Information and Computation ••• (••••) •••–•••

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

The Rabin index of parity games: Its complexity and approximation

Michael Huth [a,*], Jim Huan-Pu Kuo [a], Nir Piterman [b]

[a] Department of Computing, Imperial College London, London, SW7 2AZ, United Kingdom
[b] Department of Computer Science, University of Leicester, Leicester, LE1 7RH, United Kingdom

ARTICLE INFO

Article history:
Received 28 February 2014
Available online xxxx

Keywords:
Logic in computer science
Graph theory: paths and cycles
Computational difficulty of problems (lower bounds, completeness, difficulty of approximation, etc.)
Descriptive complexity and finite models
Specification and verification (program logics, model checking, etc.)

ABSTRACT

We study the descriptive complexity of parity games by taking into account the coloring of their game graphs whilst ignoring their ownership structure. Colorings of game graphs are identified if they determine the same winning regions and strategies, for *all* ownership structures of nodes. The Rabin index of a parity game is the minimum of the maximal color taken over all equivalent coloring functions. We show that deciding whether the Rabin index is at least k is in P for $k = 1$ but NP-hard for all *fixed* $k \geq 2$. We present an EXPTIME algorithm that computes the Rabin index by simplifying its input coloring function. When replacing simple cycle with cycle detection in that algorithm, its output over-approximates the Rabin index in polynomial time. We evaluate this efficient algorithm as a preprocessor of solvers in detailed experiments: for Zielonka's solver on random and structured parity games and for our *partial* solver `psolB` on random games.

1. Introduction

Parity games (see e.g. [13,4,17,10]) are infinite-duration, two-person, zero-sum, graph-based games. A parity game consists of a directed graph in which nodes are colored with natural numbers and controlled by two different players. A play in a parity game may start in some node and consists of an infinite sequence of nodes such that the next node is chosen by the player who owns the current node, and where the next node must be a successor node of the current node in the directed graph. For example, in the parity game in Fig. 1, player 0 controls the circle nodes whereas player 1 controls the square nodes. A play may start in node v_3, where player 1 may chose to continue the play at node v_2, and then continue the play at node v_1. At node v_1, it is now player 0 who determines the continuation of the play. Player 0 may choose to move to node v_0, where player 1 is back in control. Player 1 may now move to node v_1 from which player 0 may (this time around) decide to move to node v_2. If these choices are repeated thus forever, the play $v_3(v_2v_1v_0v_1)^\omega$ gets generated as an infinite path in the directed graph.

A play in a parity game is won by determining the minimum color of all nodes that occur infinitely often in the play: the parity of that minimum decides which of the two players wins the play. In the above example, the colors that occur infinitely often in the play are 2 and 3 – colors of nodes are specified within nodes in Fig. 1 – and so their minimum 2 is even. Therefore, player 0 wins that play. We can now define what it means for a player to win a node in a parity game. A player wins a node iff that player has a way of playing that guarantees that all plays starting at that node are won by this

* Corresponding author.
E-mail addresses: m.huth@imperial.ac.uk (M. Huth), jimhkuo@imperial.ac.uk (J.H.-P. Kuo), nir.piterman@leicester.ac.uk (N. Piterman).

http://dx.doi.org/10.1016/j.ic.2015.06.005
0890-5401/

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005

player. To solve a parity game means to determine which nodes are won by which player, and to support such decisions with strategies for each player that realize these wins.

It is a rather deep result about parity games that each node is won by exactly one player [13,4,17]. Moreover, the proofs of this result also show that neither the use of "memory" that remembers more than the current node of the play, nor the ability to "mix" strategies through randomization increase the power of players to win particular nodes. To illustrate the use of memory, player 0 makes moves in the above play that are dependent on the history of the play: at node v_1 it strictly alternates its moves between v_0 and v_2. Players do not need such abilities to win. In fact, this strict alternation in the above example is not a winning strategy for player 0, since player 1 could move from node v_2 over to node v_4 and trap the play in node set $\{v_3, v_4\}$ – and so win the play as $\min(1, 2)$ is odd.

The condition for winning a node can be expressed as an alternation of existential and universal quantification. In fact, deciding the winner of a node is equivalent up to polynomial time to local model checking of a modal mu-calculus formula that captures this alternation as one of least and greatest fixed-points [5,14]. In practice, this means that the maximal color present in a parity game is the only exponential source for the worst-case complexity of most parity game solvers, e.g. for those in [17,12,15]. One approach taken in analyzing the complexity of parity games, and in so hopefully improving the complexity of their solution, is through the study of the *descriptive* complexity of their underlying game graph. This method therefore ignores the ownership structure on parity games. An example of this approach is the notion of DAG-width in [1]. Every directed graph has a DAG-width, a natural number that specifies how well that graph can be decomposed into a directed acyclic graph (DAG). The decision problem for DAG-width, whether the DAG-width of a directed graph is at most k, is NP-complete in k [1]. But parity games whose DAG-width is below a given threshold have polynomial-time solutions [1]. The latter is a non-trivial result since DAG-width also ignores the colors of a parity game.

In this paper we want to develop a similar measure of the descriptive complexity of parity games, their *Rabin index*, a natural number that ignores the ownership of nodes, but does take into account the colors of a parity game. Intuitively, the Rabin index is the number of colors that are *required* to capture the complexity of the game structure when the information about node ownership is forgotten. By measuring and reducing the number of colors we hope to improve the complexity of analyzing parity games. The reductions we propose are related to priority compression and propagation in [8] but, in contrast, exploit the *cyclic* structure of game graphs.

Our proposed setting of forgetting ownership structure of nodes may seem surprising. But we note that if we also were to account for ownership, we could solve the parity game and assign color 0 to nodes won by player 0 and color 1 to nodes won by player 1. Thus, this would reduce the index of *all* games to at most 2. But such a reduction then no longer conveys information about the descriptive complexity of the colored, directed graph of the solved parity game and so prevents such a study even at the coarsest level of analysis. Moreover, our studies of color reductions on parity games reported in this paper show that it is possible to build preprocessors for parity game solvers that reduce coloring complexity with methods that originate from the study of the Rabin index of parity games. And we then show that such preprocessors can yield parity games that are easier for both conventional (full) solvers [9] and for so-called partial solvers [11]. These methods rely on the detection of certain cycles through given nodes. It may be possible that these methods also work for a suitable form of alternating reachability, which may then lead to the design of new full solvers; we leave the study of this question as future work.

The name for the measure developed in this paper is inspired by related work on the Wagner hierarchy for automata on infinite words [16]: Carton and Maceiras use similar ideas to compute and minimize the Rabin index of deterministic parity automata on infinite words [2]. To the best of our knowledge, our work is the first to study this notion in the realm of infinite-duration, two-person games.

The idea behind our Rabin index is that one may change the coloring function of a parity game to another one if that change neither affects the winning regions nor the choices of winning strategies. This yields an equivalence relation between coloring functions on a given game graph. For the coloring function of a parity game, we then seek an equivalent coloring function with the smallest possible maximal color, and call that minimal maximum the Rabin index of the respective parity game.

The results we report here about this Rabin index are similar in spirit to those developed for DAG-width in [1] but there are important differences, which we now elaborate on. We propose a measure of descriptive complexity that is closer to the structure of the parity game as it only forgets ownership of nodes and not their colors (which DAG-width does forget). We prove that for every *fixed* $k \geq 2$, deciding whether the Rabin index of a parity game is at least k is NP-hard, whereas the decision problems for DAG-width are hard in the *parameter* k. We then characterize the aforementioned equivalence relation in terms of the parities of minimal colors on *simple* cycles in the game graph and use that characterization to design an algorithm that computes the Rabin index and a witnessing coloring function in exponential time. A closer inspection of this algorithm reveals that it *efficiently* computes sound approximations of the Rabin index when simple cycles are abstracted by cycles in its parts that check the existence of certain cycles. As a consequence of this approximation, we derive an abstract Rabin index of parity games such that games with bounded abstract Rabin index are efficiently solvable. Finally, we conduct detailed experimental studies that corroborate the utility of that approximation, both as a preprocessor for solvers and as a means of making so-called partial solvers [11] more precise.

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005

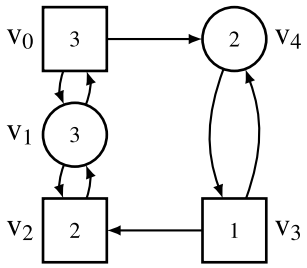


Fig. 1. A parity game with winning regions $W_0 = \{v_1, v_2\}$ and $W_1 = \{v_0, v_3, v_4\}$. Strategy $\sigma: V_0 \to V$ with $\sigma(v_1) = v_2$ and $\sigma(v_4) = v_3$ is winning on W_0. Strategy $\pi: V_1 \to V$ with $\pi(v_0) = \pi(v_3) = v_4$ and $\pi(v_2) = v_1$ is winning on W_1.

Outline of paper Section 2 contains background for our technical developments. In Section 3, we define the equivalence between coloring functions, characterize it in terms of simple cycles, and use that characterization to define the Rabin index of parity games. In Section 4 we develop an algorithm that runs in exponential time and computes a coloring function which witnesses the Rabin index of the input coloring function. The complexity of the natural decision problems for the Rabin index is studied in Section 5. An abstract version of our algorithm is shown to soundly approximate that coloring function and Rabin index in Section 6. Section 7 contains our experimental results for this abstraction. Related work is discussed in Section 8. And we conclude the paper in Section 9.

2. Background

We begin with providing minimal technical background and notation for parity games needed to appreciate the technical development of this paper. For a more detailed account of parity games and their solvers, we refer for example to [8,9]. We write $\mathbb{N}$ for the set $\{0, 1, \ldots\}$ of natural numbers. A parity game G is a tuple (V, V_0, V_1, E, c) where V is a non-empty set of nodes partitioned into possibly empty node sets V_0 and V_1, with an edge relation $E \subseteq V \times V$ (where for all v in V there is a w in V with (v, w) in E), and a coloring function $c: V \to \mathbb{N}$.

Throughout, we write s for one of 0 or 1 which (as determined by context) may denote the natural number or its corresponding player. In a parity game, player s owns the nodes in V_s. A play from some node v_0 results in an infinite play $P = v_0 v_1 \ldots$ in (V, E) where the player who owns v_i chooses the successor v_{i+1} such that (v_i, v_{i+1}) is in E. Let $\mathsf{Inf}(P)$ be the set of colors that occur in P infinitely often:

$$\mathsf{Inf}(P) = \{k \in \mathbb{N} \mid \forall j \in \mathbb{N}: \exists i \in \mathbb{N}: i > j \text{ and } k = c(v_i)\} \tag{1}$$

Player 0 wins play P iff $\min \mathsf{Inf}(P)$ is even; otherwise player 1 wins play P.

A *positional* strategy for player s is a total function $\tau: V_s \to V$ such that $(v, \tau(v))$ is in E for all $v \in V_s$. A play P is consistent with τ if each node v_i in P owned by player s satisfies $v_{i+1} = \tau(v_i)$. Subsequently, we simply write "strategies" when referring to positional strategies. A strategy τ is winning for player s from node v if all plays starting at v and consistent with τ are winning for s. Player s wins from v if she has a winning strategy from v. Her winning region is the set of nodes from which she wins. It is well known that each parity game is determined [13,4,17]: node set V is the disjoint union of two sets W_0 and W_1, the winning regions of players 0 and 1 (respectively), where one of W_0 and W_1 may be empty. Moreover, non-randomized and memoryless strategies $\sigma: V_0 \to V$ and $\pi: V_1 \to V$ can be computed such that

- all plays beginning in W_0 and consistent with σ are won by player 0; and
- all plays beginning in W_1 and consistent with π are won by player 1.

We then say that σ is a winning strategy on W_0, and π is a winning strategy on W_1, noting that winning strategies are generally not unique. In parity games, one player may win all nodes. In particular, player 1 may win all nodes in G even though player 0 owns all nodes in G, i.e. $W_1 = V = V_0$ is possible. Solving a parity game means computing such data (W_0, W_1, σ, π).

Example 1. We show a parity game and one of its possible solutions in Fig. 1. In this simple parity game, σ is the only strategy of player 0 that is winning on W_0. For example, a strategy σ' with $\sigma'(v_1) = v_0$ would allow player 1 to win node v_1 by playing consistently with strategy π.

3. Rabin index

We now formalize the definition of equivalence for coloring functions, and then use that notion in order to formally define the Rabin index of a parity game. Throughout, we assume that all nodes in directed graphs have at least one outgoing edge – an assumption we built into our definition of parity games above.

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005

We want to reduce the complexity of a coloring function c in a parity game $G = (V, V_0, V_1, E, c)$ by transforming c to some coloring function c'. Since we do not want the transformation to be based on a solution of the game G itself, we design the transformation to ignore ownership of nodes. That is, it needs to be sound for *every possible* ownership structure $V = V_0 \cup V_1$. Therefore, for *all* such partitions $V = V_0 \cup V_1$, the two parity games (V, V_0, V_1, E, c) and (V, V_0, V_1, E, c') that differ only in colors need to be equivalent in that they have the same winning regions and the same sets of winning strategies. We formalize this notion.

Definition 1. Let (V, E) be a directed graph and $c, c' : V \to \mathbb{N}$ two coloring functions.

1. A partition of V is some pair of sets V_0 and V_1 with $V_0 \cap V_1 = \{\}$ and $V_0 \cup V_1 = V$. In particular, V_0 or V_1 may be empty.
2. We say that c and c' are equivalent, written $c \equiv c'$, iff for all partitions V_0 and V_1 of V the resulting parity games (V, V_0, V_1, E, c) and (V, V_0, V_1, E, c') have the same winning regions and the same sets of winning strategies for both players.

Intuitively, changing coloring function c to c' with $c \equiv c'$ is sound: regardless of what the actual partition of V is, we know that this change will neither affect the winning regions nor the choice of their supporting winning strategies. A solver of the parity game for c' outputs winning regions and winning strategies, and this output is therefore then also sound for the parity game for coloring function c whenever $c \equiv c'$ is true.

But the definition of $\equiv$ is not immediately amenable to algorithmic simplification of c to some c'. This definition quantifies over exponentially many partitions, and for each such partition it insists that certain sets of nodes and of strategies be equal. It is thus desirable to have a more compact characterization of $\equiv$ as the basis for designing a static analysis. To that end, we require some concepts from graph theory first.

Definition 2. Let (V, E) be a directed graph with a coloring function $c : V \to \mathbb{N}$.

1. A path P in the directed graph (V, E) is a finite sequence $v_0, v_1, \ldots, v_n$ of nodes in V such that (v_i, v_{i+1}) is in E for every i in $\{0, 1, \ldots, n-1\}$.
2. A cycle C in the directed graph (V, E) is a path $v_0, v_1, \ldots, v_n$ with (v_n, v_0) in E.
3. A simple cycle C in the directed graph (V, E) is a cycle $v_0, v_1, \ldots, v_n$ such that for every $i \neq j$ in $\{0, 1, \ldots n\}$ we have $v_i \neq v_j$.
4. The c-color of a cycle $v_0, \ldots, v_n$ in (V, E) is $\min_{0 \leq i \leq n} c(v_i)$.

Simple cycles are paths that loop so that no node has more than one outgoing edge on that path. A cycle is defined similarly, except that it is allowed that v_i equals v_j for some $i \neq j$, so a node on that path may have more than one outgoing edge. The color of a cycle is the minimal color that occurs on it. We illustrate these concepts through a simple example.

Example 2. For example, for the parity game in Fig. 1, a simple cycle is v_0, v_4, v_3, v_2, v_1 and its color is 1, a cycle that is not simple is v_0, v_1, v_2, v_1 and its color is 2.

We can now characterize $\equiv$ in terms of colors of simple cycles. Crucially, we make use of the fact that parity games have pure (i.e. non-random), positional (i.e. memoryless) strategies [13,4,17].

Proposition 1. *Let (V, E) be a directed graph and $c, c' : V \to \mathbb{N}$ two coloring functions. Then $c \equiv c'$ iff for all simple cycles C in (V, E), the c-color of C has the same parity as the c'-color of C.*

Proof. Let us write $c \sim c'$ iff for all simple cycles C in (V, E), the c-color of C has the same parity as the c'-color of C. We have to show that $\sim$ equals $\equiv$.

1. We show that $\sim$ is contained in $\equiv$. Let $c \sim c'$ be given. We want to show $c \equiv c'$. So let the pair V_0 and V_1 be an arbitrary partition of V. Consider the two derived parity games

$$G_c = (V, V_0, V_1, E, c) \qquad G_{c'} = (V, V_0, V_1, E, c') \tag{2}$$

Let W_0 be the winning region of player 0 in the parity game G_c and σ a strategy for player 0 winning for player 0 on W_0 in G_c.

Now consider an arbitrary strategy π for player 1. Then π is such a strategy in both parity games G_c and $G_{c'}$. Let $v \in W_0$ and let P be the play in (V, E) that begins in v and is consistent with σ and π. Since P is consistent with deterministic strategies of both players, its ultimately periodic behavior determines a simple cycle C so that P is composed of a finite

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005

prefix and infinitely many repetitions of C. Since v is in W_0 and since σ is winning for player 0 in W_0, we infer that the c-color of C has to be even. Since $c \sim c'$, this means that the c'-color of C is even, too. And so that play is also won by player 0 in $G_{c'}$.

Since π was arbitrary, this shows that σ is also a winning strategy on W_0 in the parity game $G_{c'}$. Therefore, W_0 is a subset of the winning region W'_0 of player 0 in $G_{c'}$.

A symmetric argument for winning region W_1 in G_c for player 1 and winning strategy π for player 1 on W_1 in that game shows that π is also a winning strategy on W_1 in $G_{c'}$ and that W_1 is contained in W'_1, the winning region of player 1 in $G_{c'}$.

Combining these two insights, and since V equals $W_0 \cup W_1$, it follows that W_0 equals W'_0 and that W_1 equals W'_1. So the winning regions are equal in G_c and $G_{c'}$, and strategies that are winning on these sets in one of the games G_c and $G_{c'}$ are also winning on these sets in the other game since $c \sim c'$. (We showed this for one player, but the result follows for the other player by symmetry.)

2. We show that $\equiv$ is contained in $\sim$. Let $c \equiv c'$ be given. Let C be a simple cycle in (V, E). Let the parity of the c-color of C be even. (The case when this is odd is proved symmetrically and so we omit that proof.) Consider the parity games $(V, V, \emptyset, E, c)$ and $(V, V, \emptyset, E, c')$ where V_0 is defined to be V, and so V_1 is empty. Since V_0 equals V, player 0 has some strategy σ such that $\sigma(v)$ is again in C for all nodes v from C. Since the c-parity of C is even, it then follows that C is contained in W_0, the winning region of player 0 in (V, V_0, V_1, E, c).

Since $c \equiv c'$ is assumed, we therefore know that W_0 is also the winning region of player 0 in $(V, V, \emptyset, E, c')$, and that σ is also a winning strategy on W_0 in that game. In particular, every play beginning in some node v from C and consistent with σ is won by player 0 in $(V, V, \emptyset, E, c')$. But every such play just repeats the simple cycle C infinitely often (it cannot generate a sub-cycle of C as σ is deterministic and C is simple) and so the outcome of that play is determined by the c'-color of C. Therefore, the c'-color of C has to be even. □

Next, we define the relevant measure of descriptive complexity, which will also serve as a measure of precision for the static analyses we will develop later in the paper.

Definition 3. Let (V, E, c) be a colored arena and G a parity game whose colored arena is (V, E, c). Then we define the following expressions

$$\mu(c) = \max_{v \in V} c(v) \tag{3}$$

$$\mathsf{RI}(c) = \min\{\mu(c') \mid c \equiv c'\} \tag{4}$$

where $\mu(c)$ is the index of colored arena (V, E, c), and where $\mathsf{RI}(c)$ is the Rabin index of colored arena (V, E, c) as well as of the parity game G.

The index $\mu(c)$ reflects the maximal color occurring in c. So for a coloring function $c: V \to \mathbb{N}$ on a directed graph (V, E), its Rabin index is the minimal possible maximal color in a coloring function that is equivalent to c. This definition applies to colored arenas and parity games alike. One objection one might raise against $\mu(c)$ as a good complexity measure is that adding an even constant to all colors affects this measure: $\mu(c+n) = n + \mu(c)$ for $c+n$ with $(c+n)(v) = c(v) + n$ when n is even. Another objection might be that c can have large color gaps and so not reflect genuine alternation complexity. Fortunately, such objections do not apply to the Rabin index of c. This is so as for all $c' \equiv c$ with $\mu(c') = \mathsf{RI}(c)$ we know that the minimal color of c' is at most 1 and that c' has no color gaps – due to the minimality of the Rabin index.

A natural question is how we can compute this Rabin index. Thinking of this first as a decision problem, in order to prove that $\mathsf{RI}(c) < k$ for some $k > 0$ one has to produce a coloring c' with $\mu(c') < k$ and show that all simple cycles in the graph have the same color under c and c'. As we will see below, deciding for a given colored arena (V, E, c) the dual inequality, whether $\mathsf{RI}(c)$ is at least k is NP-hard for fixed $k \geq 2$. We first develop an algorithm that computes a coloring function which witnesses the Rabin index of a given c.

4. Computing the Rabin index

Our algorithm `rabin` for computing the Rabin index of a colored arena is shown in Fig. 2. It takes a coloring function as input and outputs an equivalent one whose index is the Rabin index of the input. Formally, `rabin` computes a coloring function c' with $c \equiv c'$ and where there is no $c \equiv c''$ with $\mu(c'') < \mu(c')$. These two properties therefore imply that $\mathsf{RI}(c)$ equals $\mu(c')$ by definition of the Rabin index in (4).

The central idea behind this algorithm is the concept of an anchor, which we now formally define.

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005

```
rabin(V, E, c) {
  rank = ∑_{v∈V} c(v);
  do {
    cache = rank;
    cycle(); pop();
    rank = ∑_{v∈V} c(v);
  } while (cache != rank)
  return c;
}

cycle() {
  sort V in ascending c-color ordering v_1, v_2, ..., v_n;
  for (i = 1..n) {
    j = getAnchor(v_i);
    if (j == −1) { c(v_i) = c(v_i)%2; }
    else { c(v_i) = j+1; }
  }
}

getAnchor(v_i) {
  for (γ = c(v_i) − 1  down to (c(v_i) − 1)%2;  step size 2) {
     if (∃ simple cycle C with color γ through v_i) { return γ; }
  }
  return −1;
}

pop() {
  m = max{ c(v) | v ∈ V};
  while (not ∃ simple cycle C with color m) {
    for (v in { w ∈ V | c(w) = m}) { c(v) = m − 1; }
    m = m − 1;
  }
}
```

Fig. 2. Algorithm `rabin` for computing the Rabin index $\mathsf{RI}(c)$ of a colored arena (V, E, c); the algorithm relies on methods `cycle`, `getAnchor`, and `pop`.

Definition 4. Let (V, E, c) be a colored arena with nodes $v_1, v_2, \ldots, v_n$. An anchor of node v_i is a color j of (V, E, c) such that

1. there is a simple cycle C through v_i whose color j is smaller and of different parity than that of v_i, and
2. for all simple cycles C' through v_i, either C' has a color of the same parity as the color of v_i or its color is less than or equal to j.

We note that an anchor j for v_i is maximal in that it is the largest color of (V, E, c) with the properties above. Our algorithm uses two insights about anchors. First, if v_i has no anchor, then it is safe to change the color of v_i to its parity. Second, if j is the anchor for v_i, then it is safe to change the color of v_i to $j + 1$. By "safe" we here mean that the new coloring function is equivalent to the old one with respect to the equivalence relation $\equiv$.

Algorithm `rabin` uses a standard iteration pattern based on a rank function which sums up all colors of all nodes. In each iteration, two methods are called:

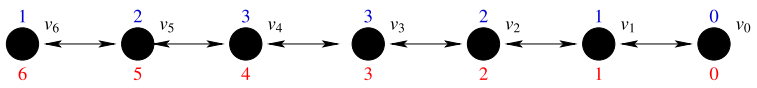
- `cycle` analyzes the cyclic structure of (V, E) to determine whether anchors exist, and then reduces the color of nodes based on such determinations.
- `pop` repeatedly lowers all occurrences of maximal colors by 1 until there is a simple cycle whose color is a maximal color.

These iterations proceed until neither `cycle` nor `pop` has an effect on the current state of the coloring function. Method `cycle` first sorts all nodes of (V, E, c) in ascending color values for c. It then processes each node v_i in that ascending order. For each node v_i it calls `getAnchor` to find (if possible) a maximal anchor for v_i. If `getAnchor` returns -1, then v_i has no anchor as all simple cycles through v_i have color of the same parity as $c(v_i)$. Therefore, it is sound to change $c(v_i)$ to its parity. Otherwise, `getAnchor` returns an index j as anchor for node v_i. In that case, method `cycle` therefore resets $c(v_i)$ to $j + 1$.

The idea behind `pop` is that one can safely lower maximal color m to $m - 1$ if there is no simple cycle whose color is m. For then all occurrences of m are dominated by smaller colors on simple cycles.

We show some example runs of `rabin`, starting with a detailed worked example.

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005



iteration	`cycle`	`pop`
1	*nil*	$c(v_6) = 5$
2	$c(v_6) = 1$	$c(v_5) = 4$
3	$c(v_5) = 2$	$c(v_4) = 3$

Fig. 3. Colored arena (V, E, c) and table showing effects of iterations in `rabin`(V, E, c).

Example 3.

1. Consider the parity game in Fig. 1. Let the initial sort of `cycle` be $v_3v_4v_2v_0v_1$. Then `cycle` changes no colors at v_3 (as the anchor of v_3 is -1), at v_4 (as the anchor of v_4 is 1 due to simple cycle v_4v_3), at v_2 (as the anchor of v_2 is 1 due to simple cycle $v_2v_1v_0v_4v_3$), but changes $c(v_0)$ to 1 (as the anchor of v_0 is -1). Also, $c(v_1)$ won't change (as the anchor of v_1 is 2 due to simple cycle v_1v_2).
   Then `pop` changes $c(v_1)$ to 2 (as there is no simple cycle with color 3). Let the sort of the second call to `cycle` be $v_0v_3v_1v_2v_4$. Then the corresponding list of anchor values is $-1, -1, 1, 1, 1$ and so `cycle` changes no colors. Therefore, the second call to `pop` changes no colors either. Thus the overall effect of `rabin` was to lower the index from 3 to 2 by lowering $c(v_1)$ to 2.
2. Consider the colored arena in Fig. 3, we see a colored arena with $c(v_i) = i$ (in red in the online version/at the bottom), the output `rabin`(V, E, c) (in blue in the online version/at the top), and a table showing how the coloring function changes through repeated calls to `cycle` and `pop`. Each iteration of `rabin` reduces the measure $\mu(c)$ by 1. This illustrates that the number of iterations of `rabin` is not bounded by a constant in general.
3. In Fig. 4(b), colored arena (V, E, c) has odd index n and Rabin index 2. Although there are cycles from all nodes with color n, e.g., to the node with color $n-1$, there are no *simple* such cycles. So all colors reduce to their parity.

We now prove the soundness of our algorithm `rabin`.

Lemma 1. *Let (V, E, c) be a given colored arena and let c' be the coloring function that is returned by the call* `rabin`(V, E, c). *Then $c \equiv c'$ holds.*

Proof. Let $c = c_0, c_1, \ldots$ be the sequence of coloring functions that reflect the state changes of c in the call `rabin`(V, E, c). By Proposition 1, it suffices to show that $c_n \sim c_{n+1}$ for all such n. So let c_n be given.

1. Consider first the case when c_{n+1} is obtained from c_n by an execution of the for-statement in `pop`. Then m is the maximal color of c_n but there is no simple cycle in (V, E) that has c_n-color m. In other words, color m will never decide the c_n-color of a simple cycle. It is therefore safe to decrease all occurrences of m to $m-1$, as this will change the color of no simple cycle in (V, E). Since this change defines c_{n+1}, we have $c_n \sim c_{n+1}$ as desired.

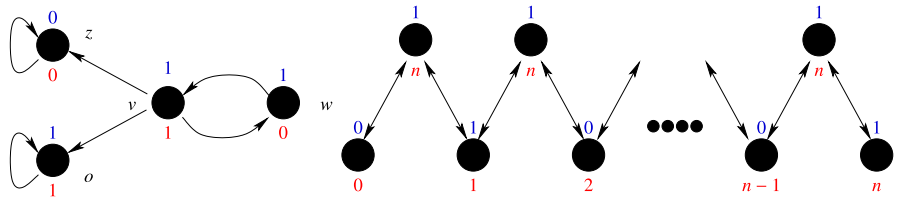
2. Now consider the case when c_{n+1} is the result of c_n through the execution of the if-branch in `cycle`. Then we consider a node v_i for which `getAnchor` returns -1. Therefore, there is no simple cycle C through v_i in (V, E) whose c_n-color is lower than $c_n(v_i)$ and has different parity than $c_n(v_i)$. But the color of cycles through v_i can be at most $c_n(v_i)$. Therefore, all simple cycles through v_i have the same parity as $c_n(v_i)$. It is therefore safe to reduce the color at v_i to that parity, as done in `cycle`. For the resulting c_{n+1} we therefore have $c_n \sim c_{n+1}$.

3. Now consider the case when c_{n+1} is the result of c_n through the execution of the else-branch in `cycle`. If the call to `getAnchor` returns $j \geq 0$ for node v_i, then consider an arbitrary simple cycle C in (V, E) through v_i whose color p has a parity other than that of $c_n(v_i)$. Then it must be that $p \leq j$ by the definition of method `getAnchor`. So every simple cycle through v_i has either a color that has the parity of $c_n(v_i)$ or has a color p with $p \leq j$. Therefore, it is safe to change the color at v_i to $j+1$ (the case $j+1 = c_n(v_i)$ will have no effect), resulting in new coloring function c_{n+1}: this is so since then all simple cycles through v_i have the same parity with respect to c_n and c_{n+1}. (And both coloring functions could only break $c_n \sim c_{n+1}$ by means of simple cycles through v_i.) □

We note that $\equiv$ cannot be captured by just insisting that the winning regions of all abstracted parity games be the same. Let us write $c \approxeq c'$ when the coloring functions c and c' always give rise to the same winning regions (but not necessarily to the same set of winning strategies). Clearly, $c \equiv c'$ implies $c \approxeq c'$ but the converse is not true: Fig. 4(a) shows a colored arena with two coloring functions c (in red in the online version of this paper/at the bottom) and c' (in blue in the online version of this paper/at the top). The player who owns node v will win all nodes as she chooses between z or o the node that has her parity. So $c \approxeq c'$ follows. But if v is owned by player 1, she has a winning strategy for c' (move from v to w) that is not winning for c, and so $c \not\equiv c'$ follows.

The example in Fig. 4(a) also suggests that computing $\approxeq$ may be difficult. Cycle and simple cycle detection won't capture $\approxeq$ precisely. Topological reasoning about strongly connected components seems hard as well: if we replace nodes o and z with entire strongly connected components C_0 and C_z so that o is in C_o and z in C_z, and if we keep the outgoing edges

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005



(a) Coloring functions c and c' give rise to the same winning regions, but not the same winning strategies. Thus $c \not\equiv c'$

(b) Coloring function c has Rabin index 2, witnessed by c'

Fig. 4. Two colored arenas that each have two coloring functions c (in red in the online version/at the bottom) and c' (in blue in the online version/at the top).

(v, o) and (v, z) of node v, then we cannot reason about node v in the manner we did above since we won't know whether nodes o and z will inevitably have different winners. Therefore, we focus our attention on $\equiv$ in this paper.

Now we can prove that algorithm `rabin` is basically as precise as it could be. First, we state and prove an auxiliary lemma which provides sufficient conditions for a coloring function c to have its index $\mu(c)$ as its Rabin index $\mathsf{RI}(c)$. Then we show that the output of `rabin` meets these conditions.

Lemma 2. *Let (V, E, c) be a colored arena where*

1. *there is a simple cycle in (V, E) whose color is the maximal one of c.*
2. *for all v in V with $c(v) > 1$, node v is on a simple cycle C with color $c(v) - 1$.*

Then there is no c' with $c \equiv c'$ and $\mu(c') < \mu(c)$. And so $\mu(c)$ equals $\mathsf{RI}(c)$.

Proof. Let k be the maximal color of c and consider an arbitrary c' with $c \equiv c'$.

Proof by contradiction: Let the maximal color k' of c' satisfy $k' < k$. By the first assumption, there is a simple cycle C_0 whose c-color is k. Since $k' < k$ and $c \equiv c'$, we know that the c'-color of C_0 can be at most $k-2$. Let v_0 be a node on C_0 such that $c'(v_0)$ is the c'-color of C_0. Then $c'(v_0) \leq k-2$. As all nodes on C_0 have c-color k, we have also $c(v_0) \geq k$. For $k < 2$, then $c'(v_0) \leq k-2$ gives us a contradiction $c'(v_0) < 0$. It thus remains to consider the case when $k \geq 2$.

By the second assumption, there is some simple cycle C_1 through v_0 such that the color of C_1 is $k-1$. In particular, there is some node v_0' in C_1 with color $k-1$. But $k-1$ cannot be the color of C_1 with respect to c' since v_0 is on C_1 and $c'(v_0) \leq k-2$. Since $c \equiv c'$, the c'-color of C_1 is therefore at most $k-3$. So there is some v_1 on C_1 such that $c'(v_1) \leq k-3 < k-1 \leq c(v_1)$.

If $c(v_1) > 1$, we repeat the above argument at node v_1 to construct a simple cycle C_2 through v_1 with color $c(v_1) - 1$. Again, there then have to be nodes v_1' and v_2 on C_2 such that the color $c'(v_1')$ is the c'-color of C_2, and such that $c'(v_2) \leq k-4 < k-2 \leq c(v_2)$ holds.

We can repeat the above argument to construct simple cycles $C_0, C_1, C_2, \ldots$ and a sequence of nodes $v_0, v_0', v_1, v_1', v_2, v_2', \ldots$ such that $c'(v_j) \leq k-j-2 < k-j \leq c(v_j)$ until j equals $k-1$. But for that value of j we obtain $c'(v_j) \leq k-j-2 \leq 1-2 = -1$, a contradiction. □

We now show that the output of `rabin` satisfies the assumptions of Lemma 2. Since `rabin` is sound for $\equiv$, we therefore infer that it computes a coloring function whose maximal color equals the Rabin index of its input coloring function.

Theorem 1. *Let (V, E, c) be a colored arena. And let c^* be the output of the call `rabin`(V, E, c). Then $c \equiv c^*$ and $\mu(c^*)$ is the Rabin index of c.*

Proof. By Lemma 1, we have $c \equiv c^*$. Since $\equiv$ is clearly transitive, it suffices to show that there is no c' with $c^* \equiv c'$ and $\mu(c') < \mu(c^*)$. By Lemma 2, it therefore suffices to establish the two assumptions of that lemma for c^*. As c^* is returned by `rabin` neither `cycle` nor `pop` have an effect on it.

The first assumption of Lemma 2 is therefore true since `pop` has no effect on c^* and so there must be a simple cycle in (V, E) whose color is the maximal one in c. This also applies to the case when c^* has only one color, as (V, E) has to contain cycles since it is finite and all nodes have outgoing edges.

As for the second assumption, let by way of contradiction there be some node v with $c^*(v) > 1$ and no simple cycle through v with color $c^*(v) - 1$. Then `cycle` would have an effect on $c^*(v)$ and would lower it, a contradiction. □

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005

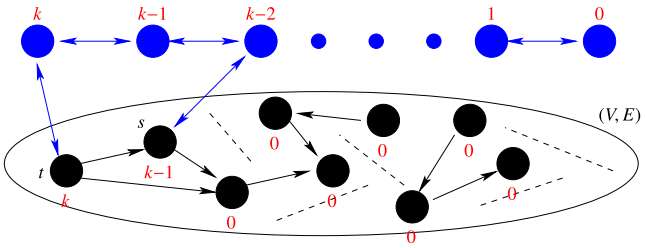


Fig. 5. Construction for NP-hardness of deciding whether $\mathsf{RI}(c) \geq k$ for $k \geq 2$.

To illustrate the need of method `pop` in this proof, consider a colored arena (V, E, c) such that all its simple cycles have even c-color but where some node has odd c-color. Then $c \equiv \mathbf{0}$ for the coloring function $\mathbf{0}$ that maps all nodes to color 0. Method `cycle` preserves the parity of nodes as invariant. And so c could never be reduced to $\mathbf{0}$ without the help of `pop`.

5. Complexity

We next analyze the computational complexity of algorithm `rabin` and study the computational complexity of the decision problems associated with the Rabin index. We turn to the complexity of `rabin` itself first.

Let us assume that we have an oracle that checks for the existence of simple cycles. Then the computation of `rabin` is efficient modulo polynomially many calls (in the size of the game) to that oracle. Since deciding whether a simple cycle exists between two nodes in a directed graph is NP-complete (see e.g. [6,7]), we infer that `rabin` can be implemented to run in exponential time. This decision problem is therefore in $\Delta_2^{\mathrm{P}} = \mathrm{P}^{\mathrm{NP}}$.

Next, we study the complexity of deciding the value of the Rabin index. We can exploit the NP-hardness of simple cycle detection to show that the natural decision problem for the Rabin index, whether $\mathsf{RI}(c)$ is at least k, is NP-hard for fixed $k \geq 2$. In contrast, for $k = 1$, we show that this problem is in P.

Theorem 2. *Deciding whether the Rabin index of a colored arena (V, E, c) is at least k is NP-hard for every fixed $k \geq 2$, and is in P for $k = 1$.*

Proof. First consider the case when $k \geq 2$. We use the fact that deciding whether there is a simple cycle through nodes $s \neq t$ in a directed graph (V, E) is NP-complete (see e.g. [7]). Without loss of generality, for all v in V there is some w in V with (v, w) in E (we can add (v, v) to E otherwise). Our hardness reduction uses a colored arena (V', E', c), depicted in Fig. 5, which we now describe:

We color s with $k-1$ and t with k, and color all remaining nodes of V with 0. Then we add $k+1$ many new nodes (shown in blue in the online version/at the top in the figure) to that graph that form a "spine" of descending colors from k down to 0, connected by simple cycles. Crucially, we also add a simple cycle between t and that new k node, and between s and the new $k-2$ node.

We claim that the Rabin index of (V', E', c) is at least k iff there is a simple cycle through s and t in the original directed graph (V, E).

1. Let there be a simple cycle through s and t in (V, E). Since there is a simple cycle between s and the new $k-2$ node, `cycle` does not change the color at s. As there is a simple cycle through s and t, method `cycle` also does not change the color at t. Clearly, no colors on the spine can be changed by `cycle`. Since there is a simple cycle between t and the new k node, method `pop` also does not change colors. But then the Rabin index of c is k and so at least k.

2. Conversely, assume that there is no simple cycle through s and t in the original graph (V, E). It follows that the anchor j of t has value 0 or, if k is even, has value -1. This is so since the only simple cycles through s and t have color either 0 or $k-2$. In this case, `cycle` changes the color at t to the parity of k. Then, `pop` reduces the color of the remaining node colored k to $k-1$. Thus, it cannot be the case that the Rabin index of c is at least k.

This therefore proves the claim. Second, consider the case when $k = 1$. Deciding whether $\mathsf{RI}(c)$ is at least 1 amounts to checking whether $c \equiv \vec{0}$ where $\vec{0}(v) = 0$ for all v in V. This is the case iff all simple cycles in (V, E, c) have even c-parity. But that is the case iff all cycles in (V, E, c) have even c-parity.

To see this, note that the "if" part is true as simple cycles are cycles. As for the "only if" part, this is true since if there were a cycle C with odd c-parity, then some node v on that cycle would have to have that minimal c-color, but v would then be on some simple cycle whose edges all belong to C.

Finally, checking whether all cycles in (V, E, c) have even c-parity is in P. □

The decision problem of whether $\mathsf{RI}(c) = 1$ cannot be in NP, unless NP equals coNP. Otherwise, the decision problem of whether $\mathsf{RI}(c) \leq 1$ would also be in NP, since we can decide in P whether $\mathsf{RI}(c) = 0$ and since NP is closed under unions. But then the complement decision problem of whether $\mathsf{RI}(c) \geq 2$ would be in coNP, and we have shown it to be NP-hard already. Therefore, all problems in NP would reduce to this problem and so be in coNP as well, a contradiction.

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005

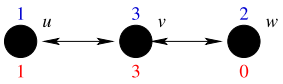


Fig. 6. Coloring functions c (blue in the online version/at the top) and c' (red in the online version/at the bottom) with $c \equiv c'$ but $c \not\equiv^{\alpha} c'$.

We now discuss an efficient version of `rabin` which over-approximates oracle calls for simple cycle detection with calls for cycle detection.

6. Abstract Rabin index

In the last section, we saw that `rabin` runs in exponential time in the worst case. Therefore, we want to trade off this complexity with the precision of color reductions in the colored arena. The idea we propose is to replace oracle calls for simple cycle detection within `rabin` with over-approximating cycle detections. This modified version of `rabin` then computes an abstract Rabin index, whose definition is based on an abstract version of the equivalence relation $\equiv$. We define these notions formally.

Definition 5.

1. Let $\texttt{rabin}^{\alpha}$ be `rabin` where all existential quantifications over simple cycles are replaced with existential quantifications over cycles.
2. Let (V, E) be a directed graph and $c, c' \colon V \to \mathbb{N}$ two coloring functions. Then $c \equiv^{\alpha} c'$ iff for all cycles C, the parities of their c- and c'-colors are equal.
3. The abstract Rabin index $\mathsf{RI}^{\alpha}(c)$ of a colored arena (V, E, c) is defined as

$$\mathsf{RI}^{\alpha}(c) = \min\{\mu(c') \mid c \equiv^{\alpha} c'\} \tag{5}$$

The abstract Rabin index $\mathsf{RI}^{\alpha}(c)$ and our algorithm $\texttt{rabin}^{\alpha}$ are very similar to a Rabin index for deterministic parity word automata and its computation in [2]; we discuss these similarities in Section 8.

Algorithm $\texttt{rabin}^{\alpha}$ uses the set of cycles in (V, E) to overapproximate the set of simple cycles in (V, E). We point out that $c \equiv^{\alpha} c'$ implies $c \equiv c'$ but not the other way around, as can be seen in the example in Fig. 6. In that example, we have $c \equiv c'$ since all simple cycles have the same parity of color with respect to c and c'. But there is a cycle that reaches all three nodes and which has odd color for c and even color for c'. Thus, $c \not\equiv^{\alpha} c'$ follows.

We now show that the overapproximation $\texttt{rabin}^{\alpha}$ of `rabin` is sound in that its output coloring function is equivalent to its input coloring function. Below, in Theorem 3, we further show that this output yields an abstract Rabin index.

Lemma 3. *Let (V, E, c) be a colored arena and let $\texttt{rabin}^{\alpha}(V, E, c)$ return c'. Then $c \equiv^{\alpha} c'$ and $\mu(c') \geq \mathsf{RI}(c)$.*

Proof. We first show that $c \equiv^{\alpha} c'$ holds. Let $c = c_0, c_1, \ldots$ be the sequence of coloring functions that reflect the state changes of c in the call $\texttt{rabin}^{\alpha}(V, E, c)$. Since $\equiv^{\alpha}$ is transitive, it suffices to show that $c_n \equiv^{\alpha} c_{n+1}$ for all such n. So let c_n be given.

1. Consider first the case when c_{n+1} is obtained from c_n by an execution of the for-statement in `pop`. Then m is the maximal color of c_n but there is no cycle in (V, E) that has c_n-color m. In other words, color m will never decide the c_n-color of a cycle. It is therefore safe to decrease all occurrences of m to $m-1$, as this will change the color of no cycle in (V, E). Since this change defines c_{n+1}, we have $c_n \equiv^{\alpha} c_{n+1}$ as desired.

2. Now consider the case when c_{n+1} is the result of c_n through the execution of the if-branch in `cycle`. Then we consider a node v_i for which `getAnchor` returns -1. Therefore, there is no cycle C through v_i in (V, E) whose c_n-color is lower than $c_n(v_i)$ and has different parity than $c_n(v_i)$. But the color of cycles through v_i can be at most $c_n(v_i)$. Therefore, all cycles through v_i have the same parity as $c_n(v_i)$. It is therefore safe to reduce the color at v_i to that parity, as done in `cycle`. For the resulting c_{n+1} we therefore have $c_n \equiv^{\alpha} c_{n+1}$.

3. Now consider the case when c_{n+1} is the result of c_n through the execution of the else-branch in `cycle`. If the call to `getAnchor` returns $j \geq 0$ for node v_i, then consider an arbitrary cycle C in (V, E) through v_i whose color p has a parity other than that of $c_n(v_i)$. Then it must be that $p \leq j$ by the definition of method `getAnchor`. So every cycle through v_i has either a color that has the parity of $c_n(v_i)$ or has a color p with $p \leq j$. Therefore, it is safe to change the color at v_i to $j+1$ (the case $j+1 = c_n(v_i)$ will have no effect), resulting in new coloring function c_{n+1}: this is so since then all cycles through v_i have the same parity with respect to c_n and c_{n+1}. (And both coloring functions could only break $c_n \equiv^{\alpha} c_{n+1}$ by means of cycles through v_i.)

Second, since we have shown that $c \equiv^{\alpha} c'$ holds, we also get that $c \equiv c'$ is true. But the latter in turn implies $\mu(c') \geq \mathsf{RI}(c)$ by the definition of the Rabin index. □

We can now adapt the results for `rabin` to this abstract setting.

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005

Lemma 4. *Let (V, E, c) be a colored arena where*

1. *there is a cycle in (V, E) whose color is the maximal one of c, and.*
2. *for all v in V with $c(v) > 1$, node v is on a cycle C with color $c(v) - 1$.*

Then there is no c' with $c \equiv^{\alpha} c'$ and $\mu(c') < \mu(c)$, and so $\mu(c) = \mathsf{RI}^{\alpha}(c)$.

Proof. Let k be the maximal color of c and consider an arbitrary c' with $c \equiv^{\alpha} c'$.

Proof by contradiction: Let the maximal color k' of c' satisfies $k' < k$. By the first assumption, there is a cycle C_0 whose c-color is k. Since $k' < k$ and $c \equiv^{\alpha} c'$, we know that the c'-color of C_0 can be at most $k-2$. Let v_0 be a node on C_0 such that $c'(v_0)$ is the c'-color of C_0. Then $c'(v_0) \leq k-2$. As all nodes on C_0 have c-color k, we have also $c(v_0) \geq k$. Again, if $k < 2$ we have a contradiction right away. So let $k \geq 2$.

By the second assumption, there is some cycle C_1 through v_0 such that the color of C_1 is $k-1$. In particular, there is some node v_0' in C_1 with color $k-1$. But $k-1$ cannot be the color of C_1 with respect to c' since v_0 is on C_1 and $c'(v_0) \leq k-2$. Since $c \equiv^{\alpha} c'$, the c'-color of C_1 is therefore at most $k-3$. So there is some v_1 on C_1 such that $c'(v_1) \leq k-3 < k-1 \leq c(v_1)$.

If $c(v_1) > 1$, we repeat this argument at node v_1 to construct a cycle C_2 through v_1 with color $c(v_1) - 1$. Again, there then have to be nodes v_1' and v_2 on C_2 such that the color $c'(v_1')$ is the c'-color of C_2, and such that $c'(v_2) \leq k-4 < k-2 \leq c(v_2)$ holds.

In this manner, we can repeat this argument to construct cycles $C_0, C_1, C_2, \ldots$ and nodes $v_0, v_0', v_1, v_1', v_2, v_2', \ldots$ such that $c'(v_j) \leq k-j-2 < k-j \leq c(v_j)$ until j equals $k-1$. But for that value of j, we get $c'(v_j) \leq k-j-2 \leq 1-2 = -1$, a contradiction. □

Similarly to the case for algorithm `rabin`, we now show that the output of $\texttt{rabin}^{\alpha}$ satisfies the assumptions of Lemma 4. Since algorithm $\texttt{rabin}^{\alpha}$ is sound for the equivalence relation $\equiv^{\alpha}$, we infer that it computes coloring functions whose maximal color equals the abstract Rabin index of their input coloring function.

Theorem 3. *Let (V, E, c) be a colored arena. And let c^* be the output of the call $\texttt{rabin}^{\alpha}(V, E, c)$. Then $c \equiv^{\alpha} c^*$ and $\mu(c^*)$ is the abstract Rabin index $\mathsf{RI}^{\alpha}(c)$.*

Proof. By Lemma 3, we have $c \equiv^{\alpha} c^*$. Since $\equiv^{\alpha}$ is transitive, it suffices to show that there is no c' with $c^* \equiv^{\alpha} c'$ and $\mu(c') < \mu(c^*)$. By Lemma 4, it therefore suffices to establish the two assumptions of that lemma for c^*. What we do know is that neither `cycle` nor `pop` have an effect on c^* as it was returned by $\texttt{rabin}^{\alpha}$.

The first assumption is therefore true since `pop` has no effect on c^* and so there must be a cycle in (V, E) whose color is the maximal one in c. (This also applies to the boundary case when c^* has only one color, as (V, E) has to contain cycles since it is finite and all nodes have outgoing edges.)

As for the second assumption, let by way of contradiction be some node v with $c^*(v) > 1$ and no cycle through v with color $c^*(v) - 1$. Then `cycle` would have an effect on $c^*(v)$ and would lower it, a contradiction. □

We now study the sets of parity games whose abstract Rabin index is below a fixed bound. It turns out that these parity games can be solved efficiently. Moreover, it is efficient to decide whether a parity game is below such a fixed bound. We first define these sets formally.

Definition 6. Let $\mathcal{P}_k^{\alpha}$ be the set of parity games (V, V_0, V_1, E, c) with $\mathsf{RI}^{\alpha}(c) < k$.

For this abstract Rabin index we can indeed prove that membership in $\mathcal{P}_k^{\alpha}$ is efficiently decidable and that games in that set are efficiently solvable. We note that deciding whether the (exact) Rabin index is below a fixed bound is the complement of an NP-hard problem by Theorem 2 and so is unlikely to have an efficient solution.

Theorem 4. *Let $k \geq 1$ be fixed. All parity games in $\mathcal{P}_k^{\alpha}$ can be solved in polynomial time. Moreover, membership in $\mathcal{P}_k^{\alpha}$ can be decided in polynomial time.*

Proof. For each parity game (V, V_0, V_1, E, c) in $\mathcal{P}_k^{\alpha}$, we first run $\texttt{rabin}^{\alpha}$ on it, which runs in polynomial time. By definition of $\mathcal{P}_k^{\alpha}$, the output coloring function c^* has index $< k$. Then we solve the parity game (V, V_0, V_1, E, c^*), which we can do in polynomial time as the index is bounded by k. But that solution is also one for (V, V_0, V_1, E, c) since $c \equiv^{\alpha} c^*$ by Lemma 3, and so $c \equiv c^*$ as well.

That the membership test is polynomial in the running time can be seen as follows: for coloring function c, compute $c' = \texttt{rabin}^{\alpha}(V, E, c)$ and return `true` if $\mu(c') < k$ and return `false` otherwise; this is correct by Theorem 3. □

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005

We note that algorithm $\texttt{rabin}^{\alpha}$ is precise for colored arenas $A = (V, E, c)$ with Rabin index 0. These are colored arenas that have only simple cycles with even color. Since a colored arena has a cycle with odd color iff it has a simple cycle with odd color, $\texttt{rabin}^{\alpha}$ correctly reduces all colors to 0 for such arenas and so computes the exact Rabin index in these instances.

For Rabin index 1, the situation is more subtle. We cannot expect algorithm $\texttt{rabin}^{\alpha}$ to always be precise, as the decision problem for $\mathsf{RI}(c) \geq 2$ is NP-hard. Algorithm $\texttt{rabin}^{\alpha}$ will correctly compute Rabin index 1 for all those arenas that do not have a simple cycle with even color. But for c from Fig. 6, e.g., algorithm $\texttt{rabin}^{\alpha}$ does not change c with index 3, although the Rabin index of c is 1.

We observe that $\texttt{rabin}^{\alpha}$ is not just an overapproximation of `rabin`, but captures the preservation of color parity for cycles generated by strategies that may not be positional, for example those that have finite or infinite memory. Thus, equivalence relation $\equiv^{\alpha}$ can be defined similarly to how $\equiv$ is defined in Definition 1, where strategies are now deterministic but not necessarily positional.

7. Experimental results

We now provide some experimental results. These experiments are meant to evaluate the algorithms `rabin` and $\texttt{rabin}^{\alpha}$. All experiments are carried out on a test server that has two Intel E5 CPUs – with 6-core each running at 2.5 GHz – and 48 GB of RAM. During the experiments, Intel performance enhancing technologies such as Turbo Boost and Hyper-Threading were turned off. The implementations of algorithms used on these experiments are written in Scala and realize, for sake of simplicity, all game elements as objects. This design decision is made since our main interest is in descriptive complexity measures and in the comparison of *relative* computation time. All computation times are reported in milliseconds.

We now sketch our implementations of algorithm $\texttt{rabin}^{\alpha}$. It reduces cycle detection to the decomposition of the graph into strongly connected components, using Tarjan's algorithm (which is linear in the number of edges). The rank function within $\texttt{rabin}^{\alpha}$ is only needed for complexity and termination analysis, we replaced it with Booleans that flag whether `cycle` or `pop` had an effect.

Our implementation of the standard static compression algorithm simply removes gaps between colors, e.g. a set of colors $\{0, 3, 4, 5, 6, 8\}$ is being compressed to $\{0, 1, 2, 3, 4\}$. Below, we write $s(c)$ for the statically compressed version of coloring function c.

7.1. First experiment

In our first experiment, we program algorithm `rabin` by reducing simple cycle detection to incremental SAT solving. This approach does not scale to games (with graph structure described in Section 7.3) with more than 40 nodes. But for those games for which this could compute the Rabin index, our efficient approximation $\texttt{rabin}^{\alpha}(V, E, c)$ of $\texttt{rabin}(V, E, c)$ often computes the Rabin index $\mathsf{RI}(c)$ or does get very close to it.

7.2. Second experiment: structured games

Our second experiment compares the effectiveness of color compression of the approximative algorithm $\texttt{rabin}^{\alpha}$ to a known color compression algorithm – the so called static color compression. We here want to see by how much $\texttt{rabin}^{\alpha}$ reduces the index of the game in comparison to static color compression. And we want to learn how effective $\texttt{rabin}^{\alpha}$ is as a preprocessor to Zielonka's solver of parity games (called `Zie` here) [17]. For the latter, we are interested in also comparing this to static color compression as a preprocessor for `Zie`.

Fig. 7 shows results of this second experiment for structured games. We use PGSolver to generate these non-random games. A detailed description of these games can be found in [9], whose notation for such games we adopt here. Seven different game types were evaluated for the listed parameter choices. Each row in Fig. 7 corresponds to such a game type and shows the average of measurements made for 100 executions of the corresponding game type. For each such type, we repeat and average results for the same game to account for average lapse time in experiments. Of course, the number of iterations within $\texttt{rabin}^{\alpha}$ does not change when repeating the execution of a game (so the average equals that constant). In the top of that figure, we see the (average of the) index of the generated game, the index obtained from this through static color compression, and the index computed by $\texttt{rabin}^{\alpha}$ applied to the generated game. We learn from this table that $\texttt{rabin}^{\alpha}$ has significantly reduced the indices of `Recursive Ladder`, `Strategy Impr`, and `Model Checker Ladder`, where $\mathsf{RI}^{\alpha}(c)$ is 0% to 35% of the index $\mu(s(c))$ of the statically compressed coloring function. Game types `Ladder` and `Tower of Hanoi` have very low indices and their colors cannot be compressed further. The definition of game type `Clique` implies that method `cycle` has no effect on such games, only `pop` manages to reduce the index of such games by 1.

At the bottom of Fig. 7, we report the time taken to execute static compression and $\texttt{rabin}^{\alpha}$, as well as the number of iterations that $\texttt{rabin}^{\alpha}$ runs until `cycle` and `pop` have no effect, i.e. the number of iterations needed for $\mu(c)$ to reach $\mathsf{RI}^{\alpha}(c)$. Finally, we here record the wall-clock time required to solve original, statically compressed, and $\texttt{rabin}^{\alpha}$-compressed games, using Zielonka's solver [17]. From this table we learn that the application of $\texttt{rabin}^{\alpha}$ improves performance of

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005

Game Type	$\mu(c)$	$\mu(s(c))$	$\mathrm{RI}^{\alpha}(c)$
`Clique`[100]	100.00	100.00	99.00
`Ladder`[100]	2.00	2.00	2.00
`Jurdziński`[5 10]	12.00	12.00	11.00
`Recursive Ladder`[15]	48.00	46.00	16.00
`Strategy Impr`[8]	237.00	181.00	9.00
`MC Ladder`[100]	200.00	200.00	0.00
`Towers of Hanoi`[5]	2.00	2.00	1.00

Game Type	#I	S	$\texttt{rabin}^{\alpha}$	`Zie`	S; `Zie`	$\texttt{rabin}^{\alpha}$; `Zie`
`Clique`[100]	2	0.15	318.57	7.79	6.97	6.88
`Ladder`[100]	1	0.14	6.34	3.57	2.92	2.61
`Jurdziński`[5 10]	2	0.11	14.98	95.84	94.40	93.66
`Recursive Ladder`[15]	2	0.05	8.43	408.09	402.87	239.89
`Strategy Impr`[8]	2	0.13	31.98	229.06	52.38	10.14
`MC Ladder`[100]	2	0.17	123.18	32.85	32.79	0.36
`Towers of Hanoi`[5]	2	0.63	114.27	31.65	31.14	49.20

Fig. 7. Experiments for structured game types. Top: index $\mu(c)$ of game, index of statically compressed coloring function $\mu(s(c))$, and index of game computed by $\texttt{rabin}^{\alpha}$. Bottom: run-time characteristics averaged over 100 runs of the game; $\#I$ is the number of iterations within $\texttt{rabin}^{\alpha}$, S is run-time of static color compression, $\texttt{rabin}^{\alpha}$ shows the run-time of $\texttt{rabin}^{\alpha}$, followed by the run-time of Zielonka's solver on original game (`Zie`), the statically compressed game (S; `Zie`) and the game preprocessed with $\texttt{rabin}^{\alpha}$ ($\texttt{rabin}^{\alpha}$; `Zie`).

Game Configs	$\mu(c)$	$\mu(s(c))$	$\mathrm{RI}^{\alpha}(c)$
100/1/20/100	99.19	46.19	36.31
200/1/40/200	198.93	92.03	80.45
400/1/80/400	399.00	184.78	172.14
800/1/160/800	799.13	371.83	357.68
1000/1/200/1000	998.97	463.91	449.35
1600/1/320/1600	1599.07	739.37	723.77

Game Configs	#I	S	$\texttt{rabin}^{\alpha}$	`Zie`	S; `Zie`	$\texttt{rabin}^{\alpha}$; `Zie`
100/1/20/100	2.09	0.20	31.12	5.21	4.35	4.28
200/1/40/200	2.04	0.18	175.95	10.58	9.98	9.93
400/1/80/400	2.03	0.18	1637.22	38.72	35.82	35.67
800/1/160/800	2.02	0.39	16666.34	146.81	142.14	141.79
1000/1/200/1000	2.06	0.49	35505.83	213.68	205.29	204.64
1600/1/320/1600	2.08	1.43	300850.49	656.06	614.70	611.26

Fig. 8. Experiments for random game types. Top: averages of the index $\mu(c)$ of game, the index of statically compressed coloring function $\mu(s(c))$, and the index of game computed by $\texttt{rabin}^{\alpha}$. Bottom: run-time characteristics averaged over 100 runs of the game; $\#I$ is the number of iterations within $\texttt{rabin}^{\alpha}$, S is run-time of static color compression, $\texttt{rabin}^{\alpha}$ shows the run-time of $\texttt{rabin}^{\alpha}$, followed by the run-time of Zielonka's solver on original game (`Zie`), the statically compressed game (S; `Zie`) and the game preprocessed with $\texttt{rabin}^{\alpha}$ ($\texttt{rabin}^{\alpha}$; `Zie`).

solvers for some game types. For the three game types mentioned above, we observe 40% to 99% in solver time reduction between solving statically compressed and $\texttt{rabin}^{\alpha}$-compressed games.

The time required to perform static compression is low compared to the time needed for $\texttt{rabin}^{\alpha}$-compression, but $\texttt{rabin}^{\alpha}$-compression followed by solving the game is still faster than solving the original or solving statically compressed games for `Recursive Ladder` and `Strategy Impr`.

7.3. Third experiment: random games

We now discuss the results of our second experiment on random games. The notation we use here to describe randomly generated parity games is

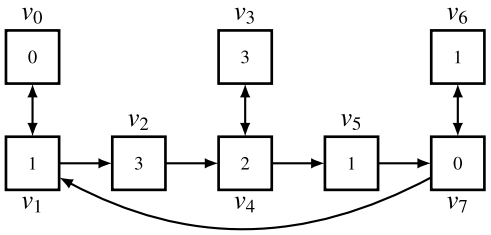
$$xx/yy/zz/cc \tag{6}$$

where xx is the number of nodes (node ownership is determined by a fair coin flip for each node independently), with between yy to zz out-going edges for each node, and with colors at nodes chosen at random from $\{0, \ldots, cc\}$. We fix the minimal number of out-going edges (yy) to be 1. This means that the games have no dead-ends. We also disallow self-loops (no (v, v) in E). The generation of these games is realized by the `random` method of the PGSolver tool [9].

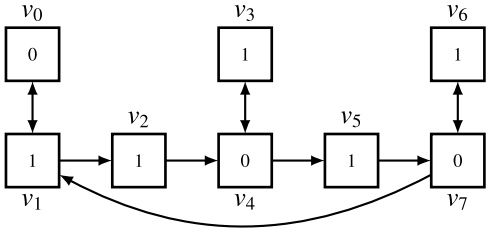
Fig. 8 shows the average statistics of 100 runs of experiments on five random game configurations. (Our experiments on larger random games are consistent with the data reported here, and so omitted.) The meaning of these tables is the same as for the tables in Fig. 7, except that the first column lists the type of *random* games generated here.

The results indicate that static compression is effective in reducing the colors for randomly generated games, it achieves around 54% index reduction for all game types reported in Fig. 8. The color compression of $\texttt{rabin}^{\alpha}$ achieves a further 2%

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005



(a) A parity game G for which psolB can solve no nodes.



(b) Parity game G' produced by applying $\texttt{rabin}^{\alpha}$ on the above parity game G.

Fig. 9. An example of a parity game G that psolB cannot solve at all, but that psolB solves completely after $\texttt{rabin}^{\alpha}$ is run as preprocessor.

to 21% reduction. Due to the relatively small additional index reduction by $\texttt{rabin}^{\alpha}$, we do not see much improvement in solving $\texttt{rabin}^{\alpha}$-compressed games over solving statically-compressed ones.

The results in Fig. 8 show that these games take an average of more than 2 $\texttt{rabin}^{\alpha}$ iterations. This indicates that certain game structure, such as the one found in the game in Fig. 3, is present in our randomly generated games.

We summarize the insights of this second experiment. The experimental results show that $\texttt{rabin}^{\alpha}$ is able to reduce the indices of parity games significantly and quickly, for certain structured games such as Recursive Ladder. Hence it effectively improves the overall solver performance for those games. However, algorithm $\texttt{rabin}^{\alpha}$ has a negative effect on the overall performance for other non-random games and for the random games we generated, when we consider $\texttt{rabin}^{\alpha}$-compression time plus solver time.

7.4. Fourth experiment: enhancing the precision of partial solvers

Partial solvers for parity games were proposed in [11]. For parity game G with node set V, a partial solver produces as output winning regions W_0 and W_1 of players, along with strategies that win on these regions for the respective players – just as a full solver does. But for partial solvers the node set $W_0 \cup W_1$ may be a proper subset of the node set V of input parity game G. Therefore, partial solvers also output a residual game G_r whose node set is $V \setminus (W_0 \cup W_1)$. It is therefore interesting to see whether algorithm $\texttt{rabin}^{\alpha}$, used as a preprocessor, can decrease the size of the node set of such residual games. We conduct this experiment for the partial solver psolB introduced in [11]. This solver examines each color k of the input game G and determines whether there are node sets $X \subseteq Y$ in G such that all nodes in X have color k, and player $k\%2$ can reach from all nodes in Y some node in X whilst preventing that nodes of color less than k are visited along the way. If so, node set Y is called a *fatal attractor* for color k and is won by player $k\%2$ in game G. The partial solver psolB removes the usual $k\%2$ attractor of a fatal attractor Y for some k. This process is repeated until no color of the residual game has a fatal attractor left.

We next give an example demonstrating that algorithm $\texttt{rabin}^{\alpha}$ can indeed have a positive effect on the partial solver psolB.

Example 4. Consider the parity game in Fig. 9(b), taken from [11] where it is shown that psolB can solve no nodes of this game. In Fig. 9(a), we see the result of $\texttt{rabin}^{\alpha}$ on that input game. This resulting game has index 2 and psolB solves all games of index 2 completely [11]. So this is an example where the preprocessor $\texttt{rabin}^{\alpha}$ turns a game that psolB cannot solve at all into one that psolB can solve completely.

In evaluating how typical such positive effects of $\texttt{rabin}^{\alpha}$ are on psolB, it also makes sense to compare the running times of partial solvers for original and preprocessed games. In [11] it is shown that the residual game of psolB is independent of the order in which it searches for fatal attractors of colors in the game. Therefore, our measure of how many additional nodes $\texttt{rabin}^{\alpha}$ can remove as a preprocessor of psolB is not affected by such an implementation choice. Since psolB solves almost all standard structured parity games from the PGSolver suite completely [11], we confine our attention

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005

i	$\#G$	$\mu(c)$	$\mathrm{RI}^{\alpha}(c)$	$\#I$	$\texttt{rabin}^{\alpha}$
2	6842	3.11	0.83	1.95	1.19
3	1888	5.01	1.63	1.95	1.67
4	1116	9.31	3.56	2.00	3.14
5	2485	19.37	6.37	2.04	4.71
6	6853	36.19	8.21	2.05	8.91
7	16925	70.18	12.94	2.13	19.40
8	43561	134.24	20.08	2.19	69.03
9	86304	237.41	35.23	2.29	375.67

i	psolB	$\texttt{rabin}^{\alpha}$; psolB	r	$c;r$	*Eff*	*max*	*avg*
2	0.33	0.28	4.0	0.48	88	4	4
3	0.32	0.32	6.73	2.83	59	8	6
4	0.71	0.67	9.55	6.22	33	16	10
5	1.18	1.10	16.77	14.45	10	32	23
6	2.69	2.09	35.67	28.07	22	64	34
7	5.49	2.93	66.46	55.98	17	128	61
8	15.41	8.96	133.72	118.89	9	256	164
9	31.38	24.71	277.75	274.79	4	85	74

Fig. 10. Experiments for randomly generated games with configuration as in (7). Top: the first two columns depict the value of i and the number of games generated in order to have 100 games that psolB does not solve completely ($\#G$); the next four columns have the same meaning as for the previous two experiments but averaged over these 100 games. Bottom: the first two columns are the average solver time using psolB on the original (psolB) and on $\texttt{rabin}^{\alpha}$-compressed ($\texttt{rabin}^{\alpha}$; psolB), respectively. The next two columns depict the average size of the residual game for psolB (the r), and for $\texttt{rabin}^{\alpha}$; psolB (the $c;r$). The last columns show the number of these 100 games for which $\texttt{rabin}^{\alpha}$; psolB removes more nodes from the input game than psolB (*Eff*), and the maximum (*max*) and average of how many more nodes $\texttt{rabin}^{\alpha}$; psolB removes.

here to randomly generated parity games with configurations specified as in (6). Specifically, we generate random games with configuration

$$2^i/1/[1..2^i]/[1..2^i] \tag{7}$$

where i ranges from 2 up to 9. These games have 2^i nodes. For each node, its out-degree in (V, E) is randomly selected from the integer interval $[1, 2^i]$ such that there are no self-loops. Also, the index of these games is randomly selected from the integer interval $[1, 2^i]$.

For each such value of i, we program an iteration that keeps generating random games for this configuration until 100 such games are identified that psolB does not solve completely. Fig. 10 shows our results for this in tabular form. For each of these 100 games we record similar information as in the previous two experiments. Since we here want to study how $\texttt{rabin}^{\alpha}$ may enhance the precision of the partial solver psolB, we did not run any static color compression in this experiment.

In Fig. 10, we see that it takes longer to first run $\texttt{rabin}^{\alpha}$ and then psolB on the output of $\texttt{rabin}^{\alpha}$ than it takes to run psolB on the original game. But we would like to stress that the intention of these experiments is to show that the changes made by $\texttt{rabin}^{\alpha}$ are beneficial to parity game solving, not to demonstrate $\texttt{rabin}^{\alpha}$ is ready to use as a practical tool. The results in Fig. 10 show that $\texttt{rabin}^{\alpha}$ as preprocessor of psolB helps with removing more nodes. However, we can also see that there are less games where this happens when the size of the game increases. Noting the sizes of games for values of i, we see a similar pattern in the maximum and average of such increased node removal when $\texttt{rabin}^{\alpha}$ is used as a preprocessor for psolB. On the other hand, the column $\#G$ shows that more and more games have to be generated as i increases to find 100 games that psolB does not solve completely.

We also see that $\texttt{rabin}^{\alpha}$ is effective in reducing indices of these 100 games, with 62% to 85% index reductions for all values of i. These reductions have a positive effect on psolB running times: the data in columns psolB and $\texttt{rabin}^{\alpha}$; psolB show a running time reduction of 0% to 47% for most i when using $\texttt{rabin}^{\alpha}$ as preprocessor of psolB (ignoring time taken to run $\texttt{rabin}^{\alpha}$).

Another insight, not shown in these data but observable in the raw data of this experiment, is that the 100 games identified for a value of i tend to have a low maximal out-degree. Our intuition for this is that in such games the probability of a node v_i having an anchor value j is lower as there are fewer cycles in the game. Therefore, $\texttt{rabin}^{\alpha}$ should become more effective in such games.

We repeated this experiment with variants of the configuration in (7). In a first variant, we changed that configuration so that the index of games is restricted to the integer interval $[2, 4]$. Then algorithm $\texttt{rabin}^{\alpha}$ is expected to be less effective. The observed solver reduction times now only vary from 0% to 39% now. The effectiveness measure *Eff* is similar here to the one observed in Fig. 10 though.

A second variant of (7) we ran changes that configuration also only in one place, so that the out-degree is now restricted to the integer interval $[2, 4]$ (and so still allows for an index in the range of 1 and 2^i). The results for this variant were also similar, except that the low out-degree meant that the 100 games were found much quicker than in the other variants.

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005

```
Rabin_WA(V, E, c) {
  m = max { c(v) | v ∈ V};
  for (v ∈ V) { c'(v) = m; }
  reduce(V, E, c, c', m);
  return pushDown(V, c');
}

reduce(V, E, c, c', m) {
  i = m;
  SCCs = set of maximally non-trivial SCCs of (V, E);
  for (R ∈ SCCs){
    if (π(R) == m) { k = m; }
    else {
      R' = {v ∈ R | c(v) ≠ π(R)};
      k = reduce(R', E|_R', c|_R', c'|_R', m);
      if (π(R) − k is odd) { k = k − 1; }
    }
    for (v ∈ {w ∈ R | c(w) = π(R)}) { c'(v) = k; }
    i = min{i, k};
  }
  return i;
}

pushdown(V, c') {
  n = min { c'(v) | v ∈ V};
  if (n is odd) { n = n−1; }
  for (v ∈ V) { c'(v) = c'(v) − n; }
  return c';
}
```

Fig. 11. Algorithm $\texttt{Rabin}_{\text{WA}}$ to compute Rabin index [2], but modified to work with *min-parity* (instead of max-parity) parity automaton $A = (V, E, c)$, where $R \subseteq V$, $\pi(R) = \min\{c(v) \mid v \in R\}$, $E|_R$ is E with restriction to nodes in R, and similarly for $c|_R$.

In [3], it has been shown that priority propagation have no effect on `psolB` when used as a preprocessor: it neither eliminates nor creates fatal attractors in the input game. In the above context, it is interesting to note that priority propagation does not exploit cycle structure of the game, unlike algorithm $\texttt{rabin}^{\alpha}$ which has a (positive) effect on `psolB` because of its cyclic analysis.

8. Related work

Carton and Maceiras define a notion of Rabin index for deterministic parity word automata in [2], where they also develop an algorithm that computes a coloring function on the same automaton that witnesses this Rabin index. We now show that our notion $\mathsf{RI}^{\alpha}(c)$ and algorithm $\texttt{rabin}^{\alpha}$ are very similar to their notion and computation of Rabin index for deterministic parity word automata.

Deterministic parity word automata can be thought of as 1-player parity games, where the player chooses input letters. An infinite word can be compared to a strategy with memory for the player. In [2], max-parity deterministic parity automata are used. We present the work in [2] with the equivalent notion of min-parity deterministic parity word automata, since this allows us to directly relate their work to our own work reported here for min-parity games. A word is accepted if its strategy is winning, that is, if the minimal color to be visited infinitely often is even. Minimization of the Rabin index should preserve the language of the automaton or, put in our terms, every winning strategy should remain to be winning.

Their algorithm for this minimization, which we call $\texttt{Rabin}_{\text{WA}}$, is shown in Fig. 11 but reformulated in this figure to work with min-parity instead of max-parity deterministic parity automata. Algorithm $\texttt{Rabin}_{\text{WA}}$ constructs the "coloring dependencies" of all states in an automaton arena by decomposing the automaton into maximal (non-trivial) strongly connected components (*SCC*s). For each R being a maximal *SCC*, algorithm $\texttt{Rabin}_{\text{WA}}$ removes the states with the minimal color (and pushes them onto a stack), then recursively *SCC* decomposes the remaining arena of R.

Eventually, the input arena is reduced to a set of states that exist in their own respective *SCC*s (hence do not exist in the same cycle as each other). We first let k be the maximal color m, then assign k to these states. The algorithm then propagates the new color k to the states in the "layer" below. Those states receive a new color k or $k-1$, depending on whether their original parities equal the parities of the states in the "layer" above. In essence, *SCC* decomposition is used to detect the cycle dependency of states. Finally, `pushDown` reduces the minimal color to 0 or 1.

Our implementation of $\texttt{rabin}^{\alpha}$ also uses SCC decompositions and so our algorithm $\texttt{cycle}^{\alpha}$ has almost the same input/output behavior as $\texttt{Rabin}_{\text{WA}}$. The input/output differences between $\texttt{Rabin}_{\text{WA}}$ and $\texttt{rabin}^{\alpha}$ are a result of subtle definitional variations. Our notion of abstract Rabin index takes the maximal color of the game arena as descriptive complexity

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005

measure; their notion of Rabin index is concerned with the maximal length of alternating (positive) chains of essential sets, where a node set is essential if it is the set of nodes of some cycle. The length of such a maximal chain differs from the maximal color in the arena with no color gaps by at most 1 (in either direction). Therefore, the Rabin index in [2] and our abstract Rabin index are almost the same and, consequently, algorithms $\texttt{Rabin}_{\texttt{WA}}$ and $\texttt{rabin}^{\alpha}$ compute almost the same output.

The correctness proof for (the equivalent of) $\texttt{Rabin}_{\texttt{WA}}$ in [2] depends upon the fact that the union of two essential sets with non-empty intersections is essential again. In other words, two cycles that intersect can be interpreted as a single cycle. But the latter is no longer true for *simple* cycles and so it does not seem possible to adapt $\texttt{Rabin}_{\texttt{WA}}$ to compute our (non-abstract) Rabin index for parity *games* – something that we showed algorithm `rabin` is able to do.

9. Conclusions

We have provided a descriptive measure of complexity for parity games that (essentially) measures the number of colors needed in a parity game if we forget the ownership structure of the game but if we do not compromise the winning regions or winning strategies by changing its colors.

We called this measure the Rabin index of a parity game. We then studied this concept in depth. By analyzing the structure of simple cycles in parity games, we arrived at an algorithm that computes this Rabin index in exponential time.

Then we studied the complexity of the decision problem of whether the Rabin index of a parity game is at least k for some fixed $k > 0$. For k equal to 1, we saw that this problem is in P, but we showed NP-hardness of this decision problem for all other values of k. These lower bounds therefore also apply to games that capture these decision problems in game-theoretic terms.

Next, we asked what happens if our algorithm `rabin` abstractly interprets all detection checks for simple cycles through detection checks for cycles. The resulting algorithm $\texttt{rabin}^{\alpha}$ was then shown to run in polynomial time, and to compute an abstract and sound approximation of the Rabin index.

We then evaluated this concept of Rabin index experimentally. We did this by studying its approximating algorithm $\texttt{rabin}^{\alpha}$. In these experiments, we wanted to understand whether $\texttt{rabin}^{\alpha}$ can be used as a preprocessor so that existing solvers would benefit from solving a parity game with colors reduced by $\texttt{rabin}^{\alpha}$. Our result indicate that $\texttt{rabin}^{\alpha}$ is only of limited value as a means of speeding up solver time, and this is shared with techniques such as priority propagation [9]. More concretely, these experiments were performed on random and non-random games. We observed that $\texttt{rabin}^{\alpha}$-compression plus Zielonka's solver [17] achieved 29% and 85% time reduction for `Jurdziński` and `Recursive Ladder` games, respectively, over solving the original games. But for other game types and random games, no such reduction was observed. We also saw that for some structured game types, the abstract Rabin index is dramatically smaller than the index of the game.

A more theoretical experiment focused on partial solvers, that run in polynomial time but may not solve a parity game completely. We asked whether $\texttt{rabin}^{\alpha}$ can help such partial solvers to solve more games completely, and our experiments do indeed confirm this. Although it appears that the effectiveness of this processor on the types of random models we studied wanes quickly as the size of models grows exponentially.

We finally list some research issues that are raised by the work reported in this paper. The complexity measure

$$\mathsf{RI}^{\alpha}(c) - \mathsf{RI}(c) \tag{8}$$

appears to be of interest. Intuitively, it measures the difference of the Rabin index based on the structure of cycles with that based on the structure of simple cycles. From Fig. 4(b) we already know that this measure can be arbitrarily large. Understanding this measure better may provide a deeper insight into the complexity of colored arenas.

It will also be of interest to study variants of $\mathsf{RI}(c)$ that are targeted for specific solvers. For example, the SPM solver in [12] favors fewer occurrences of odd colors but also favors lower index. This suggests a measure with a lexicographical order of the Rabin index followed by an occurrence count of odd colors.

There is also the question of whether there are ways of replacing method `cycle` with some other method that could soundly detect anchors in colored arenas whose nodes preserve the ownership of parity games. Deriving such methods may indeed result in the design of new parity game solvers.

Acknowledgments

We thank the anonymous referees for their careful and critical review of the submitted manuscript. Their constructive comments and suggestions helped to improve the clarity and quality of this article, notably Section 8.

References

[1] D. Berwanger, A. Dawar, P. Hunter, S. Kreutzer, Dag-width and parity games, in: STACS 2006, 2006, pp. 524–536.
[2] O. Carton, R. Maceiras, Computing the Rabin index of a parity automaton, RAIRO Theor. Inform. Appl. 33 (6) (1999) 495–506.
[3] Z. Du, Fatal attractors in parity games, Master's thesis, Department of Computing, Imperial College, London, 2013.
[4] E.A. Emerson, C.S. Jutla, Tree automata, mu-calculus and determinacy (extended abstract), in: FOCS 1991, 1991, pp. 368–377.

[5] E.A. Emerson, C.S. Jutla, A.P. Sistla, On model-checking for fragments of μ-calculus, in: CAV 1993, 1993, pp. 385–396.
[6] S. Even, A. Itai, A. Shamir, On the complexity of timetable and multicommodity flow problems, SIAM J. Comput. 5 (4) (1976) 691–703.
[7] S. Fortune, J.E. Hopcroft, J. Wyllie, The directed subgraph homeomorphism problem, Theor. Comput. Sci. 10 (1980) 111–121.
[8] O. Friedmann, M. Lange, Solving parity games in practice, in: ATVA 2009, 2009, pp. 182–196.
[9] O. Friedmann, M. Lange, The PGSolver collection of parity game solvers, Technical Report, Version 3, Institut für Informatik, LMU Munich, 2010.
[10] E. Grädel, W. Thomas, T. Wilke (Eds.), Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001], Lecture Notes in Computer Science, vol. 2500, Springer, 2002.
[11] M. Huth, J.H.P. Kuo, N. Piterman, Fatal attractors in parity games, in: FoSSaCS 2013, 2013, pp. 34–49.
[12] M. Jurdzinski, Small progress measures for solving parity games, in: STACS, 2000, pp. 290–301.
[13] A.W. Mostowski, Games with forbidden positions, Technical Report 78, University of Gdańsk, 1991.
[14] C. Stirling, Local model checking games, in: CONCUR 1995, 1995, pp. 1–11.
[15] J. Vöge, M. Jurdzinski, A discrete strategy improvement algorithm for solving parity games, in: CAV 2000, 2000, pp. 202–215.
[16] K.W. Wagner, On omega-regular sets, Inf. Control 43 (2) (1979) 123–177.
[17] W. Zielonka, Infinite games on finitely coloured graphs with applications to automata on infinite trees, Theor. Comput. Sci. 200 (1–2) (1998) 135–183.

Please cite this article in press as: M. Huth et al., The Rabin index of parity games: Its complexity and approximation, Inf. Comput. (2015), http://dx.doi.org/10.1016/j.ic.2015.06.005