PARITY GAMES AND REGISTER INDEX

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
OF CHENNAI MATHEMATICAL INSTITUTE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Thejaswini Raghavan

Supervised by:

Dr. Marcin Jurdzinski

University of Warwick November 2021

Acknowledgments

First of all, I would like to thank my advisor Dr. Marcin Jurdzinski for his support, the wonderful ideas and motivation continue to keep me enthusiastic about work. He also went to a lot of trouble securing funding and making my work at Warwick possible. I am very glad that he agreed to be my PhD advisor as well.

I would also like to thank Laure Daviaud without whom this thesis would not be possible. I particularly appreciate the long discussions both in person and on Skype that we had, that helped me in writing this work. I am very grateful for all the feedback and the long hours that she put in to help me with writing and editing.

I would be remiss in my duties if I did not thank Prof. B. Srivathsan for introducing me to games, which got me excited about this line of work.

Finally, I would like to express my gratitude to my friends, family and the dogs of CMI who provided all the love, care, and distraction that I needed outside work.

Contents

Chapter 1

Introduction

Parity games are two-player games on graphs and have been studied since early 1990's [9, 10] with applications in automata theory on infinite trees, fixpoint logics, verification and synthesis.

Two players Even and Odd move indefinitely a pebble on vertices of a game-graph. Each vertex carries an integer called its priority, and the aim of Even is to ensure that the highest priority seen infinitely often is even. Odd has the opposite goal and it is well-known that given a starting vertex, either of the two players has a (positional) winning strategy to win the game.

Early and influential works by McNaughton [17] and by Zielonka [20] developed recursive algorithms to solve those games with a running time $O(n^{d+O(1)})$, where n is the number of vertices in the game and d the number of priorities. Recently, the breakthrough result of Calude et al. [4] gave the first algorithm that achieved a quasi-polynomial running time of $n^{o(d)}$, followed by two other quasi-polynomial algorithms by Jurdziński and Lazić [12] and Lehtinen [15]. Existence of a polynomial-time algorithm for solving parity games is a fundamental long-standing open problem [10]. However, many classes of games had been proved to be solvable in polynomial time, in particular when bounding some graph-theoretic parameters such as the tree-width [18], DAG-width [1], clique-width [19], Kelly-width [11] and entanglement [2].

In [15], Lehtinen introduces yet another parameter: the register index and uses it to provide a quasi-polynomial algorithm to solve parity games. It is also highlighted that games with bounded register index can be solved in polynomial time. This class of games is orthogonal to the ones mentioned above that are known to be in P. However, the register index is defined in quite an abstract manner and our aim is to provide a more intrinsic graph-theoretic characterization of the games with bounded register index, pinpointing exactly which games are and are not solvable in polynomial time with the algorithm

of Lehtinen. Our characterization is in terms of forbidden patterns: a game has at least register index k if and only if a specific graph H_k is embedded in it (for a notion of embedding defined in the core of the paper).

Organization of the thesis We introduce the basic definitions in Chapter 2. Chapter 3 and Chapter 4 introduce previous works in parity games that are relevant to the topic at hand. In Chapter 5, we give the definition and graph-theoretic characterization of the class $\mathcal{C}_k$ of games with bounded register index and solvable in polynomial time. The rest of the thesis is devoted to proving this characterization. In the same Section 5, we prove the lower bound, namely that a game in $\mathcal{C}_k$ has register index at least k and in Sections 6 and 6.1 we prove the upper bound, namely that a game in $\mathcal{C}_k$ has register index at most k.

Chapter 2

Preliminaries

A parity game is a two player game played between two player Even and Odd. The arena is a game graph G, where the vertices $V = V_o \uplus V_e$ of the graph are partitioned into two among player Odd and player Even. Each vertex, is in addition given a *priority*, from the set of natural numbers, $\{1, 2, \ldots, d\}$. There is a token, which can be placed on a vertex, and is moved along the edge, depending on the whom the vertex belongs to. A *play* of the parity game is the sequence of vertices visited by the token in the infinite duration. It is an element in the free monoid generated by V, V^*.

A play of the game is said to be winning for the Even player if the maximum among the set of priorities visited infinitely in the play often is even and it is winning for Odd otherwise.

In Figure 2.1, we have an example game where we have player Odd's vertices denoted by triangular nodes and player Even's vertices denoted by square nodes. The numbers in the vertices denote the prority of the vertex. In this example, it so happens that the priorities are distinct, but that need not be the case.

Suppose a token is placed in player Even's vertex with priority 5, and she moves it from there to the vertex with priority 2 from which she makes a move from there to the next vertex, belonging to the Odd player with priority 1, and then the token is moved to another of Odd player's vertex which has priority 4, and odd chooses to move it back to to 1 again, and this cycle is repeated infinitely often. So, the run here is $25(14)^\omega$. So, $\inf(25(14)^\omega) = \{1, 4\}$, and the maximum is 4, which is even and player Even would win. However, if at vertex with priority 4, Odd chose to always move the token to the left, to the even player, then we they play would look like $(2145)^\omega$. For this play, we know that $\max\{\inf((2145)^\omega)\} = 5$. And this play is winning for Odd.

A *winning strategy* for player Even is, informally, an instruction for player Even to make its move,

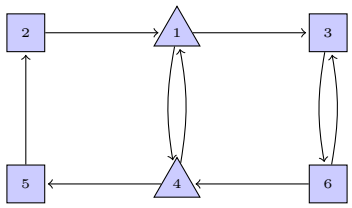


Figure 2.1: Example of Parity Game

so as to win the game irrespective of the decision of the other player. In the game in Figure 2.1, if Even follows the strategy at the vertex 3 and 6, to send the game to each other, we know that the game would be winning at these two states for Even. Such strategies are called *memoryless strategies*, where the players do not decide the move on the history of the play, rather, just the current position.

Formally, it a *strategy* is partial function from the sequence of plays to a vertex such that this vertex extends the play for Even. and a *memoryless strategy* is one such that this function is the same if the last vertex in the domain is the same.

It is known that parity games are *determined*. Further, they are known to be positionally determined for both the player. i.e, both Even and Odd have memory less strategies[9].

This immediately shows that checking if a game is winning for player Even from a vertex is in NP. This is done by guessing a positional strategy for player Even and verifying if this strategy is winning. Since the game is symmetric with respect to both the players, we can conclude that, Parity $\in NP \cap co - NP$.

2.1 Equivalence between priorities on edges and vertices

A variation of parity games considered in literature is that the priorities are assigned to edges instead of vertices and the winning condition is similar. It is not hard to see the equivalence between these two versions as one can be constructed from the other with only a polynomial difference in the size of a graph. An example of the modification is illustrated in the figure below.

Figure 2.2: Transferring priority from Vertices to Edges

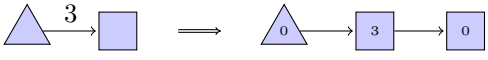


Figure 2.3: Transferring priority from Edges to Vertices

For instance, given a a parity game with priorities on vertices, one just transfers the same priorities to the edges that are outgoing from the said vertex. And for the conversion the other way, for each edge $e = (u, v)$, we introduce a new vertex w_e that has the same priority as the edge and assign the

priority 0 to the existing vertices and add edges (u, w_e) and (w_e, v) instead.

2.2 Finite version of Parity games

Also known as the first cycle game, it was introduced in [3]. The first cycle game, is also played on a similar game graph, where the two players are moved along the players' vertices. However the game comes to an end when some vertex is repeated in the duration of the play. The winner of the play is declared as Odd if the highest priority occurring infinitely often in the loop is Odd and Even is winning otherwise. [3] also shows that this game is equivalent to the Parity game defined above. Furthermore, parity games have a positional strategy.

Chapter 3

Progress Measure and Universal Trees

Since parity games are positionally determined, we have a witness of polynomial size for parity games, which just produces the strategy for any given player. However, progress measures, which we will re-introduce in this chapter, provides yet another certificate that can be verified locally. To completely understand the definition of progress measure [13], we first need to understand the definition of an ordered tree.

Suppose we have a rooted tree where the root of the tree is associated the empty sequence. Each edge is labelled from a linearly ordered set, such that for every node, all its children are labelled distinctly. For every vertex in the tree, there is a unique labelling corresponding to the edges on unique path from the root to the node. Such a tree with this labelling is an ordered tree. More formally,

Definition 1. *An ordered tree is a prefix closed set of tuples whose elements are from a linearly ordered set.*

With the usual $<$ order on natural numbers, the following prefix closed set $\{(), (1), (1,1), (1,2), (2), (2,1)\}$ can be realised as the tree in Figure 3.1, where the leaves from left to right, are labelled by $(1,1), (2,1), (2,2)$ and all left branches are labelled with 1, and the right branches with 2.

This order on the roots induces a total order on all the tuples if we impose the rule that an ancestor is strictly smaller than a child.

A *truncation* of a leaf in the tree by $p \leqslant r$, where $r \leqslant h$, is the height of the tree is the ancestor of

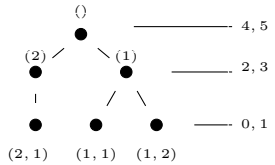


Figure 3.1: Example of an ordered tree

p at level h. So, we say the truncation of a tuple $(t_{d_1}, t_{d-3}, \ldots, t_1)|_p$, if p is odd, is $(t_{d_1}, t_{d-3}, \ldots, t_p)$ and $t_{d_1}, t_{d-3}, \ldots, t_{p+1}$ otherwise. For example, consider the node $(2,1)$ on the Figure 3.1, $(2,1) \mid_2 = (2,1) \mid_3 = (2)$ and for any node v, $v \mid_4 = ()$.

Definition 2. *Progress measure for a parity game G is a mapping μ from the vertices of the game graph winning for player Even, to an ordered tree such that*

- *If u belongs to Odd, then for every outgoing edge (u,v) from u, $\mu(u) \mid_{\pi(u)} \geqslant \mu(v) \mid_{\pi(v)}$*
- *If u is Even's vertex, then there exists an edge (u,v) such that $\mu(u) \mid_{\pi(u)} \geqslant \mu(v) \mid_{\pi(v)}$*

The inequality is strict when $\pi(u)$ is odd. And

For a game graph which is Even-winning, there exists therefore a tree with at most n vertices associated to it, called the progress measure tree. The main result of Jurdzinski and Lazic in [14], which gave a quasi-polynomial time algorithm to solve parity games was to provide an encoding of the progress measure tree so that one could search through the space of all possible progress measures faster.

It has been shown that if a parity game is winning from all vertices of the game graph for Even, then there exists a progress measure proved in [13] and later shown in [14] where the mapping to trees is made explicit along with the proof that it is necessary and sufficient to consider ordered trees of n leaves for the mapping.

Theorem 1 (Existence of Progress Measure[13],[14])**.** *Given a parity games on the vertex set V, such that the games is winning for Even from all vertices, there is a progress measure where the range is an ordered tree of at most n leaves.*

Lemma 1 (Succinct Encoding of Progress Measure Tree, [14])**.** *Any ordered tree having n leaves with depth $2.d$ can be encoded using at most $O(\log(n).\log(d))$ many bits.*

With Theorem 1 and Lemma 1, one can conclude that we can exhaustively search through all the ordered trees in $O(n^{logd})$ time, and therefore have a Quasi polynomial algorithm for Parity games.

Definition 3. *An (l, h) Universal tree is an ordered tree of height h, such that, given any ordered tree with n leaves and height l there is an injective map to this tree that preserves the order and maps the root of this ordered tree to the root of the universal tree.*

An alternative way to view the Lemma 1 is to view this as a succinct encoding of an universal tree. We know from [6] that the lower bound of the size of a universal tree is, upto a constant factor, the same of the upper bound provided in [14], in the encoding of ordered trees.

Chapter 4

Introduction to Register Index

We define here register games as introduced in [15]. Our definition slightly differs but retains all the properties proved in [15].

Given a parity game G and a positive integer k, the *k-register game* associated with G, denoted by R_G^k, is a parity game constructed from G and a set of k registers aimed at storing priorities of G.

The number of a register is called its rank. A valuation of the registers is a tuple of k integers between 0 and the highest priority of G. Given a rank i, a valuation $[x_k, x_{k-1}, \ldots, x_1]$ of the registers is said to be i-reset if it is changed into the valuation $[x_k, x_{k-1}, \ldots, x_{i+1}, x_{i-1}, \ldots, x_1, 0]$. Finally, given a priority j in G, a valuation $[x_k, x_{k-1}, \ldots, x_1]$ of the registers is said to be j-updated if every x_i smaller than j is changed into j.

For example, $\{3\}$-resetting the valuation of four registers $[6, 5, 3, 2]$ gives $[6, 3, 2, 0]$ and 4-updating $[6, 5, 3, 2]$ gives $[6, 5, 4, 4]$.

The parity game R_G^k works as follows: the registers all initially have value 0 and R_G^k follows a play in G (but without outputting priorities from G). After any move in G by any of the two players reaching some state of priority j, the following actions are performed:

- Player Even chooses a rank or 0.
- If Even has chosen 0, priority 1 is output. Otherwise, let t be the chosen rank. Priority $2t$ is output if the valuation of the register of rank t is even and $2t + 1$ if it is odd.
- The registers are t-reset.
- The registers are j-updated.

Since register valuations can be encoded in states, R_G^k defines a parity game. More formally, every vertex u in R_G^k can be defined as a 4-tuple: the vertex in G we are currently in, denoted by ν_u, the current valuation of the registers η_u, an integer in $\{0, \ldots, k\}$, denoted t_u representing a possibly chosen rank and an integer ι_u in $\{1, 2, 3\}$, representing which stage we are in. For vertices with $\iota_u \neq 3$, we restrict t_u to be 0. Vertices with $\iota_u = 1$ are owned by who owns ν_u in G, all the other vertices are owned by Even. Vertices with $\iota_u = 3$ have priority $2t$ (resp. $2t + 1$) if $t_u \neq 0$ and the valuation of the register of rank t_u in η_u is even (resp. odd). All the other vertices have priority 1.

The edges are as follows:

- for vertices $u = (\nu_u, \eta, 0, 1)$, there is an edge (u, v) if $v = (\nu_v, \eta, 0, 2)$ and there is an edge in G from ν_u to ν_v.
- for vertices $u = (\nu, \eta, 0, 2)$, there is an edge (u, v) if $v = (\nu, \eta, t_v, 3)$ for any integers in $\{0, \ldots, k\}$.
- for vertices $u = (\nu, \eta_u, t, 3)$, there is an edge (u, v) if $v = (\nu, \eta_v, 0, 1)$ and η_v is the valuation of the registers obtained from first t-resetting and then b-updating $\eta(u)$ where b is the priority of ν.

Register index. It is shown in [16] that G is winning for Even from a vertex u if and only if there is k such that R_G^k is winning for Even from u and register valuation $[0, \ldots, 0]$. In that case, equivalently R_G^k is winning for Even from u and any register valuation. The *register index* of a parity game G winning for Even from every vertex is the minimum k such that R_G^k is winning for Even from all the vertices (with any register valuation). The parity game R_G^k consists of at most $2k+1$ priorities and the number of vertices is also bounded $O(n.k.d^k)$. The main result of [16] was to show that the register index of a game with n vertices is at most $\log n$ if the vertices are winning for Even. This gives a bound on k by $\log\, n$, thereby reducing the game to a parity game on $O(n.\log n.d^{\log n})$. with at most $2\log n + 1$ vertices.

We will attempt to show the above using an alternate method, in relation to progress measures. In the previous section, we saw that all the vertices in a game are winning if and only if there exists a progress measure tree with at most n vertices such that there is a mapping from the vertices to the tree satisfying the progress conditions.

Definition 4. *The Strahler number of a tree T, denoted also by $str(T)$ is 1 plus the maximum height of a complete binary tree that is minor of T.*

Equivalently, it can be defined by induction: the Strahler number of a single node is 1 and the Strahler number of a tree with a root having subtrees of Strahler numbers $k_1, ..., k_\ell$ is the maximum of the k_i if exactly one of them is maximal and is 1 plus this maximum otherwise.

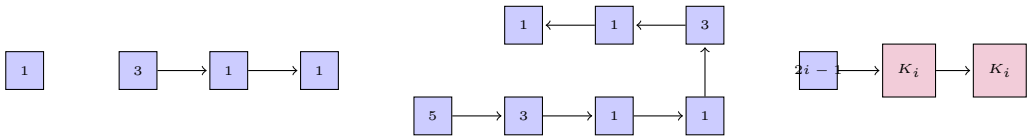


Figure 4.1: Pictures of K_1, K_2, K_3 and K_{i+1}

Theorem 2. *Given a parity game on G on vertices, where Even wins from all the vertices and k is the Strahler number of the progress measure tree of G, then the register game R^l_G is winning for all $l \geqslant k$.*

Proof. We describe the strategy for the register game R^k_G, where $k \geqslant Str(T)$ and argue rigorously in the appendix on why that this strategy is winning for player *Even*.

Description of the strategy Given that we already know that the parity game G has a progress measure associated with it and that progress measure in turn guarantees a positional strategy for the underlying parity game, we will fix such a positional strategy and call it σ. We know that any play which obeys σ will only involve progressive edges. i.e, while moving the token along the vertices, *Even* follows the positional strategy σ, such that on moving from a vertex u to v, the edge (u, v) is progressive.

Now to define a strategy σ' for the register game R^k_G. Notice that a strategy for *Even* can be thought of as having two parts. One which tells the player which edge to take to given the current register values. The other component tells if a register must be reset, and if so which one. The strategy σ' that we define here, does not take into account the value of the register at all. We say that *Even* plays the game such that Even has to move to another vertex, she follows σ. For resetting the register, *Even* does the following: If during the play an edge $u \to v$ is taken such that $\mu(u) \leqslant \mu(v)$, then in the turn to reset right after taking this edge, irrespective of the contents of the register, *Even* resets the k^{th} register where k is one more than the Strahler number of the least common ancestor in T, of the nodes $\mu(u)$ and $\mu(v)$. □

An initial conjecture, one might be tempted to say that the Strahler number of the progress measure tree is equal to the register index. However, the following example shows that the Strahler number of the progress measure tree over-estimates the register index by an arbitrary amount.

Consider the pattern K_1 which just consists of the vertices 1 with and K_i which is the path which consists of the vertex $2i - 1$ followed by two copies of K_{i-1} in succession. As an example, we will construct K_1, K_2, K_3 and K_{i+1} as in Figure 4.1

Now, for each of the graph K_i, add another vertex of priority $2i$ and complete the cycle of K_i by adding edges from this new vertex to the first vertex (vertex with in-degree 0) and from the last vertex (vertex with out-degree 0). We will call these modified graphs K'_i for each i. It is easy to verify that the associated progress measure tree for each of the structures K'_i is the complete binary tree of height $i-1$, thereby making the Strahler index of these graphs at least i for each K'_i. But since each of these graphs is a cycle, they have a register index 1. A simple description for a strategy is for Even to reset the vertex every time right after she sees the highest even priority vertex in this cycle, i.e, the vertex of priority $2i$. In the next section, we will give a tight characterisation of register index.

Chapter 5

Lower Bound of Register Index

Embedding of a game. A game G' is said to be *embedded* in a game G if there is a map ζ from the vertices of G' into the vertices of G such that:

1. ζ is order-preserving on the priorities of the vertices, i.e. if the priority of u is smaller than the priority of v then the priority of $\zeta(u)$ is smaller than the priority of $\zeta(v)$. (Note that different vertices which have the same priority can be sent to vertices having different priorities)

2. ζ is parity-preserving on the priorities of the vertices, i.e. the parities of the priorities of u and $\zeta(u)$ are the same.

3. For all edges from u to v in G', there is a path in G from $\zeta(u)$ to $\zeta(v)$ visiting only vertices of priorities no greater than the maximum of the priorities of $\zeta(u)$ and $\zeta(v)$.

A particular family of parity games. We use here a class of games given in [15] (and proved to have high register index). They are picture in Figure 5.1. The game H_k is defined by induction on k. All the vertices of H_k belong to Odd. The game H_1 contains one vertex of priority 0 and one self-loop.

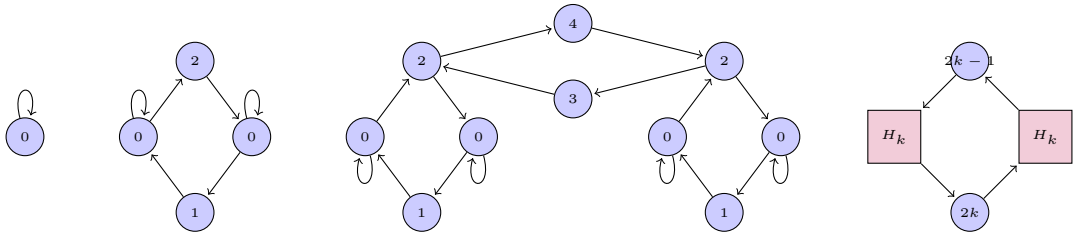


Figure 5.1: Pictures of H_1, H_2, H_3 and H_{k+1}

The game H_{k+1} is made from two copies H_k^0 and H_k^1 of H_k and two new states of priorities $2k-1$ and $2k$, and edges from the vertex of priority $2k-2$ of H_k^0 (resp. H_k^1) to the vertex of priority $2k-1$ (resp. $2k$) and from the vertex of priority $2k-1$ (resp. $2k$) to the vertex of priority $2k-2$ of H_k^1 (resp. H_k^0).

Given a parity game G and a positional strategy σ of Even, G_σ denotes the game G where the moves of Even are restricted to σ.

Theorem 3. *For all non negative integers k, one can decide in polynomial time the winner of a parity game in the following class $\mathcal{C}_k$: the class of parity games G such that H_{k+1} is not embedded in G_σ for some positional strategy σ of Even.*

The proof of Theorem 3 is a consequence of Theorem 4, following ideas from [15] and requires the notion of register games that we introduce now.

Theorem 4. *A parity game G, in which Even wins from all the vertices and Odd owns all the vertices, has register index k if and only if H_k is embedded in G but not H_{k+1}.*

Proof of Theorem 3 assuming Theorem 4. From Theorem 4, we know that $\mathcal{C}_k$ consists of exactly the games which have register index at most k.

So, deciding the winner in a game G of $\mathcal{C}_k$ is equivalent to deciding the winner in the game R_G^k, which has $O(nk(d+1)^k)$ vertices and highest priority $2k+1$, where d denotes the highest priority and n the number of vertices in G. This can be done in time $O\left(nk(d+1)^k\right)^{2k+1}$, which is polynomial in the size of G provided k is fixed. □

In this section, we prove the lower bound of Theorem 4, given in Lemma 2. The upper bound given in Lemma 3 will be proved in Sections 6.

Lemma 2. *Let k be a positive integer. If H_k is embedded in G then the register index of G is at least k.*

Lemma 3. *Let k be a positive integer. If H_{k+1} is not embedded in G then the register index of G is at most k.*

Proof of Theorem 4 assuming Lemmas 2 and 3. If H_k is embedded in G but not H_{k+1} then directly by Lemma 2 and 3, G has register index k. Conversely, if G has register index k, by using the contrapositives of the two lemmas, we obtain that H_k is embedded in G but not H_{k+1}. □

Lemma 2 is a direct consequence of Propositions 1 and 2 below.

5.0.1 Register index of H_k.

The following proposition is proved in [15]. We give the proof for the sake of completeness.

Proposition 1 ([15]). *For all positive integers k, H_k has register index k.*

Proof. Let us first prove that the register index of H_k is at most k. We show by induction that Even has a winning strategy in $R^k_{H_k}$. For $k = 1$, since H_1 consists of one vertex with priority 0 and one self-loop, the only register contains value 0 at all time and if Even resets this register after every move, the highest priority output infinitely often on every infinite play is 2, so Even wins. Now, let $k \geqslant 1$. Let us define a winning strategy for Even in $R^{k+1}_{H_{k+1}}$. The game H_{k+1} consists of two copies of H_k and two vertices of priority $2k$ and $2k-1$. As long as Odd remains in vertices belonging to either of the two copies in H_{k+1}, Even follows the winning strategy given by induction, using only the k registers of lowest rank. After visiting the vertex of priority $2k$ in H_{k+1}, Even resets the register of rank $k+1$ as soon as its value is $2k$. Consider an infinite play in $R^{k+1}_{H_{k+1}}$: if it remains eventually in either of the two copies of H_k, by induction hypothesis, Even wins using only k registers. Otherwise, the play alternates infinitely often between the two copies and thus visits infinitely many times the vertex of priority $2k$ of G. Resetting the register of rank $k+1$ outputs then priority $2k+2$ in $R^{k+1}_{H_{k+1}}$. Moreover, since the register of rank $k+1$ is only reset in this condition, no higher priority is ever output and Even wins $R^{k+1}_{H_{k+1}}$.

Let us now prove that the register index of H_k is at least k. We proceed again by induction on k. We want to prove that H_{k+1} has register index at least $k+1$, that is to say that Odd has a winning strategy in $R^k_{H_{k+1}}$ from at least one vertex with register valuation $[0, \ldots, 0]$. For this, we strengthen slightly the induction hypothesis and we are going to prove that Odd has a winning strategy in $R^k_{H_{k+1}}$ from any vertex with any register valuation. By definition H_1 is of register index 1. Consider now $k \geqslant 1$ and assume that H_k has register index k. The game H_{k+1} is constructed from two copies H^0_k and H^1_k of H_k. Let us define the strategy for Odd in $R^k_{H_{k+1}}$.

(i) Odd follows the strategy given by induction and remains in H^0_k until the register of rank k is reset at least k times.

(ii) Odd then proceeds to the vertex of priority $2k-1$ in G and then to H^1_k.

(iii) Odd follows the strategy given by induction and remains in H^1_k until the register of rank k is reset once.

(iv) Odd then proceeds to the vertex of priority $2k$ in G and then to H^0_k.

If a play eventually remains in one of the two copies of H_k in G, this means that the register of rank k is eventually never reset, so only $k-1$ registers are used and by induction hypothesis Odd wins the game. Otherwise, the play visits infinitely often the vertices of priority $2k-1$ and $2k$ of G. After visiting the vertex of priority $2k$, which is the highest priority, all the registers have value $2k$. Then, in H_k^0, the register of rank k is reset k times, which makes the values of all the registers smaller than $2k$. Then the vertex of priority $2k-1$ is visited and the values of all the registers are updated to $2k-1$. Finally in H_k^1, the register of rank k is reset at least once, which outputs priority $2k+1$, which is odd and the highest priority in the game. By repeating this argument, priority $2k+1$ is output infinitely often and Odd wins the play. □

5.0.2 Register index and embedding.

Proposition 2. *If a game G', for which all the vertices are owned by Odd and winning by Even, has register index k and is embedded in a game G, for which all the vertices are owned by Odd and winning by Even, then G has register index at least k.*

Proof. If $k = 1$ the property is trivially satisfied. Let $k \geqslant 2$. Given a winning strategy for Odd in $R_{G'}^{k-1}$, we define a strategy for Odd in R_G^{k-1}, and prove that it is winning. Let ζ be the embedding from G' to G. We say that a vertex in G is a cornerstone if it is the image of some vertex in G' via the embedding. To avoid confusion, we also denote by $(\nu, \eta, t, \iota)^G$ (resp. $(\nu, \eta, t, \iota)^{G'}$) for a vertex in R_G^{k-1} (resp. $R_{G'}^{k-1}$). We define the strategy inductively on the length of a finite play in R_G^{k-1}. Without loss of generality we can assume that the play starts in a vertex $(\nu, [0, \dots, 0], 0, 1)^G$ where ν is a cornerstone.

Let the vertex ν' of G' be such that $\zeta(\nu') = \nu$. From the vertex $(\nu', [0, \dots, 0], 0, 1)^{G'}$ of $R_{G'}^{k-1}$, the winning strategy gives a vertex $(\mu', [0, \dots, 0], 0, 2)^{G'}$ reached by Odd. By definition of the embedding, there is then a path in G from ν to $\zeta(\mu') = \mu$ visiting only priorities no greater than the maximum of the priorities of ν and μ. This means that Odd has a way to reach a vertex $(\mu, \eta, 0, 1)^G$ in R_G^{k-1} following this path, for some valuation η, independently of which registers are reset by Even on the way. Note that μ is also a corner stone. This defines the strategy for Odd from the first vertex.

Consider now a finite play $\rho_1(\nu, \eta, 0, 1)^G \rho_2(\mu, \gamma, 0, 1)^G$ in R_G^{k-1} where ν and μ are cornerstones and ρ_2 is a play only visiting vertices in G which are not cornerstones. Consider t to be the highest rank of a register reset by Even along ρ_2 having the following property: the valuation of the register of rank t was odd (when reset), or was even and was coming from the priority of a cornerstone vertex. We set $t = 0$ if such a rank does not exist.

By induction, there is a corresponding play in $R_{G'}^{k-1}$ compatible with the winning strategy of the

form $\rho'_1(\nu', \eta', 0, 1)^{G'}(\mu', \gamma', 0, 1)^{G'}$ with $\zeta(\nu') = \nu$, $\zeta(\mu') = \mu$ and γ' obtained from η' where Even has reset the register of rank t (or none if $t = 0$). The winning strategy of Odd in $R^{k-1}_{G'}$ gives a vertex $(\omega', \gamma', 0, 2)^{G'}$ to reach. As before, by the embedding there is a path from μ to $\zeta(\omega') = \omega$ in G visiting only priorities no greater than the maximum of the priorities of μ and ω. This means that Odd has a way to reach a vertex $(\omega, \xi, 0, 1)^G$ in R^{k-1}_G following this path, for some valuation ξ, independently of which registers are reset by Even on the way. This defines the strategy for Odd in R^{k-1}_G.

We prove that this strategy is winning for Odd in R^{k-1}_G. Given an infinite play in R^{k-1}_G following this strategy, there is a corresponding play in $R^{k-1}_{G'}$ compatible with the winning strategy of Odd, and visiting the preimage of the cornerstones visited by the play in R^{k-1}_G. First of all, if Even resets infinitely often in the play in R^{k-1}_G, she must reset infinitely often a register with a valuation coming from the priority of a cornerstone vertex, by property 3. of the embedding. So, in the corresponding run in $R^{k-1}_{G'}$ Even resets infinitely often. Let t be the highest rank of a register reset infinitely often. It has to be on an odd valuation, since the play is compatible with the winning strategy of Odd. By construction, the highest rank of a register reset infinitely often in the play in R^{k-1}_G is then also t and it is reset infinitely often on an odd valuation by properties 1. and 2. of the embedding. □

Chapter 6

Upper bound of register Index

In this chapter, we prove Lemma 3 which would establish an upper bound on the register index.

From now on, we fix a game G, in which Even wins from all the vertices and Odd owns all the vertices.

We first define a notion of *structural forest*, which consists of a forests of ordered trees associated with G and satisfying some specific properties. Each forest comes with a parameter called the *structural number* of the forest. We define then the *structural index* of G as the smallest structural number of a structural forest associated with G. We prove the three following properties, which finishes the proof of Lemma 3:

1. There exists a structural forest associated with G.
2. If H_{k+1} is not embedded in G then all structural forests associated with G has structural number at most k.
3. The register index of G is bounded by its structural index.

In the rest of this section, we give the definition of structural forest and structural index and prove the second and third properties given above. The first property is proved in Section 6.0.1.

6.0.1 Structural forest and structural index

We denote by $\pi(u)$ the priority of a vertex u in G. A vertex u in G is called *play-winning* if there is a play ρ such that u is visited infinitely often and of highest (necessarily even) priority in ρ.

Definition 5. *An ordered finite tree T is called a structural tree of G if there is a function α mapping each node of T to a non-empty subset of play-winning vertices of G such that:*

- *for every node s, the vertices in $\alpha(s)$ have all the same priority, denoted by $\pi(s)$, and if t is a child of s then $\pi(s) > \pi(t)$,*
- *for two distinct nodes s and t, $\alpha(s) \cap \alpha(t) = \emptyset$,*
- *for every node s and t, and every vertex $u \in \alpha(s)$ and $v \in \alpha(t)$, the following condition is satisfied:*
  - *If s is an ancestor of t, then there are paths from u to v and from v to u visiting only vertices of priority at most $\pi(u)$.*
  - *If s and t are siblings with $s < t$ and z denotes their parent, then there is:*
    1. *at least one path in G from u to v in which the highest priority is odd and between $\max(\pi(s), \pi(t))$ and $\pi(z)$,*
    2. *no path from v to u visiting only vertices of priority strictly less than $\pi(z)$,*

Given a finite set F of structural trees T and their associated mappings α_T, we denote by α_F the mapping from the nodes in F to subsets of vertices extending the α_T.

The Strahler number of a tree T is 1 plus the maximum height of a complete binary tree that is minor of T. Equivalently, it can be defined by induction: the Strahler number of a single node is 1 and the Strahler number of a tree with a root having ℓ subtrees of Strahler numbers $k_1, \ldots, k_\ell$ is the maximum of the k_i if exactly one of them is maximal and 1 plus this maximum otherwise. The Strahler number of a node in a tree is defined as the Strahler number of the subtree rooted in this node.

Definition 6. *A structural forest of G is a finite set F of structural trees $T_1, \ldots T_i$ such that:*

- *for all vertices u in G, u is a play-winning vertex if and only if $u \in \alpha(s)$ for some node s,*
- *for two nodes s and t, $\alpha(s)$ and $\alpha(t)$ are disjoint or equal,*
- *for two play-winning vertices u and v in G, if there is a path from u to v with highest priority which is odd and a path from v to u, then all paths from v to u visits a vertex x such that there is a tree T in F, with nodes s, t, z in T such that $u \in \alpha(s)$, $v \in \alpha(t)$ and $x \in \alpha(z)$; z is an ancestor of s and t; s and t are in distinct subtrees rooted in z; and the Strahler number of s (resp. t) is the maximum of the Strahler numbers of s' such that $\alpha(s) = \alpha(s')$ (resp. t' such that $\alpha(t) = \alpha(t')$).*

Structural index of G. We define the *structural number* of a structural forest as the maximum of the Strahler number of a tree belonging to the forest, and the *structural index* of G as the minimum of the structural numbers of structural forests associated with G.

6.0.2 Embedding of H_k and structural index

We prove here the second property.

Lemma 4. *If H_{k+1} is not embedded in G, then every structural forest of G has structural number at most k.*

Proof. We prove by induction on k that if there is a structural tree T associated with G of Strahler index $k+1$ then H_{k+1} is necessarily embedded in G, and more precisely, in vertices in the image of the nodes of T.

The case $k = 0$ is immediate.

Consider now $k \geqslant 0$ and a structural tree associated with G of Strahler number $k+1$. By definition of the Strahler number, there must be a node s with at least two children $t_1 < t_2$ such that the Strahler number of the subtree rooted in s is $k+1$ and the Strahler number of the subtrees T_1 and T_2 rooted in t_1 and t_2 respectively is k. Let $v_1 \in \alpha(t_1)$, $v_2 \in \alpha(t_2)$ and $v \in \alpha(s)$. Since $t_1 < t_2$, by definition of structural tree, there is a path from v_1 to v_2 for which the highest priority is odd and between $\max\{\pi(t_1), \pi(t_2)\}$ and $\pi(s)$. Let u denote a vertex on this path with such odd priority.

By induction hypothesis, H_k is embedded both in the game induced by the images of the nodes in T_1 and of those in T_2. Let η_1 and η_2 the corresponding embeddings. Let η be the mapping from the vertices of H_{k+1} which extends in a natural way η_1 and η_2 and maps the vertex of priority $2k-1$ (resp. $2k$) of H_{k+1} to u (resp. v). It is easy to prove that with this construction, the definition of structural tree and the induction hypothesis, η is an embedding. □

6.0.3 The register index is bounded by the structural index

We prove here the third property.

Lemma 5. *If the structural index of G is k then Even has a winning strategy in $\mathcal{R}_G^k$.*

Proof. Let F be a structural forest associated with G of structural number k. We are going to define a winning strategy for Even in $\mathcal{R}_G^k$. Let u be a play-winning vertex, we denote by $\kappa(u)$ the maximum of the Strahler numbers of the subtrees of F rooted in some s such that $u \in \alpha_F(s)$ (in what follows, we will drop the index F). Let us remind that we denote by $\pi(u)$ the priority of u.

The only vertices in which Even has to make a choice in $\mathcal{R}_G^k$ are those of the form $(u, \eta, 0, 2)$, where Even can decide which register is going to be reset. Let $\rho(u, \eta, 0, 2)$ be a finite play in $\mathcal{R}_G^k$. Let v be the last play-winning vertex of G appearing in ρ such that $\kappa(v) = \kappa(u)$ if it exists. The strategy is as follows:

- If v is not a play-winning vertex or v does not exist then Even does not reset any register.
- If $\pi(v) < \pi(u)$ then Even does not reset any register.
- If $\pi(v) \geqslant \pi(u)$ then Even resets the register of rank $\kappa(v)$.

Let us prove that this strategy is winning in $\mathcal{R}_G^k$. Consider an infinite play ρ. First of all, Even resets infinitely often in ρ since the sets of $\kappa(u)$ and $\pi(u)$ for vertices u are bounded. Let us denote by t the highest rank of a register which is reset infinitely often. We can assume without loss of generality (by going far enough in the play) that:

- no higher ranked register is reset,
- all the vertices u of G visited in ρ are such that $\kappa(u) \leqslant t$,
- all the vertices u of G is ρ such that $\kappa(u) = t$ are visited infinitely often. Let us note $u_1, u_2, \ldots$ for the sequence of those vertices.

Even resets the register of rank t whenever $\pi(u_i) \geqslant \pi(u_{i+1})$, and we are going to prove that in that case, the valuation in the register of rank t is always even. Otherwise, necessarily there exists $i < j$, such that $\pi(u_i) \geqslant \pi(u_j)$ and there is a path from u_i to u_j for which the highest priority p is odd, greater than $\max(\pi(u_i), \pi(u_j))$. By definition of structural forest, since u_i and u_j are visited infinitely often necessarily, in all trees containing nodes which are mapped with u_i or u_j, there are nodes s, t, z and an index ℓ such that $u_i \in \alpha(s)$, $u_j \in \alpha(t)$ and $u_\ell \in \alpha(z)$ and z is an ancestor of s and t, which are not comparable. But then, $\kappa(u_\ell)$ would be greater than t. □

6.1 Existence of a structural forest

In this section, we prove the first property, stating that there always exists a structural forest, concluding the proof of Lemma 3. We construct such a structural forest using the notion of strategy decomposition for parity games that we first recall in Section 6.1.1. We give the construction in Section 6.1.2 and we apply this construction on an example in Section 6.1.3.

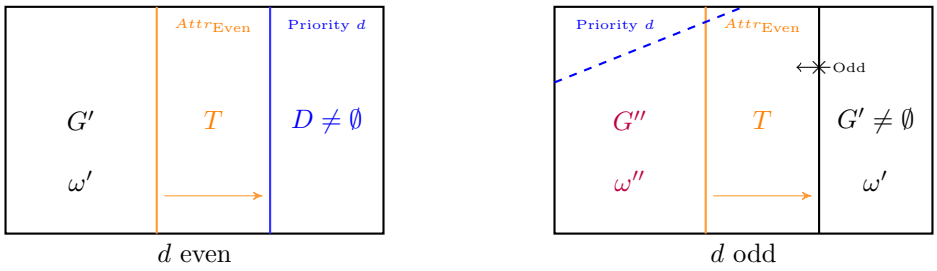


Figure 6.1: Strategy decomposition for Even

6.1.1 Strategy decomposition

We define a strategy decomposition ω for Even of a parity game G by induction (a strategy decomposition for Odd can be defined symmetrically). This is pictured in Figure 6.1.

1. If the highest priority d of G is even then $\omega = \big((G', \omega'), T, D\big)$, such that:
   (a) D is the game induced by the vertices of priority d,
   (b) T is the game induced by the vertices of the attractor of D for Even,
   (c) G' is the game induced by the remaining vertices and ω' is a strategy decomposition of G'.
2. If d is odd then $\omega = \big((G'', \omega''), T, (G', \omega')\big)$, such that:
   (a) G'', T, G' are induced games, partitioning the vertices of G,
   (b) G' is a non-empty trap for Odd,
   (c) T is the attractor of G' for Even,
   (d) ω' and ω'' are strategy decompositions of G' and G'' respectively.

Proposition 3 ([7]). *There is a strategy decomposition for Even of G if and only if Even wins from every vertex in G.*

6.1.2 Construction of a structural forest

We prove now that there exists a structural forest of G. We are going to construct it by induction.

For this, we need to strengthen the induction hypothesis and we introduce the notion of *structural expressions*. Let $\mathcal{A}$ be the following algebra: the base elements are ordered trees and the operators, all binary, are denoted by $\oplus$ and $\triangleright_b$ for all odd positive integers b no greater than the highest priority in G. The operator $\oplus$ is associative and commutative. The operators $\triangleright_b$ are associative in the following strong

sense: for all terms E_1, E_2, E_3, and all odd integers b_1, b_2, we have $(E_1 \triangleright_{b_1} E_2) \triangleright_{b_2} E_3 = E_1 \triangleright_{b_1} (E_2 \triangleright_{b_2} E_3)$. Finally, any $\triangleright_b$ distributes over $\oplus$, i.e. $(E_1 \oplus E_2) \triangleright_b E_3 = (E_1 \triangleright_b E_3) \oplus (E_2 \triangleright_b E_3)$ and symmetrically.

Terms in $\mathcal{A}$ are then either terms of the form $T_1 \triangleright_{b_1} T_2 \triangleright_{b_2} \cdots \triangleright_{b_{n-1}} T_n$ for $T_1, \ldots, T_n$ trees (possibly $n = 1$) or terms of the form $E_1 \oplus E_2$ for E_1, E_2 terms.

Definition 7. *A structural expression associated with G is a term E in $\mathcal{A}$ such that:*

- *the set of trees in E is a structural forest F of G,*
- *if E contains a subexpression $T_1 \triangleright_b T_2$, then for all nodes $s \in T_1$ and $t \in T_2$, and all vertices u of $\alpha(s)$ and v of $\alpha(t)$, there is a path with highest priority b from u to v, and there is no path from v to u.*

The existence of a structural expression implies by definition the existence of a structural forest.

Construction of a structural expression of G. We define now by induction on a strategy decomposition of G:

- a term $\phi(G)$ on $\mathcal{A}$,
- a function α_G mapping nodes in $\phi(G)$ with non-empty subsets of vertices of G.

and we prove that this is a structural expression.

The only base case is when the highest priority in G is even and G' is empty in the strategy decomposition of G. Then let $S_1, S_2, \ldots, S_t$ be the strongly connected components of G with at least two vertices or one vertex and a self-loop, and $S_1^d, S_2^d, \ldots, S_t^d$ their restriction to the vertices of priority d. We define $\phi(G) = s_1 \oplus s_2 \oplus \cdots \oplus s_t$ where each of the s_i is a one node tree; and $\alpha_G(s_i) = S_i^d$. It can be checked that $\phi(G)$ is a structural expression.

Suppose now that the highest priority in G is odd and G'' is empty in the strategy decomposition of G. Then we define $\phi(G) = \phi(G')$ and $\alpha_G = \alpha(G')$. Since none of the vertex in T can be play-winning, and G' is a trap, by induction $\phi(G)$ is a structural expression.

The third case is when G'' is not empty. Then we define $\phi(G) = \phi(G'') \# \phi(G')$ by induction on the structure of $\phi(G'')$ and $\phi(G')$.

- If $\phi(G'') = E_1 \oplus E_2$, then $\phi(G) = (E_1 \# \phi(G')) \oplus (E_2 \# \phi(G'))$ and symmetrically if $\phi(G') = E_1 \oplus E_2$. The function α_G extends naturally $\alpha_{G'}$ and $\alpha_{G''}$. By induction, since the vertices in T are not play-winning then $\phi(G)$ is a structural expression.

- Otherwise, $\phi(G'') = T_1 \triangleright_{b_1} T_2 \triangleright_{b_2} \cdots \triangleright_{b_{n-1}} T_n$ and $\phi(G') = S_1 \triangleright_{c_1} S_2 \triangleright_{c_2} \cdots \triangleright_{c_{m-1}} S_m$ with the T_i's and S_i's being trees (possibly n or $m = 1$). Let I be the set of pairs of indices (i, j) such that there are two nodes s in T_i and t in T_j and two vertices $u \in \alpha_{G'}(s)$ and $v \in \alpha_{G''}(t)$ such that there is a path in G from u to v visiting a vertex of priority d (the highest priority in the game), then

$$\phi(G) = \left(\bigoplus_{(i,j)\in I} T_1 \triangleright_{b_1} \ldots T_i \triangleright_d S_j \triangleright_{c_j} \ldots \triangleright_{c_{m-1}} S_m \right) \oplus \phi(G'') \oplus \phi(G')$$

  The function α_G extends naturally $\alpha_{G'}$ and $\alpha_{G''}$. It is easy to check by induction that $\phi(G)$ is a structural expression.

Finally, the last case is when $\omega = \big((G', \omega'), T, D\big)$ for some nonempty G'. Then let $S_1, S_2, \ldots, S_t$ be the strongly connected components of $T \cup D$ with at least two vertices or one vertex and a self-loop and $S_1^d, S_2^d, \ldots, S_t^d$ their restriction to the vertices of priority d. Then we define $\phi(G) = (\phi(G') \star v_1) \oplus \cdots \oplus (\phi(G'') \star v_t)$ where for all i, v_i is a one node tree, by induction on the structure of $\phi(G')$. For all i,

- If $\phi(G') = E_1 \oplus E_2$, then $\phi(G') \star v_i = (E_1 \star v_i) \oplus (E_2 \star v_i)$. The function α_G extends $\alpha_{G'}$ and maps the copies of v_i to S_i^d.

- Otherwise, $\phi(G') = T_1 \triangleright_{b_1} T_2 \triangleright_{b_2} \cdots \triangleright_{b_{n-1}} T_n$ with the T_i's being trees (possibly $n = 1$). Let I be the set of maximal intervals (ℓ, j) such that there are two nodes s in T_ℓ and t in T_j and two vertices $u \in \alpha_{G'}(s)$ and $v \in \alpha_{G''}(t)$ such that there is a path in G from u to a vertex in S_i and from a vertex in S_i to v, then

$$\phi(G) = \left(\bigoplus_{(i,j)\in I} E_{i,j} \right) \oplus \phi(G')$$

  where $E_{i,j}$ is a tree with root v_i and subtrees $T_i, \ldots, T_j$. The function α_G extends $\alpha_{G'}$ and maps the copies of v_i to S_i^d. It is easily checked that $\phi(G)$ is a structural expression.

6.1.3 An example

We apply the construction given in the previous section to the parity games given in Figure 6.2.

The strategy decomposition of G is (where we dropped the ω and colored the nodes to simplify the reading): $((H_1 \cup H_2, 7, H_3), \emptyset, 8)$, where H_3 can be further decomposed in $((4, 5, 4), \emptyset, 6)$ and $H_1 \cup H_2$ in $((H_1 \cup \{4\}, 5, 4), \emptyset, 6)$ where H_1 is $(((0, 1, 0), \emptyset, 2), \emptyset, 4)$.

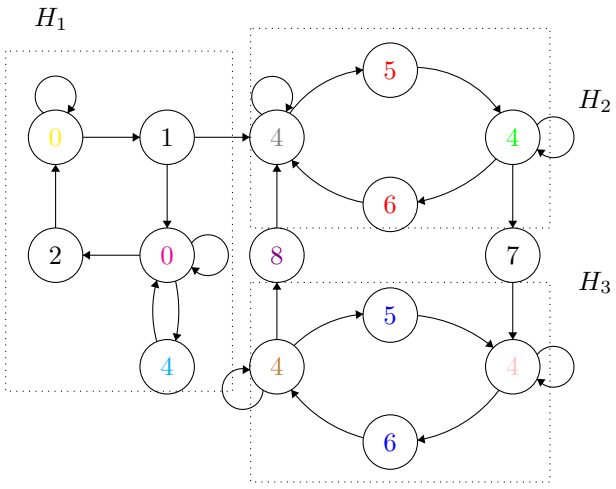
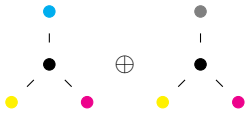


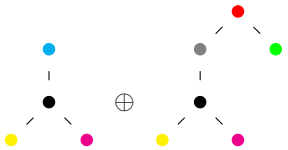
Figure 6.2: Parity game with all vertices belonging to player Odd

The decomposition $(0, 1, 0)$ is associated with the expression $\bullet \rhd_1 \bullet$. The decomposition $((0, 1, 0), \emptyset, 2)$ corresponds then to the tree:

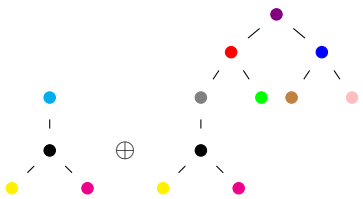
Going further, we get expressions associated with all the decompositions above:

H_1:  H_3:  $H_1 \cup \{4\}$: 

Then $(H_1 \cup \{4\}, 5, 4)$ and $H_1 \cup H_2$ corresponds to:

 and 

Finally, the expression associated with G is:



Chapter 7

Conclusion

In this thesis, the two main results are as follows

- Give an alternating proof of correctness for the quasipolynomial time algorithm proposed in [16]
- Give a characterisation of graphs with a fixed register index k.

The proof of the characterisation also introduces new gadget associated parity game, a structural expression and a structural forest, which is an interesting parameter in its own right and can be studied further in detail to understand what it entails. One could take it upon themselves to define a structural expression (and even a structural forest) for two player games in a meaningful way that might contribute further to the understanding of the game.

Another interesting question is if it is possible to prove Theorem 4 without the digression to structural expression and instead by-passing that.

An important thing to note, however is that this Theorem only considers a one player game. We conjecture that the same results hold when there are two players, and that if at all a register game can be won with a *positional strategy*, then there is a winning strategy.

The ultimate aim could be to understand register games completely and also to bring down the complexity of the algorithm to solve parity games using register indices to the state of the art ones for solving parity games. In [6], we see that there are some restriction on the lower-bounds that one can achieve by this technique and we observe that there is a gap that can be closed for the algorithm proposed in [16].

Bibliography

[1] D. Berwanger, A. Dawar, P. Hunter, and S. Kreutzer. Dag-width and parity games. In *STACS*, pages 524–536, 2006.

[2] D. Berwanger and E. Grädel. Entanglement - A measure for the complexity of directed graphs with applications to logic and games. In *LPAR*, pages 209–223, 2004.

[3] Henrik Björklund, Sven Sandberg, and Sergei Vorobyov. Memoryless determinacy of parity and mean payoff games: A simple proof. *THEORETICAL COMPUTER SCIENCE*, 310:365–378, 2004.

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

[5] Thomas Colcombet and Nathanaël Fijalkow. Parity games and universal graphs. *CoRR*, abs/1810.05106, 2018.

[6] Wojciech Czerwinski, Laure Daviaud, Nathanaël Fijalkow, Marcin Jurdzinski, Ranko Lazic, and Pawel Parys. Universal trees grow inside separating automata: Quasi-polynomial lower bounds for parity games. *CoRR*, abs/1807.10546, 2018.

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

[8] Laure Daviaud, Marcin Jurdzinski, and Ranko Lazic. A pseudo-quasi-polynomial algorithm for solving mean-payoff parity games. *CoRR*, abs/1803.04756, 2018.

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

[10] E. A. Emerson, C. Jutla, and A. P. Sistla. On model-checking for fragments of μ-calculus. *Theoretical Computer Science*, 258(1–2):491–522, 2001.

[11] P. Hunter and S. Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. In *SODA*, pages 637–644, 2007.

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

[13] Marcin Jurdzinski. Small progress measures for solving parity games. volume 1770, pages 290–301, 08 2007.

[14] Marcin Jurdzinski and Ranko Lazic. Succinct progress measures for solving parity games. *CoRR*, abs/1702.05051, 2017.

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

[16] Karoliina Lehtinen. A modal μ perspective on solving parity games in quasi-polynomial time. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 639–648, New York, NY, USA, 2018. ACM.

[17] R. McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.

[18] J. Obdrzálek. Fast mu-calculus model checking when tree-width is bounded. In *CAV*, pages 80–92, 2003.

[19] J. Obdrzálek. Clique-width and parity games. In *CSL*, pages 54–68, 2007.

[20] W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200:135–183, 1998.

Appendix A

Proof of Upper bound of register index

We will see this proof step by step.

Description of strategy: We will first give a strategy for the register game R_G^k, where $k \geqslant Str(T)$ and argue that this strategy is winning for player *Even*. Given that we already know that the parity game G has a progress measure associated with it and that progress measure in turn guarantees a positional strategy for the underlying parity game, we will fix such a positional strategy and call it σ. We know that any play which obeys σ will only involve progressive edges. i.e, while moving the token along the vertices, *Even* follows the positional strategy σ, such that on moving from a vertex u to v, the edge (u, v) is progressive.

Now to define a strategy σ' for the register game R_G^k. Notice that a strategy for *Even* can be thought of, as having two parts. One which tells the player which edge to take to given the current register values. The other component tells if a register must be reset, and if so which one. The strategy σ' that we define here, does not take into account the value of the register at all and if restricted to We say that *Even* plays the game such that even has to move to another vertex, she follows σ. For resetting the register, *Even* does the following: If during the play an edge $u \to v$ is taken such that $\mu(u) \leqslant \mu(v)$, then in the turn to reset right after taking this edge, irrespective of the contents of the register, *Even* resets the k^{th} register where k is one more than the Strahler number of the least common ancestor in T, of the nodes $\mu(u)$ and $\mu(v)$.

Note that if $\mu(u) \leqslant \mu(v)$ and the edge is progressive, it must be the case that $\pi(u)$ is even. With this observation in hand, we will show the following inducting on the height of the tree associated with

the progress measure in two different cases. When the strahler index is 1 and if it is > 1.

Induction Hypothesis: Suppose height of the tree T is h,

(a) For Strahler number k and if the k^{th} register contains an even value in it, greater than or equal to $2h$, then for any play, on following the strategy σ', player *Even* would only reset the register when the register of rank k contains an even number.

(b) Any play following the strategy σ' in the register game R_G^k is winning

Strahler index of the tree is 1

For each of these, we will proceed by induction on the height of the tree corresponding to the progress measure of the game. We will slightly strengthen the induction hypothesis. Other than the two conditions (a) and (b), we will also add the following for the case:

(c) For trees with Strahler index 1, if the only register contains an even value in it, or if it contains any value less then $2h$, then a play following σ' would always output an even number on resetting that register.

Let us prove the above statement for games G with Strahler index 1. If Strahler index of the graph is 1, then the tree T associated with the progress measure must look like a path.

We will proceed by induction on the height of the tree T. Note that the height of the tree would be $2.d$ where d is the highest priority in the game, which we will assume to be even, without loss of generality.

Base Case: If the height of the tree is 0, then we know that the only priorities seen is 0. This ensures that the strategy σ' so defined, resets the register which sees 0 at every step. This clearly satisfies both (a) and (b) of the statement. *Induction step:* Now assume the height of the tree is h. Also, the induction hypothesis holds for plays where the height of the tree (of Strahler index 1) is strictly smaller than h, the conditions (a), (b) and (c) are satisfied. We will show that the statement is true for h also. Suppose we have a play satisfying σ', we know that if the play was such that its corresponding values on the tree T (Or the projection of the play on T) only visited the root of the tree, i.e, all the edges (u, v) taken are such that $\mu(u)$ and $\mu(v)$ correspond to the root of T, then we know that the priority $\pi(u)$ must be even, since the edge is progressive and also that $\mu(u) = \mu(v)$. This also means that from our definition of σ', the register is reset every time after visiting u and proceeds to v.

Suppose the play does not stay on the root T and remains completely within the subtree of T, without reaching the root infinitely often, then by induction hypothesis, this play would satisfy (a), (b) and (c). If the play visits T, then we consider the position where the play reaches the root of the

tree T, from which it proceeds to visit other nodes of T, we again note that if there is an edge between u and v such that $\mu(u) \leqslant \mu(v)$ and when (u, v) is progressive, then it must be the case that again such an edge has priority even, and since we know that $\pi(u)$ is at least twice the height of the node $\mu(u)$ in the tree, we can again see that the strategy σ' would always reset the register after such an edge (u, v). If the register already contained an even number or a value smaller than $2.h$, then σ' on taking the edge (u, v) would definitely contain an even number at least as large as $2.h$.

Since σ' is such that *Even* resets the register only after seeing an edge (u, v) such that $\mu(u) \leqslant \mu(v)$ and also from the fact that (u, v) is progressive, *Even* would reset only when $\mu(u) = \mu(v)$ or $\mu(v)$ is a child of $\mu(u)$. It is easy to see that if we have $\mu(u) \leqslant \mu(v)$ for a progressive edge it must be the case that $\pi(u)$ is even and again, if the register already has index that is even or less than $2.h$, then σ' would only output even values at that point on resetting the register, and once the game proceeds to the lower parts of the tree, the register is reset, so it has a value lower than the height of the tree and therefore, by induction, we arrive at play on a tree T of strictly smaller height. Now as long as the play remain within T, by induction hypothesis, we know the statements (a), (b) and (c) hold. If the play reaches the root, we know that the reset only outputs an even number. Therefore, we have shown the induction hypothesis.

Strahler index of tree is > 1

Now we proceed to the case where the Strahler index is at least two, and we will prove the following similarly by induction on the height of the tree with Strahler index T and making (a) and (b) as our statement of the induction hypothesis. In the proof, we will always assume without loss of generality that the maximum priority d is even.

Suppose, for some d such that $d \geqslant 2$, if all the priorities occurring in a play following σ' in the game graph G are bounded by d, then we have that the register game R^k_G is winning for player *Even* on that play. Other than that, if the configuration of the registers at the beginning of the play is such that the register with rank equal to the Strahler index contains d or a higher even priority, resetting that register always outputs an even value.

We will show that the induction hypothesis holds while following the strategy σ'

Base case: h=1

Suppose $h = 1$, then the ordered tree T would have a root as a node and all other nodes of T would be children of T.

The Strahler number of T here is 2 if the tree has more than one child. We need to show that the above strategy σ' which would use two registers is winning for *Even* from all vertices. But that

is because the only time we reset registers is when we have an edge such that the $\mu(u)$ is the root and $\mu(v)$ is its child. Those edges occur only after we see edges of priority 2. This must mean that the register that is being reset would also contain at least 2, which is an even number, and therefore the register would output an even number. We can also guarantee that since we are always taking progressive edges, the register would be reset infinitely often as the progress measure of vertices in a play cannot always continue to increase. Suppose the register contained an even number, then we can also see that on following σ', resetting the register always outputs an even value.

Induction step

Now, we will assume the hypothesis for $d \geqslant 2$ and prove it for a play on G where the highest priority is $d+2$. Let T be the ordered tree associated with the progress measure on G. T looks like (inset picture) where $T_1, T_2, \ldots T_l$ are the sub-trees whose corresponding Strahler numbers are $k_1, k_2, \ldots, k_l$.

Case I: Strahler number of T is equal Straher number of $\max\{k_1, \ldots, k_l\}$ Consider a play denoted by $v_0, v_1, \ldots, v_i, \ldots$. Look at the projection of this play to nodes of tree T, i.e, $\mu(v_0), \mu(v_1), \ldots, \mu(v_i), \ldots$. If this sequence completely stays inside the sub trees T_i, then we know that all the priorities seen by in this play would be at most d, and therefore, by induction, we know that the play is winning for *Even* and also that every time the register of highest rank is reset, it outputs an even number.

If not, we know that the play reaches nodes where u, where $\mu(u)$ is the root of the tree T after which the projection of the play either stays at the root or it enters a sub-tree T_j If the play stays at $\mu(u)$, then a simple argument will show that the edge taken to do so always has priority $d+2$ and the register is reset after each time the edge is taken and therefore our induction hypothesis holds as resetting the register of rank k outputs an even value. If not, then the edge (u, v) taken is such that $\mu(u)$ is the root of the tree T and v is such that $\mu(v)$ is an element in T_j. Since such an edge can be taken only if $\mu(u) \leqslant \mu(v)$ and this edge is progressive, so we also have $\mu(u) \mid_{\pi(u)} \geqslant \mu(v) \mid_{\pi(u)}$. This must mean that $\pi(u)$ is even and from the position of $\mu(u)$ in T, we know that $\pi(u) \geqslant d+2$. This ensures that on entering the sub tree T_j below, we would have reset the register of rank k, which would have had an even value at least as large as $d+2$. So, when the register of rank k is reset, it would produce only an even number. On entering the sub-tree T_j, the value of each register with rank greater than 1 (We assumed that there are at least 2 registers) is an even value at least as large as $d+2$ too. While the game is restricted to the sub-trees, the largest rank register never outputs an odd value when reset, as there is exactly one sub-tree of Strahler index the same as the root and no other subtree would reset the register of the highest rank within this subtree. This ensures, that by induction hypothesis and the fact

that there is a unique sub-tree with Strahler number k that if the play thereon does not visit the node of T would output an even number of resetting the register of the highest rank. Since, while visiting the root at T, we have seen that the edge taken while leaving the root has priority $d+2$, and therefore resetting the register there, we see an even value as output.

Case II: Strahler number of T is one more than the Straher number of $\max\{k_1, \ldots, k_l\}$ This is a simpler case, as we know that if the root of the tree is never visited at all, then the priorities seen by a play would not contain $d+1$ and $d+2$ and none of the sub-trees would reset the register of index k. Otherwise, then we know that the only time the register of index equal to the Strahler number is reset, is when an edge (u, v) where $\mu(u)$ corresponds to the root of the tree T is taken. We have already seen that such a vertex u must have priority $\pi(u) = d+2$. This makes sure that on following strategy σ', where after taking the edge (u, v) the k^{th} register is reset, all the other registers would have an even priority which is at least as large as $d+2$. Again, we have shown the statement that the largest register outputs an even value only. To show that *Even* wins the game is easy, since this follows by induction hypothesis. Since while entering each sub-tree, the registers with the Strahler number have even values in them, and by induction hypothesis we have that any play following σ' must be winning for *Even*