Universal trees grow inside separating automata: Quasi-polynomial lower bounds for parity games

Wojciech Czerwiński[1], Laure Daviaud[2], Nathanaël Fijalkow[3,4,2], Marcin Jurdziński[2], Ranko Lazić[2], and Paweł Parys[1]

[1]MIMUW, University of Warsaw, Poland
[2]DIMAP, Department of Computer Science, University of Warwick, UK
[3]CNRS, LaBRI, France
[4]The Alan Turing Institute, UK

Abstract

Several distinct techniques have been proposed to design quasi-polynomial algorithms for solving parity games since the breakthrough result of Calude, Jain, Khoussainov, Li, and Stephan (2017): *play summaries*, *progress measures* and *register games*. We argue that all those techniques can be viewed as instances of the *separation approach* to solving parity games, a key technical component of which is constructing (explicitly or implicitly) an automaton that *separates* languages of words encoding plays that are (decisively) won by either of the two players. Our main technical result is a quasi-polynomial lower bound on the size of such separating automata that nearly matches the current best upper bounds. This forms a *barrier* that all existing approaches must overcome in the ongoing quest for a polynomial-time algorithm for solving parity games. The key and fundamental concept that we introduce and study is a *universal ordered tree*. The technical highlights are a quasi-polynomial lower bound on the size of universal ordered trees and a proof that every *separating safety automaton* has a universal tree hidden in its state space.

1 Introduction

1.1 Parity games

The algorithmic problem. A *parity game* is played on a directed graph by two players who are called Even and Odd. A play starts at a designated vertex and then the players move by following outgoing edges forever, thus forming an infinite path. Every vertex of the graph is owned by one of the two players and it is always the owner of the vertex who moves by following an outgoing edge from the current vertex to the next one.

This completes the description of the dynamics of a play, but how do we declare the winner of an infinite path formed in this way? For this, we need to inspect positive integers that label all edges in the graph, which we refer to as *edge priorities*, or simply priorities. Player Even is declared the winner of a play if the highest priority that occurs infinitely many times is even, and otherwise player Odd wins; equivalently, the winner is the parity of the limsup (limes superior) of the priorities that occur in the play.

The principal algorithmic problem studied in the context of parity games is *deciding the winner*: given a game graph as described above and a starting vertex, does player Even have a winning strategy—a recipe for winning every play starting from the designated vertex, no matter what edges her opponent Odd follows whenever it is his turn to move.

Determinacy and complexity. A *positional strategy* for Even is a set of edges that go out of vertices she owns—exactly one such edge for each of her vertices; Even uses such a strategy by always—if the current vertex is owned by her—following the unique outgoing edge that is in the strategy. Note that when Even uses a positional strategy, her moves depend only on the current vertex—they are oblivious to what choices were made by the players so far. A basic result for parity games that has notable implications is their *positional determinacy* [11, 28]: for every starting vertex, exactly one of the players has a winning strategy and hence the set of vertices is partitioned into the *winning set* for Even and the winning set for Odd; moreover, each player has a *positional strategy* that is winning for her from all starting vertices in her winning set.

An important corollary of positional determinacy is that deciding the winner in parity games is *well characterized*, i.e., it is both in NP and in co-NP [12]. Several further complexity results suggest that it may be difficult to provide compelling evidence for hardness of solving parity games: deciding the winner is known to be also in UP and in co-UP [22], and computing winning strategies is in PLS, PPAD, and even in their subclass CLS [8, 9]. Parity games share this intriguing complexity-theoretic status with several other related problems, such as mean-payoff games [33], discounted games, and simple stochastic games [7], but they are no harder than them since there are polynomial reductions from parity games to mean-payoff games, to discounted games, and to simple stochastic games [22, 33].

Significance and impact. Parity games play a fundamental role in automata theory, logic, and their applications to verification and synthesis. Specifically, the algorithmic problem of deciding the winner in parity games is polynomial-time equivalent to the model checking in the modal μ-calculus and to checking emptiness of automata on infinite trees with parity acceptance conditions [12], and it is at the heart of algorithmic solutions to the Church's synthesis problem [29].

The impact of parity games goes well beyond their place of origin in automata theory and logic. We illustrate it by the resolutions of two long-standing open problems in *stochastic planning* and in *linear programming*, respectively, that were directly enabled by the ingenious examples of parity games given by Friedmann [17], on which the *strategy improvement algorithm* [32] requires exponentially many iterations. Firstly, Fearnley [14] has shown that Friedmann's examples can be adapted to prove that *Howard's policy iteration* algorithm for Markov decision processes (MDPs) requires exponentially many iterations. Policy iteration has been well-known and widely used in stochastic planning and AI since 1960's, and it has been celebrated for its fast termination: until Fearnley's surprise result, no examples were known for which a super-linear number of iterations was necessary. Secondly, Friedmann, Hansen, and Zwick [18] have adapted the insights from the lower bounds for parity games and MDPs to prove that natural *randomized pivoting rules* in the *simplex algorithm* for linear programming may require subexponentially many iterations. The following quote from the full version of Friedmann et al. [18] highlights the role that parity games (PGs) played in their breakthrough:

> "our construction can be described and understood without knowing about PGs. We would like to stress, however, that most of our intuition about the problem was obtained by thinking in terms of PGs. Thinking in terms of MDPs seems harder, and we doubt whether we could have obtained our results by thinking directly in terms of linear programs."

In both cases, Friedmann's examples of parity games and their analysis have been pivotal in resolving the theoretical worst-case complexity of influential algorithms that for many decades resisted rigorous analysis while performing outstandingly well in practice.

Current state-of-the-art. It is a long-standing open question whether there is a polynomial-time algorithm for solving parity games [12]. The study of algorithms for solving parity games has been dominated for over two decades by algorithms whose run-time was exponential in the number of distinct priorities [13, 4, 31, 23, 32, 30], or mildly subexponential for large number of priorities [2, 26]. The breakthrough came in 2017 from Calude et al. [5] who gave the first quasi-polynomial-time algorithm using the novel idea of *play summaries*. Several other quasi-polynomial-time algorithms were developed soon after, including space-efficient *progress-measure* based algorithms of Jurdziński and Lazić [24] and of Fearnley, Jain, Schewe, Stephan, and Wojtczak [15], and the algorithm of Lehtinen [27], based on her concept of *register games*.

1.2 The separation approach

Bojańczyk and Czerwiński [3, Section 3] have observed that the main technical contribution of Calude et al. [5] can be elegantly phrased using concepts from automata theory. They have pointed out that in order to reduce solving a parity game to solving a much simpler *safety game*, it suffices to provide a finite *safety automaton* that achieves the task of *separating* two sets AllCyclEven and AllCyclOdd of (infinite) words that are *decisively won* by the two players, respectively. For encoding plays in parity games, they use words in which every letter is a pair that consists of a vertex and a priority. The definition of such a word being decisively won by a player that was proposed by Bojańczyk and Czerwiński is that the biggest priority that occurs on every cycle—an infix in which the first vertex and the vertex immediately following the infix coincide—is of her parity. Concerning separation, for two disjoint languages K and L, we say that a language S *separates* K from L if $K \subseteq S$ and $S \cap L = \emptyset$, and we say that an automaton $\mathcal{A}$ is a *separator* of two languages if the language $L(\mathcal{A})$ of words recognized by $\mathcal{A}$ separates them. The main technical contribution of Calude et al. [5] can then be stated as constructing separators—of quasi-polynomial size—of the languages AllCyclEven and AllCyclOdd.

Note that a separator of AllCyclEven and AllCyclOdd has a significantly easier task than a *recognizer* of exactly the set LimsupEven of words that are won by Even—that is required to accept all words in LimsupEven, and to reject all words in LimsupOdd, the set of all words that are won by Odd. Instead, a separator may reject some words won by Even and accept some words won by Odd, as long as it accepts all words that are decisively won by Even, and it rejects all words that are decisively won by Odd.

What Calude et al. [5] exploit is that if one of the players uses a *positional* winning strategy then all plays are indeed encoded by words that are won decisively by her, no matter how the opponent responds. The formalization of Bojańczyk and Czerwiński [3] is that—using positional determinacy of parity games [11, 28]—in order to solve a parity game, it suffices to solve a strategically and algorithmically much simpler *safety game* that is obtained as a simple synchronized product of the parity game and a safety automaton that is a separator of AllCyclEven and AllCyclOdd.

1.3 Our contribution

Our main conceptual contributions include making explicit the notion of a *universal ordered tree* and unifying all the existing quasi-polynomial algorithms for parity games [5, 24, 19, 15, 27] as instances of the *separation approach* proposed by Bojańczyk and Czerwiński [3].

We point out that a universal tree is the fundamental combinatorial object that can serve as the data structure on which *progress measure lifting* algorithms [23, 1, 30, 24, 10] operate, and that the running time of such algorithms is dictated by the size of the universal tree. As our main technical results show, however, universal trees are fundamental not only for progress measure lifting algorithms, but for all algorithms that follow the separation approach.

We argue that in the separation approach, it is appropriate to slightly adjust the choice of languages to be separated, from AllCyclEven and AllCyclOdd proposed by Bojańczyk and

Czerwiński [3] to the more suitable PosCyclEven and PosCyclOdd (see Section 2.1 for the definitions and the rationale). We also verify, in Section 5, that all the three distinct techniques of solving parity games in quasi-polynomial time considered in the recent literature (*play summaries* [5, 19, 15], *progress measures* [24], and *register games* [27]) yield separators for languages PosCyclEven and LimsupOdd, which (as we argue in Section 2.2) makes them suitable for the separation approach.

The main technical contribution of the paper, described in Sections 3.2 and 4 is a proof that every (non-deterministic) safety automaton that separates PosCyclEven from LimsupOdd has a number of states that is at least quasi-polynomial. First, in Section 3.2 we establish a quasi-polynomial lower bound on the size of universal trees. Then, in Section 4, our argument is based on proving that in every such an automaton, one can define a sequence of *linear quasi-orders* on the set of states, in which each quasi-order is a refinement of the quasi-order that follows it in the sequence. Such a sequence of linear quasi-orders can be naturally interpreted as an *ordered tree* in which every leaf is populated by at least one state of the automaton. We then also prove that the ordered tree must contain a *universal ordered tree* [24], and the main result follows from the earlier quasi-polynomial lower bound for universal trees.

Another technical highlight, presented in Section 5.3, is a construction of a separator from an arbitrary universal tree, which together with the main technical result implies that the sizes of smallest universal trees and of smallest separators coincide. The correctness of the construction relies on existence of the least *progress measures* that map from vertices of game graphs into universal trees and witness winning strategies.

The significance of our main technical results is that they provide evidence against the hope that any of the existing technical approaches to developing quasi-polynomial algorithms for solving parity games [5, 24, 15, 27] may lead to further improvements to sub-quasi-polynomial algorithms. In other words, our quasi-polynomial lower bounds for universal trees and separators form a *barrier* that all existing approaches must overcome in the ongoing quest for a polynomial-time algorithm for solving parity games.

We leave open the question whether a stronger version of our main technical result holds, namely whether every safety automaton separating PosCyclEven from PosCyclOdd has at least quasi-polynomial number of states. Our argument cannot be directly extended to that setting, as already the proof of Lemma 2 heavily relies on that fact that no word from LimsupOdd is accepted.

2 Preliminaries

2.1 Game graphs and play languages

Game graphs and strategy subgraphs. Throughout the paper, we write V for the set of vertices and E for the set of edges in a parity game graph, and we use n to denote the numbers of vertices. Typically, we assume that $V = \{1, 2, \ldots, n\}$. For every edge $e \in E$, its *priority* $\pi(e)$ is a positive integer, and we use d to denote the smallest even number larger than or equal to priorities of all edges. Without loss of generality, we assume that every vertex has at least one outgoing edge. We say that a cycle in a game graph is *even* if the largest edge priority that occurs on it is even; otherwise it is *odd*.

Recall that a positional strategy for Even is a set of edges that go out of vertices she owns—exactly one such edge for each of her vertices. The *strategy subgraph* of a positional strategy for Even is the subgraph of the game graph that includes all outgoing edges from vertices owned by Odd and exactly those outgoing edges from vertices owned by Even that are in the positional strategy. Observe that the set of plays that arise from Even playing her positional strategy is exactly the set of all plays in the strategy subgraph. Moreover, note that every cycle in the strategy subgraph of a positional strategy for Even that is winning for her is even: otherwise,

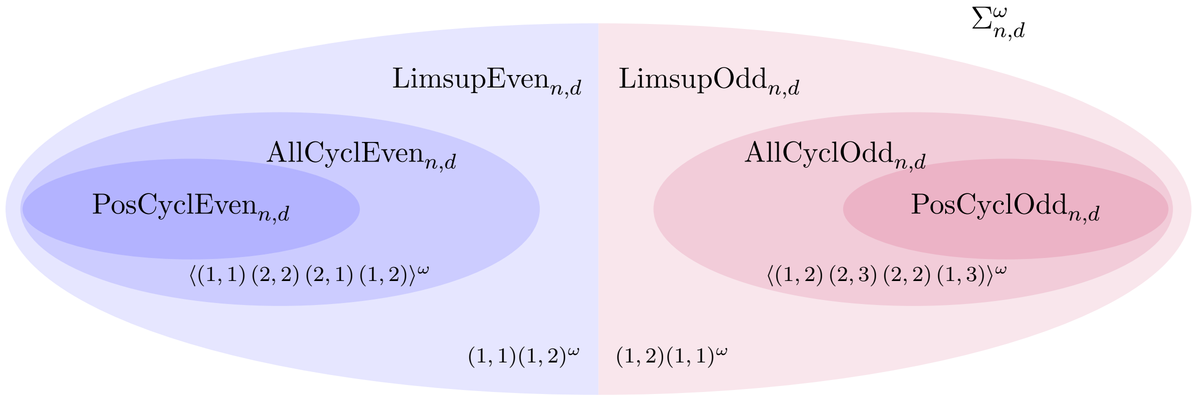


Figure 1: Sets of infinite words in $\Sigma^\omega_{n,d}$.

by repeating an odd cycle indefinitely, we would get a play that is winning for Odd.

Languages of play encodings. The outcome of the two players interacting in a parity game by making moves is an infinite path in the game graph. We encode such infinite paths as infinite words over the alphabet $\Sigma_{n,d} = \{1, 2, \ldots, n\} \times \{1, 2, \ldots, d\}$ in a natural way: each move—in which from vertex v an outgoing edge e is followed—is encoded by the letter $(v, \pi(e))$ that consists of the *vertex component* v and the *priority component* $\pi(e)$.

We write $\text{LimsupEven}_{n,d}$ for the set of infinite words in which the biggest number that occurs infinitely many times in the priority components of the letters is even, and we write $\text{LimsupOdd}_{n,d}$ for the set of infinite words in which that number is odd. Observe that sets $\text{LimsupEven}_{n,d}$ and $\text{LimsupOdd}_{n,d}$ form a partition of the set $\Sigma^\omega_{n,d}$ of all infinite words over the alphabet $\Sigma_{n,d}$. As intended, an infinite play in a parity game graph with n vertices and edge priorities not exceeding d is winning for Even if and only if the infinite-word encoding of the play is in $\text{LimsupEven}_{n,d}$.

In order to be usable in the separation approach, automata do not need to recognize the set $\text{LimsupEven}_{n,d}$ of winning plays—a prohibitive requirement. Instead, Bojańczyk and Czerwiński argue that it is sufficient that the automaton separates sets $\text{AllCyclEven}_{n,d}$ and $\text{AllCyclOdd}_{n,d}$, that are strict subsets of $\text{LimsupEven}_{n,d}$ and $\text{LimsupOdd}_{n,d}$, respectively. We say that a nonempty infix of a word over the alphabet $\Sigma_{n,d}$ is a *cycle* if the vertex component of its first letter coincides with the vertex component of the first letter appearing directly after this infix. We say that a cycle is *even* if the highest number that occurs as the priority component of any letter in the cycle is even; otherwise the cycle is *odd*. We then define $\text{AllCyclEven}_{n,d}$ to be the set of infinite words in which every cycle is even, and $\text{AllCyclOdd}_{n,d}$ to be the set of infinite words in which every cycle is odd.

We observe that the sets $\text{AllCyclEven}_{n,d}$ and $\text{AllCyclOdd}_{n,d}$ contain degenerate words that do not encode plays arising from positional winning strategies, and hence their acceptance or rejection has no significance in the correctness argument for the separation approach. Instead, we propose to work with sets of words $\text{PosCyclEven}_{n,d}$ and $\text{PosCyclOdd}_{n,d}$—we define them to consist of all the encodings of plays that arise from positional winning strategies for Even and Odd, respectively, in some game graph with n vertices and priorities up to d.

It is easy to argue that inclusions between the six sets of $\Sigma^\omega_{n,d}$, are as in Figure 1. The inclusions are strict, as witnessed by the examples of infinite words in the color-shaded areas.

2.2 Safety automata and the separation approach

Safety automata and games. The fundamental and simple model that the statement of our main technical result formally refers to is a (non-deterministic) *safety automaton*. Superficially, it closely resembles the classic model of *finite automata*: each safety automaton has a finite set of *states*, a designated *initial state*, and a *transition relation*. (Without loss of generality, we assume that the transition relation is *total*, i.e., for every state s and letter a, there is at least one state s', such that the triple (s, a, s') is in the transition relation.) The differences between our model of safety automata and the classic model of finite automata with designated *accepting states* are as follows:

- safety automata are meant to accept or reject *infinite words*, not finite ones;
- a safety automaton does not have a designated set of accepting states; instead it has a designated set of *rejecting states*;
- a safety automaton *accepts* an infinite word if there is an infinite run of the automaton on the word in which no rejecting state ever occurs; otherwise it *rejects* the infinite word.

We say that a safety automaton is *deterministic* if the transition relation is a function: for every state s and letter a, there is a unique state s', such that the triple (s, a, s') is a transition.

Finally, we define the elementary concept of *safety games*, which are played by two players on finite directed graphs in a similar way to parity games, but the goals of the players are simpler than in parity games: the *safety player* wins if a designated set of *unsafe vertices* is never visited, and otherwise the opponent (sometimes called the *reachability player*) wins.

The separation approach. We now explain how safety separating automata allow to reduce the complex task of solving a parity game to the (conceptually and algorithmically) straightforward task of solving a safety game, by exploiting positional determinacy of parity games. This is the essence of the *separation approach* that implicitly underpins the algorithms of Bernet, Janin, and Walukiewicz [1] and of Calude et al. [5], as formalized by Bojańczyk and Czerwiński [3, Chapter 3]. Here, we only consider the simple case of *deterministic* automata. We postpone the discussion of using non-deterministic automata in the separation approach to Section 5.6, which is the only place where non-determinism seems to be needed.

Given a parity game G with at most n vertices and priorities up to d, and a deterministic safety automaton $\mathcal{A}$ with input alphabet $\Sigma_{n,d}$, we define a safety game as the *synchronised product* $G \times \mathcal{A}$, in which

- the dynamics of play and ownership of vertices is inherited from the parity game G;
- the automaton $\mathcal{A}$ is fed the vertex-priority pairs corresponding to moves made by the players;
- the safety winning condition is the safety acceptance condition of the automaton $\mathcal{A}$.

Proposition 1. *If G is a parity game with n vertices and priorities up to d, and $\mathcal{A}$ is a deterministic safety automaton that separates* $\text{PosCyclEven}_{n,d}$ *from* $\text{PosCyclOdd}_{n,d}$*, then Even has a winning strategy in G if and only if she has a winning strategy in the synchronized-product safety game $G \times \mathcal{A}$.*

Proof. If Even has a winning strategy in the parity game G then—by positional determinacy [11, 28]—she also has one that is positional. We argue that if Even uses such a positional winning strategy in G when playing the synchronized-product game $G \times \mathcal{A}$, then she also wins the latter. This follows from every infinite play in the strategy subgraph of a positional winning strategy for Even having only even cycles, which implies that the encoding of such a play is an infinite word in $\text{PosCyclEven}_{n,d}$, and hence it is accepted by the automaton $\mathcal{A}$.

Otherwise, Odd has a positional winning strategy in G, and it can be transferred to a winning strategy for him in $G \times \mathcal{A}$ in the same way as we argued for Even above. □

In the rest of the paper, we focus on separators of $\mathrm{PosCyclEven}_{n,d}$ and $\mathrm{LimsupOdd}_{n,d}$ (rather than on separators of $\mathrm{PosCyclEven}_{n,d}$ and $\mathrm{PosCyclOdd}_{n,d}$), for two reasons. Firstly—as described in Section 5—all the known quasi-polynomial algorithms for parity games are underpinned by separators of $\mathrm{PosCyclEven}_{n,d}$ and $\mathrm{LimsupOdd}_{n,d}$, which also separate $\mathrm{PosCyclEven}_{n,d}$ from $\mathrm{PosCyclOdd}_{n,d}$ and hence meet the assumption of Proposition 1. Secondly, although some separators that we describe in Section 5 even separate $\mathrm{AllCyclEven}_{n,d}$ from $\mathrm{LimsupOdd}_{n,d}$, the quasi-polynomial lower bound that we give in Section 4 relies on the property that the safety automaton separates $\mathrm{PosCyclEven}_{n,d}$ and $\mathrm{LimsupOdd}_{n,d}$, which again holds for all automata that separate $\mathrm{AllCyclEven}_{n,d}$ and $\mathrm{LimsupOdd}_{n,d}$.

3 Universal ordered trees

3.1 Definition

An *ordered tree* is a prefix-closed set of sequences of a linearly ordered set. We refer to such sequences as tree *nodes*, we call the elements of such sequences *branching directions*, and we use the standard ancestor-descendent terminology for nodes. For example: node $\langle\rangle$ is the *root* of a tree; node $\langle x\rangle$ is the *child* of the root that is reached from the root via the branching direction x; node $\langle x, y\rangle$ is the *parent* of node $\langle x, y, z\rangle$; nodes $\langle\rangle$, $\langle x\rangle$, and $\langle x, y\rangle$ are *ancestors* of node $\langle x, y, z\rangle$; and nodes $\langle x, y\rangle$ and $\langle x, y, z\rangle$ are *descendants* of nodes $\langle\rangle$ and $\langle x\rangle$. Moreover, a node is a *leaf* if it does not have any descendants. A natural linear order on nodes in an ordered tree that we are going to use is the *lexicographic order* on sequences induced by the assumed linear order on the set of branching directions. For example, we have $\langle x\rangle < \langle x, y\rangle$, and $\langle x, y, w\rangle < \langle x, z\rangle$ if $y < z$.

An (n, h)*-universal (ordered) tree* is an ordered tree of height h, such that every ordered tree of height at most h and with at most n leaves can be isomorphically embedded into it; in such an embedding, the root of the tree must be mapped onto the root of the universal tree, and the children of every node must be mapped—injectively and in an order-preserving way—onto the children of its image.

Ordered trees are the key technical concept that underpins the design of the progress measure lifting algorithm [23] for solving parity games. For nearly two decades, this algorithm and its variants [1, 30, 6, 24, 10] have been consistently matching or beating the worst-case performance guarantees of the state-of-the-art algorithms. In this paper, we make explicit the implicit observations of Jurdziński and Lazić [24] that:

- any $(n, d/2)$-universal tree can be used as the data structure using which the progress measure lifting algorithm can solve any parity game with at most n vertices and at most d distinct priorities;
- the worst-case run-time analysis of the progress measure algorithm is dominated by the size of the universal tree that it uses.

In particular, Jurdziński and Lazić have (implicitly) given a construction of universal trees of quasi-polynomial size, and they have argued that the progress measure lifting algorithm can use their universal trees to achieve the state-of-the-art worst-case performance for solving parity games [24].

Theorem 1 (Jurdziński and Lazić [24, Lemmas 1 and 6])**.** *For all positive integers n and h, there is an (n, h)-universal tree with at most quasi-polynomial number of leaves; more specifically, it is at most $2n\binom{\lceil \lg n\rceil + h + 1}{h}$, which is polynomial in n if $h = O(\lg n)$, and it is $O\left(hn^{\lg(h/\lg n)+1.45}\right)$ if $h = \omega(\lg n)$.*

3.2 Lower bounds for universal trees

The main technical result in this section is a quasi-polynomial lower bound on the size of universal ordered trees that matches the upper bound of Jurdziński and Lazić [24] up to a small polynomial factor. It follows that the smallest universal ordered trees have quasi-polynomial size, and hence the worst-case performance of the succinct progress measure algorithm of Jurdziński and Lazić [24] cannot be improved to sub-quasi-polynomial by designing smaller universal ordered trees.[1]

Theorem 2. *For all positive integers* n *and* h*, every* (n, h)*-universal tree has at least* $\binom{\lfloor \lg n \rfloor + h - 1}{\lfloor \lg n \rfloor}$ *leaves, which is at least* $n^{\lg(h/\lg n)-1}$ *provided* $2h \leq n$*.*

This lower bound result shares some similarities with a result of Goldberg and Lifschitz [20], which is for universal trees of a different kind: the height is not bounded and the trees are not ordered.

Proof. We first give a derivation of the latter bound from the former: we show that

$$\binom{\lfloor \lg n \rfloor + h - 1}{\lfloor \lg n \rfloor} \geq n^{\lg(h/\lg n)-1} \quad \text{provided} \quad 2h \leq n$$

We start from the inequality $\left(\frac{k}{\ell}\right)^{\ell} \leq \binom{k}{\ell}$ applied to the binomial coefficient $\binom{\lfloor \lg n \rfloor + h - 1}{\lfloor \lg n \rfloor}$, and take the lg of both sides. This yields

$$\lg \binom{\lfloor \lg n \rfloor + h - 1}{\lfloor \lg n \rfloor} \geq \lfloor \lg n \rfloor \cdot \Big[\lg \big(\lfloor \lg n \rfloor + h - 1 \big) - \lg \lfloor \lg n \rfloor \Big] \geq$$
$$(\lg n - 1) \cdot (\lg h - \lg \lg n) \geq \lg n \cdot \big(\lg(h/\lg n) - 1 \big)$$

The second inequality follows since $n \geq 2$, and the third by the assumption $2h \leq n$.

To prove the first bound, we proceed by induction and show that any (n, h)-universal tree has at least $g(n, h)$ leaves, where

$$g(n, h) = \sum_{\delta=1}^{n} g(\lfloor n/\delta \rfloor, h - 1)$$

and $g(n, 1) = n, g(1, h) = 1$.

The bounds are clear for $h = 1$ or $n = 1$.

Let T be a (n, h)-universal tree, and $\delta \in \{1, \ldots, n\}$. We claim that the number of nodes at depth $h - 1$ of degree greater than or equal to δ is at least $g(\lfloor n/\delta \rfloor, h - 1)$.

Let T_δ be the subtree of T obtained by removing all leaves and all nodes at depth $h - 1$ of degree less than δ: the leaves of the tree T_δ have height exactly $h - 1$.

We argue that T_δ is $(\lfloor n/\delta \rfloor, h - 1)$-universal. Indeed, let t be a tree with $\lfloor n/\delta \rfloor$ leaves all at depth $h - 1$. To each leaf of t we append δ children, yielding the tree t_+ which has $\lfloor n/\delta \rfloor \cdot \delta \leq n$ leaves all at depth h. Since T is (n, h)-universal, the tree t_+ embeds into T. Observe that the embedding induces an embedding of t into T_δ, since the leaves of t have degree δ in t_+, hence are also in T_δ.

Let ℓ_δ be the number of nodes at depth $h - 1$ with degree exactly δ. So far we proved that the number of nodes at depth $h-1$ of degree greater than or equal to δ is at least $g(\lfloor n/\delta \rfloor, h-1)$, so

$$\sum_{i=\delta}^{n} \ell_i \geq g(\lfloor n/\delta \rfloor, h - 1).$$

[1] The quasi polynomial lower bound on the size of universal trees has appeared in the technical report [16], which has been merged into this paper.

Thus the number of leaves of T is

$$\sum_{i=1}^{n} \ell_i \cdot i = \sum_{\delta=1}^{n} \sum_{i=\delta}^{n} \ell_i \geq \sum_{\delta=1}^{n} g(\lfloor n/\delta \rfloor, h-1) = g(n,h).$$

It remains to prove that:

$$g(n,h) \geq \binom{\lfloor \lg n \rfloor + h - 1}{\lfloor \lg n \rfloor}.$$

Define $G(p,h) = g(2^p, h)$ for $p \geq 0$ and $h \geq 1$. Then we have

$$\begin{array}{rcl} G(p,h) & \geq & \sum_{k=0}^{p} G(p-k, h-1), \\ G(p,1) & \geq & 1, \\ G(0,h) & = & 1. \end{array}$$

To obtain a lower bound on G we define $\overline{G}$ by

$$\begin{array}{rcl} \overline{G}(p,h) & = & \overline{G}(p,h-1) + \overline{G}(p-1,h), \\ \overline{G}(p,1) & = & 1, \\ \overline{G}(0,h) & = & 1, \end{array}$$

so that $G(p,h) \geq \overline{G}(p,h)$. We verify by induction that $\overline{G}(p,h) = \binom{p+h-1}{p}$, which follows from Pascal's identity

$$\binom{p+h-1}{p} = \binom{p+h-2}{p} + \binom{p+h-2}{p-1}.$$

This implies that $G(p,h) \geq \binom{p+h-1}{p}$. Putting everything together we obtain

$$g(n,h) \geq \binom{\lfloor \lg n \rfloor + h - 1}{\lfloor \lg n \rfloor}.$$

□

4 Universal trees grow inside separating automata: the lower bound

The main result of this section is to show that any separating automaton contains a universal tree. Combined with Theorem 2 this implies a quasipolynomial lower bound on the number of states of any separating automata. Since all known quasipolynomial time algorithms for solving parity games implicitly or explicitly construct a separating automaton, the size of which dictates the complexity, this result explains in a unified way the quasipolynomial barrier and pinpoints the underlying combinatorial structure behind all recent algorithms.

Theorem 3. *Every non-deterministic safety automaton that separates* $\mathrm{PosCyclEven}_{n,d}$ *from* $\mathrm{LimsupOdd}_{n,d}$ *has at least* $\binom{\lfloor \lg n \rfloor + d/2 - 1}{\lfloor \lg n \rfloor}$ *states, which is at least* $n^{\lg(d/\lg n) - 2}$.

The rest of this section is devoted to the proof of Theorem 3. The proof shows that, if an automaton $\mathcal{A}$ fulfils conditions of Theorem 3, then it necessarily has the structure of an $(n, d/2)$-universal tree (cf. Section 3.1 for a definition; recall that d is assumed to be even) in its states. The core of the proof of Theorem 3 is the following lemma.

Lemma 1. *Let* $\mathcal{A}$ *be a non-deterministic safety automaton that separates* $\mathrm{PosCyclEven}_{n,d}$ *from* $\mathrm{LimsupOdd}_{n,d}$. *Then there is an injective mapping from the leaves of some* $(n, d/2)$*-universal tree into the states of* $\mathcal{A}$.

Note that once the lemma is established, our main theorem follows from the quasi-polynomial lower bounds for universal trees stated in Theorem 2, since $d \leq n$.

We prove Lemma 1 in two steps. In the first step, we show that any safety automaton separating $\mathrm{PosCyclEven}_{n,d}$ from $\mathrm{LimsupOdd}_{n,d}$ must have a special structure, which we call *tree-like* (Lemma 2). Then, assuming this special structure, we prove that there is a universal tree whose leaves embed in the states of $\mathcal{A}$ (Lemma 3).

Linear quasi-order. A binary relation $\preceq$ on a set X is called a *linear quasi-order* if it is reflexive, transitive, and total (i.e. such that for all $x, y \in X$ either $x \preceq y$, or $y \preceq x$, or both). If $x \preceq y$ and $y \not\preceq x$, then we write $x \prec y$. An equivalence class of $\preceq$ is a maximal set $e \subseteq X$ such that $x \preceq y$ and $y \preceq x$ for all $x, y \in e$. It is well-known that the equivalence classes of $\preceq$ form a partition of X and given two equivalence classes e and e', there exist $x \in e$ and $y \in e'$ such that $x \preceq y$ if and only if for all $x \in e$ and $y \in e'$, $x \preceq y$. When this is the case, it is denoted by $e \preceq e'$, and $e \prec e'$ when additionally $e' \not\preceq e$.

Given two linear quasi-orders $\preceq_1$ and $\preceq_2$, we write $\preceq_1 \subseteq \preceq_2$ if for all $x, y \in X$, $x \preceq_1 y$ implies $x \preceq_2 y$. In that case, any equivalence class of $\preceq_2$ is formed with a partition of equivalence classes of $\preceq_1$. In other word, an equivalence class of $\preceq_1$ is included in a unique equivalence class of $\preceq_2$ and disjoint from the other ones.

Tree decomposition.

Recall that we consider the alphabet $\Sigma_{n,d} = \{1, 2, \ldots, n\} \times \{1, 2, \ldots, d\}$. The second component of a letter a is called its priority and is denoted by $\mathrm{pr}(a)$.

For an automaton $\mathcal{A}$ over $\Sigma_{n,d}$, a *tree-decomposition* of $\mathcal{A}$ is a sequence of linear quasi-orders $\preceq_1 \subseteq \preceq_3 \subseteq \cdots \subseteq \preceq_{d+1}$ on the set of non-rejecting states of $\mathcal{A}$ such that:
1) if $p \xrightarrow{a} q$ then $p \succeq_{2i+1} q$ whenever $2i + 1 > \mathrm{pr}(a)$,
2) if $p \xrightarrow{a} q$ and $\mathrm{pr}(a)$ is odd then $p \succ_{2i+1} q$ whenever $2i + 1 \leq \mathrm{pr}(a)$, and
3) $\preceq_{d+1}$ has a single equivalence class, containing all non-rejecting states of $\mathcal{A}$.
In other words, reading a letter cannot cause an increase with respect to orders with indices greater than its priority, and additionally reading a letter with odd priority necessarily causes a decrease with respect to orders with indices smaller than or equal to this latter. If there is a tree-decomposition of $\mathcal{A}$, we call $\mathcal{A}$ to be *tree-like*. Given a tree decomposition D of $\mathcal{A}$, we define the *D-tree* of $\mathcal{A}$, denoted $\mathrm{tree}_D(\mathcal{A})$, as follows (recall the notation on trees from Section 5.3):

- nodes of $\mathrm{tree}_D(\mathcal{A})$ are sequences $\langle e_{d-1}, e_{d-3}, \ldots, e_p \rangle$, where $p \in \{1, 3, 5, \ldots, d-1\}$, and where every branching direction e_i is an equivalence class of the quasi-order $\preceq_i$, such that $e_{d-1} \supseteq e_{d-3} \supseteq \cdots \supseteq e_p$,
- the order between branching directions e_i, e'_i being equivalence classes of $\preceq_i$ is $e_i < e'_i$ when $e_i \prec_i e'_i$.

Notice that for a non-rejecting state q of $\mathcal{A}$, there is a unique sequence $e_{d-1} \supseteq e_{d-3} \supseteq \cdots \supseteq e_1$ where for all i, e_i is an equivalence class of $\preceq_i$ containing q. One can thus assign a non-rejecting state q to the corresponding leaf $\langle e_{d-1}, e_{d-3}, \ldots, e_1 \rangle$, in such a way that for all odd priority $p \in \{1, 3, \ldots, d_1\}$, $q \prec_p q'$ if and only if the p-truncation of the leaf assigned to q is smaller in the lexicographic order than the p-truncation of the leaf assigned to q'.

An automaton is said *accessible* if for every state q there exists a run from an initial state to q, and moreover, if q is non-rejecting, there exists such a run which does not go through any rejecting state. The first of the two steps of the proof of Lemma 1 can be summarized by the following lemma.

Lemma 2. *Every accessible non-deterministic safety automaton separating* $\mathrm{PosCyclEven}_{n,d}$ *from* $\mathrm{LimsupOdd}_{n,d}$ *is tree-like.*

Proof. We define the linear quasi-orders inductively starting from higher indices, that is, in the order $\preceq_{d+1}, \preceq_{d-1}, \dots, \preceq_1$. As $\preceq_{d+1}$ we take the quasi-order in which all non-rejecting states are in a single equivalence class. Condition 3) is already satisfied.

Assume now that the quasi-orders $\preceq_{d+1}, \preceq_{d-1}, \dots, \preceq_{2i+3}$ are already defined, and that they fulfil Conditions 1) and 2). We define $\preceq_{2i+1}$ as a refinement of $\preceq_{2i+3}$. If $q \prec_{2i+3} r$ then we set $q \prec_{2i+1} r$ as well. So it remains to define whether $\preceq_{2i+1}$ holds for states q and r in the same equivalence class of $\preceq_{2i+3}$. Before this, notice that we are already guaranteed that $\preceq_{2i+1}$ satisfies Conditions 1) and 2) for letters a of priority higher than $2i+1$. Indeed, Condition 1) talks only about letters with priorities smaller than $2i+1$. By Condition 2) applied to $\preceq_{2i+3}$ we know that if $\mathrm{pr}(a)$ is odd, and $\mathrm{pr}(a) \geq 2i+3$ (i.e., $\mathrm{pr}(a) > 2i+1$), and $q \xrightarrow{a} r$, then $q \succ_{2i+3} r$, so also $q \succ_{2i+1} r$. Therefore we only need to define $\preceq_{2i+1}$ in such a way that Condition 1) is satisfied for letters with priorities from the set $\{1, \dots, 2i\}$ and Condition 2) is satisfied for letters with priority $2i+1$. Intuitively speaking, now we only have to care about letters with priorities at most $2i+1$.

Let e be an arbitrary equivalence class of $\preceq_{2i+3}$. Consider an automaton $\mathcal{A}_{e,2i+1}$, which contains a part of $\mathcal{A}$ consisting of only these states that belong to the class e, and only these transitions that have endpoints inside e and are labelled by letters with priorities at most $2i+1$ (notice that $\mathcal{A}_{e,2i+1}$ needs not to be complete). Observe now that there cannot be any cycle in $\mathcal{A}_{e,2i+1}$ such that the maximal priority of a letter on that cycle is odd. Indeed, otherwise we could consider the infinite run that first reaches this cycle from an initial state in $\mathcal{A}$ and without visiting any rejecting state (which is possible under the assumption that $\mathcal{A}$ is accessible), and then goes around this cycle forever (note that none of the state in the cycle is rejecting); the word read by this run would be in $\mathrm{LimsupOdd}_{n,d}$, but it would be accepted by the automaton (while it should not). Notice that this exactly the point in the proof of Lemma 2 where we need assumption about rejecting all the words from $\mathrm{LimsupOdd}_{n,d}$. Rejecting all the words from $\mathrm{PosCyclOdd}_{n,d}$ would not be sufficient, as the above described word may have some even cycle and thus does not have to belong to $\mathrm{PosCyclOdd}_{n,d}$. Therefore, on every path in $\mathcal{A}_{e,2i+1}$ at most $|\mathcal{A}_{e,2i+1}| - 1$ edges are labelled by letters with priority $2i+1$. Let the *resistance* of a state in $\mathcal{A}_{e,2i+1}$ be the maximal number of edges labelled by letters with priority $2i+1$ over all paths starting in that state. By the previous observation, the resistance of a state is always finite. Having defined the resistance, for two states q, r in $\mathcal{A}_{e,2i+1}$ we say that $q \preceq_{2i+1} r$ if the resistance of q is not greater than the resistance of r.

We have to show that such a definition of $\preceq_{2i+1}$ indeed fulfils Conditions 1) and 2). For Condition 1) we have to check that letters with priority smaller than $2i+1$ never cause an increase in the quasi-order $\preceq_{2i+1}$. Consider a with $\mathrm{pr}(a) < 2i+1$ such that $q \xrightarrow{a} r$. We know by the induction assumption that $q \succeq_{2i+3} r$. If $q \succ_{2i+3} r$ then as well $q \succ_{2i+1} r$, and we are done with Condition 1). Otherwise, q and r are in the same equivalence class of $\preceq_{2i+3}$, and it is enough to show that q has resistance not greater than r. However, if r has resistance k and $q \xrightarrow{a} r$, then q as well has resistance at least k, as one can take a path starting from r that has k edges labelled by letters with priority $2i+1$, and prepend it by the edge from q to r reading a. Thus, q cannot have smaller resistance than r. This implies that indeed $q \succeq_{2i+1} r$, and in effect Condition 1) is fulfilled. For Condition 2) we need to show that letters with priority $2i+1$ cause a decrease with respect to $\preceq_{2i+1}$. Let $q \xrightarrow{a} r$, where $\mathrm{pr}(a) = 2i+1$. By the induction assumption for $\preceq_{2i+3}$, from Condition 1) we know that $q \succeq_{2i+3} r$. If $q \succ_{2i+3} r$, then we are done as before, so assume that q and r are in the same equivalence class of $\preceq_{2i+3}$. In order to show that $q \succ_{2i+1} r$ it is enough to show that q has greater resistance than r. Assume that the resistance of r equals k, so there is a path ρ starting in r with k edges labelled by letters with priority $2i+1$. Then by $q \xrightarrow{a} r$ and $\mathrm{pr}(a) = 2i+1$ we know that the path using the edge from q to r, and then continuing as ρ, starts at q and has $k+1$ edges labelled by letters with priority $2i+1$. Thus, the resistance of q is greater than the resistance of r, hence $q \succ_{2i+1} r$, which finishes the proof of Condition 2). □

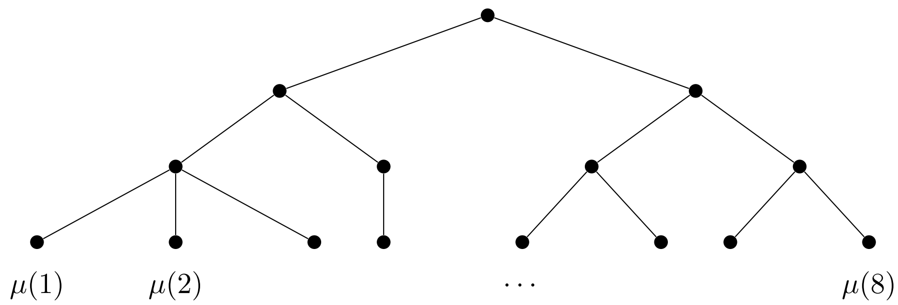
Now we aim at proving Lemma 1. First notice that we can assume that $\mathcal{A}$ is accessible, by removing the states that are not reachable from initial states and by making rejecting the non-rejecting states which are only reachable by visiting a rejecting state. By Lemma 2 we know that there is a tree-decomposition D of $\mathcal{A}$. We are going to prove that $\text{tree}_D(\mathcal{A})$ is a $(n, d/2)$-universal tree and that there is an injective mapping from its leaves into the states of $\mathcal{A}$, which will finish the proof of Lemma 1.

By definition, there is a one-to-one function between the leaves of $\text{tree}_D(\mathcal{A})$ and the equivalence classes of $\preceq_1$, mapping a leaf $\langle e_{d-1}, e_{d-3}, \ldots, e_1\rangle$ to e_1. For every equivalence class e of $\preceq_1$, pick a state q_e in e. Consider the function mapping a leaf $\langle e_{d-1}, e_{d-3}, \ldots, e_1\rangle$ of $\text{tree}_D(\mathcal{A})$ to the state q_{e_1}. This function is injective from the leaves of $\text{tree}_D(\mathcal{A})$ into the states of the automaton. Thus, in order to prove Lemma 1 it remains to show the following lemma.

Lemma 3. *For every tree-decomposition D of an automaton $\mathcal{A}$, the D-tree of $\mathcal{A}$ is $(n, d/2)$-universal.*

Proof. It is enough to show that every tree of height at most $d/2$ and with at most n leaves can be embedded in $\text{tree}_D(\mathcal{A})$. Thus, take such a tree t. Without loss of generality, we assume that t has exactly n leaves, and that all the leaves are on depth $d/2$. Otherwise, if the number of leaves is less than n, add some branches to the tree so as to have exactly n leaves; and if a leaf is not at depth $d/2$, add a path from this leaf so as to reach depth $d/2$. If such a tree can be embedded in $\text{tree}_D(\mathcal{A})$, the original one too.

As a running example, consider the following tree t with $n = 8$ and $d = 6$.



The proof follows the following steps:

1. Construction of a game G_t from t, where Even owns no vertices but wins from every vertex,

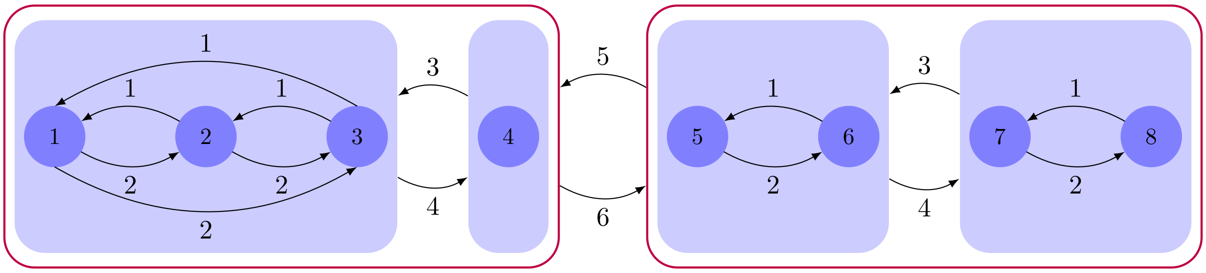
2. Construction of an infinite play w_t in G_t (which is thus in $\text{PosCyclEven}_{n,d}$),

3. Using w_t, proof that t is embedded in $\text{tree}_D(\mathcal{A})$.

1. Construction of G_t.

As the set of vertices of G_t we take $\{1, \ldots, n\}$, where all vertices belong to Odd, and n is the starting vertex. For a vertex $u \in \{1, \ldots, n\}$, let $\mu(u)$ denote the u-th leaf of t in the lexicographic order. Let u and v be two vertices such that $\mu(u) > \mu(v)$ and p the smallest element of $\{1, \ldots, d\}$ such that $\mu(u)|_p = \mu(v)|_p$. Note that p must be even (in the tree this corresponds to consider the smallest common ancestor of $\mu(u)$ and $\mu(v)$). We define an edge from u to v with priority $p - 1$ and one from v to u with priority p. We also define self-loops labelled by every even priority p around every node. Remark that by definition Even wins from every vertex.

On our running example, this construction is depicted below. For the sake of readability, an edge drawn between two sets of vertices A and B means that there is such an edge between all the pairs of vertices from $A \times B$; and the self-loops are not represented.



2. Construction of w_t.

We now construct an infinite word w_t which is a play in G_t. Note that in such a case, $w_t \in \text{PosCyclEven}_{n,d}$.

Let us first explain the construction on our running example. The play is as follows:

1. Follow the cycle between vertices 8 and 7 a big number of times, going through edge priority 1, 2, 1, 2...,
2. Reach vertex 6 from vertex 7, with edge priority 3,
3. Follow the cycle between vertices 6 and 5 a big number of times, going through edge priority 1, 2, 1, 2...,
4. Reach vertex 8 from vertex 5, with edge priority 4,
5. Repeat steps 1, 2, 3 and 4 a big number of times,
6. Reach vertex 4 from vertex 5, with edge priority 5,
7. Take the self-loop around vertex 4 with edge priority 2 (the self-loops are not drawn on the picture of G_t) a big number of times,
8. Reach vertex 3 from vertex 4, with edge priority 3,
9. Follow the cycle going through vertices 3, 2 and 1 a big number of times, with edge priority 1, 1, 2, 1, 1, 2...,
10. Reach vertex 4 from vertex 1, with edge priority 4,
11. Repeat steps 7, 8, 9 and 10 a big number of times.
12. After all those steps, to make the run infinite, once reached vertex 1 after step 9, take the self-loop around 1 with edge priority 2 infinitely many times.

We give now the technical definition of such a play. To this end, for every node x of t, we define two finite words w_x and u_x. If x is a leaf, we set $w_x = u_x = \varepsilon$.

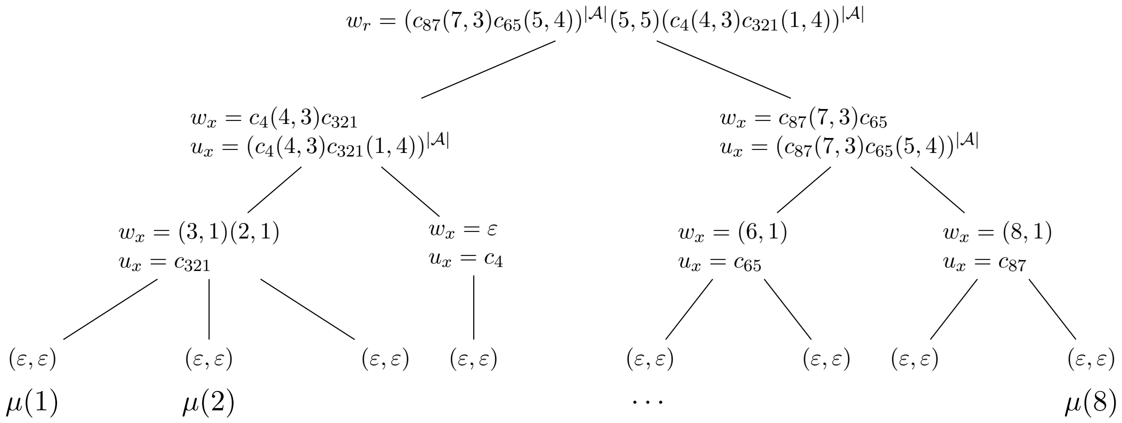
Otherwise, for a node y, let $\mathit{fl}(y)$ denote the smallest number v such that $\mu(v)$ is a descendant of y. In other words, $\mu(\mathit{fl}(y))$ is the leftmost leaf in the subtree rooted in y. In our example, if r is the root of the tree and y its right child then $\mathit{fl}(r) = 1$ and $\mathit{fl}(y) = 5$.

Let x be an inner node at distance $k > 0$ from the leaves, and let $x_1, \ldots, x_\ell$ be the children of x, listed from left to right. Then we set:

$$w_x = u_{x_\ell}\,(\mathit{fl}(x_\ell), 2k-1)\,u_{x_{\ell-1}}\,(\mathit{fl}(x_{\ell-1}), 2k-1)\;\ldots\;u_{x_2}\,(\mathit{fl}(x_2), 2k-1)\,u_{x_1} \qquad \text{and}$$
$$u_x = \left(w_x\,(\mathit{fl}(x), 2k)\right)^{|\mathcal{A}|}.$$

where $|\mathcal{A}|$ is the number of states in $\mathcal{A}$. We set $w_t = w_r(1,2)^\omega$ where r is the root of the tree $\text{tree}_D(\mathcal{A})$. Notice that the highest priority appearing in w_r is at most d (because the height of t is $d/2$), and so $w_t \in \Sigma^\omega_{n,d}$.

In the following picture, we give the pairs (w_x, u_x) associated with the nodes x of our running example, where c_{321} denote the word $((3,1)(2,1)(1,2))^{|\mathcal{A}|}$, c_4 the word $(4,2)^{|\mathcal{A}|}$, c_{65} the word $((6,1)(5,2))^{|\mathcal{A}|}$ and c_{87} the $((8,1)(7,2))^{|\mathcal{A}|}$.



Thus, we obtain:

$$w_t = (c_{87}(7,3)c_{65}(5,4))^{|\mathcal{A}|}(5,5)(c_4(4,3)c_{321}(1,4))^{|\mathcal{A}|}(1,2)^{\omega}$$

Let us prove now that w_t encodes a play in G_t. To this end, notice that all vertices v appearing in w_x or u_x are such that $\mu(v)$ is a descendant of x. In w_t we have three kinds of letters:

- letters $(\mathit{fl}(x_i), 2k-1)$, appearing in the definition of w_x;
- letters $(\mathit{fl}(x), 2k)$, appearing in the definition of u_x;
- letters $(1,2)$ appearing in the definition of w_t.

Consider x at distance k from the leaves, and $x_1, \dots, x_\ell$ be the children of x, listed from left to right. A letter $(\mathit{fl}(x_i), 2k-1)$ is followed either by a letter (v,p) of $u_{x_{i-1}}$, or (if this word is empty) by $(v,p) = (\mathit{fl}(x_{i-1}), 2k-1)$, or by $(v,p) = (\mathit{fl}(x), 2k)$; By definition, in all of those cases, because x_i is the right sibling of x_{i-1}, the lowest common ancestor of $\mu(\mathit{fl}(x_i))$ and $\mu(v)$ is x. Thus there is an edge from $\mathit{fl}(x_i)$ to v with priority $2k-1$. A letter $(\mathit{fl}(x), 2k)$ is followed either by a letter (v,p) of w_x, in which case, because $\mu(v)$ must be on the right of $\mu(\mathit{fl}(x))$, and their lowest common ancestor is x, there is an edge in G_t from $\mathit{fl}(x)$ to v with priority $2k$; or by $(v,p) = (\mathit{fl}(x), 2k-1)$, which is valid because there is a self-loop around $\mathit{fl}(x)$ of priority $2k$; or if x is the root, by $(v,p) = (1,2)$, in which case, necessarily $(\mathit{fl}(x), 2k) = (1, 2k)$ and it is valid since there is a self-loop around 1 with priority $2k$. A letter $(1,2)$ is followed by $(1,2)$, and by definition there is a self-loop around 1 with priority 2. Thus, w_t encodes a play in G_t.

3. Proof that t is embedded in $\mathbf{tree}_D(\mathcal{A})$.

We prove the following claim for every node x of t, being at distance k from the leaves.

Claim 1. *Let ρ be a finite run of $\mathcal{A}$ not visiting rejecting states (ρ needs not to start in an initial state), that either:*

- *reads w_x and is such that all states visited by ρ belong to the same equivalence class of $\preceq_{2k+1}$, or*
- *reads u_x.*

Then there exists a node $x' = \langle e_{d-1}, e_{d-3}, \dots, e_{2k+1}\rangle$ of $\mathrm{tree}_D(\mathcal{A})$ such that
1) the classes $e_{d-1}, e_{d-3}, \dots, e_{2k+1}$ contain some state visited by ρ,
2) the subtree of t rooted at x embeds in the subtree of $\mathrm{tree}_D(\mathcal{A})$ rooted at x'.

Let us first show how this claim finishes the proof of Lemma 3. Because $w_t \in \mathrm{PosCyclEven}_{n,d}$, there is a run of $\mathcal{A}$ that reads w_t and never visits rejecting states; let ρ be the prefix of this run that reads w_r where r is the root of t. Recall that the distance from the root of t to its leaves equals $d/2$. All states visited by ρ belong to the same equivalence class of $\preceq_{d+1}$, because this quasi-order has only one equivalence class. Using the claim for the run ρ we obtain that t embeds in $\mathrm{tree}_D(\mathcal{A})$ (notice that x' is necessarily the root of $\mathrm{tree}_D(\mathcal{A})$).

We now prove the claim by induction on k, that is, on the distance of the node x to the leaves. The induction base, for x being a leaf, is simple. In this case $w_x = u_x = \varepsilon$, and as branching directions of the node $x' = \langle e_{d-1}, e_{d-3}, \ldots, e_1\rangle$ we take equivalence classes of $\preceq_{d-1}, \preceq_{d-3}, \ldots, \preceq_1$ containing the only state of ρ; such a node indeed fulfils Conditions 1)–2). Let us focus now on the induction step. Suppose first that ρ reads w_x, and that all visited states belong to the same equivalence class of $\preceq_{2k+1}$. Let $x_1, \ldots, x_\ell$ be the children of x. Recall that

$$w_x = u_{x_\ell}\,(\mathit{fl}(x_\ell), 2k-1)\,u_{x_{\ell-1}}\,(\mathit{fl}(x_{\ell-1}), 2k-1)\,\ldots\,u_{x_2}\,(\mathit{fl}(x_2), 2k-1)\,u_{x_1}\,.$$

We can divide ρ into fragments $\rho_\ell, \rho_{\ell-1}, \ldots, \rho_1$, where ρ_i reads u_{x_i}. For every $i \in \{1, \ldots, \ell\}$, by the induction assumption, we can find a node $x'_i = \langle e_{i,d-1}, e_{i,d-3}, \ldots, e_{i,2k-1}\rangle$ that fulfils Conditions 1)–2) with respect to ρ_i and x_i. By assumption, all states visited by ρ belong to the same equivalence class of $\preceq_{2k+1}$, hence also to the same equivalence class of $\preceq_j$ for all $j \in \{2k+1, 2k+3, \ldots, d-1\}$. On the other hand, by Condition 2), the classes $e_{i,j}$ for $i \in \{1, \ldots, \ell\}$ and $j \in \{2k+1, 2k+3, \ldots, d-1\}$ contain some state visited by ρ. It follows that $e_{i,j} = e_{i',j}$ for all $i, i' \in \{1, \ldots, r\}$ and $j \in \{2k+1, 2k+3, \ldots, d-1\}$, which implies that the nodes $x'_1, \ldots, x'_\ell$ are siblings, having a common parent x'. It is easy to see that the node x' satisfies Conditions 1)–2).

Suppose now that ρ reads $u_x = (w_x\,(\mathit{fl}(x), 2k))^{|\mathcal{A}|}$. By the fact that D is a tree-decomposition of $\mathcal{A}$ we know that no transition in ρ goes up with respect to the quasi-order $\preceq_{2k+1}$, because all letters of u_x have priority at most $2k$. As $\mathcal{A}$ has at most $|\mathcal{A}|$ states, it means that at most $|\mathcal{A}|-1$ transitions of the considered run cause a decrease with respect to $\preceq_{2k+1}$. This implies that there is a part of this run that reads w_x and contains no increase nor decrease with respect to $\preceq_{2k+1}$, that is, all states visited by this part of the run belong to the same equivalence class of $\preceq_{2k+1}$. We continue with this part of the run as in the previous paragraph, and in this way we finish the proof of the claim. □

5 Separators everywhere

In this section we survey the three distinct techniques that have been developed so far in design of quasi-polynomial algorithms for solving parity games. There are some original observations and results here, but they are not as significant and original as our main technical result in Section 4. We view this section as a constructive first step towards a conceptual unification of the three distinct technical approaches. The main unifying aspect that we highlight in this section is that all the three approaches yield constructions of separating safety automata of quasi-polynomial size, which provides evidence of significance of our main technical result: the quasi-polynomial lower bound on the size of separators (Theorem 3) forms a barrier that all of those approaches need to overcome in order to further significantly improve the known complexity of solving parity games. We note that, in contrast to the results of Calude et al. [5] and Lehtinen [27], not all of the proposed quasi-polynomial algorithms explicitly construct separating automata or other objects of (at least) quasi-polynomial size [24, 15], but in the worst case, they too enumerate structures that form the states of the related separating automata (leaves in a universal tree and play summaries, respectively).

Inspired by the prominent role that universal trees play in the structure of separating automata, as exploited in our main technical result (Theorem 3), we place a particular spotlight on the concepts of progress measures and universal trees, whose theory we summarize in Section 5.2. In Section 5.1, as a warm-up, we describe a separating safety automaton due to Bernet et al. [1] whose size is not of quasi-polynomial (it is exponential). Its simple design offers the pedagogic value of motivating the more general construction based on universal trees in Section 5.3. In Section 5.4, we briefly discuss the observation of Bojańczyk and Czerwiński [3] that Calude et al.'s [5] play summaries construction can be straightforwardly interpreted as defining a separating automaton; we refer the reader to their technical exposition, which is highly readable. Finally, in Section 5.6, we discuss how to adapt the separation approach to also encapsulate the most recent quasi-polynomial algorithm for solving parity games by Lehtinen [27], based on register games. This requires care because, unlike the constructions based on play summaries and universal trees, separating automata that underpin Lehtinen's reduction from parity games to register games seem to require non-determinism, and in general, Proposition 1 does not hold for non-deterministic automata.

5.1 Simple separating safety automata

As a warm-up, we present a simple "multi-counter" separating automaton that is implicit in the work of Bernet et al. [1]. For all positive integers n and d, such that d is even, we define the automaton $\mathcal{C}_{n,d}$ that, for every odd priority p, $1 \leq p \leq d-1$, keeps a counter c_p that stores the number of occurrences of priority p since the last occurrence of a priority larger than p (even or odd). It is a safety automaton: it rejects a word immediately once the integer stored in any of the $d/2$ counters exceeds n.

In fact, instead of "counting up" (from 0 to n), we prefer to "count down" (from n to 0), which is equivalent, but it aligns better with the definition of progress measures. More formally, we define the deterministic safety automaton $\mathcal{C}_{n,d}$ in the following way:

- the set of states of $\mathcal{C}_{n,d}$ is the set of $d/2$-sequences $\langle c_{d-1}, c_{d-3}, \ldots, c_1 \rangle$, such that c_p is an integer such that $0 \leq c_p \leq n$ for every odd p, $1 \leq p \leq d-1$; and it also contains an additional *rejecting* state `reject`;
- the initial state is $\langle n, n, \ldots, n \rangle$;
- the transition function δ is defined as follows:

$$\delta\big(\langle c_{d-1}, c_{d-3}, \ldots, c_1 \rangle, (v,p)\big) = \begin{cases} \langle c_{d-1}, c_{d-3}, \ldots, c_{p+1}, n, \ldots, n \rangle & \text{if } p \text{ is even,} \\ \langle c_{d-1}, c_{d-3}, \ldots, c_p - 1, n, \ldots, n \rangle & \text{if } p \text{ is odd and } c_p > 0, \\ \texttt{reject} & \text{if } p \text{ is odd and } c_p = 0; \end{cases}$$

  and $\delta\big(\texttt{reject}, (v,p)\big) = \texttt{reject}$ for all $(v,p) \in \Sigma_{n,d}$.

Note that the size of automaton $\mathcal{C}_{n,d}$ is $\Theta(n^{d/2})$.

Proposition 2 (Bernet et al. [1], Bojańczyk and Czerwiński [3])**.** *The automaton $\mathcal{C}_{n,d}$ is a safety (n,d)-separator of* $\text{AllCyclEven}_{n,d}$ *and* $\text{LimsupOdd}_{n,d}$.

Proof. Firstly, we argue that if the unique run of $\mathcal{C}_{n,d}$ on an infinite word contains an occurrence of the rejecting state then the word is not in $\text{AllCyclEven}_{n,d}$. Indeed, the only reason for the unique run of $\mathcal{C}_{n,d}$ to reach the rejecting state is that the state reached after reading some prefix of the word is $\langle c_{d-1}, c_{d-3}, \ldots, c_1 \rangle$ with $c_p = 0$ for an odd p, and (v,p) is subsequently read. It is easily seen that for this to happen, there must be a suffix of the prefix in which there are n occurrences of priority p and no priority higher than p occurs, and the currently read letter is the $(n+1)$-st occurrence of priority p in the prefix. By the pigeonhole principle, it follows that there is a cycle in the infinite word whose highest priority is p, which is odd.

Secondly, we argue that if a word is in $\text{LimsupOdd}_{n,d}$ then the unique run of $\mathcal{C}_{n,d}$ on the word contains an occurrence of the rejecting state. Consider an infinite suffix of the word in which all priorities occur infinitely many times. Unless the unique run reached the rejecting state on the corresponding prefix already, let $\langle c_{d-1}, c_{d-3}, \ldots, c_1 \rangle$ be the state reached in the unique run at the end of the prefix. Note that the highest priority that occurs in the suffix is odd, and by the assumption that the word is in $\text{LimsupOdd}_{n,d}$, there are infinitely many occurrences of it. Take the shortest prefix of the suffix in which the highest priority p occurs $n - c_p + 1$ times. The unique run of $\mathcal{C}_{n,d}$ on the original infinite word reaches the rejecting state upon reading that prefix. □

5.2 Progress measures and universal trees

A fundamental concept that has continued to play a key role throughout the development of the theory of and algorithms for solving parity games is *progress measures* [11, 23, 30, 24, 15]. In this section we summarise the concepts underlying the recently discovered link between quasi-polynomial solvability of parity games and the existence of small (quasi-polynomial size) universal trees that was implicitly established by Jurdziński and Lazić [24]. This is highly relevant to this paper on the power and limitations of the separation approach because:

- as we have shown in Section 4, every separating automaton contains a universal tree hidden in its state space, and hence every separating automaton is at least as big as any lower bound on the size of universal trees;
- as we show in Section 5.3, small universal trees yield small separating automata.

A *progress labelling* of a parity game is a mapping μ from the vertices in the game graph to leaves in an ordered tree; without loss of generality we assume that every leaf has depth $d/2$. We write $\langle m_{d-1}, m_{d-3}, \ldots, m_1 \rangle$ to denote such a leaf, and for every priority p, $1 \leq p \leq d$, we define its p-truncation $\langle m_{d-1}, m_{d-3}, \ldots, m_1 \rangle|_p$ to be the sequence $\langle m_{d-1}, m_{d-3}, \ldots, m_p \rangle$ if p is odd, and $\langle m_{d-1}, m_{d-3}, \ldots, m_{p+1} \rangle$ if p is even. We say that a progress labelling μ of the game is a *progress measure* if the following *progress condition* holds for every edge (v, u) in the strategy subgraph of some positional strategy for Even:

- if p is even then $\mu(v)|_{\pi(v,u)} \geq \mu(u)|_{\pi(v,u)}$;
- if p is odd then $\mu(v)|_{\pi(v,u)} > \mu(u)|_{\pi(v,u)}$.

We recommend inspecting the (brief and elementary) proof of [24, Lemma 2], which establishes that every cycle in the strategy subgraph whose all edges satisfy the progress condition is even. It gives a quick insight into the fundamental properties of progress measures and it shows the easy implication in the following theorem that establishes progress measures as witnesses of winning strategies in parity games.

Theorem 4 (Emerson and Jutla [11], Jurdziński [23])**.** *Even has a winning strategy from every vertex in a parity game if and only if there is a progress measure on the game graph.*

It is a straightforward but fruitful observation of Jurdziński and Lazić [24] that a progress measure on a game graph with n vertices and at most d distinct edge priorities is a mapping from the vertices in the game graph to nodes in an ordered tree *of height at most $d/2$ and with at most n leaves* (all subtrees that no vertex is mapped to can be pruned). It further motivates the second fundamental concept we explore in this section—*universal trees*, which we have already defined in Section 3.1. The following proposition follows directly from the above "straightforward but fruitful" observation and the definition of a universal tree.

Proposition 3 (Jurdziński and Lazić [24])**.** *Every progress measure on a graph with n vertices and with at most d priorities can be modified to map into any $(n, d/2)$-universal tree.*

A key conclusion following from this simple fact is that every universal tree is a natural data structure in which the search for a progress measure (i.e., a witness for a winning strategy for Even) can be carried out, and that the time needed for this search to uncover one can be (up to a small polynomial factor) made proportional to the size of the universal tree [24].

5.3 Separating safety automata from progress measures and universal trees

Jurdziński and Lazić have shown how to construct separators of AllCyclEven and LimsupOdd that are of quasi-polynomial size, using nodes in their particular universal trees as states [25, Theorem 9]. We show, instead, how to construct separators of PosCyclEven and LimsupOdd, using nodes of *any* universal tree as states. The harder part of the separation property for those automata is proved using existence of witnesses of winning strategies in the form of progress measures that map into a universal tree (Theorem 4 and Proposition 3).

For positive integers n and d, such that d is even, let $L_{n,d/2}$ be the set of leaves in an $(n, d/2)$-universal tree. The definition of the deterministic safety automaton $\mathcal{U}_{n,d}$ bears similarity to the definition of the simple "multi-counter" separator $\mathcal{S}_{n,d}$ from Section 5.1. Again, the states are $d/2$-sequences of "counters", but the "counting down" is done in a more abstract way than in $\mathcal{S}_{n,d}$, using instead the natural lexicographic order on the nodes of the universal tree.

More formally, we define a deterministic safety automaton $\mathcal{U}_{n,d}$ in the following way:

- the set of states of $\mathcal{U}_{n,d}$ is the set $L_{n,d/2}$ of leaves in the $(n, d/2)$-universal tree;
- the initial state is the largest leaf (in the lexicographic tree order);
- the transition function δ is defined as follows:

$$\delta\big(s, (v,p)\big) = \begin{cases} \text{the largest leaf } s' \text{ such that } s|_p = s'|_p & \text{if } p \text{ is even,} \\ \text{the largest leaf } s' \text{ such that } s|_p > s'|_p & \text{if } p \text{ is odd,} \\ \texttt{reject} & \text{if } p \text{ is odd and no such leaf exists} \end{cases}$$

  if $s \neq \texttt{reject}$; and $\delta\big(\texttt{reject}, (v,p)\big) = \texttt{reject}$ for all $(v,p) \in \Sigma_{n,d}$.

Remark 1. *Let $\mathcal{T}_{n,d}$ be an ordered tree in which all the leaves have depth $d/2$ and every non-leaf node has exactly $n+1$ children; note that $\mathcal{T}_{n,d}$ is trivially an $(n, d/2)$-universal tree. An instructive exercise that we recommend is to compare the structures of the automaton $\mathcal{U}_{n,d}$ based on the $(n, d/2)$-universal tree $\mathcal{T}_{n,d}$ and of the simple separating automaton $\mathcal{S}_{n,d}$ from Section 5.1.*

Theorem 5. *For every $(n, d/2)$-universal tree, the automaton $\mathcal{S}_{n,d}$ is a safety (n,d)-separator of* $\text{PosCyclEven}_{n,d}$ *and* $\text{LimsupOdd}_{n,d}$.

Proof sketch. In order to prove that $\mathcal{U}_{n,d}$ is indeed an (n,d)-separator it suffices to establish the following:

- every strategy subgraph of a positional winning strategy for Even (with at most n vertices and priorities up to d) has a progress measure that maps vertices into the set of leaves $L_{n,d/2}$ in the $(n, d/2)$-universal tree;
- for every such progress measure μ, for every word encoding a play in the strategy subgraph, and for every position in the word containing a letter (v,p), the state that labels the position in the unique run of automaton $\mathcal{U}_{n,d}$ on the word is a leaf that is (in the lexicographic tree order) larger than or equal to $\mu(v)$;
- for every infinite word in $\text{LimsupOdd}_{n,d}$, the unique run of $\mathcal{U}_{n,d}$ on the word reaches the rejecting state.

The first item follows from Theorem 4 and Proposition 3. The second item can be proved by an easy induction on the position number in the infinite word, and the third item can be proved by modestly generalizing the corresponding proof for the simple automaton $\mathcal{C}_{n,d}$ that we have given in Section 5.1. □

5.4 Separating safety automata from play summaries

Bojańczyk and Czerwiński [3, Chapter 3] give a pedagogically well-polished explanation of how the main technical result of Calude et al. [5] can be viewed as a construction of a deterministic automaton of quasi-polynomial size separating $\text{AllCyclEven}_{n,d}$ from $\text{AllCyclOdd}_{n,d}$. We recommend reading it since it is the most transparent exposition of the breakthrough result of Calude et al. that we are aware of. Although they state that the automaton accepts all words from $\mathrm{AllCyclEven}_{n,d}$, they actually prove that it accepts all words from $\mathrm{LimsupEven}_{n,d}$, and hence it separates $\mathrm{LimsupEven}_{n,d}$ from $\mathrm{AllCyclOdd}_{n,d}$.

One superficial difference between our exposition and theirs is that we use the model of safety automata, while they consider the dual model of *reachability automata* instead. (In reachability automata, an infinite word is accepted if and only if one of the designated *accepting states* is reached; otherwise it is rejected.) If, in Bojańczyk and Czerwiński's construction, we swap the roles of players Even and Odd, and we make the accepting states rejecting, we get a safety automaton that separates $\mathrm{AllCyclEven}_{n,d}$ (and hence also its subset $\mathrm{PosCyclEven}_{n,d}$) from $\mathrm{LimsupOdd}_{n,d}$.

Theorem 6 (Calude et al. [5], Bojańczyk and Czerwiński [3])**.** *The play summaries data structure of Calude et al. yields safety (n, d)-separators of* $\mathrm{AllCyclEven}_{n,d}$ *and* $\mathrm{LimsupOdd}_{n,d}$ *that are of quasi-polynomial size.*

5.5 Non-deterministic automata and the separation approach

The possible usage of non-deterministic automata in the separation approach to solving parity games is less straightforward. First of all, the game dynamics needs to be modified to explicitly include the choices that resolve non-determinism in every step. We give the power to make those choices to Even, but this extra power does not suffice to make her win the synchronized-product game whenever she has a winning strategy in the original parity game. The reason for this failure in transferring winning strategies from the parity game to the synchronized-product safety game is that in arbitrary non-deterministic automata it may be impossible to successfully resolve non-deterministic choices at a position in the input word without knowing the letters at the later positions. In the game, however, the resulting play depends on the future choices of the opponent, which the player cannot predict.

A well-known example of non-deterministic automata for which the synchronized-product game is equivalent to the original game is the class of *good-for-games* automata [21]. They have exactly the desired property that the non-deterministic choices of the automaton can always be resolved based only on the letters in the word at the positions up to the current one, thus making it possible to continue constructing an accepting run for all words accepted by the non-deterministic automaton that have the word read so far as a prefix.

Our analysis of Lehtinen's techniques [27] suggests that the good-for-games property may not be possible to achieve in the context of her work. However, we argue that we can still achieve the desired strategy transfer using a weaker property: given a non-deterministic automaton $\mathcal{A}$ that separates $\mathrm{PosCyclEven}_{n,d}$ and $\mathrm{LimsupOdd}_{n,d}$—there is a way to resolve non-deterministic choices based only on the prefix read so far, in such a way that a construction of an accepting run of $\mathcal{A}$ can be continued for all words in $\mathrm{PosCyclEven}_{n,d}$ (but not necessarily for all words accepted by $\mathcal{A}$).

Note that our lower bound in Section 4 holds for all non-deterministic automata separating $\mathrm{PosCyclEven}_{n,d}$ from $\mathrm{LimsupOdd}_{n,d}$, regardless of their suitability for strategy transfer, and hence for solving parity games using the separation approach.

5.6 Separating safety automata from register games

In her very recent work, Lehtinen [27] has given yet another technique for designing a quasi-polynomial algorithm for solving parity games. We argue here that her register-games technique can also be seen to fit the separation approach.

For every parity game G, Lehtinen defines the corresponding *register game* R_G, whose vertices consist of vertices of game G together with $\lfloor 1 + \lg n \rfloor$-sequences $\langle r_{\lfloor 1+\lg n \rfloor}, \ldots, r_2, r_1 \rangle$ of the so-called *registers* that hold priorities, i.e., numbers from the set $\{1, 2, \ldots, d\}$. The game is played on a copy of G in the usual way, additionally at her every move player Even is given a chance—but not an obligation—to "reset" one of the registers, and each register always holds the biggest priority that has occurred since it was last reset.

What needs explaining is what "resetting a register" entails. When the register at position k is reset, then the next register sequence is $\langle r_{\lfloor 1+\lg n \rfloor}, \ldots, r_{k+1}, r_{k-1}, \ldots, r_1, 1\rangle$, that is registers at positions 1 to $k-1$ are promoted to positions 2 to k, respectively, and the just-reset register is now at position 1 and it has value 1. Moreover—and very importantly for Lehtinen's construction—resetting the register at position k causes the even priority $2k$ to occur in game R_G if the value in the register was even, and the odd priority $2k+1$ otherwise. If, instead, Even decides not to reset any register then the odd priority 1 occurs.

Lehtinen's main technical result is that the original parity game G and the register game R_G have the same winners. She proves it by arguing that if Even has a (positional) winning strategy in G then she has a strategy of resetting registers in R_G so that she again wins the parity condition (albeit with the number of priorities reduced from an arbitrarily large d in G to only $\lfloor 1 + \lg n \rfloor$ in R_G). Our approach is to separate the graph structure from Lehtinen's mechanism to capture the original parity winning condition using registers. We now define an automata-theoretic analogue of her construction in which we use non-determinism to model the ability to pick various resetting strategies.

For all positive numbers n and d, such that d is even, we define the *non-deterministic parity automaton* $\mathcal{R}_{n,d}$ in the following way.

- The set of states of $\mathcal{R}_{n,d}$ is the set of non-increasing $\lfloor 1+\lg n \rfloor$-sequences $\langle r_{\lfloor 1+\lg n \rfloor}, \ldots, r_2, r_1\rangle$ of "registers" that hold numbers in $\{1, 2, \ldots, d\}$. The initial state is $\langle 1, 1, \ldots, 1\rangle$.
- For every state $s = \langle r_{\lfloor 1+\lg n \rfloor}, \ldots, r_2, r_1\rangle$ and letter $a = (v, p) \in \Sigma_{n,d}$, we define the *update of s by a* to be the state $\langle r_{\lfloor 1+\lg n \rfloor}, \ldots, r_k, p, \ldots, p\rangle$, where k is the smallest such that $r_k > p$.
- For every state $s = \langle r_{\lfloor 1+\lg n \rfloor}, \ldots, r_2, r_1\rangle$ and for every k, $1 \leq k \leq \lfloor 1 + \lg n \rfloor$, we define the *$k$-reset of s* to be the state $\langle r_{\lfloor 1+\lg n \rfloor}, \ldots, r_{k+1}, r_{k-1}, \ldots, r_2, 1\rangle$.
- Non-deterministic transitions in *parity* automata are quadruples (s, a, p, s'): upon reading a letter a in state s, the automaton can perform a transition of *priority* p and move to state s'.
- For every state s and letter $a = (v, p) \in \Sigma_{n,d}$, there is a transition $(s, a, 1, s')$ in the transition relation, where s' is the update of s by a; we call this a *non-reset* transition.
- For every state s, letter $a = (v, p) \in \Sigma_{n,d}$, and for every k, $1 \leq k \leq \lfloor 1 + \lg n \rfloor$, if $s'' = \langle r_{\lfloor 1+\lg n \rfloor}, \ldots, r_2, r_1\rangle$ is the update of s by a and r_k is even, then there is a transition $(s, a, 2k, s')$ in the transition relation, where s' is the k-reset of s''. We say that this transition is an *even reset of register k*.
- For every state s, letter $a = (v, p) \in \Sigma_{n,d}$, and for every k, $1 \leq k \leq \lfloor 1 + \lg n \rfloor$, if $s'' = \langle r_{\lfloor 1+\lg n \rfloor}, \ldots, r_2, r_1\rangle$ is the update of s by a and r_k is odd, then there is a transition $(s, a, 2k + 1, s')$ in the transition relation, where s' is the k-reset of s''. We say that this transition is an *odd reset of register k*.

Note that a non-deterministic parity automaton accepts an infinite word if and only if there is an infinite run of the automaton on the word in which the largest priority that occurs infinitely many times is even. The number of states in the automaton $\mathcal{R}_{n,d}$ is $d^{\lfloor 1+\lg n \rfloor}$. One can observe that the Lehtinen's register game R_G constructed basing on a parity game G is essentially the same as the synchronized product $G \times \mathcal{R}_{n,d}$.

Since our quasi-polynomial lower bounds apply to *safety* automata, but not to the more powerful parity automata, we now show how the non-deterministic parity automaton directly inspired by Lehtinen's construction of register games can be turned into a non-deterministic safety automaton that meets the requirements for the separation approach to work. To this end, it is enough to take a "synchronized product" of $\mathcal{R}_{n,d}$ with any automaton separating $\text{PosCyclEven}_{n',d'}$ from $\text{LimsupOdd}_{n',d'}$, where $n' = n \cdot d^{\lfloor 1+\lg n \rfloor}$ (the number of vertices in a product $G \times \mathcal{R}_{n,d}$) and $d' = 2\lfloor \lg n \rfloor + 4$ (an even upper bound for priorities emitted by $\mathcal{R}_{n,d}$), for example with the automaton $\mathcal{U}_{n',d'}$ from Section 5.3. This product $\mathcal{S}_{n,d} = \mathcal{R}_{n,d} \times \mathcal{U}_{n',d'}$ is constructed as follows:

- We assume that $\mathcal{U}_{n',d'}$ reads letters of the form $((v, s), p)$, where $v \in \{1, \dots, n\}$, and s is a state of $\mathcal{R}_{n,d}$, and $p \in \{1, \dots, d'\}$ (i.e., we identify elements of $\{1, \dots, n'\}$ with pairs (v, s) of the above form).
- The set of states of $\mathcal{S}_{n,d}$ is the set of pairs of states from $\mathcal{R}_{n,d}$ and $\mathcal{U}_{n',d'}$, respectively.
- The initial state is the pair of initial states of the two automata.
- A state in the product is rejecting, if its second coordinate is rejecting.
- For each non-reset transition $(s, a, 1, s')$ in $\mathcal{R}_{n,d}$, we have a transition $\big((s, c), a, (s', c')\big)$ in $\mathcal{S}_{n,d}$, where $c' = \langle c_{\lfloor 1+\lg n \rfloor}, \dots, c_1, c_0 - 1\rangle$, if $c_0 > 0$.
- For each letter $a = (v, p) \in \Sigma_{n,d}$, each transition (s, a, k, s') of $\mathcal{R}_{n,d}$, and each transition $\delta(q, ((v, s), k)) = q'$ of $\mathcal{U}_{n',d'}$, we have a transition $\big((s, q), a, (s', q')\big)$ in $\mathcal{S}_{n,d}$.

Lemma 4. *The safety automaton $\mathcal{S}_{n,d}$ separates* $\mathrm{PosCyclEven}_{n,d}$ *from* $\mathrm{LimsupOdd}_{n,d}$.

Proof. Consider first a word $w \in \mathrm{PosCyclEven}_{n,d}$; we have to prove that it is accepted by $\mathcal{S}_{n,d}$. By definition, w is the encoding of a play that arises from a positional winning strategy for Even in some game graph G with n vertices and priorities up to d. It is shown in Lehtinen [27] that if Even has a (positional) winning strategy in G, then she also has a winning strategy in R_G, that is, in the synchronized product $G \times \mathcal{R}_{n,d}$. Moreover, by inspecting the Lehtinen's proof we notice that the constructed winning strategy in $G \times \mathcal{R}_{n,d}$ is of a special form: on the first coordinate (i.e., in the game G) Even simply follows her positional winning strategy from G. This implies that w is a projection of a play w' in $G \times \mathcal{R}_{n,d}$ that is winning for Even (to obtain w' from w, we follow the choices of Odd from w and the Even's winning strategy in $G \times \mathcal{R}_{n,d}$). By definition of the synchronized product, this means that $\mathcal{R}_{n,d}$ accepts w.

Next, we observe that $w' \in \mathrm{PosCyclEven}_{n',d'}$; this is because Even has a (positional) winning strategy in $G \times \mathcal{R}_{n,d}$, and w' follows this strategy. When the product $\mathcal{S}_{n,d} = \mathcal{R}_{n,d} \times \mathcal{U}_{n',d'}$ reads w, then the $\mathcal{U}_{n',d'}$ component (for an appropriate way of resolving non-deterministic choices in the $\mathcal{R}_{n,d}$ component) reads w', and hence accepts it (because $\mathcal{U}_{n',d'}$ accepts all words from $\mathrm{PosCyclEven}_{n',d'}$).

For the opposite direction consider a word $w \in \mathrm{LimsupOdd}_{n,d}$. Let us first observe that $\mathcal{R}_{n,d}$ rejects this word. Consider thus a run ρ of $\mathcal{R}_{n,d}$ on w. If from some moment, in this run, there are no resets, then indeed $\mathcal{R}_{n,d}$ rejects. Otherwise, consider the maximal (odd) priority p occuring in w infinitely often, and consider the maximal number k such that there are infinitely many resets of register k in ρ. From some moment on, in ρ no priority higher than p is read, and there is no reset of any register $l > k$. A little bit later, there is a reset of register k; from this moment on, the value of register k is at most p. Moreover, infinitely many times the priority p is read, it is stored to register k, never overwritten by anything larger, and then reset. This means that there are infinitely many odd resets of register k (recall that p is odd), and no resets (except in the finite prefix that we have skipped) of registers $l > k$. In effect, ρ is rejecting.

Now consider a word w', being a result of composing w with some run ρ of $\mathcal{R}_{n,d}$ on w; this word is given as input to the $\mathcal{U}_{n',d'}$ component of the product automaton $\mathcal{S}_{n,d}$. Because ρ is rejecting, we have $w' \in \mathrm{LimsupOdd}_{n',d'}$. By assumption $\mathcal{U}_{n',d'}$ rejects w', thus $\mathcal{S}_{n,d}$ rejects w. ☐

Acknowledgements

This research has been supported by the EPSRC grant EP/P020992/1 (Solving Parity Games in Theory and Practice). W. Czerwiński and P. Parys are partially supported by the Polish National Science Centre grant 2016/21/D/ST6/01376. N. Fijalkow is supported by The Alan

Turing Institute under the EPSRC grant EP/N510129/1 and the DeLTA project (ANR-16-CE40-0007).

References

[1] J. Bernet, D. Janin, and I. Walukiewicz. Permissive strategies: from parity games to safety games. *Informatique Théorique et Applications*, 36(3):261–275, 2002.

[2] H. Björklund and S. G. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Applied Mathematics*, 155(2):210–229, 2007.

[3] M. Bojańczyk and W. Czerwiński. An automata toolbox, February 2018. `https://www.mimuw.edu.pl/~bojan/papers/toolbox-reduced-feb6.pdf`.

[4] A. Browne, E. M. Clarke, S. Jha, D. E. Long, and W. Marrero. An improved algorithm for the evaluation of fixpoint expressions. *Theor. Comput. Sci.*, 178(1–2):237–255, 1997.

[5] C. S. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan. Deciding parity games in quasipolynomial time. In *STOC*, pages 252–263, 2017.

[6] K. Chatterjee, M. Henzinger, and V. Loitzenbauer. Improved algorithms for parity and Streett objectives. *Logical Methods in Computer Science*, 13(3), 2017.

[7] A. Condon. The complexity of stochastic games. *Inf. Comput.*, 96(2):203–224, 1992.

[8] C. Daskalakis and Ch. Papadimitriou. Continuous local search. In *SODA*, pages 790–804, 2011.

[9] C. Daskalakis, Ch. Tzamos, and M. Zampetakis. A converse to Banach's fixed point theorem and its CLS completeness. In *STOC*, pages 44–50, 2018.

[10] L. Daviaud, M. Jurdziński, and M. Lazić. A pseudo-quasi-polynomial algorithm for solving mean-payoff parity games. In *LICS*, pages 325–334, 2018.

[11] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *FOCS*, pages 368–377, 1991.

[12] E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model-checking for fragments of μ-calculus. In *CAV*, pages 385–396, 1993.

[13] E. A. Emerson and Ch.-L. Lei. Efficient model checking in fragments of the propositional mu-calculus (Extended abstract). In *LICS*, pages 267–278, 1986.

[14] J. Fearnley. Exponential lower bounds for policy iteration. In *ICALP*, pages 551–562, 2010.

[15] J. Fearnley, S. Jain, S. Schewe, F. Stephan, and D. Wojtczak. An ordered approach to solving parity games in quasi polynomial time and quasi linear space. In *SPIN*, pages 112–121, 2017.

[16] N. Fijalkow. An optimal value iteration algorithm for parity games. *CoRR*, abs/1801.09618, 2018. `https://arxiv.org/abs/1801.09618`.

[17] O. Friedmann. An exponential lower bound for the parity game strategy improvement algorithm as we know it. In *LICS*, pages 145–156, 2009.

[18] O. Friedmann, T. D. Hansen, and U. Zwick. Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In *STOC*, pages 283–292, 2011.

[19] H. Gimbert and R. Ibsen-Jensen. A short proof of correctness of the quasi-polynomial time algorithm for parity games. *CoRR*, abs/1702.01953, 2017. `https://arxiv.org/abs/1702.01953`.

[20] M. Goldberg and E. Lifshitz. On minimal universal trees. *Matematicheskie Zametki*, 4(3):371–380, 1968. (In Russian).

[21] T. A. Henzinger and N. Piterman. Solving games without determinization. In *CSL*, pages 395–410, 2006.

[22] M. Jurdziński. Deciding the winner in parity games is in UP $\cap$ co-UP. *Inf. Process. Lett.*, 68(3):119–124, 1998.

[23] M. Jurdziński. Small progress measures for solving parity games. In *STACS*, pages 290–301, 2000.

[24] M. Jurdziński and R. Lazić. Succinct progress measures for solving parity games. In *LICS*, pages 1–9, 2017.

[25] M. Jurdziński and R. Lazić. Succinct progress measures for solving parity games. *CoRR*, abs/1702.05051, 2018. `https://arxiv.org/abs/1702.05051v3`.

[26] M. Jurdziński, M. Paterson, and U. Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM J. Comp.*, 38(4):1519–1532, 2008.

[27] K. Lehtinen. A modal μ perspective on solving parity games in quasi-polynomial time. In *LICS*, pages 639–648, 2018.

[28] A. W. Mostowski. Games with forbidden positions. Technical Report 78, Uniwersytet Gdański, 1991.

[29] M. O. Rabin. *Automata on Infinite Objects and Church's Problem.* American Mathematical Society, 1972.

[30] S. Schewe. Solving parity games in big steps. *J. Comput. Syst. Sci.*, 84:243–262, 2017.

[31] H. Seidl. Fast and simple nested fixpoints. *Inf. Process. Lett.*, 59(6):303–308, 1996.

[32] J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *CAV*, pages 202–215, 2000.

[33] U. Zwick and M. Paterson. The complexity of mean-payoff games on graphs. *Theor. Comput. Sci.*, 158:343–359, 1996.