



OPEN ACCESS

EDITED BY

Jonni Virtema,
The University of Sheffield,
United Kingdom

REVIEWED BY

Shaull Almagor,
Technion Israel Institute of
Technology, Israel
Engel Lefauchaux,
Inria Nancy-Grand-Est Research
Centre, France

*CORRESPONDENCE

Daniele Dell'Erba
daniele.dell-erba@liverpool.ac.uk
Sven Schewe
sven.schewe@liverpool.ac.uk

SPECIALTY SECTION

This article was submitted to
Theoretical Computer Science,
a section of the journal
Frontiers in Computer Science

RECEIVED 05 May 2022

ACCEPTED 26 August 2022

PUBLISHED 20 September 2022

CITATION

Dell'Erba D and Schewe S (2022)
Smaller progress measures and
separating automata for parity games.
Front. Comput. Sci. 4:936903.
doi: 10.3389/fcomp.2022.936903

COPYRIGHT

© 2022 Dell'Erba and Schewe. This is
an open-access article distributed
under the terms of the [Creative
Commons Attribution License \(CC BY\)](#).
The use, distribution or reproduction
in other forums is permitted, provided
the original author(s) and the copyright
owner(s) are credited and that the
original publication in this journal is
cited, in accordance with accepted
academic practice. No use, distribution
or reproduction is permitted which
does not comply with these terms.

Smaller progress measures and separating automata for parity games

Daniele Dell'Erba* and Sven Schewe*

Department of Computer Science, University of Liverpool, Liverpool, United Kingdom

Calude et al. have recently shown that parity games can be solved in quasi-polynomial time, a landmark result that has led to several approaches with quasi-polynomial complexity. Jurdzinski and Lazic have further improved the precise complexity of parity games, especially when the number of priorities is low (logarithmic in the number of positions). Both of these algorithms belong to a class of game solving techniques now often called separating automata: deterministic automata that can be used as witness automata to decide the winner in parity games up to a given number of states and colors. We suggest several adjustments to the approach of Calude et al. that lead to smaller statespaces. These include and improve over those earlier introduced by Fearnley et al. We identify two of them that, together, lead to a statespace of exactly the same size Jurdzinski and Lazic's concise progress measures, which currently hold the crown as the smallest statespace. The remaining improvements, hence, lead to a further reduction in the size of the statespace, making our approach the most succinct progress measure available for parity games.

KEYWORDS

parity games, progress measures, value iteration, separating automata, quasi-polynomial algorithms

1. Introduction

Parity games are two-player perfect information turn-based zero-sum games of infinite duration played on finite directed graphs. Each vertex, that is labeled with an integer *color*, is assigned to one of the two players *even* or *odd*. A play consists of an infinite sequence of players' moves around the graph, and the winner is determined by the *parity* of the largest color encountered along the play. Hence, the player even wins if it is an even color, and player odd wins otherwise.

Parity games have been extensively studied for their practical applications, to determine their complexity status, and to find efficient solutions.

From a practical point of view, many problems in formal verification and synthesis can be reformulated in terms of solving parity games. Computing winning strategies for these games is linear-time equivalent to solving the modal μ -calculus model-checking problem (Emerson and Lei, 1986; Emerson et al., 2001). Parity games can be applied to solve the complementation problem for alternating automata (Grädel et al., 2002) or the

emptiness of the corresponding nondeterministic tree automata (Kupferman and Vardi, 1998). These automata, in turn, can be used to solve the satisfiability and model-checking problems for several expressive logics (Berwanger and Grädel, 2004; Chatterjee et al., 2010; Mogavero et al., 2010, 2012; Benerecetti et al., 2013), such as μ -calculus (Wilke, 2001; Schewe and Finkbeiner, 2006) and ATL* (Alur et al., 2002; Schewe, 2008).

On the complexity-theoretic side, determining the winner of a parity game is a problem that lies in $\text{NP} \cap \text{co-NP}$ (Emerson et al., 2001), being memoryless determined (Martin, 1975; Emerson and Jutla, 1991; Mostowski, 1991), and it has even been proved to belong to $\text{UP} \cap \text{co-UP}$ (Jurdziński, 1998), and later to be solvable in quasi-polynomial time (Calude et al., 2017). However, determining whether they belong also to P is still an open problem.

The existing algorithms for solving parity games can be divided into two classes. The first one collects approaches that solve the game by creating a winning strategy for one of the two players in the entire game. This can be done by either employing a value iteration over progress measures (Jurdziński, 2000) or iteratively improving the current strategy (Vöge and Jurdziński, 2000; Fearnley, 2010; Friedmann, 2013). To the second class belong approaches that decompose the solution of a game into the analysis of its subgames in a divide-and-conquer concept. To do so, these approaches partition the game into a set of positions that satisfy certain properties. Then, these portions of the game are recursively solved and by suitably composing them also the initial game is solved. The name of the sets employed by the technique depends on the properties: attraction set (Zielonka, 1998; Schewe, 2007), region (Benerecetti et al., 2016, 2018a,b), tangle (van Dijk, 2018), and justification (Lapauw et al., 2020).

Many algorithms from both classes have been refined to achieve a quasi-polynomial upper bound since the contribution of Calude et al. (2017). This seminal algorithm works as a value-iteration approach with compact measures for which a poly-logarithmic size witness is sufficient, rather than storing the entire history of the play that is exponential. The same approach has been refined improving the complexity result (Fearnley et al., 2017; Jurdziński and Lazic, 2017), while the same complexity has been achieved by several different approaches such as the register-index algorithm (Lehtinen, 2018) and the bounded version of the recursive algorithm (Parys, 2019). Interestingly, all known quasi-polynomial algorithms can be derived by the separation approach which also provides a lower bound for these techniques (Czerwinski et al., 2018).

1.1. Contribution

We adjust the definitions of *witnesses*, the data structure first used by Calude et al. (2017) and the way they are updated in several ways. In the following, we provide a high-level idea of

our contribution in comparison with other approaches (Calude et al., 2017; Jurdziński and Lazic, 2017; Fearnley et al., 2019). This description makes use of the notion of witness and can be skipped by non expert readers.

The most clear-cut improvement is the increased succinctness of the resulting structures. The main contributor to this improvement is the one that skips the double occurrence of odd colors in the classic witnesses. Where the highest color is even, this improvement alone obtains a statespace of quite different structure to, but the same size as, the currently smallest statespace from Jurdziński and Lazic approach (Jurdziński and Lazic, 2017). Integrating this with the small refinement from Fearnley et al. (2019), in which when the maximal color is odd, we can just reset the witness to its initial value instead of recording this maximal odd value, extends the refinement to the case where the maximal color is odd.

The other advantages are minor. They include not recording odd colors in the least relevant position (known from Fearnley et al., 2019 broadly halving the statespace) and a novel way of counting the length of even chains represented by witnesses, which broadly halves the statespace at the points where the complexity jumps—when at the even chains that need to be considered reach a new power of 2—but successively loses this advantages, losing it completely which the even chains that need to be represented can reach the next power of 2 minus 1). Integrating these two improvements provide a statespace reduction from just under 2 to just under 4 of our approach over (Jurdziński and Lazic, 2017). A minor additional saving is reached by also skipping minimal colors when they are odd (Fearnley et al., 2019).

The second improvement is a re-definition of the semantics of witnesses, moving from the classic witnesses to *color witnesses*, where all positions with the same color in a witness refer to one chain, instead of referring to many. This re-definition accelerates the convergence, though the acceleration is muted where it is used for value iteration.

1.2. Outline

We discuss a variation of the algorithm of Calude et al. (2017), partly in the original version and partly in the variation suggested by Fearnley et al. (2019), to extend this approach to value iteration.

After the general preliminaries, we recap this approach in Section 3, using a mild variation of the witness from Calude et al. (2017) for a basic update rule up' that updates a witness \mathbf{b} when reading a state with color v , and an antagonistic update rule au' that updates a witness \mathbf{b} when reading a state with color v .

We then amend those rules in two steps. The first step (Section 4) reduces the statespace but otherwise retains the classic lines of Fearnley et al. (2019). It is a simple extension that carries the easiest to spot improvement of this study: the

reductions from the statespace of the witnesses used, leading to more concise witnesses.

The backbone of the statespace reduction is to simply restrict the number of times an odd color occurs in a witness to at most once. Where the maximal color is even, this change alone leads to a perfect match in size with the statespace of Jurdziński and Lazic (2017), which is currently the smallest. This perfect match is a bit surprising, as the structure of the statespace is very different.

The remaining changes extend the restriction to the case where the maximal color is odd and collect some further minor reductions that roughly lead to a statespace reduction that is usually in a range between 2 and 4, where the advantage is strongest when the number of states with an even number of colors is a power of 2.

Subsequently, we re-interpret the semantics of a witness in Section 5. This change in semantics does not change the statespace, but it allows for updating the witnesses faster. Faster updating is mainly improving the basic update rule, but to some extent also the antagonistic update rule, leading to faster convergence in both cases.

We then turn to an estimation of the new statespace in Section 6. For this, we proceed in several steps. We first look at the two classic statespaces, considering the previously most concise one those of Calude et al.'s original QP algorithm (Calude et al., 2017).

We then turn on using those improvements to Calude et al. (2017) that lead to a statespace of size equal to that of Jurdziński and Lazic (2017). This is done in order to be able to show that the two statespaces are of precisely the same size, but also to have a clear understanding of which improvements remain beyond this, and to focus on how they influence the statespace.

We then exemplify how the three statespaces compare in size in Section 7.

2. Preliminaries

Parity games are turn-based zero-sum games played between two players, even and odd, over finite graphs. A parity game \mathcal{P} is a tuple (V_e, V_o, E, C, ϕ) , where

- $(V = V_e \cup V_o, E)$ is a finite directed graph, where the set V of vertices is partitioned into a set V_e of vertices controlled by the player *even* and a set V_o of vertices controlled by player *odd*, and where $E \subseteq V \times V$ is the set of edges;
- $C \subseteq \mathbb{N} = \{1, 2, 3, \dots\}$ is a finite consecutive set of colors, such that $C = \{1, 2, \dots, \max\{C\}\}$ or $C = \{2, 3, \dots, \max\{C\}\}$ holds; and
- $\phi : V \rightarrow C$ is the coloring functions that maps each vertex to a color.

We define $C^- = C \setminus \{\max\{C\}\}$ if the highest color $\max\{C\}$ is odd, and $C^- = C$ if the highest color $\max\{C\}$ is even. We also require that every vertex has at least one outgoing edge.

Intuitively, a parity game \mathcal{P} is played between the two players by moving a token along the edges of the directed graph (V, E) . A play of such a game starts at some initial vertex $v_0 \in V$ where the token is placed at the beginning. The player controlling this vertex then chooses a successor vertex v_1 such that $(v_0, v_1) \in E$, and the token is moved to this successor vertex. In the next turn, the player controlling the vertex v_1 makes his choice by picking a successor vertex v_2 where to move the token, such that $(v_1, v_2) \in E$, and so on. In this manner, both players move the token over the arena and, thus, form an infinite play of the game.

Formally, a play of a game \mathcal{P} is an infinite sequence of vertices $\langle v_0, v_1, \dots \rangle \in V^\omega$ such that, for all $i \geq 0$, we have that $(v_i, v_{i+1}) \in E$. We denote as $\text{Plays}_{\mathcal{P}}(v)$ the set of plays of the game \mathcal{P} that origins in a vertex $v \in V$ and as $\text{Plays}_{\mathcal{P}}$ the set of all plays of the game. We omit the subscript when the arena is clear from the context. The color mapping $\phi : V \rightarrow C$ can be extended from vertices to plays by defining the mapping $\phi : \text{Plays} \rightarrow C^\omega$ as $\langle v_0, v_1, \dots \rangle \mapsto \langle \phi(v_0), \phi(v_1), \dots \rangle$.

A play $\langle v_0, v_1, \dots \rangle$ is won by the player *even* if $\limsup_{i \rightarrow \infty} \phi(v_i)$ is even, and by player *odd* otherwise.

A *prefix* of a play (or play prefix) is a non-empty initial sequence $\langle v_0, v_1, \dots, v_m \rangle$ of a play $\langle v_0, v_1, \dots \rangle$.

For a play $\rho = \langle v_0, v_1, \dots \rangle$ or play prefix $\rho = \langle v_0, v_1, \dots, v_n \rangle$, an *even chain* of length ℓ is a sequence of positions $p_1 < p_2 < p_3 < \dots < p_\ell$ (with $0 \leq p_1$ and, for plays prefixes, $p_\ell \leq n$) in ρ that has the following properties:

- for all $j \in \{1, \dots, \ell\}$, we have that $\phi(v_{p_j})$ is even, and
- for all $j \in \{1, \dots, \ell - 1\}$ the colors in the subsequence defined by p_j and p_{j+1} are less than or equal to $\phi(p_j)$ or $\phi(p_{j+1})$. More formally, we have that all colors $\phi(v_{p_j}), \phi(v_{(p_j)+1}), \dots, \phi(v_{p_{(j+1)}})$ are less than or equal to $\max\{\phi(v_{p_j}), \phi(v_{p_{j+1}})\}$.

A strategy for the player *even* is a function $\sigma : V^*V_e \rightarrow V$ such that $(v, \sigma(\rho, v)) \in E$ for all $\rho \in V^*$ and $v \in V_e$. If a strategy σ only depends on the last state, then is called memoryless ($\sigma(\rho, v) = \sigma(\rho', v)$ for all $\rho, \rho' \in V^*$ and $v \in V_e$). A play $\langle v_0, v_1, \dots \rangle$ is consistent with σ if, for every prefix $\rho_n = v_0, v_1, \dots, v_n$ of the play that ends in a state of player *even* ($v_n \in V_e$), $\sigma(\rho_n) = v_{n+1}$ holds. The player *even* wins the game starting at v_0 if the player has a strategy σ such that either all plays $\langle v_0, v_1, \dots \rangle$ consistent with σ satisfying $\limsup_{i \rightarrow \infty} \phi(v_i)$ is even, that, being the game finite (i.e., the number of states is finite), simplifies in all plays $\langle v_0, v_1, \dots \rangle$ consistent with σ contain a loop $v_i, v_{i+1}, \dots, v_{i+k}$, that satisfies $v_i = v_{i+k}$ and that $\max\{\phi(v_i), \dots, \phi(v_{i+j})\}$ is even. In both cases, σ might be memoryless.

A *separating automaton* (Bojańczyk and Czerwiński, 2018) for parity games with a set of colors C and a bounded number of up to b states with even color¹, is a deterministic reachability automaton $\mathcal{A} = (Q, C, q_0, \delta, \text{won})$, where

- Q is the set of states, with $q_0, \text{won} \in Q$, q_0 is the initial state and won is the target state (and sink), and
- $\delta : Q \times C \rightarrow Q$ is the transition function (with $\delta(\text{won}, v) = \text{won}$ for all $v \in C$),

such that, for all parity games with colors $\subseteq C$ that have no more than b states of even color, the following holds:

- if $v \in V$ is a winning state for the player *even*, then there is a positional strategy σ for the player *even* such that, for every play ρ from v consistent with σ , $\phi(\rho)$ is accepted by \mathcal{A} (i.e., \mathcal{A} reaches the target state won); and
- if $v \in V$ is a winning state for player *odd*, then there is a positional strategy σ for player *odd* such that, for every play ρ from v consistent with σ , $\phi(\rho)$ is rejected by \mathcal{A} (i.e., \mathcal{A} does not reach the target state won).

3. Classic Witnesses

We adjust the approach from Calude et al. (2017) and Fearnley et al. (2019), and this section is predominantly taking the representation from Fearnley et al. (2019). It does, however, change some details in the definitions of *i*-witnesses that end in an odd priority and the definition of the value of a witness slightly to suit the rest of the article better. Where the proofs are affected, they are adjusted and given, but the proofs are mostly unaffected by these minor details.

3.1. Classic forward witness

We start with describing the *old* witness without making its semantics formal (as we do not need it in this article), and will turn to the new *concise witness* (Section 4) and the *color witness* (Section 5) afterwards.

3.1.1. *i*-Witnesses

Let $\rho = v_1, v_2, \dots, v_m$ be a prefix of a play of the parity game. An *even i-witness* is a sequence of (not necessarily consecutive) positions of ρ

$$p_1, p_2, p_3, \dots, p_{2^i},$$

¹ The bound b is often given instead of the number of states.

of length, exactly 2^i , and an *odd i-witness* is a sequence of (not necessarily consecutive) positions of ρ

$$p_0, p_1, p_2, \dots, p_{2^i}$$

of length exactly $2^i + 1$, that satisfy the following properties:

- **Position:** Each p_j specifies a position in the play ρ , so each p_j is an integer that satisfies $1 \leq p_j \leq m$.
- **Order:** The positions are ordered. Thus, we have $p_j < p_{j+1}$ for all $j < 2^i$.
- **Evenness:** All positions but the final one are even. Formally, for all $j < 2^i$, the color $\phi(v_{p_j})$ of the vertex in position p_j is even.
For position p_{2^i} , its color $\phi(v_{p_{2^i}})$ is even for an *even i-witness*, and odd for an *odd i-witness*.
- **Inner domination:** The color of every vertex between p_j and p_{j+1} is dominated by the color of p_j or the color of p_{j+1} . Formally, for all $j < 2^i$, the color of every vertex in the subsequence $v_{p_j}, v_{(p_j)+1}, \dots, v_{p_{(j+1)}}$ is less than or equal to $\max\{\phi(v_{p_j}), \phi(v_{p_{j+1}})\}$.
- **Outer domination:** The color of p_{2^i} is greater than or equal to the color of every vertex that appears after p_{2^i} in ρ . Formally, for all k in the range $p_{2^i} < k \leq m$, we have that $\phi(v_k) \leq \phi(v_{p_{2^i}})$.

It follows from these properties that an *i-witness* contains an even chain of length 2^i .

3.1.2. Witnesses

We define $C_- = C^- \cup \{_ \}$ as the set of colors plus the $_$ symbol. A *witness* is a sequence²

$$b_k, b_{k-1}, \dots, b_1, b_0,$$

such that each element $b_i \in C_-$, and that satisfies the following properties:

- **Witnessing:** there exists a family of *i-witnesses*, one for each element b_i with $b_i \neq _$. We refer to such an *i-witness* in the run ρ . We will refer to this witness as

$$p_{i,1}, p_{i,2}, \dots, p_{i,2^i}$$

for even *i-witnesses* and

$$p_{i,0}, p_{i,1}, \dots, p_{i,2^i}$$

² While k can be viewed as "big enough" or as "of arbitrary size" for the definition, we will later see that a length $k+1$, with $k = \lceil \log_2(e) \rceil$, where e is the number of vertices with an even color, or any other sufficient criterion for the maximal length of an even chain, is sufficient.

for odd i -witnesses. Thus, even i -witnesses are in particular even chains of length 2^i , while odd i -witnesses extended even chains that start with an even chain of length 2^i (and are extended by a further position).

- **Dominating color:** For each $b_i \neq _$, we have that $b_i = \phi(v_{p_{i,2^i}})$. That is, b_i is the outer domination color of the i -witness.
- **Ordered sequences:** The i -witness associated with b_i starts after a j -witness associated with b_j whenever $i < j$. Formally, for all i and j with $i < j$, if $b_i \neq _$ and $b_j \neq _$, then $p_{j,2^j} < p_{i,1}$ when the i -witness is even, and $p_{j,2^j} < p_{i,0}$ otherwise.

For a little bit of extra conciseness, we also require that b_0 is either even or $_$.

Note that the witness does not store the i -witnesses associated with each position b_i . However, the sequence is a witness only if the corresponding i -witnesses exist. Moreover, the colors in a witness are monotonically increasing for growing indices (and thus increase from right to left), since each color b_j (weakly) dominates all colors that appear afterwards ρ as a consequence of the dominating color property and the ordered sequences property.

3.1.2.1. Forward and backward witnesses

The forward witnesses described so far were introduced in Calude et al. (2017), while we now describe the backward witnesses and an ordering over them that have been introduced in Fearnley et al. (2019). For each play, prefix $\rho = v_1, v_2, \dots, v_m$, we define a reverse play $\overleftarrow{\rho} = v_m, v_{m-1}, \dots, v_1$; a backward witness is a witness for $\overleftarrow{\rho}$, or for a prefix of it.

3.1.2.2. Order on witnesses

The set $C__$ is ordered by the relation \succeq such that even numbers are better than odd numbers, higher even numbers are better than smaller even numbers, smaller odd numbers are better than higher odd numbers, and every number is better than $_$. Formally, $a \succeq b$ if $b = _$; or a is even and b is either odd or $_$; or $a \leq b$ and they are both odd.

Using \succeq , we define an order \sqsupset' over witnesses that compares two witnesses of the same size lexicographically, where the most significant element is b_k and the least significant element is b_0 . Each element is compared using the order \succeq . The biggest witness has a special value WON; i.e., $\text{WON} \sqsupset' \mathbf{b}$ holds for all witnesses \mathbf{b} .

3.1.2.3. The value of a witness

While there is, in principle, a countable set of witnesses, it suffices to take into consideration only those witnesses that do not require the existence of an even chain that is longer than the number states with even color. With this in mind, we define the value of a witness as the length of an even chain.

For example, $\mathbf{b} = 8, 5, 2$ is a forward witness for a run prefix with a color trace 9, 6, 7, 8, 7, 2, 8, 3, 2, 4, 5, 3, 2, 3, 2, where the red, blue, and green color are used to visualize the even

chains (extended, for the sequence in blue). A run for which \mathbf{b} is a forward witness always includes an even chain of length 6, which consists of the even chain of the leading positions with even priority (in this case: 8) as well as the even chain defined by the first odd color (in this case: 5). For example, this is the even chain shown in cyan: 9, 6, 7, 8, 7, 2, 8, 3, 2, 4, 5, 3, 2, 3, 2.

Formally, we define the following functions for each witness $\mathbf{b} = b_k, b_{k-1}, \dots, b_0$:

- **Even positions:** $\text{even}(\mathbf{b}) = \{i \in \mathbb{N}_0 \mid b_i \text{ is an even number}\}$
(in the example above, these are positions with index 2 and 0, whose label is 8 and 2, respectively);
- **Relevant i -witnesses:** $\text{evenodd}(\mathbf{b}) = \text{even}(\mathbf{b})$ if \mathbf{b} does not contain an odd number,
otherwise, $\text{evenodd}(\mathbf{b}) = \{i \in \text{even}(\mathbf{b}) \mid i > o\} \cup \{o\}$, with $o = \max\{i \in \mathbb{N} \mid b_i \text{ is odd}\}$;
(in the example above, these are positions with index 2 and 1, whose label is 8 and 5, respectively); and
- **Value of witness:** $\text{value}(\mathbf{b}) = \sum_{i \in \text{evenodd}(\mathbf{b})} 2^i$;
(in the example above, this is equal to 6).

Remark. The value function from Fearnley et al. (2019) is different from the one defined above., as it uses $\sum_{i \in \text{even}(\mathbf{b})} 2^i$. The latter is the sum of the length of the even chains that refer to even entries in the witness.

In the example above, the value function from Fearnley et al. (2019) is the sum of the concatenated even chains in red and green, with a joint length of 5. We will discuss the impact that this difference has on the statespace at the end of Section 6.

We can show that the value of \mathbf{b} corresponds to the length of an even chain in ρ that is witnessed by \mathbf{b} .

Lemma 1. Fearnley et al. (2019) If \mathbf{b} is a (forward or backward) witness of ρ , then there is an even chain of length $\text{value}(\mathbf{b})$ in ρ .

If we count the number of vertices with even colors in the game as $e = |\{v \in V : \phi(v) \text{ is even}\}|$, then we can observe that in case we have an even chain longer than e then ρ contains a cycle, as there is a vertex with even color visited twice in this even chain. Moreover, the cycle is winning for the player even, since the largest priority of its vertices must be even. As a consequence, if the player even can force a play that has a witness whose value is strictly greater than e , even wins the game.

Lemma 2. Fearnley et al. (2019) If, from an initial state v_0 , the player even can force the game to run through a sequence ρ , such that ρ has a (forward or backward) witness \mathbf{b} such that $\text{value}(\mathbf{b})$ is greater than the number of vertices with even color, then player even wins the parity game starting at v_0 .

For this reason, we only need witnesses with value $\leq e$, as every witness of value $> e$ must contain a winning cycle.

We will refer to the set of classic witnesses as $\mathbb{W} = \{\mathbf{b} \mid \mathbf{b} \text{ is a witness with } \text{value}(\mathbf{b}) \leq e\} \cup \{\text{won}\}$.

3.2. Updating witnesses

Forward witnesses can be constructed incrementally by processing the play one vertex at a time. The following lemmas assume that we have a play prefix $\rho = v_0, v_1, \dots, v_m$, and a new vertex v_{m+1} that we are going to append to ρ in order to create ρ' . The value $d = \phi(v_{m+1})$ denotes the color of the new vertex v_{m+1} . We will suppose that $\mathbf{b} = b_k, b_{k-1}, \dots, b_1, b_0$ is a witness for ρ , and we will construct a witness $\mathbf{c} = c_k, c_{k-1}, \dots, c_1, c_0$ for ρ' .

We present three lemmas that allow us to perform this task.

Lemma 3. *Fearnley et al. (2019)* Suppose that d is even, there exists an index j such that:

- b_i is even for all $i < j$,
- b_j is odd or equal to $_$ and
- $b_i \geq d$ or equal to $_$ for all $i > j$.

If we set $c_i = b_i$ for all $i > j$, $c_j = d$, and $c_i = _$ for all $i < j$, then \mathbf{c} is a witness for ρ' .

Note that we returned to the original definition from Calude et al. (2017) by restricting this updating rule from Lemma 3, called “overflow rule,” to even numbers, whereas the witnesses from Fearnley et al. (2019) also allowed this operation to be performed in the case where d is odd. The reason for this change is that it reduces the statespace: while this reduction is insignificant in most cases, it is quite substantial if $e = 2^p - 1$ for some power $p \in \mathbb{N}$, as it leads to an increase in the length of the witness. Since statespace reduction is a core target of this article, we opted to be precise here. A full discussion about the statespace is reported in Section 6.

Note that the next lemmas (and their proofs) are essentially independent of the variation of Lemma 3 with respect to the version reported in Fearnley et al. (2019).

Lemma 4. *Fearnley et al. (2019)* Suppose that $d \in C^-$ and there exists an index j such that:

- $d > b_j \neq _$ and
- $b_i \geq d$ or equal to $_$ for all $i > j$.

Then setting $c_i = b_i$ for all $i > j$, setting $c_j = d$ if $j \neq 0$ (and $c_j = _$ if $j = 0$), and setting $c_i = _$ for all $i < j$ yields a witness for ρ' .

There is a tiny difference in the proof of this lemma with the one from Fearnley et al. (2019), as we require the length of the j -witness to be $2^j + 1$ when c_j is set to d . But either b_j was odd

before, in which case replacing the last index of the j -witness by $m + 1$ still produces a witness of length $2^j + 1$, or it was even, and in that case, we can instead append $m + 1$ to the old j -witness.

Lemma 5. *Fearnley et al. (2019)* Suppose that $d \in C^-$ is odd and, for all $j \leq k$, either $b_j = _$ or $b_j \geq d$. If we set $c_i = b_i$ for all $i \leq k$ (i.e., if we set $\mathbf{c} = \mathbf{b}$), then \mathbf{c} is a witness for ρ' .

When we want to update a witness with the raw update rule upon scanning another state v_{m+1} with color $d = \phi(v_{m+1})$, we select the according lemma if $d \in C^-$. Otherwise, i.e., when $d = \max\{C\}$ and odd, we re-set the witness to $_, \dots, _$ (which is a witness for every play prefix).

For a given witness \mathbf{b} and a vertex v_{m+1} , we denote with

- **Raw update:** $\text{ru}'(\mathbf{b}, d)$ the raw update of the witness to \mathbf{c} , as obtained by the update rules described above.
- **Update:** $\text{up}'(\mathbf{b}, d)$ is either $\text{ru}'(\mathbf{b}, d)$ if $\text{value}(\text{ru}'(\mathbf{b}, d)) \leq e$ (where e is the number of vertices with even color), or $\text{up}'(\mathbf{b}, d) = \text{won}$, otherwise.
In particular, $\text{up}'(\text{won}, d) = \text{won}$ holds for all $d \in C$.
- **Antagonistic update:** $\text{au}'(\mathbf{b}, d) = \min_{\mathbf{c} \in \mathbb{W}} \{\text{up}'(\mathbf{c}, d) \mid \mathbf{b} \sqsubseteq' \mathbf{c} \in \mathbb{W}\}$.

3.2.1. Basic and antagonistic update game

With these update rules, we define a forward and a backward basic update game played between the two players *even* and *odd*. In this game, they produce a play of the game as usual: if the pebble is in a position assigned to even, then *even* selects a successor, and if the pebble is in a position assigned to odd, then *odd* selects a successor.

Player *even* can stop any time he likes and evaluate the game using $\mathbf{b}_0 = _, \dots, _$ as a starting point and the update rule $\mathbf{b}_{i+1} = \text{up}'(\mathbf{b}_i, v_i)$ (in the basic update game) and $\mathbf{b}_{i+1} = \text{au}'(\mathbf{b}_i, v_i)$ (in the antagonistic update game), respectively.

For a forward game, *even* would process the partial play $\rho^+ = v_0, v_1, v_2, \dots, v_n$ from left to right, and for the backward game he would process the partial play $\rho^- = v_n, v_{n-1}, \dots, v_0$. In both cases, *even* has won if, and only if, $\mathbf{b}_{n+1} = \text{won}$.

Theorem 1. *Fearnley et al. (2019)* Player *even* has a strategy to win the parity game if, and only if, *even* has a strategy to win the classic forward, respectively, backward basic, respectively, antagonistic update game.

This can be formulated in a way that, for a given set C of colors and e states with even color, the deterministic reachability automaton with states \mathbb{W} , initial state $_, \dots, _$, update rules up' (or au'), and reachability goal to reach won is a separating automaton.

Corollary 1. For a parity game with e states and $k = \lfloor \log_2(e) \rfloor$ and \mathbb{W} the space for witnesses of value $\leq e$, length $k + 1$

and colors C , both $\mathcal{U} = (\mathbb{W}; C; _, \dots, _ ; \text{up}' ; \text{won})$ and $\mathcal{A} = (\mathbb{W}; C; _, \dots, _ ; \text{au}' ; \text{won})$ are separating automata. \square

The advantage of the antagonistic update rule is that it is monotone: $\mathbf{b} \sqsubseteq' \mathbf{c} \rightarrow \text{au}'(\mathbf{b}, d) \sqsubseteq \text{au}'(\mathbf{c}, d)$. This allows for using au' in a value iteration algorithm (Fearnley et al., 2019).

4. Concise witness

In this article, we suggest a change in the semantics of the witness, and a related reduction of the statespace of witnesses to $\mathbb{C} \subsetneq \mathbb{W}$. While this section focuses on condensing the statespace, in the next one we will adjust the semantics and improve the update rule.

The main theoretical advancement is the smaller statespace we will define in this section, as it directly translates into improved bounds, slightly outperforming the currently leading QP algorithm in this regard.

Toward this goal, we define a truncation operator

$$\downarrow_1 : \mathbb{W} \rightarrow \mathbb{C}$$

that, for every odd color $o \in C^-$, leaves only the leftmost occurrences of o in a witness and replaces all other occurrences of o in \mathbf{b} by $_$.

For example, $\downarrow_1 _ , 7, _ , 7, 5, 4, _ , 3, 3, _ , 2 = _ , 7, _ , 5, 4, _ , 3, _ , 2$, and $\downarrow_1 3, 3, 2 = 3, _ , 2$. We also have $\downarrow_1 \text{won} = \text{won}$.

Note that the definition of \downarrow_1 entails

$$\text{even}(\downarrow_1 \mathbf{b}) = \text{even}(\mathbf{b}),$$

as well as

$$\text{value}(\downarrow_1 \mathbf{b}) = \text{value}(\mathbf{b}).$$

Building on the definition of \downarrow_1 , we continue with the following definitions.

- $\mathbb{C} = \{\downarrow_1 \mathbf{b} \mid \mathbf{b} \in \mathbb{W}\}$,
- **Raw update:** $\text{ru}(\mathbf{b}, d) = \downarrow_1 \text{ru}'(\mathbf{b}, d)$,
- **Update:** $\text{up}(\mathbf{b}, v) = \downarrow_1 \text{up}'(\mathbf{b}, d)$,
- **Order over witnesses:** for all $\mathbf{b}, \mathbf{c} \in \mathbb{C}$, $\mathbf{b} \sqsubseteq \mathbf{c}$ if, and only if, $\mathbf{b} \sqsubseteq' \mathbf{c}$ (i.e., \sqsubseteq is simply a restriction of \sqsubseteq' from \mathbb{W} to \mathbb{C}), and
- **Antagonistic update:** $\text{au}(\mathbf{b}, d) = \min_{\sqsubseteq} \{\text{up}(\mathbf{c}, d) \mid \mathbf{b} \sqsupseteq \mathbf{c} \in \mathbb{C}\}$

To justify the use of the concise witness space \mathbb{C} , we first show that, for all witnesses, $\mathbf{c} \in \mathbb{W}$ in the *old* witness space, it holds that $\downarrow_1 \text{ru}'(\mathbf{c}, d) = \downarrow_1 \text{ru}'(\downarrow_1 \mathbf{c}, d)$. Thus, if we truncate *only* after, or both before *and* after, a raw update does not change the result.

Lemma 6. *If $\mathbf{b} = \downarrow_1 \mathbf{c}$, then $\text{ru}(\mathbf{b}, d) = \downarrow_1 \text{ru}'(\mathbf{c}, d)$.*

Proof. We look at the effect the different update rules have on \mathbf{b} and \mathbf{c} . Lemma 3 would (for the same j) change the tail (starting with the j -witness) of \mathbf{c} and \mathbf{b} in the same way to $d, _ , \dots, _$, and they either both do or do not satisfy the prerequisites for its application. Thus, the \downarrow_1 operator would remove exactly those positions $> j$ from $\text{ru}'(\mathbf{c}, d)$ that it removed from \mathbf{c} .

When Lemma 4 applies, then it does so for the same index j , and it simply overrides the tail starting there with $d, _ , \dots, _$ (or with $_$ if $j = 0$). As all higher positions are unchanged and greater or equal to d , $\mathbf{b} = \downarrow_1 \mathbf{c}$ implies $\text{ru}(\mathbf{b}, d) = \downarrow_1 \text{ru}'(\mathbf{c}, d)$.

When the conditions of Lemma 5 apply either for both, \mathbf{b} and \mathbf{c} or for neither of them, then we note that Lemma 5 does not change the witness.

Finally, if $d = \max\{C\}$ and is also odd, then $\text{ru}'(\mathbf{b}, d) = \text{ru}'(\mathbf{c}, d) = _ , \dots, _$, which implies $\text{ru}(\mathbf{b}, d) = \downarrow_1 \text{ru}'(\mathbf{b}, d) = \downarrow_1 _ , \dots, _ = \downarrow_1 \text{ru}'(\mathbf{c}, d)$. \square

This immediately extends to the update rule up/up' , as they differ from ru/ru' only in treating *won*, and to au/au' by monotonicity.

Corollary 2. *If $\mathbf{b} = \downarrow_1 \mathbf{c}$, then $\text{up}(\mathbf{b}, v) = \downarrow_1 \text{up}'(\mathbf{c}, v)$ and $\text{au}(\mathbf{b}, v) \sqsupseteq \downarrow_1 \text{au}'(\mathbf{c}, v)$.* \square

The observation that $\downarrow_1 \text{up}'(\downarrow_1 \mathbf{c}, v) = \text{up}'(\downarrow_1 \mathbf{c}, v) = \downarrow_1 \text{up}'(\mathbf{c}, v)$ can be extended to every run prefix: truncating in every step and truncating at the end has the same effect. Thus, by a simple inductive argument, the state *won* is reached with up and up' at the same time (or not at all in either case).

Theorem 2. *The player even has a strategy to win the parity game if, and only if, even has a strategy to win the concise forward / backward basic update game.*

Proof. This follows from Theorem 1: because the same runs are winning when using up and up' due to Corollary 2, the same player wins the classic and the concise basic update game. \square

Theorem 3. *The player even has a strategy to win the parity game if, and only if, even has a strategy to win the concise forward / backward antagonistic update game.*

Proof. For the 'if' case, we observe that Corollary 2 implies with the monotonicity of au that, when *even* wins the classic antagonistic update game, *even* also wins the concise antagonistic update game (with the same strategy). Together with Theorem 1, this provides the "if" case.

For the "only if" case, we observe that the monotonicity of au entails that, when *odd* wins the basic concise update game, then *odd* wins the antagonistic update game (with the same strategy). Together with Theorem 2, this provides the "only if" case. \square

Corollary 3. *For a parity game with e states of even color and $k = \lfloor \log_2(e) \rfloor$ and \mathbb{W} the space for witnesses of value $\leq e$, length $k + 1$ and colors C , both $\mathcal{U} = (\mathbb{C}; C; _, \dots, _ ; \text{up} ; \text{won})$ and $\mathcal{A} = (\mathbb{C}; C; _, \dots, _ ; \text{au} ; \text{won})$ are separating automata.* \square

To give intuition to their states, if \mathcal{U} is in a state $\mathbf{b} \neq \text{won}$, it means that \mathbf{b} is a witness for the play prefix, while won means that the play prefix contains an even chain of length $> e$, and thus an even cycle.

If \mathcal{A} is in a state \mathbf{b} then there is a state $\mathbf{c} \sqsupseteq \mathbf{b}$ with this property.

As a final remark, in the rare cases where even colors are scarce, their appearance in \mathbb{C} can also be restricted: if only $e_{\#}$ states have an even color e , then the number of occurrences of e in a concise witness can be capped to $e_{\#}$, too, as more occurrences of e without an intermediate occurrence of a higher color would imply that an accepting cycle is in the word.

However, for this to reduce the statespace, $e_{\#} \leq \lfloor \log_2(e) \rfloor$ is required, and the closer it comes to $\lfloor \log_2(e) \rfloor$, the lesser the saving. In particular, for $e_{\#} = \lfloor \log_2(e) \rfloor$, we would just save a single state.

5. Color witnesses

In this section, we use the same data structure as before—the concise witnesses from the previous section—but adjust its semantics.

We introduce two changes to the semantics of witnesses that accelerate the progress to victory for *even* when update operations are made. The changes are applied to the concise witnesses \mathbb{C} .

Before formalizing how we make our witnesses more flexible and how we use this to re-define the raw update function (and, through this, the update function and the antagonistic update), we describe several examples of how we change the semantics of witnesses.

5.1. Motivating examples

5.1.1. Merging witnesses

If we consider the classic witness $\mathbf{b} = 4, 4, _$, it referred to two i -witnesses that each end on a state with color 4, one of length 4 and one of length 2.

For example, $\mathbf{b} = 4, 4, _$ is a classic forward witness for a run prefix with color trace 9, 6, 7, 8, 7, 2, 4, 3, 2, 4 (where the **red** and **blue** color are used to visualize the even chains), whereas 9, 6, 7, 8, 7, 2, 2, 3, 4, 4 is not.

We will instead view this as a *single* color witness for color 4, which then refers to a single even chain of length *at least* 6.

For example, $\mathbf{b} = 4, 4, _$ is a forward *color* witness for a run prefix with color trace 9, 6, 7, 8, 7, 2, 2, 3, 4, 4: as we are content with an even chain of length 6, it does not matter that the fourth position of this chain has an entry different to 4.

As a consequence, when passing by a state with the color 6, we can now update the witness to 6, 6, 6, as this would require a single even chain of at least length 7 that ends in a 6.

5.1.2. Shifting witnesses

If we consider the witness $\mathbf{b} = 4, 2, _$, it referred to two i -witnesses, where the first has length 4 and ends on color 4, while the second has length 2 and ends on color 2.

We will allow making the latter sequence shorter, so long as the former sequence is extended accordingly. For example, when the sequence that ends in 4 has length 5, then it would suffice, for the witness to represent the even chains if the sequence that ends in 2 has length 1.

For example, $\mathbf{b} = 4, 2, _$ is a forward color witness for a run prefix with color trace 9, 6, 7, 8, 7, 2, 2, 4, 3, 2 (where the **red** and **blue** colors are used to visualize the even chains)³.

Similarly, for $\mathbf{b} = 6, _, 4, 2, 2$, it would be allowed that the length of the even chain that ends in 6 is 18, the subsequent sequence that ends in 4 is 3, and the length of the sequence that ends in 2 is 2. If the length of the sequences ending in 6, 4, and 2 are ℓ_6, ℓ_4 , and ℓ_2 , respectively, the constraints would be $\ell_6 \geq 16$, $\ell_6 + \ell_4 \geq 20$, and $\ell_6 + \ell_4 + \ell_2 \geq 23$.

When passing by a state with the color 8, we can now update the witness to 8, 8, $_, _, _$, as this would require a single sequence of at least length 24 that ends in an 8.

5.1.3. Blocked shifting

This shifting cannot be done through an odd color: for $\mathbf{b} = 4, 3, 2, 2$, the requirement for the rightmost sequence would be to be of length at least three and to end in a 2. It is, however, possible to shift some of the required lengths of the sequence that ends in 3 to the sequence that ends in 4: if the length of the sequences ending in 4 and 3 are ℓ_4 and ℓ_3 , respectively, the constraints would be $\ell_4 \geq 8$ and $\ell_4 + \ell_3 \geq 13$. (Recall the odd witnesses need to be one position longer to contain an even chain of the same length.) Thus, reading a 6 would lead to the witness 6, 6, $_, 6$.

5.2. Color witness

The biggest change is that an i -color witness (i -cowit) refers to the *color* i , rather than to the position b_i in the witness. Consequently, we do not have a fixed length of an i -color witness and refer to the length of the witness for each color i that appears in a witness as ℓ_i .

As before, we focus in our description on forward witnesses, with backward witnesses being defined accordingly.

³ Note that $\mathbf{b} = 4, 2, _$ is not a classic forward witness for this sequence, whereas $\mathbf{b}' = 4, _, 2$ is: the **red even chain** can be shortened (by dropping one 2, e.g., to 9, 6, 7, 8, 7, 2, 2, 4, 3, 2) to an even chain of length 4, which would in and by itself be a 2-witness, whereas the **blue even chain** is merely a 0-witness.

5.2.1. i -color witness (i -cowit) with value ℓ_i

Let $\rho = v_1, v_2, \dots, v_m$ be a prefix of a play of the parity game. An *even i -cowit* is a sequence of (not necessarily consecutive) positions of ρ

$$p_1, p_2, p_3, \dots, p_{\ell_i}$$

of length exactly ℓ_i , and an *odd i -cowit* is a sequence of (not necessarily consecutive) positions of ρ

$$p_0, p_1, p_2, \dots, p_{\ell_i}$$

of length exactly $\ell_i + 1$, that satisfy the following properties:

- **Position:** Each p_j specifies a position in the play prefix ρ , so each p_j is a positive integer that satisfies $1 \leq p_j \leq m$.
- **Order:** The positions are ordered. Thus, we have $p_j < p_{j+1}$ for all $j < \ell_i$.
- **Evenness:** All positions but the final one are even. Formally, for all $j < \ell_i$ the color $\phi(v_{p_j})$ of the vertex in position p_j is even.
For position p_{ℓ_i} , its color $\phi(v_{p_{\ell_i}}) = i$. Then, the color of that position is even for *even i -cowit*, and odd for *odd i -cowit*.
- Note that this entails that an i -cowit has ℓ_i initial even positions that define an even chain of length ℓ_i .
- **Inner domination:** The color of every vertex between p_j and p_{j+1} is dominated by the color of p_j or the color of p_{j+1} . Formally, for all $j < \ell_i$, the color of every vertex in the subsequence $v_{p_j}, v_{p_{j+1}}, \dots, v_{p_{j+1}}$ is less than or equal to $\max\{\phi(v_{p_j}), \phi(v_{p_{j+1}})\}$.
- **Outer domination:** The color of the vertex $v_{p_{\ell_i}}$ in position p_{ℓ_i} is i , i.e., $i = \phi(v_{p_{\ell_i}})$. Moreover, i is greater than or equal to the color of every vertex that appears after position p_{ℓ_i} in ρ . Formally, for all k in the range $p_{\ell_i} \leq k \leq m$, we have that $\phi(v_k) \leq i$.

5.2.2. Color witnesses

Similar to a concise witness, a *color witness* is a sequence

$$b_k, b_{k-1}, \dots, b_1, b_0,$$

of length⁴ $k+1$, such that each element $b_i \in C_-$, and that satisfies the following properties.

- **Properties of the sequence:** defining **i -positions** as the positions in \mathbf{b} that have value i ,
 $\text{positions}(i, \mathbf{b}) = \{j \leq k \mid b_j = i\}$ for every $i \in C^-$, the sequence has to satisfy the following constraints:

⁴ $k = \lfloor \log_2(e) \rfloor$ again suffices, where e is the number of vertices with an even color.

- **order:** for $i > j$, we have that $b_i \geq b_j$ or $- \in \{b_i, b_j\}$ holds; and
- **conciseness:** for all odd $i \in C^-$, $|\text{positions}(i, \mathbf{b})| \leq 1$ and $b_0 \neq i$ hold.

• Witnessing:

- **ordered witnesses:** for $i > j$ with $\text{positions}(i, \mathbf{b}) \neq \emptyset$ and $\text{positions}(j, \mathbf{b}) \neq \emptyset$, the j -witness starts after the i -witness ends. That is $p_{i, \ell_i} < p_{j, 1}$ if j is even and $p_{i, \ell_i} < p_{j, 0}$ if j is odd.
- using the following definitions,

- **next odd color:** $\text{odd}(i, \mathbf{b}) = \inf\{j > i \mid j \text{ odd and } \text{positions}(j, \mathbf{b}) \neq \emptyset\}$ defines the next higher odd color than i that occurs in the color witness (note that $\text{odd}(i, \mathbf{b}) = \infty$ if no such color exists),

- **unblocked colors:** $\text{unblocked}(i, \mathbf{b}) = \{j < \text{odd}(i, \mathbf{b}) \mid j \geq i \text{ and } \text{positions}(i, \mathbf{b}) \neq \emptyset\}$ is the set of all colors that are at least i , but strictly smaller than $\text{odd}(i, \mathbf{b})$, and

- **unblocked positions:** $\text{ubp}(i, \mathbf{b}) = \bigcup_{j \in \text{unblocked}(i, \mathbf{b})} \text{positions}(i, \mathbf{b})$ is the set of positions labeled by an unblocked color,

$$\text{we have that } \sum_{\text{unblocked}(i, \mathbf{b})} \ell_i \geq \sum_{j \in \text{ubp}(i, \mathbf{b})} 2^j \text{ holds}$$

for all $i \in C^-$.

It should be noted that neither the i -cowit-s associated with each color, nor the value of the ℓ_i are stored in a color witness. However, in order for a sequence to be a color witness for an initial sequence of a run, the corresponding i -cowit-s must *exist*.

5.3. Updating color witnesses

We now show how forward color witnesses can be constructed incrementally by processing the play one vertex at a time. Throughout this subsection, we will suppose that we have a play $\rho = v_0, v_1, \dots, v_m$, and a new vertex v_{m+1} that we would like to append to ρ to create ρ' . We will use $d = \phi(v_{m+1})$ to denote the color of this new vertex. We will suppose that $\mathbf{b} = b_k, b_{k-1}, \dots, b_1, b_0$ is a color witness for ρ , and has i -cowit-s with individual lengths ℓ_i . We will construct a witness $\mathbf{c} = c_k, c_{k-1}, \dots, c_1, c_0$ for ρ' and discuss how its inferred i -cowit-s look like.

We present four lemmas that allow us to perform this task.

Lemma 7. *Suppose that $d \in C^-$ is odd and, for all $j \leq k$, either $b_j = -$ or $b_j > d$. If we set $c_i = b_i$ for all $i \leq k$, then \mathbf{c} is a color witness for ρ' .*

Proof. Since $d < b_j$ for all j , the outer domination of every e -color witness implied by \mathbf{b} is not changed. Moreover, no other

property of any e -color witness is changed by the inclusion of v_{m+1} in the initial sequence, thus, by setting $\mathbf{c} = \mathbf{b}$, we obtain a color witness for ρ' . \square

Note that the proof of Lemma 7 does not use that d is odd and holds similarly when d is even; however, in that case, Lemma 10 provides a better update for the color witness.

Lemma 8. *Suppose that $d \in C^-$ is odd, and there exists an index j such that $b_j \neq _$, $d \geq b_j$, and, for all $i > j$, either $b_i = _$ or $b_i > d$ hold. Then setting:*

- $c_i = b_i$ for all $i > j$,
- $c_j = d$ if $j \neq 0$ and $c_j = _$ if $j = 0$, and
- $c_i = _$ for all $i < j$

yields a color witness for ρ' .

Proof. For all $e > d$, the e -color witness (if any) implied by \mathbf{b} can be kept: the outer domination of every such e -color witness implied by \mathbf{b} is not changed. Moreover, no other property of any such e -color witness is changed by the inclusion of v_{m+1} in the initial sequence.

For the b_j -color witness, we either update the last vertex to $m + 1$ (if b_j is odd) or append $m + 1$ to it (if b_j is even). In both cases, the inner domination rules are valid (due to the inner and outer domination rules for the b_j -color witness) and the outer domination rule holds trivially. Moreover, the side constraints for the length carry over from those for \mathbf{b} (when b_j is odd), for ℓ_d , by adding one to the length constraint while also appending one state (when b_j is even).

Thus, \mathbf{c} is a color witness for ρ' . \square

Lemma 9. *Suppose that d is even, there exists a maximal index j such that $b_j < d$, and b_j is odd. Then setting:*

- for all $i \geq j$, $c_i = d$ if $b_i < d$ and $c_i = b_i$, otherwise,
- for all $j > i \geq 1$, $c_i = _$, and
- $c_0 = d$.

yields a color witness for ρ' .

Proof. We simply append all i -cowit-s that exist for \mathbf{b} in the interval $i \in \{b_j, \dots, d\}$. We append them in the given order (from the largest i to the lowest, b_j), and then replace the last index (which is from the b_j -covit) by $m + 1$.

The inner domination rules are valid (due to the inner and outer domination rules for the i -cowit-s involved, and by $b_j < d$). The outer domination rule trivially holds.

The only new rule to be considered is the rule on the joint length of the i -cowit-s in $\text{ubp}(d, \mathbf{c})$, but this is the same length (as only the last element is changed) and the same constraint as for the sum of the length of the i -cowit-s in $\text{ubp}(b_j, \mathbf{b})$. \square

Lemma 10. *Suppose that d is even and there is no index j' such that $b_{j'} < d$ and $b_{j'}$ are odd. Let j be the maximal index (which might be 0) such that:*

- for all $i > j$, b_i is even, $b_i = _$ or $b_i > d$;
- either $b_j = _$ or $b_j > d$ and b_j is odd; and
- for all $i < j$, b_i is even.

If we set:

- $c_i = b_i$ for all $i > j$ with $b_i > d$ or $b_i = _$
- $c_i = d$ for all $i > j$ with $b_i \leq d$ (and thus even),
- $c_j = d$, and
- for all $i < j$, $b_j = _$

then \mathbf{c} is a color witness for ρ' .

Proof. We simply append all i -cowit-s that exist for \mathbf{b} in the interval $i \in \{2, \dots, d\}$. We append them in the given order (from the largest i to the lowest), and then append $m + 1$.

The inner domination rules are valid (due to the inner and outer domination rules for the i -cowit-s involved). The outer domination rule trivially holds.

The only new rule to be considered is the rule on the joint length of the i -cowit-s in $\text{ubp}(d, \mathbf{c})$, but this is one more than the length (as only the last element is appended) and the same constraint as for the sum of the length of the i -cowit-s in $\text{ubp}(2, \mathbf{b})$. \square

Again, if $d = \max\{C\}$ and odd, then the raw update of the color witness is $_ \dots _$, which is a color witness for every play prefix.

When we want to update a witness upon scanning another state v_{m+1} with color $d = \phi(v_{m+1})$, we can apply the update rule from one of Lemmas 7 through 10.

For a given witness \mathbf{b} and a vertex v_{m+1} , we denote with

- **Raw update:** $\text{ru}_+(\mathbf{b}, d)$ the raw update of the witness to \mathbf{c} , as obtained by the update rules described above.
- **Update:** $\text{up}_+(\mathbf{b}, d)$ is either $\text{ru}_+(\mathbf{b}, d)$ if $\text{value}(\text{ru}(\mathbf{b}, d)) \leq e$ (where e is the number of vertices with even color), or $\text{up}_+(\mathbf{b}, v_{m+1}) = \text{won}$ otherwise.

In particular, $\text{up}_+(\text{won}, d) = \text{won}$ for all $d \in C$.

- **Antagonistic update:** $\text{au}_+(\mathbf{b}, v) = \min_{\mathbf{c} \sqsubseteq \mathbf{c} \in C} \{\text{up}_+(\mathbf{c}, v) \mid \mathbf{b} \sqsubseteq \mathbf{c} \in C\}$.

We first observe that up_+ is indeed “faster” than up in that it always leads to a better (with respect to \sqsubseteq) state:

Lemma 11. *For all $\mathbf{b} \in C$ and all $d \in C$, $\text{up}_+(\mathbf{b}, d) \sqsupseteq \text{up}(\mathbf{b}, d)$.* \square

This is easy to check by the raw update rules, and it entails:

Corollary 4. *For all $\mathbf{b} \in C$ and all $d \in C$, $\text{au}_+(\mathbf{b}, d) \sqsupseteq \text{au}(\mathbf{b}, d)$.* \square

Theorem 4. *The following three claims are equivalent for both, forward and backward witnesses:*

1. *player even has a strategy to win the parity game,*
2. *player even has a strategy to win the fast basic update game (using up_+), and*
3. *player even has a strategy to win the fast antagonistic update game (using au_+).*

Proof. (1) implies (3): By Theorem 3, that *player even* wins the parity game entails that he wins the concise antagonistic update game. As au_+ provides (not necessarily strictly) better updates (with respect to \sqsubseteq) than au , and by the antagonistic update being monotone by definition, this entails (3).

(3) implies (2): as up_+ provides (not necessarily strictly) better updates (with respect to \sqsubseteq) than au_+ and au_+ is monotone when au^+ produces a winning sequence, so does up^+ .

(2) implies (1): when up^+ produces a win if, and only if, ru^+ produces a color witness with value $> e$, which according to Lemmas 7 through 10 entails that it has an even chain whose length is strictly greater than e . The play ρ must, at that point, contain a cycle, since there must be a vertex with even color that has been visited twice. Moreover, the largest priority on this cycle must be even, so this is a winning cycle for the *player even*. \square

Corollary 5. *For a parity game with e states of even color, colors C , $k = \lfloor \log_2(e) \rfloor$ and \mathbb{C} the space for concise witnesses of value $\leq e$, length $k+1$ and colors C , both $\mathcal{U} = (\mathbb{C}; C; _, \dots, _ ; \text{up}_+; \text{won})$ and $\mathcal{A} = (\mathbb{C}; C; _, \dots, _ ; \text{au}_+; \text{won})$ are separating automata.* \square

To give intuition to their states, for \mathcal{U} being in a state $\mathbf{b} \neq \text{won}$ means that \mathbf{b} is a color witness for the play prefix, while won means that the play prefix contains an even chain of length $> e$, and thus an even cycle.

For \mathcal{A} being in a state \mathbf{b} means that there is a state $\mathbf{c} \sqsupseteq \mathbf{b}$ with this property.

5.4. Faster conversion

While the method will speed up the progress to the solution made by the update operations when using $\mathcal{A} = (\mathbb{C}; C; _, \dots, _ ; \text{au}_+; \text{won})$ a little, the difference is easier to see when using $\mathcal{U} = (\mathbb{C}; C; _, \dots, _ ; \text{up}_+; \text{won})$.

The classic QP algorithms (Calude et al., 2017; Jurdziński and Lazic, 2017; Fearnley et al., 2019) have very simple pathological examples. For example, Jurdziński and Lazic (2017) would traverse the complete statespace for a state with color 2 and a selfloop (or for a state with color 1 and a selfloop, depending on whose player's side the algorithm takes). Similarly, Calude et al. (2017) would traverse very large parts of its statespace when fed with only even colors.

Using our update rules for color witnesses, a loop with even colors will always lead to acceptance within $e + 1$ steps.

It is possible to make the update rules a bit more robust against the occurrence of odd priorities that are then immediately followed by higher even priorities by returning to \mathbb{W} as a statespace⁵.

6. Statespace

In this section, we compare the size of the statespace with both the statespace from the construction of Jurdziński and Lazic (Jurdziński and Lazic, 2017)—which comes with the best current bounds— and the original statespace from Calude et al. (2017).

We then discuss the effect of the five improvements over the original approach from Calude et al. (2017):

1. the restriction of the number of occurrences of odd colors in a witness to once,
2. not using any color that is higher than any even color;
3. not allowing for odd colors in the rightmost position (i.e., b_0);
4. the removal of the color 1; and
5. moving from length to value restriction.

The first of these improvements are, individually, the most powerful one. Three of the other improvements, (2), (3), and (4), have already been discussed in this form in Fearnley et al. (2019).

We will show in Subsection 6.3.1 that applying *only* improvements (1)—the progression from \mathbb{W} to \mathbb{C} from Section 4—and (2) leads to a statespace of exactly the same size as that of Jurdziński and Lazic (2017).

Consequently, further improvements, (3)–(5), lead to a strictly smaller statespace. The improvement from (3) alone almost halves the statespace, while (4) alone has only a small effect. The effect of rule (5) — which refers to the change of the functions `evenodd` and `value` described in Section 3.1.2.3 — varies greatly: it is strongest when the bound on the length of an even chain is a power of 2 (2^p for some $p \in \mathbb{N}$), where it leads to halving the statespace, and vanishes if it is one less ($2^p - 1$ for some $p \in \mathbb{N}$).

After briefly visiting the statespace from Fearnley et al. (2019), we then turn to an experimental comparison of the three statespaces of interest, confirming the quantification of the advantage we have obtained over (Jurdziński and Lazic, 2017).

⁵ When using \mathbb{W} , there would need to be some care taken with odd σ -witnesses: while the rules for overwriting lower numbers are as expected, the point to bear in mind that the treatment of odd colors that already occur in a witness (covered by Lemma 8) generalize to if there is already a lowest position j with $b_j = \sigma$, then just replace all b_i with $i < j$ by $_$. This is because, while two even color witnesses can be merged, two odd color witnesses cannot, and there would be no means to mark them as different color witnesses.

In this section, we use $\text{count}_{alg}^{size}$ for counting the number of state minus one, estimating the number of states except for the winning state (“won”), which all progress measures under consideration have. The superscript *size* can be ℓ , saying that only the length of the data structure (or: the $\lceil \log_2(e + 1) \rceil$ for the maximal length e of an even chain) is taken into account; v if the value of witness is taken into account (or: the maximal length e of an even chain) is taken into account, and ℓ, v if both are used. The subscript is either JL when counting the concise progress measures from Jurdziński and Lazic (2017), O when considering the original approach from Calude et al. (2017), ‘1, 2’ when adding improvements (1) and (2), or blank when considering either improvement (1) through (4) or all improvements. In a closing comparison with the statespace of Fearnley et al. (2019), we use the subscript $JKSSW$.

6.1. Concise progress measures

We will not describe the algorithm, but the data structure, which holds a winning state besides the states we describe. For each even priority, there is a (possibly empty) word over two symbols, say + and −, such that the words concatenated have length at most $\ell = \lceil \log_2(e + 1) \rceil$, where e is the number of states with an even priority. That is, ℓ is the length of the witness and color witness from the previous section ($\ell = k + 1$).

For Jurdziński and Lazic (2017) (i.e., $alg = JL$), with c priorities $\{1, \dots, c\}$ and n states with even priority (not counting the winning state) we have the following counts:

- Induction basis, length: we start with the case in which we bound the sum of the lengths of + and − by 0 or 1.

When we bound the sum of the lengths by 0, then there is only one sequence:

$$\text{count}_{JL}^{\ell}(c, 0) = 1,$$

and when we bound it by 1, then we get:

$$\text{count}_{JL}^{\ell}(2c, 1) = \text{count}_{JL}^{\ell}(2c + 1, 1) = 2c + 1,$$

as there are c positions in which a sequence of length 1 can occur (one for each even priority in $\{1, \dots, 2c + 1\}$ or $\{1, \dots, 2c\}$, respectively), and there is one for the case in which all sequences have length 0.

- Induction basis, colors: when there is only one even color (i.e., 2), we have:

$$\text{count}_{JL}^{\ell}(3, l) = \text{count}_{JL}^{\ell}(2, l) = 2^{l+1} - 1.$$

These are the binary words of length at most l .

- For all other cases, we define inductively:

$$\begin{aligned} \text{count}_{JL}^{\ell}(2c + 1, l) &= \text{count}_{JL}^{\ell}(2c, l) + \text{count}_{JL}^{\ell}(2c - 2, l) \\ &\quad + 2\text{count}_{JL}^{\ell}(2c, l - 1), \end{aligned}$$

where the first summand refers to the case where the leading sequence (which refers to color $2c$) is empty. In this case, the length of the remaining sequences is still bound by l , but the number of even colors has dropped by one. The two summands $\text{count}_{JL}^{\ell}(2c, l - 1)$ represent the cases, where the sequence assigned to the highest even priority starts with a + and −, respectively. Cutting off this leading sign leaves $\text{count}_{JL}^{\ell}(2c, l - 1)$ in different states.

To estimate the number of concise progress measures, $\text{count}_{JL}^{\ell}(c, l) + 1$, we put aside the winning state and the function that maps all even priorities to the empty sequence.

For the remaining states, we first fix a positive length $i \leq l$ of the concatenated words, and then the $j \leq \lfloor c/2 \rfloor$ of the even priorities that have a non-empty word assigned to them.

There are $\binom{i-1}{j-1}$ assignments of positive lengths to j positions that add up to i . For each distribution of lengths, there are 2^i different assignments to words. Finally, there are $\binom{\lfloor c/2 \rfloor}{j}$ possibilities to assign j of the $\lfloor c/2 \rfloor$ different even priorities.

This provides an overall statespace of

$$2 + \sum_{i=1}^l \sum_{j=1}^{\min\{i, \lfloor c/2 \rfloor\}} 2^i \cdot \binom{\lfloor c/2 \rfloor}{j} \cdot \binom{i-1}{j-1}.$$

6.2. Calude et al.

‘We now continue with the statespace of the original quasi-polynomial approach of Calude et al. (2017), hence, $alg = O$. The statespace used in Calude et al. (2017) is slightly larger than \mathbb{W} , as it only uses the length of the witness as a restriction and does not exclude odd values for the rightmost position (“ b_0 ”) in a witness.

For the precise count of this statespace without the winning state “won,” we have the following counts:

- Induction basis, length: sequences of length 1 (only containing b_0) whose color values are bounded by c , can take $c + 1$ values in $\{1, \dots, c\}$ plus $_$. We, therefore, have:

$$\text{count}_O^{\ell}(c, 1) = c + 1.$$

- For longer tails of sequences, we define inductively:

$$\text{count}_O(c, l + 1) = \text{count}_O(c, l) + \sum_{i=1}^c \text{count}_O(i, l).$$

The summands refers to the possible values taken by the leftmost position (“ b_l ”) of the tail $(b_l, b_{l-1}, \dots, b_0)$. The first summand refers to the leading position being $_$ (“ $b_l = _$ ”). This does not restrict the values the rest of the tail may take any further as the highest color allowed to appear is still c . The other summands refer to the value i (“ $b_l = i$ ”). When the value of the leftmost position is $i \leq c$, then the highest color that may occur in the remaining positions is i .

The size of $|\mathbb{W}^+|$ of the statespace can be given as:

$$2 + \sum_{i=1}^l \binom{l}{i} \cdot \binom{i+c-1}{i}.$$

The “2” refers to the winning state and the “empty” sequence consists only of $_$ symbols (which is more convenient for us to treat separately), and the sum refers to the states represented by non-empty sequences of length $l = \lceil \log_2(e + 1) \rceil$, where e is the number of states with an even priority. Note that the estimation given in Calude et al. (2017) is slightly coarser, and their game definition is slightly different from the normal definition of parity games, but the bound can be taken from Fearnley et al. (2019).

For the estimation, after fixing the positive $i \leq l$ positions with values different to $_$, there are $\binom{i+c-1}{i}$ different valuations when we have c priorities. For each $i \leq l$, there are $\binom{l}{i}$ different choices for the i positions containing some number $d \in C$. This leads to $\sum_{i=1}^l \binom{l}{i} \cdot \binom{i+c-1}{i}$ different states that contain $i \leq l$ positions that have a value in C .

To obtain a better inroad to outline the differences, we first take a closer look at the $\binom{i+c-1}{i}$ different valuations that we may have for c priorities when we have fixed the $i \leq l$ positions that are not marked as $_$.

For these positions, we can look at the cases where there are $j \leq \min(i, r)$ fixed different priorities. For that case, there are $\binom{i-1}{j-1}$ many assignments of these j priorities to the i positions. Moreover, there are $\binom{c}{j}$ different options to select j of the available r priorities. Thus, we get

$$\binom{i+c-1}{i} = \sum_{j=1}^{\min\{i,c\}} \binom{c}{j} \cdot \binom{i-1}{j-1}$$

different combinations for all number of priorities put together, providing the following size:

$$2 + \sum_{i=1}^l \sum_{j=1}^{\min\{i,c\}} \binom{l}{i} \cdot \binom{c}{j} \cdot \binom{i-1}{j-1}.$$

6.3. Improvements

We now discuss the differences obtained when moving from \mathbb{W}^+ to \mathbb{C} by looking at the effect of the three optimisations we have introduced. These are:

1. the restriction of the number of occurrences of odd colors in a witness to once,
2. not using any color that is higher than any even color;
3. not allowing for odd colors in the rightmost position (“ b_0 ”);
4. the removal of the color 1; and
5. moving from length to value restriction.

6.3.1. (1) and (2) Restricted occurrence of odd colors

Restricting the occurrence of odd colors to once, together with the optimization of not using any color that is higher than any even color, leads to a situation where the highest color allowed in any position is even. To see this, we observe that the banning of a potential odd color higher than any even color guarantees this initially, where the highest color allowed is the highest even color.

When an odd color o is used in the witness (“ $b_l = o$ ”), then the highest color allowed to its right is $o - 1$, whereas when an even color e is used in the witness (“ $b_l = e$ ”), then the highest color allowed to its right is e .

We, therefore, only have to define our improved counting function for even colors:

- Induction basis, length: apart from using only even bounds, the base case remains the same:

$$\text{count}_{1,2}^\ell(2c, 1) = 2c + 1.$$

- For longer tails of sequences, we define inductively:

$$\text{count}_{1,2}^\ell(2c, l + 1) = 1 + 2 \sum_{i=1}^c \text{count}_{1,2}^\ell(2i, l).$$

The summands refer to the possible values taken by the leftmost position (“ b_l ”) of the tail $(b_l, b_{l-1}, \dots, b_0)$.

The first summand refers to the leading position being 1 (“ $b_l = 1$ ”). If this is the case, then all entries to its right must be *strictly smaller* than 1 (which is not possible) or $_$ —consequently, they must all be $_$, which just leaves one such tail.

The other summands refer to the leading position taking the value $2i$ or $2i + 1$ when $i < c$ (“ $b_l = 2i$ ” or “ $b_l = 2i+1$ ”), in either case, the maximal value of the colors occurring in the remaining tale is $2i$.

The final two summands (for $i = c$) refer to the leading position taking the value $2c$ or $_$ (“ $b_l = 2c$ ” or “ $b_l = _$ ”). In both cases, the maximal value of the colors occurring in the remaining tale is $2c$.

While the representation is different, it is easy to see that $\text{count}_{1,2}^\ell(2c, l) = \text{count}_{JL}^\ell(2c, l)$ holds.

To see this, we first observe that $\text{count}_{JL}^\ell(2, l + 1) = 1 + 2\text{count}_{JL}^\ell(2, l)$ holds, and then by induction over c that:

$$\text{count}_{JL}^\ell(2, l + 1) = 1 + 2 \sum_{i=1}^c \text{count}_{JL}^\ell(2i, l).$$

Given that we also have $\text{count}_{1,2}^\ell(2c, 1) = \text{count}_{JL}^\ell(2c, 1)$, we get the claim, because $\text{count}_{JL}^\ell(2c, l)$ cannot be derived in the same way as $\text{count}_{1,2}^\ell(2c, l)$.

6.3.2. (3) and (4) Removing odd colors from the rightmost position and 1s

Removing odd colors from the rightmost positions only changes the base case, while banning 1 from the other positions merely removes the “+” part from the inductive definition. This leaves:

- Induction basis, length:

$$\text{count}^\ell(2c, 1) = c + 1.$$

- For longer tails of sequences, we define inductively:

$$\text{count}^\ell(2c, l + 1) = 2 \sum_{i=1}^c \text{count}^\ell(2i, l).$$

When evaluating the term count^ℓ , the reduction from $2c + 1$ to $c + 1$ is halving the value (rounded up) at the leaf of each call tree, which provides more than the removal of “= 1” in each node of the call tree. Together, they **broadly halve the value**.

6.3.3. (5) Taking the value into account

We start with using both the length and the value and then remove the length in the next step to get a more concise representation, but we note that, for a given length l , the value v allowed always satisfies $v < 2^l$.

First, we get another induction base, one by value:

- Induction basis, value:

$$\text{count}^{\ell,v}(2c, l, 0) = 1.$$

Regardless of the remaining length, if the *value* of the tail is bounded by (and thus needs to be) 0, then it can only consist of $_$ signs.

- Induction basis, length:

$$\text{count}^{\ell,v}(2c, 1, 1) = c + 1.$$

- For longer tails of sequences and positive values, we distinguish several cases. The first case is that $v < 2^l$. Then we have

$$\text{count}^{\ell,v}(2c, l + 1, v) = \text{count}^{\ell,v}(2c, l, v).$$

This is simply because filling the position $l + 1$ with any number, even or odd, would exceed the value budget.

This leaves the case $v \geq 2^l$, i.e.:

$$\begin{aligned} \text{count}^{\ell,v}(2c, l + 1, v) &= \sum_{i=1}^c \text{count}^{\ell,v}(2i, l, v - 2^l) \\ &+ \sum_{i=1}^c \text{count}^{\ell,v}(2i, l, 2^l - 1). \end{aligned}$$

This is because, when filling position l with an even number, it takes 2^l from the budget of the value, leaving a remaining budget of $v - 2^l$.

When filling this position with an odd number, while the value would be increased by 2^l , this is within the value budget. Moreover, if this position is still relevant to the value, then the positions to its right no longer add to the value of the sequence, as the leftmost odd position would be the last to be considered.

We, therefore, set the value for the remaining tail to the right to be the maximal value that can be obtained by this tail, which is $2^l - 1$; this is a rendering of saying that for the tail the values are not constrained.

The effect of adding value can vary greatly. It is larger when the number of positions with even color is a power of 2, say 2^l , and it has no effect at all if it is $2^2 - 1$. In the former case, if the initial position is even, then all other positions need to be $_$. Generally, we have

$$\begin{aligned} \text{count}^{\ell,v}(2c, l, 2^l - 1) &= \text{count}^\ell(2c, l) \quad \text{and} \\ \text{count}^{\ell,v}(2c, l, 2^{l-1}) &= \text{count}^\ell(2c, l)/2 + c \end{aligned}$$

for all $l > 1$.

Taking the value into account therefore broadly **halves the statespace when e is a power of 2**, and **has no effect when e is**

a predecessor of a power of 2, falls from 2^{l-1} to $2^l - 1$ for all $l > 1$.

Looking at the definition of $\text{count}^{\ell,v}$, it is easy to see that an explicit reference to the length can be replaced by a reference to the next relevant length, $\lfloor \log_2 v \rfloor$. This provides:

$$\begin{aligned} \text{count}^v(2c, 0) &= 1, \\ \text{count}^v(2c, 1) &= c + 1, \text{ and} \\ \text{count}^v(2c, v) &= \sum_{i=1}^c \text{count}^v(2i, v - 2^{\lfloor \log_2 v \rfloor}) \\ &\quad + \sum_{i=1}^c \text{count}^v(2i, 2^{\lfloor \log_2 v \rfloor} - 1) \text{ otherwise.} \end{aligned}$$

6.4. Comparison with the statespace of

While improvement (1) is the most powerful of the optimizations, the improvements (2)–(4) were present in Fearnley et al. (2019), where the algorithm makes use of a value function, namely $\text{value}'(\mathbf{b}) = \sum_{i \in \text{even}(\mathbf{b})} 2^i$. It is, therefore, interesting to provide a count function for Fearnley et al. (2019). We use the subscript JKSSW, and only use the count that uses both length and value.

We get the following state counts:

$$\begin{aligned} \text{count}_{JKSSW}^{\ell,v}(c, 1, 0) &= 1 \\ \text{count}_{JKSSW}^{\ell,v}(c, 1, 1) &= \lfloor c/2 \rfloor + 1 \\ \text{if } v < 2^l: \text{count}_{JKSSW}^{\ell,v}(c, l + 1, v) &= \text{count}_{JKSSW}^{\ell,v}(c, l, v) \\ &\quad + \sum_{i=2}^{\lfloor c/2 \rfloor} \text{count}_{JKSSW}^{\ell,v}(2i - 1, l, v) \\ \text{if } v \geq 2^l: \text{count}_{JKSSW}^{\ell,v}(c, l + 1, v) &= \text{count}_{JKSSW}^{\ell,v}(c, l, 2^l - 1) \\ &\quad + \sum_{i=1}^{\lfloor c/2 \rfloor} \text{count}_{JKSSW}^{\ell,v}(2i, l, v - 2^l) \\ &\quad + \sum_{i=2}^{\lfloor c/2 \rfloor} \text{count}_{JKSSW}^{\ell,v}(2i - 1, l, 2^l - 1). \end{aligned}$$

To explain the difference to $\text{count}^{\ell,v}$, one major difference is that the highest color allowed in a position can be odd. The other is that positions with odd color do not contribute to the weight, which allows for adding positions with odd color the remaining budget is lower than 2^l .

Thus, the call tree for the calculation of $\text{count}_{JKSSW}^{\ell,v}$ has $\lfloor c/2 \rfloor$ successors where $v < 2^l$ while the call tree for $\text{count}^{\ell,v}$ has just one. For $v \geq 2^l$, the call tree has the same number of successors (for even c) or just one additional successor (for odd c), but the parameter falls slower.

7. Statespace comparison

In this subsection, we provide a graphical representation of the statespace size for the three algorithms: Calude et al. (2017), Jurdziński and Lazic (2017), and the improvement described in this article. The size of the statespace on which

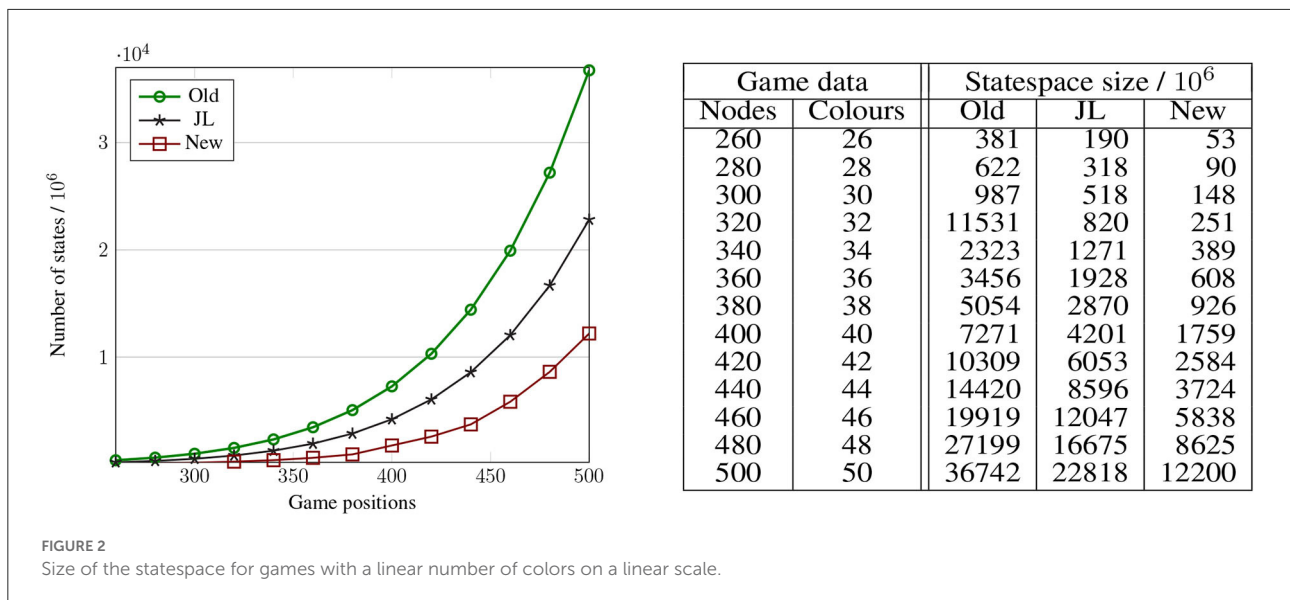
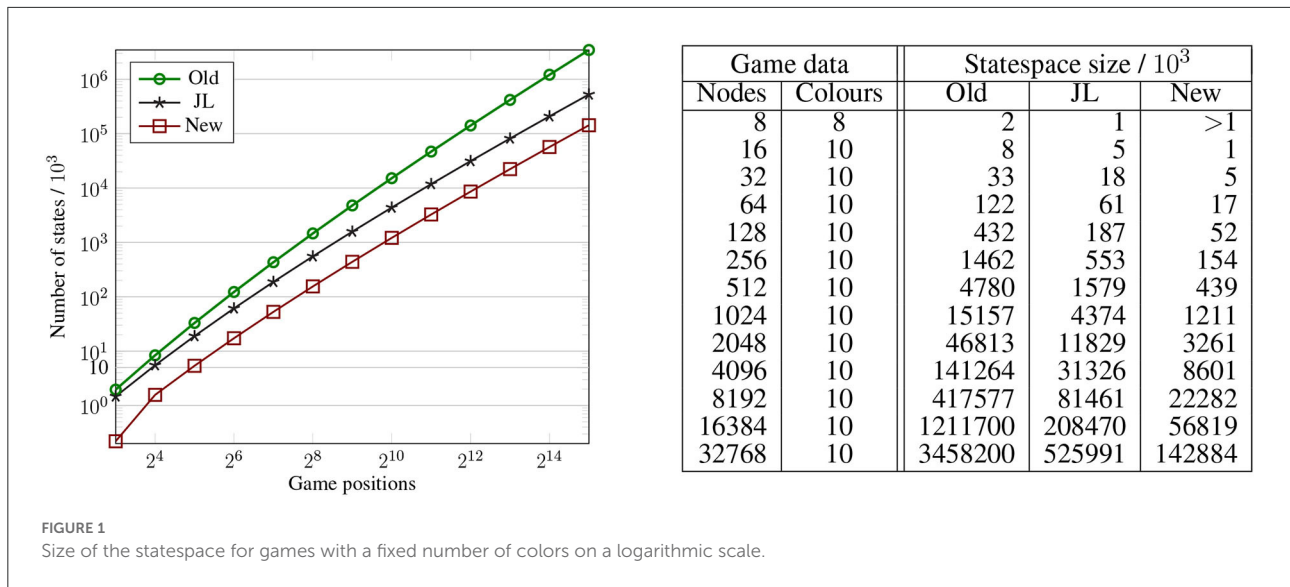
an algorithm works does not represent how well the algorithm performs in practice. Indeed, in the context of parity games, there are quasi-polynomial time algorithms that behave like brute-force approaches. Therefore, they always require quasi-polynomial many steps to compute the solution, while most of the exponential time algorithms, instead, almost visit a polynomial fraction of their statespace. The first improvement we described does not affect the performance of the algorithm, since both the original and the improved algorithm require the same number of steps to solve a game, but the latter works on a reduced statespace. To measure how big is the cut we consider Figure 1 games with a fixed number of colors and Figure 2 games with a linear number of colors in the size of the game. The games in Figure 1 range from 2^3 to 2^{15} positions n . Therefore, the length of the measure, which is logarithmic in n , constantly increases, while the colors are fixed to a value of 10. As a consequence, the ratio of colors with respect to n range from 80 to 0.02%. As expected, the cut with the original algorithm significantly increases for games that are not dense in colors as the lines tend to diverge on a logarithmic scale. The ratio between Jurdzinski and Lazic approach (JL) and the new improvement, instead, converges to a cut of 73% of the statespace. The games of Figure 2, instead range from 2^8 to 2^9 positions n , so that the length of the measure is fixed, while the number of colors constantly grows from 26 to 50. As a consequence, we have that the ratio of colors with respect to n is fixed to 10%. In this case, the scale is linear and, even if the improved statespace is always smaller than the other two, the cut tends to shrink.

8. Discussion

We have introduced three technical improvements over the progress measures used in the original quasipolynomial approach by Calude et al. (2017) and its improvements by Fearnley et al. (2019). The first two reduce the statespace.

The more powerful of the two is a simple limitation of the occurrences of odd colors in a witness to one. Where the highest color is even, this alone reduces the size of the statespace of Calude et al.'s approach to the currently smallest one of Jurdziński and Lazic (2017). Where the highest color is odd, we obtain the same by borrowing the simple observation that this highest color does not need to be used from Fearnley et al. (2019).

The second new means to reduce the statespace is the use of witnesses that only refer to even chains of plausible size, namely those that do not contain more dominating even states than the game has to offer. A similar idea had been explored in Fearnley et al. (2019), but our construction is more powerful in reducing the size of the statespace. The effect of this step ranges from none (where the number of states with even color is the predecessor of a power of 2 ($2^\ell - 1$ for some $\ell \in \mathbb{N}$), then rises steeply to a



factor of 2 for a power of 2 (2^ℓ), and then slowly falls again, until it vanishes at the next predecessor of a power of 2.

These improvements work well with the other improvements from Fearnley et al. (2019), namely not using the color 1 and disallowing odd values for the rightmost position (“ b_0 ”) in a witness. These improvements broadly halve the statespace, leading to a statespace reduction that broadly oscillates between 2 and 4 when compared to the previously leading approach.

The second improvement we have introduced is a re-definition of the semantics of witnesses, moving from classic witnesses to color witnesses. While it does not lead to a difference in the size of the statespace, it does accelerate its traversal,

especially for the “standard” update rule that does not extend to value iteration; in particular, it gets rid of the most trivial kind of silly hard examples, such as cliques of states of player odd that all have even color.

While the use of color witnesses clearly accelerates the analysis, it is not as easy as for the statespace reduction to formally quantify this advantage.

Data availability statement


The original contributions presented in the study are included in the article/supplementary

material, further inquiries can be directed to the corresponding author.

Author contributions

Both authors listed have made a substantial, direct, and intellectual contribution to the work and approved it for publication.

Funding

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 101032464 .

References

- Alur, R., Henzinger, T., and Kupferman, O. (2002). Alternating-time temporal logic. *J. ACM* 49, 672–713. doi: 10.1145/585265.585270
- Benerecetti, M., Dell'Erba, D., and Mogavero, F. (2016). "Improving priority promotion for parity games," in *HVC'16, LNCS 10028* (Haifa: Springer), 1–17.
- Benerecetti, M., Dell'Erba, D., and Mogavero, F. (2018a). A delayed promotion policy for parity games. *Inf. Comput.* 262, 221–240. doi: 10.1016/j.ic.2018.09.005
- Benerecetti, M., Dell'Erba, D., and Mogavero, F. (2018b). Solving parity games via priority promotion. *Form. Methods Syst. Des.* 52, 193–226. doi: 10.1007/s10703-018-0315-1
- Benerecetti, M., Mogavero, F., and Murano, A. (2013). "Substructure temporal logic," IN *Logic in Computer Science'13* (New Orleans, LA: IEEECS), 368–377.
- Berwanger, D., and Grädel, E. (2004). Fixed-point logics and solitaire games. *Theor. Comput. Sci.* 37, 675–694. doi: 10.1007/s00224-004-1147-5
- Bojańczyk, M., and Czerwiński, W. (2018). Available online at: <https://www.mimuw.edu.pl/bojan/papers/toolbox-reduced-feb6.pdf>An Automata Toolbox.
- Calude, C., Jain, S., Khossainov, B., Li, W., and Stephan, F. (2017). "Deciding parity games in quasipolynomial time," in *Symposium on Theory of Computing'17* (Montreal, QC: Association for Computing Machinery), 252–263.
- Chatterjee, K., Henzinger, T., and Piterman, N. (2010). Strategy logic. *Inf. Comput.* 208, 677–693. doi: 10.1016/j.ic.2009.07.004
- Czerwiński, W., Daviaud, L., Fijalkow, N., Jurdzinski, M., Lazić, R., and Parys, P. (2018). "Universal trees grow inside separating automata: quasi-polynomial lower bounds for parity games," in *SODA'18* (SIAM), 2333–2349.
- Emerson, E., and Jutla, C. (1991). "Tree automata, mu-calculus, and determinacy," in *FOCS'91* (San Juan: IEEECS), 368–377.
- Emerson, E., Jutla, C., and Sistla, A. (2001). On model checking for the mu-calculus and its fragments. *Theor. Comput. Sci.* 258, 491–522. doi: 10.1016/S0304-3975(00)00034-7
- Emerson, E., and Lei, C.-L. (1986). "Temporal reasoning under generalized fairness constraints," in *Symposium on Theoretical Aspects of Computer Science'86, LNCS 210* (Orsay: Springer), 267–278.
- Fearnley, J. (2010). "Non-oblivious strategy improvement," in *LPAR'10, LNCS 6355* (Dakar: Springer), 212–230.
- Fearnley, J., Jain, S., Keijzer, B., Schewe, S., Stephan, F., and Wojtczak, D. (2019). An ordered approach to solving parity games in quasi polynomial time and quasi linear space. *Software Tools Technol. Transfer* 21, 325–349. doi: 10.1007/s10009-019-00509-3
- Fearnley, J., Jain, S., Schewe, S., Stephan, F., and Wojtczak, D. (2017). "An ordered approach to solving parity games in quasi polynomial time and quasi linear space," in *SPIN'17* (Santa Barbara, CA: Association for Computing Machinery), 112–121.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Friedmann, O. (2013). A superpolynomial lower bound for strategy iteration based on snare memorization. *Discrete Appl. Math.* 161, 1317–1337. doi: 10.1016/j.dam.2013.02.007

Grädel, E., Thomas, W., and Wilke, T. (2002). "Automata, logics, and infinite games: a guide to current research," in *LNCS 2500* (Dagstuhl: Springer).

Jurdzinski, M. (1998). Deciding the winner in parity games is in $UP \cap co-UP$. *Inf. Process. Lett.* 68, 119–124. doi: 10.1016/S0020-0190(98)00150-1

Jurdzinski, M. (2000). "Small progress measures for solving parity games," in *Symposium on Theoretical Aspects of Computer Science'00, LNCS 1770* (Lille: Springer), 290–301.

Jurdzinski, M., and Lazić, R. (2017). "Succinct progress measures for solving parity games," in *Logic in Computer Science'17* (Reykjavik: Association for Computing Machinery), 1–9.

Kupferman, O., and Vardi, M. (1998). "Weak alternating automata and tree automata emptiness," in *Symposium on Theory of Computing'98* (Dallas, TX: Association for Computing Machinery), 224–233.

Lapauw, R., Bruynooghe, M., and Denecker, M. (2020). "Improving parity game solvers with justifications," in *VMCAI'20, LNCS 11990* (New Orleans, LA: Springer), 449–470.

Lehtinen, K. (2018). "A modal mu perspective on solving parity games in quasi-polynomial time," in *Logic in Computer Science'18* (Oxford: Association for Computing Machinery & IEEECS), 639–648.

Martin, A. (1975). Borel determinacy. *Ann. Math.* 102, 363–371. doi: 10.2307/1971035

Mogavero, F., Murano, A., Perelli, G., and Vardi, M. (2012). "What makes ATL^* decidable? a decidable fragment of strategy logic," in *Concurrency Theory'12, LNCS 7454* (Newcastle upon Tyne: Springer), 193–208.

Mogavero, F., Murano, A., and Vardi, M. (2010). "Reasoning about strategies," in *FSTTCS'10, LIPIcs 8* (Chennai: Leibniz-Zentrum fuer Informatik), 133–144.

Mostowski, A. (1991). *Games with Forbidden Positions*. Technical report, University of Gdańsk, Gdańsk, Poland.

Parys, P. (2019). "Parity games: Zielonka's algorithm in quasi-polynomial time," in *Proceedings of MFCS, LIPIcs 138* (Leibniz-Zentrum fuer Informatik), 1–10.

Schewe, S. (2007). "Solving parity games in big steps," in *FSTTCS'07, LNCS 4855* (New Delhi: Springer), 449–460.

Schewe, S. (2008). " ATL^* satisfiability is $2ExpTime$ -complete," in *International Colloquium on Automata, Languages, and Programming'08, LNCS 5126* (Reykjavik: Springer), 373–385.

Schewe, S., and Finkbeiner, B. (2006). "Satisfiability and finite model property for the alternating-time mu-calculus," in *CSL'06, LNCS 6247* (Szeged: Springer), 591–605.

van Dijk, T. (2018). "Attracting tangles to solve parity games," in *CAV'18, LNCS 10982* (Oxford: Springer), 198–215.

Vöge, J., and Jurdziński, M. (2000). "A discrete strategy improvement algorithm for solving parity games," in *CAV'00, LNCS 1855* (Chicago, IL: Springer), 202–215.

Wilke, T. (2001). Alternating tree automata, parity games, and modal μ Calculus. *Bull. Belg. Math. Soc.* 8, 359–391. doi: 10.36045/bbms/1102714178

Zielonka, W. (1998). Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.* 200, 135–183. doi: 10.1016/S0304-3975(98)00009-7