Journal of Computer and System Sciences 84 (2017) 243–262

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Solving parity games in big steps

Sven Schewe

Department of Computer Science, University of Liverpool, Ashton Building, Ashton Street, Liverpool L69 3BX, United Kingdom

ARTICLE INFO

Article history:
Received 22 April 2013
Received in revised form 9 January 2015
Accepted 28 September 2016
Available online 11 October 2016

Keywords:
Parity games
Finite games of infinite duration

ABSTRACT

This article proposes a new algorithm that improves the complexity bound for solving parity games. Our approach combines McNaughton's iterated fixed point algorithm with a preprocessing step, which is called prior to every recursive call. The preprocessing uses ranking functions similar to Jurdziński's, but with a restricted co-domain, to determine all winning regions smaller than a predefined parameter. The combination of the preprocessing step with the recursive call guarantees that McNaughton's algorithm proceeds in big steps, whose size is bounded from below by the chosen parameter. Higher parameters lead to smaller call trees, but they also result in an expensive preprocessing step. An optimal parameter balances the cost of the recursive call and the preprocessing step, resulting in an improvement of the known upper bound for solving parity games from $O\left(m\left(\frac{2n}{c}\right)^{\frac{1}{2}c}\right)$ to approximately $O\left(m\left(\frac{6e^{1.6}n}{c^2}\right)^{\frac{1}{3}c}\right)$.

1. Introduction

Parity games have many applications in model checking [13,7,6,1,28,14] and synthesis [28,13,26,24,19,23,25]. In particular, modal and alternating-time μ-calculus model checking [28,1], synthesis [25,19,23] and satisfiability checking [28,13,26,24] for reactive systems, module checking [14], and ATL* model checking [6,1] can be reduced to solving parity games. This relevance of parity games led to a series of different approaches to solving them [17,8,16,20,30,5,29,10,11,27,18,15,2,4,12,9].

The complexity of solving parity games is still an open problem. Parity games are memoryless determined [7,3], which implies that nondeterministic algorithms can determine winning regions and strategies for both players. Due to their symmetry, they are therefore in NP∩CoNP [7], and by reduction to payoff games [30], in UP∩CoUP [10]. Determining their membership in P continues to be a major challenge.

All current deterministic algorithms have complexity bounds which are (at least) exponential in the number of colours [17,8,30,5,29,11,4] ($n^{O(c)}$), or in the square-root of the number of game positions [16,12,4] (approximately $n^{O(\sqrt{n})}$). Practical considerations suggest that we should assume that the number of colours is small compared to the number of positions. Indeed, almost all of the applications listed above result in parity games where the number of colours is (sub-)logarithmic in the size of the game arena. μ-calculus model checking is the only exception. In μ-calculus model checking, however, the size of the game is determined by the product of the transition system under consideration (which is usually large), and the size of the formula (which is usually small). The number of colours is determined by the alternation depth of the specification, which, in turn, is usually small compared to the specification itself. Algorithms that are exponential only in the number of colours are therefore considered to be the most attractive.

E-mail address: sven.schewe@liverpool.ac.uk.

http://dx.doi.org/10.1016/j.jcss.2016.10.002
0022-0000/

The first representatives of algorithms in the complexity class $n^{O(c)}$ follow the iterated fixed point structure induced by the parity condition [17,8,29]. The iterated fixed point construction leads to a time complexity of $O\left(m\left(\frac{n}{c}+1\right)^{c-1}\right)$ for parity games with m edges, c colours, and n game positions. The upper complexity bound for solving parity games was first reduced by Browne et al. [5] to $O\left(m\left(\frac{2n}{c}\right)^{\lceil 0.5c\rceil+1}\right)$, and slightly further by Jurdziński [11] to $O\left(c\,m\,(\frac{n}{\lfloor 0.5c\rfloor})^{\lfloor 0.5\,c\rfloor}\right)$.

The weakness of recursive algorithms that follow the iterated fixed point structure [17,8,29] is the potentially incremental update achieved by each recursive call. Recently, a big-step approach [12] has been proposed to reduce the complexity of McNaughton's algorithm for games with a high number of colours ($c \in \omega(\sqrt{n})$) to the bound $n^{O(\sqrt{n})}$ known from randomized algorithms [16,4].

1.1. The lineage of our approach

The approach we discuss is drawing from McNaughton's approach [17,8,29] and the extension to big steps of Jurdziński, Paterson, and Zwick [12]. The core observation of McNaughton's approach is that it helps to find solutions to paradises, a particular type of sub-games. A paradise is a region of a game where one player can force a win without leaving the paradise. Once a paradise is known, one can divide solving the game into three parts: the paradise, the attractor of the paradise, and the co-game of the attractor.

In [17,8,29], a paradise for the player who loses on the highest colour is constructed by solving a parity game with one colour less. (If there is no such paradise, solving the game becomes simple.)

Jurdziński, Paterson, and Zwick [12] observed that this provides very weak guarantees if the number of colours is high, say in the order of the number of game positions. They adjusted the algorithm by first producing all 'small' paradises up to size $\sqrt{n}$. This can be done by individually considering all sets up to the size of $\sqrt{n}$ and checking whether or not they are paradises of the player who loses on the highest colour of the game. The union of these paradises form a paradise that must contain all small paradises. Their algorithm first uses this novel way of constructing a paradise and then uses the recursive call from McNaughton's algorithm [17,8,29]. It thus either provides a paradise strictly bigger than $\sqrt{n}$ or an immediate solution.

The limitation of this construction is that a brute force construction of a paradise that contains all small paradises does not benefit from a small number of colours. We overcome this limitation by introducing a technique for the construction of small paradises that does benefit from small number of colours. This technique is a simple generalisation of Jurdziński's 'small progress measures' [11]. His approach is adapted by restricting the co-domain of the used ranking function. The resulting algorithm is exploited in the construction of paradises that are bounded by the size of a parameter par. Compared to [12], this results in a significant cut in the cost for finding small winning regions, since the running time for the preprocessing algorithm is polynomial in the parameter, and exponential only in the number of colours:

$$O\left(c\,m\,\binom{\mathsf{par}+\lceil 0.5c\rceil}{\mathsf{par}}\right).$$

1.2. Contribution

The different way of constructing paradises that contain all small paradises (up to a parameter) improves the complexity of McNaughton's algorithm for the relevant lower end of the spectrum of colours, resulting in approximately the complexity

$$O\left(m\left(\frac{6e^{1.\bar{6}}n}{c^2}\right)^{\gamma(c)}\right)$$

for solving parity games under the assumption that $c \in o(\sqrt{n})$, where

$$\gamma(c) = \frac{c}{3} + \frac{1}{2} - \frac{1}{3c} - \frac{4}{c^2}$$

if c is even, and

$$\gamma(c) = \frac{c}{3} + \frac{1}{2} - \frac{4}{c^2-1}$$

if c is odd.

Using a the parameter of approximately

$$\sqrt[3]{\frac{(3\sqrt[6]{e}n)^2}{2c}}$$

results in an $O\left(m\left(\frac{\kappa\, n}{c^2}\right)^{\gamma(c)}\right)$ complexity (for $\kappa \approx 6e^{1.\bar{6}}$) for solving parity games, which improves over the previously known $O\left(m\left(\frac{2n}{c}\right)^{\lfloor 0.5c\rfloor}\right)$ bound [11].

This reduces the exponential factor from $\lfloor \frac{c}{2} \rfloor$ to less than $\frac{c}{3} + \frac{1}{2}$. It is, after the reduction from $c - 1$ [17,8,29] to $\lceil \frac{c}{2} \rceil + 1$ by Browne et al. [5], the second improvement that reduces the exponential growth with the number of colours. The development of the known complexity bounds is outlined in the following table.

# colours	3	4	5	6	7	8	9
McNaughton [17]	$O(mn^2)$	$O(mn^3)$	$O(mn^4)$	$O(mn^5)$	$O(mn^6)$	$O(mn^7)$	$O(mn^8)$
Browne & al. [5]	$O(mn^3)$	$O(mn^3)$	$O(mn^4)$	$O(mn^4)$	$O(mn^5)$	$O(mn^5)$	$O(mn^6)$
Jurdziński [11]	$O(mn)$	$O(mn^2)$	$O(mn^2)$	$O(mn^3)$	$O(mn^3)$	$O(mn^4)$	$O(mn^4)$
Big Steps	$O(mn)$	$O(mn^{1\frac{1}{2}})$	$O(mn^2)$	$O(mn^{2\frac{1}{3}})$	$O(mn^{2\frac{3}{4}})$	$O(mn^{3\frac{1}{16}})$	$O(mn^{3\frac{9}{20}})$

Besides the improved complexity for a fixed number of colours, the approach also provides an improved development of the base of the exponential expression. While previous algorithms had a base of $O\left(\frac{n}{c}\right)$, this has shrunk to $O\left(\frac{n}{c^2}\right)$ in this approach.

When solving parity games, we are often interested in winning strategies for the players. For example, they serve as witnesses and counter examples in model checking, and as models in synthesis. When constructing these strategies, the improvement in the complexity of the discussed approach is even higher. Constructing winning strategies for both players does not increase the complexity of the proposed algorithm. The best previously known bound for constructing winning strategies [11] has been $O\left(c\,m\,(\frac{n}{\lceil 0.5c \rceil})^{\lceil 0.5\,c \rceil}\right)$.

# colours	3	4	5	6	7	8	9
Jurdziński [11]	$O(mn^2)$	$O(mn^2)$	$O(mn^3)$	$O(mn^3)$	$O(mn^4)$	$O(mn^4)$	$O(mn^5)$
Big Steps	$O(mn)$	$O(mn^{1\frac{1}{2}})$	$O(mn^2)$	$O(mn^{2\frac{1}{3}})$	$O(mn^{2\frac{3}{4}})$	$O(mn^{3\frac{1}{16}})$	$O(mn^{3\frac{9}{20}})$

This extra advantage is yielded by an adjustment of the evaluation of three colour games by a simple adjustment of Jurdziński's 'small progress measures' approach [11], which allow for determining the winning strategies of both players.

The article is an extended version of the paper *Solving Parity Games in Big Steps* [21] including the improved analysis of three colour games from [22].

2. Infinite games

Infinite games on finite graphs are composed of a game arena and an evaluation function. Most of the time, we are interested in finite games of infinite duration, the special case where the game arena is finite. We will first discuss arenas and then turn to the evaluation functions for safety, reachability, and parity games.

2.1. Arena

Games are played on arenas. An *arena* is a triple $\mathcal{A} = (V_0, V_1, E)$, where

- V_0 and V_1 are disjoint finite sets of positions, called the positions of Player 0 and Player 1, respectively,
- $V = V_0 \uplus V_1$ denotes the set of game positions, and
- $E \subseteq V \times V$ is a set of edges,

such that (V, E) is a directed graph. The arena is also required not to contain sinks; that is, every position $p \in V$ has at least one outgoing edge $(p, p') \in E$.

An arena is called a *single player arena* if all positions in V_0 or all positions in V_1 have out-degree 1. Games are called single player games, if their arena is a single player arena.

2.2. Plays

Intuitively, a game is played by placing a pebble on the arena. If the pebble is on a position $p \in V_0$, Player 0 chooses an edge $e = (p, p') \in E$ from p to a successor p' and moves the pebble to p'. Symmetrically, if the pebble is on a position $q \in V_1$, Player 1 chooses an edge $e' = (q, q') \in E$ from q to a successor q' and moves the pebble to q'. This way, they successively construct an infinite *play* $\pi = p_0 p_1 p_2 p_3 \ldots \in V^\omega$.

2.3. Strategies

For an arena $\mathfrak{A} = (V_0, V_1, E)$, a *strategy* for Player 0 is a function $f : V^* V_0 \to V$ that maps each finite history of a play that ends in a position $p \in V_0$ to a successor p' of p. (That is, there is an edge $(p, p') \in E$ from p to p'.) A play is *f-conform* if every decision of Player 0 in the play is in accordance with f.

A strategy is called *memoryless* if it only depends on the current position. A memoryless strategy for Player 0 can be viewed as a function $f : V_0 \to V$ such that $(p, f(p)) \in E$ holds for all $p \in V_0$.

For a memoryless strategy f, we denote with $\mathcal{A}_f = (V_0, V_1, E_f)$ the arena obtained from $\mathcal{A}$ by deleting the transitions from positions of Player 0 that are not in accordance with f. ($\mathcal{A}_f$ defines a directed graph where all positions of Player 0 have out-degree 1.) The analogous definitions are made for Player 1. Note that $\mathcal{A}_f$ is a single player arena.

2.4. Safety and reachability games

A *safety game* is a game $\mathcal{S} = (V_0, V_1, E, F)$ with arena $\mathfrak{A} = (V_0, V_1, E)$ and a set $F \subseteq V$ of final (or: bad) positions.

Each play of a safety game is evaluated by checking whether or not it is contained in $V \setminus F$: Player 0 wins a play $\pi = p_0 p_1 p_2 p_3 \dots$ if, for all $i \in \omega$, $p_i \notin F$. All games considered in this article are 0-sum games. For boolean outcome, this means that one player wins while the other player loses. In safety games, Player 1 thus wins if there is an $i \in \omega$ with $p_i \in F$. If we take the point of view of Player 1, the game becomes a *reachability* game, as Player 1 has the objective to eventually reach a position in F.

2.5. Parity games

A *parity game* is a game $\mathcal{P} = (V_0, V_1, E, \alpha)$ with arena $\mathfrak{A} = (V_0, V_1, E)$ and a surjective colouring function $\alpha : V \to \mathcal{C} \subset \omega$ that maps each position of $\mathcal{P}$ to a natural number. The co-domain of α is called the set of colours (or: priorities) and denoted by $\mathcal{C}$. Note that the co-domain $\mathcal{C}$ of α is finite as the domain V is finite. For technical convenience[1] we usually assume without loss of generality that the minimal colour of a parity game is $0 = \min\{\mathcal{C}\}$, and that $\mathcal{C}$ is an initial sequence[2] of the integers.

Each play is evaluated by the highest colour that occurs infinitely often. Player 0 wins a play $\pi = p_0 p_1 p_2 p_3 \dots$ if the highest colour occurring infinitely often in the sequence $\alpha(\pi) = \alpha(p_0)\alpha(p_1)\alpha(p_2)\alpha(p_3)\dots$ is even, while Player 1 wins if the highest colour occurring infinitely often in $\alpha(\pi)$ is odd.

2.6. Winning strategies and winning regions

A strategy f of Player 0 (Player 1) is called *p-winning* if all f-conform plays starting in p are winning for Player 0 (Player 1). A position p in V is *winning* for Player 0 (Player 1) if Player 0 (Player 1) has a p-winning strategy. We call the winning positions for Player 0 (resp. Player 1) the *winning region* of Player 0 (resp. Player 1), denoted W_0 (resp. W_1).

2.7. Notation

All operations on arenas extend to games. E.g., for a strategy f and a parity game $\mathcal{P} = (V_0, V_1, E, \alpha)$, $\mathcal{P}_f$ is the parity game with the arena consisting of the arena $\mathfrak{A}_f$ and the colouring function α.

For ease of notation, we sometimes use games when we refer to their arenas only. We also use the common intersection and subtraction operations on digraphs for arenas and games: $\mathcal{P} \cap V'$ and $\mathcal{P} \setminus V'$, for example, denote the parity games we get when we restricting the arena $\mathfrak{A}(V_0, V_1, E)$ of $\mathcal{P}$ to $\mathfrak{A} \cap V' = (V_0 \cap V', V_1 \cap V', E \cap V' \times V')$ and $\mathcal{A} \setminus F = (V_0, V_1, E) \cap V \setminus V'$, respectively. Note that our restriction to arenas without sinks forces us to check that the resulting arenas preserve this property.

As many algorithms have to refer to both players, we use Player σ for the player $\sigma \in \{0, 1\}$ (usually the player who wins when the maximal colour occurs infinitely many times), and we use $\overline{\sigma} = 1 - \sigma$ to refer to the other player.

2.8. Memoryless determinacy

A class of games is called *determined* if the union of the winning regions equals the set of positions. It is called *memoryless determined* if each player $\sigma \in \{0, 1\}$ has, for a game $\mathcal{G}$, a memoryless strategy f such that all plays in $\mathcal{G}_f$ that start in W_σ are winning for Player σ. Parity games are memoryless determined [7], and [3] contains a simple proof for their memoryless determinacy.

2.9. Solving parity games

When solving parity games, we distinguish two questions: the non-constructive problem is to determine, for a given a parity game $\mathcal{P}$ the winning regions of both players. The constructive extension requires additionally requires the construction of winning strategies for both players.

Most algorithms are presented with the non-constructive question in mind, but the constructive extension is usually simple. The only point where it requires special care is in the three colour games from Section 4.5.

[1] The restriction that the minimal colour is 0 is only technical. If no position with colour 0 exists, then we can reduce all colours by 1 and change the roles of Player 0 and 1. Winning regions and strategies for Player 0 (Player 1) in the resulting game are the winning regions and strategies for Player 1 (Player 0) in the original game.

[2] If a number, is missing in this sequence, we reduce all greater colours by 2 without changing acceptance of any play. Hence, winning regions and strategies are not affected by this transformation.

Procedure McNaughton($\mathcal{P}$):
1. set c to the highest colour occurring in $\mathcal{P}$
2. if $c = 0$ or $V = \emptyset$ then return $(V, \emptyset)$
3. set σ to c mod 2
4. set $W_{\overline{\sigma}}$ to $\emptyset$
5. repeat
   (a) set $\mathcal{P}'$ to $\mathcal{P} \setminus \sigma\text{-}Attractor(\alpha^{-1}(c), \mathcal{P})$
   (b) set (W'_0, W'_1) to $McNaughton(\mathcal{P}')$
   (c) if $W'_{\overline{\sigma}} = \emptyset$ then
      i. set W_σ to $V \setminus W_{\overline{\sigma}}$
      ii. return (W_0, W_1)
   (d) set $W_{\overline{\sigma}}$ to $W_{\overline{\sigma}} \cup \overline{\sigma}\text{-}Attractor(W'_{\overline{\sigma}}, \mathcal{P})$
   (e) set $\mathcal{P}$ to $\mathcal{P} \setminus \overline{\sigma}\text{-}Attractor(W'_{\overline{\sigma}}, \mathcal{P})$

Fig. 1. The algorithm $McNaughton(\mathcal{P})$ takes a parity game $\mathcal{P}$ as input and returns the ordered pair (W_0, W_1) of winning regions of the players 0 and 1, respectively. V and α denote the positions and the colouring function of the parity game $\mathcal{P}$.

3. McNaughton's algorithm

In this section, we summarise McNaughton's algorithm for solving parity games. The algorithm dates back to McNaughton [17] and has first been published in this form by Emerson and Lei [8,29].

The algorithm is discussed in some detail and some of the proofs are repeated because the algorithm discussed in Section 5 builds on them.

The algorithm is the algorithmic version of a simple proof of the memoryless determinacy for parity games. The proof uses an inductive argument over the number of positions. As an induction basis, games with only one game position are clearly memoryless determined: there is only one strategy, and it is memoryless. The game is won by Player 0 if the colour of this position is even and by Player 1 if the colour of this position is odd.

For general parity games $\mathcal{P}$ with highest colour c, McNaughton's algorithm (Fig. 1) first determines the set $\alpha^{-1}(c)$ of positions with maximal colour.

For the Player $\sigma = c$ mod 2 that wins if c occurs infinitely often (and is therefore the dominating colour), this algorithm then constructs the σ-attractor A of $\alpha^{-1}(c)$.

For an arena $\mathfrak{A} = (V_0, V_1, E)$, a set $T \subseteq V$ or target positions, and a Player $\sigma \in \{0, 1\}$, the *σ-attractor of T* is the set of game positions, from which Player σ can force the pebble into the set T of target positions. The σ-attractor A of a set T can be defined as the least set that contain T and that contain a game position p of Player σ in A if it contains some successor (all successors) of p:

$$\sigma - Attractor(T, \mathcal{A}) = \bigcap \{S \supset T \mid \forall p \in V_\sigma \forall p' \in S.\, (p, p') \in E \Rightarrow p \in S \text{ and } \forall p \in V_{\overline{\sigma}}.\, (\neg\exists p' \notin S.\, (p, p') \in E) \Rightarrow p \in S\}.$$

The σ-attractor A of a set T of target positions can be constructed by choosing

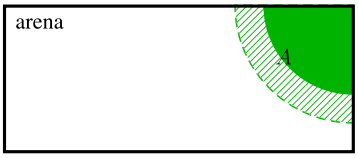
- $A_0 = T$,
- $A_{j+1} = A_j \cup \{p \in V_\sigma \mid \exists p' \in A_j.\, (p, p') \in E\} \cup \{p \in V_{\overline{\sigma}} \mid \forall (p, p') \in E.\, p' \in A_j\}$, and
- $A = \bigcup_{j \in \omega} A_j$.

The construction also provides a memoryless strategy for Player σ to move the pebble to T from all positions in A. Let $i_p = \min\{n \in \omega \mid p \in A_n\}$ denote the index of the first set A_{i_p} a position $p \in A$ is in. For a position in $p \in V_\sigma \cap A \setminus T$, p has a successor in $p' \in A_{i_p - 1}$ by definition, and we choose the attractor strategy f such that it maps p to such a successor. (For all $p \in V_\sigma \cap A \setminus T$. $f(p) \in A_{i_p - 1}$.) It is then easy to see that each f-conform play $p_0 p_1 p_2 \ldots$ that starts in A either eventually reaches T, or satisfies $i_{p_0} > i_{p_1} > i_{p_2} > i_{p_3} > \ldots$. However, as the integers are well founded, no such infinite chain exists, such that the latter alternative can be discarded. The play therefore eventually reaches the target set T.

A itself provides a memoryless strategy to keep the pebble out of A (and hence out of T) for Player $\overline{\sigma}$: Player $\overline{\sigma}$ can choose a strategy g, such that, for all $p \in V_{\overline{\sigma}} \setminus A$, $g(p) \notin A$. Note that such an option must exist in a finite game, as p would otherwise be in A. Let us assume for contradiction that a g-conform play $p_0 p_1 p_2 \ldots$ that starts outside of A

eventually reaches A. Let p_i be the first position of this play in A. Then $i > 0$ (as $p_0 \notin A$ holds by definition). The definition of A then implies that $p_{i-1} \in A$ (contradiction to p_i being the first position of the play in A).

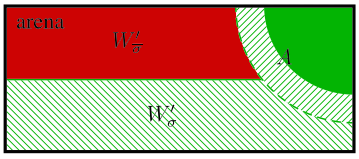
Lemma 3.1. *For an arena $\mathcal{A}$ and a set T of target positions, the σ-attractor of T can be constructed in time linear in the edges of A.*



In the next step, the co-game $\mathcal{P}' = \mathcal{P} \setminus A$ of $\mathcal{P}$ is solved. The co-set $C = V \setminus A$ of the σ-attractor A for some target set T is called a *σ-trap*, because Player σ cannot leave C; he is trapped there.

The co-game $\mathcal{P}'$ is smaller than $\mathcal{P}$: compared to $\mathcal{P}$, it contains less positions. By induction hypothesis, it is therefore memoryless determined.

By induction over the size of the game, $\mathcal{P}'$ can therefore be solved by a recursive call of the algorithm.



We call a subset $P_\sigma \subseteq W_\sigma$ of a winning region of Player $\sigma \in \{0,1\}$ a *σ-paradise* if it is a $\overline{\sigma}$-trap and Player σ has a memoryless strategy f that is p-winning for all $p \in P_\sigma$ in $\mathcal{P} \cap P_\sigma$. That is, if Player σ has a winning strategy, such that P_σ cannot be left in any f-conform play ($E_f \cap P_\sigma \times V \setminus P_\sigma = \emptyset$).

Lemma 3.2. *[17,8,29] For a parity game $\mathcal{P}$ with σ-trap T_σ, and a $\overline{\sigma}$-paradise $P_{\overline{\sigma}}$ of $\mathcal{P}' = \mathcal{P} \cap T_\sigma$, $P_{\overline{\sigma}}$ is a $\overline{\sigma}$-paradise for $\mathcal{P}$.*

In fact, Player $\overline{\sigma}$ can simply use the same winning strategy f for $\mathcal{P}$ as for $\mathcal{P}'$: as T_σ is a σ-trap, Player σ has no additional moves in $\mathcal{P}$, and every f conform play that starts in $P_{\overline{\sigma}}$ in $\mathcal{P}$ is also an f conform play in $\mathcal{P}'$.

In particular, the winning region $W'_{\overline{\sigma}}$ of $\mathcal{P}'$ is a $\overline{\sigma}$-paradise in $\mathcal{P}$ by construction. So is its $\overline{\sigma}$-attractor in $\mathcal{P}$.

Lemma 3.3. *[17,8,29] The σ-attractor A_σ of a σ-paradise P_σ for a parity game $\mathcal{P}$ is a σ-paradise for $\mathcal{P}$, and a winning strategy for player σ on A_σ can be composed of the winning strategy for Player σ on P_σ and an attractor strategy on $A_\sigma \setminus P_\sigma$.*

For a given σ-paradise P_σ for Player $\sigma \in \{0,1\}$ in a parity game $\mathcal{P}$, we can reduce solving $\mathcal{P}$ to computing the σ-attractor A_σ of P_σ, and solving $\mathcal{P} \setminus A_\sigma$.

Lemma 3.4. *[17,8,29] Let $\mathcal{P}$ be a parity game, P_σ be a σ-paradise with σ-attractor A_σ, and let W'_σ and $W'_{\overline{\sigma}}$ be the winning regions of Player σ and Player $\overline{\sigma}$, respectively, on $\mathcal{P}' = \mathcal{P} \setminus A_\sigma$. Then*

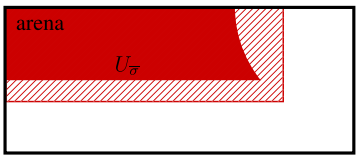
- *$W_{\overline{\sigma}} = W'_{\overline{\sigma}}$ is the winning region of Player $\overline{\sigma}$ on P, and she can win by following her winning strategy from $\mathcal{P}'$ on her winning region, and*
- *$W_\sigma = W'_\sigma \cup A$ is the winning region of Player σ and he can win by following his winning strategy for A (see Lemma 3.3) on A_σ and his winning strategy from $\mathcal{P}'$ on W'_σ.*

Proof. First, Player $\overline{\sigma}$ can use her winning strategy for her winning region $W_{\overline{\sigma}}$ of $\mathcal{P} \setminus A_\sigma$, and use it in the larger game $\mathcal{P}$, because Player σ has no additional choices in $W_{\overline{\sigma}}$ in $\mathcal{P}$. Consequently, the set of $g_{\overline{\sigma}}$-conform plays in $\mathcal{P}$ starting in $W_{\overline{\sigma}}$ coincides with the set of $f_{\overline{\sigma}}$-conform plays in $\mathcal{P} \setminus A_\sigma$ starting in $W_{\overline{\sigma}}$.

For the same reason, Player σ wins with his strategy from every position in A_σ, by a composition on the attractor strategy on $A_\sigma \setminus P_\sigma$ and her winning strategy on P_σ, see Lemma 3.3.

Let g_σ be a winning strategy for player σ in $\mathcal{P}'$. Every g_σ-conform play in $\mathcal{P}$ starting in a position not in W'_σ either eventually reaches A_σ, and is then followed by a tail (remainder of the play) in $\mathcal{P}$ that starts in A_σ, which is winning for σ by Lemma 3.3, or stays for ever in the sub-game $\mathcal{P}'$, and is thus winning for Player σ, too. □

We now distinguish two cases: Firstly, if $W'_{\overline{\sigma}}$ is non-empty, we can reduce solving $\mathcal{P}$ to constructing the $\overline{\sigma}$-attractor $U_{\overline{\sigma}}$ of $W'_{\overline{\sigma}}$, and solving the co-game $\mathcal{P}'' = \mathcal{P} \setminus U_{\overline{\sigma}}$ by Lemma 3.4.



The co-game $\mathcal{P}''$ is simpler than $\mathcal{P}$: Compared to $\mathcal{P}$, it contains less positions (though not necessarily less colours). By induction over the size of the game, $\mathcal{P}''$ can therefore be solved by a recursive call of the algorithm.

Secondly, if $W'_{\overline{\sigma}}$ is empty, we can compose the winning strategy for Player σ on $\mathcal{P}'$ with his attractor strategy for the σ-attractor of the target set $\alpha^{-1}(c)$ to a winning strategy on $\mathcal{P}$.

Lemma 3.5. *[17,8,29] Let $\mathcal{P}$ be a parity game with maximal colour c, let $\sigma = c \bmod 2$ be the player who wins if c occurs infinitely many times, let A be the σ-attractor of $\alpha^{-1}(c)$ and let f be an attractor strategy for Player σ on her positions on $A \setminus \alpha^{-1}(c)$. If Player σ has a winning strategy f' for every position in $\mathcal{P}' = \mathcal{P} \setminus A$, then f and f' can be composed to a winning strategy for Player σ for every position in $\mathcal{P}$.*

Proof. Let g be a strategy for Player σ that agrees with f and f' on their respective domain. We distinguish two types of g-conform plays: those that eventually stay in $\mathcal{P}'$, and those that visit A infinitely often. The latter plays contain infinitely many c-coloured positions and are therefore winning for player σ. Games that eventually stay in $\mathcal{P}'$ consist of a finite prefix, followed by an f'-conform play in $\mathcal{P}'$. The highest colour occurring infinitely often is therefore even for $\sigma = 0$ and odd for $\sigma = 1$, respectively. □

Theorem 3.6. *[17,8,29] For every parity game $\mathcal{P} = (V_0, V_1, E, \alpha)$, the game positions are partitioned into a winning region W_0 of Player 0 and a winning region W_1 of Player 1. Moreover, Player 0 and Player 1 have memoryless strategies that are p-winning for every position p in their respective winning region.*

Proof. The starting point of the inductive argument are games with a single position. They are trivially won by the player that wins on the colour of this position (induction basis).

For the induction step, assume that the memoryless determinacy holds for games with up to n positions. For a parity game with $n+1$ positions, we can then select the highest colour $c_{\max}$, set σ to $c_{\max} \bmod 2$ to identify the Player σ who wins if $c_{\max}$ occurs infinitely often (note that $c_{\max}$ is the dominating colour in this case), and set $A = \sigma\text{-}Attractor(\alpha^{-1}(c_{\max}), \mathcal{P})$, where α^{-1} is the pseudo inverse of α.

Then $\mathcal{P}' = \mathcal{P} \setminus A$ is a—possibly empty—parity game with strictly less positions and colours. (Note that, by the attractor construction, every position in $\mathcal{P}'$ has a successor, and the co-set of A is a σ-trap.)

By our induction hypothesis, the positions in $\mathcal{P}'$ are partitioned into winning regions of the two players, and both players have memoryless winning strategies on their winning regions.

We can now distinguish two cases:

1. The winning region of Player $\overline{\sigma}$ on $\mathcal{P}'$ is empty. In this case, Player σ wins memoryless by Lemma 3.5.
2. The winning region of Player $\overline{\sigma}$ is non-empty.
   Then $W''_{\overline{\sigma}} = \overline{\sigma}\text{-}Attractor(W'_{\overline{\sigma}}, \mathcal{P})$ is a $\overline{\sigma}$ paradise for $\mathcal{P}$ by Lemmata 3.2 and 3.3. We can therefore solve the remainder of the game, $\mathcal{P} \setminus W''_{\overline{\sigma}}$, individually and use the respective winning regions and (by induction, memoryless winning strategies) of the players by Lemma 3.4.

In Case (1) we are done and in Case (2) we have reduced the problem to solving a game with less positions. By induction, memoryless determinacy extends to the complete game. □

The worst case running time of McNaughton's algorithm [17,8,29] (cf. Procedure *McNaughton* of Fig. 1) occurs if $U_{\overline{\sigma}}$, the $\overline{\sigma}$-attractor of the winning $W_{\overline{\sigma}}$' of $\mathcal{P} \setminus A$, always has a small intersection with A and contains exactly one position with maximal colour c.

For parity games with c colours, McNaughton's algorithm requires $O\left(m \cdot \left(\frac{n}{c} + 1\right)^{c-1}\right)$ steps for games with n positions and m edges. It can be extended to also return the winning strategies for both players on their complete winning region.

4. Progress measures

An alternative and structurally different approach is due to Jurdziński [11]. In his algorithm, the progress of Player 0 towards proving that she can force the highest colour to be even (or Player 1 towards proving that he can force the highest colour to be odd) is intuitively measured by a vector that represents the worst possible future.

We start by generalising his approach by using coarser progress measures. Using coarser progress measures leads to an underapproximation of the winning region of one player, and we will use this underapproximation of a winning region in the following section.

While we have to re-prove the results of Jurdziński [11] for the more general case, the structure of the proofs is very similar to the original ones. We close this section by looking at the special case of three colour games, which forms the base case of the algorithm proposed in the following section.

4.1. Progress measures

For a parity game $\mathcal{P} = (V_0, V_1, E, \alpha)$ with maximal colour d, the *maximal σ-progress measure* is, for $\sigma \in \{0, 1\}$, a function $\varrho : V_0 \uplus V_1 \to \mathcal{M}^\sigma$ whose co-domain

$$\mathcal{M}^\sigma_\infty = \{h : \{0, \dots, d\} \to \mathbb{N} \mid h(c) = 0 \text{ if } c \bmod 2 = \sigma, \text{ and } h(c) \leq |\alpha^{-1}(c)| \text{ otherwise}\} \cup \{\top\}$$

contains a maximal element $\top$ and a set of functions from $\{0, \dots, d\}$ to the integers. The co-domain $\mathcal{M}^\sigma_\infty$ satisfies the requirement that

- every integer $i \leq d$ is mapped to 0 if $i \bmod 2 = \sigma$, while
- all other integers i are mapped to a value bounded by the number $|\alpha^{-1}(i)|$ of i-coloured game positions.

h is often considered as a tuple. A *σ-progress measure*

$$\mathcal{M}^\sigma \subseteq \mathcal{M}^\sigma_\infty$$

is a downward closed subset of $\mathcal{M}^\sigma_\infty$ that contains the maximal element ($\top \in \mathcal{M}^\sigma$). Downward closedness means that, if $\mathcal{M}^\sigma$ contains a function $h \in \mathcal{M}^\sigma$, then every function $h' \in \mathcal{M}^\sigma_\infty$ that is point-wise smaller than h ($h'(i) \leq h(i)\ \forall i \leq d$) is also contained in $\mathcal{M}^\sigma$.

4.2. Linear pre-orders on M^σ

For each colour $c \leq d$, we define a relation $\rhd_c \subseteq \mathcal{M}^\sigma \times \mathcal{M}^\sigma$, which is essentially the lexicographic order, ignoring all colours smaller than c. $\rhd_c$ is defined as the smallest relation

- that contains $\{\top\} \times \mathcal{M}^\sigma$ and
- that contains a pair of functions $(h, h') \in \rhd_c$ if
  - there is a colour $c' \geq c$ such that $h(c') > h'(c')$, and $h(c'') = h'(c'')$ holds for all colours $c'' > c'$, or
  - $c \bmod 2 = \sigma$, and $h(c') = h'(c')$ holds for all $c' \geq c$.

$\rhd_0$ defines an order on $\mathcal{M}^\sigma$ — the lexicographic order when h is read as a tuple, where higher colours have higher priority. $\rhd_c$ defines a linear pre-order — the lexicographic order when h is read as a tuple but cut off after colour c.

4.3. Pre-order on progress measures

From this order, we infer the linear pre-order $\sqsubseteq$ on progress measures, which requires that $\rhd_0$ is satisfied on every position of the game ($\varrho \sqsubseteq \varrho' \Leftrightarrow \forall p \in V . \varrho(p) \rhd_0 \varrho'(p)$).

We call a σ-progress measure ϱ *valid* if

- every position $p \in V_\sigma$ has some successor $p' \in V$ with $\varrho(p) \rhd_{\alpha(p)} \varrho(p')$, and
- for every position $p \in V_{\overline{\sigma}}$ and every successor $p' \in V$ of p, $\varrho(p) \rhd_{\alpha(p)} \varrho(p')$ holds.

Progress measures are ranking functions that can intuitively be used to estimate the worst-case future occurrence of 'bad' positions prior to positions with higher colour. A valid σ-progress measure that is not constantly $\top$ can be used to partly evaluate a parity game. Let, for a σ-progress measure ϱ, $\mathsf{win}(\varrho) = V \setminus \varrho^{-1}(\top)$ denote the game positions that are not mapped to the maximal element $\top$ of $\mathcal{M}^\sigma$.

Theorem 4.1. *[11] Let $\mathcal{P} = (V_0, V_1, E, \alpha)$ be a parity game with valid σ-progress measure ϱ. Then Player σ wins on $\mathsf{win}(\varrho)$ with any memoryless winning strategy that maps a position $p \in \mathsf{win}(\varrho) \cap V_\sigma$ to a position p' with $\varrho(p) \rhd_{\alpha(p)} \varrho(p')$.*

Such a successor must exist, since the progress measure is valid. The $\sqsubseteq$-*least* valid σ-progress measure is well defined and can be computed efficiently for small $\mathcal{M}^\sigma$.

Theorem 4.2. *The $\sqsubseteq$-least valid σ-progress measure ϱ_μ exists and can, for a parity game with m edges and c colours, be computed in time $O(c\,m\,|\mathcal{M}^\sigma|)$.*

The proof is very similar to the proof of a similar claim for the maximal co-domain $\mathcal{M}^\sigma_\infty$ in Jurdziński's work [11]. We first introduce some notation.

For a given progress measure ϱ_i, we call an edge (p, p') a *lift-edge* if $\varrho_i(p) \not\rhd_{\alpha(p)} \varrho(p')$. We call a position $p \in V_\sigma$ of Player 0 *liftable* if all outgoing edges are lift edges, and we call a position $p \in V_1$ of Player 1 liftable if some outgoing edge is a lift edge.

We *lift* a liftable position by applying the following local update:

- at some liftable position $p \in V_\sigma$ where the validity criterion is locally violated to $\varrho_{i+1}(p) = \min\{\varrho \in \mathcal{M}^\sigma \mid \exists (p, p') \in E.\ \varrho \rhd_{\alpha(p)} \varrho_i(p')\}$, or
- at some liftable position $p \in V_{\bar\sigma}$ where the validity criterion is locally violated to $\varrho_{i+1}(p) = \min\{\varrho \in \mathcal{M}^\sigma \mid \forall (p, p') \in E.\ \varrho \rhd_{\alpha(p)} \varrho_i(p')\}$.

and $\varrho_{i+1}(q) = \varrho_i(q)$ for all positions $q \neq p$.

Proof. First, it is easy to see that the position-wise minimum of two valid σ-progress measures forms a valid σ-progress measure. With the finite domain, this implies that ϱ_μ is well defined as the position-wise minimum over all valid σ-progress measures.

To compute it, we can start with an arbitrary σ-progress measure smaller than ϱ_μ – in particular, with the progress measure ϱ_0 that assigns the constant function to 0 to all positions. While ϱ_i is not valid, we update it to ϱ_{i+1} by updating the function locally, using a lift operation.

Obviously, the update is still smaller or equal to ϱ_μ. For all $q \in V$, $\varrho_\mu(q) \rhd_0 \varrho_i(q)$, we get for the lifted position p:

$$\begin{aligned}
\varrho_{i+1}(p) &= \min\{\varrho \in \mathcal{M}^\sigma \mid \exists (p, p') \in E.\ \varrho \rhd_{\alpha(p)} \varrho_i(p')\} \\
&\rhd_0 \min\{\varrho \in \mathcal{M}^\sigma \mid \exists (p, p') \in E.\ \varrho \rhd_{\alpha(p)} \varrho_\mu(p')\} \\
&\rhd_0 \varrho_\mu(p) \text{ if } p \in V_\sigma \text{ and} \\
\varrho_{i+1}(p) &= \min\{\varrho \in \mathcal{M}^\sigma \mid \forall (p, p') \in E.\ \varrho \rhd_{\alpha(p)} \varrho_i(p')\} \\
&\rhd_0 \min\{\varrho \in \mathcal{M}^\sigma \mid \forall (p, p') \in E.\ \varrho \rhd_{\alpha(p)} \varrho_\mu(p')\} \\
&\rhd_0 \varrho_\mu(p) \text{ if } p \in V_{\bar\sigma}.
\end{aligned}$$

As ϱ_μ is valid, this implies $\varrho_{i+1} \sqsubseteq \varrho_\mu$.

The finiteness of the domain guarantees termination. □

When using the maximal co-domain $\mathcal{M}^\sigma_\infty$, which contains the function ϱ that assigns each colour c with $c \bmod 2 \neq \sigma$ to $\varrho(c) = |\alpha^{-1}(c)|$, for the progress measures, the $\sqsubseteq$-least valid σ-progress measure ϱ_μ determines the *complete* winning region of Player σ.

Theorem 4.3. *[11] For a parity game $\mathcal{P} = (V_0, V_1, E, \alpha)$ and for the co-domain $\mathcal{M}^\sigma_\infty$ for the progress measures, $\mathsf{win}(\varrho_\mu)$ coincides with the winning region W_σ of Player σ for the $\sqsubseteq$-least valid σ-progress measure ϱ_μ.*

4.4. σ/k-paradise

Instead of using this technique to solve the parity game, we will use the algorithm to construct a particular type of paradises, which we call σ/k-paradises.

Definition 4.4 *(σ/k-Paradise).* We call a σ-paradise P^k_σ a *σ/k-paradise* if it contains all σ-paradises of size $\leq k$.

The efficient construction of σ/k paradises is an essential ingredient in the algorithm discussed in the following sections. For their construction, we draw from the efficient computation of the $\sqsubseteq$-least valid σ-progress measure (Theorem 4.2).

Instead of using the maximal co-domain $\mathcal{M}^\sigma_\infty$, the smaller co-domain $\mathcal{M}^\sigma_k$ is used for the progress measures, which contains only those functions h that satisfy $\sum_{c=0}^{d} h(c) \leq k$ for some parameter $k \in \mathbb{N}$. (d denotes the highest colour of the parity game). The size of $\mathcal{M}^\sigma_k$ can be estimated by

$$\left|\mathcal{M}^\sigma_k\right| \leq \binom{k + \lceil 0.5(d+1)\rceil}{k} + 1.$$

Using $\mathcal{M}^\sigma_k$ instead of $\mathcal{M}^\sigma_\infty$, $\mathsf{win}(\varrho_\mu)$ contains all σ-paradises of size $\leq k+1$ (where ϱ_μ denotes the $\sqsubseteq$-least valid σ-progress measures).

Theorem 4.5. *Let $\mathcal{P} = (V_0, V_1, E, \alpha)$ be a parity game, and let $P_\sigma \subseteq V$ be a σ-paradise of size $|P_\sigma| \leq k+1$. Then there is a valid σ-progress measure $\varrho : V \to \mathcal{M}_k^\sigma$ with $P_\sigma = \mathsf{win}(\varrho)$.*

Proof. Since P_σ is a σ-paradise, E and $V_{\overline{\sigma}} \cap P_\sigma \times V \setminus P_\sigma$ are disjoint, and Player σ can stay in P_σ. Moreover, Player σ has a memoryless strategy f that is winning on every game position in P_σ such that $f(p) \in P_\sigma$ for all $p \in V_\sigma \cap P_\sigma$.

If we restrict $\mathcal{P}$ to $\mathcal{P}' = \mathcal{P}_f \cap P_\sigma$, then the winning region of Player σ must therefore cover the whole set P_σ of game positions of $\mathcal{P}'$.

To solve $\mathcal{P}'$, we can use the maximal co-domain $\mathcal{M}_\infty^{\sigma\,\prime}$. By Theorem 4.3, the $\sqsubseteq'$-least progress measure ϱ'_μ for this co-domain satisfies $\mathsf{win}(\varrho'_\mu) = P_\sigma$. Since $\mathcal{M}_\infty^{\sigma\,\prime} \subseteq \mathcal{M}_k^\sigma$ is contained in $\mathcal{M}_k^\sigma$ (P_σ must contain at least one position with even colour if $\sigma = 0$, resp. one position with odd colour if $\sigma = 1$), we can extend ϱ'_μ to a valid σ-progress measure ϱ on $\mathcal{P}$ by setting $\varrho(p) = \varrho'_\mu(p)$ for all $p \in P_\sigma$, and $\varrho(p) = \top$ otherwise. □

By Theorem 4.2, we can compute the $\sqsubseteq$-least valid σ-progress measure ϱ_μ in time $O(cm|\mathcal{M}_k^\sigma|)$, and, by Theorem 4.1, we can construct a winning strategy for Player σ on $\mathsf{win}(\varrho_\mu)$ within the same complexity bound.

Corollary 4.6. *For a given parity game $\mathcal{P}$ with c colours and m edges, we can construct a $\sigma/(k+1)$-paradise P_σ^{k+1} for Player σ in time $O\left(cm\binom{k+\lceil 0.5c\rceil}{k}\right)$. A winning strategy for Player σ on P_σ^{k+1} can be constructed within the same complexity bound.*

4.5. Three colour games

When using Jurdziński's algorithm [11] for solving parity games with c colours, the size $|\mathcal{M}_\infty^\sigma|$ of the maximal co-domain can be estimated by $(\frac{n}{\lceil 0.5c\rceil})^{\lfloor 0.5c\rfloor}+1$ if $\sigma = 0$, and by $(\frac{n}{\lceil 0.5c\rceil})^{\lceil 0.5c\rceil}+1$ if $\sigma = 1$. From Theorem 4.3 we therefore get the well established complexity for finding the winning regions of and the winning strategy for *one* of the players in three colour games.

Corollary 4.7. *[11] Parity games with maximal colour 2 can be solved and a winning strategy for Player 0 can be constructed in time $O(mn)$.*

The algorithm described in the previous subsection provides a partition of the winning regions and a winning strategy for Player σ, but not for a winning strategy of Player $\overline{\sigma}$. In principle, her winning strategy can be computed using a $\overline{\sigma}$-progress measure, but, for games with an odd number of colours, this is slightly more expensive. We dedicate this subsection to the special case of three colour games, because they play a role as a base case for the algorithm discussed in the following section.

We call parity games with maximal colour 2 *three colour games*. Corollary 4.7 shows that a non-constructive solution for three colour games as well as a winning strategy for Player 0 can be obtained in time $O(mn)$. To see why Jurdziński's algorithm [11] does not provide a strategy for Player 1, let us summarise his algorithm for the simple case of a three colour games $\mathcal{P} = (V_0, V_1, E, \alpha)$.

For three colour games, the 0-progress measures can be viewed as mappings $\varrho : V \to \{0, \ldots, n_1\} \cup \{\top\}$, where $n_1 = |\alpha^{-1}(1)|$ denotes the number of 1-coloured positions.

The starting point of the algorithm is the trivial progress measure ϱ_0 that maps all positions of $\mathcal{P}$ to 0. Starting from ϱ_0, we *lift* the progress measure stepwise at a liftable position $p \in V$ until a fixed point is reached.

For the trivial progress measure ϱ_0, an edge is a lift-edge if, and only if, it originates from a 1-coloured position, and a position is liftable if, and only if, it is 1-coloured. For an efficient implementation, it suffices to attach a flag to every edge that indicates whether this edge is a lift-edge, to keep track of the number of outgoing lift-edges for every game position, and to keep the liftable positions in a doubly linked list.

In order to lift ϱ_i, any liftable position p can be taken from the list of liftable positions. (If no liftable position remains, the least fixed point is reached.) After lifting ϱ_i at position p, it suffices to check for each incoming and outgoing edge of p if the flag that indicates liftability needs to be adjusted and, if so, to increase the number of outgoing lift-edges for the respective predecessor of p (for incoming edges), or to decrease the number of outgoing lift-edges for p (for outgoing edges), respectively. If a position becomes liftable (non-liftable), it is added to (removed from) the list of liftable positions.

While this algorithm provides good complexity bounds for the non-constructive analysis of three colour games, it does not provide a winning strategy for player 1 on her winning region. Note that the naive extension – fixing the edge used for the last update as strategy for player one – is not sound: Fig. 2 shows a small example of a single player Büchi game (Büchi games are games with only the colours 1 and 2), where all positions are positions of Player 1 ($V_0 = \emptyset$). The positions a, b, and c are coloured by 1, while position d is coloured by 2. Player 1 can choose a self-loop at position a (in which case she wins), or move in a Hamiltonian cycle (in which case she loses). If we start with twice lifting at position a ($\varrho_1(a) = 1$, $\varrho_2(a) = 2$) followed by lifting at position b ($\varrho_3(b) = 3$), c ($\varrho_4(c) = \top$), d ($\varrho_5(d) = \top$), and again at a ($\varrho_6(a) = \top$) and b ($\varrho_7(b) = \top$), all positions are correctly marked as winning for Player 1; but the last update of position a relies on $\varrho_5(d) = \top$, and the naive approach would result in a losing strategy.

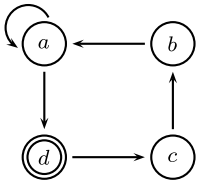


Fig. 2. The example shows a single player Büchi game (that is, a game where all positions are coloured by 1 or 2), where all positions belong to Player 1 ($V_0 = \emptyset$). The positions a, b, and c are coloured by 1, while position d is coloured by 2 (indicated by the double line).

We show that a variant of the algorithm can be used to also construct a winning strategy of Player 1 on her complete winning region. It suffices to store intermediate strategies for Player 1, and to keep two sets of liftable positions instead of one – one set for positions that are liftable without changing the intermediate strategy of Player 1, and one set of positions that are liftable, but only if the strategy of Player 1 is changed. The adapted algorithm always gives preference to liftable positions from the first set. If only liftable positions from the latter set remain, one of these positions is lifted and the intermediate strategy is updated accordingly.

In the single player game from the example of Fig. 2, we can either start with the self-loop at position a and thus with a winning strategy, or with the losing strategy to move from a to d. In the first case, we never have to adjust the strategy. (One possible sequence of progress measure updates is $\varrho_1(a) = 1$, $\varrho_2(a) = 2$, $\varrho_3(a) = 3$, $\varrho_4(a) = \top$, $\varrho_5(b) = \top$, $\varrho_6(c) = \top$, $\varrho_7(d) = \top$.) In the latter case, we first compute the fixed point for the single player game, where the moves of Player 1 are restricted by her strategy. (One possible sequence of progress measure updates is $\varrho_1(a) = 1$, $\varrho_2(b) = 2$, $\varrho_3(c) = 3$.) Once the fixed point for this strategy is reached, the strategy is adjusted by choosing the self-loop at position a. (One possible sequence of further progress measure updates is $\varrho_4(a) = 2$, $\varrho_5(a) = 3$, $\varrho_6(a) = \top$, $\varrho_7(b) = \top$, $\varrho_8(c) = \top$, $\varrho_9(d) = \top$.)

Theorem 4.8. *For parity games with maximal colour 2, the proposed algorithm can be used to solve the parity game and to construct winning strategies for both players on their respective winning region in time $O(mn)$.*

Proof. The proposed changes to Jurdziński's algorithm only impose a particular order on the lifting operations, which could coincidentally occur in his algorithm, too. This implies the correctness of the least fixed point and thus the correctness of the resulting winning regions and strategy of Player 0 (cf. Corollary 4.7).

For the correctness of the winning strategy of Player 1 on her winning region, we show by induction that every time the intermediate strategy needs to be changed, say from f to f', the intermediate progress measure ϱ_μ^f is the $\sqsubseteq$-least valid 0-progress measure ϱ_μ^f for $\mathcal{P}_f$.

> Induction Basis: For any initial strategy f the claim holds trivially – up to the first adjustment of the intermediate strategy the algorithm resembles the original algorithm for $\mathcal{P}_f$.
> Induction Step: Consider the situation after changing the intermediate strategy from f to f' by choosing a lift-edge (p, p'). Let us compare the $\sqsubseteq$-least valid 0-progress measure ϱ_μ^f for $\mathcal{P}_f$ with the $\sqsubseteq$-least valid 0-progress measure $\varrho_\mu^{f'}$ for $\mathcal{P}_{f'}$.

We first show $\varrho_\mu^f(p) \neq \varrho_\mu^{f'}(p)$. To see this, we develop the $\sqsubseteq$-least valid 0-progress measure $\varrho_\mu^{f'}$ for $\mathcal{P}_{f'}$ from the trivial progress measure ϱ_0, where we apply an update at position p *only*, if no update at any other position is possible.

Let $\varrho_0, \varrho_1, \varrho_2, \varrho_3, \ldots$ be the sequence of progress measures constructed this way, where $\varrho_\mu^{f'}$ is the limit. Note that $\varrho_0 \sqsubseteq \varrho_1 \sqsubseteq \varrho_2 \sqsubseteq \varrho_3 \sqsubseteq \ldots \sqsubseteq \varrho_\mu^{f'}$ and $\varrho_0(p) \leq \varrho_1(p) \leq \varrho_2(p) \leq \varrho_3(p) \leq \ldots \leq \varrho_\mu^{f'}(p)$ hold.

We show by induction that $\varrho_\mu^{f'}(p) \leq \varrho_\mu^f(p)$ implies $\varrho_i \sqsubseteq \varrho_\mu^f(p)$ for all $i \in \omega$. Let us assume $\varrho_\mu^{f'}(p) \leq \varrho_\mu^f(p)$.

Induction Basis: $\varrho_0 \sqsubseteq \varrho_\mu^f$ trivially holds.

Induction Step: We distinguish two cases. First, if position p is lifted, we have that $\varrho_{i+1}(p) \leq \varrho_\mu^{f'}(p)$, which is $\leq \varrho_\mu^f(p)$ by assumption. For all other positions $q \in V$ with $q \neq p$, we have $\varrho_i(q) \leq \varrho_\mu^f(q)$ (by induction hypothesis) and $\varrho_{i+1}(q) = \varrho_i(q)$, which implies $\varrho_{i+1}(q) \leq \varrho_\mu^f(q)$.

Second, if position $q \neq p$ is lifted, we first observe that q has the same successors in $\mathcal{P}_f$ and $\mathcal{P}_{f'}$. $\varrho_i \sqsubseteq \varrho_\mu^f$ implies for all successors q' of q that $\varrho_i(q') \leq \varrho_\mu^f(q')$ holds. Taking into account that ϱ_μ^f is a valid 0-progress measure, this implies $\varrho_{i+1}(q) \leq \varrho_\mu^f(q)$.

For all other positions $q'' \in V$ with $q'' \neq q$, we have $\varrho_i(q'') \leq \varrho_\mu^f(q'')$ (by induction hypothesis) and $\varrho_{i+1}(q'') = \varrho_i(q'')$, which implies $\varrho_{i+1}(q'') \leq \varrho_\mu^f(q'')$.

Procedure Winning-Regions($\mathcal{P}$):
1. set d to the highest colour occurring in $\mathcal{P}$
2. if $d \leq 2$ then return *ThreeColour*($\mathcal{P}$)
3. set σ to d mod 2
4. set n to the size $|V|$ of $\mathcal{P}$
5. set $W_{\overline{\sigma}}$ to $\emptyset$
6. repeat
   (a) set $W'_{\overline{\sigma}}$ to $\overline{\sigma}$-*Attractor*(*Approximate*($\mathcal{P}$, par(n,d), $\overline{\sigma}$), $\mathcal{P}$)
   (b) set $W_{\overline{\sigma}}$ to $W_{\overline{\sigma}} \cup W'_{\overline{\sigma}}$
   (c) set $\mathcal{P}$ to $\mathcal{P} \setminus W'_{\overline{\sigma}}$
   (d) set $\mathcal{P}'$ to $\mathcal{P} \setminus \sigma$-*Attractor*($\alpha^{-1}(d)$, $\mathcal{P}$)
   (e) set (W'_0, W'_1) to *Winning-Regions*($\mathcal{P}'$)
   (f) if $W'_{\overline{\sigma}} = \emptyset$ then
      i. set W_σ to $V \setminus W_{\overline{\sigma}}$
      ii. return (W_0, W_1)
   (g) set $W_{\overline{\sigma}}$ to $W_{\overline{\sigma}} \cup \overline{\sigma}$-*Attractor*($W'_{\overline{\sigma}}$, $\mathcal{P}$)
   (h) set $\mathcal{P}$ to $\mathcal{P} \setminus \overline{\sigma}$-*Attractor*($W'_{\overline{\sigma}}$, $\mathcal{P}$)

Fig. 3. The Procedure *Winning-Regions*($\mathcal{P}$) takes a parity game $\mathcal{P}$ as input and returns the ordered pair (W_0, W_1) of winning regions for Player 0 and Player 1, respectively. V and α denote the game positions and the colouring function of the parity game $\mathcal{P}$. *ThreeColour*($\mathcal{P}$) solves a three colour game $\mathcal{P}$ (c.f. Theorem 4.8), *Approximate*($\mathcal{P}$, par, σ) computes a $\sigma/(\text{par}+1)$-paradise (c.f. Corollary 4.6), and σ-*Attractor*(F, $\mathcal{P}$) computes the σ-attractor of a set F of game positions in a game $\mathcal{P}$ (c.f. Lemma 3.1).

As $\varrho_\mu^{f'}$ is the limit of these progress measures, we get $\varrho_\mu^{f'} \sqsubseteq \varrho_\mu^f$.

Similarly, we can establish that $\varrho_\mu^f(p) \leq \varrho_\mu^{f'}(p)$ implies $\varrho_\mu^f \sqsubseteq \varrho_\mu^{f'}$.

Note that both directions together show that $\varrho_\mu^f(p) = \varrho_\mu^{f'}(p)$ implies $\varrho_\mu^f = \varrho_\mu^{f'}$, which contradicts the assumption that (p, p') was a lift-edge.

It also shows that $\varrho_\mu^{f'}(p) > \varrho_\mu^f(p)$ implies that the next switch in strategy takes place when $\varrho_\mu^{f'}(p)$ is reached. (Unless $\varrho_\mu^{f'}(p)$ is also the $\sqsubseteq$-least valid 0-progress measure for $\mathcal{P}$, in which case the procedure terminates there.)

What remains is to exclude $\varrho_\mu^f(p) > \varrho_\mu^{f'}(p)$. We first observe that $\varrho_\mu^f(p) \neq \bot$, because (p, p') would not be a lift-edge in this case. Let us now assume for contradiction that $\delta = \varrho_\mu^f(p) - \varrho_\mu^{f'}(p) > 0$.

We now re-calculate the 0-progress measure ϱ_μ^f for $\mathcal{P}_f$ from the trivial progress measure ϱ_0.

Let ϱ_0, ϱ_1, ϱ_2, ϱ_3, ... be the sequence of progress measures constructed this way, where ϱ_μ^f is the limit. We show by induction that, for all ϱ_i, we have $\varrho_i(q) \leq \varrho_\mu^{f'}(q) + \delta$ for all positions $q \in V$.

Induction Basis: For ϱ_0, this is implied by $\varrho_0 \sqsubseteq \varrho_\mu^{f'}$.

Induction Step: We distinguish two cases. First, if position p is lifted, we have that $\varrho_{i+1}(p) \leq \varrho_\mu^f(p)$ because ϱ_μ^f is the limit of the sequence of progress measures, and we have $\varrho_\mu^f(p) = \varrho_\mu^{f'}(p) + \delta$ by assumption. For all other positions $q \in V$ with $q \neq p$, we have $\varrho_{i+1}(q) = \varrho_i(q)$, which implies $\varrho_{i+1}(q) \leq \varrho_\mu^{f'}(q) + \delta$ with the induction hypothesis.

Second, if a position $q \neq p$ is lifted, we first observe that q has the same successors in $\mathcal{P}_f$ and $\mathcal{P}_{f'}$. For each successor $q' \in V$ it holds that $\varrho_i(q') \leq \varrho_\mu^{f'}(q') + \delta$. As $\varrho_\mu^{f'}$ is valid, these inequations imply with the equal set of successors $\varrho_{i+1}(q) \leq \varrho_\mu^{f'}(q) + \delta$ by the lifting rules.

For all other positions $q'' \in V$ with $q'' \neq q$, we have $\varrho_i(q'') \leq \varrho_\mu^{f'}(q'') + \delta$ (by induction hypothesis) and $\varrho_{i+1}(q'') = \varrho_i(q'')$, which implies $\varrho_{i+1}(q'') \leq \varrho_\mu^{f'}(q'') + \delta$.

This implies in particular $\varrho_\mu^f(p') \leq \varrho_\mu^{f'}(p') + \delta$. This contradicts the assumption, that (p, p') was a lift edge. □

5. Big steps

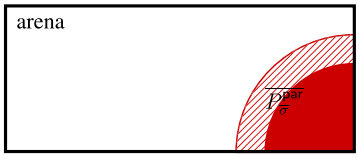
As observed by Jurdziński, Paterson, and Zwick [12], the draw-back of McNaughton's algorithm is the potentially small change that occurs in every recursive call: Each recursive call provides a paradise for the player who loses on the highest colour, and if the attractor of the paradise includes one (or, more generally, few) positions with maximal colour, many iterations are needed. This can be changed by coupling it with an alternative way to compute σ/k-paradises for this player, where $k = \text{par}(n, d)$ is set to a parameter par that may depend on the number of positions and the highest occurring colour.

Fig. 3 provides an overview on the proposed algorithm. The input to the algorithm is a parity game $\mathcal{P}$, and the output is the ordered pair consisting of the winning regions for the players.

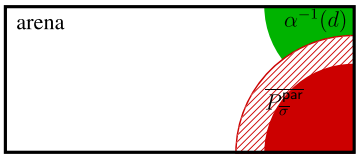
The algorithm first determines the highest colour d of $\mathcal{P}$ (line 1). In line 2, three colour games are covered, that is, games with highest colour ≤ 2. Such games are solved using the constructive algorithm discussed in Subsection 4.5. For games with a higher maximal colour than 2, the algorithm proceeds by determining the Player $\sigma = d \bmod 2$ that wins if the highest colour d occurs infinitely often (line 3).

In every iteration of the repeat loop, the proposed big step algorithm (Fig. 3) first constructs a $\overline{\sigma}/(\mathsf{par}+1)$-paradise (cf. Subsection 4.4) for an appropriate parameter par.

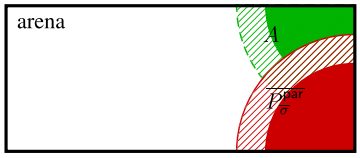
By Lemma 3.4, we can now reduce solving $\mathcal{P}$ to constructing the $\overline{\sigma}$-attractor $\overline{P^{\mathsf{par}}_{\overline{\sigma}}}$ of $P^{\mathsf{par}}_{\overline{\sigma}}$ (line 6a), and to solving $\mathcal{P}' = \mathcal{P} \setminus \overline{P^{\mathsf{par}}_{\overline{\sigma}}}$.



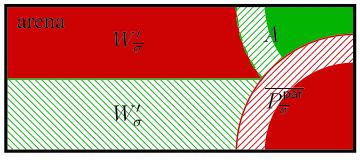
The algorithm then continues with the steps known from McNaughton's algorithm. That is, it next determines the set $\alpha^{-1}(d)$ of positions with maximal colour in $\mathcal{P}$,



and then constructs the σ-attractor A of $\alpha^{-1}(d)$ in $\mathcal{P}'$ (line 6d).

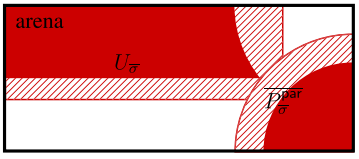


In the next step, the co-game $\mathcal{P}'' = \mathcal{P}' \setminus A$ of $\mathcal{P}'$ is solved by a recursive call of Procedure *Winning-Regions* (line 6e).



By Lemma 3.2, the winning region of player $\overline{\sigma}$ in $\mathcal{P}''$ is a $\overline{\sigma}$ paradise in $\mathcal{P}'$.

If $W'_{\overline{\sigma}}$ is empty, we can again evaluate the game immediately by Lemma 3.5 (line 7f). If $W'_{\overline{\sigma}}$ is non-empty, we can reduce solving $\mathcal{P}'$ to constructing the $\overline{\sigma}$-attractor $U_{\overline{\sigma}}$ of $W'_{\overline{\sigma}}$, which is a $\overline{\sigma}$-paradise in $\mathcal{P}'$ by Lemma 3.3, and to solving $\mathcal{P}'' = \mathcal{P}' \setminus W'_{\overline{\sigma}}$ (line 6h) by Lemma 3.4.



$$W'_{\overline{\sigma}} \neq \emptyset \Rightarrow |U_{\overline{\sigma}} \cup \overline{P^{\mathsf{par}}_{\overline{\sigma}}}| > \mathsf{par}$$

The Procedure *Winning-Regions* therefore computes the winning regions correctly.

Theorem 5.1. *[12] For a given parity game $\mathcal{P}$, Procedure* Winning-Regions *computes the complete winning regions of both players.*

Note that all operations can be extended to also return the winning strategies for both players without extra cost.

Efficiency While we know little about the size of $\overline{P_{\overline{\sigma}}^{\mathsf{par}}}$ (which may be empty) and $W'_{\overline{\sigma}}$ (which may be singleton), we know that their union is greater than par, because their union is a $\overline{\sigma}$-paradise (as the union of two $\overline{\sigma}$-paradises), and would otherwise be contained in $P_{\overline{\sigma}}^{\mathsf{par}}$.

We can therefore impose an upper bound on the number of iterations, which depends on the size of the parameter. While bigger parameters slow down the approximation procedure (c.f. Corollary 4.6), they restrict the size of the call tree.

For reasonable numbers of colours (that is, if the number of colours is in $O(\sqrt{n})$), the best results are obtained if the parameter is chosen such that the cost of calling the approximation procedure (line 6*a*) and the cost of the recursive call (line 6*e*) are approximately equivalent. This is the case if we set the parameter approximately to $\sqrt[3]{\frac{n^2}{c}}$.

For a high number of colours (that is, if the number of colours is in $\omega(\sqrt{n})$), the best results are obtained if the cost of calling the approximation procedure (line 6*a*) approximately coincides with the size of the call tree.

The key ingredients for our efficient big step approach are:

1. an algorithm for the efficient construction of σ/par-paradises (Section 4.4),
2. a correctness proof for the overall algorithm, and
3. a complexity analysis for suitable parameters for a reasonable and a high number of colours, respectively (Section 6).

6. Complexity

While the correctness of the algorithm is independent of the chosen parameter, its complexity crucially depends on this choice. We argue in favour of choosing the parameter par such that the cost of constructing a $\overline{\sigma}$/par-paradise and the cost of a recursive call are balanced. We analyse this for games with a fixed number of colours in two steps. In a first step, we identify the complexity in the usual terms, showing that parity games can be solved in time

$$O\left(m \cdot n^{\gamma(c)}\right),$$

where

$$\gamma(c) = \frac{c}{3} + \frac{1}{2} - \frac{4}{c^2 - 1} \qquad \text{if } c \text{ is odd,}$$

and

$$\gamma(c) = \frac{c}{3} + \frac{1}{2} - \frac{1}{3c} - \frac{4}{c^2} \qquad \text{if } c \text{ is even.}$$

We then discuss how the base of the exponent is affected when the colours are viewed as a parameter. It is customary to give this growth in a form of $O\left(m \cdot \left(\frac{n}{c} + 1\right)^{c-1}\right)$ for McNaughton's algorithm [17,8,29] and $O\left(c \cdot m \cdot \left(\frac{n}{\lfloor 0.5\,c \rfloor}\right)^{\lfloor 0.5\,c \rfloor}\right)$ for Jurdziński's [11], and something similar might be expected here. However, it turns out that the constant factor is falling much faster: we show that the complexity is approximately

$$O\left(m \cdot \left(\frac{6e^{1.\bar{6}}n}{c^2}\right)^{\gamma(c)}\right)$$

for a "small" $(o(\sqrt{n}))$ number of colours. The term "approximately" is used as we only show the complexity to be in $O\left(m \cdot \left(\frac{\kappa n}{c^2}\right)^{\gamma(c)}\right)$ for all $\kappa > 6e^{1.\bar{6}}$.

Finally, we show that, when the number of colours is high, we obtain the $n^{O(\sqrt{n})}$ complexity known from the older big-step approach of Jurdziński, Paterson, and Zwick [12].

6.1. Coarse analysis – fixed number of colours

For the important class of parity games with a reasonable number of colours — $c \in O(\sqrt{n})$ — we choose the parameter such that the cost for the recursive call (line 6*e*) coincides with the complexity of computing the approximation (line 6*a*). First, we show that the Procedure *Winning-Regions* indeed proceeds in big steps.

Lemma 6.1. *For a parameter* $\mathsf{par}(n, c)$*, the repeat loop of the algorithm is iterated at most* $\left\lfloor \frac{n}{\mathsf{par}(n,c)+2} \right\rfloor + 1$ *times.*

Proof. As discussed in the proof of Theorem 5.1, the $\overline{\sigma}$-attractor A of the computed approximation $P_{\overline{\sigma}}^{\mathsf{par}}$ (line 6a) and the winning region $W'_{\overline{\sigma}}$ of Player $\overline{\sigma}$ are $\overline{\sigma}$-paradises on $\mathcal{P}$ and $\mathcal{P} \setminus A$, respectively. Thus, their union U is a $\overline{\sigma}$-paradise of $\mathcal{P}$. If the size of U does not exceed $\mathsf{par}+1$, U is contained in $P_{\overline{\sigma}}^{\mathsf{par}}$ by Corollary 4.6. In this case, $W'_{\overline{\sigma}}$ is empty, and the loop terminates. Otherwise, a superset of U is subtracted from P during the iteration (lines 6c and 7h), which can happen at most $\left\lfloor \frac{n}{\mathsf{par}(n,c)+2} \right\rfloor$ times. □

Building on this observation, we define a parameter par such that the requirement of equal complexities is approximately satisfied. In order to do so, we proceed in two steps, starting with establishing the complexity for a fixed number of colours. In this coarse analysis, the number of colours is not treated as a parameter, but assumed to be fixed. The following table provides an overview.

number of colours	3	4	5	6	7	8
paradise construction	–	$O(mn)$	$O(mn^{1\frac{1}{2}})$	$O(mn^2)$	$O(mn^{2\frac{1}{3}})$	$O(mn^{2\frac{3}{4}})$
chosen parameter par	–	$n^{\frac{1}{2}}$	$n^{\frac{1}{2}}$	$n^{\frac{2}{3}}$	$n^{\frac{7}{12}}$	$n^{\frac{11}{16}}$
number of iterations $\frac{n}{\mathsf{par}}$	–	$n^{\frac{1}{2}}$	$n^{\frac{1}{2}}$	$n^{\frac{1}{3}}$	$n^{\frac{5}{12}}$	$n^{\frac{5}{16}}$
solving complexity	$O(mn)$	$O(mn^{1\frac{1}{2}})$	$O(mn^2)$	$O(mn^{2\frac{1}{3}})$	$O(mn^{2\frac{3}{4}})$	$O(mn^{3\frac{1}{16}})$

The colour coding shall help to identify similar complexities. The starting point is the solving complexity for three colours from Theorem 4.8. Once we have determined the cost of solving parity games with c colours, we invest a similar amount of time into constructing the $\overline{\sigma}$/par paradise for games with $c+1$ colours. For four colours, this is $O(mn)$. Once we have determined how much we are willing to invest into constructing the $\overline{\sigma}$/par paradise, we can infer the parameter. For four colour games, this is $\sqrt{n}$. And once we have determined the parameter, we can infer first the number of iterations, e.g., $\sqrt{n}$ for four player games, and then the complexity as the cost of each iteration times the number of iterations, e.g., $O(mn^{1.5})$ for games with four colours.

To capture this behaviour, we fix the function γ such that

$$\gamma(c) = \frac{c}{3} + \frac{1}{2} - \frac{1}{\lceil 0.5c \rceil \lfloor 0.5c \rfloor} = \frac{c}{3} + \frac{1}{2} - \frac{4}{c^2-1}$$

if c is odd, and

$$\gamma(c) = \frac{c}{3} + \frac{1}{2} - \frac{1}{3c} - \frac{1}{\lceil 0.5c \rceil \lfloor 0.5c \rfloor} = \frac{c}{3} + \frac{1}{2} - \frac{1}{3c} - \frac{4}{c^2}$$

if c is even, and choose

$$\beta(c) = \frac{\gamma(c)}{\lceil 0.5(c+1) \rceil}.$$

These definitions imply $\gamma(c+1) = \gamma(c) + 1 - \beta(c)$.

In the following proof, c is treated as a *constant*.

Theorem 6.2. *Solving a parity game $\mathcal{P}$ with $c > 2$ colours, m edges, and n game positions can be performed in time $O\left(mn^{\gamma(c)}\right)$.*

Proof. This is simple to prove by induction, where the **induction basis** ($c = 3$) is provided by Theorem 4.8.

For the **induction step** ($c \mapsto c+1$), we choose the parameter of

$$\mathsf{par}(n, c) = n^{\beta(c)}.$$

This provides a complexity of $O\left(mn^{\gamma(c)}\right)$ for the approximation in line 6(a) by Theorem 4.2, which together with the induction basis provides a complexity of $O\left(mn^{\gamma(c)}\right)$ of each iteration of the loop (lines 6(a) to 6(h)).

With Lemma 6.1, we can infer the claimed complexity first of the loop, and consequently (as the cost of lines 1 through 5 is dwarfed by the cost of the loop) for the algorithm. □

6.2. Parameter for $c \in o(\sqrt{n})$ – finer analysis

In this subsection, we provide an analysis for the complexity of the algorithm that treats the number of colours as a parameter. It is removed from the previous subsection, because we believe that most users would be happy with the simpler coarse analysis.

In a more fine-grained analysis, we start with adjusting the algorithm slightly, such that the parameter is adjusted in every iteration of the loop (Fig. 4).

In the remainder, we assume $c \in o(\sqrt{n})$, and establish the consistency of the estimations that, for

$$\mathsf{par} \approx \frac{(\kappa' n)^{\beta(c)}}{\sqrt[3]{c+1}},$$

Procedure Winning-Regions($\mathcal{P}$):
1. set d to the highest colour occurring in $\mathcal{P}$
2. if $d \leq 2$ then return *ThreeColour*($\mathcal{P}$)
3. set σ to d mod 2
4. set $W_{\overline{\sigma}}$ to $\emptyset$
5. repeat
   (a) set n to the size $|V|$ of $\mathcal{P}$
   (b) set $W'_{\overline{\sigma}}$ to $\overline{\sigma}$-*Attractor*(*Approximate*($\mathcal{P}$, par(n, d), $\overline{\sigma}$), $\mathcal{P}$)
   (c) set $W_{\overline{\sigma}}$ to $W_{\overline{\sigma}} \cup W'_{\overline{\sigma}}$
   (d) set $\mathcal{P}$ to $\mathcal{P} \setminus W'_{\overline{\sigma}}$
   (e) set $\mathcal{P}'$ to $\mathcal{P} \setminus \sigma$-*Attractor*($\alpha^{-1}(d)$, $\mathcal{P}$)
   (f) set (W'_0, W'_1) to *Winning-Regions*($\mathcal{P}'$)
   (g) if $W'_{\overline{\sigma}} = \emptyset$ then
      i. set W_σ to $V \setminus W_{\overline{\sigma}}$
      ii. return (W_0, W_1)
   (h) set $W_{\overline{\sigma}}$ to $W_{\overline{\sigma}} \cup \overline{\sigma}$-*Attractor*($W'_{\overline{\sigma}}$, $\mathcal{P}$)
   (i) set $\mathcal{P}$ to $\mathcal{P} \setminus \overline{\sigma}$-*Attractor*($W'_{\overline{\sigma}}$, $\mathcal{P}$)

Fig. 4. The adjusted Procedure *Winning-Regions*($\mathcal{P}$), which changes the parameter in each iteration of the loop.

which converges to $\sqrt[3]{\frac{(\kappa' n)^2}{c}}$ for a growing number of colours c, the time the algorithm takes is estimated by a function $t'(n, c)$ with

$$t'(n, c) \in O\Big(\frac{(\kappa''_g n)^{\gamma(c)}}{\sqrt[3]{c!^2}}\Big)$$

time for small constants κ''_g and κ'.

Theorem 6.3. *Parity games with $o(\sqrt{n})$ colours can be solved in $O\left(m \cdot \frac{(\kappa''_g n)^{\gamma(c)}}{\sqrt[3]{c!^2}}\right)$ time for all $\kappa''_g > \kappa = \frac{6}{\sqrt[3]{e}}$.*

Before turning to the proof, we provide an intuition for the problems that occur and the functions and parameters we will use in the proof.

Intuition and definitions We will use an inductive argument, which is slightly complicated by the problem that, after removing the attractor of a paradise, the number of positions is reduced, but not necessarily the number of colours. It is therefore possible that neither the assumption $c \in o(\sqrt{n})$ nor $c \ll \sqrt{n}$ extend to all parts of the call tree.

This proves to be a minor technical problem, which we meet with some re-writing. We first define a parameter

$$n' = c \cdot \sqrt{n},$$

which has the properties $n' \ll n$, $n' \in o(n)$, $c \in o(\sqrt{n'})$, and $c \ll n'$.

It is important to note that n' is *global*: it is calculated *once* before executing the algorithm, and it is never updated during the execution of the algorithm.

Intuitively, we start with $n + n'$ positions rather than with n positions, and calculate the complete time as if we had n' positions more than we actually have. Let us refer to them as n' *shadow positions*.

Thus, the time we need to solve parity games with n positions and c colours is estimated by $t'(n, c) = t(n + n', c)$—and not by $t(n, c)$—and we can use

$$t(\bar{n}, c) = 0 \text{ for all } \bar{n} \leq n'.$$

(As the n' shadow positions do not exist and are only used for our estimations, we know that, if there are at most n' positions left, they are all shadow positions: the game is empty.)

Next, we define the constants that used in the proof.

$$\overline{\kappa}'_g > \overline{\kappa}_g \geq \kappa' = \frac{3\sqrt[6]{e}}{\sqrt{2}}$$

are used for the estimation of the running time and the calculation of the parameter, and

$$\kappa''_g > \kappa'_g > \kappa_g > 2\sqrt{\frac{2}{e}} \cdot \overline{\kappa}'_g > 2\sqrt{\frac{2}{e}} \cdot \kappa' = \frac{6}{\sqrt[3]{e}} = \kappa$$

are used in the estimations of the running time. Note that κ_g'' can first be selected arbitrarily close to κ, and the remaining constants can be assigned afterwards. Finally, we define a constant

$$\bar{\kappa}_1' = \sqrt[6]{\left(\frac{\kappa}{\kappa_g'}\right)},$$

$\bar{\kappa}_1'$ is a constant (slightly) below 1, which is used in a proof.

With these constants in place, we fix the parameter to be

$$\mathsf{par}(\bar{n}, \bar{c}) = \left\lceil \frac{\left(\kappa' \bar{n}\right)^{\beta(\bar{c})}}{\sqrt[3]{\bar{c}+1}} \right\rceil.$$

As it is difficult to argue with ceiling operators, we also define $\kappa_{\bar{c},\bar{n}} \geq \kappa'$ to be the smallest constant greater or equal to κ', such that $\frac{\left(\kappa_{\bar{c},\bar{n}} \bar{n}\right)^{\beta(\bar{c})}}{\sqrt[3]{\bar{c}+1}}$ is an integer. We then define

$$\kappa_{\bar{c}} = \sup\{\kappa_{\bar{c},\bar{n}} \mid \bar{n} > \bar{c}^2\},$$

and observe that $\lim_{\bar{c}\to\infty} \kappa_{\bar{c}} = \kappa'$. Note that we only use a $\kappa_{\bar{c},\bar{n}}$ in a context, where $\bar{n} > n'$, while $\bar{c}^2 \leq c^2 \ll n' < \bar{n}$ holds. While the definition avoids the use of n', $\bar{c}^2 < \bar{n}$ is an important property for the definition to be useful.

Lemma 6.4. *For fixed $\bar{\kappa}_g' > \bar{\kappa}_g \geq \kappa_c$ and $\kappa_g > 2\sqrt{\frac{2}{e}} \cdot \bar{\kappa}_g'$, and for $c \in o(\sqrt{n})$, the running time of the approximation algorithm with parameter $\mathsf{par}(n, c)$ for a parity game with $c+1$ colours and m edges is in $O\left(m \cdot \frac{(\kappa_g n)^{\gamma(c)}}{\sqrt[3]{c!^2}}\right)$.*

Proof. By Corollary 4.6, the running time for the approximation is in $O\left(c\, m \binom{\mathsf{par}(n,c) + \lceil 0.5(c+1)\rceil}{\mathsf{par}(n,c)}\right)$. Recalling that

$$\mathsf{par}(n, c) = \frac{\left(\kappa_c n\right)^{\beta(c)}}{\sqrt[3]{c+1}},$$

we can obtain with $\gamma(c) = \beta(c)\lceil 0.5(c+1)\rceil$ that the running time is in

$$O\left(m \cdot (\kappa_c n + \sqrt[3]{c+1} \cdot \lceil 0.5(c+1)\rceil)^{\gamma(c)} \cdot \frac{c}{\lceil 0.5(c+1)\rceil! \cdot \sqrt[3]{c+1}^{\lceil 0.5(c+1)\rceil}}\right).$$

Using $c \in o(\sqrt{n})$, we obtain

$$O\left((\kappa_c n + \sqrt[3]{c+1} \cdot \lceil 0.5(c+1)\rceil)^{\gamma(c)}\right) \subseteq O\left((\bar{\kappa}_g' n)^{\gamma(c)}\right).$$

Finally, we use

$$\frac{c}{\lceil 0.5(c+1)\rceil! \cdot \sqrt[3]{c+1}^{\lceil 0.5(c+1)\rceil}} \approx \frac{(2e)^{\frac{c}{2}}}{c^{\frac{2c}{3}}} \quad \text{and} \quad \frac{1}{\sqrt[3]{c!^2}} \approx \frac{e^{\frac{2c}{3}}}{c^{\frac{2c}{3}}}$$

to infer

$$\frac{c}{\lceil 0.5(c+1)\rceil! \cdot \sqrt[3]{c+1}^{\lceil 0.5(c+1)\rceil}} \approx \left(2\sqrt{\frac{2}{e}}\right)^{\gamma(c)} \cdot \frac{1}{\sqrt[3]{c!^2}},$$

where the three '$\approx$' refer to factors between the respective two terms, which are subexponential in c. The subexponential factor of the third '$\approx$' is swallowed by the strict inequation $\kappa_g > 2\sqrt{\frac{2}{e}} \cdot \bar{\kappa}_g'$ when we estimate the running time by $O\left(m \cdot \frac{(\kappa_g n)^{\gamma(c)}}{\sqrt[3]{c!^2}}\right)$. □

The time consumed by an iteration of the loop is dominated by the recursive call and the approximation. For the proof, we make the constants hidden by the O notation explicit.

Corollary 6.5. *For $c+1$ colours (i.e., for maximal colour c) and a fixed $\kappa_g > \kappa$, there is a constant $\bar{\kappa}$ such that the time consumed during one iteration of a loop is bounded by the time spent by the recursive call plus $\bar{\kappa} \cdot m \frac{(\kappa_g n)^{\gamma(c)}}{\sqrt[3]{c!^2}}$ time steps.* □

In reference to the dominating role played by the approximation in this bound, we estimate the running time for n positions, m edges (left implicit), and $c+1$ colours by

$$t_a(n, c+1) = \bar{\kappa} \cdot m \frac{(\kappa_g n)^{\gamma(c)}}{\sqrt[3]{c!^2}} .$$

Proof of Theorem 6.3. We first sharpen $c \in o(\sqrt{n})$ to $n' = c\sqrt{n} \leq \frac{\kappa_g'' - \kappa_g'}{\kappa_g'} n$. We then provide a function $t'(n, c)$ that bounds the running time of the algorithm, where

$$t'(n, c) = t(n + n', c),$$

and $t(\bar{n}, c)$ is the function we discuss below. $t(\bar{n}, c)$ is intuitively the running time for $\bar{n} - n'$ real and n' shadow positions, whereas $t'(\bar{n}, c)$ would refer to $\bar{n}$ real positions. Recalling that, for $\bar{n} \leq n'$, there are only shadow and no real positions, we set $t(\bar{n}, c) = 0$ for all $c \in \omega$ and all $\bar{n} \leq n'$.

For given constants as defined above, we choose the minimal $k \in \omega$ such that the following properties hold for all $\bar{c} \geq k$.

1. $\kappa_{\bar{c}} \leq \bar{\kappa}_g$,
2. $\bar{\kappa}_1' \cdot {\kappa_g'}^{1-\beta(\bar{c})} \cdot {\kappa'}^{\beta(\bar{c})} \cdot \frac{\gamma(\bar{c}+1)}{\bar{c}+1} \geq 1$, and
3. $\bar{n}^{\gamma(\bar{c}+1)} - (\bar{n} - \mathsf{par}(\bar{n}, \bar{c}))^{\gamma(\bar{c}+1)} \geq \bar{\kappa}_1' \cdot \mathsf{par}(\bar{n}, \bar{c}) \cdot \gamma(\bar{c}+1) \cdot n^{\gamma(\bar{c}+1)-1}$ holds for all $\bar{n} > \frac{\kappa_g'}{\kappa_g'' - \kappa_g'} \bar{c}^2$ (and thus for all values of interest: smaller values $\bar{n}$ are also smaller than n').

We now prove our claim by induction. For the **induction basis**, we observe that Theorem 6.2 implies that the following holds for all $c \leq k$ for an arbitrary (but fixed) constant $k \in \mathbb{N}$: the running time of the algorithm is bound by $\bar{\kappa}' \cdot m \cdot \frac{(\kappa_g n)^{\gamma(c)}}{\sqrt[3]{c!^2}}$. (Thus, for all $c \leq k$, we can follow the simplified version of the algorithm referred to in Theorem 6.2.)

Before continuing with the induction step, we let

$$\bar{\kappa}_0 = \frac{\max\{\bar{\kappa}, \bar{\kappa}'\}}{\bar{\kappa}_1'}$$

be the maximum of the constant $\bar{\kappa}'$ from above and the constant $\bar{\kappa}$ from Corollary 6.5, divided by the constant (slightly) below 1.

We then define the following series of constants.

$$\bar{\kappa}_{i+1} = \bar{\kappa}_i + \bar{\kappa}_0 \cdot \left(\frac{\kappa_g}{\kappa_g'}\right)^{\gamma(i+1)} .$$

We use the constant $\bar{\kappa}_\infty = \lim_{i \to \infty} \bar{\kappa}_i$ for the limit of this series. We then relax the induction basis to the observation that, for all $c \leq k$ and $\bar{n} > n'$ (recall that $n' \gg c^2$),

$$t(n, c) = \bar{\kappa}_c \cdot m \cdot \frac{(\kappa_g' n)^{\gamma(c)}}{\sqrt[3]{c!^2}}$$

is an upper bound on the running time of the algorithm for n positions (including shadow positions).

For the **induction step** from c to $c+1$, we can use as induction hypothesis that $t(\bar{n}, c)$ bounds the running time (with shadow positions) of the algorithm.

We now start an inductive proof for $c+1$, which is provided as an induction over $\bar{n}$.

IB: For the induction basis, we recall that $t(\bar{n}, c+1) = 0$ for all $\bar{n} \leq n'$.

IS: Let $\bar{n} > n'$. For the induction step from all $\bar{n}' < \bar{n}$ to $\bar{n}$, we assume that we have shown the property for all $\bar{n}' < \bar{n}$ (induction hypothesis) and show that it also holds for $\bar{n}$.

The first relevant observation is that $c \ll \sqrt{\bar{n}}$ implies $\gamma(c) \ll \mathsf{par}(\bar{n}, c+1)$. We have used this in (3) to estimate

$$\begin{aligned} \bar{n}^{\gamma(c+1)} - (\bar{n} - \mathsf{par}(\bar{n}, c))^{\gamma(c+1)} &\geq \bar{\kappa}_1' \cdot \mathsf{par}(\bar{n}, c) \cdot \gamma(c+1) \cdot \bar{n}^{\gamma(c+1)-1} \\ &\geq \bar{\kappa}_1' \cdot \left(\frac{{\kappa'}^{\beta(c)}}{\sqrt[3]{c+1}}\right) \cdot \gamma(c+1) \cdot \bar{n}^{\gamma(c)} . \end{aligned}$$

From here, we can estimate the running time (with shadow positions) as follows.

$$\begin{aligned} &t(\bar{n}, c+1) - t(\bar{n} - \mathsf{par}(\bar{n}, c), c+1) \\ &\quad = \bar{\kappa}_{c+1} \cdot m \cdot \frac{{\kappa_g'}^{\gamma(c+1)}}{\sqrt[3]{(c+1)!^2}} \cdot \left(\bar{n}^{\gamma(c+1)} - (\bar{n} - \mathsf{par}(\bar{n}, c))^{\gamma(c+1)}\right) \end{aligned}$$

$$\begin{aligned}
&\geq \overline{\kappa}_{c+1} \cdot \left(\overline{\kappa}'_1 \cdot {\kappa'_g}^{1-\beta(c)} \cdot {\kappa'}^{\beta(c)} \cdot \frac{\gamma(c+1)}{c+1}\right) \cdot m \cdot \frac{(\kappa'_g \bar{n})^{\gamma(c)}}{\sqrt[3]{c!^2}} \\
&\geq \overline{\kappa}_{c+1} \cdot m \cdot \frac{(\kappa'_g \bar{n})^{\gamma(c)}}{\sqrt[3]{c!^2}} \\
&\geq \overline{\kappa}_{c} \cdot m \cdot \frac{(\kappa'_g \bar{n})^{\gamma(c)}}{\sqrt[3]{c!^2}} + t_a(\bar{n}, c+1) \\
&= t(\bar{n}, c) + t_a(\bar{n}, c+1) \,.
\end{aligned}$$

Consequently, we can infer that

$$t(\bar{n}, c+1) \geq t(\bar{n}, c) + t_a(\bar{n}, c+1) + t(\bar{n} - \mathsf{par}(\bar{n}, c+1)$$

holds for the chosen function, which establishes the estimation in the inner induction.

This, in turn, closes the estimation for the outer induction.

What remains it to see that the $t'(n, c) = t(n + n', c)$ has the required property. But this is implied by $n' \leq \frac{\kappa''_g - \kappa'_g}{\kappa'_g} \cdot n$. □

While one can use this proof to establish this bound for κ''_g arbitrarily close to κ, note that this has a significant impact on the constant factor.

Using again $c! \approx \left(\frac{c}{e}\right)^c$, we obtain:

Corollary 6.6. *Parity games with $o(\sqrt{n})$ colours can be solved in $O\left(m \left(\frac{\kappa n}{c^2}\right)^{\gamma(c)}\right)$ time for all $\kappa > 6e^{1\frac{2}{3}}$.*

6.3. High number of colours

We close with a rough analysis of the cost for medium and high numbers of colours. In both cases, we aim at roughly aligning the size of the call tree and the cost of the approximation.

Different to the previous subsection, we fix the parameter initially and do *not* adjust it during the algorithm. We assume that the number of colours and the parameter are large, e.g., $c, \mathsf{par} \in \omega(\sqrt[3]{n})$, in our estimation (the main target is $c \in \Omega(\sqrt{n})$).

For $c = \sqrt{n}$, we roughly balance the cost of the approximation for a parameter in $\theta(\sqrt{n})$ and the size of the call tree.

To get an impression of the size of the call tree, we can encode each node in a call tree by a sequence of *call* and *return* symbols. If we assume c colours and a parameter par (which remains constant for simplicity), then each node in such a call tree can be encoded by a sequence, where

- the number of calls is smaller than the number of colours and
- the number of calls and par times the number of returns is at most the number of positions.

The number of leaves encoded by these sequences can be estimated from above by $\binom{c + \frac{n}{\mathsf{par}}}{c}$, where the estimation from above allows any sequence of c calls and $\frac{n}{\mathsf{par}}$ returns. The number of positions in the call tree is of the same order.

The cost of a single estimation with parameter par is $O\left(c\, m \binom{\mathsf{par} + \lceil 0.5c \rceil}{\mathsf{par}}\right)$.

For $c \leq \sqrt{n}$, we select $\mathsf{par} = c$. This results in an estimation of the cost of the approximations of $2^{O(c)}$ (assuming $c \in \Omega(\log n)$), and an estimation of the size of the call tree of $(1 + \frac{n}{c^2})^{O(c)}$.

Theorem 6.7. *Parity games with n positions and $c \leq \sqrt{n}$ colours can be solved in $\left(1 + \frac{n}{c^2}\right)^{O(c)}$ time.*

For $c \geq \sqrt{n}$, we choose a parameter around $\sqrt{n}$, e.g., $\mathsf{par} = \lceil \sqrt{n} \rceil$. This results in a call tree of approximate size and approximation which takes approximate time $\left(1 + \frac{c}{\sqrt{n}}\right)^{O(\sqrt{n})}$. The cost of the overall computation is therefore also in $\left(1 + \frac{c}{\sqrt{n}}\right)^{O(\sqrt{n})}$.

Noting that $c \geq \sqrt{n}$ is not used in the estimation, we get:

Theorem 6.8. *Parity games with n positions and c colours can be solved in $\left(1 + \frac{c}{\sqrt{n}}\right)^{O(\sqrt{n})}$ time.*

This essentially boils down to the $n^{O(\sqrt{n})}$ result of Jurdziński, Paterson, and Zwick [12], with a slight improvement when $\log\left(\frac{c}{\sqrt{n}}\right)$ is in $o(\log n)$.

Acknowledgments

I would like to thank the exceptionally through and helpful reviewers for their useful comments. They helped to make especially Section 6.2 significantly more accessible.

This work was partially supported by the Engineering and Physical Science Research Council (EPSRC) through grant EP/H046623/1.

References

[1] R. Alur, T.A. Henzinger, O. Kupferman, Alternating-time temporal logic, J. ACM 49 (2002) 672–713.
[2] D. Berwanger, A. Dawar, P. Hunter, S. Kreutzer, Dag-width and parity games, in: Proc. STACS, Springer-Verlag, 2006, pp. 524–536.
[3] H. Björklund, S. Sandberg, S.G. Vorobyov, Memoryless determinacy of parity and mean payoff games: a simple proof, Theor. Comput. Sci. 310 (2004) 365–378.
[4] H. Björklund, S. Vorobyov, A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games, Discrete Appl. Math. 155 (2007) 210–229.
[5] A. Browne, E.M. Clarke, S. Jha, D.E. Long, W. Marrero, An improved algorithm for the evaluation of fixpoint expressions, Theor. Comput. Sci. 178 (1997) 237–255.
[6] L. de Alfaro, T.A. Henzinger, R. Majumdar, From verification to control: dynamic programs for omega-regular objectives, in: Proc. LICS, IEEE Computer Society Press, June 2001, pp. 279–290.
[7] E.A. Emerson, C.S. Jutla, A.P. Sistla, On model-checking for fragments of μ-calculus, in: CAV, 1993, pp. 385–396.
[8] E.A. Emerson, C. Lei, Efficient model checking in fragments of the propositional μ-calculus, in: Proc. LICS, IEEE Computer Society Press, 1986, pp. 267–278.
[9] J. Fearnley, S. Schewe, Time and parallelizability results for parity games with bounded tree and DAG width, Log. Methods Comput. Sci. 467 (2013) 1–31.
[10] M. Jurdziński, Deciding the winner in parity games is in UP ∩ co-UP, Inf. Process. Lett. 68 (1998) 119–124.
[11] M. Jurdziński, Small Progress Measures for Solving Parity Games, Lect. Notes Comput. Sci., vol. 1770, Springer, 2000, pp. 290–301.
[12] M. Jurdziński, M. Paterson, U. Zwick, A deterministic subexponential algorithm for solving parity games, 38 (2008) 1519–1532.
[13] D. Kozen, Results on the propositional μ-calculus, Theor. Comput. Sci. 27 (1983) 333–354.
[14] O. Kupferman, M. Vardi, Module checking revisited, in: Proc. CAV, in: Lect. Notes Comput. Sci., vol. 1254, Springer-Verlag, 1997, pp. 36–47.
[15] M. Lange, Solving parity games by a reduction to SAT, in: Proc. Int. Workshop on Games in Design and Verification, 2005.
[16] W. Ludwig, A subexponential randomized algorithm for the simple stochastic game problem, Inf. Comput. 117 (1995) 151–155.
[17] R. McNaughton, Infinite games played on finite graphs, Ann. Pure Appl. Logic 65 (1993) 149–184.
[18] J. Obdržálek, Fast mu-calculus model checking when tree-width is bounded, in: Proc. CAV, Springer-Verlag, 2003, pp. 80–92.
[19] N. Piterman, From nondeterministic Büchi and Streett automata to deterministic parity automata, Log. Methods Comput. Sci. 3 (2007).
[20] A. Puri, Theory of hybrid systems and discrete event systems, PhD thesis, Computer Science Department, University of California, Berkeley, 1995.
[21] S. Schewe, Solving parity games in big steps, in: Proceedings of the 27th Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2007, 12–14 December, New Delhi, India, in: Lect. Notes Comput. Sci., vol. 4805, Springer-Verlag, 2007, pp. 449–460.
[22] S. Schewe, Synthesis of distributed systems, PhD thesis, Universität des Saarlandes, 2008.
[23] S. Schewe, Tighter bounds for the determinisation of Büchi automata, in: Proceedings of the Twelfth International Conference on Foundations of Software Science and Computation Structures, FoSSaCS 2009, 22–29 March, York, England, UK, in: Lect. Notes Comput. Sci., vol. 5504, Springer-Verlag, 2009, pp. 167–181.
[24] S. Schewe, B. Finkbeiner, Satisfiability and finite model property for the alternating-time μ-calculus, in: Proceedings of the 15th Annual Conference of the European Association for Computer Science Logic, CSL 2006, 25–29 September, Szeged, Hungary, in: Lect. Notes Comput. Sci., vol. 4207, Springer-Verlag, 2006, pp. 591–605.
[25] S. Schewe, B. Finkbeiner, Synthesis of asynchronous systems, in: Proceedings of the 16th International Symposium on Logic-Based Program Synthesis and Transformation, LOPSTR 2006, 12–14 July, Venice, Italy, in: Lect. Notes Comput. Sci., vol. 4407, Springer-Verlag, 2006, pp. 127–142.
[26] M.Y. Vardi, Reasoning about the past with two-way automata, in: Proc. ICALP, Springer-Verlag, 1998, pp. 628–641.
[27] J. Vöge, M. Jurdziński, A discrete strategy improvement algorithm for solving parity games (Extended abstract), in: Proc. CAV, Springer-Verlag, July 2000, pp. 202–215.
[28] T. Wilke, Alternating tree automata, parity games, and modal μ-calculus, Bull. Soc. Math. Belg. 8 (2001).
[29] W. Zielonka, Infinite games on finitely coloured graphs with applications to automata on infinite trees, Theor. Comput. Sci. 200 (1998) 135–183.
[30] U. Zwick, M. Paterson, The complexity of mean payoff games on graphs, Theor. Comput. Sci. 158 (1996) 343–359.