A modal $\mu$ perspective on solving parity games in quasi-polynomial time*

Karoliina Lehtinen
Kiel University
Kiel, Germany
kleh@informatik.uni-kiel.de

Abstract
We present a new quasi-polynomial algorithm for solving parity games. It is based on a new bisimulation invariant measure of complexity for parity games, called the register-index, which captures the complexity of the priority assignment. For fixed parameter $k$, the class of games with register-index bounded by $k$ is solvable in polynomial time.

We show that the register-index of parity games of size $n$ is bounded by $O(\log n)$ and derive a quasi-polynomial algorithm. Finally, we give a descriptive complexity account of the quasi-polynomial complexity of parity games: The winning regions of parity games with $p$ priorities and register-index $k$ are described by a modal $\mu$ formula of which the complexity, as measured by its alternation depth, depends on $k$ rather than $p$.

CCS Concepts · Theory of computation → Complexity theory and logic; Logic and verification; Modal and temporal logics;

Keywords parity games, modal $\mu$ calculus, algorithmic complexity, descriptive complexity

ACM Reference Format:

1 Introduction
Parity games are two-player games of infinite duration, played on finite graphs labelled with integer priorities. They arise at the intersection of logic, games, and automata theory, with applications in reactive synthesis and verification. In particular, they are the model-checking games for the modal $\mu$ calculus, $L_\mu$. Solving parity games — that is, deciding which player has a winning strategy — is the subject of much research, focused both on understanding their complexity and developing practical solvers [25].

Despite extended efforts, the exact complexity of solving parity games remains an open problem. The problem is in $\text{UP} \cap \text{coUP}$ [21] and admits a quasi-polynomial algorithm [9, 12, 16, 23]. While a general polynomial algorithm is still elusive, many restricted classes of parity games are known to be solvable in polynomial, or even sub-polynomial time. This is the case for classes of games with a logarithmic number of priorities [9] as well as for classes of games where the structure of the underlying graph is restricted: parity games of bounded Kelly-width [19], DAG-width [3] and entanglement [4] are all solvable in polynomial time, while those with bounded tree-width and clique-width are even in $\text{LogCFL}$ [15], the class of languages logspace reducible to context-free languages.

These structural measures on the graph of a parity game are orthogonal to the size of the priority assignment. However, the complexity of a parity game is not solely due to its structure nor to the size of its priority assignment, but to how both interact. For example, the cycles in a graph may be intertwined, causing high entanglement, yet unless the priorities are equally entangled within these cycles, this measure is not an accurate reflection of algorithmic complexity. The key insight of this paper is to capture the complexity of the priority assignment with a new measure which is logarithmically bounded in the size of the graph. This yields a new quasi-polynomial algorithm and a descriptive complexity account of the quasi-polynomial solvability of parity games.

A. A new measure of complexity
Our first contribution is to parametrise the complexity of the priority assignment of a parity game. We introduce the register-index as a new, $L_\mu$ describable, bisimulation invariant measure of complexity for parity games, which captures, roughly speaking, how many priorities the winner of a parity game needs to keep in memory in order to produce a witness of their victory. Like entanglement, register-index...
is defined in game-theoretic terms, using a parametrised \(k\)-register game that is played on a parity game arena.

We then define a \(k\)-parameterised polynomial-time algorithm which, for fixed \(k\), solves parity games of register-index up to \(k\). This adds parity games of bounded register-index to the classes of parity games known to be solvable in polynomial time. Notably, even the class of games of register-index 1 contains non-trivial parity games of arbitrarily high entanglement, tree-width, Rabin-index, Kelly-width, DAG-width and arbitrarily many priorities, as well as the known to non-trivial syntactic inefficiencies.

We then show that the register-index of a parity-game of size \(n\) is \(O(\log n)\). This yields a novel proof that parity games are solvable in quasi-polynomial time, based on reducing solving a parity game of size \(n\) with \(p\) priorities to solving one of size \(O(p^{\log p} n \log n)\) with \(O(\log n)\) priorities.

### B. An \(L_\mu\) account of register-index

Although the runtime of this algorithm does not improve over existing quasi-polynomial algorithms, it provides a novel, \(L_\mu\) perspective on solving parity games in quasi-polynomial time. In the second half of the paper, we link the quasi-polynomial solvability of parity games to their descriptive complexity. It is well known that the winning regions of parity games with maximal priority \(p\) are described by a \(L_\mu\) formula of which the alternation depth depends only on \(p\). We show that the alternation depth of the \(L_\mu\) formula describing the winning regions of parity games of register-index up to \(k\) and of maximal priority \(p\) depends only on \(k\), rather than \(p\). Its size is \(O(kp^k)\). Since register-index is logarithmic in the size of the game, a \(L_\mu\) formula of alternation depth \(k\) and size \(O(kp^k)\) describes the winning regions of parity games of size up to \(2^k\) with up to \(p\) priorities.

Finally, we show that register-index, when extended to \(L_\mu\) formulas, is a decidable upper bound on the semantic complexity of formulas, and demonstrate that it is sensitive to non-trivial syntactic inefficiencies.

### C. Overview

In brief, we provide new insights into the complexity of parity games and link their quasi-polynomial solvability to their descriptive complexity. Its main contributions are:

- Introducing a new, bisimulation invariant measure of complexity for parity games (Section 3);
- Showing that for fixed \(k\) solving parity games of register-index up to \(k\) is in \(P\), thus adding parity games of bounded register-index to the classes of parity games known to be solvable in polynomial time (Theorem 3.6); Classes of games of different register-index are studied in Section 4;
- Bounding the register-index of parity games of size \(n\) with \(O(\log n)\) (Theorem 4.7), resulting in a new quasi-polynomial parity game algorithm (Section 5);
- Relating register-index to complexity in \(L_\mu\) (Section 6).

## 2 Background

### 2.1 Parity games

**Definition 2.1 (Parity games).** A parity game is an infinite-duration two-player zero-sum path-forming game, played between Even and her opponent Odd on a finite game graph \(G = (V, V_e, V_o, v_0, E, \Omega)\) called the arena. The vertices \(V\) of the arena are partitioned into those belonging to Even, \(V_e\), and those belonging to Odd, \(V_o\). The priority assignment \(\Omega : V \rightarrow I\) maps every vertex to a priority in \(I\), a finite prefix of the non-negative integers. Starting at an initial state \(v_i \in V\), a play proceeds with the owner of the current state \(v\) choosing a state \(v'\) among its successors in the directed edge relation \(E \subseteq V \times V\). Thus the players collaboratively form a play, consisting of a potentially infinite path along the edges of the game graph. If the play is finite, the owner of its last position loses; Otherwise Even wins if the highest priority visited infinitely often is even, else Odd wins.

A (positional) strategy \(\sigma\) for a player \(P \in \{\text{Even}, \text{Odd}\}\) maps every position \(v\) belonging to \(P\) in a parity game to one of its successors \(\sigma(v)\). A play \(\pi = v_0v_1...\) is said to agree with \(\sigma\) when for all \(i\), if \(v_i\) belongs to \(P\), then \(v_{i+1}\) is \(\sigma(v_i)\). A strategy \(\sigma\) for player \(P\) is said to be winning for \(P\) if all plays starting at \(v_i\) that agree with \(\sigma\) are winning for \(P\).

**Theorem 2.2.** [11] In all parity games, one of the players has a positional winning strategy.

It will often be convenient to assign priorities to edges instead of vertices, with \(\Omega : E \rightarrow I\). A parity game with edge priorities can be converted into one with node priorities by introducing intermediate, priority-carrying nodes onto all edges and assigning a low priority to other vertices.

We denote the set \(\{1, 2, ..., n\}\) with \(\lceil n \rceil\).

### 2.2 Entanglement

Register-index is inspired by and related to entanglement, which was introduced by Berwanger and Grädel [4] as a measure of how intertwined the cycles of a graph are.

**Definition 2.3 (Entanglement).** Entanglement is defined with a detective game in which a team of \(k\) detectives tries to catch a thief. The thief picks his initial position, while the detectives start outside of the graph. At each move, the detectives can either stay as they are or one of them can move to the current position of the thief. The thief then chooses a successor state not occupied by a detective. If such a state does not exist, the detectives win. If the thief can keep playing infinitely, he wins. Then, the entanglement of a graph is the minimal \(k\) for which \(k\) detectives have a winning
strategy. For example, \(n\)-cliques have entanglement \(n - 1\) and acyclic graphs have entanglement 0.

A positional strategy \(\sigma\) for player \(P\) on a parity game \(G\) induces a subgraph \(G_\sigma\) in which positions belonging to player \(P\) only have one successor. The entanglement of a parity game is the minimum entanglement of a subgame \(G_\sigma\) induced by a positional winning strategy \(\sigma\).

**Theorem 2.4.** [4] Parity games of bounded entanglement are solvable in polynomial time.

The proof of this theorem uses a \(k\)-super-detective game, solvable in polynomial time for fixed \(k\). For parity games on graphs of entanglement \(k\), Even wins the parity game if and only if she wins the \(k\)-super-detective game. The \(k\)-register game, introduced next, can be seen as a bisimulation-invariant, infinite duration variation of this game\(^1\).

Although not bisimulation invariant, entanglement is related to \(L_\mu\): the entanglement of a finite graph corresponds to the number of fixpoint variables needed to describe it up to bisimulation with a \(L_\mu\) formula – a fact used to show the strictness of the \(L_\mu\) variable hierarchy [6]. The winning regions of parity games of bounded entanglement are LFP-describable [10]. The relationship between entanglement and other structural measures is discussed in Berwanger et al. 2012 [5], while directed and undirected graphs of entanglement two are studied in Grädel et al. 2009 [17] and Belkhir and Santocanale 2007 [1] respectively.

### 3 The register-index of parity games

Like entanglement, the register-index of parity games is defined using a parametrised game: the \(k\)-register game. This game is played on a parity game arena, on which in addition to winning the parity game, player Even has produced a witness of her victory using a fixed amount of memory. This section introduces the \(k\)-register game and shows that it is an under-approximation of a parity game: for any \(k\) Even wins the \(k\)-register game on a parity game \(G\), she also wins in the parity game on \(G\); for every parity game \(G\) in which Even wins, there is \(k\) such that Even wins in the \(k\)-register game on \(G\). We call the minimal such \(k\) the register-index of \(G\). Solving a parity game \(G\) of bounded register-index then reduces to solving a \(k\)-register game on \(G\) which, for fixed \(k\), can be done in polynomial time.

#### 3.1 Register games

The \(k\)-register game consists of a normal parity game, augmented with \(k\) registers. Each register records the highest priority that has occurred in the parity game since it was last reset. The registers are ranked according to how long it has been since their last reset, with a newly reset register having rank 1. Initially, all registers contain 0 and the ranking is arbitrary. One player, let’s say Even, is given control of the registers. At each turn, Even can choose to reset a register of any rank \(r\). If the register contains the priority \(p\), this produces output \(2r\) if \(p\) is even and \(2r + 1\) otherwise. As long as Even resets registers infinitely often, this produces an infinite sequence in \([2k + 1]^\omega\). To win, Even has to either win finitely in the underlying parity game, or produce an infinite sequence of outputs that satisfies her parity condition: the maximal priority output infinitely often must be even.

We now define formally the \(k\)-register game on a parity game \(G\) with priority domain \(I\). Since the winning condition of the register game is a parity condition, we present the \(k\)-register game on a parity game arena \(G\) as a parity game on an arena \(R^k(G)\), of which the positions are positions of \(G\) paired with vectors in \(l^k\) that represent the contents of the registers. An additional binary variable \(r\) indicates whether the next move is a potential register reset \((t = 0)\), or a move in the underlying parity game \((t = 1)\).

**Definition 3.1** \((k\)-Register game). Let \(G\) be a parity game \((V^G, V^G_e, V^G_o, \Omega^G, \Omega^G)\) and let be \(I\) the co-domain of \(\Omega^G\): \(V^G \rightarrow I\). For a fixed non-zero parameter \(k \in \mathbb{N}\), the arena of the \(k\)-register game \(R^k(G)\) on \(G\) in which Even controls the registers, consists of \(R^k(G) = (V, V^e, V^o, v^i, E, \Omega)\) as follows. While \(G\) carries its priorities on its vertices, for the sake of clarity, \(R^k(G)\) carries them on its edges: \(\Omega: V \rightarrow [2k + 1]\).

- \(V\) is a set of positions \((p, x, t)\) in \(V^G \times l^k \times \{0, 1\}\).
- \(V^e\) consists of \((p, x, 1)\) such that \(p \in V^G_o\).
- \(V^o\) consists of \(V \setminus V^o\).
- \(v^i\) is \((v^i, 0, 0)\).
- \(E\) is the disjoint union of sets of edges \(E_{move}, E_{skip}\) and \(E_r\) for all \(r \in [k]\) where:
  \(E_{move}\) consists of edges \(((p, x, 1), (p', x', 0))\) such that:
  - \((p, p') \in \Omega^G\) and
  - \(x'_i = \max(\Omega^G(p'), x_i)\).
- \(E_{skip}\) consists of edges \(((p, \bar{x}, 0), (p, \bar{x}, 1))\).
- For each \(r \in [k]\), \(E_r\) consists of edges \(((p, \bar{x}, 0), (p, \bar{x}', 1))\) such that:
  - \(x'_i = x_i\) for \(i > r\),
  - \(x'_i = x_{i-1}\) for \(1 \leq i \leq r\),
  - \(x'_1 = 0\).

\(\Omega\) assigns priorities from \([2k + 1]\) to edges as follows:

- Edges of \(E_{move}\) and \(E_{skip}\) have priority 1;
- An edge \(((p, \bar{x}, 0), (p, \bar{x}', 1)) \in E_r\) has priority 2 if \(x_r\) is even, and priority 2 + 1 otherwise.

**Terminology:** Given a play in \(R^k(G)\), we call the underlying play its projection onto the first element of each visited position. At a position \((p, x, t)\), we write that:

- a priority \(q \in I\) occurs if \(\Omega^G(p) = q\);
- a register of rank \(r \in [k]\) holds priority \(q \in I\) if \(x_r = q\);
- Even resets the register of rank \(r\) and outputs \(j \in [2k+1]\) if the play follows an edge in \(E_r\) of priority \(j\);

\(\text{Very roughly, instead of having to catch the thief just once, the super-detective needs to catch the thief infinitely often after a high even priority and only finitely often after a higher odd priority.}\)
Figure 1. Parity game of register-index 2

![Parity Game Diagram]

**Figure 2.** Fragment of a play in a 2-register game.

- $q$ is recorded at rank $r$ if Even resets the register of rank $r$ at a position $(p, \bar{x}, 0)$ where $x_r = q$.
- The interval of a position $p$ at which Even resets a register $a$ is the fragment of the play since the previous reset of $a$ (not necessarily at the same rank).

A strategy for Odd in $G$ induces a strategy for Odd in $R^k_G(G)$. A strategy for Even in $G$, paired with a resetting strategy in $R^k_G(G)$ from positions $(p, \bar{x}, 0)$ induces a strategy for Even in $R^k_G(G)$.

The $k$-register game arena $R^k_G(G)$ where Odd controls the registers is similar except that positions $(p, \bar{x}, 0)$ are in $V_o$, edges in $E_{\text{move}} \cup E_{\text{skip}}$ have priority 0, and edges $((p, \bar{x}, 0), (p, \bar{x}', 1))$ such that Even resets infinitely often, causing Even to lose.

The register contents during a play in the 2-register game is shown in Figure 2. Note how a reset of rank 2 causes the contents of the register of rank 1 to move into the register of rank 2. Even can win by resetting the register of rank 1 after seeing 1 or 0, and the one of rank 2 after seeing 2.

Dedicated readers may find it helpful to work out the register-index of the 4-node parity game in Figure 5.

For fixed $k$, the $k$-register game on $G$ is a parity game on an arena of polynomial size and of priority domain $[2k+1]$: it can be solved in polynomial time in the size of $G$.

### 3.2 Register-index

We first prove two fundamental properties of $k$-register games.

**Lemma 3.3.** If Odd has a winning strategy in $G$, he also has a winning strategy in $R^k_G(G)$, for all $k$. Dually, if Even has a winning strategy in $G$, she also has a winning strategy in $R^k_G(G)$, for all $k$.

**Proof.** We show that a winning strategy $\tau$ for Odd in $G$ is also winning in $R^k_G(G)$, for any $k$.

Odd obviously wins finite plays that agree with $\tau$. Odd also wins if a play only contains a finite number of resets. For an infinite play with infinitely many resets that agrees with $\tau$, consider the highest rank $r$ at which registers are reset infinitely often. Observe that for a newly reset register to reach rank $r$, a register of rank $r$ or higher has to be reset at least once. Therefore if a reset of rank $r$ occurs at a position $p$, then the the interval of $p$ begins no later than at the previous position at which a reset of rank $r$ or higher occurred. As a result, as long as there are infinitely many resets, the dominant odd priority of the play is recorded infinitely often at rank $r$. The maximal priority output infinitely often is therefore odd. The proof of the dual statement is similar. $\square$

The converse does not hold: Even can have a winning strategy in a parity game $G$ but still lose the $k$-register game on $G$ (see Example 3.2). However, for every parity game $G$, there is a $k$, bounded by the number of even priorities in $G$ (and, as we will see later, by $1 + \log n$), such that Even has a winning strategy in $G$ if and only if she has a winning strategy in $R^k_G(G)$.

**Lemma 3.4.** Let $G$ be a parity game with $k$ even priorities, in which Odd has a winning strategy. Even has a winning strategy in $R^k_G(G)$.

**Proof.** Even’s strategy in $R^k_G(G)$ is to play a positional winning strategy in the underlying parity game $G$ and reset registers as follows: when an even priority $p$ occurs in the underlying game for the first time, she resets the register of highest rank, and whenever $p$ occurs again, she resets that same register, regardless of its current rank.

A reset then records the highest priority seen between two occurrences of an even priority (excluding the first occurrence, but including the second). The rank of a register $a$ at the occurrence of an even priority $p$ is the number of distinct even priorities that have occurred since the last occurrence of $p$, plus one: every occurrence of a new even priority triggers the reset of a register that was last reset earlier than $a$, thus increasing the rank of $a$; repeated occurrences of even priorities between consecutive occurrences of $p$ trigger the reset of registers of smaller rank than $a$. Therefore, if a register is reset at $v$ and another one at $v'$, and the interval of $v$ is contained within the interval of $v'$, then the reset at $v'$ is of strictly higher rank than the reset at $v$. 

<table>
<thead>
<tr>
<th>Priority</th>
<th>1</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>2</th>
<th>1</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>Reg. 1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Reg. 2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Output</td>
<td>-</td>
<td>5</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>4</td>
</tr>
</tbody>
</table>
Let \( q \) be the highest priority that occurs in a play \( \pi \) infinitely often, \( q' \) the highest odd priority that is recorded infinitely often, and \( r \) the highest rank at which \( q' \) is recorded infinitely often. Now consider a suffix of \( \pi \) in which \( q \) is the first occurring priority and in which no odd priority larger than \( q' \) occurs. In this suffix, whenever at a position \( v \) a register is reset and \( q' \) is recorded, the interval at \( v \) is nested within two consecutive occurrences of \( q \), as else \( q \) would be recorded instead of \( q' \). Therefore, if \( q' \) is recorded at rank \( r \), the next reset at a position where \( q \) occurs records \( q \) at a higher rank than \( r \). As a result, \( r \) is recorded infinitely often at a rank higher than \( r \), so the maximal output produced infinitely often is even.

Similarly, if Odd wins a parity game \( G \) with \( k \) odd priorities, he has a winning strategy in \( \mathcal{R}_k^e(G) \).

**Definition 3.5** (Register-index). The *register-index* of a parity game \( G \) is the least integer \( k \) such that Even wins both \( G \) and \( \mathcal{R}_k^e(G) \) or Odd wins both \( G \) and \( \mathcal{R}_k^o(G) \). This amounts to the least \( k \) such that \( \mathcal{R}_k^e(G), \mathcal{R}_k^o(G) \) and \( G \) have the same winner.

**Theorem 3.6.** Parity games of bounded register-index are solvable in polynomial time.

**Proof.** For a fixed parameter \( k \), let \( G = (V, V_e, V_o, v, E, \Omega) \) be a parity game of register-index at most \( k \) with priority co-domain \( I \): Even has a winning strategy in \( G \) if and only if she has a winning strategy in \( \mathcal{R}_k^e(G) \). The game \( \mathcal{R}_k^e(G) \) is itself a parity game, with state space \( V \times I^k \times \{0, 1\} \) and priorities \( [2k + 1] \). As \( k \) is fixed, this is \( \text{PTime} \)-solvable.

\( \square \)

## 4 Some register-index properties

This section studies register-index in more details. First, we compare register-index to other measures, then we prove one of the core theorems of this paper: register-index is logarithmic in the size of the game. In brief:

- The number of priorities in a parity game is an upper bound on its register-index (Lemma 3.4);
- Parity games in which a \( k \)-set of vertices intersects all cycles have register-index at most \( k \) (Lemma 4.1);
- Games of register-index 1 can have arbitrarily high entanglement and number of priorities;
- For all \( m \) there are parity games of register-index \( m \) with entanglement 2.
- Rabin index and register-index are orthogonal.
- A parity game of size \( n \) has register-index at most \( 1 + \log n \) (Theorem 4.7).

As with entanglement, the register-index depends only on the subgraphs induced by winning strategies. Therefore all examples are one-player games where Odd controls all positions but Even wins.

### 4.1 Register-index and entanglement

In some cases, entanglement and register-index coincide:

**Lemma 4.1.** If there is a set of positions \( S \) of size \( k \) that intersects all cycles, then the parity game has register-index (and entanglement) at most \( k \).

**Proof.** The winning player can use a positional winning strategy in the underlying game and reset registers as follows: at the first occurrence of \( v \in S \) she resets the register of maximal rank, and at each subsequent occurrence of \( v \), she resets the same register, regardless of its rank. Each register that is reset at least twice records the highest priority that occurs in a cycle from a position in \( S \) to itself. As \( S \) intersects all cycles, registers get reset infinitely often. The underlying strategy forces all cycles to be dominated by a priority of the winning parity; all outputs are of this parity.

\( \square \)

In general entanglement and register-index are orthogonal:

**Lemma 4.2.** There are parity games of register-index 1 with both arbitrarily high entanglement and number of priorities.

**Proof.** Consider the \( n \) clique \( C_n \) (Figure 3) where all nodes \( v_i, i \in [n] \) belong to Odd, with the following edge-priorities: the edge from \( v_i, i \in [n] \) to \( v_j, j \in [n] \) if \( i > j \) has priority \( 2i \); the edge from \( v_j \) to \( v_i \) has priority \( 2i - 1 \). This parity game has entanglement \( n \) and \( 2n \) priorities. Even wins \( \mathcal{R}_1(C_n) \): she resets the register whenever an even priority is seen. Any path with only odd priorities, followed by an even priority \( p \), sees no priorities higher than \( p \). Therefore, every recorded priority is of entanglement \( p \). Hence every output is even.

\( \square \)

**Example 4.3.** To see how changes in the order of priorities affect the register-index, reverse the edges of \( C_n \) (Figure 4) to obtain a family of parity games of register-index 2 and of size, entanglement and number of priorities \( \Omega(n) \). Even’s winning strategy in the 2-register game consists of resetting at each turn the register of rank 1 if it contains an odd priority, and resetting the register of rank 2 otherwise.

**Lemma 4.4.** For all \( m \), there exists a parity game of register-index at least \( m \) and entanglement 2.

**Proof.** Let \( H_0 \) be the game arena consisting of a single node, belonging to Odd, with a self-loop of priority 0. This unique node is also the initial node of \( H_0 \).

Then, for all \( n > 0 \), the arena \( H_n \) consists of two distinct copies of \( H_{n-1} \) with initial positions \( v_0 \) and \( v_1 \) respectively, with an edge \((v_0, v_1)\) of priority \( 2n - 1 \) and an edge \((v_1, v_0)\) of priority \( 2n \). The position \( v_0 \) is also the initial position of \( H_n \). See Figure 5.

Even wins these parity games \( H_n \) because all cycles are dominated by an even priority. \( H_0 \) is of entanglement 1, while \( H_i \) for \( i > 0 \) is of entanglement 2. We will show that for \( n > 0 \), Odd has a winning strategy in the \( n \)-register game \( \mathcal{R}_n^o(H_n) \).
We reason inductively, and show that for each $H_n$, i) from the initial position in $\mathcal{R}^n_n(H_n)$ for $m \geq n$, with a register configuration in which register contents are bounded by $2n - 1$, Odd can force the game to output $2n + 1$ or a higher odd priority, before returning to the initial position, and ii) Even loses in $\mathcal{R}^n_n(H_n)$.

**Base case.** We first show that Odd has a winning strategy in $\mathcal{R}^1_1(H_1)$. His strategy is to always loop in the current position until Even resets the unique register, then move to the other position. This causes both 1 and 2 to be recorded infinitely often at rank 1, producing 2 and 3 infinitely often. Note that if Odd uses this same strategy in $\mathcal{R}^m_m(H_m)$ for $m \geq 1$, starting from a register configuration in which register contents are bounded by 1, although he can't necessarily win, he can force the game to either stay away from the initial position, or output $2n + 1$ or a higher odd priority before returning to the initial position.

**Inductive step.** Assume i) and ii) for $H_n$.

i) Consider the following strategy for Odd in $\mathcal{R}^m_m(H_{n+1})$ for $m \geq n + 1$. He first moves from $v$, the initial position of $H_{n+1}$ onto the initial position $v'$ of the second component of $H_{n+1}$, via the odd priority $2n + 1$. Then, he plays in $H_n$, which only contains priorities smaller than $2n + 1$, with a strategy that is winning in $\mathcal{R}^n_n(H_n)$. To counter this strategy, Even has eventually to reset a register of rank $n + 1$ or higher, after which Odd returns to the initial position. This is his strategy $\tau_{n+1}$. Observe that if at the initial position all register contents are bounded by $2n + 1$, then Even will either lose in the second $H_n$ component, or reset a register of rank $n + 1$ or higher when it contains the odd priority $2n + 1$, outputting $2n + 3$ or a higher odd priority.

ii) We now show that he also has a winning strategy in $\mathcal{R}^{n+1}_n(H_{n+1})$. He begins by playing $\tau_{n+1}$ until $2n + 3$ is output and the play is back at the initial position. Note that the registers now all contain $2n + 2$, so he can not yet repeat $\tau_{n+1}$. Instead, he plays a strategy that is winning in $\mathcal{R}^n_n(H_n)$ in the first $H_n$ component of $H_{n+1}$. Again, Even will lose unless she resets register $n + 1$ infinitely often. After she has reset the register at rank $n + 1$ at least $n + 1$ times, all the registers hold values smaller than $2n + 1$. Odd can then return to the initial position, and again use $\tau_{n+1}$ to force output $2n + 3$. Thus by alternating between using $\tau_{n+1}$ to force the maximal odd output, and clearing the registers of higher priorities in order to be able to use $\tau_{n+1}$ again, Odd forces $2n + 3$ to be output infinitely often.

Hence $H_n$ has register-index at least $n + 1$.

\[ \square \]

### 4.2 Rabin-index and register-index

Another related measure is the Rabin index [20] which, like register-index, considers both the structure of cycles in a parity game and its parity assignment. The idea of the Rabin index is to capture the minimal priority assignment that preserves winning regions and strategies regardless of the ownership partition of the game graph.

**Definition 4.5** (Rabin index). Given a directed graph $G = (V, E)$, two priority assignments $\Omega, \Omega'$ over $V$ are equivalent if for all partitions of $V$ into $V_e$ and $V_o$, and all $v \in V$, the parity games $(V, V_o, V_e, v, \Omega)$ and $(V, V_o, V_e, v, \Omega')$ have the same winner and same set of winning strategies. The Rabin index of a parity game $G$ with priority assignment $\Omega$ is then the size of the smallest co-domain of any equivalent priority assignment over the graph of $G$.

**Lemma 4.6.** For all $k$ there are parity games of register-index at least $k$ and Rabin index 1 and vice versa.

**Proof.** Since the Rabin index is agnostic to the ownership of nodes and register-index depends on the subgraphs induced
by winning strategies, we can build games of low register-index and high Rabin index as follows: a game consisting of a starting position belonging to Even with one successor in a winning region of register-index 1, and another in a parity game of Rabin index \( k \) has register-index 1 and Rabin index at least \( k \). Conversely, the graphs \( H_n \) of register-index at least \( n + 1 \) are all of Rabin index 1, as witnessed by the trivial colouring which assigns 0 to all positions. Rabin index and entanglement are therefore orthogonal. \( \square \)

4.3 A logarithmic bound on register-index

In this section we prove one of the main results of this paper:

**Theorem 4.7.** The register-index of parity games of size \( n \) is at most \( 1 + \log n \).

Before diving into the main proof, we define the following:

**Definition 4.8.** A strategy \( \sigma \) for Even in a \( k \)-register-game of maximal priority \( q \) is **defensive** if, from a configuration where the highest ranking register contains \( q \) or a higher even priority, plays which agree with \( \sigma \) never output \( 2k + 1 \).

**Definition 4.9.** The **defensive** register-index of \( G \) where Even wins the parity game is the least integer \( k \) such that Even wins \( R_k^E(G) \) with a defensive strategy. The register-index of \( G \) is bounded by its defensive register-index.

**Proof.** Let \( G = (V, Vr, Vw, v, E, \Omega) \) be the parity game induced by a positional winning strategy \( \sigma \) for Even in some parity game: nodes \( v \in V_r \) have a unique successor, namely \( \sigma(v) \). Let \( q \) be the least even priority larger or equal to all priorities appearing in \( G \). We start by proving that if \( G \) has defensive register-index \( r > 1 \), it has two distinct subgames of defensive register-index \( r - 1 \).

If \( r = 2 \), this is trivial: all singleton games have register-index 1. We now consider the case of \( r > 2 \).

Let \( G_1, ..., G_i \) be the maximal strongly connected subgames of \( G \) induced by vertices of priority up to \( q - 2 \), and let their defensive register-indices be \( k_1, ..., k_j \) respectively: \( k_i \) is the number of registers Even needs in order to win with a defensive strategy in the \( k_i \)-register game from all positions of \( G_i \). Let \( k_m \) be a maximum among these, or 1 if \( G \) has no strongly connected subgames of priority up to \( q - 2 \).

We show that \( k_m \geq r - 1 \). For contradiction, assume \( k_m < r - 1 \). Even’s strategy in the \( r - 1 \)-register game on \( G \) is as follows. She resets the register of rank \( r - 1 \) only when \( q \) occurs. Then, in each subgame \( G_i \), she simulates her winning strategy in the \( k_i \)-register game using only the \( k_i \) lowest ranking registers, leaving the register of rank \( r - 1 \) untouched. This forces Odd to eventually leave every subgame \( G_i \), or lose. Since every cycle is dominated by an even priority, a play that eventually leaves every subgame \( G_i \), or which eventually doesn’t enter subgames any more, is either finite and winning for Even, or sees \( q \) infinitely often. This strategy is winning and defensive, since it resets the highest ranking register infinitely often when it contains \( q \), and never when it contains an odd priority. \( G \) therefore has defensive register-index at most \( r - 1 \), rather than \( r \), a contradiction.

Therefore \( k_m \geq r - 1 \). If \( k_m = r \), we consider the strictly smaller subgame \( G_m \) instead of \( G \) and show that it too has two subgames of defensive register-index \( r - 1 \).

If only one subgame \( G_i \) has defensive register-index \( k_m = k_i = r - 1 \), we show that Even has a defensive winning strategy in the \( r - 1 \) register game on \( G \), a contradiction. Her strategy begins by first resetting all registers that initially contain odd priorities (this might take several turns), regardless of the progress in the parity game. Note that if the register of rank \( r - 1 \) initially contains \( q \) or a higher even priority, this register does not get reset. After this initial series of resets, her strategy is as follows. She resets the register \( r - 1 \) whenever she sees \( q \). Note that since \( q \) is the maximal priority and \( r - 1 > 1 \), immediately after this reset, the register of maximal rank still contains \( q \) or a higher even priority. In subgames \( G_j \) where \( j \neq i \), she simulates a winning strategy in \( R_k^E(G_i) \) without using the register of rank \( r - 1 \). When she enters \( G_i \) she plays a defensive winning strategy, this time using all \( r - 1 \) registers.

This strategy is winning: finite plays are trivially winning for Even; a play that eventually stays in a subgame agrees
with a winning strategy within that subgame; a play that either leaves subgames infinitely often or eventually does not enter a subgame must see $q$ infinitely often, and always between two visits to $G_i$, and therefore output $2(r−1)$ but not $2(r−1)+1$ infinitely often. If the initial register-configuration contains $q$ or a higher even priority in the register of rank $r−1$, then this strategy never enters $G_i$ without $q$ or a higher even priority in the register of rank $r−1$, and therefore never outputs $2(r−1)+1$. Hence it is defensive. The defensive register-index of $G$ is therefore $r−1$, a contradiction. We conclude that a game of defensive register-index $r > 1$ has at least two distinct subgames of defensive register-index $r−1$.

Then, as the same argument applies to all subgames of $G$, a game of defensive register-index $r$ has $2^{r−1}$ distinct subgames of defensive register-index 1. This bounds the defensive register-index of a parity game with $1 + \log n$ where $n$ is the size of the parity game. The same bound applies to the register-index, which must be lower than the defensive register-index.

\[ \square \]

\section{Algorithmic complexity}

The algorithmic consequence of Theorem 4.7 is that one can always solve $R^k_{\leq \log |G|}(G)$ instead of $G$, which gives an alternative quasi-polynomial time algorithm for solving parity games:

\textbf{Corollary 5.1} (Also from [9,23]). Parity games are solvable in quasi-polynomial time.

\textbf{Proof.} Let $n$ be the number of vertices in a parity game $G$ with register-index $k$, with $p$ distinct priorities, and $m$ edges. The parity game $R^k_{\leq \log |G|}(G)$ has $O(np^k)$ positions, $2k+1$ priorities and $O((m+kn)p^k)$ edges. Since $R^k_{\leq \log |G|}(G)$ has its priorities on its edges, for the complexity analysis we take the size of $R^k_{\leq \log |G|}(G)$ to be $O(knp^k)$, to account for an additional vertex for each of the $O(knp^k)$ edges of significant priority – that is, those in $E_r$ for some $r \in [k]$.

The register-index of a parity game of size $n$ is at most $1 + \log n$. Therefore solving parity games $G$ reduces to solving $R^k_{\leq \log |G|}(G)$ where $k = 1 + \log n$. The $R^k_{\leq \log |G|}(G)$ game can then be solved with an algorithm exponential in the number of priorities of one’s choice, say the small progress measure algorithm [22], to obtain a quasi-polynomial algorithm. \[ \square \]

For a tighter complexity analysis, one can use Jurdiński and Lazic’s quasi-polynomial succinct progress measure algorithm [23] which runs in $O(mp^{1.38})$ if the number of priorities is less than $\log \eta$, where $\eta$ is the number of positions of odd priority. This is the case for $R^k_{\leq \log |G|}(G)$, which has over $np^k$ positions of odd priority and $2k+1$ priorities (as long as $p \geq 4$). With size $O(knp^k)$ where $k = 1 + \log n$, this yields a $O(n^{4.48+3.38 \log n}(\log n)^{2.38})$ algorithm. The space complexity remains quasi-polynomial.

Although this algorithm does not improve over the runtime of existing quasi-polynomial time algorithms, its parameterised versions, consisting of solving $R^k_{\leq \log |G|}(G)$ instead of $G$ for a parameter $k$, provide polynomial algorithms for the rich classes of games of bounded register-index. Even the class of games of register-index 1 contains games which are otherwise of arbitrary complexity. In particular, the families of parity games which exhibit worst-case behaviour for strategy improvement, divide-and-conquer, and one of the existing quasi-polynomial algorithms [2,12–14] all have constant register-index 1. In the former two cases, this is simply because although the games have complex features, the winner has a simple winning strategy, according to which all plays eventually only see one priority. The latter case is not trivial, but every odd priority is still immediately followed by a larger even priority, so Even still has an easy 1-register game strategy consisting of resetting the unique register whenever an even priority occurs. However, the class of parity games of register-index 1 is not restricted to such simple games, as illustrated by Figure 3. Larger register-indices only occur if the winning strategy induces a connected region with the particular structure described in Lemma 3.2 and Theorem 4.7. Solving $k$-register games is therefore a complementary approach to existing parity game solving strategies. It seems plausible that solving $R^k_{\leq \log |G|}(G)$ and $R^k_{\leq \log |G|}(G)$ for a $k$ as low as 2, or 3, and falling back onto a different algorithm if these don’t have the same winner could be an effective approach to solving parity games in practice.

Beyond its potential for practical parity-game solving, this algorithm gives a descriptive complexity account of the quasi-polynomial solvability of parity games, as we will see in the next section.

\section{Register-index and $L_\mu$}

The complexity of solving parity games is closely related to complexity in $L_\mu$. The priorities in the model-checking parity games of an $L_\mu$-formula reflect the complexity of the formula, as measured by the complexity of its fixpoint structure, known as its alternation depth [26]. Conversely, solving parity games is equivalent to model-checking a particular $L_\mu$ formula [18, Section 3.3.6], of which the alternation depth depends on the priorities in the parity game. In this section we argue that not only can we model-check this formula in quasi-polynomial time, but the formula itself can be chosen to be of logarithmic alternation depth and quasi-polynomial size.

We state the polynomial solvability of parity games of bounded register-index in terms of descriptive complexity: the winning regions of the $k$-register games on parity game
arenas with a bounded number $p$ of priorities can be described by a $L_\mu$-formula of size $O(kp^k)$, and alternation-depth $k$.

After establishing the $L_\mu$ expressivity of register-games in Section 6.2 we consider some of its consequences:

- Register-index is bisimulation invariant (Corollary 6.6).
- The winning regions of parity games of bounded size are trivially described by a $L_\mu$ formula without fixpoints, however, this formula is exponential in the size bound. The logarithmic bound on the register-index with respect to the size of games means that the winning regions of parity games with $p$ priorities of size up to $2^{k-1}$ are described by a $L_\mu$ formula of size $O(kp^k)$ with alternation depth $k$ (Corollary 6.7)
- Finally, we extend the definition of register-index to $L_\mu$-formulas and show that the register-index of a $L_\mu$ formula is a computable upper bound on its semantic alternation depth: formulas of register-index $k$ are equivalent to formulas of alternation depth $k$.

6.1 $L_\mu$

For ease of representation we will use $L_\mu$ equational systems. Fix countably infinite sets $\text{Prop} = \{P, Q, \ldots\}$ of propositional variables, and $\text{Var} = \{X, Y, \ldots\}$ of fixpoint variables.

**Definition 6.1.** ($L_\mu$) The syntax of unimodal basic formulas is given by:

$$\phi ::= P \mid X \mid \neg P \mid \phi \land \phi \mid \phi \lor \phi \mid \phi \land \phi \mid \phi \lor \phi \mid \bot \mid T$$

Conjunctions take precedence over disjunctions. The scope of modalities extends as little as possible to the right.

A $L_\mu$ sentence $(S, X, \Omega)$ is a finite set of $S$ equations of the form $X_i = \psi_i$, where the left-hand sides are distinct fixpoint variables, of which one is the designated entry point $X_0$, coupled with a priority assignment $\Omega : \text{Var} \rightarrow I$ where $I$ is a finite prefix of the non-negative integers.

The semantics of $L_\mu$ and its equational notation are standard – see for example [8]. In the context of this paper, the semantics are best described in terms of the model-checking parity games.

**Definition 6.2.** Given $S = (S, X, \Omega)$ a $L_\mu$ sentence and $M = (V^M, E^M \subseteq V \times V, s_0 \in V^M, Q^M : \text{Prop} \rightarrow \mathcal{P}(V^M))$ a structure we define the model-checking parity game $G(M, S)$ as the structure $(V, E, V_e \subseteq V, P_0, \ldots, P_q)$ where $q$ is the maximal priority in the co-domain $I$ of $\Omega$, and:

- Positions in $V$ consist of pairs $(s, \phi)$ where $s \in V^M$ and $\phi$ is a subformula of a basic formula in $S$;
- There in an edge from $(s, \phi \land \phi')$ to $(s, \phi)$ and $(s, \phi')$; from $(s, \phi \lor \phi')$ to $(s, \phi)$ and $(s, \phi')$; from $(s, \square \phi)$ to $(s', \phi)$ for each successor $s'$ of $s$ in $E^M$; from $(s, \diamond \phi)$ to $(s', \phi)$ for each successor $s'$ of $s$ in $E^M$; from $(s, X)$ to $(s, \phi_X)$ where $(X = \phi_X) \in S$;
- Positions $(s, \phi \lor \phi')$ and $(s, \phi \land \phi')$ satisfy $V_e$;
- Positions $(s, P)$ satisfy $V_e$ if $s \notin Q^M(P)$;
- Positions $(s, \neg P)$ satisfy $V_e$ if $s \in Q^M(P)$;
- Position $(s, X)$ satisfies $P_{Q(X)}$ while other positions satisfy $P_0$.
- $(s, X_0)$ is the initial position.

**Definition 6.3.** A structures $M$ is said to satisfy a sentence $S$ of $L_\mu$, written $M \models S$ if $\text{Even}$ has a winning strategy in the parity game represented by $G(M, S)$ when $V_e$ represents positions belong to Even, $V \setminus V_e$ represents positions belonging to Odd and $P_i$ represents positions of priority $i$.

**Definition 6.4.** The alternation depth of an $L_\mu$ system of equations consists of the number of even priorities in the co-domain $I$ of $\Omega$.

This presentation and the choice to only count even priorities is meant to emphasize the correspondence between alternation-depth, priorities in parity games and their register-index. A thorough discussion on how to define alternation depth, and comparison of definitions used in the literature can be found in Bradfield and Stirling 2007 [8].

6.2 Descriptive complexity

**Theorem 6.5.** There is a formula $\text{Win}^k_1$ of $L_\mu$ of alternation-depth $k$ that describes the class of parity games $G$ of priority domain $I$ for which $\text{Even}$ has a winning strategy in $\mathcal{R}^k_1(G)$.

**Proof.** Let $E_i$ stand for $V_e \land P_i$ and $O_i$ for $\neg V_e \land P_i$; let $M_1(X) = (E_i \land O_j) \lor (O_i \land E_j)$. This subformula describes the owner of the current position of priority $i$ taking a step in the parity game. Then, the formula for $Y_{\bar{a}, \bar{o}}$ describes the existence of a winning strategy from a register configuration $\bar{x}$, while $\bar{o}$ indicates the last output value:

$$Y_{\bar{x}, \bar{o}} = \bigvee_{j \in I} M_j(Y_{x_j, 1}) \lor \bigvee_{j \in I, r \leq k} P_j \land M_j(Y_{\text{next}(x_j', r, j), \text{out}(x_j', r)})$$

Where $x_j' = \max(x_j, j)$ and next($x_j', r, j$) is the new register contents after Even resets the register of rank $r$. That is to say next($x_j', r, j$) = $x_j' \oplus r$ for $i > r$, but $x_j' \oplus 1$ for $1 < i \leq r$, and 0 for $i = 1$. Furthermore, out($p, r$) is $2r + 1$ for odd $p$ and 2$r$ otherwise.

Then, the formula $\text{Win}^k_1$ is given by the system of equations consisting of $Y_{\bar{x}, p}$ where $x \in I^k$ and $p \in [1, 2k + 1]$, paired with $\Omega(Y_{\bar{x}, s, o}) = o$ and the entry point $Y_{\bar{0}, 1}$.

In the formula $Y_{\bar{x}, p}$, the outermost disjunction gives Even the choice to proceed in the underlying parity game by playing to $M_j(Y_{x_j, 1})$ where $j$ is the current priority, or to reset a register. In either case, $\bar{x}$ is updated. $\Omega$ guarantees that when a register is reset, a priority of the appropriate magnitude and parity is produced, as computed by out($t$). Hence by design, the model-checking parity game of $\text{Win}^k_1$ on any structure $G$ representing a parity game with priority domain $I$ is functionally equivalent to the parity game $\mathcal{R}^k_1(G)$. □
Win\(^k\) has alternation depth \(k\) and size \(O(kd^k)\): the descriptiv- 
tive complexity of parity games of bounded register-index 
collapses, just like their algorithmic complexity.

**Corollary 6.6.** Register-index is bisimulation invariant.

*Proof.* Win\(^k\) holds in a parity game \(G\) with priority domain 
\(I\) whenever Even wins \(\mathcal{R}_G^k\) (G). Dually, we can define OWin\(^k\) 
which holds in parity games \(G\) of priority domain \(I\) whenever Odd 
wins \(\mathcal{R}_G^k\) (note that this is not just the negation of Win\(^k\)). 
Recall that a parity game has register-index at most \(k\) if \(\mathcal{R}_G^k\) 
and \(\mathcal{R}_e^k\) have the same winner.

Then, parity games with priority domain \(I\) and of register-
index at most \(k\) are exactly those that satisfy \((\lnot\text{OWin}_G^k \land \text{Win}_G^k)\lor(\text{OWin}_G^k \land \lnot\text{Win}_G^k)\). Since \(L_\mu\) 
is bisimulation invariant, so is register-index.

**Corollary 6.7.** The winning regions of parity games of size 
up to \(2^k\) with \(p\) priorities are described by an \(L_\mu\) formula of 
size \(O(kp^k)\) and alternation-depth \(k\).

Note that the winning regions of parity games of bounded 
size \(n\) are also described by a formula without any fixpoints. 
However, that formula, obtained is by unfolding all the fix-
points of the standard parity game formula \(n\) times. For parity 
games with \(p\) priorities, this is a formula of size \(O(2^{pn})\).

We extend the definition of register-index to \(L_\mu\)-formulas:

**Definition 6.8** (Register-index of a formula). A formula \(\psi\) 
of \(L_\mu\) has register-index \(k\) if for all structures \(M\), the register-
index of the model checking parity game for \(M\) and \(\psi\) 
is bounded by \(k\).

Using the \(L_\mu\) description of the winner of \(k\)-register games, 
we then argue that a formula of register-index \(k\) is equiva-
 lent to a formula of alternation depth \(k\). In other words, 
the alternation hierarchy is finite for the fragments of \(L_\mu\) 
of bounded register-index. This is a simple consequence of the 
following theorem which relates the descriptive complexity 
of the model-checking games of a formula to the formula’s 
semantic complexity.

**Theorem 6.9.** [24] Given a formula \(\psi\) of \(L_\mu\), if for all struc-
tures \(M\) it is the case that \(M \models \psi\) if and only if \(G(M, \psi) \models \phi\), 
then there exists a formula of the alternation depth of \(\phi\), 
constructed from \(\psi\) and \(\phi\) that is equivalent to \(\psi\).

**Corollary 6.10.** If \(\psi\) is an \(L_\mu\)-formula of register-index \(k\), 
then \(\psi\) is equivalent to a formula of alternation depth \(k\).

*Proof.* By definition of register-index for all structures \(M\), 
the formula \(\psi\) holds in \(M\) exactly when Even wins \(\mathcal{R}_G^k\) (\(G(M, \psi)\)). 
From Lemma 6.5, it is therefore the case that \(M \models \psi\) if and 
only if \(G(M, \psi) \models \text{Win}_G^I\) where \(I\) is the priority domain 
of \(G(M, \psi)\) which depends only on \(\psi\). Then, from Theorem 
6.9, \(\psi\) is equivalent to a formula of alternation depth \(k\), built 
from \(\psi\) and \(\text{Win}_G^I\).

Register-index is therefore a decidable upper bound on 
the semantic complexity of a formula. While the register-
index can be as high as the syntactic alternation depth of 
a formula, it is less affected by syntactic inefficiencies: for 
example, if \(\phi\) is unsatisfiable, but of high syntactic complex-
ity, a formula \(\psi(\phi)\) with subformula \(\phi\) will have syntactic 
complexity at least as high as \(\phi\). However, since to win in 
the model-checking parity game of \(\psi(\phi)\) a winning strategy 
for Even never needs to visit the subgame induced by \(\phi\), the 
register-index will only depend on \(\psi(\bot)\).

Note that the entanglement of the model-checking games 
of a formula is unbounded even for simple formulas so en-
taglement does not lend itself to a similar analysis.

### 7 Discussion

* A. Measuring the complexity of parity games

We introduced register-index, a measure of complexity 
of parity games that captures the complexity of the priority 
assignment. It is logarithmically bounded with respect to the 
size of a parity game.

Existing measures of complexity for parity games seem 
to be in one way or another agnostic to vertex ownership. 
Entanglement and register-index are measured on the one-
player subgame induced by a winning strategy while the 
Rabin index and purely structural measures are completely 
independent of node-ownership. While register-index takes 
to account two sources of complexity – structure and pri-
ority assignment – it leaves the complexity arising from 
alternations between the two players untouched. However, 
since one-player games are known to be solvable in poly-
nomial time, it seems plausible that node ownership could 
be incorporated into a more refined measure which would 
measure how cycles, priorities and alternations interact.

* B. Solving parity games in practice

The algorithm induced by the logarithmic bound on the 
register-index of parity games uses different techniques from 
previous algorithms [9, 23] to prove yet again that parity 
games are solvable in quasi-polynomial time. It does not 
improve on the running time of state-of-the-art algorithms; 
however, in addition to its rich theory, it enjoys a family of pa-
rameterised versions, each of which solves parity games for a 
class of games of bounded register-index. The class of parity 
games of register-index as low as 1 already contains parity 
games of otherwise arbitrary complexity, including families 
of games which trigger the worst-case performance of recur-
sive algorithms [14], strategy improvement algorithms [13], 
and one of the existing quasi-polynomial algorithms [12]. 
Conversely, single-player games with high register-index 
are trivial for algorithms like strategy-improvement which 
recognise a winning strategy quickly. Hence \(k\)-register games 
are a complementary solving method to state-of-the-art algo-
rithms. Families of high register-index require a particularly
intricate structure, as demonstrated in Lemma 4.7; how often such games are encountered in the wild is left as an open question. However, it seems likely that i) solving \( k \)-register games for small \( k \) could be a practical approach to solving parity games, and ii) existing solvers could potentially benefit from taking into account the register-index of games.

C. Parity games algorithms and \( L_\mu \)

The final part of this paper discussed how register-index and the new quasi-polynomial algorithm relate to complexity in \( L_\mu \). This discussion connects two long-standing open problems: the decidability of the \( L_\mu \) alternation hierarchy – that is, deciding, given a formula, whether it is equivalent to a formula of lower alternation depth – and the complexity of model-checking \( L_\mu \) – that is, the complexity of solving parity games. There are trade-offs between formula size and alternation depth that echo the trade-offs between solving a smaller parity game with many priorities, and a larger game with fewer priorities. It remains open whether the other quasi-polynomial algorithms can also be understood in logical terms. Bojańczyk and Czerwiński present the original quasi-polynomial algorithm in terms of reachability automata [7], which is a step in this direction.

Acknowledgements. I thank Udi Boker and Patrick Totzke for their comments on drafts of this paper. I am also grateful to several anonymous reviewers for their insights and suggestions.

References


