Deciding Parity Games in Quasipolynomial Time*

Cristian S. Calude[1], Sanjay Jain[2], Bakhadyr Khoussainov[1], Wei Li[3] and Frank Stephan[2,3]

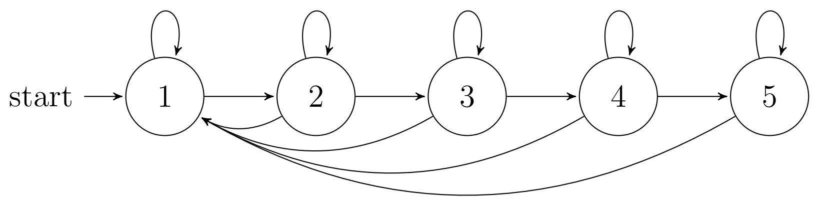
[1] Department of Computer Science, University of Auckland
Private Bag 92019, Auckland, New Zealand
{cristian,bmk}@cs.auckland.ac.nz
[2] Department of Computer Science, National University of Singapore
13 Computing Drive, COM1, Singapore 117417, Republic of Singapore
{sanjay,fstephan}@comp.nus.edu.sg
[3] Department of Mathematics, National University of Singapore
10 Lower Kent Ridge Road, S17, Singapore 119076, Republic of Singapore
liwei.sg@gmail.com

Abstract. It is shown that the parity game can be solved in quasipolynomial time. The parameterised parity game – with n nodes and m distinct values (aka colours or priorities) – is proven to be in the class of fixed parameter tractable (**FPT**) problems when parameterised over m. Both results improve known bounds, from runtime $n^{O(\sqrt{n})}$ to $O(n^{\log(m)+6})$ and from an **XP**-algorithm with runtime $O(n^{\Theta(m)})$ for fixed parameter m to an **FPT**-algorithm with runtime $O(n^5 + 2^{m\log(m)+6m})$. As an application it is proven that coloured Muller games with n nodes and m colours can be decided in time $O((m^m \cdot n)^5)$; it is also shown that this bound cannot be improved to $2^{o(m \cdot \log(m))} \cdot n^{O(1)}$ in the case that the Exponential Time Hypothesis is true. Further investigations deal with memoryless Muller games and multi-dimensional parity games.

1 Introduction

A parity game is given by a directed graph (V, E), a starting node $s \in V$, a function val which attaches to each $v \in V$ an integer value (also called colour) from a set $\{1, 2, 3, \ldots, m\}$; the main parameter of the game is n, the number of nodes, and the second parameter is m. Two players Anke and Boris move alternately in the graph with Anke moving first. A move from a node v to another node w is valid if (v, w) is an edge in the graph; furthermore, it is required that from every node one can make at least one valid move. The alternate moves by Anke and Boris and Anke and Boris and ... define an infinite sequence of nodes which is called a play. For the evaluation, it is defined that each value is owned by one player; without loss of generality one player owns the odd numbers and the other player owns the even numbers. Anke wins a play through nodes $a_0, a_1, a_2, \ldots$ iff the limit superior (that is, the largest value appearing infinitely often) of the sequence $val(a_0), val(a_1), val(a_2), \ldots$ is a number she owns, that is, a number of her parity. An example is the following game.

* S. Jain was supported in part by NUS grant C252-000-087-001; B. Khoussainov was supported in part by the Marsden Fund grant of the Royal Society of New Zealand; furthermore, S. Jain, B. Khoussainov and F. Stephan have been supported in part by the Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2016-T2-1-019 / R146-000-234-112. A conference version of this work was presented at the Symposium on the Theory of Computing, STOC 2017 [11].



Here the nodes are labeled with their values, which are unique (but this is not obligatory); furthermore, Anke has even and Boris has odd parity. Boris has now the following memoryless (that is, moves are independent of the history) winning strategy for this game: $1 \to 1$, $2 \to 3$, $3 \to 3$, $4 \to 5$, $5 \to 5$. Whenever the play leaves node 1 and Anke moves to node 2, then Boris will move to node 3. In case Anke moves to node 4, Boris will move to node 5. Hence, whenever the play is in a node with even value (this only happens after Anke moved it there), in the next step the play will go into a node with a higher odd value. So the largest infinitely often visited node value is odd and therefore the limit superior of these numbers is an odd number which justifies Boris' win. Hence Boris has a winning strategy for the parity game given above.

Please see the next section for a more formal definition of the games and complexity classes discussed in this introduction.

It is known that for parity games, in general, the winner can always use a *memoryless winning strategy* [6,29,30,32,61,62,80]; see Corollary 21 below. This fact will be one central point in the results obtained in this paper: the parity game will be augmented with a special statistics – using polylogarithmic space – which indicates the winner correctly after a finite time whenever the winner employs a memoryless winning strategy. By the way, the existence of memoryless winning-strategies are also a convenient tool to prove that solving parity games is in $\mathbf{NP} \cap \mathbf{coNP}$ — fixing a memoryless strategy for one player transforms the parity game into a one-player game with a parity objective and one can check in polynomial time whether this game can be won.

Parity games are a natural class of games which are not only interesting in their own right, but which are also connected to fundamental notions like μ-calculus, modal logics, tree automata and Muller games [4,7,8,18,30,32,48,71,75,76,78,79]. Faster algorithms for solving parity games could be used to improve the algorithms deciding the theory of certain tree automatic structures [35,36,58] and to employ them to understand better these structures.

For investigating the complexity side of the game, it is assumed that the game is given by a description in size polynomial in the number n of nodes and that one can evaluate all relevant parts of the description in logarithmic space. A possibility is to store the following three items for each game (where Anke moves first and starts from node 1):

- two numbers m, n with $1 \leq m \leq n$ and one bit which says whether the values owned by player Anke are the even or the odd numbers;
- the game graph given by a table, that is, for each pair of nodes, a bit which says whether there is a directed edge between the two nodes (which can be same) or not;
- the values of the nodes given by another table which holds, for each node, a binary number from $\{1, 2, 3, \ldots, m\}$.

An important open problem for parity games is the *time complexity* for finding the winner of a parity game, when both players play optimally; the first algorithms took exponential time [61,80] and subsequent studies searched for better algorithms [51,53,55,64,70,71,72]. Many researchers, including Emerson and Jutla [32] in 1991, asked whether the winner of a parity game can be determined in polynomial time.

Emerson, Jutla and Sistla [33] showed that the problem is in **NP** $\cap$ **coNP** and Jurdzinski [52] improved this bound to **UP** $\cap$ **coUP**. This indicates that the problem is not likely to be hard for **NP** and might be solvable faster than in exponential time. Indeed, Petersson and Vorobyov [64] devised a subexponential randomised algorithm and Jurdzinski, Paterson and Zwick [55] a deterministic algorithm of similar complexity (more precisely, the subexponential complexity was approximately $n^{O(\sqrt{n})}$).

Besides this main result, there are also various practical approaches to solve special cases [4,26,41] or to test out and analyse heuristics [12,44,53]; however, when Friedmann and Lange [39] compared the various parity solving algorithms from the practical side, they found that Zielonka's recursive algorithm [80] was still the most useful one in practice.

McNaughton [61] showed that the winner of a parity game can be determined in time $n^{m+O(1)}$ and this was subsequently improved to $n^{m/2+O(1)}$ [9,73] and to $n^{m/3+O(1)}$ [70,72], where n is the number of nodes and m is the maximum value of the nodes.

The consideration of the parameter m is quite important for analysing the algorithmic complexity of solving parity games; it is furthermore also a very natural choice. Schewe [71,72] argues that for many applications which are solved using parity games, the parameter m is much smaller than n, often by an exponential gap.

For example, when translating coloured Muller games into parity games in the way done by McNaughton [61] and Björklund, Sandberg and Vorobyov [5], the number of values is, for all but finitely many games, bounded by the logarithm of the number of nodes, see the proof of Theorem 23 below. A similar result holds for the translation of multi-dimensional parity games into standard parity games.

A further important application of parity games is the area of reactive synthesis. Here one translates LTL-formulas into a Büchi automaton which needs to be determinised by translating it into a parity automaton. Building on work of Safra [68,69], Piterman [65] showed that one can translate non-deterministic Büchi automata with n states into parity automata with $2 \cdot n^n \cdot n!$ states and $2n$ values. In other words, one can evaluate various conditions on these parity automata by determining the winner in the corresponding parity game. Also Di Stasio, Murano, Perelli and Vardi [25] investigated in their experiments various scenarios where the number m is logarithmic in n.

The present work takes therefore the parameter m into consideration and improves the time bounds in two ways:

- The overall time complexity is $O(n^{\lceil \log(m) \rceil + 6})$ which provides a quasipolynomial bound on the runtime, as one can always choose $m \leq n$;
- Furthermore, if $m < \log(n)$, then the overall time complexity is $O(n^5)$, which shows that the problem is fixed parameter tractable when parameterised by m; the parity games are therefore in the lowest time complexity class usually considered in parameterised complexity.

Prior investigations have already established that various other parameterisations of parity games are fixed-parameter tractable, but the parameterisation by m was left open until now. Chatterjee

[14] pointed out to the authors that one can also write the result in a product form with parity games being solvable in time $O(2^m \cdot n^4)$ for all m, n; the proof uses just the methods of Theorem 16 but keeping m as a parameter and not using explicitly the bound of $m \leq \log(n)$ which, when invoked into above formula, would give the bound $O(n^5)$.

An application of the results presented here is that coloured Muller games with n nodes and m colours can be decided in time $O((m^m \cdot n)^5)$; Theorem 25 below shows that this bound cannot be improved to $2^{o(m \cdot \log(m))} \cdot n^{O(1)}$ provided that the Exponential Time Hypothesis is true.

Subsequent research [34,42,54,74] has provided the additional runtime bound

$$O(\lceil m/\log(n)\rceil^4 \cdot n^{3.45+\log(\lceil m/\log(n)\rceil+2)})$$

where the bound cited here stems from Stephan's teaching material [74, Theorem 20.22] while the research papers [34,42,54] obtained slightly better bounds due to some assumptions they make on the game and due to the usage of better bounds for binomials. However, the main contribution of the subsequent research [34,54] is that the quasipolynomial time algorithm can be modified such that, in addition to the time bound, the workspace the algorithm uses is only quasilinear in the number of nodes n. This improves over the algorithm presented here which uses quasipolynomial space. Furthermore, various authors provided their own version of the verification of the algorithm presented in this paper [34,42,54]. Before the presentation of the results, the following section summarise the basic definitions and properties of the games and also provide the basic complexity classes needed.

2 Basic Notions Used

This section summarises the basic properties of the two games (parity game and coloured Muller game) and also explains related games (multi-dimensional parity game, Rabin game and Streett game). It furthermore provides the basic complexity-theoretic notions used in this paper.

Definition 1. A *game* is given by a directed finite graph of n nodes, a starting node and a set G of sets of nodes which are called the *winning set* of player Anke. The two players, Anke and Boris, move alternately a marker through the graph, where Anke starts from the starting node and the players each time move along an outgoing edge of the current node; here it is required that every node has at least one outgoing edge (which can go to the node itself). A *play* is the infinite sequence of nodes visited by the marker while Anke and Boris are playing. To decide the *winner of a play*, one considers the set of infinitely often visited nodes U. Now Anke wins the play iff $U \in G$.

In a *parity game*, each node v carries a value, denoted $val(v)$. In a *Muller game*, each node v carries a set of colours.

In a *parity game*, the set G can be derived from values from 1 to m (where $m \leq n$) which are associated with the nodes. For this, one associates with each player Anke and Boris a parity and a set U is in G iff the maximum value of U is of Anke's parity. Alternatively one can require that G respects the parity, that is, if U and U' satisfy that the the maximum values of nodes in U and in U', respectively, have the same parity then either U, U' are both inside G or U, U' are both outside G.

In a *coloured Muller game*, every node is associated with a set of colours. For a set U, $colour(U)$ is the set of all colours which are associated with at least one node in U. The set G has to respect the colours, that is, if $colour(U) = colour(U')$ then either both U and U' are inside G or both U and U' are outside G.

In a *k-dimensional parity game*, each node is associated with a k-dimensional vector of values. Now a set U of nodes is winning for player Anke iff the component wise maximum of the value-vectors of the nodes in U is a vector of k even numbers.

Rabin games and *Streett games* have as additional information a list $(V_1, W_1), (V_2, W_2), (V_3, W_3), \ldots, (V_m, W_m)$ of pairs such that in the Rabin case, a set of nodes is in U iff some pair (V_h, W_h) satisfies $V_h \cap U \neq \emptyset$ and $W_h \cap U = \emptyset$; in the Streett case, $U \in G$ iff all pairs (V_h, W_h) satisfy $V_h \cap U \neq \emptyset \Rightarrow W_h \cap U \neq \emptyset$.

A *strategy for a player*, say for Anke, maps, for every situation where Anke has to move, the current node and history of previous moves to a suggested move for Anke. A *winning strategy* for Anke is a strategy for Anke which guarantees that Anke wins a play whenever she follows the suggested moves. A strategy is called *memoryless* iff it only depends on the current node and not on any other aspects of the history of the play.

The *winner of a game* is that player who has a winning strategy for this game.

Remark 2. All games considered in this paper (including parity games and coloured Muller games) have always a winner; this winner wins every play in the case that the winner follows a winning strategy.

The additional structures of parity games, coloured Muller games and other games enforce that the winning set G is of a certain form; in particular in the case that the parameter m (number of colours of a coloured Muller game or number of values of a parity game) is small compared to n, the algorithms to solve these games have a better time bound than in the general case.

As choosing for each node a unique colour not shared with any other node does not impose any restriction on G, one can without loss of generality require that $m \leq n$.

For parity games, if a value $k > 1$ does not occur in a game, but $k + 1$ does, then one can for all nodes v with $val(v) > k$ replace $val(v)$ by $val(v) - 2$ without changing the winner of the game. Furthermore, if the value 1 does not occur in the game, then one can replace $val(v)$ by $val(v) - 1$ throughout the game and invert the parity of the players. For that reason, the maximum value m of a parity game can always be assumed to satisfy $m \leq n$.

In coloured Muller games, representations of G as tables might have the size 2^m and one has several choices on how to handle this situation: (a) one only considers such coloured Muller games where G can be decided by a Boolean circuit not larger than $p(n)$ size for some polynomial p; (b) the same as (a) with a polynomial time algorithm instead with program size $p(n)$; (c) one uses the space needed for representing G as a Boolean circuit as an additional parameter for the game. The approach taken in the present paper is (b).

Remark 3. It should be pointed out that one can also consider games where it only depends on the node which player moves and the players do not necessarily take turns. Both versions of parity or Muller games can be translated into each other with a potential increase of the number of nodes by a factor 2.

In the case that one goes from turn-based to position-based Muller games, one doubles up each node: Instead of the node v one uses a node (Anke, v), when it is Anke's turn to move, and a node (Boris, v), when it is Boris' turn to move; the nodes (Anke, v) and (Boris, v) in the new game have the same values or colours as v in the old game. For every edge from v to w in the old game, one puts the edges from (Anke, v) to (Boris, w) and from (Boris, v) to (Anke, w) into the game.

For the other direction, each node w receives a prenode w' with exactly one outgoing edge from w' to w. Now, for each edge (v, w) from the original game, if the same player moves at v and at w in the original game, then one puts the edge (v, w') into the new game, else one puts the edge (v, w) into the new game. The rationale behind this is that the longer path $v - w' - w$ has even length in the case that the players moving at v and w should be the same for alternating moves. Furthermore, if Anke moves at the original starting node s, then s is also the starting node of the new game, else s' is the starting node of the new game. Again, the nodes w and w' in the new game have the same value or colour as the node w in the old game.

Parameterised Complexity studies the complexity to solve a problem not only in dependence of the main parameter n (size of input), but also other related parameters $m, k, \ldots$ which are expected to arise naturally from the problem description. In the following, let n denote the main parameter and m a natural further parameter.

Definition 4. A problem is called fixed parameter tractable (**FPT**) iff there is a polynomial p and a further function f such that all instances of the problem can be solved in time $f(m)+p(n)$.

The class of all problems in **FPT** can also be characterised as those problems which can be solved in $g(m) \cdot p(n)$ for some polynomial p and an arbitrary function g.

For the current work, the main parameter n is the number of nodes and the parameter m is the number of values in the parity game or the number of colours in the coloured Muller game. The so chosen second parameter m is a very natural parameter to the games considered and occurs widely in prior work studying the complexity of the games [5,9,61,70,72,73]. However, in the literature also other parameters and parameter combinations have been studied.

The number m of colours used in the game is an important parameter of coloured Muller games; for complexity-theoretic considerations, the exact complexity class of solving coloured Muller games with n nodes and m colours may also depend on how G is represented, in particular in case when m is large. The size of this representation can thus be a further parameter for determining the complexity class of solving coloured Muller games. However, this parameter is not studied in the present work.

Definition 5. A problem is in the class **XP** if it can be solved in time $O(n^{f(m)})$ for some function f.

Between **FPT** at the bottom and **XP** at the top, there are the levels of the **W**-hierarchy $\mathbf{W[1]}$, $\mathbf{W[2]}$, $\mathbf{W[3]}$, $\ldots$; it is known that **FPT** is a proper subclass of **XP** and it is widely believed that the levels of the **W**-hierarchy are all different. The books of Downey and Fellows [27,28] and Flum and Grohe [37] give further information on parameterised complexity.

Definition 6. The Exponential Time Hypothesis says that for the usual satisfiability problems like **3SAT**, **4SAT** and **SAT** itself, for n being the number of variables, any algorithm solving them needs at least time c^n worst case complexity for some rational number $c > 1$ and almost all n.

The Exponential Time Hypothesis implies that $\mathbf{W[1]}$ differs from **FPT**, but not vice versa. Note that the **NP**-complete problems are spread out over all the levels of this hierarchy and that even the bottom level **FPT** also contains sets outside **NP**. The level of a problem can depend on the choice of the parameters to describe the problem, therefore one has to justify the choice of the parameters.

Chandra, Kozen and Stockmeyer [13] investigated alternating Turing machines. Such machines can be defined in an asymmetric and a symmetric way; the latter is in particular needed for lower complexity bounds in certain settings. Furthermore, Cook [23] and Levin [60] initiated the systematic study of **NP** and formalised the question whether $\mathbf{NP} = \mathbf{P}$.

Definition 7. Alternating Turing machines can be viewed as a game: Besides the usual Turing machine steps, there are also branching Turing machine steps. In the case of an existential branching, one player, say Anke, decides which of the possible steps the Turing machine is taking; in the case of a universal branching, the other player, here Boris, decides which of the possible steps the Turing machine is taking. Anke wins iff Anke can always force the game in an accepting state. Boris wins iff the game never goes into an accepting state. Now for every x as input, one of the players has a winning strategy; the alternating Turing machine decides L iff the following holds: For all $x \in L$, Anke has a winning strategy, for all $x \notin L$, Boris has a winning strategy.

A language L is in alternating time / space $f(n)$ iff for every $x \in L$, Anke can play such that x is accepted and the play does not violate the resource bound $f(n)$; for $x \notin L$, Boris can play such that x is never accepted and, in the case of a space resource bound, the play does not violate the resource bound.

A language L is in non-deterministic time / space $f(n)$ iff it is in an alternating time / space $f(n)$ via a Turing machine where Boris has always only one choice. A language is in **NP** iff there is a non-deterministic Turing machine and a polynomial p such that if $L(x) = a$ then Anke can play such that the input (x, a) is accepted within time $|p(x)|$ and if $L(x) \neq a$ then Anke cannot achieve that (x, a) gets accepted. A language is in **UP** iff it is in **NP** via a machine which has, for every pair $(x, L(x))$, exactly one computation path which Anke can choose such that $(x, L(x))$ gets accepted.

A language L satisfies $L \in \mathbf{\Sigma_2^P}$ iff there is an alternating Turing machine recognising L in polynomial time such that on every computation path, all the points where Anke can branch the computation come before those points where Boris can branch the computation.

In the case of alternating computation, for small complexity classes where one cannot check the complexity within the mechanism given, one employs for alternating computations a symmetric setting where the alternating Turing machine has explicit accepting and explicit rejecting states and it halts in both. Now L is in the given time class iff the following holds: For all $x \in L$, Anke has a winning strategy which guarantees that while obeying the given resource bound the game ends up in an accepting state; for all $x \notin L$, Boris has a winning strategy which guarantees that while obeying the given resource bound the game ends up in an rejecting state.

If the space-bound or the time-bound are constructible within the given complexity class, then the alternating computation for the standard model can also be equipped with a counter; then the machine can go to the rejecting state when the run-time is exhausted; here one uses that if an alternating machine using space $f(n)$ does not accept within $c^{f(n)}$ steps for a suitable constant c then one can safely reject the computation. The first approach to solving the parity games in polylogarithmic space below also has this symmetric approach implicitly, even without using explicit counters for the used up time.

3 The Complexity of the Parity Game

The main result in this section is an alternating polylogarithmic space algorithm to decide the winner in parity games; later more concrete bounds will be shown. The idea is to collect, in polylogarithmic space, for both players in the game, Anke and Boris, the statistics of their performance in the play. In particular, these statistics store information about whether the play has surely gone through a loop where the largest valued node has the parity of the corresponding player. Though these statistics do not capture all such loops, in case that one player plays a memoryless winning strategy, the player's own statistics will eventually find evidence for such a loop while the opponent statistics will not provide false evidence which would lead into the opposite direction.

The following notation will be used throughout the paper. In order to avoid problems with fractional numbers and $\log(0)$, let $\lceil \log(k) \rceil = \min\{h \in \mathbb{N} : 2^h \geq k\}$. Furthermore, a function (or sequence) f is called *increasing* whenever for all i, j the implication $i \leq j \Rightarrow f(i) \leq f(j)$ holds.

Theorem 8. *There exists an alternating polylogarithmic space algorithm deciding which player has a winning strategy in a given parity game. When the game has n nodes and the values of the nodes are in the set $\{1, 2, 3, \ldots, m\}$, then the algorithm runs in $O(\log(n) \cdot \log(m))$ alternating space.*

Proof. The idea of the proof is that, in each play of the parity game, one maintains winning statistics for both players Anke and Boris. These statistics are updated after every move for both players. In case a player plays according to a memoryless winning strategy for the parity game, the winning statistics of this player will eventually indicate the win (in this case one says that the "winning statistics of the player mature") while the opponent's winning statistics will never mature. This will be explained in more detail below.

The winning statistics of Anke (Boris) has the following goal: to track whether the play goes through a loop where the largest value of a node in the loop is of Anke's (Boris') parity. Note that if Anke follows a memoryless winning strategy then the play will eventually go through a loop and the node with the largest value occurring in any loop the play goes through is always a node of Anke's parity. Otherwise, Boris can repeat a loop with the largest value being of Boris' parity infinitely often and thus win, contradicting that Anke is using a memoryless winning strategy.

The naïve method to do the tracking is to archive the last $2n + 1$ nodes visited, to find two identical moves out of the same node by the same player and to check whose parity has the largest value between these two moves. This would determine the winner in case the winner uses a memoryless winning strategy. This tracking needs $O(n \cdot \log(n))$ space – too much space for the intended result. To save space one constructs a winning statistics which still leads to an Anke win in case Anke plays a memoryless winning strategy, but memorises only partial information.

The winning statistics of the players are used to track whether certain sequences of nodes have been visited in the play so far and the largest value of a node visited at the end or after the sequence is recorded. The definitions are similar for both players. For simplicity the definition is given here just for player Anke.

Definition 9. In Anke's winning statistics, an i-sequence is a sequence of nodes $a_1, a_2, , a_3, , \ldots,$ a_{2^i} which have been visited (not necessarily consecutively, but in order) during the play so far such that, for each $k \in \{1, 2, 3, \ldots, 2^i - 1\}$,

$$\max\{val(a) : a = a_k \vee a = a_{k+1} \vee a \text{ was visited between } a_k \text{ and } a_{k+1}\},$$

is of Anke's parity.

The aim of Anke is to find a sequence of length at least $2n+1$, as such a sequence must contain a loop. So she aims for a $(\lceil \log(n) \rceil + 2)$-sequence to occur in her winning statistics. Such a sequence is built by combining smaller sequences over time in the winning statistics.

Here a winning statistics $(b_0, b_1, \ldots, b_{\lceil \log(n) \rceil + 2})$ of a player consists of $\lceil \log(n) \rceil + 3$ numbers between 0 and m, both inclusive, where $b_i = 0$ indicates that currently no i-sequence is being tracked and $b_i > 0$ indicates that

> **Property-**b_i: an i-sequence is being tracked and that the largest value of a node visited at the end of or after this i-sequence is b_i.

Note that for each i at most one i-sequence is tracked. The value b_i is the only information of an i-sequence which is kept in the winning statistics.

The following invariants are kept throughout the play and are formulated for Anke's winning statistics; those for Boris' winning statistics are defined with the names of Anke and Boris interchanged. In the description below, "i-sequence" always refers to the i-sequence being tracked in the winning statistics.

- (I1) Only b_i with $0 \leq i \leq \lceil \log(n) \rceil + 2$ are considered and each such b_i is either zero or a value of a node which occurs in the play so far.
- (I2) An entry b_i refers to an i-sequence which occurred in the play so far iff $b_i > 0$.
- (I3) If b_i, b_j are both non-zero and $i < j$ then $b_i \leq b_j$.
- (I4) If b_i, b_j are both non-zero and $i < j$, then in the play of the game so far, the i-sequence starts only after a node with value b_j was visited at or after the end of the j-sequence.

When a play starts, the winning statistics for both players are initialised with $b_i = 0$ for all i. During the play when a player moves to a node with value b, the winning statistics of Anke is updated as follows – the same algorithm is used for Boris with the names of the players interchanged everywhere.

1. If b is of Anke's parity or $b > b_i > 0$ for some i, then one selects the largest i such that
   (a) either b_i is not of Anke's parity – that is, it is either 0 or of Boris' parity – but all b_j with $j < i$ and also b are of Anke's parity
   (b) or $0 < b_i < b$

   and then one updates $b_i = b$ and $b_j = 0$ for all $j < i$.
2. If this update produces a non-zero b_i for any i with $2^i > 2n$ then the play terminates with Anke being declared winner.

Note that it is possible that both 1.(a) and 1.(b) apply to the same largest i. In that case, it does not matter which case is chosen, as the updated winning statistics is the same for both cases. However, the tracked i-sequences referred to may be different; this does not effect the rest of the proof.

Example 10. Here is an example of i-sequences for player Anke. This example is only for illustrating how the i-sequences and b_i's work; in particular this example does not use memoryless strategy for either of the players. Consider a game where there is an edge from every node to every node (including itself) and the nodes are $\{1, 2, 3, \ldots, 7\}$ and have the same values as names; Anke has odd parity. Consider the following initial part of a play:

```
1 6 7 5 1 4 5 3 2 1 3 2 3 1 3 3 1 2 1
```

The i-sequences and the b_i's change over the course of above play as given in the following table. In the table, the nodes prefixed by "i :" are the nodes of the corresponding i-sequence.

Move	b_4, b_3, b_2, b_1, b_0	i-sequences in play so far	rule
1	0,0,0,0,1	0:1	1.(a)
6	0,0,0,0,6	0:1 6	1.(b)
7	0,0,0,0,7	1 6 0:7	1.(a)
5	0,0,0,5,0	1 6 1:7 1:5	1.(a)
1	0,0,0,5,1	1 6 1:7 1:5 0:1	1.(a)
4	0,0,0,5,4	1 6 1:7 1:5 0:1 4	1.(b)
5	0,0,0,5,5	1 6 1:7 1:5 1 4 0:5	1.(a)
3	0,0,3,0,0	1 6 2:7 2:5 1 4 2:5 2:3	1.(a)
2	0,0,3,0,0	1 6 2:7 2:5 1 4 2:5 2:3 2	
1	0,0,3,0,1	1 6 2:7 2:5 1 4 2:5 2:3 2 0:1	1.(a)
3	0,0,3,3,0	1 6 2:7 2:5 1 4 2:5 2:3 2 1:1 1:3	1.(a)
2	0,0,3,3,0	1 6 2:7 2:5 1 4 2:5 2:3 2 1:1 1:3 2	
3	0,0,3,3,3	1 6 2:7 2:5 1 4 2:5 2:3 2 1:1 1:3 2 0:3	1.(a)
1	0,1,0,0,0	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1	1.(a)
3	0,3,0,0,0	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3	1.(b)
3	0,3,0,0,3	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3 0:3	1.(a)
1	0,3,0,1,0	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3 1:3 1:1	1.(a)
2	0,3,0,2,0	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3 1:3 1:1 2	1.(b)
1	0,3,0,2,1	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3 1:3 1:1 2 0:1	1.(a)

If at an update of an i-sequence both possible updates 1.(a) and 1.(b) apply to the same level i then it does not matter for the statistics which is chosen. However, for the i-sequences, one has to commit to one choice and for simplicity, (for the above table) one assumes that 1.(a) has priority. So the formal algorithm for updating the sequences is the following one.

1. If b is of Anke's parity or $b > b_i > 0$ for some i, then one selects the largest i such that

(a) either b_i is not of Anke's parity – that is, it is either 0 or of Boris' parity – but all b_j with $j < i$ and also b are of Anke's parity

(b) or $0 < b_i < b$

else there is no update and one goes to step 3.

2. For the selected i, one does the following update according to the first of the two above cases which applies:

   (a) Let $b_i = b$.
   Let the new i-sequence contain all the nodes of the old j-sequences, with $j < i$, plus the new node with value b.
   Let $b_j = 0$ for all $j < i$ as the corresponding j-sequences are merged into the new i-sequence;

   (b) Let $b_i = b$ and let the i-sequence be unchanged except for the update of the associated value b_i and all j-sequences with $j < i$ are made void by setting $b_j = 0$ for all $j < i$.

   Furthermore, all j-sequences with $j > i$ are maintained as they are.

3. If this update produces a non-zero b_i for any i with $2^i > 2n$ then the play terminates with Anke being declared winner and no further tracking of i-sequences is needed.

The 3-sequence in the above table already has a loop, as there are three occurrences of "3 : 3" and the second and third of these have that the same player moves. However, as the sequences are not stored but only the b_i, Anke's winning statistics only surely indicates a win of player Anke when there is an $i \geq \log(2n + 1)$ with $b_i > 0$; this i is 4 as $2^4 > 2 \cdot 7 + 1$.

Remark 11. An anonymous referee suggested to make the following intuition more explicit before giving the verification of the algorithm: The winning statistics of both players are maintained via a deterministic algorithm which updates each statistics based on the prior value and the current node visited, more precisely, the value of the node visited. These statistics use only $\log(m) \cdot \log(n)$ bits of memory. If a player, during a play, follows a memoryless winning strategy then the player's winning statistics will eventually indicate a win while the opponent's winning statistics will never do. However, if neither of the players follow a memoryless winning strategy then no guarantees on the outcome of the evolution of the statistics are made. Furthermore, if one identifies "Anke's winning strategy indicates a win" with "accept" and "Boris' winning strategy indicate a win" with "reject" then one can view the game as a run of an alternating $O(\log(n) \cdot \log(m))$ space Turing machine which keeps in its memory only the statistics, the current node and the player to move and which explicitly accepts a computation in the case that Anke can win the game and explicitly rejects a computation in the case that Boris can win the game. For the case of checking whether Anke can win, the existential branchings are the choice of the next move by Anke and the universal branchings are the choice of the next move by Boris. The obtained characterisation is heavily based on the fact that in every parity game one of the players has a memoryless winning strategy, see Corollary 21 below. In the case that one maintains only Anke's winning statistics, provided that the winning player plays a memoryless winning strategy, then the alternating Turing machine would explicitly accept if Anke can win and will reject by "running forever" without ever visiting an accepting state in the case that Boris can win. The anonymous referee informed the authors that such an algorithm which maintains the winning statistics might be called a "space-efficient one-pass streaming algorithm inspecting the play".

Verification of the algorithm. Note that, in the updating algorithm for Anke's winning statistics, if b is of Anke's parity, then there is an i that satisfies 1.(a), as otherwise the algorithm would have terminated earlier. Initially, the invariants clearly hold as all b_i's are 0. Now it is shown that the invariants are preserved at updates of the b_i's according to cases 1.(a) or 1.(b).

It is easy to verify that the invariants are maintained if the update is due to 1.(b), and it also ensures that Property-b_i is maintained for the i-sequences being tracked. In case the update is done due to 1.(a), then the Property-$b_{i'}$ is maintained for all i'-sequences being tracked for $i' > i$ (with $b_{i'} \geq b$ in these cases). For $i' < i$, $b_{i'}$ is made 0 by the update algorithm. The next paragraph argues about an appropriate i-sequence being formed. Thus, it is easy to verify that (I1) to (I4) are maintained by the update algorithm. Note that (I1) implies that the space bound needed is at most $O(\log n \log m)$, (I2) is used implicitly to indicate which i-sequences are being tracked, and (I3, I4) give the order of the i-sequences tracked: a $(j+1)$-sequence appears earlier in the play than j-sequence. This is used implicitly when one combines the smaller j-sequences into a larger one as mentioned below.

When updating Anke's winning statistics by case 1.(a), one forms a new i-sequence of length 2^i by putting the older j-sequences for $j = i-1, i-2, \ldots, 1, 0$ together and appending the newly visited one-node sequence with value b; when $i = 0$, one forms a new 0-sequence of length 2^0 consisting of just the newly visited node with value b. Note that in case $i > 0$ both b and b_0 are of Anke's parity and therefore the highest valued node between the last member a of the older 0-sequence and the last node in the new i-sequence (both inclusive) has the value $\max\{b_0, b\}$ (by (I4) and Property-b_0 for the older 0-sequence). Furthermore, for every $j < i-1$, for the last node a of the older $(j+1)$-sequence and the first node a' of the older j-sequence, in the new i-sequence a highest valued node in the play between these two nodes a, a' (both inclusive) has value b_{j+1} (by (I4) and Property-b_{j+1} of older $(j+1)$-sequence) which, by choice, has Anke's parity. Thus the overall combined new sequence indeed satisfies the properties needed for an i-sequence, b is the value of the last node of this sequence and thus, currently, also the largest value of a node visited at or after the end of the sequence. All older j-sequences with $j < i$ are discarded and thus their entries are set back to $b_j = 0$.

The same rules apply to the updates of Boris' winning statistics with the roles of Anke and Boris interchanged everywhere.

Claim 12. *If a player is declared a winner by the algorithm, then the play contains a loop with its maximum valued node being a node of the player.*

To prove the claim, it is assumed without loss of generality that Anke is declared the winner by the algorithm. The play is won by an i-sequence being observed in Anke's winning statistics with $2^i > 2n$; thus some node occurs at least three times in the i-sequence and there are $h, \ell \in \{1, 2, 3, \ldots, 2^i\}$ with $h < \ell$ such that the same player moves at a_h and a_ℓ and furthermore $a_h = a_\ell$ with respect to the nodes $a_1, a_2, a_3, \ldots, a_{2^i}$ of the observed i-sequence. The maximum value b' of a node between a_h and a_ℓ in the play is occurring between some a_k and a_{k+1} (both inclusive) for a k with $h \leq k < \ell$. Now, by the definition of an i-sequence, b' has Anke's parity. Thus a loop has been observed for which the maximum value of a node in the loop has Anke's parity.

Claim 13. *If a player follows a memoryless winning strategy, then the opponent is never declared a winner.*

To prove the claim, suppose that a player follows a memoryless winning strategy but the opponent is declared a winner. Then the opponent, by Claim 12, goes into a loop with the maximum node of the opponent's parity. Hence, the opponent can cycle in that loop forever and win the play, a contradiction.

Claim 14. *If a player follows a memoryless winning strategy then the player is eventually declared a winner.*

To prove the claim, it is assumed that the player is Anke, as the case of Boris is symmetric. The values b_i analysed below refer to Anke's winning statistics. Assume that the sequence of values of the nodes in an infinite play of the game has the limit superior c which, by assumption, is a value of Anke's parity. To prove the claim one needs to argue that eventually b_i becomes non-zero for an i with $2^i > 2n$. For this purpose it will be argued that a counter to be defined, associated with the values of b_i's, eventually keeps increasing (except for some initial part of the play, where it may oscillate). This is argued by using $count(c, t)$ below, which gives the value of the counter after t steps of the play.

Consider a step as making a move and updating of the statistics. For each step t let $b_k(t)$ refer to the value of b_k at the end of step t (that is, after the updates in the statistics following the t-th move in the play). Let $B_c(t)$ be the set of all k such that $b_k(t)$ has Anke's parity and $b_k(t) \geq c$. Let

$$count(c, t) = \textstyle\sum_{k \in B_c(t)} 2^k.$$

Now it is shown that whenever at steps t, t' with $t < t'$, a move to a node with value c was made and no move, strictly between steps t, t', was made to any node with value $c' \geq c$, then $count(c, t) < count(c, t')$. To see this, let i be the largest index for which there is a step t'' with $t < t'' \leq t'$ such that b_i is updated at step t''.

Note that this implies [$b_i(t) < c$ or $b_i(t)$ is of Boris' parity], and [$0 < b_i(t'') \leq c$]. Now, in the case that $b_i(t'') < c$, it holds that $t'' < t'$ and at time t', condition 1.(b) of the update algorithm will ensure that an update (either 1.(a) or 1.(b)) is done to enforce $b_i(t') = c$. Thus

$$count(c, t') - count(c, t) \geq 2^i - \textstyle\sum_{j \in B_c(t): j < i} 2^j \geq 1.$$

Accordingly, once all moves involving nodes larger than c in value have been done in the play, there will still be infinitely many moves to nodes of value c and for each two subsequent such moves at t, t' the inequality $count(c, t) + 1 \leq count(c, t')$ will hold. Consequently, the number $count(c, t)$, for sufficiently large t where a move to a node with value c is made at step t, needs to have, for some i, $b_i(t) \geq c$ and $2^i > 2n$; thus the termination condition of Anke will terminate the play with a win.

The above arguments show that an alternating Turing machine can simulate both players and, taking the winning statistics into account, will accept the computation whenever Anke has a winning strategy for the game.

Recall that an alternating Turing machine can be viewed as a game between two players, Anke (existential) and Boris (universal) which perform in turns part of the computations and can branch in the part they do; when the game at the end terminates, it says which player has won; if Anke wins it means "accept" and if Boris wins it means "reject"; if it never terminates, it means "undecided".

An alternating Turing machine can decide a set iff for every input x, if $x \in L$ then Anke has a winning strategy for the alternating Turing machine and can force an "accept" else Boris has a winning strategy for the alternating Turing machine and can avoid that it comes to an "accept". For the alternating Turing machine, in order to simulate the game, one has to keep track of the following pieces of information: the winning statistics of the players; the current node in the play and the player who is to move next. Thus, the alternating Turing machine uses only $O(\log(n) \cdot \log(m))$ space to decide whether the parity game, from some given starting point, will be won by Anke (or Boris), provided the winner plays a memoryless winning strategy (which always exists when the player can win the parity game). $\Box$

Chandra, Kozen and Stockmeyer [13] showed how to simulate an alternating Turing machine working in polylogarithmic space by a deterministic Turing machine working in quasipolynomial time. Their simulation bounds for the alternating Turing machine described in Theorem 8 give a deterministic Turing machine working in time $O(n^{c \log(m)})$ for some constant c. As mentioned above, one can always assume that in a parity game with n nodes, with values from $\{1, 2, 3, \ldots, m\}$, one can choose $m \leq n$, so one gets the following parameterised version of the main results that parity games can be solved in quasipolynomial time.

Theorem 15. *There is an algorithm which finds the winner of a parity game with n nodes and values from $\{1, 2, 3, \ldots, m\}$ in time $O(n^{c \log(m)})$.*

For some special choices of m with respect to n, one can obtain even a polynomial time bound. McNaughton [61] showed that for every constant m, one can solve a parity game with n nodes having values from $\{1, 2, 3, \ldots, m\}$ in time polynomial in n; however, in all prior works the degree of this polynomial depends on m [40]; subsequent improvements were made to bring the dependence from approximately $n^{m+O(1)}$ first down to $n^{m/2+O(1)}$ [9,73] and then to approximately $n^{m/3+O(1)}$ [53,72]. The following theorem shows that one can bound the computation time by a fixed-degree polynomial in n, for all pairs (m, n) with $m < \log(n)$.

Theorem 16. *If $m \leq \log(n)$ then one can solve the parity game with n nodes having values from $\{1, 2, 3, \ldots, m\}$ in time $O(n^5)$.*

Proof. Note that Theorem 8 actually shows that the following conditions are equivalent:

- Anke can win the parity game;
- Anke can play the parity game such that her winning statistics matures while Boris' winning statistics does not mature.

Thus one can simplify the second condition and show that it is equivalent to the following two games [57,74]:

- One only maintains Anke's winning statistics and a play terminates with a win for Anke iff she is eventually declared a winner and the play is a win for Boris iff it runs forever;
- One only maintains Boris' winning statistics and a play is a win for Anke iff it never happens that the winning statistics of Boris make him to be declared a winner.

The first game is called a reachability game [57] and the second game a survival game [74, Chapter 9]. Both games are isomorphic, as they are obtained from each other only by switching the player who is supposed to win. Such type of reductions, though not with good complexity bounds, were also considered by Bernet, Janin and Walukiewicz [3]. The reachability game to which one reduces the parity game, can now be described as follows.

- The set Q of nodes of the reachability game consists of nodes of the form $(a, p, \tilde{b})$ where a is a node of the parity game, the player $p \in \{\text{Anke}, \text{Boris}\}$ moves next and $\tilde{b}$ represents the winning statistics of Anke.
- The starting node is $(s, p, \tilde{0})$, where $\tilde{0}$ is the vector of all b_i with value 0, s is the starting node of the parity game and p is the player who moves first.
- Anke can move from $(a, \text{Anke}, \tilde{b})$ to $(a', \text{Boris}, \tilde{b}')$ iff she can move from a to a' in the parity game and this move causes Anke's winning statistics to be updated from $\tilde{b}$ to $\tilde{b}'$ and $\tilde{b}$ does not yet indicate a win of Anke.
- Boris can move from $(a, \text{Boris}, \tilde{b})$ to $(a', \text{Anke}, \tilde{b}')$ iff he can move from a to a' in the parity game and this move causes Anke's winning statistics to be updated from $\tilde{b}$ to $\tilde{b}'$ and $\tilde{b}$ does not yet indicate a win of Anke.

The number of elements of Q can be bounded by $O(n^4)$. First note that the number of increasing functions from $\{0, 1, 2, \ldots, \lceil \log(n) \rceil + 2\}$ to $\{1, 2, 3, \ldots, \lceil \log(n) \rceil\}$ can be bounded by $O(n^2)$, as any such sequence $(b'_0, b'_1, b'_2 \ldots, b'_{\lceil \log(n) \rceil + 2})$ can be represented by the subset $\{b'_k + k : 0 \leq k \leq \lceil \log(n) \rceil + 2\}$ of $\{1, 2, 3, \ldots, 2\lceil \log(n) \rceil + 2\}$ and that there are at most $O(n^2)$ such sets. Further, note that $b'_k \leq b'_{k+1}$ implies $b'_k + k < b'_{k+1} + k + 1$ and thus all b'_k can be reconstructed from the set. Given a winning statistics $\tilde{b} = (b_0, b_1, b_2, \ldots, b_{\lceil \log(n) \rceil + 2})$, one defines $b'_0 = \max\{1, b_0\}$ and $b'_{k+1} = \max\{b'_k, b_{k+1}\}$ and notes that only those b_k with $b_k = 0$ differ from b'_k. Thus one needs at most $\lceil \log(n) \rceil + 3$ additional bits to indicate which b_k is 0. The overall winning statistics can then be represented by $3\lceil \log(n) \rceil + 5$ bits. Furthermore, one needs 1 bit to represent the player and $\lceil \log(n) \rceil$ bits to represent the current node in the play. Accordingly, each node in Q can be represented with $4\lceil \log(n) \rceil + 6$ bits resulting in $O(n^4)$ nodes in Q. The set Q itself can be represented by using a set of such representations of nodes.

Note that one can compute the set Q of vertices and determine a list of nodes $Q' \subseteq Q$ where Anke's winning statistics indicate a win in time $O(|Q| \cdot n)$; the set Q' is the set of target nodes in the reachability game.

The next proposition shows that the so constructed reachability game can be decided in time $O(|Q| \cdot n)$ by a well-known algorithm. For the general case of a reachability game, the time complexity is linear in the number of vertices plus number of edges of the game graph; note that the reachability game constructed has $|Q|$ nodes and $|Q| \cdot n$ edges. This completes the proof. $\square$

The below algorithm is listed explicitly by Khaliq and Imran [56] and appeared much earlier in the literature, though sometimes in other or only related settings [1,22,43,45,50]. The algorithm is now included for the reader's convenience.

Proposition 17 (Beeri [1], Cook [22], Gurevich and Harrington [45], Immerman [50]). *Given a reachability game with a set Q of nodes, a subset $Q' \subseteq Q$ of target nodes Q', out degree up to n per node and start node s, one can decide in time $O(|Q| \cdot n)$ which player can win the game.*

Proof. One computes for each node $q \in Q$, a linked list of q's successors (which are at most n in number) and a linked list of q's predecessors. Note that the collection of all the successor and predecessor lists for different nodes in Q taken together has the length at most $|Q| \cdot n$. These lists can also be generated in time $O(|Q| \cdot n)$.

Note that a node q is a winning node for Anke if $q \in Q'$ or either Anke moves from q and one successor node of q is a winning node for Anke or Boris moves from q and all successor nodes of q are winning node for Anke. This idea leads to the algorithm below.

Next, for each node q, a tracking number k_q is introduced and maintained such that the winning nodes for Anke will eventually all have $k_q = 0$, where k_q indicates how many further times one has to visit the node until it can be declared a winning node for Anke. The numbers k_q are initialised by the following rule:

- On nodes $q \in Q'$ the number k_q is 1;
- On nodes $q = (a, \text{Anke}, \tilde{b}) \notin Q'$, the number k_q is initialised as 1;
- On nodes $q = (a, \text{Boris}, \tilde{b}) \notin Q'$, the number k_q is initialised as the number of nodes q' such that Boris can move from q to q'.

These numbers can be computed from the length of the list of successors of q, for each $q \in Q$. Now one calls the following recursive procedure, initially for all $q \in Q'$ such that each call updates the number k_q. The recursive call does the following:

- If $k_q = 0$ then return without any further action else update $k_q = k_q - 1$.
- If after this update it still holds $k_q > 0$, then return without further action.
- Otherwise, that is when k_q originally was 1 when entering the call, recursively call all predecessors q' of q with the same recursive call.

After the termination of all these recursive calls, one looks at k_q for the start node q of the reachability game. If $k_q = 0$ then Anke wins else Boris wins.

In the above algorithm, the predecessors of each node $q \in Q$ are called at most once from a call in q, namely when k_q goes down from 1 to 0; furthermore, this is the time where it is determined that the node is a winning node for Anke. Thus there are at most $O(|Q| \cdot n)$ recursive calls and the overall complexity is $O(|Q| \cdot n)$.

For the verification, the main invariant is that, for nodes $q \in Q - Q'$, k_q indicates how many more successors of q one still has to find which are winning nodes for Anke until q can be declared a winning node for Anke. In case that Anke's winning statistics has matured in the node q, the value k_q is taken to be 1 so that the node is processed once in all the recursive calls in the algorithm. For nodes where it is Anke's turn to move, only one outgoing move which produces a win for Anke is needed. Consequently, one initialises k_q to 1 and as soon as this outgoing node is found, k_q goes to 0, which means that the node is declared a winning node for Anke. In case the node q is a node where Boris moves then one has to enforce that Boris has no choice but to go to a winning node for Anke. Thus k_q is initialised to the number of moves which Boris can make in this node; each time when one of these successor nodes is declared a winning node for Anke, k_q goes down by one. Observe that once the algorithm is completed, the nodes with $k_q = 0$ are exactly the winning nodes for Anke in the reachability game. $\square$

The next result carries over the methods of Theorem 16 to the general case, that is, use everything except those parts which make use of $m \leq \log(n)$. So the size of the code representing a winning statistics for Anke is given by $\lceil \log(n) \rceil + 3 \leq \log(n) + 4$ numbers of $\lceil \log(m+1) \rceil \leq \log(m) + 1$ bits. As $\log(m) \leq \log(n)$, the overall size of representation of a node in the set Q of nodes of the reachability game can be bounded by $\log(n) \cdot (\log(m) + 5) + c$. Hence, the size of $|Q|$ is $O(n^{\log(m)+5})$ and the number of edges in the reachability game is $O(n^{\log(m)+6})$.

For many decision problems in **NP**, in particular for the **NP**-complete ones, one can find solutions witnessing the given answer (like the winning strategy for the winner of the parity game) by solving several variants of the decision problem where more and more parameters of the problem are fixed by constants [2]. This is now outlined for finding the memoryless winning strategy of the winner of a parity game using an algorithm which decides who is the winner. For the ease of notation, assume that Anke can win the game on a graph (V, E). Now one does the following steps to retrieve the winning strategy:

1. Maintain, for each node $a \in V$, a list of possible successors V_a which is initialised as $\{b : (a, b) \in E\}$ at the beginning.
2. If there is no node $a \in V$ with, currently, $|V_a| > 1$,
   then one terminates with a winning strategy for Anke in the parity game being to move from every node a to the unique node in V_a,
   else one selects a node $a \in V$ with $|V_a| > 1$.
3. Now one splits V_a into two nearly equal sized subsets V_a' and V_a'' with $|V_a'| \leq |V_a''| \leq |V_a'| + 1$.
4. One replaces V_a by V_a' and permits, in the derived reachability game, moves from $(\tilde{a}, \text{Anke}, \tilde{b})$ to $(\tilde{a}', \text{Boris}, \tilde{b}')$ only when $\tilde{a}' \in V_{\tilde{a}}$ for all nodes $\tilde{a}$.
5. If Anke does not win this game, then one replaces $V_a = V_a''$, else one keeps $V_a = V_a'$.
6. Go to step 2.

The above algorithm works since whenever Anke has a winning strategy for the parity game, then there is a memoryless one and therefore when splitting the options at node a, some memoryless winning strategy either always takes a node from V_a' or always takes a node from V_a''. It is straightforward to verify that the above loop runs $n \log(n)$ rounds and each round involves $O(|Q| \cdot n)$ time plus one solving of the reachability game, which can also be solved in time $O(|Q| \cdot n)$. Thus one can derive the following result.

Theorem 18. *There is an algorithm which finds the winner of a parity game with n nodes and values from $\{1, 2, 3, \ldots, m\}$ in time $O(n^{\log(m)+6})$. Furthermore, the algorithm can compute a memoryless winning strategy for the winner in time $O(n^{\log(m)+7} \cdot \log(n))$.*

Thus, as shown, when $m \leq \log(n)$ the runtime is $O(n^5)$; if $m > \log(n)$ then $2^m > n$ and one can bound $n^{\log(m)+6}$ from above by $2^{m\cdot(\log(m)+6)}$. Thus one has the bound $O(n^5 + 2^{m\cdot(\log(m)+6)})$ for the runtime of solving a parity game with n nodes and values from $\{1, 2, 3, \ldots, m\}$. In other words, parity games are fixed-parameter tractable for their main parameter m.

Corollary 19. *Parity games are in the class* **FPT** *and can be solved in time* $O(n^5 + 2^{m(\log(m)+6)})$.

Note that the the number of possible increasing seqeunces over $\{1, 2, \ldots, m\}$ of length $\lceil\log(n)\rceil+3$ is $\binom{m+2\cdot(\lceil\log(n)\rceil+3)}{\lceil\log(n)\rceil+3}$. Using this, follow-up work obtained better bounds on the runtime by using that the translation into the reachability game provides a game with at most

$$\binom{m + 2\cdot(\lceil\log(n)\rceil + 3)}{\lceil\log(n)\rceil + 3} \cdot n^2$$

edges.

The above formula led to the bound $O(2^m \cdot n^4)$ [14] which is based on the fact that $\binom{i}{j} \leq 2^i$ for all i, j. A further estimate can be obtained by slightly increasing the binomial upper bound to

$$\binom{(\lceil m/\log(n)\rceil + 2)\cdot(\lceil\log(n)\rceil + 3)}{\lceil\log(n)\rceil + 3} \cdot n^2$$

and then using common estimates on binomials, where the upper number is a multiple of the lower number. The calculations provide a runtime bound of

$$O(\lceil m/\log(n)\rceil^4 \cdot n^{3.45+\log(\lceil m/\log(n)\rceil+2)});$$

this. Similar bounds of this type were obtained by several researchers [34,42,54,74]. Subsequent improvements included replacing the term n^2 in the above formulas by the number of edges in the parity game [34,42,54].

The main improvement over the current algorithm by follow-up work is, however, the usage of space. The current algorithm uses quasipolynomial time and quasipolynomial space. Subsequent work has brought down this complexity from quasipolynomial to quasilinear [34,54]; more precisely Jurdziński and Lazić have the space bound $O(n\cdot\log(n)\cdot\log(m))$ and Fearnley, Jain, Schewe, Stephan and Wojtczak [34] have the space bound $O(n\cdot\log(n)\cdot\log(m) + \ell\cdot\log\log(n))$, where ℓ is the number of edges in the parity game and thus $\ell \leq n^2$; the time bounds of both algorithms are approximately the same as those of the algorithm presented here, but due to the better space bound, an additional overhead from managing large space can be avoided in an implementation.

Lehtinen [59] introduced the notion of the register index complexity of a parity game and showed that every parity game has register index complexity of at most $\log(n)+1$. She then gave an algorithm to translate the given parity game of register index k into a usual parity game of size $O(m^k \cdot n)$ with $2k+1$ values on the edges. This game can then be solved in polynomial time (with respect to $m^k \cdot n$) as the number $2k+1$ of values is bounded logarithmically in the number of nodes; furthermore, results prior to the current work would also have already shown that the translated game can be solved in quasipolynomial time and thus Lehtinen [59] has supplied a quasipolynomial time algorithm for solving parity games which can be verified without making reference to the present work.

4 Parity Games versus Muller Games

Muller games are a well-studied topic [7,8,61,76,80] and they had been investigated as a general case already before researchers aimed for the more specific parity games. A Muller game (V, E, s, G) consists of a directed graph (V, E), a starting node s and a set $G \subseteq \{0, 1\}^V$. For every infinite play starting in s, one determines the set U of nodes visited infinitely often during the play: if $U \in G$ then Anke wins the play else Boris wins the play. In a Muller game the complement of G is closed under union iff for all $U, U' \notin G$, the set $(U \cup U')$ is not in G.

For complexity assumptions, it is natural to consider the case where G is not given as an explicit list, but as an algorithm, which is polynomial in size and which runs in polynomial time and which computes the membership of a set U (given by its explicit list) in the set G or some similar equivalent effective representation. The reason for considering such a representation for G is that Horn [47] showed that if G is given as an explicit list of all possible sets of nodes infinitely visited when Anke wins, then the resulting game is solvable in polynomial time in the sum of the number of nodes and the number of explicitly listed sets. Hence, only more flexible ways of formulating winning conditions lead to interesting cases of Muller games.

For Muller games, Björklund, Sandberg and Vorobyov [5] considered a parameter which is given by the number of colours. For this, they assign to every node a colour from $\{1, 2, 3, \ldots, m\}$ and take G to be some set of subsets of $\{1, 2, 3, \ldots, m\}$. Then U is not the set of infinitely often visited nodes, but instead, the set of colours of the infinitely often visited nodes. Again, if $U \in G$, then Anke wins the play, else Boris wins the play. Coloured Muller games permit more compact representations of the winning conditions. In the worst case there is a 2^m-bit vector, where m is the number of colours; however, one also considers the case where this compressed winning condition is given in a more compact form, say by a polynomial sized algorithm or formula.

In the following, the interactions between Muller games, memoryless winning strategies and parity games are presented. The first result is due to Emerson [30] and Zielonka [80, Corollary 11] and the second one is in Hunter's Thesis [48].

Theorem 20 (Emerson [30] and Zielonka [80]). *Consider a Muller game (V, E, s, G) in which the complement of the set G of winning conditions is closed under union. If Anke has a winning strategy in this Muller game then Anke has also a memoryless winning strategy.*

Proof. The possible choices for Anke at any node will be progressively constrained. The proof is by induction on the number of possible moves of Anke in the constrained game. The result holds when, for each node, Anke has only one choice of move. For the induction step, suppose some node v for Anke's move has more than one choice. It is now shown that for some fixed Anke's move at node v, Anke has a winning strategy; thus one can constrain the move of Anke at node v and by induction this case is done. Suppose, by way of contradiction, that for every Anke's move w at v, Boris has a winning strategy S_w. This allows Boris to have a winning strategy for the whole game as follows.

Assume without loss of generality that the play starts with Anke's move at v. Intuitively, think of Boris playing several parallel plays against Anke (each play in which Anke moves w at node v, for different values of w) which are interleaved. For ease of notation, consider the individual play with Anke using move w at node v as play H_w, and the interleaved full play as H.

Initially H and all the plays H_w, are at the starting point. At any time in the play H, if it is Anke's move at v and Anke makes the move w', then Boris continues as if it is playing the play $H_{w'}$ (and suspends the previous play H_w if $w \neq w'$). Thus the nodes visited in H can be seen as the merger of the nodes visited in the plays H_w, for each choice w of Anke at node v. This implies that the set of nodes visited infinitely often in H is equal to the union of the sets of nodes visited infinitely often in the various H_w. As Boris wins each play H_w which is played for infinitely many moves, by closure of the complement of G under union, Boris wins the play H. $\square$

As a parity game is also a Muller game in which G is closed under union for both Anke and Boris, the following corollary holds.

Corollary 21 (Emerson and Jutla [32], Mostowski [62]). *The winners in parity games have memoryless winning strategies.*

Hunter [48, page 23] showed the following characterisation for Muller games. Note that McNaughton [61] also investigated Muller games with memoryless strategies and characterised them through the concept of splitting [61], which is just another way of stating that both G and its complement are union-closed. However, his paper does not connect these Muller games with parity games explicitly.

Theorem 22 (Hunter [48]). *Every Muller game (V, E, s, G) in which both G and its complement are closed under the union operation is a parity game and the translation can be done in polynomial time whenever the winning set G can be decided in polynomial time.*

Proof. In this proof a parity game isomorphic to the given Muller game will be constructed. In this parity game player Anke owns the nodes with even value and Boris owns the nodes with odd value. Given V, let

$$V_1 = \{a \in V : \{a\} \in G\} \text{ and } V_2 = \{b \in V : \{b\} \notin G\}.$$

Obviously V is the disjoint union of V_1 and V_2. By the closure under union, any subset $V' \subseteq V_1$ is in G and no subset $V' \subseteq V_2$ is in G.

To prove the theorem, values will be inductively assigned to the nodes one by one.

Suppose values have already been assigned to all nodes in $V - V'$, where V' is initially V. Then, assign the value to one node in V' as follows. Let $V_1' = V' \cap V_1$ and $V_2' = V' \cap V_2$.

Case 1: Suppose $V' \in G$. Then, there is a node $a \in V_1'$ such that $\{a\} \cup V_2' \in G$; otherwise, $V' \notin G$ since complement of G is closed under the union operation. Now let $V_1'' \subseteq V_1'$ and $V_2'' \subseteq V_2'$. The set $\{a\} \cup V_2''$ is in G, as otherwise $(\{a\} \cup V_2'') \cup V_2'$ is not in G, in contradiction to the choice of a. Furthermore, as $V_1'' \cup \{a\} \in G$, $(V_1'' \cup \{a\}) \cup (\{a\} \cup V_2'') = \{a\} \cup V_1'' \cup V_2''$ is in G. Thus whenever $V'' \subseteq V'$ and $a \in V''$, $V'' \in G$. Hence, the value $2|V'|$ is assigned to a accordingly.

Case 2: Suppose $V' \notin G$. Then, there exists a node $b \in V_2'$ such that $\{b\} \cup V_1' \notin G$, by reasons similar to those given in Case 1. Note that this implies that whenever $V'' \subseteq V'$ and $b \in V''$ then

$V'' \notin G$. Hence, the value $2|V'|+1$ is assigned to b.

The above process of assigning values to nodes is clearly consistent, as if $V'' \subseteq V'$ is the set of infinitely visited nodes, and as in Case 1, if a is V'', then Anke wins and in Case 2, if b is in V'' then Boris wins. It follows that this Muller game is a parity game. □

Besides the standard coloured Muller game of Björklund, Sandberg and Vorobyov [5], one can also consider the memoryless coloured Muller game. These are considered in order to see whether the game is easier to solve if one permits Anke only to win when she follows a memoryless strategy, otherwise she loses by the rules of the game. The main finding comparing memoryless coloured Muller games with standard coloured Muller games is as follows: On one hand, memoryless coloured Muller games are easier in terms of the best known complexity class to which they belong, memoryless coloured Muller games are in $\mathbf{\Sigma}_2^P$ while the decision complexity of standard coloured Muller games is in **PSPACE**; on the other hand, the time complexity of memoryless coloured Muller games is worse, as one cannot exploit small number of colours to bring the problem into P, already four colours makes it **NP**-hard to find the winner in memoryless coloured Muller games (see Theorem 27).

Björklund, Sandberg and Vorobyov [5] proved that the coloured Muller game is fixed-parameter tractable iff the parity game is fixed-parameter tractable (with respect to the number of values m of the parity game). It follows from Theorem 16 that also the coloured Muller game is fixed-parameter tractable. More precisely, McNaughton [61] and Björklund, Sandberg and Vorobyov [5] showed the following result.

Theorem 23 (Björklund, Sandberg and Vorobyov [5]; McNaughton [61]). *One can translate a coloured Muller game with m colours and n nodes in time polynomial in $m! \cdot n$ into an equivalent parity game with $2m$ colours and $m! \cdot n$ nodes.*

Proof. In this proof, one considers Muller games with nodes possibly having multiple colours. The idea is based on the last appearance record of the colours.

Each node v from the original game will be replaced by all nodes of the form (v,r) in the new game, where r denotes an ordered list of colours as to how recently they were observed in the nodes visited before the current node.

One lets Anke have the odd and Boris the even numbers. The value of the node (v,r) is computed in two steps. First one computes the set U of colours in r which are at least as recent as one of the colours of v in the Muller game, that is, U is the set of colours whose position might be affected by an update of r when leaving the current node for the next node. For example, if the game has four colours which were observed in the order (c_1,c_2,c_3,c_4) (c_1 is the most recent colour) and if the node v in the Muller game carries the colours c_2 and c_3 then $U=\{c_1,c_2,c_3\}$ and when passing to the next node r will be updated to $r'=(c_2,c_3,c_1,c_4)$. Second, one lets the value of the node (v,r) be $2\cdot|U|+1$ in the case that U is a winning set for Anke in the Muller game and $2\cdot|U|+2$ in the case that U is a winning set for Boris in the Muller game.

If a player can move from v to w in the original Muller game, then the player can now move from (v,r) to (w,r') in the constructed parity game where r' is obtained from r by moving all the colours belonging to v to the front, as they are most recent when arriving in w, and by keeping the other colours in their order behind the new recent colours; other moves than those derived ones are not possible. Furthermore, when s is the starting node in the original coloured Muller game, then the new starting node in the parity game is of the form (s, r) for some arbitrary but fixed record r.

Given now a play $(v_0, r_0), (v_1, r_1), (v_2, r_2), \ldots$ in the parity game, it defines a play $v_0, v_1, v_2, \ldots$ in the original Muller game and a set U which consists of the colours of the infinitely often visited nodes. For almost all n, these colours in U are in the front of the last appearance record r_n. As each of them is occurring infinitely often, there are infinitely many nodes (v_n, r_n) in the run where one of the colours of v_n is the last member of U in the current record r_n. It follows that U is the set of selected colours for (v_n, r_n) and the node (v_n, r_n) has Anke's parity iff U is a winning set for Anke. Furthermore, only the nodes where all colours of U are taken into account have the maximal parity of the run. For that reason, Anke wins the run in the parity game iff she wins the corresponding run in the original Muller game.

Assume now that Anke has a winning strategy for the parity game. Then, when playing the original Muller game, in her memory Anke can keep track of the appearance record r_n for the current node v_n and then, in the case that it is her turn, move to that v_{n+1} such that in the parity game she would have made a move to a node of the form (v_{n+1}, r_{n+1}). As it is a winning strategy, the derived play in the parity game would be winning for Anke and thus also winning in the original play in the Muller game. The situation when Boris has a winning strategy for the parity game is similar, as he can then translate by the same method his winning strategy into one for the coloured Muller game. Thus the winner of the original Muller game is the same as the winner of the translated parity game, that is, the original game is equivalent to the translated game.

The bound on the number of nodes is $n \cdot m!$, the number of values in the game is $2m + 2$ in the case that one allows nodes without colours so that the set U of the colours of the infinitely often visited nodes can be empty. It is $2m$ if every node needs to have at least one colour, as then one can cut out the case of no colour and would assign to the set U computed for a node (v, r) either the value $2|U| - 1$ or $2|U|$, depending on the parity of the player who wins when U is the set of colours of the infinitely often visited nodes. ☐

Now one uses this result in order to prove the bounds on the algorithm to solve the coloured Muller games. Note that $\log(m! \cdot n) \geq 2m$ for all $m \geq 24$ and $n \geq m$: $\log(m!) \geq \log(8^{m-8}) \geq 3 \cdot (m - 8) = 3m - 24$. For $m \geq 24$, $3m - 24 \geq 2m$. Thus, the remaining cases can be reduced to finite ones by observing that for all m and $n \geq \max\{m, 2^{48}\}$, $\log(m! \cdot n) \geq 2m$. So, for almost all pairs of (m, n), $\log(m! \cdot n) \geq 2m$ and therefore one can use the polynomial time algorithm of Theorem 16 to get the following explicit bounds.

Theorem 24. *One can decide in time $O(m^{5m} \cdot n^5)$ which player has a winning strategy in a coloured Muller game with m colours and n nodes.*

For the special case of $m = \log(n)$, the corresponding number of nodes in the translated parity game is approximately $n^{\log(\log(n))+2}$ and the polynomial time algorithm of Theorem 16 becomes an $O(n^{5 \log\log(n)+10})$ algorithm. The algorithm is good for this special case, but the problem is in general hard and the algorithm is slow.

One might ask whether this bound can be improved. Björklund, Sandberg and Vorobyov

[5] showed that under the Exponential Time Hypothesis it is impossible to improve the above algorithm to $O(2^{o(m)} \cdot n^c)$, for any constant c. Here the Exponential Time Hypothesis says that the problem **3SAT** with n variables is not solvable in time $O(2^{o(n)})$. The following result enables to get a slightly better lower bound.

Theorem 25. *A Muller game with m colours and n nodes and $1 \leq m \leq n$ cannot be solved in time $2^{o(m \cdot \log(m))} \cdot Poly(n)$, provided that the Exponential Time Hypothesis is true.*

Proof. Note that for this result, multiple colours per node are allowed. However, one can translate a coloured Muller game with multiple colours per node into one with one colour per node and $m' = m+1$ colours and $n' = n \cdot m$ nodes and as it is required that $m \leq n$, the expressions $2^{o(m \cdot \log(m))} \cdot Poly(n)$ and $2^{o(m' \cdot \log(m'))} \cdot Poly(n')$ contain the same runtimes of algorithms.

Theorem 30 provides as a special case a translation of k-dimensional parity games with n nodes and 3 values per dimension into coloured Muller games with n nodes and $m = 2k$ colours without changing the winner; the underlying game is not changed, only the way the the plays are evaluated by the auxiliary structure of multi-dimensional parities is replaced by colours for the nodes. Furthermore, Theorem 31 shows that if a k-dimensional parity game with 3 values per dimension can be solved in time $2^{o(k \cdot \log(k))} \cdot Poly(n)$ then the Exponential Time Hypothesis would fail. The proof of current theorem then follows from the fact that if $m = 2k$ then $2^{o(k \cdot \log(k))} = 2^{o(m \cdot \log(m))}$, which is based on the equations $o(m \cdot \log(m)) = o(2k \cdot \log(2k)) = o(k \cdot \log(2k)) = o(k \cdot \log(k) + k \cdot 2) = o(k \cdot \log(k))$. This completes the proof. $\square$

Memoryless games are games where Anke wins iff she (a) plays a memoryless strategy and (b) wins the game according to the specification of the game. If she does not do (a), this is counted as a loss for her. This was already defined by Björklund, Sandberg and Vorobyov [5, Section 5] for Streett games and it can also be defined for Muller games.

The complexity of the memoryless games differs from those of normal games. Björklund, Sandberg and Vorobyov [5, Section 5] considered memoryless Streett games (called Quasi-Streett games in their paper) and showed that these are **W**[1]-hard. This result implies that memoryless coloured Muller games are **W**[1]-hard.

The next theorem establishes the complexity of finding memoryless strategies for player Anke at Muller games. For this one needs some effective way of representing the winning conditions on the colours and here it is assumed that they are given by a Boolean formula or circuit of size polynomial in the game (one has to fix such a polynomial and any polynomial which is at least cubic in the number of colours would be sufficient for the hardness). The hardness part in (b) slightly extends what is known in the literature.

Dawar, Horn and Hunter [24] extended a conference publication of Horn [46] in which it is shown that Muller games, where the winning condition is given as an explicit list of all sets of infinitely often visited nodes which are winning, is decidable in polynomial time; here the polynomial time algorithm, for input size, also takes into account the length of the explicit list. Dziembowski, Jurdziński and Walukiewicz [29] investigated mainly the space complexity needed to implement strategies and provided some applications towards the complexity of solving the problem. Zielonka [80] used similar methods to show **NP**-hardness of the Muller games, even in the special case of games where player Anke, in case she wins, also has a memoryless winning strategy.

Theorem 26 (See also Dawar, Horn and Hunter [24], Dziembowski, Jurdziński and Walukiewicz [29], Horn [46], Zielonka [80]).

(a) *The problem whether Anke can win a memoryless coloured Muller game is* $\mathbf{\Sigma_2^P}$*-complete.*

(b) *Suppose A is a polynomial time computable set of instances of satisfiability formulas $F(x_1, \ldots, x_i, y_1, \ldots, y_j)$ with two types of variables which satisfy that for each choice of $(x_1, \ldots, x_i)$ there is at most one choice of $(y_1, \ldots, y_j)$ which makes $F(x_1, \ldots, x_i, y_1, \ldots, y_j)$ true. Let B be the set of all satisfiability formulas F for which the statement $(*)$ given as*

$$\exists x_1 \ldots \exists x_i \, \forall y_1 \ldots \forall y_j \, [F(x_1, \ldots, x_i, y_1, \ldots, y_j) \text{ is not satisfied}]$$

is true. Then there is a polynomial time many-one reduction from $A \cap B$ to the set of all coloured Muller games in which the winning conditions of Boris are closed under union such that $F \in A \cap B$ iff Anke is the winner of the game constructed for F. Furthermore, the problem whether Anke can win such a game is in $\mathbf{\Sigma_2^P}$.

Proof. First to see the membership in $\mathbf{\Sigma_2^P}$, consider the following well-known method: One guesses the memoryless winning strategy of Anke and then one computes the resulting graph where the successors of a node are not the original ones, but the new ones which can be reached if one first follows one step of Anke's strategy to a neighbour and then takes a move from that neighbour. In this new graph, only Boris is moving, so it is effectively a one-player-game. Now Boris can only win that new game iff there is the corresponding periodic path — a period is not longer than the number n of nodes times the number of colours — and one guesses a path of up to length n from the starting node to this period as well as the periodic part of the path and verifies that the periodic part produces a set of colours on which Boris wins. Thus if this cannot happen, then Anke has a winning strategy and that verification is in **coNP** so that the overall complexity is in $\mathbf{\Sigma_2^P}$.

The set of formulas F which satisfy $(*)$ is in general $\mathbf{\Sigma_2^P}$-complete. However, in the case of (b) one will enforce a promise, that is, take only those formulas which are members of a certain polynomial time computable set A satisfying the promise from the statement of the theorem; this makes the set $A \cap B$ incomplete for $\mathbf{\Sigma_2^P}$.

To show hardness, one reduces in both cases (a) and (b), satisfiability formulas of the form $F(x_1, \ldots, x_i, y_1, \ldots, y_j)$ to Muller games. First one adds additional variables $\tilde{x}_1, \ldots, \tilde{x}_i$ and modifies the formula $(*)$ to the following formula (@):

$\exists x_1 \ldots \exists x_i \, \forall \tilde{x}_1 \ldots \forall \tilde{x}_i \, \forall y_1 \ldots \forall y_j \, [x_1 \neq \tilde{x}_1 \vee \ldots \vee x_i \neq \tilde{x}_i \vee F(\tilde{x}_1, \ldots, \tilde{x}_i, y_1, \ldots, y_j)$ is not satisfied$]$.

The intuition behind the reduction is that Anke chooses the truth-values $x_1, \ldots, x_i$ and copies them to $\tilde{x}_1, \ldots, \tilde{x}_i$. Boris is then responsible for finding a satisfying assignment and this assignment is valid iff it does not produce any inconsistencies in the variables $\tilde{x}_1, \ldots, \tilde{x}_i, y_1, \ldots, y_j$. This will make it easier to detect which player is responsible for an inconsistent situation in the game and the evaluation of a winner of a play takes this into account.

For the reduction from a formula $F(x_1, \ldots, x_i, y_1, \ldots, y_j)$, in the Muller game, one introduces for each variable z the colours $pos(z)$ and $neg(z)$.

The Muller game graph consists of a list of subunits, where each subunit consists of one player choosing an option and then colours are assigned. First Anke chooses in i subunits among two choices which have in the positive case the colours $\{pos(x_h), pos(\tilde{x}_h)\}$ and in the negative case the colours $\{neg(x_h), neg(\tilde{x}_h)\}$ for $h = 1, 2, 3, \ldots, i$. After each subunit, the corresponding nodes lead to the entry node of the next subunit except for the last subunit where they lead (through a dummy node) to the entry node of the condition for the first clause. There are subunits for each clause in F and Boris has to choose between nodes representing the literals with the corresponding colours. So if the clause is $\tilde{x}_3 \vee y_1 \vee \neg(y_5)$ then Boris can move into one of three nodes with colours $\{pos(\tilde{x}_3)\}$, $\{pos(y_1)\}$, $\{neg(y_5)\}$ and these nodes all exit to next entry node of the next clause except for the last clause where they exit through a dummy node to the start node of the game.

Now given a set U of colours of the infinitely often visited nodes of a play, the winning condition for Boris is that either there is an $z \in \{x_1, \ldots, x_i\}$ where both $pos(z), neg(z)$ are in U or there is no $z \in \{\tilde{x}_1, \ldots, \tilde{x}_i, y_1, \ldots, y_j\}$ where both $pos(z), neg(z)$ are in U. In other words, Anke wins iff $\{z : pos(z) \in U \wedge neg(z) \in U\}$ is a nonempty subset of $\{\tilde{x}_1, \ldots, \tilde{x}_i, y_1, \ldots, y_j\}$.

Now, for the set of colours U on the infinitely often visited nodes in a play, if the condition on U is winning for Boris, then either Anke has played inconsistently (that is, it has made two different choices of $x_1, x_2, \ldots, x_i$) as witnessed by the colours $\{pos(z), neg(z)\}$ for some $z \in \{x_1, \ldots, x_i\}$ or Boris has played in a way that all variables are always instantiated the same way in the literals selected by Boris to witness the trueness of the clauses; furthermore, those z which are in $\{\tilde{x}_1, \ldots, \tilde{x}_i\}$ coincide with Anke's choice. Thus U witnesses that the formula F can be satisfied with Anke's choice of the $x_1, \ldots, x_i$. Therefore, if Boris has a winning strategy then all choices of $(x_1, \ldots, x_i)$ can be extended to a satisfying assignment for F. Note that Anke can win playing consistently whenever the $(x_1, \ldots, x_i)$ witnessing that $F \in B$ exists, indeed she can only win when she plays memorylessly. On the other hand, if each choice of $(x_1, \ldots, x_i)$ can be extended to a satisfying assignment for F then whatever Anke does, Boris can win the game: If Anke plays inconsistently, she loses; if Anke commits to some choice for $(x_1, \ldots, x_i)$ and always moves accordingly, then also Boris can always choose the literal witnessing the truth of clauses and the resulting colours do not give an inconsistent choice for any variable; those variables with neither $pos(z)$ nor $neg(z)$ appearing in the colours are not relevant for making the formula F true and can be ignored.

The argument above directly proves the result (a) and therefore the problem whether Anke can win a memoryless coloured Muller game is $\mathbf{\Sigma_2^P}$-complete.

For (b), assume that A and B are as in the theorem. As the non-members of A can be detected in polynomial time, without loss of generality, for the following analysis it is always assumed that the formulas F are from A. Furthermore, as above, Anke wins the constructed parity game iff the modified F satisfies (@) iff F satisfies $(*)$. Thus one only has to prove that the winning condition for Boris is closed under union when the promise is satisfied.

Thus consider two sets of colours V, W where Boris wins and let $U = V \cup W$. If there is a $z \in \{x_1, \ldots, x_i\}$ such that both $pos(z), neg(z) \in U$ then Boris wins. If such a z does not exist then U and thus V, W encode a fixed choice of the truth-values of $\{x_1, \ldots, x_i\}$. By the winning condition on the game, the variables $\{\tilde{x}_1, \ldots, \tilde{x}_i, y_1, \ldots, y_j\}$ all have at most one truth-assignment in the colours, as otherwise Boris would lose. Due to the promise of F, this truth-assignment depends uniquely on the choice of the truth-values of $\{x_1, \ldots, x_i\}$ and is thus same for both V and W and, furthermore, both V and W have, for every $z \in \{y_1, \ldots, y_j\}$, at least one of the colours $pos(z), neg(z)$, as otherwise there would be at least two satisfying assignments (as no value of z is enforced). Thus the union U equals to both V and W; it follows that U is a set of colours which is winning for Boris. So the winning conditions of Boris are closed under union. $\square$

Note that one can reduce sets in $\mathbf{NP} \cup \mathbf{coUP}$ to sets A (with corresponding B) satisfying the promise condition in part (b). To see this, consider sets in $\mathbf{NP}$ of the form $X = \{z : (\exists x_1, x_2, \ldots, x_i)[G(z, x_1, x_2, \ldots, x_i)]\}$, where $G(z, x_1, \ldots, x_i)$ can be solved in deterministic polynomial time. Then, for each z, one can construct CNF formulas $F_z(x_1, x_2, \ldots, x_i, y_1, \ldots, y_j)$ such that $F_z(x_1, x_2, \ldots, x_i, y_1, \ldots, y_j)$ is true iff $y_1, \ldots, y_j$ codes the deterministic computation of $G(z, x_1, \ldots, x_i)$ and $G(z, x_1, \ldots, x_i)$ is false. Here A would be the set of all formulas F_z. As there is only one deterministic computation of $G(z, x_1, \ldots, x_i)$, for each $x_1, \ldots, x_i$, there is at most one satisfying assignment for $F_z(x_1, x_2, \ldots, x_i)$. Furthermore, if $z \in X$, then for some appropriate choice of $x_1, x_2, \ldots, x_i$, $G(z, x_1, x_2, \ldots, x_i)$ is true, and thus $F_z(x_1, x_2, \ldots, x_i, y_1, \ldots, y_j)$ is not satisfied for at least one possible value of $y_1, \ldots, y_j$ (the one which codes the deterministic computation of $G(z, x_1, x_2, \ldots, x_i)$). In case $z \notin X$, for all $x_1, x_2, \ldots, x_i$, $G(z, x_1, x_2, \ldots, x_i)$ is false, and thus, for all $x_1, x_2, \ldots, x_i$, for $y_1, \ldots, y_j$ coding the deterministic computation of $G(z, x_1, \ldots, x_i)$, $F_z(x_1, x_2, \ldots, x_i, y_1, \ldots, y_j)$ is satisfiable. Thus, the requirements as in part (b) are satisfied.

Similar reduction can be done for problems X in $\mathbf{coUP}$, by using $i = 0$ (and thus no x_i's are used) and using $y_1, \ldots, y_j$ to code the computations of the $\mathbf{UP}$ machine. This would give that F_z satisfies $(*)$ iff $z \in X$.

The result that memoryless coloured Muller games can be solved in $\mathbf{\Sigma_2^P}$ stands in contrast to the fact that Dawar, Horn and Hunter [24] showed that deciding the winner of a Muller game is a **PSPACE**-complete problem.

The next result shows that unless **NP** can be solved in quasipolynomial time there is no analogue of the translation of Björklund, Sandberg and Vorobyov [5] from memoryless coloured Muller games into parity games. In contrast, solving memoryless coloured Muller games with four colours is already **NP**-complete and thus solving memoryless coloured Muller games is not in **XP**, unless $\mathbf{P} = \mathbf{NP}$.

Theorem 27. *Solving memoryless coloured Muller games with four colours is* **NP***-complete.*

Proof. For seeing that the game is in **NP**, one guesses the strategy and translates the game into a coloured Muller game with $2n$ nodes where each original node v is represented by two nodes (Anke, v) and (Boris, v) and the unique move from (Anke, v) goes to (Boris, w) for that w which is picked by the memoryless winning strategy and (Boris, v) can move to all (Anke, w) for which there is an edge in the original Muller game. By Theorem 23 mentioned above, one can first translate this intermediate coloured Muller game into a parity game with 8 values and $24 \cdot n$ nodes [5,61] and then solve the parity game in polynomial time $O(n^5)$, as $\log(24 \cdot n) \geq 8$ whenever $n \geq 3$.

For the **NP**-hardness, satisfiability is reduced to memoryless coloured Muller game as follows. For ease of writing the proof, Muller games where nodes determine the player moving are considered. This could be easily converted to a game where the moves of Anke and Boris alternate by inserting intermediate nodes if needed.

Suppose $x_1, x_2, x_3, \ldots, x_k$ are the variables and $y_1, y_2, y_3, \ldots, y_h$ are the clauses in a satisfiability instance. Without loss of generality assume that no variable appears both as positive and negative literal in the same clause. Then, the above instance of satisfiability is reduced to the following Muller game (where the graph is undirected graph):

1. $V = \{s\} \cup \{u_1, u_2, u_3, \ldots, u_k\} \cup \{v_1, v_2, v_3, \ldots, v_h\} \cup \{w_{i,j} : [1 \leq i \leq h]$ and $[1 \leq j \leq k]$ and $[x_j$ or $\neg x_j$ appears in the clause $y_i]\}$.
   Boris moves at nodes s and u_j with $1 \leq j \leq k$. Anke moves at all other nodes.
2. $E = \{(v_i, w_{i,j}), (w_{i,j}, u_j), (w_{i,j}, v_i), (u_j, w_{i,j}) : x_j$ or $\neg x_j$ appears in $y_i\} \cup \{(s, u_j), (u_j, s) : 1 \leq j \leq k\}$.
3. The colours are $\{x, y, +, -\}$; s has the colour y, all nodes u_j have the colour x; all nodes v_i have the colour y; for every node $w_{i,j}$ in the graph, if x_j appears in the clause y_i positively then the colour is $+$ else $\neg x_j$ appears in y_i and the colour is $-$.
4. The winning sets for Boris are $\{x, +, -\}$ and all subsets of $\{y, +, -\}$; the winning sets for Anke are $\{x, +\}$, $\{x, -\}$, $\{x\}$ and all supersets of $\{x, y\}$.

Now it is shown that the instance of satisfiability problem is satisfiable iff Muller game is a win for Anke playing in a memoryless way.

Suppose the instance is satisfiable. Then fix a satisfying assignment $f(x_j)$ for the variables, and let $g(y_i) = j$ such that x_j (or $\neg x_j$) makes the clause y_i true. Now Anke has the following winning strategy: At node v_i, move to $w_{i,g(y_i)}$. At node $w_{i,j}$, if $g(y_i) = j$ then move to u_j else move to v_i. Intuitively, at nodes v_i, Anke directs the play to the node $u_{g(y_i)}$ (via $w_{i,g(y_i)}$). Similarly, for the nodes $w_{i,j}$, Anke directs the play to $u_{g(y_i)}$ either directly or via nodes v_i and $w_{i,g(y_i)}$.

Thus, clearly, if an infinite play goes through colour y infinitely often, then it also goes through colour x infinitely often; thus Anke wins. On the other hand, if an infinite play does not go through colour y infinitely often, then the set of nodes the play goes through infinitely often is, for some fixed j, u_j and some of the nodes of the form $w_{i,j}$. But then, by the definition of Anke's strategy, the play can only go through nodes of colour $-$ finitely often (if $f(x_j)$ is true) and through nodes of colour $+$ finitely often (if $f(x_j)$ is false). Thus, Anke wins the play.

Now suppose Anke has a winning strategy. If there is an i such that Anke moves from $w_{i,j}$ to u_j then do the following: If x_j appears positively in the clause then let $f(x_j)$ be true else let $f(x_j)$ be false. If there is no i such that Anke moves from $w_{i,j}$ to u_j then truth value of $f(x_j)$ does not matter (and can assigned either true of false).

To see that above defines a satisfying assignment, first note that for each clause y_i, there exists a $w_{i,j}$ such that Anke moves from $w_{i,j}$ to u_j. Otherwise, Boris can first move from the start node to u_j and then to $w_{i,j}$ such that x_j appears in clause y_i; afterwards the play will go infinitely often only through a subset of the nodes of the form $v_i, w_{i,j}$ and thus the colours which appear infinitely often in the above play is a subset of $\{y, +, -\}$.

Furthermore, for no j and two nodes $w_{i,j}$ and $w_{i',j}$ such that x_j appears in y_i and $\neg x_j$ appears in $y_{i'}$, does Anke move from $w_{i,j}$ and $w_{i',j}$ to node u_j. Otherwise, Boris could win by first moving from s to u_j and then alternately going to nodes $w_{i,j}$ and $w_{i',j}$. It follows that f gives a satisfying assignment for the instance of satisfiability. $\square$

5 Multi-Dimensional Parity Games

Point [67] considered a generalisation of parity games where each node has a vector of k values and each value is a number from 1 to m. To evaluate a play, one determines for each coordinate of the vector the largest infinitely often occurring value in the play and calls the so obtained vector of k values the limit superior of the sequence of the play. The same idea has recently also been applied to mean payoff games, Rabin and Streett games as well as combinations of these games with parity games [10,16,17,19,20,77]. The winner of a play is determined as follows: If all values of the limit superior vector are odd then Anke wins the play else Boris wins the play. The approach in which the first player Anke has a conjunction and the second player Boris a disjunction of the player's winning conditions in each dimension is quite common in the field [16,19,20,77]. In this section, it is assumed that $n \geq 2$, $m \geq 2$ and $k \geq 2$.

Rabin games and Streett games are games where the winner of a play is determined by a list of pairs of sets of nodes $(V_1, W_1), (V_2, W_2), (V_3, W_3), \ldots, (V_m, W_m)$. Now, in the Rabin case, Anke wins a play iff there is an i such that the set of infinitely often visited nodes U intersects V_i and is disjoint to W_i; in the Streett case, Anke wins a play iff all i satisfy that U intersects W_i or U is disjoint to V_i.

Proposition 28 (Chatterjee, Henzinger and Piterman [17]). *One can translate k-dimensional parity games with values from $\{1, 2, 3, \ldots, m\}$ in each dimension into Streett games with $k \cdot \lceil (m-1)/2 \rceil$ pairs and Streett games with k pairs into k-dimensional parity games with values from $\{1, 2, 3\}$.*

Proof. Both directions do not change the graph of the game, they only replace the value vectors by conditions in the Streett pair and vice versa. Recall that each Streett pair is a pair (V, W) of two subsets of the set of nodes and a winning play for Anke satisfies the pair if whenever a node in V is infinitely often visited then also some node in W is infinitely often visited.

For the direction from k-dimensional parity games to Streett games, one generates for every even value $i \in \{1, 2, 3, \ldots, m\}$ and every dimension $j \in \{1, 2, 3, \ldots, k\}$ a pair (V, W) where V consists of all nodes where the j-th component of the value vector is i and W consists of all nodes where the j-th component of the value vector is strictly larger than i. Now the limit superior of the values in each dimension of the given play is odd iff the play of the game satisfies all these Streett pairs.

For the direction from a game with k Streett pairs to the k-dimensional parity game, one assigns to the h-th Streett pair (V, W) the h-th dimension where every node outside $V \cup W$ has the h-th value 1, every node in $V - W$ has the h-th value 2 and every node in W has the h-th value 3. $\square$

The following corollary is due to previously known results on Streett games like the **coNP**-completeness by Emerson and Jutla [31]; note that Chatterjee, Henzinger and Piterman [17] showed that **coNP**-hardness can even be achieved when only considering two-dimensional parity games.

Corollary 29. *If Boris has a winning strategy for a multi-dimensional parity game then he has a memoryless winning strategy. Furthermore, the problem whether Anke can win a multi-dimensional parity game is* **coNP***-complete.*

The following result provides an algorithm with runtime $O((2^{k \cdot \log(k) \cdot m} \cdot n)^{5.45})$ for multi-dimensional parity games which translates into a bound of $O((2^{k \cdot \log(k)} \cdot n)^5)$ for solving Streett games and Rabin games with n nodes and k conditions, where $k \geq 4$. For a comparison, a direct solution without translating into other games by Piterman and Pnueli [66] has the runtime $O(n^{k+1} \cdot k!)$.

Theorem 30. *The winner of a multi-dimensional parity game with k values from $\{1, 2, 3, \ldots, m\}$ per node and n nodes can be determined in time $O((2^{k \cdot \log(k) \cdot m} \cdot n)^{5.45})$. If $k \geq 4$ then the formula can be improved to $O((2^{k \cdot \log(k) \cdot m} \cdot n)^5)$.*

Proof. The algorithm is based on ideas of Point [67] and also later by Chatterjee, Henzinger and Piterman [17] who observed that the algorithm of Björklund, Sandberg and Vorobyov [5] for translating Muller games into parity games can be adjusted to translate multi-dimensional parity games into normal parity games. The idea is to use colours $c_{m',k'}$ with $m' \in \{2, 3, 4, \ldots, m\}$ and $k' \in \{1, 2, 3, \ldots, k\}$. Now, a node has a colour $c_{m',k'}$ iff its value vector $(\tilde{m}_1, \tilde{m}_2, \tilde{m}_3, \ldots, \tilde{m}_k)$ satisfies that $m' \leq \tilde{m}_{k'}$ (note that a node may have multiple colours). Note that it is not needed to use $c_{1,k'}$ as always $1 \leq \tilde{m}_{k'}$ and therefore the colour $c_{1,k'}$ would not carry any information. Now one tweaks the translation of the last appearance records in Theorem 23. Recall from the proof of Theorem 23 that the translation was realised by mapping each node v to a collection of nodes (v, r) where r is the record of colours in the order of their last appearance in prior visited nodes; those never visited can be in any order at the end of r. As every node which contains a colour $c_{m',k'}$ also contains all colours $c_{m'',k'}$ with $m'' < m'$, one can assume the tie-breaker rule that whenever $m'' < m'$ then the colour $c_{m'',k'}$ comes in the record r before the colour $c_{m',k'}$. This permits to consider and update only vectors where, for each fixed coordinate k', the colours are in their natural order. Thus one can describe the last appearance records by giving a $k \cdot m$-vector which gives, for each entry of a colour $c_{m',k'}$, only the value k', as m' is just equal to the number of k' in this record up to the position of the current entry. As a result, the overall number of last appearance records per node can be bounded by $k^{k \cdot (m-1)}$ and thus a k-dimensional parity game with each coordinate having a range from 1 to m and with n nodes can be translated into a parity game with $2^{\log(k) \cdot k \cdot (m-1)} \cdot n$ nodes and $2 \cdot k \cdot (m-1)$ values.

One computes as before from v and r the set U of current colours and then assigns to the node (v, r) in the parity game the value as follows: If U is winning for Anke then the value is $2|U|+1$ else it is $2|U|+2$, where one defines that Anke has the odd and Boris the even numbers. Note that $|U| \leq 2 \cdot k \cdot (m-1)$ and the number of values is bounded by $2 \cdot k \cdot (m-1) + 2 \leq 2 \cdot k \cdot m$. In the resulting parity game the number of values divided by the logarithm of the number of nodes is at most 2.

Thus the parity game can be solved in $O((2^{\log(k) \cdot k \cdot m} \cdot n)^{5.45})$ time and the time for computing the translation is also bounded by this term. So the same bound applies for the overall running time, as summarised in the following theorem, which makes use of the observation of Point [67]. Furthermore, if $k \geq 4$ then

$$\log(2^{\log(k) \cdot k \cdot m} \cdot n) \geq 2 \cdot k \cdot m,$$

as $\log(k) \geq 2$ and one can therefore apply the better bound $O((2^{\log(k) \cdot k \cdot m} \cdot n)^5)$ on the runtime. $\square$

Now it is shown that the result is optimal in the following sense: If one can decide multi-dimensional parity games in time $2^{o(k \cdot \log(k) \cdot m)} \cdot Poly(n)$ then the Exponential Time Hypothesis fails. An even stronger result will be shown: Consider the runtime of a solver for multi-dimensional parity games in case that one fixes either $m \geq 3$ or $k \geq 2$ (but not both). If the resulting runtime is either $2^{o(k \cdot \log(k))} \cdot Poly(n)$ (when m is constant and at least 3) or $2^{o(m)} \cdot n^{O(1)}$ (when k is constant and at least 2), then the Exponential Time Hypothesis fails. Both results are based on reducing the dominating set problem into the two decision problems. Here a dominating set of a graph is a set of nodes such that from every node in the graph there is an edge to one of the nodes in the dominating set; for this property one deviates from the usual convention of the non-existence of self-edges and assumes that every node has an edge to itself.

Theorem 31. *Assume that one can solve k-dimensional parity games with values from $\{1, 2, 3\}$ and n' nodes in time $2^{o(k \cdot \log(k))} \cdot Poly(n')$. Then there is an algorithm which solves the dominating set problem for graphs with n nodes and a target size of m for the dominating set in time $n^{o(m)}$ and thus the Exponential Time Hypothesis fails.*

Proof. Assume that one can solve the k-dimensional parity game problem as in the hypothesis. Suppose a graph H with n nodes $\{1, 2, 3, \ldots, n\}$ and a target size m of the dominating set are given. Now one chooses k to be the least even integer satisfying $k \geq 2$ and

$$m \cdot \lceil \log(n) \rceil \leq k/2 \cdot \lfloor \log(k/2) \rfloor.$$

Note that the dominating set can be described by listing the m nodes using $\lceil \log(n) \rceil$ bits each. The idea now is to represent the $m \cdot \lceil \log(n) \rceil$ bits to describe the dominating set by a sequence of $k/2$ numbers $a_1, a_2, a_3, \ldots, a_{k/2}$ from $\{1, 2, 3, \ldots, k\}$ with the additional requirement that a_i is among the first $k/2$ members of $\{1, 2, 3, \ldots, k\} - \{a_j : j < i\}$ for all i. The statement "choice (j, r) is consistent with $(w, \tilde{m})$" means the following condition: the binary representations $d_1 d_2 d_3 \ldots d_{\lfloor \log(k/2) \rfloor}$ of $(r-1)$ and $w_1 w_2 w_3 \ldots w_{\lceil \log(n) \rceil}$ of w satisfy that for all i, h with $1 \leq i \leq \log(k/2)$ and $1 \leq h \leq \lceil \log(n) \rceil$, if $(j-1) \cdot \lfloor \log(k/2) \rfloor + i = (\tilde{m} - 1) \cdot \lceil \log(n) \rceil + h$ then $d_i = w_h$.

The game goes in rounds and the game graph will then be constructed accordingly. Boris has in mind a dominating set and Anke tries to check out on Boris' answers in order to make sure that the set in mind is correct.

The game goes infinitely often through the following rounds where in each round the game goes through steps 1., 2. and then a finite number of repetitions of steps 3., 4. where the number of repetitions is bounded by $k/2$, followed by step 5. which takes the game back to step 1.

The following descriptions of a round also give the nodes which are in the game, along with edges, values of the nodes and the players to move. All the nodes, except the nodes of the form $(0, b, B)$ described in step 5, have value vector $(1, 1, 1, \ldots, 1)$. Below B is always a subset of $\{1, 2, 3, \ldots, k\}$, $a_1, a_2, a_3, \ldots, a_{k/2} \in \{1, 2, 3, \ldots, k\}$ and v, w are vertices of H. Intuitively, B gives the choices $a_1, a_2, \ldots$, used by Boris, to describe the dominating set as mentioned above — here the ordering of members of B is based on the order they entered the set B in the play.

1. In each round, the game starts in a node called (0). There are edges from node (0) to nodes (v), for each vertex v in H.
   Thus, at node (0) Anke chooses a node v of the graph, for which it is asking Boris to give a neighbour from the dominating set, and moves to node (v).
2. The nodes (v), for vertices v in H, have edges to nodes of the form $(\tilde{m}, w, a_i, B)$, where $i = 1$, $B = \emptyset$, w is a neighbour of v in H, $1 \leq \tilde{m} \leq m$ and the choice $(1, a_1)$ is consistent with $(w, \tilde{m})$ (note that a_1 is a_1-th member of $\{1, 2, 3, \ldots, k\}$). Boris moves in the nodes (v), for v being a vertex in H.
   Intuitively, the intention of Boris moving from (v) to $(\tilde{m}, w, a_i, B)$ with $i = 1$, $B = \emptyset$ and w being a neighbour of v, is that w is the $\tilde{m}$-th vertex in the dominating set chosen by Boris.
3. For $\tilde{m} \in \{1, 2, 3, \ldots, m\}$, w a vertex of H, $a_i \in \{1, 2, 3, \ldots, k\} - B$ and the cardinality of B being less than $k/2$, there exists a node $(\tilde{m}, w, a_i, B)$. The node $(\tilde{m}, w, a_i, B)$ with $a_i \notin B$, has edges to $(\tilde{m}, w, a_i, B \cup \{a_i\})$ and to $(0, b, B \cup \{a_i\})$, where $b \in \{1, 2, 3, \ldots, k\} - (B \cup \{a_i\})$. Anke moves in nodes of the form $(\tilde{m}, w, a_i, B)$, with $a_i \notin B$.
   Intuitively, Anke can from $(\tilde{m}, w, a_i, B)$, where $a_i \notin B$, either move to $(\tilde{m}, w, a_i, B \cup \{a_i\})$ and indicate that Boris should reveal more information (only possible when $|B \cup \{a_i\}| < k/2$) or move to a node $(0, b, B \cup \{a_i\})$ where $b \in \{1, 2, 3, \ldots, k\} - \{a_i\} - B$, which indicates visiting a node with certain value (see item 5 below).
4. For $\tilde{m} \in \{1, 2, 3, \ldots, m\}$, w a vertex of H, $a_{i-1} \in B$ and the cardinality of B being less than $k/2$, there is a node $(\tilde{m}, w, a_{i-1}, B)$ and this has edges to nodes of the form $(\tilde{m}, w, a_i, B)$, where $a_i \notin B$ and the choice (i, r) is consistent with $(w, \tilde{m})$, where a_i is the r-th member of $\{1, 2, 3, \ldots, k\} - B$. Boris moves in nodes of the form $(\tilde{m}, w, a_{i-1}, B)$ with $a_{i-1} \in B$.
   Intuitively, Boris has to select a_i and move to $(\tilde{m}, w, a_i, B)$ where $a_i \notin B$; at that node it is then Anke's turn to move as described in Step 3.
5. There are nodes of the form $(0, b, B)$ with $B \subset \{1, 2, 3, \ldots, k\}$ and $b \in \{1, 2, 3, \ldots, k\} - B$. There is exactly one edge from such a node and it goes to (0). Boris moves in the nodes of the form $(0, b, B)$.
   The nodes $(0, b, B)$ are the only nodes with a value-vector different from $(1, 1, 1, \ldots, 1)$. Here the value vector $(m_1, m_2, m_3, \ldots, m_k)$ of a node $(0, b, B)$ is defined by the equation

$$m_h = \begin{cases} 1 & \text{if } h \notin B \cup \{b\}, \\ 2 & \text{if } h = b, \\ 3 & \text{if } h \in B. \end{cases}$$

   Intuitively, Boris moves from this node to (0) and the next round of the game starts in Step 1.

In the case that there is a dominating set of size m, Boris can choose in the game always nodes $(\ldots, B)$ such that the sets B of the form $\{a_j : j < i\}$ occurring there are ordered under inclusion and these sets can be computed from a fixed sequence $a_1, a_2, a_3, \ldots, a_{k/2}$ derived from a binary representation describing the dominating set. In a play, whenever it is turn for Boris to move, the sets B in the last component of the names of the nodes would be derived using $a_1, a_2, \ldots, a_{k/2}$ as above. Thus, in any particular play there is a largest set B such that nodes of the form $(\cdot, \cdot, \cdot, B)$ are visited infinitely often in the play, and all other sets B', with node $(\cdot, \cdot, \cdot, B')$ occurring in the play, satisfy $B' \subseteq B$. Thus for this largest set B, player Anke has to choose b, when going to node $(0, b, B)$, to be non-member of B and so the vectors $(m_1, m_2, m_3, \ldots, m_k)$ when moving to $(0, b, B)$ will have that $m_b = 2$ and $m_h = 3$ for all $h \in B$; furthermore, m_b will never be 3. It follows that Anke cannot satisfy the condition that the limit superior of each m_h over the play is odd and thus Boris is winning the game.

In the case that there is no dominating set of size m, Boris cannot achieve that all the sets B occurring in nodes of the form $(\ldots, B)$ are comparable. To see this, one can assume without loss of generality that the strategy of Boris is fixed, that Anke knows the strategy and Anke exploits its weakness. Now, as there is no dominating set of size m, Boris has selected two different nodes $w, \tilde{w}$ at the same position $\tilde{m}$, when Anke asks for the node in the dominating set that are neighbours of suitable nodes v and $\tilde{v}$. As w, $\tilde{w}$ get coded into different witnesses $(a_1, a_2, a_3, \ldots, a_{k/2})$ and $(\tilde{a}_1, \tilde{a}_2, \tilde{a}_3, \ldots, \tilde{a}_{k/2})$, there is a first i where $a_i \neq \tilde{a}_i$. Thus Anke can go alternately from (0) to (v) and $(\tilde{v})$ and then run through the cycles of building up the witnesses until she reaches the node $(\tilde{m}, w, a_i, B)$ and $(\tilde{m}, \tilde{w}, \tilde{a}_i, B)$, respectively, where $B = \{a_j : j < i\} = \{\tilde{a}_j : j < i\}$. From these nodes, Anke goes to $(0, \tilde{a}_i, B \cup \{a_i\})$ and $(0, a_i, B \cup \{\tilde{a}_i\})$, respectively and the game returns from them to (0). Thus the limit superior $(m_1, m_2, m_3, \ldots, m_k)$ of the value vectors of the play will satisfy that $m_h = 3$ for all $h \in B \cup \{a_i, \tilde{a}_i\}$ and $m_h = 1$ for all $h \notin B \cup \{a_i, \tilde{a}_i\}$. So the m_h are odd for all $h \in \{1, 2, 3, \ldots, k\}$ and Anke wins the game.

In summary, Boris can win the so constructed multi-dimensional parity game iff the given graph has a dominating set of size m.

One can bound the number n' of nodes in this game by the formula $1 + n + m \cdot n \cdot k \cdot 2^k + k \cdot 2^k \leq 4n^2 k 2^k$, as $m \leq n$. Thus, $2^{o(k \log(k))} poly(n')$ is in $2^{o(k \log(k))} poly(n)$, which in turn is in $2^{o(m \log(n))} poly(n)$ and thus in $n^{o(m)}$. Thus, if there is an algorithm which solves k-dimensional parity games with n' nodes in time $2^{o(k \cdot \log(k))} \cdot Poly(n')$, then one can solve the dominating set problem in time $n^{o(m)}$.

Now one can use the following result of Chen, Huang, Kanj and Xia [21, Theorem 5.8]: If one can solve the problem whether a graph of n nodes has a dominating set of size m in time $n^{o(m)}$ then the Exponential Time Hypothesis fails. This connection then translates into the following bound: If the k-dimensional parity games with n' nodes and values from $\{1, 2, 3\}$ can, uniformly in n', k, be decided in time $2^{o(k \cdot \log(k))} \cdot Poly(n')$ then the Exponential Time Hypothesis fails. $\square$

Also the second result is again a translation of the dominating set problem. One needs dimension two and the main technique is to compare the bits in the witnesses for a dominating set. Note that dimension one is equivalent to the normal parity games, thus requiring dimension two is unavoidable.

Theorem 32. *Given a graph H with n nodes and a number m with the constraint that $2 \leq m \leq n$, one can compute in time polynomial in n a two-dimensional parity game with n' nodes and m' colours such that the following conditions hold:*

- $m' = 2m \cdot \lceil \log(n) \rceil$,

- *$n' = 1 + (m+1) \cdot n + 2m \cdot \lceil \log(n) \rceil$ and*
- *the given graph H has a dominating set of size up to m iff player Boris has a winning strategy in the resulting two-dimensional parity game.*

Furthermore, the so obtained two-dimensional parity games cannot be solved in time $2^{o(m')} \cdot Poly(n')$, provided that the Exponential Time Hypothesis holds.

Proof. Consider the nodes of the graph H and let them have as names the first n strings from $\{0,1\}^{\lceil \log n \rceil}$. The proof is similar to the proof of Theorem 31 except that the graph construction and the checking of consistency of dominating set is modified to have a constant bound on the dimension rather than on the number of values. The basic idea of the game is to go through following rounds:

1. Anke selects a vertex v in the graph H.
2. Boris selects a neighbouring vertex w of v in the graph H and a number $\tilde{m}$, to indicate that w is the $\tilde{m}$-th member of the dominating set.
3. Anke selects a bit-position $o \in \{1,2,3,\ldots,\lceil \log n \rceil\}$; if the o-th bit of the name of w is 1 then Anke moves in the game to a node with value $(2(\tilde{m}-1) \cdot \lceil \log(n) \rceil + 2o - 1, 2(\tilde{m}-1) \cdot \lceil \log(n) \rceil + 2o)$ else Anke moves to a node with value $(2(\tilde{m}-1) \cdot \lceil \log(n) \rceil + 2o, 2(\tilde{m}-1) \cdot \lceil \log(n) \rceil + 2o - 1)$.
4. Boris moves back to the start of the game, where Anke selects a node in the graph H.

The values of all nodes except as at Step 3. above in the game will be "small". In case that there is a dominating set of size m, Boris can play a memoryless winning strategy for the game by always selecting the right node in the second step — this will ensure that the limit superior of the values in the two dimensions are of different parity. In case there is no dominating set, when playing memoryless, Boris has to be inconsistent and choose for two different vertices v, v' chosen by Anke in Step 1. above, two different vertices w, w' at the same position $\tilde{m}$ of the candidate for the dominating set. These w, w' will differ in some bit position o; thus Anke can then force the game to go through the nodes with value $(2(\tilde{m}-1) \cdot \lceil \log(n) \rceil + 2o, 2(\tilde{m}-1) \cdot \lceil \log(n) \rceil + 2o - 1)$ and $(2(\tilde{m}-1) \cdot \lceil \log(n) \rceil + 2o - 1, 2(\tilde{m}-1) \cdot \lceil \log(n) \rceil + 2o)$ infinitely often to win the game.

Based on the above motivation, the nodes and edges of the game is described as follows. Note that 0 is not a name of any vertex in H.

1. The node $(0,0)$ has the value $(1,1)$. The node $(0,0)$ is the starting node and Anke moves in this node. There is an edge from $(0,0)$ to $(v,0)$, for all vertices v in H.
2. Nodes $(v,0)$, for v being a vertex in H. Value of these nodes are $(1,1)$. Boris moves in these nodes. For any w such that (v,w) is an edge in H, there is an edge from $(v,0)$ to $(w,\tilde{m})$ for $\tilde{m}$ with $1 \leq \tilde{m} \leq m$.
   Intuitively, a move from $(v,0)$ to $(w,\tilde{m})$ denotes that Boris is specifying the neighbour w of v as being the $\tilde{m}$-th element of the dominating set chosen by it.
3. There are nodes $(w,\tilde{m})$ for $w,\tilde{m}$ with w being a vertex in H and $1 \leq \tilde{m} \leq m$; the values of these nodes are $(1,1)$ and Anke moves in these nodes.
   For each $o \in \{1,2,3,\ldots,\lceil \log n \rceil\}$, there is an edge from $(w,\tilde{m})$ to node $(0, 2(\tilde{m}-1) \cdot \lceil \log(n) \rceil + 2o - b)$, where b is the o-th bit of w, that is $b = b_o$ where $w = b_1 b_2 b_3 \ldots b_{\lceil \log n \rceil}$.

Intuitively, Anke chooses o to ask Boris to prove that the o-th bit of $\tilde{m}$-th vertex in the dominating set is always consistent.

4. There are nodes $(0, h)$ for all $h \in \{1, 2, 3, \ldots 2m\lceil\log n\rceil\}$. The value of the node $(0, h)$ is $(h, h-1)$ when h is even and its value is $(h, h+1)$ when h is odd. Boris moves in these nodes. There is an edge from $(0, h)$ to $(0, 0)$.

In case there is a dominating set $\{w_1, w_2, w_3, \ldots, w_m\}$, Boris moves in Step 2. above always from a node $(v, 0)$ to a node $(w_{\tilde{m}}, \tilde{m})$ such that there is an edge in H from v to $w_{\tilde{m}}$. This is a winning strategy, as then for all positions o in a $w_{\tilde{m}}$, as chosen by Anke in Step 3. above, the bit b is always the same, and thus the limit superior of the values attained in a play is of the form $(2(\tilde{m}-1) \cdot \lceil\log(n)\rceil + 2o - b, 2(\tilde{m}-1) \cdot \lceil\log(n)\rceil + 2o + b - 1)$, for some $\tilde{m}$ and o with b being the o-th bit of $w_{\tilde{m}}$.

If there is no dominating set of size m and Boris plays a memoryless winning strategy, then he will on two nodes $(0, v)$ and $(0, v')$ move to two different nodes $(w, \tilde{m})$ and $(w', \tilde{m})$, as otherwise Boris would have a consistent dominating set contradicting the assumption. Now there is a position o such that the bits b and b' of w and w' at this position differ. Therefore Anke can move to nodes with value $(2(\tilde{m}-1) \cdot \lceil\log(n)\rceil + 2o - b, 2(\tilde{m}-1) \cdot \lceil\log(n)\rceil + 2o + b - 1)$ and $(2(\tilde{m}-1) \cdot \lceil\log(n)\rceil + 2o - b', 2(\tilde{m}-1) \cdot \lceil\log(n)\rceil + 2o + b' - 1)$ which are of the form $(h-1, h)$ and $(h, h-1)$ for some even h. That is, by alternating moving to the nodes $(0, v)$ and $(0, v')$ when in node $(0, 0)$, and by moving to the node $(0, 2(\tilde{m}-1) \cdot \lceil\log n\rceil + 2o - b)$ when in node $(w, \tilde{m})$, where b is the o-th bit of w, Anke will achieve that the limit superior of a play is (h, h) for some even h and therefore the game is won by Anke. It follows that Boris' memoryless strategy is not a winning strategy and therefore he does not have a winning strategy at all. In summary, Boris wins the two-dimensional parity game iff there is a dominating set of size m in H.

The number n' of nodes is the sum of 1 (for node $(0, 0)$) and n (for nodes $(v, 0)$ with v being a vertex in H) and $n \cdot m$ (for nodes $(w, \tilde{m})$ with w being a vertex in H and $\tilde{m} \in \{1, 2, 3, \ldots, m\}$) and $2m \cdot \lceil\log(n)\rceil$ (for nodes $(0, h)$). The number m' is just $2m \cdot \lceil\log(n)\rceil$, as h is bounded by $2m\lceil\log n\rceil$.

Now assume that there would be an algorithm for this problem which runs in time $2^{o(m')} \cdot Poly(n')$. Let $f(m')$ be a function in $o(m')$ such that the runtime is in $2^{f(m')} \cdot Poly(n')$. Now, one can replace $f(m')$ by $g(m') \cdot m'$ where $g(m') = \sup\{f(m'')/m'' : m'' \geq m'\}$, note that g is monotonically decreasing and $o(1)$. As g is $o(1)$, one can also obtain that

$$2^{f(m \cdot \lceil\log(n)\rceil)} \leq 2^{g(m) \cdot m \cdot \lceil\log(n)\rceil} = n^{o(m)}.$$

As $n' \leq n^2 \cdot \lceil\log(n)\rceil$, one can conclude that the runtime for finding a solution to the existence of a dominating set is $n^{o(m)} \cdot Poly(n)$ which is $n^{o(m)}$. However, Chen, Huang, Kanj and Xia [21, Theorem 5.8] showed that under these hypotheses, the Exponential Time Hypothesis fails. This completes the proof. □

Recall that the question whether a problem is in **FPT** depends on which parameters are considered as constants and which are running parameters. The dependence of the algorithm run time on the constant parameters can be arbitrary but that on the running parameters has to be a polynomial of fixed degree which is independent on the constant parameters. Theorem 30 shows that if one fixes both parameters m and k as constants then multi-dimensional parity games are in **FPT**. Theorems 31 and 32 show that, unless the Exponential Time Hypothesis is wrong, multi-dimensional parity games are not fixed parameter tractable in the case that only one of the parameters m and k is fixed as a constant. Bruyére, Hautem and Raskin [10] investigate the fixed-parameter tractability of generalisations of multi-dimensional parity games and related games in detail.

There is some connection between parity games and mean payoff games; for the latter, Velner, Chatterjee, Doyen, Henzinger, Rabinovich and Raskin [77] studied the computational complexity of the multi-dimensional analogue of mean payoff games and discovered that one has to distinguish the cases of evaluation by limit superior and evaluation by limit inferior in the multi-dimensional game. For the case of evaluation by limit superior, they are in $\mathbf{NP} \cap \mathbf{coNP}$; for the case of evaluation by limit inferior, they are **coNP**-complete. In the light of the above result, multi-dimensional parity games are more related to the evaluation of limit inferior.

6 Conclusion

The progress reported in this paper shows that solving parity games is not as difficult as it was widely believed. Indeed, parity games can be solved in quasipolynomial time – the previous bounds were roughly $n^{O(\sqrt{n})}$ – and they are fixed parameter tractable with respect to the number m of values (aka colours or priorities) – the previously known algorithms were roughly $O(n^{m/3})$. These results are in agreement with earlier results stating that parity games can be solved in $\mathbf{UP} \cap \mathbf{coUP}$ [52] and that there are subexponential algorithms to solve the problem [55].

In spite of the current progress, the original question, as asked by Emerson and Jutla [32] in 1991 and others, whether parity games can be decided in polynomial time still remains an important open question.

The above results on parity games are then used to give an algorithm of runtime $O((m^m \cdot n)^5)$ for coloured Muller games with n nodes and m colours; this upper bound is almost optimal, since an algorithm with runtime $O((2^m \cdot n)^c)$, for some constant c, only exists in the case that the Exponential Time Hypothesis fails.

One might ask whether the results obtained for parity games permit further transfers to Muller games, for example, in the special cases where (a) player Anke can employ a memoryless winning strategy due to the special type of the game or (b) one does not permit player Anke to use other strategies than memoryless ones. Note that case (b) differs from case (a), as in case (b) the condition on using memoryless strategies can be restrictive while case (a) applies to Muller games where one knows that "if Anke has a winning strategy then she has a memoryless winning strategy". Case (a) was analysed by Emerson [30], McNaughton [61] and Zielonka [80]; it applies to Muller games where the winning condition of player Boris is closed under union [30,80].

The above mentioned lower bound directly also applies to case (a). For case (b), the complexity class of the general problem is also in the polynomial hierarchy but not **PSPACE**-complete (unless $\mathbf{PSPACE} = \mathbf{\Sigma}_2^P$) as the decision problem for coloured Muller games; however, the algorithmic bounds are much worse, as one can code **NP**-hard problems into instances with four colours.

Another variant of parity games is to consider vectors of values where in the default case player Anke wins if the limit superior of all of each of these values is odd and player Boris wins if the limit superior of one of the values is even. For this type of game, the k-dimensional parity game with values from 1 to m and n nodes can be decided in time $O((2^{k \cdot \log(k) \cdot m} \cdot n)^{5.45})$ and slight improvements of the exponent 5.45 might be possible. However, really better algorithms, even for the special case where either k or m is constant, would imply that the Exponential Time Hypothesis fails, which seems unlikely. More precisely, under the assumption that the Exponential Time Hypothesis is true, there are no algorithms which solve k-dimensional parity games with m values and n nodes in time $2^{o(k \cdot \log(k) \cdot m)} \cdot n^{O(1)}$ and this even holds when either m is fixed to be a constant at least 3 or k is fixed to be a constant which is at least 2, but not both are fixed. This shows that the multi-dimensional parity games are very similar to coloured Muller games with respect to the runtime behaviour of algorithms to solve them.

Acknowledgements. The authors would like to thank Krishnendu Chatterjee, Sasha Rubin, Sven Schewe and Moshe Vardi for correspondence and comments. Further thanks go to the referees of the STOC 2017 paper and its journal version for numerous suggestions. A referee of the journal version as well as Rod Downey pointed out that the lower bound for Muller games is more suitably be done with reference to the Exponential Time Hypothesis and that there was a flaw (miscitation) in the previous lower bound result of the conference version at STOC 2017.

References

1. Catriel Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Transactions on Database Systems*, 5:241–259, 1980.
2. Mihir Bellare and Shafi Goldwasser. The complexity of decision versus search. *SIAM Journal on Computing*, 23(1):97–119, 1994.
3. Julien Bernet, David Janin and Igor Walukiewicz. Permissive strategies: from parity games to safety games. *RAIRO - Theoretical Informatics and Applications*, EDP Sciences, 36:251–275, 2002.
4. Dietmar Berwanger and Erich Grädel. Fixed-point logics and solitaire games. *Theory of Computing Systems*, 37(6):675–694, 2004.
5. Henrik Björklund, Sven Sandberg and Sergei Vorobyov. *On fixed-parameter complexity of infinite games.* Technical report 2003-038, Department of Information Technology, Uppsala University, Box 337, SE-751 05 Uppsala, Sweden.
6. Henrik Björklund, Sven Sandberg and Sergei Vorobyov. Memoryless determinacy of parity and mean payoff games: a simple proof. *Theoretical Computer Science*, 310(1–3):365–378, 2004.
7. Hans L. Bodlaender, Michael J. Dinneen and Bakhadyr Khoussainov. On game-theoretic models of networks. *Algorithms and Computation*, Twelfth International Symposium, ISAAC 2001, Christchurch, New Zealand, December 2001, Proceedings. *Springer LNCS*, 2223:550–561, 2001.
8. Hans L. Bodlaender, Michael J. Dinneen and Bakhadyr Khoussainov. Relaxed Update and Partition Network Games. *Fundamenta Informaticae*, 49(4):301–312, 2002.

9. Anca Browne, Edmund M. Clarke, Somesh Jha, David E. Long and Wilfredo R. Marrero. An improved algorithm for the evaluation of fixpoint expressions. *Theoretical Computer Science*, 178(1–2):237–255, 1997.
10. Véronique Bruyère, Quentin Hautem and Jean-François Raskin. Games with lexicographically ordered ω-regular objectives. Technical report on `http://arxiv.org/abs/1707.05968`, 2017.
11. Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li and Frank Stephan. Deciding parity games in quasipolynomial time. *STOC 2017 Theory Fest: Forty Ninth Annual ACM Symposium on the Theory of Computing*, 19–23 June 2017, Proceedings, ACM, Montreal, Canada, 12 pages, 2017.
12. Felix Canavoi, Erich Grädel and Roman Rabinovich. The discrete strategy improvement algorithm for parity games and complexity measures for directed graphs. *Theoretical Computer Science*, 560:235–250, 2014.
13. Ashok K. Chandra, Dexter C. Kozen and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.
14. Krishnendu Chatterjee. *Comments on the Quasipolynomial Time Algorithm.* Private communication, 2017.
15. Krishnendu Chatterjee and Thomas A. Henzinger. Strategy Improvement and Randomized Subexponential Algorithms for Stochastic Parity Games. *Twenty Third Annual Symposium on Theoretical Aspects of Computer Science*, STACS 2006, Marseille, France, 23–25 February 2006, Proceedings. *Springer LNCS*, 3885:512–523, 2006.
16. Krishnendu Chatterjee, Thomas A. Henzinger and Marcin Jurdziński. Mean-payoff parity games. *Twentieth Annual IEEE Symposium on Logic in Computer Science*, LICS 2005, Proceedings, 178–187, 2005.
17. Krishnendu Chatterjee, Thomas A. Henzinger and Nir Piterman. Generalized parity games. *Foundations of Software Science and Computational Structures*, Tenth International Conference, FOSSACS 2007, *Springer LNCS*, 4423:153–167, 2007.
18. Krishnendu Chatterjee, Marcin Jurdzinski and Thomas A. Henzinger. Quantitative stochastic parity games. SODA 2004, *Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 121–130, 2004.
19. Krishnendu Chatterjee, Mickael Randour and Jean-François Raskin. Strategy synthesis for multi-dimensional quantitative objectives. *Acta Informatica*, 51:129–163, 2014.
20. Krishnendu Chatterjee and Yaron Velner. Hyperplane separation technique for multidimensional mean-payoff games. *Concurrency Theory* – Twenty-Fourth International Conference, CONCUR 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings. *Springer LNCS*, 8052:500–515, 2013.
21. Jianer Chen, Xiuzhen Huang, Iyad A. Kanj and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346–1367, 2006.
22. Stephen A. Cook. Path systems and language recognition. *Proceedings of the Second Annual ACM Symposium on Theory of Computing*, STOC 1970, 4–6 May 1970, Northampton, Massachusetts, USA, pages 70–72, 1970.

23. Sephen A. Cook. The complexity of theorem proving procedures. *Proceedings of the Third Annual ACM Symposium on the Theory of Computing*, STOC 1971, 3–5 May 1971, Shaker Heights, Ohio, USA, pages 151–158, 1971.
24. Anuj Dawar, Florian Horn and Paul Hunter. *Complexity Bounds for Muller Games.* Manuscript, 2011.
25. Antonio Di Stasio, Aniello Murano, Giuseppe Perelli and Moshe Y. Vardi. Solving parity games using an automata-based algorithm. *Twenty first International Conference on Implementation and Application of Automata*, CIAA 2016, 19–22 July 2016, Seoul, South Korea, *Springer LNCS*, 9705:64–76, 2016.
26. Christoph Dittmann, Stephan Kreutzer and Alexandru I. Tomescu. Graph operations on parity games and polynomial-time algorithms. *Theoretical Computer Science*, 614: 97–108, 2016.
27. Rodney G. Downey and Michael R. Fellows. *Parameterised Complexity.* Springer, Heidelberg, 1999.
28. Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity Theory.* Springer, Heidelberg, 2013.
29. Stefan Dziembowski, Marcin Jurdziński and Igor Walukiewicz. How much memory is needed to win infinite games? *Twelfth Annual IEEE Symposium on Logic in Computer Science*, LICS 1997, Proceedings, IEEE, pages 99–110, 1997.
30. E. Allen Emerson. Automata, tableaux, and temporal logics. *Proceedings of the Workshop on Logic of Programs*, *Springer LNCS*, 193:79–88, 1985.
31. E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs. *Annals of IEEE Symposium on Foundations of Computer Science*, pages 328–337, 1988.
32. E. Allen Emerson and Charanjit S. Jutla. Tree automata, μ-calculus and determinacy. *Annals of IEEE Symposium on Foundations of Computer Science*, pages 368–377, 1991.
33. E. Allen Emerson, Charanjit S. Jutla, A. Prasad Sistla. On model checking for the μ-calculus and its fragments. *Theoretical Computer Science*, 258(1-2):491–522, 2001.
34. John Fearnley, Sanjay Jain, Sven Schewe, Frank Stephan and Dominik Wojtczak. An ordered approach to solving parity games in quasi polynomial time and quasi linear space. *Twenty-fourth International SPIN Symposium on Model Checking of Software*, SPIN 2017. See also the technical report on `http://arxiv.org/abs/1703.01296`, 2017.
35. Olivier Finkel and Stevo Todorčević. The isomorphism relation between tree-automatic structures. *Central European Journal of Mathematics*, 8(2):299–313, 2010.
36. Olivier Finkel and Stevo Todorčević. A hierarchy of tree-automatic structures. *The Journal of Symbolic Logic*, 77(1):350–368, 2012.
37. Jörg Flum and Martin Grohe. *Parameterized Complexity Theory.* Springer, 2006.
38. Oliver Friedmann. An exponential lower bound for the parity game strategy improvement algorithm as we know it. *Logic in Computer Science*, LICS 2009, pages 145–156, 2009.
39. Oliver Friedmann and Martin Lange. Solving parity games in practice. *Automated Technology for Verification and Analysis*, Seventh International Symposium, ATVA 2009, Macao, China, 14–16 October 2009, *Springer LNCS*, 5799:182–196, 2009.

40. Jakub Gajarský, Michael Lampis, Kazuhisa Makino, Valia Mitsou and Sebastian Ordyniak. Parameterized algorithms for parity games. *Mathematical Foundations of Computer Science*, MFCS 2015. *Springer LNCS*, 9235:336–347, 2015.
41. Aniruddh Gandhi, Bakhadyr Khoussainov and Jiamou Liu. Efficient algorithms for games played on trees with back-edges. *Fundamenta Informaticae*, 111(4):391–412, 2011.
42. Hugo Gimbert and Rasmus Ibsen-Jensen. A short proof of correctness of the quasi-polynomial time algorithm for parity games. Technical report on `http://arxiv.org/abs/1702.01953`, 2017.
43. Erich Grädel, Phokion G. Kolaitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y. Vardi, Yde Venema and Scott Weinstein. *Finite Model Theory and Its Applications*. Springer, 2007.
44. Andrey Grinshpun, Pakawat Phalitnonkiat, Sasha Rubin and Andrei Tarfulea. Alternating traps in Muller and parity games. *Theoretical Computer Science*, 521:73–91, 2014.
45. Yuri Gurevich and Leo Harrington. Trees, automata and games. *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC 1982, May 5–7, 1982, San Francisco, California, USA, pages 60-65, 1982.
46. Florian Horn. Dicing on the Streett. *Information Processing Letters*, 104(1):1–9, 2007.
47. Florian Horn. Explicit Muller games are **PTIME**. *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, FSTTCS 2008, pages 235–245, *Dagstuhl Technical Reports*, 1756, 2008.
48. Paul William Hunter. *Complexity and Infinite Games on Finite Graphs.* PhD Thesis, University of Cambridge, Computer Laboratory Hughes Hall, 2007.
49. Paul Wiliam Hunter and Anuj Dawar. Complexity bounds for regular games. *Proceedings of the Thirtieth International Symposium on Mathematical Foundations of Computer Science*, MFCS 2005, *Springer LNCS*, 3618:495–506, 2005.
50. Neil Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22(3):384–406, 1981.
51. Hajime Ishihara, Bakhadyr Khoussainov. Complexity of some infinite games played on finite graphs. *Graph-Theoretic Concepts in Computer Science*, Twenty-Eighth International Workshop, WG 2002, Cesky Krumlov, Czech Republic, 13–15 June 2002, Proceedings. *Springer LNCS*, 2573:270–281, 2002.
52. Marcin Jurdzinski. Deciding the winner in parity games is in $\mathbf{UP} \cap \mathbf{coUP}$. *Information Processing Letters*, 68(3):119–124, 1998.
53. Marcin Jurdziński. Small progress measures for solving parity games. *STACS 2000, Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science*, *Springer LNCS*, 1770:290–301, 2000.
54. Marcin Jurdziński and Ranko Lazić. Succinct progress measures for solving parity games. *Logic in Computer Science*, LICS 2017. Technical report on `http://arxiv.org/abs/1702.05051`, 2017.
55. Marcin Jurdziński, Mike Paterson and Uri Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM Journal on Computing*, 38(4):1519–1532, 2008.
56. Imran Khaliq and Gulshad Imran. Reachability games revisited. *Second International Conference on Advances and Trends in Software Engineering*, SOFTENG 2016, 21–25 February

2016, Lisbon, Portugal, Proceedings, International Academy, Research and Industry Association (IARIA), Wellington, DE 19810, USA, pages 129–133, 2016.
57. Bakhadyr Khoussainov and Anil Nerode. *Automata Theory and its Applications.* Birkhäuser, 2001.
58. Dietrich Kuske, Jiamou Liu and Markus Lohrey. The isomorphism problem for omega-automatic trees. Proceedings of *Computer Science Logic*, CSL 2010, *Springer LNCS*, 6247:396–410, 2010.
59. Karoliina Lehtinen. A modal μ perspective on solving parity games in quasi-polynomial time. *Thirtythird Conference on Logic in Computer Science*, LICS 2018, Oxford, UK, 9–12 July 2018, pages 639–648, 2018.
60. Leonid A. Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 3(9):115–116, 1973.
61. Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.
62. Andrzej Wlodzimierz Mostowski. Games with forbidden positions. Technical Report 78, Uniwersytet Gdanski, Instytut Matematyki, 1991.
63. Jan Obdrzalek. Algorithmic analysis of parity games. PhD thesis, University of Edinburgh, 2006.
64. Viktor Petersson and Sergei G. Vorobyov. A randomized subexponential algorithm for parity games. *Nordic Journal of Computing*, 8:324–345, 2001.
65. Nir Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3:5):1–21, 2007.
66. Nir Piterman and Amir Pnueli. Faster solutions of Rabin and Streett games. *Twenty First IEEE Symposium on Logic in Computer Science*, LICS 2006, 12–15 August 2006, Seattle, WA, USA, Proceedings. IEEE Computer Society, pages 528–539, 2006.
67. Gérald Point. *The Synthesis Toolbox – From modal automata to controller synthesis.* Research Report 1342–05, LaBRI – UMR CNRS 5800, 2005.
68. Shmuel Safra. On the complexity of ω-automata. Proceedings twenty-ninth IEEE Symposium on Foundations of Computer Science, pages 319-327, 1988.
69. Shmuel Safra. Exponential determinization for omega-Automata with a strong fairness acceptance condition. *SIAM Journal on Computing*, 36(3):803–814, 2006.
70. Sven Schewe. Solving parity games in big steps. *FCTTCS 2007: Foundations of Software Technology and Theoretical Computer Science*, *Springer LNCS*, 4855:449–460, 2007; *Journal of Computer and System Sciences*, available online, 2016.
71. Sven Schewe. From parity and payoff games to linear programming. *Mathematical Foundations of Computer Science 2009*, Thirty-Fourth International Symposium, MFCS 2009, Novy Smokovec, High Tatras, Slovakia, August 24-28, 2009. Proceedings. *Springer LNCS*, 5734:675–686, 2009.
72. Sven Schewe. Solving parity games in big steps. *Journal of Computer and System Sciences*, 84:243—262, 2017.
73. Helmut Seidl. Fast and simple nested fixpoints. *Information Processing Letters*, 59:303–308, 1996.

74. Frank Stephan. *Methods and Theory of Automata and Languages.* Lecture Notes, School of Computing, National University of Singapore, 2016.
`http://www.comp.nus.edu.sg/~fstephan/fullautomatatheory-nov2016.ps`.
75. Colin Stirling. Bisimulation, modal logic and model checking games. *Logic Journal of IGPL*, 7(1):103–124, 1999.
76. Wolfgang Thomas. On the Synthesis of Strategies in Infinite Games. *Twelfth International Symposium on Theoretical Aspects of Computer Science*, STACS 1995, *Springer LNCS*, 900:1–13, 1995.
77. Yaron Velner, Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, Alexander Rabinovich and Jean-François Raskin. The complexity of multi-mean-payoff and multi-energy games. *Information and Computation*, 241:177–196, 2015.
78. Igor Walukiewicz. Pushdown processes: games and model-checking. *Information and Computation*, 36(3):261–275, 2001.
79. Thomas Wilkie. Alternating tree automata, parity games and modal μ-calculus. *Bulletin of the Belgian Mathematical Society*, 8(2):359–391, 2001.
80. Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200:135–183, 1998.