Rabin Games and Colourful Universal Trees[⋆⋆⋆]

Rupak Majumdar[1], Irmak Sağlam[1], and K. S. Thejaswini[2,3(✉)]

[1] Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany
{rupak,isaglam}@mpi-sws.org
[2] Department of Computer Science, University of Warwick, Coventry, UK
[3] Institute of Science and Technology, Klosterneuburg, Austria
thejaswini.k.s@ista.ac.at

Abstract. We provide an algorithm to solve Rabin and Streett games over graphs with n vertices, m edges, and k colours that runs in $\tilde{O}\left(mn(k!)^{1+o(1)}\right)$ time and $O(nk\log k\log n)$ space, where $\tilde{O}$ hides poly-logarithmic factors. Our algorithm is an improvement by a super quadratic dependence on $k!$ from the currently best known run time of $O\left(mn^2(k!)^{2+o(1)}\right)$, obtained by converting a Rabin game into a parity game, while simultaneously improving its exponential space requirement.
Our main technical ingredient is a characterisation of progress measures for Rabin games using *colourful trees* and a combinatorial construction of succinctly-represented, *universal* colourful trees. Colourful universal trees are generalisations of universal trees used by Jurdziński and Lazić (2017) to solve parity games, as well as of Rabin progress measures of Klarlund and Kozen (1991). Our algorithm for Rabin games is a progress measure lifting algorithm where the lifting is performed on succinct, colourful, universal trees.

Keywords: Rabin games· Parity games· Colourful trees

1 Introduction

A *Rabin game* is a two-player infinite-duration game played on a directed, coloured graph, where each vertex has a finite set of *good* colours and a finite set of *bad* colours associated with it [29]. The two players Controller and Environment take turns to move a token along an edge to form a *play*, an infinite path in the graph. Such a play is winning for Controller if there is a colour that is a good colour for some vertex seen infinitely often along the path and is not a bad colour for any vertex seen infinitely often. Rabin games lie at the core of reactive synthesis for omega-regular specifications and efficient algorithms for Rabin games are of practical interest in synthesis tools.

⋆ This work is a part of the project VAMOS that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreements No 101020093. Rupak Majumdar was partially supported by the DFG project 389792660 TRR 248—CPEC.

⋆⋆ The full version of the paper [25] is available on arXiv at http://arxiv.org/abs/2401.07548

B. Finkbeiner and L. Kovács (Eds.): TACAS 2024, LNCS 14572, pp. 213–231, 2024.
https://doi.org/10.1007/978-3-031-57256-2_11

Rabin automata already appear in McNaughton's solution of Church's synthesis problem [4,26] and in Rabin's proof of the decidability of SnS [29], where it was first defined in the setting of infinite trees. To solve Church's synthesis problem for ω-regular specifications, represented by non-deterministic Büchi automata, there are two well-studied (polynomial-time equivalent) approaches: either reduce it to the emptiness problem for Rabin tree automata or solve a Rabin game.

Rabin conditions are also known to be suitable specifications for *general fairness constraints* [15]. Klarlund and Kozen [20] defined Rabin measures over graphs and applied them to prove program termination under a general fairness constraint. Indeed, the acceptance condition that defines *strong fairness*, i.e., if a given set of actions (edges) is enabled infinitely often (the source vertex is seen infinitely often), it is taken infinitely often, is naturally expressed by the complement of the Rabin condition, called the Streett condition [30].

Algorithmically, the problem of solving Rabin games was shown to be NP-complete by Emerson and Jutla [11,13] in the late 1980s. In the same paper, Emerson and Jutla, and independently, Pnueli and Rosner [28], gave an algorithm that takes time $O\big((nk)^{3k}\big)$ time, where n is the number of vertices of the game graph and k the number of colours.

Steady progress was made to solve Rabin games, and within a decade, Kupferman and Vardi [23] reduced the cubic dependence on n^k to a quadratic one by giving an algorithm to check non-emptiness in a Rabin tree automata in time $O\big(mn^{2k}k!\big)$. Later, Horn [16] gave a different solution to solve Streett games—and therefore Rabin games—with the same running time.

A lot of progress was simultaneously made on *parity games* [12], a special case of Rabin games where colours are assigned to each subset of states in a chain of subsets. Inspired by fixpoint evaluation algorithms [12] and the small progress measure algorithm [17] of Jurdziński for parity games, Piterman and Pnueli [27] gave a fast $O\big(mn^{k+1}kk!\big)$-time, $O(nk)$-space, algorithm for Rabin games. This algorithm used a concept of a measure to solve Rabin games.

The work of Piterman and Pnueli remained state-of-the-art for Rabin games until the quasi-polynomial breakthrough for parity games by Calude, Jain, Khoussainov, Li, and Stephan [1]. They gave a fixed parameter tractable algorithm (FPT) for Rabin games on k colours by converting it to a parity game and using the quasi-polynomial algorithm.

A Rabin game with n vertices, m edges, and k colours, can be reduced to a parity game over $N = nk^2k!$ vertices, $M = nk^2k!m$ edges, and $K = 2k+1$ colours [12]. By combining the reduction from Rabin to parity games and state-of-the-art algorithms for parity games [18,8,14,9] in a "space-efficient" manner, say of Jurdziński and Lazić [18], one can solve Rabin games in time $O\big(\max\{MN^{2.38}, 2^{O(K\log K)}\}\big)$, but exponential space (since the parity game is exponentially bigger).

On substitution of the values of M and N, the algorithm of Jurdziński and Lazić would take time at least proportional to $m(nk^2 \cdot k!)^{3.38}$ for games with n vertices, m edges and k colours. However, observe that the parity game obtained from a Rabin game is such that the number of vertices $N = nk^2k!$ is much larger than the number of colours $K = 2k+1$. Indeed, this results in $K \in o\big(\log(N)\big)$. For cases where the

number of vertices of the resulting parity game is much larger than the number of priorities, say the number of colours $(2k+1)$ is $o\big(\log(N)\big)$—which is the case above as k grows—Jurdziński and Lazić also give an analysis of their algorithm that would solve Rabin games in time $O\big(nmk!^{2+o(1)}\big)$. Closely matching this are the run times in the work of Fearnley et al. [14] who provide, among other bounds, a quasi-bi-linear bound of $O\big(MN\mathfrak{a}(N)^{\log\log N}\big)$, 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 space proportional to $(nk^2k!)\log(nk^2k!)$, which has a $k!$ dependence again.

Our Contribution. Our result breaks through the $2+o(1)$ barrier, while simultaneously using polynomial space, to give a fixed-parameter tractable algorithm for Rabin games. We show a new algorithm for Rabin games on graphs that runs in time $\tilde{O}(mn(k!)^{1+o(1)})$ time and $O(nk\log k\log n)$ space, for a game on n vertices, m edges, and k colours. Our algorithm improves the quadratic $(k!)^2$ dependence in the number of colours in the best current algorithms, while simultaneously using only polynomial space.

Our first technical contribution is a characterisation of winning states in Rabin games using "*colourful trees*," by generalizing previous work on Rabin measures on graphs by Klarlund and Kozen [20]. Using our characterisation, we provide an algorithm to compute winning states and strategies as a fixed point of a lifting function over the lattice of functions from vertices of a game to nodes of a colourful tree.

Our second contribution is the construction of a *universal* colourful tree that embeds any colourful tree with a given number of leaves and fixed set of colours. Universal trees are found underlying all the quasi-polynomial algorithms for parity games [18,6,19,8,21]. Our construction uses the theory of universal trees developed for parity games, especially that of Jurdziński and Lazić [18]. From our construction of universal colourful trees, we can also naturally construct an instance of universal graphs for Rabin objectives, where the definition of universal graph is as introduced by Colcombet and Fijalkow. Although constructing universal graphs directly give us a lifting algorithm, for the sake of completeness, we also provide a lifting algorithm that uses our construction of colourful universal trees. Therefore, we show how to construct a small universal colourful tree (our upper bound is tight up to a polynomial factor) that can be succinctly encoded and efficiently navigated.

By applying the lifting algorithm to our succinct universal colourful tree, we get our time and space bounds.

Just as Piterman and Pnueli's result generalized ranking techniques and progress measures for parity games, we generalize the notion of measures [20] and universal trees [18] central to the fastest algorithms for parity games to obtain our algorithm.

2 Preliminaries

We use $\mathbb{N}$ to denote the set of all natural numbers $\{0,1,2,\ldots\}$. A directed graph consists of a finite set of vertices V along with a binary relation E over the set of vertices

called the *edge set*. We write $u \to v$ to denote an edge $(u, v) \in E$. A finite (resp. infinite) *path* in a directed graph is a finite (resp. infinite) sequence of vertices such that a tuple formed by any two consecutive vertices in this sequence is an edge in E.

(c_0, C)-Colourful Ordered Trees. Let C be a finite set of colours and let $c_0 \notin C$ be a distinguished *root colour*. Informally, a (c_0, C)-colourful ordered tree with root colour c_0 is an ordered tree of height at most $|C| + 1$ whose root is associated with the colour $c_0 \notin C$, and whose every other node has a colour from C associated to it. As an exception, we allow some leaves to be left uncoloured, denoted by a "dummy colour" $\perp \notin C$. We also require that along any path from the root to a leaf, each node must have a different colour.

Formally, for a finite set C, we recursively define (c_0, C)-colourful trees

- if $C = \emptyset$, $(c_0, \langle\rangle)$ and $(c_0, \langle(\perp, \langle\rangle), \ldots, (\perp, \langle\rangle)\rangle)$ are $(c_0, \emptyset)$-colourful trees.
- if $C \neq \emptyset$, we say $\mathcal{T}$ is (c_0, C)-colourful tree if it is either
  - a (c_0, C')-colourful tree rooted at c_0 for some $C' \subsetneq C$; or
  - $\mathcal{T} = (c_0, \langle\mathcal{T}_1, \ldots, \mathcal{T}_\ell\rangle)$, and for all $i \in \{1, \ldots, \ell\}$, either there is a $c_i \in C$ and $\mathcal{T}_i$ is a $(c_i, C \setminus \{c_i\})$-colourful ordered tree, or $\mathcal{T}_i = (\perp, \langle\rangle)$. Note that these c_i need not be different from one another.

We define the *concatenation* of a (c_0, C_1)-colourful tree $\mathcal{T}_1 = \left(c_0, \left\langle \mathcal{T}_1^1, \ldots, \mathcal{T}_1^m \right\rangle\right)$ and a (c_0, C_2)-colourful tree $\mathcal{T}_2 = \left(c_0, \left\langle \mathcal{T}_2^1, \ldots, \mathcal{T}_2^\ell \right\rangle\right)$ as the $(c_0, C_1 \cup C_2)$-colourful tree denoted by $\mathcal{T}_1 \cdot \mathcal{T}_2$ as $\left(c_0, \left\langle \mathcal{T}_1^1, \ldots, \mathcal{T}_1^m, \mathcal{T}_2^1, \ldots, \mathcal{T}_2^\ell \right\rangle\right)$. For a root colour c_0, a number $\ell \in \mathbb{N}$, and a (c, C)-colourful ordered tree $\mathcal{T}$, we denote $\mathcal{T}^\ell$ to be the tree with ℓ many copies of $\mathcal{T}$, $(c_0, \langle\mathcal{T}, \mathcal{T}, \ldots, \mathcal{T}\rangle)$. When (c_0, C) is clear from context, we simply say "colourful tree."

Embedding Colourful Trees. Given a (c_0, C)-colourful tree $\mathcal{U}$ and a (c_0, C')-colourful tree $\mathcal{T}$, such that $C' \subseteq C$, we say $\mathcal{U}$ *embeds* $\mathcal{T}$ if $\mathcal{T} = (c_0, \langle\rangle)$, or $\mathcal{T} = (c_0, \langle\mathcal{T}_1, \ldots, \mathcal{T}_\ell\rangle)$ and $\mathcal{U} = (c_0, \langle\mathcal{U}_1, \ldots, \mathcal{U}_m\rangle)$ for some ℓ, m, and there is some increasing sequence of indices $1 \leq i_1 < i_2 < \cdots < i_\ell \leq m$ such that $\mathcal{U}_{i_j}$ embeds $\mathcal{T}_j$ recursively. Notice both $\mathcal{U}_{i_j}$ and $\mathcal{T}_j$ must be rooted at the same colour, say c_j and both are $\left(c_j, C \setminus \{c_j\}\right)$-colourful and $\left(c_j, C' \setminus \{c_j\}\right)$-colourful trees respectively.

Labelled Colourful Trees. In what follows, we shall additionally label colourful trees with labels from some linearly ordered set. It is more convenient to define such labelled colourful trees as prefix-closed sets of sequences, using the isomorphism between a (recursively defined) tree and its set of paths.

Let $\mathbb{L}$ be a set of labels with a linear ordering $<_{\mathbb{L}} \subseteq \mathbb{L} \times \mathbb{L}$. An $\mathbb{L}$-labelled (c_0, C)-colourful tree is a prefix-closed set of sequences over $\mathbb{L} \times (C \cup \{\perp\})$ where $\mathbb{L} \times (C \cup \{\perp\})$ is the Cartesian product of $\mathbb{L}$ and $(C \cup \{\perp\})$.

Given an element $\tau_0 \in \mathbb{L} \times (C \cup \{\perp\})$ and a sequence $\left(\tau_1, \tau_2, \ldots, \tau_j\right)$ in $(\mathbb{L} \times (C \cup \{\perp\}))^*$, we use $\odot$ to denote concatenation to the tuple, where we say $\tau_0 \odot \left(\tau_1, \tau_2, \ldots \tau_j\right) = (\tau_0, \tau_1, \tau_2, \ldots \tau_j)$. We extend this notation to sets of sequences $\mathscr{L}$, by also defining $\tau_0 \odot \mathscr{L} = \left\{(\tau_0, \tau_1, \tau_2, \ldots \tau_j) \mid (\tau_1, \tau_2, \ldots \tau_j) \in \mathscr{L}\right\}$.

We say a prefix-closed set $\mathscr{L} \subseteq (\mathbb{L} \times (C \cup \{\perp\}))^*$ is an $\mathbb{L}$-labelling of a (c_0, C)-colourful ordered tree $\mathcal{T}$

- if $\mathscr{T} = (c_0, \langle(\bot, \langle\rangle)^m\rangle)$, and $\mathscr{L}$ is the prefix closure of the set $\{(\alpha_1, \bot), \ldots, (\alpha_m, \bot)\}$ for some $\alpha_1 <_{\mathbb{L}} \alpha_2 <_{\mathbb{L}} \cdots <_{\mathbb{L}} \alpha_m \in \mathbb{L}$,
- if $\mathscr{T} = (c_0, \langle\mathscr{T}_1, \ldots, \mathscr{T}_m\rangle)$ then $\mathscr{L}$ is the prefix closure of the set

$$(\alpha_1, c_1) \odot \mathscr{L}_1 \cup (\alpha_2, c_2) \odot \mathscr{L}_2 \cup \cdots \cup (\alpha_m, c_m) \odot \mathscr{L}_m$$

for some $\alpha_1 \leq_{\mathbb{L}} \alpha_2 \leq_{\mathbb{L}} \cdots \leq_{\mathbb{L}} \alpha_m$ in $\mathbb{L}$, such that for all j,

- $\mathscr{T}_j$ is a $C \setminus \{c_j\}$-colourful tree rooted at c_j and $\mathscr{L}_j$ is an $\mathbb{L}$-labeling of $\mathscr{T}_j$,
- $c_j \in C \cup \{\bot\}$, and
- whenever $\alpha_j = \alpha_{j+1}$, we have $c_j \neq c_{j+1}$

Note that the root colour c_0 of $\mathscr{T}$ does not appear in $\mathscr{L}$; instead of tracking c_0 along with $\mathscr{L}$ explicitly, we implicitly assume the root colour of the tree $\mathscr{L}$ above is c_0.

We refer to elements of the prefix-closed set $\mathscr{L}$ of a labelled tree as *nodes* of the tree. For two nodes n_1 and n_2 in $\mathscr{L}$, we define *the greatest common ancestor*, written $\mathsf{GCA}(n_1, n_2)$, as the longest common prefix of n_1 and n_2. We define n_1 to be an *ancestor* of n_2 if $n_1 = \mathsf{GCA}(n_1, n_2)$. In particular, n_1 is a parent of n_2, written $n_1 = \mathrm{parent}(n_2)$, if n_1 is the largest node other than n_2 such that $n_1 = \mathsf{GCA}(n_1, n_2)$; we then say n_2 is a child of n_1.

The colouring of a node is defined to be the last colour occurring in the sequence: For the empty sequence $()$, we define $\mathsf{colour}(()) = c_0$, and $\mathsf{colour}((\alpha_1, c_{i_1}), \ldots, (\alpha_j, c_{i_j})) = c_{i_j}$. Furthermore we define $\mathsf{ColourSet} : \mathscr{L} \to 2^{C \cup \{c_0\}}$, which maps a node to the set of colours seen from the root to that node: $\mathsf{ColourSet}(n) = \{\mathsf{colour}(n') \mid n' = \mathsf{GCA}(n', n)\} \setminus \{\bot\}$.

Ordering. We define an ordering $\prec_{\mathscr{L}}$ on $\mathscr{L}$. First, we fix some arbitrary linear order on the set C and set colour $\bot$ to be larger than all the colours in C in the ordering. We compare elements by extending the linear order $<_{\mathbb{L}}$ over $\mathbb{L}$ and an arbitrary fixed order $<$ over C to a linear order over the set $\mathbb{L} \times (C \cup \{\bot\})$ lexicographically as follows: for two elements in $\mathbb{L} \times (C \cup \{\bot\})$, we declare $(\alpha_1, c_1) < (\alpha_2, c_2)$ if either $\alpha_1 <_{\mathbb{L}} \alpha_2$ or $\alpha_1 = \alpha_2$ and $c_1 < c_2$.

For two nodes $n_1, n_2 \in \mathscr{L}$, we define $n_1 \prec_{\mathscr{L}} n_2$ if either n_1 is a strict prefix of n_2, or if n_1 is lexicographically smaller than n_2 when viewed as sequences over $\mathbb{L} \times (C \cup \{\bot\})$.

Due to space constraints, the missing proofs can be found in the full version of the paper.

Example 1. Figure 1 depicts a $(\bullet, \{\bullet, \bullet, \bullet\})$-colourful tree, where the nodes denoted by $\circ$ represents uncoloured nodes. A fixed ordering on the set of colours $\bullet < \bullet < \bullet < \circ$, a labelling of this tree over $\mathbb{L} = \{1, 2\} \subseteq \mathbb{N}$ is the prefix closure of the following set $\{(1\bullet, 1\bullet, 1\circ), (1\bullet, 1\bullet, 2\bullet, 1\circ), (1\bullet, 1\bullet, 2\bullet, 2\circ), (1\bullet, 1\bullet, 1\bullet), (1\bullet, 1\bullet, 2\bullet, 2\circ), (1\circ), (2\bullet, 2\bullet), (2\bullet, 1\bullet, 1\bullet, 1\circ), (2\bullet, 1\bullet, 2\circ)\}$. The ordering $\prec_{\mathscr{L}}$, (represented by $\prec$) on some nodes is as follows: $() \prec (1\bullet) \prec (1\bullet, 1\bullet) \prec (1\circ) \prec (2\bullet, 2\bullet)$. The ordering in the nodes of the tree in the figure decreases when we go from a child to a parent, or we go "left" in the tree, but otherwise increases.

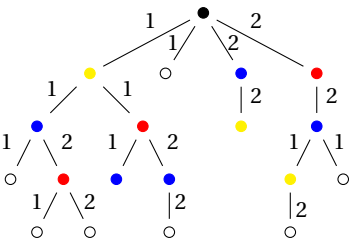


Fig. 1: A colourful tree.

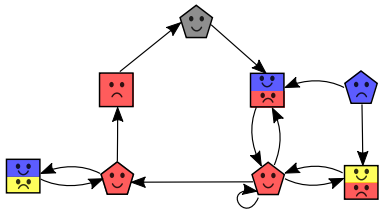


Fig. 2: A colourful Rabin graph $\mathcal{G}$ where all infinite paths satisfy the Rabin condition

3 Rabin measure and Colourful Decompositions

In this section, our aim is to understand the Rabin acceptance condition on graphs. We define such acceptance conditions and provide a local witness called a *Rabin measure* for graphs where all paths satisfy the Rabin condition.

A (c_0, C)-colourful *Rabin graph* $\mathcal{G}$ consists of (1) a directed graph (V, E), (2) a finite set C of *colours* and a special colour $c_0 \notin C$, and (3) for each vertex $v \in V$, a set of *good* colours $G_v \subseteq C \cup \{c_0\}$ for v and a set of *bad* colours $B_v \subseteq C$ for v. Observe that $c_0 \notin B_v$ for any v. We call each colour c in G_v a *good* colour for v, and each colour in B_v a *bad* colour for v.

We assume every vertex has some outgoing edge in the directed graph. An infinite path in $\mathcal{G}$ satisfies the *Rabin condition* if there is some colour c in $C \cup \{c_0\}$ such that c is a good colour for some v seen infinitely often along the path and c is not a bad colour for any v seen infinitely often along the path.

Example 2. Consider the $(\bullet, \{\bullet, \bullet, \bullet\})$-colourful Rabin game in Fig. 2. The colours that are in the good set of each vertex are represented with a smiley face in the same colour and those that are bad colours appear with a sad face. Although a vertex can have more than one colour assigned to it as a good colour (or a bad colour), we only consider at most one good and bad colour per vertex for this example. In our example, the leftmost vertex in the graph $\mathcal{G}$ in Fig. 2 has the singleton set $\{\bullet\}$ as the set of good colours and the set $\{\bullet\}$ as the set of bad colours. Similarly, the topmost vertex in Fig. 2 has the set $\{\bullet\}$ as the set of good colours and an empty set of bad colours. Observe that in the graph $\mathcal{G}$, any infinite path satisfies the Rabin condition. Indeed, for any infinite path there is some colour that is not a bad colour for any of the vertices that occur infinitely often and is a good colour for some vertex that occurs infinitely often. For example, if a path is such that all the vertices of $\mathcal{G}$ are visited infinitely often, then the colour $\bullet$ is not a bad colour of any vertex and the same colour $\bullet$ is a good colour of the topmost vertex.

As opposed to preexisting definition in literature of Rabin games that use Rabin pairs to represent the acceptance condition, we instead define two sets of colours associated to a vertex rather than a pair of subsets of vertices associated to a colour. This does not add more than a constant factor in terms of representation size.

A Measure for Rabin Graphs. We fix a (c_0, C)-colourful Rabin graph $\mathscr{G}$ with the underlying graph (V, E) with good colours for a vertex v denoted by G_v and the bad colours denoted by B_v. Let $\mathbb{L}$ be a linearly ordered set of labels, and let $\mathscr{L}$ be an $\mathbb{L}$ labelled (c_0, C) -coloured tree. We define $\mathscr{L}^\top = \mathscr{L} \cup \{\top\}$ by adjoining an element $\top$ to $\mathscr{L}$ and we extend the ordering $\prec_{\mathscr{L}}$ (denoted henceforth by $\prec$) to $\mathscr{L}^\top$, by declaring $t \prec \top$ for all $t \in \mathscr{L}$.

Consider a map $\mu : V \to \mathscr{L}^\top$. We call an edge $u \to v$ *consistent* with respect to μ, if either $\mu(u)$ is mapped to $\top$ or it satisfies the condition ($\mathsf{G}_\succ$ OR $\mathsf{G}_\downarrow$) AND B; for $\mathsf{G}_\succ$, $\mathsf{G}_\downarrow$, and B defined below.

($\mathsf{G}_\succ$) $\mu(u) \succ \mu(v)$
($\mathsf{G}_\downarrow$) $\mathsf{GCA}(\mu(u), \mu(v)) = \mu(u)$ and $\mathsf{colour}(\mu(u)) \in G_u$.
(B) $\mathsf{ColourSet}(\mu(u)) \cap B_u = \emptyset$

In words, $\mathsf{G}_\succ$ conveys that the measure μ decreases along the edge $u \to v$ and $\mathsf{G}_\downarrow$ says that the measure can increase along an edge but only into a descendent node and only when the colour of the node that is currently mapped to is a good colour for u. The condition represented by B says that none of the colours assigned to any ancestor of u is a bad colour for it.

If the map μ is clear from the context, we call an edge or a vertex consistent without mentioning the mapping. We say the relation and function $\mathsf{GCA}(\cdot, \top)$ and $\mathsf{colour}(\top)$ are undefined, and the condition $\mathsf{G}_\downarrow$ or B are not satisfied when $\mu(v)$ is mapped to $\top$ and $\mu(u)$ is not mapped to $\top$.

We say the map μ is a (c_0, C)-colourful *Rabin measure* for a graph $\mathscr{G}$ if all edges in E are consistent with respect to μ. A mapping from the vertices of a Rabin graph to the nodes of a tree ensures that an infinite play corresponds to an infinite set of nodes in a tree. If a mapping is consistent, then such a mapping serves as a witness to the fact that an infinite path in the Rabin graph satisfies the Rabin condition.

Our definition is a modification of Klarlund and Kozen's [20] notion of Rabin measures, following recent approaches to faster algorithms for parity games [18,8].

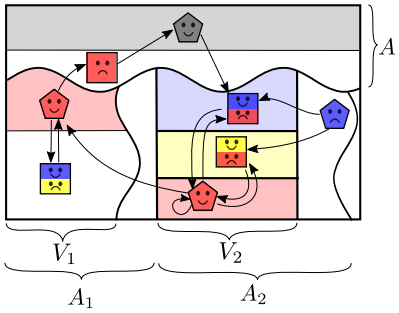
Colourful Decomposition. The Rabin measure, as with other progress measures, is based exclusively on local properties. Indeed, in the above case, we have a progress measure when each edge satisfies certain conditions. Before we show that Rabin measures capture winning sets of a graph, we define an intermediate structure, which we call *colourful decompositions*. These colourful decompositions of a Rabin graph highlight a recursive structure that captures the acceptance of all paths in a way which relates naturally to colourful trees. Colourful decompositions generalise attractor decompositions of parity games to Rabin games [7,19,8].

Consider a (c_0, C)-colourful Rabin graph $\mathscr{G}$. A (c_0, C)-colourful decomposition $\mathscr{D}$ of $\mathscr{G}$ is a recursive sub-division of vertices V of $\mathscr{G}$ into subsets of vertices defined as follows. If $C = \emptyset$, then we say $\mathscr{D} := \langle V \rangle$ is a (c_0, C)-colourful decomposition if and only if all infinite paths from all vertices in V visit a vertex v such that $c_0 \in G_v$. Else, if $C \neq \emptyset$ and if $|V| \geq 1$, and

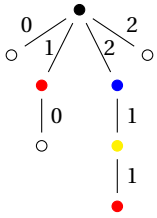
$$\mathscr{D} = \left\langle A, (c_1, V_1, \mathscr{D}_1, A_1), \ldots, \left(c_j, V_j, \mathscr{D}_j, A_j\right) \right\rangle$$

satisfies the following conditions:

– A is the set of all vertices in V such that all infinite paths starting from A in $\mathcal{G}$ visit some vertex $v \in V$ such that $c_0 \in G_v$;
– Set $W_1 = V \setminus A$. For $i \in 1, \ldots, j$,
  • V_i is a set of vertices which has no path to $W_i \setminus V_i$ and $c_i \notin B_v$ for all $v \in V_i$;
  • $\mathcal{D}_i$ is a $(c_i, C \setminus \{c_i\})$-colourful decomposition of V_i.
  • A_i is the set of all vertices in W_i such that all infinite paths from A_i within W_i visits some vertex in V_i;
  • $W_{i+1} = W_i \setminus A_i$.
– $W_{j+1} = \emptyset$.



(a) A colourful decomposition of a Rabin graph $\mathcal{G}$ where all paths satisfy the Rabin condition



(b) A labelled colourful tree into which the graph $\mathcal{G}$ has a Rabin measure.

Fig. 3: A colourful decomposition and tree for Rabin measure

The crux of this section is Theorem 1 below which shows the equivalence between Rabin measure, the existence of a colourful decomposition and a Rabin graph where all paths satisfy the Rabin condition.

Theorem 1. *The following three statements are equivalent for a* (c_0, C)*-colourful Rabin graph* $\mathcal{G}$*.*

1. *All infinite paths in* $\mathcal{G}$ *satisfy the Rabin condition.*
2. *There is a* (c_0, C)*-colourful decomposition* $\mathcal{D}$ *of the vertices of* $\mathcal{G}$*.*
3. *There is an* $\mathbb{L}$*-labelled* (c_0, C)*-colourful Rabin measure for* $\mathcal{G}$*, where no vertex is mapped to* $\top$ *for some linearly ordered infinite set* $\mathbb{L}$*.*

The theorem above is proved by showing $1 \Longrightarrow 2$, $2 \Longrightarrow 3$, and finally $3 \Longrightarrow 1$.

Proof Sketch $1 \Longrightarrow 2$. If C is empty, then the decomposition is $\mathcal{D} = \langle V \rangle$ for a $(c_0, \emptyset)$-colourful graph where all paths satisfy the Rabin condition. If C is not empty, we first remove all vertices A from $\mathcal{G}$ that can visit a vertex for which c_0 is a good colour. In the decomposition of the graph into Strongly Connected Components (SSCs) induced by $V \setminus A$, each infinite path satisfies the Rabin condition, and therefore especially the

infinite path which consists of all the vertices of some bottom SCC, V_1. Hence, there must be one colour c that is not a bad colour for any vertex and is a good colour for at least some of the vertices V_1. One can therefore inductively construct a $(c, C \setminus \{c\})$-colourful decomposition $\mathscr{D}_1$ for the vertices of V_1. Later, in the graph $\mathscr{G}$ without the vertices of A and V_1 and all vertices A_1 from which all paths lead to V_1, we again get an other graph where all infinite paths satisfy the Rabin condition. This graph, again by induction has a (c_0, C)-colourful Rabin decomposition $\mathscr{D}'$. We finally 'glue' together $\langle A, (c, V_1, \mathscr{D}_1, A_1)\rangle$ and $\mathscr{D}'$ obtained above.

Proof Sketch 2 $\Longrightarrow$ 3. The proof follows a recursive construction of an $\mathbb{L}$-labelled (c_0, C)-colourful tree where the recursion is based on the structure of the decomposition. An example of how such a mapping to a tree is obtained from a picture is exemplified in Fig. 3. The decomposition $\mathscr{D}$ of the game $\mathscr{G}$ is $(A, (\bullet, V_1, \mathscr{D}_1, A_1), (\bullet, V_2, \mathscr{D}_2, A_2))$. Some of the sets of the decomposition are indicated in Fig. 3a. The measure obtained from the decomposition into the given tree is intuitive. For example, the measure obtained from the given decomposition of the game $\mathscr{G}$ is such that the vertex for which the colour $\bullet$ is a good colour is mapped to the root of the tree. Similarly, this measure maps the vertex in V_1 for which the colour $\bullet$ is a good colour to the node $1\bullet$ in the tree. The only vertex in $A_2 \setminus V_2$ is mapped to the node $2\circ$.

Proof Sketch 3 $\Longrightarrow$ 1. If there is a Rabin measure, each edge in the infinite path satisfies B, as well as $\mathsf{G}_{\succ}$ or $\mathsf{G}_{\downarrow}$. For such an infinite path, we consider the infinite sequence of nodes of the colourful tree, obtained by taking the image of μ on the run. In this sequence obtained, consider the smallest node of the tree t that is visited infinitely often, and let $c = \mathsf{colour}(t)$. We show that t is a common ancestor for all elements of the sequence after a finite prefix. Since all edges satisfy $\mathsf{G}_{\succ}$ or $\mathsf{G}_{\downarrow}$, c is a colour such that $c \in G_v$ for some v visited infinitely often. As all edges satisfy B, we have $c \notin B_v$ for all vertices v in the run after some finite prefix.

Remark 1. A similar statement to the equivalence of item 1 and item 2 has been proved in the work of Klarlund and Kozen [20], however, a reader familiar with their work might have observed some differences in the definition of a measure as well as a colourful tree. Our definition of colourful trees is more restrictive than theirs. For instance, colourful trees in the work of Klarlund and Kozen have no restrictions about the colours along a path in a tree, i.e, in their definition, the trees can have the same colour along a path, and in fact only a partial colouring is required. However, an examination of their proof reveals that in the direction of the proof where they construct a Rabin measure, they inherently use a construction which produces a mapping into colourful trees as we have defined and therefore, it is enough to only consider such trees. We make this explicit and also prove Theorem 1 in the appendix to suit our situation.

4 A lifting algorithm

In this section, we define Rabin game formally and first show how such Rabin games also have a notion of a Rabin measure.

Inspired by the breakthrough algorithms to solve parity games, Colcombet, Fijalkow, Gawrychowski, and Ohlmann [5] proposed a formalism for algorithms that solve games where one player has a positional strategy. They showed that if there is a special kind of graph homomorphism into a graph with a total order on its vertices, then one can obtain a lifting algorithm for such games. From their work [5, Theorem 3.1] combined with Theorem 4, we can show that Rabin measures defined in our previous section can also be used to provide a lifting algorithm for Rabin games. However, to make this work self-contained and to provide an explicit space-efficient algorithm using our non-trivial totally ordered set, we show how such lifting is performed step-by-step in this section. We believe our following section would help future implementation of such algorithms.

A (c_0, C)-colourful *Rabin game* $\mathfrak{G}$ consists of an *arena* which is a (c_0, C)-colourful Rabin graph $\mathcal{G}$ with vertices V, a *start vertex* $v_0 \in V$, and a partition of V into V_c and V_e, the vertices of two players, whom we call Controller and Environment, respectively.

A *positional strategy* σ for Controller over the game graph is a subset of edges outgoing from each of Controller's set of vertices V_c. We denote the graph restricted to a strategy σ for the Controller by $\mathcal{G}|_\sigma$ and it is defined as the Rabin graph over the same vertex set with a new edge relation which contains exactly the edges in σ along with all the edges from all vertices belonging to Environment.

The Rabin game $\mathfrak{G}$ is *winning* for the Controller if and only if there exists a positional strategy σ for the Controller where, all infinite paths starting from v_0 in $\mathcal{G}|_\sigma$ satisfy the Rabin condition. We describe an algorithm that identifies whether a Rabin game $\mathfrak{G}$ is winning for Controller, using Rabin measures on graphs.

Remark 2. We only consider strategies of the Controller that are positional, but this is enough from the results of Emerson and Jutla [13], which shows that the Controller always has a positional winning strategy in Rabin games if there is any winning strategy at all.

Consistency in games. Consider a (c_0, C)-colourful Rabin game $\mathfrak{G}$. Let μ be a function from V, the vertices of the game graph to an $\mathbb{L}$-labelled (c_0, C)-colourful tree $\mathscr{L}$. We simply extend the definition of consistency from graphs to games by defining a vertex to be *consistent* with respect to μ in $\mathfrak{G}$ if either it belongs to the Environment and all outgoing edges from it are consistent in $\mathcal{G}$ or if it belongs to the Controller and there is at least one outgoing edge that is consistent in $\mathcal{G}$. A map μ from V to a $\mathscr{L}^\top$ is a *Rabin measure* for a (c_0, C)-colourful Rabin game $\mathfrak{G}$ if and only if *all* vertices are consistent with respect to μ.

An overview of the algorithm. We describe an algorithm that identifies whether a Rabin game $\mathfrak{G}$ is winning for Controller, using Rabin measures defined earlier for Rabin graphs. The basic principle in the algorithm is that given a colourful tree, the algorithm finds if there is a Rabin measure that maps vertices of the game into nodes of that tree. The algorithm does so by starting with the smallest map (all vertices are mapped to the root of this tree) and then at each step, if a vertex is not consistent, increase the value of this map just at this vertex which is not consistent. The value

is modified (increased) until either all vertices are consistent, or the value cannot be increased anymore.

Toward our goal of formally defining this algorithm, we define monotonic, inflationary operators on the set of all maps from vertices of a game to a tree such that the simultaneous fixpoints of these operators exactly correspond to a Rabin measure.

Consider a Rabin measure μ which is a function mapping the vertices V of a (c_0, C)-colourful Rabin game $\mathfrak{G}$ into an $\mathbb{L}$-labelled (c_0, C)-colourful tree $\mathscr{L}$. We define a function lift_μ, which maps edges E of the arena of the game to $\mathscr{L}^\top$. For an edge $u \to v$ of $\mathscr{G}$, we define $\text{lift}_\mu(u, v)$ to be the smallest element t in $\mathscr{L}^\top$ such that (1) $t \geq \mu(u)$ and (2) edge $u \to v$ is consistent with respect to the mapping $\mu[u := t]$, where we use the notation $\mu[u := t]$ to indicate the mapping μ' where $\mu'(x) = \mu(x)$ if $x \neq u$ and $\mu'(x) = t$ if $x = u$.

For each vertex v, we define an operator Lift_v on the lattice of all maps from V to $\mathscr{L}^\top$. The operator Lift_v only modifies an input map μ at v and nowhere else. We define

$$\text{Lift}_v(\mu)(u) = \begin{cases} \mu(u) & \text{for } u \neq v \\ \min_{(v,w)\in E} \{\text{lift}_\mu(v, w)\} & \text{if } u = v \in V_c \\ \max_{(v,w)\in E} \{\text{lift}_\mu(v, w)\} & \text{if } u = v \in V_e \end{cases}$$

Proposition 1. *The function* Lift_v *is monotonic for each* v.

The above proposition follows from our definition of the Lift_v function. Now that we know that each Lift_v is inflationary and monotonic. Therefore, the simultaneous least fixpoint of Lift_v on the map μ, which maps all vertices to the root of $\mathscr{L}$ exists (from the Knaster-Tarski theorem [31]). We can moreover state the following proposition that such fixpoints correspond to the Rabin measures, which almost follows from our definitions.

Proposition 2. *For a* (c_0, C)*-colourful Rabin game* $\mathfrak{G}$ *where the vertex set is* V *and a fixed* $\mathbb{L}$*-labelled* (c_0, C)*-colourful tree* $\mathscr{L}$*,*

- *any simultaneous fixpoint of the set of functions* Lift_v *for all* $v \in V$ *is a Rabin measure;*
- *any Rabin measure is a simultaneous fixpoint of* Lift_v *for all* $v \in V$.

Our algorithm, like any other progress-measure algorithm, computes a fixpoint and is described as follows. The correctness follows from Propositions 1 and 2.

Algorithm 1 The lifting algorithm on game (c_0, C)-colourful Rabin game $\mathfrak{G}$ with vertices V to tree $\mathscr{L}$

Require: For each $v \in V$, $\mu(v)$ is declared to be root in $\mathscr{L}$

1: **while** there is some vertex v that is inconsistent with respect to μ. **do**
2: $\quad \mu \leftarrow \text{Lift}_v(\mu)$.
3: **end while**
4: **return** μ

Remark 3. If there is a (c_0, C)-colourful Rabin game $\mathfrak{G}$ and an $\mathbb{L}$-labelled (c_0, C)-colourful tree $\mathscr{L}'$, such that there is a Rabin measure μ' from V to $\mathscr{L}'$, and $\mathscr{L}$ embeds $\mathscr{L}'$, then there is also a Rabin measure μ to $\mathscr{L}$ such that all the elements that are not mapped to $\top$ by μ' are still not mapped to $\top$ by μ. This map is obtained by composing μ' with the embedding of $\mathscr{L}'$ into $\mathscr{L}$.

Runtime. For a finer analysis of the runtime, we need to understand the size of the lattice where the lifting algorithm takes place. In this section however, we restrict ourselves to analysing the runtime of our algorithm for a fixed $\mathscr{L}$. We write $|\mathscr{L}|$ to represent the number of nodes in the labelled tree $\mathscr{L}$. We write n to denote the number of vertices in a Rabin game, m to denote the number of edges, and $k = |C \cup \{c_0\}|$ to denote the number of colours.

Lemma 1. *Given a mapping from the vertices of a (c_0, C)-colourful Rabin game $\mathfrak{G}$ to an $\mathbb{L}$-labelled (c_0, C)-colourful tree $\mathscr{L}$, the value of $\mathrm{Lift}_v(\mu)(v)$ can be computed in time $O\big(\deg(v) \cdot T_{\mathsf{next}}\big)$, where $\deg(v)$ is the degree (number of outgoing edges) of v and T_{next} is defined as the maximum of the time taken to*

- *make a linear pass on a node in $\mathscr{L}$ (assuming the node is represented by a sequence of elements of $\mathbb{L} \times C$),*
- *compute the next node in $\mathscr{L}$, and*
- *find the next node that uses colours only from $C' \cup \{\bot\}$ for a given node $t \in \mathscr{L}$ and subset of colours $C' \subseteq C$ such that $\mathsf{colour}(t) \in C'$.*

Proof (sketch). The proof of the above lemma reduces to arguing carefully that using these above items as subroutines, we can find the node larger than $\mathrm{Lift}_v(\mu)(v)$ in the given tree that satisfies the conditions B along with at least one of $\mathsf{G}_{>}$ or $\mathsf{G}_{\downarrow}$. To satisfy condition B, we need to find the next node that does not use a bad colour of v (using item 3 of the lemma) and then find its first child larger than the current node value of $\mu(v)$ that either satisfies $\mathsf{G}_{>}$ or $\mathsf{G}_{\downarrow}$ using the operations described above. The exact details of how the last step are done are provided in the full version of the paper.

Theorem 2. *For a (c_0, C)-colourful Rabin game $\mathfrak{G}$ with n vertices and m edges, and an $\mathbb{L}$-labelled (c_0, C)-colourful tree $\mathscr{L}$, Algorithm 1 on $(\mathfrak{G}, \mathscr{L})$ returns the smallest Rabin measure to $\mathscr{L}^{\top}$ in time $O(m|\mathscr{L}|T_{\mathsf{next}})$ where T_{next} is as defined in Lemma 1 and $|\mathscr{L}|$ denotes the number of nodes in $\mathscr{L}$.*

Proof. First, we observe that performing Lift_v on the mapping strictly increases the mapping for a vertex that is not consistent. Each operation of Lift_v also calls at most $\deg(v)$ many calls of $\mathrm{lift}_\mu(v, u)$ for some edge $v \to u$. Suppose each operation $\mathrm{lift}_\mu(v)$ takes time T_{next}, to find the value of $\mathrm{Lift}_v(\mu)(v)$ takes time at most $\deg(v) \cdot T_{\mathsf{next}}$. Since each non-trivial application of Lift_v strictly increases the value that v is mapped to, it can be called at most as many times as the number of nodes in tree $\mathscr{L}$, this ensures that the time taken is

$$\sum_{v \in V} \deg(v)|\mathscr{L}|\,(T_{\mathsf{next}}) \in O(m|\mathscr{L}|T_{\mathsf{next}})$$

where m denotes the number of edges.

5 Small Colourful Universal Trees

In the previous section, we concluded that our algorithm identifies correctly the smallest Rabin measure into a fixed labelled colourful tree $\mathscr{L}$. However, from Theorem 1, *there exists* a Rabin measure into an $\mathbb{L}$-labelled (c_0, C)-colourful tree $\mathscr{L}$ with at most n leaves. Observe that we only need to consider n leaves of $\mathscr{L}$ which correspond exactly to the image of the Rabin measure. Therefore, for a Rabin game, there is a Rabin measure into $\mathscr{L}^{\top}$ where all start vertices from which the game is winning for Controller are not mapped to $\top$. In order for the algorithm to successfully determine the winner of all (c_0, C)-colourful Rabin games with n vertices, we need to ensure that the tree $\mathscr{L}$ used in Algorithm 1 would be able to embed all (c_0, C)-colourful trees with n leaves. Since the runtime is linearly dependent on the tree size, smaller trees that satisfy the above property are desirable.

We now show that we can obtain colourful *universal* trees, i.e., colourful trees that are large enough to embed *any* (c_0, C)-colourful $\mathscr{L}$ with n-nodes. We also modify the technique of succinct universal trees of Jurdziński and Lazić [18] to encode each node of these colourful universal trees using polynomial space, which helps navigate these labelled colourful trees efficiently.

Colourful universal trees. A (c_0, C)-colourful tree $\mathscr{U}$ is *n-universal*, if it embeds *any* (c_0, C)-colourful tree $\mathscr{T}$ with at most n leaves. We henceforth assume that the set C consists exactly of the colours $c_1, \ldots, c_h$, with the fixed ordering $c_1 < c_2 < \cdots < c_h$ on the colours, and use k to denote $h+1$.

A näive attempt at constructing an n-universal (c_0, C)-colourful tree could be to take all possible (c_0, C)-colourful trees with at most n leaves with the root colour c_0 and concatenate them. Clearly, such an n-universal (c_0, C)-colourful tree can be created as there are only finitely many such trees up to isomorphism (for a fixed C and n). But of course, this tree is not only large, but can also be difficult to navigate. A more tractable attempt is to construct a tree that branches $n \cdot h$ many times at the root. The subtrees at the root that occur from this $n \cdot h$ branching have n repetitions of the h colours $c_1, c_2, \ldots, c_h$, in that order. Each of the children in-turn branch into $n \cdot (h-1)$ many times similarly, thus creating a tree of size bounded by $n^h h!$. We claim that indeed such a tree was exactly the one underlying the algorithm of Piterman and Pnueli [27], which led to their $O(mn^{k+1}kk!)$ algorithm.

Below, we give a more involved construction of a significantly smaller universal tree. In our construction, we inductively describe such a (c_0, C)-colourful n-universal tree, which we call $\mathscr{U}^{\ell}_{C}$, for a fixed $n \leq 2^{\ell}$.

- if $C = \emptyset$, then there is exactly one tree to embed, and therefore

$$\mathscr{U}^{\ell}_{(c_0,C)} = \left(c_0, \left\langle (\bot, \langle\rangle)^{2^{\ell}} \right\rangle\right)$$

- if $\ell = 0$, then the tree to be embedded has exactly one leaf and therefore, for each colour c_i in C, we have a child of colour c_i which hosts a subtree whose colour at the root is c_i. This is defined inductively as

$$\mathscr{U}^{0}_{(c_0,C)} = \left(c_0, \left\langle \mathscr{U}^{0}_{(c_1,C_1)}, \ldots, \mathscr{U}^{0}_{(c_h,C_h)}, (\bot, \langle\rangle) \right\rangle\right)$$

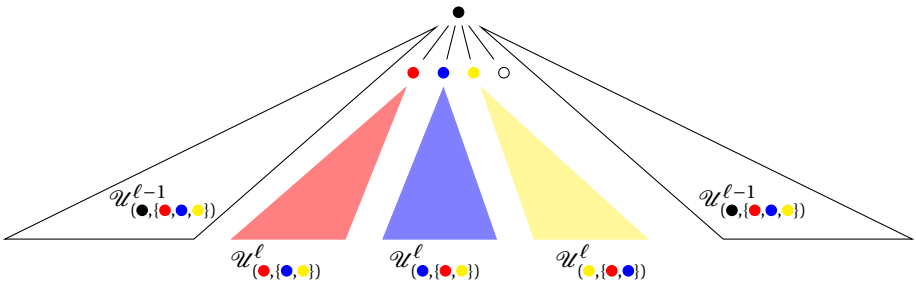


Fig. 4: Inductive construction of a smaller colourful n-universal tree

where C_i is $C \setminus \{c_i\}$.

– if $C \neq \emptyset$ and $\ell > 0$, then we define the coloured tree to be two copies of an $n/2$-universal tree, and h many copies of the n-universal tree where one colour is dropped each time. More formally,

$$\mathcal{U}^{\ell}_{(c_0,C)} = \mathcal{U}^{\ell-1}_{(c_0,C)} \cdot \left(c_0, \left\langle \mathcal{U}^{\ell}_{(c_1,C_1)}, \ldots, \mathcal{U}^{\ell}_{(c_h,C_h)}, (\bot, \langle\rangle) \right\rangle\right) \cdot \mathcal{U}^{\ell-1}_{(c_0,C)}.$$

In Fig. 4, we demonstrate how the inductive construction is done if $c_0 = \bullet$ and the set of colours is $C = \{\bullet, \bullet, \bullet\}$. To the left and right are the $(\bullet, C)$-colorful $n/2$-universal trees and between them, there are $|C|$ many n-universal trees each of which uses one fewer colour and one node with just the dummy colour represented there by $\circ$.

Theorem 3. *For $C \neq \emptyset$, and $k = |C| + 1$, $\mathcal{U}^{\ell}_{(c_0,C)}$ constructed is a (c_0, C)-colourful n-universal tree with at most*

$$nk! \left(\min \left\{ n2^k, \binom{\ell + k}{k-1} \right\} \right)$$

many leaves, where $\ell = \lceil \log n \rceil$.

Proof. Firstly, we need to show that $\mathcal{U}^{\ell}_{(c_0,C)}$ is (c_0, C)-colourful n-universal tree. Then, we prove using induction that $\mathcal{U}^{\ell}_{(c_0,C)}$ has at most $2^k \cdot k! \cdot 4^{\ell}$ leaves and later show that it also has at most $\binom{\ell+k}{k-1} \cdot 2^{\ell} \cdot k!$ leaves, leading to the proof of our theorem.

In fact, we have a lower bound for n-universal (c_0, C)-colourful trees, which is within a polynomial factor of the upper bound.

It is known from the work of Calude et al., as well as from Casares et al. [1,2] that there are no algorithms that solve Rabin games in time $n^{O(1)} \cdot 2^{o(k \log k)}$. But observe that this does not exclude algorithms which is dependant on $k!$ by only a constant smaller than 1 in the exponent. We have improved the current state-of-the art from $2 + o(1)$ to $1 + o(1)$ in the exponent. A natural question to ask would be if the $k!$ component can be reduced further. We show below that we cannot improve our running time much further using our techniques.

Lemma 2 (Lower bound). *Any n-universal (c_0, C)-colourful tree must have size at least $\binom{\ell+k-1}{\ell}(k-1)!$ where $k = |C|+1$ and $\ell = \lfloor \log n \rfloor$.*

Proof. Fix a permutation $c_{i_1}, \ldots, c_{i_h}$ of the colours in C and consider any tree with n leaves where the order of colours from the root to the leaf is exactly the same as the given permutation. Moreover, we assume that the leaves all have the same depth from the root. This tree must have size at least the size of a 2^ℓ-universal tree (defined for ordered tree without colours). Such universal trees have size at least $\binom{\ell+h-1}{h-1}$ in the work of Czerwiński et. al [6]. For each choice of permutation, the universal tree restricted to that permutation must have size $\binom{\ell+h-1}{h-1}$. Furthermore, two universal trees obtained by fixing different permutations cannot share a leaf since distinct colours are assigned to some ancestor of such leafs. Therefore, we obtain a lower bound of $\binom{\ell+h-1}{h-1}h!$ on the size of any (c_0, C)-colourful n-universal trees.

This immediately gives us the bound $\binom{\ell+k-2}{\ell}(k-1)!$ for $k = |C|+1$. Our lower bound also matches one of the upper bounds of our construction up to a polynomial factor in n and k.

Labelling Colourful Universal Trees. Here, we give a labelling of a universal colourful tree described in the previous section by giving an $\mathbb{W}$-labelling of any (c_0, C)-colourful tree where the set $\mathbb{W} = \{0,1\}^*$. We let ε denote the empty string in $\{0,1\}^*$. We define the ordering on $\{0,1\}^*$ as follows, similar to the succinct encoding of ordered trees [18]: $0 < \varepsilon < 1$ and for $b_1, b_2 \in \{0,1\}$ we have $b_1 \cdot w_1 < b_2 \cdot w_2$ if and only if $b_1 < b_2$ or $b_1 = b_2$ and $w_1 < w_2$.

Any node t in a $\mathbb{W}$-labelled (c_0, C)-colourful tree can be represented by a word generated by the following regular expression

$$\{0,1\}^* c_{i_1} \cdot \{0,1\}^* c_{i_2} \cdot \ldots \cdot \{0,1\}^* c_{i_m}$$

where $c_{i_j} \neq c_{i_k}$ if $j \neq k$ and $c_{i_j} = \perp$ if and only if $j = m$. We call the number of 0s and 1s occurring in the word, *the number of bits used to label t*. We show in the following lemma that it is possible to have a labelling of our colourful universal tree $\mathscr{U}^\ell_{(c_0,C)}$ such that the labelling of each node in it is 'short'.

Lemma 3. *There is a $\mathbb{W}$-labelling of the tree $\mathscr{U}^\ell_{(c_0,C)}$, denoted by $\mathscr{L}^\ell_C$ such that the number of bits used to label any node of $\mathscr{L}^\ell_C$ is at most ℓ.*

Proof (sketch). For $\ell > 0$, we have $\mathscr{U}^\ell_{(c_0,C)} = \mathscr{U}^{\ell-1}_{(c_0,C)} \cdot \left(c_0, \left\langle \mathscr{U}^\ell_{(c_1,C_1)}, \ldots, \mathscr{U}_{(c_h,C_h)} \right\rangle\right) \cdot \mathscr{U}^{\ell-1}_{(c_0,C)}$. We obtain recursively a labelling of $\mathscr{U}^{\ell-1}_{(c_0,C)}$ and append the bit 0 for the copy on the left and append with 1 for the copy on the right. For all the labellings of $\mathscr{U}^\ell_{(c_i,C_i)}$, we add the element $\varepsilon \cdot c_i$ as a prefix.

We rigorously prove this in the full version of the paper, but only state here that the three operations defined in the statement of Lemma 1 can be computed in time $O(k\ell \log k)$ (denoted by T_{next}), where $k = |C|+1$.

Theorem 4. *Finding the winner in a (c_0, C)-colourful Rabin game with n vertices, m edges, and $k = |C| + 1$, takes time*

$$\widetilde{O}\left(mnk^2 k! \min\left\{n2^k, \binom{\lceil \log n \rceil + k}{k-1}\right\}\right)$$

and $O(nk \log k \log n)$ space.

Proof. We know that the lifting Algorithm 1 for a (c_0, C)-colourful tree finds the Rabin measure into the tree $\mathscr{L}$ in time $O(m|\mathscr{L}|T_{\mathsf{next}})$ from Theorem 2. All that remains is to plug in the values of the size of the universal tree obtained from Theorem 3. For a game with n vertices, we instantiate the algorithm with $\mathscr{L}$ being the $\mathbb{W}$-labelling of the (c_0, C)-colourful 2^ℓ-universal tree $\mathscr{U}^\ell_{(c_0,C)}$ constructed, where $\ell = \lceil \log(n) \rceil$. The tree $\mathscr{L}$ therefore has at most $\left(nk! \min\left\{n2^k, \binom{\lceil \log n \rceil + k}{k-1}\right\}\right)$ many leaves from Theorem 3, and hence at most k times as many nodes, since each node has at most k ancestors. Moreover, the time taken to navigate the tree T_{next} is at most $O(k\ell \log k)$. The space required by the algorithm at each step is just the space required to store the map. To store a map, we need to store a node in the tree for each vertex. But from Lemma 3, storing each node only requires us to store a sequence of the k colours and at most $\log n$ bits. Since to store these k colours, we need $k \log k$ bits and, the total space complexity is $O(k \log k \log n)$ for each of the n vertices, giving us the desired space complexity.

6 Conclusions, Discussion and Future Work

We have shown an algorithm for Rabin games that requires almost quadratic space and takes time that is polynomial in n and $(k!)^{1+o(1)}$. Significantly more asymptotic improvement to the running time may be difficult, as it was shown in the work of Calude et al. [1,2] that there are no algorithms to solve Rabin games (as well as Muller games) in time $n^{O(1)} \cdot 2^{o(k \log k)}$ unless the Exponential Time Hypothesis fails (informally, it is the assumption that 3-SAT has no sub-exponential algorithms). However, improvements in the exponents of the parameter $k!$, which contributes to the majority of the running time would prove useful in any algorithm that solves Rabin games. We have shown that using colourful universal trees cannot provide a significant improvement bound because of the $k!^{1+o(1)}$ lowerbound on the size of such a tree. However, any technique that improves, even on a few targeted cases, this $1 + o(1)$ bound could lead to faster algorithms. For instance, the recent unpublished work of Liang, Khoussainov, and Xiao [24] improve the running time for specific values of k, where the size of k is large (comparable to n).

While we focus on the theoretical advance in this paper, an obvious future direction is to implement the algorithm. There are tools that convert LTL specifications to Rabin automata—such as Rabinizer 4 [22]. It will be interesting to see if solving the obtained Rabin games using our algorithms outperforms converting them instead to parity games and then using state-of-the-art parity game solvers such as Oink [10] framework. We believe improvement in state space of solving Rabin games through

our paper might lead to more efficient algorithms for the problem of reactive synthesis of LTL formulas.

Our algorithm, like other progress measure algorithms, can display worst-case behaviour in certain asymmetric examples. To show a vertex is losing for Controller, the measure needs to increase until it reaches $\top$. This lack of symmetric treatment of the players by our algorithm might lead to worst case behaviour on several examples. But circumventing this problem by constructing similar measures for Environment in the hopes of finding a symmetric algorithm is not as straightforward, as Environment does not have a positional strategy in this game.

In a different direction, symbolic algorithms for parity games are either implicitly or explicitly guided by universal trees [3,19] constructed for both players. We believe with some effort, our small colourful universal trees can be exploited to make symbolic algorithms to solve Rabin games. One such algorithm would look like an asymmetric variation of the universal algorithm in the work of Jurdziński, Morvan, and Thejaswini [19] for parity games, combined with our construction of colourful universal trees. Indeed, we already have a definition of colourful decompositions which one might hope to obtain as an end-result of such a recursive symbolic algorithm.

Acknowledgements. We would like to thank Marcin Jurdziński and Anne-Kathrin Schmuck for valuable discussions and references. We also thank Aditya Prakash for his valuable comments and, in particular, for reading the section on colourful trees despite his colour blindness.

References

1. Calude, C.S., Jain, S., Khoussainov, B., Li, W., Stephan, F.: Deciding parity games in quasi-polynomial time. SIAM Journal on Computing **51**(2), STOC17–152–STOC17–188 (2022). https://doi.org/10.1137/17M1145288
2. Casares, A., Pilipczuk, M., Pilipczuk, M., Souza, U., Thejaswini, K.S.: Simple and tight complexity lower bounds for solving Rabin games (2023), accepted at SOSA 24.
3. Chatterjee, K., Dvořák, W., Henzinger, M., Svozil, A.: Quasipolynomial set-based symbolic algorithms for parity games. In: LPAR-22. EPiC Series in Computing, vol. 57, pp. 233–253. EasyChair, Awassa, Ethiopia (2018). https://doi.org/10.29007/5z5k
4. Church, A.: Application of recursive arithmetic to the problem of circuit synthesis. Summaries of the Summer Institute of Symbolic Logic **1**, 3–50 (1957). https://doi.org/10.2307/2271310
5. Colcombet, T., Fijalkow, N., Gawrychowski, P., Ohlmann, P.: The theory of universal graphs for infinite duration games. Log. Methods Comput. Sci. **18**(3) (2022). https://doi.org/10.46298/lmcs-18(3:29)2022
6. Czerwiński, W., Daviaud, L., Fijalkow, N., Jurdziński, M., Lazić, R., Parys, P.: Universal trees grow inside separating automata: Quasi-polynomial lower bounds for parity games. In: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019. pp. 2333–2349. SIAM (2019). https://doi.org/10.1137/1.9781611975482.142
7. Daviaud, L., Jurdziński, M., Lehtinen, K.: Alternating weak automata from universal trees. In: 30th International Conference on Concurrency Theory, CONCUR 2019. Leibniz International Proceedings in Informatics (LIPIcs), vol. 140, pp. 18:1–18:14.

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Amsterdam, the Netherlands (2019). https://doi.org/10.4230/LIPIcs.CONCUR.2019.18
8. Daviaud, L., Jurdziński, M., Thejaswini, K.S.: The Strahler number of a parity game. In: A. Czumaj, A.D., Merelli, A. (eds.) 47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference). LIPIcs, vol. 168, pp. 123:1–123:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.ICALP.2020.123
9. Dell'Erba, D., Schewe, S.: Smaller progress measures and separating automata for parity games. Frontiers Comput. Sci. **4** (2022). https://doi.org/10.3389/fcomp.2022.936903
10. van Dijk, T.: Oink: An implementation and evaluation of modern parity game solvers. In: Tools and Algorithms for the Construction and Analysis of Systems, 24th International Conference, TACAS 2018. LNCS, vol. 10805, pp. 291–308. Springer, Thessaloniki, Greece (2018). https://doi.org/10.1007/978-3-319-89960-2_16
11. Emerson, E.A., Jutla, C.S.: The complexity of tree automata and logics of programs (extended abstract). In: 29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988. pp. 328–337. IEEE Computer Society (1988). https://doi.org/10.1109/SFCS.1988.21949
12. Emerson, E.A., Jutla, C.S.: Tree automata, mu-calculus and determinacy (extended abstract). In: 32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991. pp. 368–377. IEEE Computer Society (1991). https://doi.org/10.1109/SFCS.1991.185392
13. Emerson, E.A., Jutla, C.S.: The complexity of tree automata and logics of programs. SIAM Journal on Computing **29**(1), 132–158 (1999). https://doi.org/10.1137/S0097539793304741
14. Fearnley, J., Jain, S., de Keijzer, B., Schewe, S., Stephan, F., Wojtczak, D.: An ordered approach to solving parity games in quasi-polynomial time and quasi-linear space. International Journal on Software Tools for Technology Transfer **21**(3), 325–349 (2019). https://doi.org/10.1007/s10009-019-00509-3
15. Francez, N., Kozen, D.: Generalized fair termination. In: Proceedings of the 11th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages. p. 46–53. POPL '84, Association for Computing Machinery, New York, NY, USA (1984). https://doi.org/10.1145/800017.800515
16. Horn, F.: Streett games on finite graphs. In: Games in Design and Verification (2005)
17. Jurdziński, M.: Small progress measures for solving parity games. In: 17th Annual Symposium on Theoretical Aspects of Computer Science. LNCS, vol. 1770, pp. 290–301. Springer, Lille, France (2000). https://doi.org/10.1007/3-540-46541-3_24
18. Jurdziński, M., Lazić, R.: Succinct progress measures for solving parity games. In: 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017. pp. 1–9. IEEE Computer Society, Reykjavik, Iceland (2017). https://doi.org/10.1109/LICS.2017.8005092
19. Jurdziński, M., Morvan, R., Thejaswini, K.S.: Universal algorithms for parity games and nested fixpoints. In: Raskin, J.F., Chatterjee, K., Doyen, L., Majumdar, R. (eds.) Principles of Systems Design - Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday. Lecture Notes in Computer Science, vol. 13660, pp. 252–271. Springer (2022). https://doi.org/10.1007/978-3-031-22337-2_12
20. Klarlund, N., Kozen, D.: Rabin measures and their applications to fairness and automata theory. In: [1991] Proceedings Sixth Annual IEEE Symposium on Logic in Computer Science. pp. 256–265 (1991). https://doi.org/10.1109/LICS.1991.151650
21. Koh, Z.K., Loho, G.: Beyond value iteration for parity games: Strategy iteration with universal trees. In: S. Szeider, R.G., Silva, A. (eds.) 47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022, August 22-26, 2022, Vienna, Austria.

LIPIcs, vol. 241, pp. 63:1–63:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.MFCS.2022.63
22. Kretínský, J., Meggendorfer, T., Sickert, S., Ziegler, C.: Rabinizer 4: From LTL to your favourite deterministic automaton. In: Chockler, H., Weissenbacher, G. (eds.) Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10981, pp. 567–577. Springer (2018). https://doi.org/10.1007/978-3-319-96145-3_30
23. Kupferman, O., Vardi, M.Y.: Weak alternating automata and tree automata emptiness. In: Symposium on the Theory of Computing (1998). https://doi.org/10.1145/276698.276748
24. Liang, Z., Khoussainov, B., Xiao, M.: Two new algorithms for solving Muller games and their applications. CoRR **abs/2311.04655** (2023). https://doi.org/10.48550/ARXIV.2311.04655
25. Majumdar, R., Saglam, I., Thejaswini, K.S.: Rabin games and colourful universal trees. CoRR **abs/2311.04655** (2024). https://doi.org/10.48550/ARXIV.2401.07548
26. McNaughton, R.: Testing and generating infinite sequences by a finite automaton. Information and Control **9**(5), 521–530 (1966). https://doi.org/10.1016/S0019-9958(66)80013-X
27. Piterman, N., Pnueli, A.: Faster solutions of Rabin and Streett games. In: 21st Annual IEEE Symposium on Logic in Computer Science (LICS'06). pp. 275–284 (2006). https://doi.org/10.1109/LICS.2006.23
28. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. p. 179–190. POPL '89, Association for Computing Machinery, New York, NY, USA (1989). https://doi.org/10.1145/75277.75293
29. Rabin, M.O.: Decidability of second-order theories and automata on infinite trees. Transactions of the American Mathematical Society **141**, 1–35 (1969). https://doi.org/10.2307/1995086
30. Streett, R.S.: Propositional dynamic logic of looping and converse. In: Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing. p. 375–383. STOC '81, Association for Computing Machinery, New York, NY, USA (1981). https://doi.org/10.1145/800076.802492
31. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pacific Journal of Mathematics **5**(2), 285 – 309 (1955). https://doi.org/10.2140/pjm.1955.5.285