ is a $\bar\sigma$-trap in $\mathcal{A}$. Hence, the set $\mathcal{A}(Win_\sigma(\mathcal{G}(V \setminus X)))$ is a $\bar\sigma$-trap in the arena $\mathcal{A}$. Therefore, we have the equality $Win_\sigma(\mathcal{G}) = Win_\sigma(\mathcal{G}(V \setminus X))$. ◀

As an immediate corollary we get the following lemma for Player σ.

▶ **Lemma 9.** *Let $\mathcal{G}$ be a colored Muller game and let $\sigma \in \{0, 1\}$ be such that $c(V) \in \mathcal{F}_\sigma$. Player σ fully wins $\mathcal{G}$ if and only if for all $c' \in c(V)$, $Attr_\sigma(c^{-1}(c'), \mathcal{A}) = V$ or Player σ fully wins $\mathcal{G}(V \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}))$.* ⌟

Now we provide the next lemma that we call Trichotomy lemma. We will use this lemma in our dynamic programming based algorithms.

▶ **Lemma 10 (Trichotomy Lemma).** *Let $\mathcal{G}$ be a colored Muller game and let $\sigma \in \{0, 1\}$ be such that $c(V) \in \mathcal{F}_\sigma$. Then we have the following three cases:*

1. *If for all $c' \in c(V)$, $Attr_\sigma(c^{-1}(c'), \mathcal{A}) = V$ or Player σ fully wins $\mathcal{G}(V \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}))$, then Player σ fully wins $\mathcal{G}$.*
2. *Otherwise, if for all $v \in V$, $Attr_{\bar\sigma}(\{v\}, \mathcal{A}) = V$ or Player $\bar\sigma$ fully wins $\mathcal{G}(V \setminus Attr_{\bar\sigma}(\{v\}, \mathcal{A}))$, then Player $\bar\sigma$ fully wins $\mathcal{G}$.*
3. *Otherwise, none of the players fully wins.*

Proof. By Lemma 9, Part 1 is proved. For the remaining parts of the lemma, we are under the assumption that Player σ doesn't fully win $\mathcal{G}$. For the second part, if Player $\bar\sigma$ fully wins $\mathcal{G}$, then for any $v \in V$, $Attr_{\bar\sigma}(\{v\}, \mathcal{A}) = V$ or Player $\bar\sigma$ fully wins the game in $\bar\sigma$-trap $\mathcal{A}(V \setminus Attr_{\bar\sigma}(\{v\}, \mathcal{A}))$. Otherwise, for all $v \in Win_{\bar\sigma}(\mathcal{G})$, $Attr_{\bar\sigma}(\{v\}, \mathcal{A}) \neq V$ and Player $\bar\sigma$ doesn't fully win $\mathcal{G}(V \setminus Attr_{\bar\sigma}(\{v\}, \mathcal{A}))$. ◀

Part 2 of the lemma assumes that Player σ for which $c(V) \in \mathcal{F}_\sigma$ does not fully win the game. With this assumption, the second part characterizes the condition when Player $\bar\sigma$ fully wins the game; without this assumption, Part 2 does not hold true.

3 Recursive algorithms for deciding regular games

In this short section, our goal is to provide recursive algorithms that solve regular games. To do so we utilize Lemma 8. Naturally, we first start with a generic recursive algorithm that decides colored Muller games, see Figure 1. Lemma 8 guarantees correctness of the algorithm. Initially, the algorithm memorizes $\mathcal{G}$ globally. Then the function SolveCMG(V') is called. The algorithm returns $(Win_0(\mathcal{G}(V')), Win_1(\mathcal{G}(V')))$.

```
**Global Storage:** A colored Muller game $\mathcal{G} = (\mathcal{A}, c, (\mathcal{F}_0, \mathcal{F}_1))$
**Function:** SolveCMG$(V')$
**Input**: A vertex set $V'$ with $\mathcal{A}(V')$ is an arena
**Output**: $(Win_0(\mathcal{G}(V'))$, $Win_1(\mathcal{G}(V')))$
Let $\sigma \in \{0,1\}$ such that $c(V') \in \mathcal{F}_\sigma$;
**for** $c' \in c(V')$ **do**
    $(W_0', W_1') \leftarrow$ SolveCMG$(V' \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}(V')))$
    **if** $W_\sigma' \neq V' \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}(V'))$ **then**
        $X \leftarrow Attr_{\bar{\sigma}}(W_{\bar{\sigma}}', \mathcal{A}(V'))$;
        $(W_0'', W_1'') \leftarrow$ SolveCMG$(V' \setminus X)$;
        $W_\sigma \leftarrow W_\sigma''$, $W_{\bar{\sigma}} \leftarrow V' \setminus W_\sigma$;
        **return** $(W_0, W_1)$
    **end**
**end**
$W_\sigma \leftarrow V'$, $W_{\bar{\sigma}} \leftarrow \emptyset$;
**return** $(W_0, W_1)$
```

Figure 1 The recursive algorithm for colored Muller games.

A standard analysis of this algorithm produces running time $O(|C|^{|C|} \cdot |V|^{|V|})$, see [16]. Our analysis below improves this by showing that the multiplicative factors $|C|^{|C|}$ and $|V|^{|V|}$ in this estimate can be replaced with $|C|!$ and $\binom{|V|}{|C|}$, respectively.

▶ **Lemma 11.** *During the call of SolveCMG(V), the function SolveCMG is recursively called at most $|C|!\binom{|V|}{|C|}|V|$ times.*

Proof. If $|c(V')| = 0$, then no SolveCMG function is recursively called. Because $\mathcal{A}(V')$ is an arena, if SolveCMG(V') is called then $|V'| \neq 1$. If $|V'| = 2$ then for all non-empty sets $V'' \subseteq V'$ and $\sigma \in \{0,1\}$, $Attr_\sigma(V'', \mathcal{A}(V')) = V'$; hence, SolveCMG is recursively called $|c(V')|$ times. If $|c(V')| = 1$ then SolveCMG is recursively called for $|c(V')|$ times.

Assume that $|V'| > 2$, $|c(V')| > 1$, and for all V'' with $|V''| < |V'|$, during the call of SolveCMG(V''), the function SolveCMG is recursively called at most $|c(V'')|!\binom{|V''|}{|c(V'')|}|V''|$ times. For each $c' \in c(V')$, the set $V' \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}(V'))$ has at most $|V'| - 1$ vertices and $|c(V')| - 1$ colours. For $c' \in c(V')$ with $Win_\sigma(\mathcal{G}(V' \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}(V')))) \neq V' \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}(V'))$, we have $W_{\bar{\sigma}}' = Win_{\bar{\sigma}}(\mathcal{G}(V' \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}(V'))))$. Let $X = Attr_{\bar{\sigma}}(W_{\bar{\sigma}}', \mathcal{A}(V'))$. Since $|W_{\bar{\sigma}}'| \geq 2$, the set $V' \setminus X$ contains at most $|V'| - 2$ vertices and $|c(V')|$ colours. By hypothesis, during the call of SolveCMG(V'), the function SolveCMG is recursively called at most

$$|c(V')| + 1 + |c(V')|(|c(V')| - 1)!\tbinom{|V'|-1}{|c(V')|-1}(|V'| - 1) + |c(V')|!\tbinom{|V'|-2}{|c(V')|}(|V'| - 2)$$

times. This value is bounded from above by

$$|c(V')| + 1 + |c(V')|!\binom{|V'|}{|c(V')|}(|V'| - 1).$$

Now there are 2 cases:

1. $|c(V')| = 2$: Then

$$\begin{aligned}&|c(V')|!\binom{|V'|}{|c(V')|}|V'| - (|c(V')| + 1 + |c(V')|!\binom{|V'|}{|c(V')|}(|V'| - 1))\\ =&|c(V')|!\binom{|V'|}{|c(V')|} - |c(V')| - 1 \geq 2!\binom{3}{2} - 3 = 3\end{aligned}$$

2. $|c(V')| > 2$: Then

$$\begin{aligned}&|c(V')|!\binom{|V'|}{|c(V')|}|V'| - (|c(V')| + 1 + |c(V')|!\binom{|V'|}{|c(V')|}(|V'| - 1))\\ =&|c(V')|!\binom{|V'|}{|c(V')|} - |c(V')| - 1 \geq |c(V')|! - |c(V')| - 1 > 0\end{aligned}$$

Therefore, during the call of SolveCMG(V'), the function SolveCMG is recursively called at most $|c(V')|!\binom{|V'|}{|c(V')|}|V'|$ times. By hypothesis, the proof is done. ◀

▶ **Theorem 12.** *There is an algorithm that, given colored Muller game $\mathcal{G}$ computes $Win_0(\mathcal{G})$ and $Win_1(\mathcal{G})$ in time $O(|C|!\binom{|V|}{|C|}|V||E|)$ and space $O(|\mathcal{G}| + |C||V|)$.*

Proof. Consider the algorithm in Figure 1. Apply SolveCMG(V) to compute $Win_0(\mathcal{G})$ and $Win_1(\mathcal{G})$. The recursive depth of the algorithm is at most $|C|$ and $\mathcal{G}$ is memorized globally. In each iteration, only $O(|V|)$ space is applied to memorize the vertex set. Therefore, the algorithm takes $O(|\mathcal{G}| + |C||V|)$ space. By Lemma 11, the function SolveCMG is recursively called for at most $|C|!\binom{|V|}{|C|}|V|$ times. We need to estimate the running time in two parts of the algorithm:

- *Part 1: The running time within the loop "for $c' \in c(V')$ do".* In each enumeration of the color c', there is a corresponding recursive call on SolveCMG. Every time when we have $W'_\sigma \neq V' \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}(V'))$, there is also a corresponding recursive call on SolveCMG. Since the function SolveCMG is recursively called for at most $|C|!\binom{|V|}{|C|}|V|$ times, this part takes $O(|C|!\binom{|V|}{|C|}|V||E|)$ time.
- *Part 2: The running time outside the loop "for $c' \in c(V')$ do".* As SolveCMG is recursively called for at most $|C|!\binom{|V|}{|C|}|V|$ times, the running time bound for this part of the algorithm is also $O((|C|!\binom{|V|}{|C|}|V| + 1)|V|)$.

Therefore, the algorithm takes $O(|C|!\binom{|V|}{|C|}|V||E|)$ time. Note that the correctness of the algorithm is provided by Lemma 8. ◀

3.1 Application to Muller, Rabin and Streett games

Since Muller games are colored Muller games in which each vertex has its own color, there is also a recursive algorithm for computing winning regions of Muller games. In this case, Lemma 11 shows that the function SolveCMG is recursively called at most $|V|!|V|$ times. Hence, Theorem 12 implies the next corollary:

▶ **Corollary 13.** *There is a recurisve algorithm that, given Muller game $\mathcal{G}$ computes $Win_0(\mathcal{G})$ and $Win_1(\mathcal{G})$ in time $O(|V|!|V||E|)$ and space $O(|\mathcal{G}| + |V|^2)$.*

Through this corollary, by transforming Rabin conditions into Muller conditions, we can also provide a recursive algorithm for deciding Rabin games. The algorithm is presented in Figure 2.

▶ **Theorem 14.** *We have the following:*

1. *There exists an algorithm that, given Rabin or Streett game $\mathcal{G}$, computes $Win_0(\mathcal{G})$ and $Win_1(\mathcal{G})$ in time $O(|V|!|V|(|E|+k|V|))$ and space $O(|\mathcal{G}|+|V|^2)$.*
2. *There exists an algorithm that, given KL game $\mathcal{G}$ computes $Win_0(\mathcal{G})$ and $Win_1(\mathcal{G})$ in time $O(|V|!|V|(|E|+t|V|))$ and space $O(|\mathcal{G}|+|V|^2)$.*

Proof. Consider the algorithm above for Rabin games. We apply SolveRG(V) to compute $Win_0(\mathcal{G})$ and $Win_1(\mathcal{G})$. Compared with the recursive algorithm of Muller games, the algorithm only changes the computing of σ. Therefore, the function SolveRG is recursively called at most $|V|!|V|$ times. Also each computation of σ takes $O(|k||V|)$ time. By Corollary 13, the algorithm takes time $O(|V|!|V|(|E|+k|V|))$ and space $O(|\mathcal{G}|+|V|^2)$. For Streett games and KL games, similar arguments are applied. ◀

```
**Global Storage:** A Rabin game $\mathcal{G} = (\mathcal{A}, (U_1, V_1), \ldots, (U_k, V_k))$
**Function:** SolveRG($V'$)
**Input**: A vertex set $V'$ with $\mathcal{A}(V')$ is an arena
**Output**: $(Win_0(\mathcal{G}(V')), Win_1(\mathcal{G}(V')))$
If for all $i \in \{1, \ldots, k\}$ we have $V \cap U_i \neq \emptyset \implies V \cap V_i \neq \emptyset$ then $\sigma = 1$, otherwise $\sigma = 0$.
**for** $v \in V'$ **do**
    $(W'_0, W'_1) \leftarrow$ SolveRG$(V' \setminus Attr_\sigma(\{v\}, \mathcal{A}(V')))$
    **if** $W'_\sigma \neq V' \setminus Attr_\sigma(\{v\}, \mathcal{A}(V'))$ **then**
        $X \leftarrow Attr_{\bar{\sigma}}(W'_{\bar{\sigma}}, \mathcal{A}(V'))$, $(W''_0, W''_1) \leftarrow$ SolveRG$(V' \setminus X)$;
        $W_\sigma \leftarrow W''_\sigma$, $W_{\bar{\sigma}} \leftarrow V' \setminus W_\sigma$;
        **return** $(W_0, W_1)$
    **end**
**end**
$W_\sigma \leftarrow V'$, $W_{\bar{\sigma}} \leftarrow \emptyset$;
**return** $(W_0, W_1)$
```

Figure 2 The recursive algorithm for Rabin games.

4 Dynamic programming algorithms for deciding regular games

In this section, we provide dynamic programming algorithms for all regular games. First, in Section 4.1 we provide a dynamic version of the recursive algorithm in Figure 3. Then in Sections 4.2–4.5, the next set of all dynamic algorithms for solving the regular games will utilize both Lemma 8 and Lemma 10.

Let $\mathcal{G} = (\mathcal{A}, c, (\mathcal{F}_0, \mathcal{F}_1))$ be a colored Muller game where $V = \{v_1, v_2, \ldots, v_n\}$. We need to code subsets of V as binary strings. Therefore, we assign a n-bit binary number i to each non-empty pseudo-arena $\mathcal{A}(S_i)$ in $\mathcal{G}$ so that $S_i = \{v_j \mid \text{the } j\text{th bit of } i \text{ is } 1\}$. We will use this notation for all our algorithms in this section.

4.1 Algorithm 1 for Colored Muller Games

Consider the algorithm in Figure 3. This is a dynamic programming version of the recursive algorithm in Figure 1. The algorithm, given a colored Muller game $\mathcal{G}$ as input, returns the collections $Win_0(\mathcal{G})$ and $Win_1(\mathcal{G})$. The correctness of the algorithm is guaranteed by both Lemma 8 and Lemma 10. Thus, we have the following theorem:

▶ **Theorem 15.** *There is an algorithm that solves colored Muller games in time $O(2^{|V|}|C||E|)$ and space $O(|\mathcal{G}|+2^{|V|}|V|)$.*

Proof. We use the Algorithm 1 in Figure 3. Note that we apply the binary trees to maintain $\mathcal{F}_\sigma$s, W_0 and W_1. For each S_i, the algorithm takes $O(|C||E|)$ time to compute $W_0(S_i)$ and $W_1(S_i)$. Therefore, this algorithm runs in $O(2^{|V|}|C||E|)$ time. Since $\mathcal{F}_\sigma$s, W_0 and W_1 are encoded by binary trees, the algorithm takes $O(|\mathcal{G}| + 2^{|V|}|V|)$ space. ◀

```
**Input**: A colored Muller game $\mathcal{G} = (\mathcal{A}, c, (\mathcal{F}_0, \mathcal{F}_1))$
**Output**: $Win_0(\mathcal{G})$, $Win_1(\mathcal{G})$
$W_0 \leftarrow \emptyset$, $W_1 \leftarrow \emptyset$, $W_0(\emptyset) \leftarrow \emptyset$, $W_1(\emptyset) \leftarrow \emptyset$;
for $i = 1$ to $2^n - 1$ do
    $S_i \leftarrow \{v_j \mid$ the $j$th bit of $i$ is $1\}$;
    if $\mathcal{A}(S_i)$ is not an arena then
        break;
    Let $\sigma \in \{0,1\}$ such that $c(S_i) \in \mathcal{F}_\sigma$;
    $is\_win$ =true;
    for $c' \in c(S_i)$ do
        $Y \leftarrow Attr_\sigma(c^{-1}(c'), \mathcal{A}(S_i))$
        if $Y \neq S_i$ and $W_\sigma(S_i \setminus Y) \neq S_i \setminus Y$ then
            $is\_win$ =false, $X \leftarrow Attr_{\bar{\sigma}}(W_{\bar{\sigma}}(S_i \setminus Y), \mathcal{A}(S_i))$;
            $W_\sigma(S_i) \leftarrow W_\sigma(S_i \setminus X)$, $W_{\bar{\sigma}}(S_i) \leftarrow S_i \setminus W_\sigma(S_i)$;
            break;
        end
    end
    if $is\_win$ =true then
        $W_\sigma(S_i) \leftarrow S_i$, $W_{\bar{\sigma}}(S_i) \leftarrow \emptyset$;
end
return $W_0(V)$ and $W_1(V)$
```

Figure 3 Algorithm 1 for colored Muller games.

4.2 Algorithm 2 for Colored Muller Games

In this section, we utilize the concept of full win for the players, see Definition 7. The new dynamic algorithm, Algorithm 2, is presented in Figure 4. The algorithm takes colored Muller game $\mathcal{G} = (\mathcal{A}, c, (\mathcal{F}_0, \mathcal{F}_1))$ as input. Lemma 10 guarantees correctness of the algorithm. During the running process, this dynamic algorithm partitions all subgames $\mathcal{G}(S_i)$ into the following three collections of subsets of V:

- $P_0 = \{S_i \mid i \in [1, 2^n - 1]$ and Player 0 fully wins $\mathcal{G}(S_i)\}$,
- $P_1 = \{S_i \mid i \in [1, 2^n - 1]$ and Player 1 fully wins $\mathcal{G}(S_i)\}$, and
- $Q = \{S_i \mid i \in [1, 2^n - 1]$ and no player fully wins $\mathcal{G}(S_i)\}$.

Now we provide analysis of Algorithm 2 presented in Figure 4.

▶ **Lemma 16.** *Algorithm 2 computes P_0, P_1 and Q for a colored Muller game in $O(2^{|V|}|V||E|)$ time and $O(|\mathcal{G}| + 2^{|V|})$ space.*

Proof. We use the Algorithm 2 in Figure 4. Note that we apply the binary trees to maintain $\mathcal{F}_\sigma$s, P_0, P_1 and Q. For each S_i, the algorithm takes $O(|V||E|)$ time to determine the set to add S_i. Therefore, this algorithm runs in $O(2^{|V|}|V||E|)$ time. Since P_0, P_1 and Q are encoded by binary trees, the algorithm takes $O(|\mathcal{G}| + 2^{|V|})$ space. ◀

▶ **Lemma 17.** *Let $\mathcal{A}(X)$ and $\mathcal{A}(Y)$ be 1-traps. If Player 0 fully wins $\mathcal{G}(X)$ and $\mathcal{G}(Y)$ then Player 0 fully wins $\mathcal{G}(X \cup Y)$.*

Proof. We construct a winning strategy for Player 0 in $\mathcal{G}(X \cup Y)$ as follows. If the token is in $Attr_0(X, \mathcal{A}(X \cup Y))$, Player 0 forces the token into X and once the token arrives at X, Player 0 follows the winning strategy in $\mathcal{G}(X)$. Otherwise, Player 0 follows the winning strategy in $\mathcal{G}(Y)$. ◀

▶ **Lemma 18.** *If for all $S_i \in P_0$, the arena $\mathcal{A}(S_i)$ is not a 1-trap in $\mathcal{G}$, then $Win_0(\mathcal{G}) = \emptyset$ and $Win_1(\mathcal{G}) = V$. Otherwise, let $\mathcal{A}(S_{max})$ be the maximal 1-trap in $\mathcal{G}$ so that $S_{max} \in P_0$. Then $Win_0(\mathcal{G}) = S_{max}$ and $Win_1(\mathcal{G}) = V \setminus S_{max}$.*

Proof. For the first part of the lemma, assume that $Win_0(\mathcal{G}) \neq \emptyset$. Now note that $\mathcal{A}(Win_0(\mathcal{G}))$ is 1-trap such that Player 0 fully wins $\mathcal{G}(Win_0(\mathcal{G}))$. This contradicts with the assumption of the first part. For the second part, consider all 1-traps $\mathcal{A}(X)$ with $X \in P_0$. Player 0 fully wins the games $\mathcal{G}(X)$ in each of these 1-traps by definition of P_0. By Lemma 17, Player 0 fully wins the union of these 1-traps. Clearly, this union is $S_{max} \in P_0$. Consider $V \setminus S_{max}$. This set determines a 0-trap. Suppose Player 1 does not win $\mathcal{G}(V \setminus S_{max})$ fully. Then there exists a 1-trap $\mathcal{A}(Y)$ in game $\mathcal{G}(V \setminus S_{max})$ such that Player 0 fully wins $\mathcal{G}(Y)$. For every Player 1 position in $y \in Y$ and outgoing edge (y, x) we have either $x \in Y$ or $x \in S_{max}$. This implies $\mathcal{A}(S_{max} \cup Y)$ is 1-trap such that Player 0 fully wins $\mathcal{G}(S_{max} \cup Y)$. So, $S_{max} \cup Y$ must be in P_0. This contradicts with the choice of S_{max}. ◀

By Lemmas 16 and 18, we have proved the following theorem.

▶ **Theorem 19.** *There exists an algorithm that decides the colored Muller games $\mathcal{G}$ in time $O(2^{|V|}|V||E|)$ and space $O(|\mathcal{G}| + 2^{|V|})$.* ⌟

```
Input: A colored Muller game $\mathcal{G} = (\mathcal{A}, c, (\mathcal{F}_0, \mathcal{F}_1))$
Output: The partitioned sets $P_0$, $P_1$ and $Q$.
$P_0 \leftarrow \emptyset$, $P_1 \leftarrow \emptyset$, $Q \leftarrow \emptyset$;
for $i = 1$ to $2^n - 1$ do
    $S_i \leftarrow \{v_j \mid$ the $j$th bit of $i$ is $1\}$;
    if $\mathcal{A}(S_i)$ is not an arena then
        break;
    Let $\sigma \in \{0, 1\}$ such that $c(S_i) \in \mathcal{F}_\sigma$;
    $AllAttr_0$ =true, $AllAttr_1$ =true;
    for $c' \in c(S_i)$ do
        if $Attr_\sigma(c^{-1}(c'), \mathcal{A}(S_i)) \neq S_i$ and $S_i \setminus Attr_\sigma(c^{-1}(c'), \mathcal{A}(S_i)) \notin P_\sigma$ then
            $AllAttr_\sigma$ =false;
            break
        end
    if $AllAttr_\sigma$ =true then
        $P_\sigma \leftarrow P_\sigma \cup \{S_i\}$;
    else
        for $v \in S_i$ do
            if $Attr_{\bar\sigma}(\{v\}, \mathcal{A}(S_i)) \neq S_i$ and $S_i \setminus Attr_{\bar\sigma}(\{v\}, \mathcal{A}(S_i)) \notin P_{\bar\sigma}$ then
                $AllAttr_{\bar\sigma}$ =false;
                break
            end
        if $AllAttr_{\bar\sigma}$ = true then
            $P_{\bar\sigma} \leftarrow P_{\bar\sigma} \cup \{S_i\}$;
        else
            $Q \leftarrow Q \cup \{S_i\}$;
        end
    end
end
return $P_0$, $P_1$ and $Q$
```

Figure 4 Algorithm 2 for partitioning subgames of a colored Muller game.

4.3 Applications to Muller and McNaughton games

It is not too hard to see that for Muller games and McNaughton games, we can easily recast the algorithms presented in Sections 4.1 and 4.2. Indeed, the transformation of Muller games to colored Muller games is obvious. Hence, by applying Theorem 19 to Muller games we get the following result:

▶ **Theorem 20.** *There exists an algorithm that decides Muller game $\mathcal{G}$ in time $O(2^{|V|}|V||E|)$ and space $O(|\mathcal{G}| + 2^{|V|})$.* ⌟

The transformation of McNaughton games into colored Muller games is also easy. Each position v in W gets its own color, and all positions outside of W get the same new colour. Hence, we can apply both Theorems 15 and 19 to McNaughton games:

▶ **Theorem 21.** *Each of the following is true:*
1. *There exists an algorithm that decides McNaughton games $\mathcal{G}$ in $O(2^{|V|}|W||E|)$ time and $O(|\mathcal{G}| + 2^{|V|}|V|)$ space.*
2. *There exists an algorithm that decides McNaughton games $\mathcal{G}$ in $O(2^{|V|}|V||E|)$ time and $O(|\mathcal{G}| + 2^{|V|})$ space.* ⌟

4.4 Enumeration Lemma

This is an auxiliary section that will provide us with an enumeration technique. This technique will then be used in designing an algorithm to decide Rabin and Streett games by transforming these games into Muller games in a more efficient manner.

Let n be a natural number and $\mathcal{S} = \{b_1, \ldots, b_t\}$ be a set of n-bit binary integers, where n is the size of the vertex set $V = \{v_1, \ldots, v_n\}$ of the arena. Each b_i represents the characteristic function of the set $V_i \subseteq V$: $b_i(v) = 1$ iff $v \in V_i$. We want to efficiently enumerate the collection $2^{V_1} \cup \ldots \cup 2^{V_t}$. Note that

$$2^{V_1} \cup \ldots \cup 2^{V_t} = \{x \in [0, 2^n) \mid \exists b \in \mathcal{S}(x \;\&\; b = x)\},$$

where x is the binary integer of length at most n, and the operation & is the bitwise *and* operation. Later we will use our enumeration of the collection

$$\mathcal{X} = \{x \in [0, 2^n) \mid \exists b \in \mathcal{S}(x \;\&\; b = x)\}$$

to transform the KL condition into Muller condition.

```
**Function**: Enumerate($\mathcal{S}, n$).
**Input**: $\mathcal{S}$ and $n$ where $\mathcal{S}$ is a set of $n$-bit binary integers.
**Output**: $\mathcal{X} = \{x \in [0, 2^n) \mid \exists b \in \mathcal{S}(x \;\&\; b = x)\}$.
if $\mathcal{S} = \emptyset$ then
    return $\emptyset$
if $n = 0$ then
    return $\{0\}$
$\mathcal{S}'_0 \leftarrow \emptyset$, $\mathcal{S}'_1 \leftarrow \emptyset$
for $b \in \mathcal{S}$ do
    if $b \mod 2 = 0$ then
        $\mathcal{S}'_0 \leftarrow \mathcal{S}'_0 \cup \{\frac{b}{2}\}$
    else
        $\mathcal{S}'_0 \leftarrow \mathcal{S}'_0 \cup \{\frac{b-1}{2}\}$, $\mathcal{S}'_1 \leftarrow \mathcal{S}'_1 \cup \{\frac{b-1}{2}\}$
    end
end
$\mathcal{X}'_0 \leftarrow$ Enumerate($\mathcal{S}'_0, n-1$), $\mathcal{X}'_1 \leftarrow$ Enumerate($\mathcal{S}'_1, n-1$), $\mathcal{X} \leftarrow \emptyset$
for $x' \in \mathcal{X}'_0$ do
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{2x'\}$
for $x' \in \mathcal{X}'_1$ do
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{2x' + 1\}$
return $\mathcal{X}$
```

Figure 5 Algorithm for Enumerate($\mathcal{S}, n$).

Note that the brute-force algorithm that enumerates the collection $\mathcal{X} = 2^{V_1} \cup \ldots \cup 2^{V_t}$ runs in time $O(2^n \cdot t)$. In our enumeration we want to remove the dependence on t as t can be exponential on n. We apply the function Enumerate($\mathcal{S}, n$) shown in Figure 5 and obtain the next lemma.

▶ **Lemma 22 (Enumeration Lemma).** *Given the set $\mathcal{S} = \{b_1, \ldots, b_t\}$ of n-bit binary integers, we can enumerate the collection $\mathcal{X} = \{x \in [0, 2^n) \mid \exists b \in \mathcal{S}(x \ \& \ b = x)\}$ in time $O(2^n n)$ and space $O(2^n)$.* ⌟

4.5 Applications to Rabin and Streett games

We can naturally transform Rabin games, Streett games, and KL games into Muller games, and then apply our dynamic algorithms from Section 4.3 to thus obtained Muller games. These transformations are the following:

- For Rabin games and $X \subseteq V$, if for $i \in \{1, \ldots, k\}$ we have $X \cap U_i \neq \emptyset \implies X \cap V_i \neq \emptyset$ then $X \in \mathcal{F}_1$, otherwise $X \in \mathcal{F}_0$.
- For Streett games and $X \subseteq V$, if there is an $i \in \{1, \ldots, k\}$ such that $X \cap U_i \neq \emptyset$ and $X \cap V_i = \emptyset$, then $X \in \mathcal{F}_1$, otherwise $X \in \mathcal{F}_0$.
- For KL games and $X \subseteq V$, if for $i \in \{1, \ldots, t\}$ we have $u_i \in X \implies X \not\subseteq S_i$ then $X \in \mathcal{F}_1$, otherwise $X \in \mathcal{F}_0$.

In these transformations one needs to be careful with the parameters k and t for Rabin and Streett games and KL games, respectively. They add additional running time costs, especially k and t can have exponential values in $|V|$. For instance, the direct translation of Rabin games to Muller games requires, for each pair (U_i, V_i) in the Rabin winning condition, to build the collection of sets X such that $X \cap U_i \neq \emptyset$ and $X \cap V_i = \emptyset$. The collection of all these sets X form the Muller condition set $(\mathcal{F}_0, \mathcal{F}_1)$. As the index k is $O(2^{2|V|})$, the direct transformation above is expensive. Our goal is to analyze the transformations of Rabin games to Muller games.

We start with transforming KL games $\mathcal{G} = (\mathcal{A}, (u_1, S_1), \ldots, (u_t, S_t))$ to Muller games. So, for the given $\mathcal{G}$, we define the Muller game $\mathcal{G}'$ with $(\mathcal{F}_0, \mathcal{F}_1)$ as follows:

$$X \in \mathcal{F}_0 \text{ if for some pair } (u_i, S_i) \text{ we have } u_i \in X \text{ and } X \subseteq S_i, \text{ otherwise } X \in \mathcal{F}_1$$

▶ **Lemma 23.** *The transformation from KL games $\mathcal{G}$ to Muller games $\mathcal{G}'$ takes $O(2^{|V|}|V|^2)$ time and $O(|\mathcal{G}| + 2^{|V|})$ space.*

Proof. We apply the binary encoding so that for $i \in [1, t]$, we have $u_i \in [0, n)$ and $S_i \in [0, 2^n)$. In the following, we apply binary trees to maintain sets of binary integers. We transform $\mathcal{G}$ into the Muller game $\mathcal{G}' = (\mathcal{A}, (\mathcal{F}_0, \mathcal{F}_1))$ where we also apply the binary encoding so that $\mathcal{F}_0 = \{X \in [0, 2^n) \mid$ there exists an $i \in [1, t]$ so that the u_i-th bit of X is 1 and $X \ \& \ S_i = X\}$ and $\mathcal{F}_1 = \{0, 1, \ldots, 2^n - 1\} \backslash \mathcal{F}_0$. Then let $\mathcal{S}_i = \{S_j \mid j \in [1, t]$ and $u_j = i\}$ for $i \in [0, n)$. By Lemma 22, for each $i \in [0, n)$, we compute $\{X \in [0, 2^n) \mid \exists_{S \in \mathcal{S}_i} X \ \& \ S = X\}$ in time $O(2^n \cdot n)$ and space $O(2^n)$, and then compute $\{X \in [0, 2^n) \mid$ the i-th bit of X is 1 and $\exists_{S \in \mathcal{S}_i} X \ \& \ S = X\}$ in time $O(2^n)$ by traversing the binary trees, checking and deleting subtrees at depth i. Since $\bigcup_{i \in [0,n)} \{X \in [0, 2^n) \mid$ the i-th bit of X is 1 and $\exists_{S \in \mathcal{S}_i} X \ \& \ S = X\} = \mathcal{F}_0$, we reuse $O(2^n)$ space for each $i \in [0, n)$ and use another $O(2^n)$ space to record the prefix union results. Since $\mathcal{F}_1$ is computed from $\mathcal{F}_0$ in time $O(2^{|V|})$ by computing the complement of the tree, this is a transformation from KL games to Muller games and the transformation takes $O(2^{|V|}|V|^2)$ time and $O(|\mathcal{G}| + 2^{|V|})$ space. ◀

As an immediate corollary we get the following complexity-theoretic result for KL games.

▶ **Theorem 24.** *There exists an algorithm that, given a KL game $\mathcal{G}$, decides $\mathcal{G}$ in $O(2^{|V|}|V||E|)$ time and $O(|\mathcal{G}| + 2^{|V|})$ space.* ⌟

Now we transform Rabin games $\mathcal{G}$ to Muller games. As we mentioned above, the direct translation to Muller games is costly. Our goal is to avoid this cost through KL games. The following lemma is easy:

▶ **Lemma 25.** *Let $X \subseteq V$ and let (U_i, V_i) be a winning pair in Rabin game $\mathcal{G}$. Set $Y_i = U_i \setminus V_i$ and $Z_i = V \setminus V_i$. Then $X \cap U_i \neq \emptyset$ and $X \cap V_i = \emptyset$ if and only if $X \cap Y_i \neq \emptyset$ and $X \subseteq Z_i$.*

Thus, we can replace the winning condition $(U_1, V_1), \dots (U_k, V_k)$ in Rabin games with the equivalent winning condition $(Y_1, Z_1), \dots, (Y_k, Z_k)$. We still have Rabin winning condition but we use this new winning condition $(Y_1, Z_1), \dots, (Y_k, Z_k)$ to build the desired KL game:

▶ **Lemma 26.** *The transformation from Rabin games $\mathcal{G}$ to KL games takes time $O(k|V|^2)$ and space $O(|\mathcal{G}| + 2^{|V|}|V|)$.*

Proof. Enumerate all pairs (U_i, V_i), compute $Y_i = U_i \setminus V_i$, $Z_i = V \setminus V_i$ and add all pairs (u_j, S_j) with $u_j \in Y_i$ and $S_j = Z_i$ into KL conditions. By applying binary trees, the transformation takes $O(k|V|^2)$ time and $O(|\mathcal{G}| + 2^{|V|}|V|)$ space. This preserves the winning sets W_0 and W_1. ◀

Thus, the transformed KL games can be viewed as a compressed version of Rabin games.

▶ **Corollary 27.** *The transformation from Rabin games $\mathcal{G}$ to Muller games $\mathcal{G}'$ takes $O((k + 2^{|V|})|V|^2)$ time and $O(|\mathcal{G}| + 2^{|V|}|V|)$ space.* ⌟

Note that deciding Rabin games is equivalent to deciding Streett games. Thus, combining the arguments above, we get the following complexity-theoretic result:

▶ **Theorem 28.** *Rabin and Streett games $\mathcal{G}$ can be decided in $O((k|V| + 2^{|V|}|E|)|V|)$ time and $O(|\mathcal{G}| + 2^{|V|}|V|)$ space.* ⌟

5 Conclusion

The algorithms presented in this work give rise to numerous questions that warrant further exploration. For instance, we know that explicitly given Muller games can be decided in polynomial time. Yet, we do not know if there are polynomial time algorithms that decide explicitly given McNaughton games. Another intriguing line of research is to investigate if there are exponential time algorithms that decide colored Muller games when the parameter $|C|$ ranges in the interval $[\sqrt{|V|}, |V|/a]$, where $a > 1$. It could also be very interesting to replace the factor $2^{|V|}$ with $2^{|W|}$ in the running time that decides McNaughton games. If this can be done, then one implies that the ETH is not applicable to McNaughton games as opposed to colored Muller games and Rabin games. In addition, this work could be extended to dynamic settings or experimental studies, as in [17, 25]. Its dynamic programming-style approach may also inspire new advances, akin to DP-assisted methods in games on graphs [29, 30, 31]. These all may uncover new insights and lead to even more efficient algorithms.

References

1 Henrik Björklund, Sven Sandberg, and Sergei Vorobyov. On fixed-parameter complexity of infinite games. In *The Nordic Workshop on Programming Theory (NWPT 2003)*, volume 34, pages 29–31. Citeseer, 2003.

2 Hans L Bodlaender, Michael J Dinneen, and Bakhadyr Khoussainov. On game-theoretic models of networks. In *International Symposium on Algorithms and Computation*, pages 550–561. Springer, 2001. `doi:10.1007/3-540-45678-3_47`.

3 Udi Boker. Why these automata types? In *LPAR*, volume 18, pages 143–163, 2018. `doi:10.29007/C3BJ`.

4 Cristian S Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 252–263, 2017. STOC 2017 Best Paper Award. `doi:10.1145/3055399.3055409`.

5 Antonio Casares, Thomas Colcombet, Nathanaël Fijalkow, and Karoliina Lehtinen. From muller to parity and rabin automata: Optimal transformations preserving (history) determinism. *TheoretiCS*, 3, 2024. `doi:10.46298/THEORETICS.24.12`.

6 Antonio Casares, Marcin Pilipczuk, Michał Pilipczuk, Uéverton S Souza, and KS Thejaswini. Simple and tight complexity lower bounds for solving rabin games. In *2024 Symposium on Simplicity in Algorithms (SOSA)*, pages 160–167. SIAM, 2024.

7 Laure Daviaud, Marcin Jurdziński, and KS Thejaswini. The strahler number of a parity game. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020*, page 123. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, 2020.

8 Daniele Dell'Erba and Sven Schewe. Smaller progress measures and separating automata for parity games. *Frontiers in Computer Science*, 4:936903, 2022. `doi:10.3389/FCOMP.2022.936903`.

9 Michael J Dinneen and Bakhadyr Khoussainov. Update networks and their routing strategies. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 127–136. Springer, 2000. `doi:10.1007/3-540-40064-8_13`.

10 Michael J Dinneen and Bakhadyr Khoussainov. Update games and update networks. *Journal of discrete Algorithms*, 1(1):53–65, 2003. `doi:10.1016/S1570-8667(03)00006-6`.

11 Stefan Dziembowski, Marcin Jurdzinski, and Igor Walukiewicz. How much memory is needed to win infinite games? In *Proceedings of Twelfth Annual IEEE Symposium on Logic in Computer Science*, pages 99–110. IEEE, 1997. `doi:10.1109/LICS.1997.614939`.

12 E Allen Emerson and Charanjit S Jutla. The complexity of tree automata and logics of programs. In *FoCS*, volume 88, pages 328–337, 1988.

13 E Allen Emerson and Charanjit S Jutla. Tree automata, mu-calculus and determinacy. In *FoCS*, volume 91, pages 368–377. Citeseer, 1991.

14 E Allen Emerson and Charanjit S Jutla. The complexity of tree automata and logics of programs. *SIAM Journal on Computing*, 29(1):132–158, 1999. `doi:10.1137/S0097539793304741`.

15 John Fearnley, Sanjay Jain, Sven Schewe, Frank Stephan, and Dominik Wojtczak. An ordered approach to solving parity games in quasi polynomial time and quasi linear space. In *Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software*, pages 112–121, 2017. `doi:10.1145/3092282.3092286`.

16 Nathanaël Fijalkow, Nathalie Bertrand, Patricia Bouyer-Decitre, Romain Brenguier, Arnaud Carayol, John Fearnley, Hugo Gimbert, Florian Horn, Rasmus Ibsen-Jensen, Nicolas Markey, Benjamin Monmege, Petr Novotný, Mickael Randour, Ocan Sankur, Sylvain Schmitz, Olivier Serre, and Mateusz Skomra. Games on graphs, 2023. To be published by Cambridge University Press. Editor: Nathanaël Fijalkow. `doi:10.48550/arXiv.2305.10546`.

17 Aniruddh Gandhi, Bakhadyr Khoussainov, and Jiamou Liu. Efficient algorithms for games played on trees with back-edges. *Fundamenta Informaticae*, 111(4):391–412, 2011. `doi:10.3233/FI-2011-569`.

18 Erich Grädel, Wolfgang Thomas, and Thomas Wilke. Automata, logics, and infinite games. lncs, vol. 2500, 2002.

19 Florian Horn. Streett games on finite graphs. In *Proc. 2nd Workshop Games in Design Verification (GDV)*. Citeseer, 2005.

20 Florian Horn. Explicit muller games are ptime. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2008.

21 Paul Hunter and Anuj Dawar. Complexity bounds for muller games. *Theoretical Computer Science (TCS)*, 2008.

22 Hajime Ishihara and Bakhadyr Khoussainov. Complexity of some infinite games played on finite graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 270–281. Springer, 2002. doi:10.1007/3-540-36379-3_24.

23 Marcin Jurdziński and Ranko Lazić. Succinct progress measures for solving parity games. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–9. IEEE, 2017.

24 Bakhadyr Khoussainov. Finite state strategies in one player mcnaughton games. In *International Conference on Discrete Mathematics and Theoretical Computer Science*, pages 203–214. Springer, 2003. doi:10.1007/3-540-45066-1_16.

25 Bakhadyr Khoussainov, Jiamou Liu, and Imran Khaliq. A dynamic algorithm for reachability games played on trees. In *Mathematical Foundations of Computer Science 2009: 34th International Symposium, MFCS 2009, Novy Smokovec, High Tatras, Slovakia, August 24-28, 2009. Proceedings 34*, pages 477–488. Springer, 2009. doi:10.1007/978-3-642-03816-7_41.

26 Bakhadyr Khoussainov and Anil Nerode. *Automata theory and its applications*, volume 21. Springer Science & Business Media, 2012.

27 Karoliina Lehtinen. A modal μ perspective on solving parity games in quasi-polynomial time. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 639–648, 2018. doi:10.1145/3209108.3209115.

28 Zihui Liang, Bakh Khoussainov, Toru Takisaka, and Mingyu Xiao. Connectivity in the presence of an opponent. In *31st Annual European Symposium on Algorithms (ESA 2023)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

29 Zihui Liang, Bakh Khoussainov, and Mingyu Xiao. Network control games played on graphs. *Theoretical Computer Science*, page 115123, 2025. doi:10.1016/J.TCS.2025.115123.

30 Zihui Liang, Bakh Khoussainov, and Haidong Yang. Topological network-control games. In *International Computing and Combinatorics Conference*, pages 144–156. Springer, 2023. doi:10.1007/978-3-031-49193-1_11.

31 Zihui Liang, Bakh Khoussainov, and Haidong Yang. Topological network-control games played on graphs. In *International Computing and Combinatorics Conference*, pages 15–26. Springer, 2024. doi:10.1007/978-981-96-1093-8_2.

32 Rupak Majumdar, Irmak Sağlam, and KS Thejaswini. Rabin games and colourful universal trees. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 213–231. Springer, 2024.

33 Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993. doi:10.1016/0168-0072(93)90036-D.

34 Daniel Neider, Roman Rabinovich, and Martin Zimmermann. Down the borel hierarchy: Solving muller games via safety games. *Theoretical Computer Science*, 560:219–234, 2014. doi:10.1016/J.TCS.2014.01.017.

35 Anil Nerode, Jeffrey B Remmel, and Alexander Yakhnis. Mcnaughton games and extracting strategies for concurrent programs. *Annals of Pure and Applied Logic*, 78(1-3):203–242, 1996. doi:10.1016/0168-0072(95)00032-1.

36 Paweł Parys. Parity games: Zielonka's algorithm in quasi-polynomial time. In *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2019. MFCS 2019 Best Paper Award.

37 Nir Piterman and Amir Pnueli. Faster solutions of rabin and streett games. In *21st Annual IEEE Symposium on Logic in Computer Science (LICS'06)*, pages 275–284. IEEE, 2006. doi:10.1109/LICS.2006.23.

38 Michael Oser Rabin. *Automata on infinite objects and Church's problem*, volume 13. American Mathematical Soc., 1972.

39 Wieslaw 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.