

On the Complexity of Parity Games

Arnold Beckmann Faron Moller
Department of Computer Science
Swansea University
{A.Beckmann, F.G.Moller}@swansea.ac.uk

Abstract

Parity games underlie the model checking problem for the modal μ -calculus, the complexity of which remains unresolved after more than two decades of intensive research. The community is split into those who believe this problem – which is known to be both in NP and coNP – has a polynomial-time solution (without the assumption that $P = NP$) and those who believe that it does not. (A third, pessimistic, faction believes that the answer to this question will remain unknown in their lifetime.)

In this paper we explore the possibility of employing Bounded Arithmetic to resolve this question, motivated by the fact that problems which are both NP and coNP, and where the equivalence between their NP and coNP description can be formulated and proved within a certain fragment of Bounded Arithmetic, necessarily admit a polynomial-time solution. While the problem remains unresolved by this paper, we do propose another approach, and at the very least provide a modest refinement to the complexity of parity games (and in turn the μ -calculus model checking problem): that they lie in the class PLS of Polynomial Local Search problems. This result is based on a new proof of memoryless determinacy which can be formalised in Bounded Arithmetic.

The approach we propose may offer a route to a polynomial-time solution. Alternatively, there may be scope in devising a reduction between the problem and some other problem which is hard with respect to PLS, thus making the discovery of a polynomial-time solution unlikely according to current wisdom.

Keywords: Parity games, Total NP Search Problems, Polynomial Local Search, PLS

1. INTRODUCTION

Infinite two-player games played on finite directed graphs play an important role in various problems within the field of verification of systems, specifically with regard to automata theory, modal and temporal logics, and monadic second-order logics [1]. Of particular interest are so-called *parity games*, in which each node of the graph is owned by one of the two players and labelled by some nonnegative integer. There is at least one outgoing edge from every node, and a play consists of moving a token from some initial node along an infinite path through this graph, with the token moved from each node by the player owning that node along an outgoing edge of their choosing. One of the two players tries to ensure that the minimal label encountered infinitely often in the path is even, and is declared to be the winner in such a play; otherwise the other player is declared to be the winner.

Following Stirling [2], we will assume (without loss of generality) that:

- no two nodes have the same label; that is, we can associate the set of nodes with a finite subset of the natural numbers; and
- the nodes owned by the player seeking to ensure that the minimal label encountered infinitely often is even are precisely those nodes with even labels.

In this way, the winner of a play is the owner of the smallest node which is encountered infinitely often. With little thought it can be realised that any parity game can be transformed into an

equivalent such simple graph game without substantially increasing the size of the underlying graph (in fact, by no more than a linear factor).

The **determinacy** of parity games – ie, the fact that one of the two players has a winning strategy in any parity game – follows from the determinacy of the far more general Borel Games as established by Martin [3]. Furthermore, the problem of deciding which of the two players has a winning strategy in any parity game is polynomial-time equivalent to the emptiness problem for alternating tree automata as well as the model checking problem for the μ -calculus [4]. For this reason, there has been much interest in devising efficient algorithms for determining who has the winning strategy in parity games.

This task is made simpler by the realisation that parity games satisfy **memoryless determinacy**, which is to say that a winning strategy exists for one of the two players which is strictly **positional**: the fortunate player in possession of the winning strategy can decide in advance what move to make from each node which they own. They need not consider the history of the play at the time of the move, and they can even reveal this strategy to their opponent at the start of play. This realisation is easily – if somewhat inaccurately – motivated: since the winning condition is concerned solely with which nodes appear infinitely often in a play, the nodes which have appeared in the finite history of a play are of no consequence, and there is no reason to choose different outgoing edges from a given node each time it is encountered. This intuition suffices for an undergraduate Computer Science lecture, particularly in light of the complexity of the formal proofs that have been devised, but apart from its informality, it is generally unsound (when considering more complicated winning conditions).

Emerson [5] outlined the first formal proof of memoryless determinacy for parity games, with further proofs given by Emerson and Jutla [4], McNaughton [6] and Zielonka [7]. Somewhat earlier, Ehrenfeucht and Mycielski [8] demonstrated memoryless determinacy for the related mean payoff games. More recently, Björklund et al. [9] have devised an elegant and elementary proof of memoryless determinacy of parity, and mean payoff, games based on induction over the size of such games. In actual fact, they prove memoryless determinacy for a finite variant of parity games, and relate this directly to the same property of the standard infinite version.

Once memoryless determinacy of parity games is established, it becomes immediately apparent that the problem of determining if one of the players has a winning strategy for a given parity game is in NP: guess a positional winning strategy for this player, and verify that this strategy is indeed a winning strategy. This verification relies on simply confirming that – assuming the player uses this supposed winning strategy – the other player cannot force a loop through the graph in which this other player owns the smallest node in the loop: such a loop would allow this other player to win the game; and such a loop would have to exist in order for this other player to win.

The next equally clear observation is that the problem is also in coNP, due to the symmetric roles of the two players. Being in $NP \cap coNP$ then naturally leads to the question as to whether this problem can in fact be solved in polynomial time. However, all efforts to find such an efficient algorithm have proved futile; the tightest known bound, a slight improvement to the above due to Jurdziński [10], is that the problem lies in $UP \cap coUP$. Jurdziński et al. [11] have, however, devised a subexponential algorithm for the problem.

One further observation can be usefully made at this point: the decision problem of determining who has the winning strategy in a parity game is equivalent with respect to polynomial-time reducibility to the search problem of determining a memoryless winning strategy. The reduction in one direction is obvious: if we can compute a winning strategy in polynomial time, then we can immediately determine to whom it belongs. For the other direction, we assume that we have an algorithm for determining who has a winning strategy in a parity game, and show how to use this to compute such a strategy. Specifically, for each node from which its owner has a winning strategy, we repeatedly remove outgoing edges until we find one whose removal results in that player no longer having a winning strategy (which may well be the last outgoing edge available).

This edge represents an appropriate move for this player to make as part of the winning strategy. This equivalence with respect to polynomial-time reducibility allows us to make a statement about the decision problem by studying the search problem, which we will do in the following.

Bounded Arithmetic is a logical framework suitable for reasoning about polynomial-sized objects based on various restrictions on the use of induction. As we will employ it, Bounded Arithmetic has been introduced by Buss [12] who established the first elegant connections between Bounded Arithmetic theories and computational complexity classes. Where other logical frameworks like descriptive complexity are naturally related to decision problems, Bounded Arithmetic directly connects to search problems and (multi-)functions. It is well-known that one can always reduce a search problem to a related decision problem which often produces a polynomially-equivalent decision problem. But in general, this may not be the case [13]. An important class of search problems are total NP search problem in the sense of Beame et al. [13]. They are given by a binary, polynomial time computable relation R which is polynomially balanced, ie. any pair (x, y) satisfying R has the property that $|y| \leq |x|^{O(1)}$, and total, ie. for any x there is a y with $R(x, y)$. The search task is for a given input x to find y with $R(x, y)$. The combinatorial principles guaranteeing totality allow NP search problems to be grouped into complexity classes for search problems (see [13] for a discussion of this). Johnson et al. [14] introduced, amongst others, one such class called Polynomial Local Search (PLS). PLS consists essentially of optimisation problems for which polynomial-time local-search heuristics exist.

We have already pointed out that the problem of determining who has the winning strategy from a node in a parity game is equivalent with respect to polynomial-time reducibility to the problem of determining a memoryless winning strategy which stores for each node the winning move (for the player who wins from that node). The latter is a typical example of a total NP search problem in the above sense, which we denote by

MEMDET: Given a graph G , find a positional winning strategy for G .

In Bounded Arithmetic, the combinatorial principle to prove totality of search problems, which is directly at hand, is induction. The meta-theory of Bounded Arithmetic then puts the search problem into a related complexity class. In particular, we will utilise the following well-known theorems for Bounded Arithmetic: Buss [12] has shown that the NP search problems whose totality can be shown using induction of polynomial length on NP properties (a theory denoted S_2^1) can already be solved in polynomial time. This means that if memoryless determinacy would be provable in this theory, then determining who has the winning strategy in a parity game would necessarily be a polynomial-time problem. Buss and Krajíček [15] have (implicitly) shown that the NP search problems whose totality can be shown using induction of polynomial length on NP^{NP} properties (a theory denoted S_2^2) are in PLS. We will utilise the latter theorem, together with a new proof of memoryless determinacy which can be formalised in S_2^2 , to show that the problem of computing a positional winning strategy in a simple graph game is in PLS.

The remainder of this paper is structured as follows. In the next three sections we formalise the above discussion. Specifically, in section 2 we formally define simple graph games, the variant of parity games that we study; in section 3 we formally define strategies and related notions as well as formally state the memoryless determinacy theorem; and in section 4 we provide a brief introduction to the bounded arithmetic which we employ. In section 5 we carefully present a proof of memoryless determinacy within bounded arithmetic, ending with the corollary which is the main result of the paper, placing the problem in PLS. Finally in the concluding section 6 we reflect on our achievements.

2. SIMPLE GRAPH GAMES

In this section we provide the formal definitions of the simple graph games which we shall study, which are played between two players P_0 and P_1 . As noted above, these are equivalent to parity games as typically defined in the literature.

Definition 2.1 (Graph Games). $G = (V_0, V_1, E)$ is a **graph game of size n** if

1. V_i are the positions of player P_i , for $i = 0, 1$. They have to satisfy $V_0 \cap V_1 = \emptyset$, and $V_0 \cup V_1 \subseteq \{1, \dots, n\}$. $V := V_0 \cup V_1$ is the set of all positions.
2. $E \subseteq V \times V$ is the set of possible moves.
3. In graph-theoretic terms, V is the set of nodes, and E the set of edges of graph G . They have to satisfy in addition that at least one edge is leaving each node.

We let \mathcal{G}_n be the collection of all graph games of size n , and use $G = (V_0, V_1, E)$ to range over \mathcal{G}_n . Finally, for $v \in V_i$ we say that player P_i **owns** v .

Definition 2.2 (Playing and Winning). A **play** from a node $v \in V$ is an infinite path $v = v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \dots$ in G with each edge $v_i \rightarrow v_{i+1} \in E$ chosen by the player owning v_i . The **winner** of a play is the player owning the least node which is visited infinitely often in the play.

3. MEMORYLESS DETERMINACY

Instead of defining strategies for a particular player, we will consider general strategies and ignore the moves of the opposite player.

Definition 3.1. A partial map $\sigma: V \xrightarrow{P} V$ is a **pre-strategy for G** if $(\forall v \in \text{dom}(\sigma)) (v, \sigma(v)) \in E$. It is a **strategy** if it is total.

For a strategy σ we let $\sigma_i := \sigma \upharpoonright_{V_i}$ denote the restriction of σ to the moves of player P_i .

Definition 3.2. If σ is a pre-strategy for G , then G^σ denotes the subgraph of G in which the moves from the nodes of $\text{dom}(\sigma)$ are fixed by σ .

Lemma 3.3. P_i wins from v in G using strategy σ iff the least node in any cycle reachable from v in G^{σ_i} is owned by P_i .

Proof. If a cycle is reachable from v in G^{σ_i} in which the least node is owned by P_{1-i} , then P_{1-i} can use this cycle to win the game. Conversely, if P_{1-i} has a winning play from v in G when P_i uses strategy σ , then the least node which appears infinitely often in this play is owned by P_{1-i} and must eventually be the least node which appears on a cycle from that node to itself in the play. \square

Definition 3.4. Let $\text{win}_i(\sigma, G, v)$ denote that σ is a strategy in G , $v \in V$, and every cycle reachable from v in G^{σ_i} is won by P_i (ie, the least node on the cycle is owned by P_i).

Definition 3.5 (Winning Positions). Let

$$W_i(G, \sigma) := \left\{ v \in V : \text{win}_i(\sigma, G, v) \right\}$$

be the set of nodes from which player P_i can force a win using σ , and

$$W_i(G) := \left\{ v \in V : (\exists \sigma) \text{win}_i(\sigma, G, v) \right\}$$

be the set of nodes from which P_i can force a win using some strategy.

Observe that although $\text{win}_i(\sigma, G, v)$ and $W_i(G, \sigma)$ are defined for arbitrary strategies σ , ie. for total maps $\sigma: V \rightarrow V$, the behaviour of σ on V_{1-i} is totally irrelevant by definition of win_i and W_i . Thus, we could have defined them equally well for partial strategies σ with $\text{dom}(\sigma) = V_i$.

Definition 3.6 (Winning Strategy). Let $\text{det}(G, \sigma)$ denote that σ determines the game G , ie., that σ is a strategy for G and that

$$(\forall v \in V) \left(\text{win}_0(\sigma, G, v) \vee \text{win}_1(\sigma, G, v) \right)$$

Thus, $(\exists \sigma) \text{det}(G, \sigma)$ expresses memoryless determinacy for G .

Again we point out that, due to our definition of $\text{win}_i(\sigma, G, v)$, $(\exists\sigma) \text{det}(G, \sigma)$ describes classical memoryless determinacy: from any given node, one of the two players has a positional winning strategy **regardless of the strategy (memoryless or not) of the other player**.

Theorem 3.7 (Memoryless Determinacy). $(\forall G)(\exists\sigma) \text{det}(G, \sigma)$

Corollary 3.8. *The predicate $v \in W_i(G)$ is in $\text{NP} \cap \text{coNP}$*

Proof. By Memoryless Determinacy we know $W_0(G) \cup W_1(G) = V$; and by definition of games we have $W_0(G) \cap W_1(G) = \emptyset$. Hence:

$$\begin{aligned} v \in W_0(G) &\leftrightarrow (\exists\sigma) \text{win}_0(\sigma, G, v) \\ v \notin W_0(G) &\leftrightarrow (\exists\sigma) \text{win}_1(\sigma, G, v) \end{aligned}$$

The problem of determining $\text{win}_i(\sigma, G, v)$, relying only on checking cycles, is clearly in P. □

4. BOUNDED ARITHMETIC

In this section we briefly review basic definitions and results of Bounded Arithmetic which are necessary for the understanding of our paper. For a full introduction we refer the interested reader to Buss [12].

Bounded Arithmetic as defined in [12] is a collection of theories formulated in a language of arithmetic over first-order logic with equality. The language \mathcal{L}_{BA} of Bounded Arithmetic is given by the following set of non-logical symbols:

$$0, s, +, \cdot, | \cdot |, \lfloor \frac{1}{2} \cdot \rfloor, \#, \leq$$

The first four symbols denote, in their standard interpretation over the natural numbers \mathbb{N} , the constant zero, the unary successor function, addition and multiplication; $|x|$ computes the length of the binary representation of x ; the binary “shift right” function $x\#y$, which computes $2^{|x|-|y|}$, produces polynomial growth rate; and the final symbol denotes the “less than or equal” relation. To smoothen our presentation we allow some further non-logical symbols denoting further polynomial-time computable functions. This is unproblematic as our base theory can define *all* polynomial-time computable function—Buss [12] has shown that such an extension of the language is conservative. The additional function symbols we will use are

$$\dot{-}, (x)_y, \langle \rangle, *, **, \text{lh}$$

These denote the arithmetical minus function $x \dot{-} y = \max(0, x - y)$ and functions to manipulate sequences. We assume some feasible sequence coding as defined, e.g., in [12] which assigns a single number to a sequence of numbers. Then $(x)_y$ denotes the y th element in sequence x , $\langle \rangle$ denotes the code of the empty sequence, $x * y$ denotes the function which appends number x to sequence y , $x ** y$ denotes concatenation of two sequences x and y , and $\text{lh}(x)$ denotes the length of the sequence x .

Terms and formulae over \mathcal{L}_{BA} are defined as usual for first-order logic, using the logical symbol “=” for equality and logical connectives $\neg, \wedge, \vee, \rightarrow, \forall, \exists$ for negation, conjunction, disjunction, implication, and first-order universal and existential quantification.

Among the first-order formulae built over \mathcal{L}_{BA} , so called bounded formulae play an important role. Bounded quantifiers can be introduced as abbreviations as follows:

$$\begin{aligned} (\forall x \leq t)A &\text{ abbreviates } (\forall x)(x \leq t \rightarrow A) \quad \text{and} \\ (\exists x \leq t)A &\text{ abbreviates } (\exists x)(x \leq t \wedge A) \end{aligned}$$

where t is a term of the language not containing x . **Bounded formulae** are then formulae in which all quantifiers are bounded, while **sharply bounded formulae** are bounded formulae in which all bounded quantifiers are of the form $(\forall x \leq |t|)$ or $(\exists x \leq |t|)$.

Buss [12] has introduced classes of bounded formulae Σ_i^b and Π_i^b which correspond to the complexity classes Σ_i^p and Π_i^p in the polynomial-time hierarchy. For an exact definition we refer the reader to Buss [12, p.20]; we will only use the following observations:

- Formulae of the form $(\exists x_1 \leq s_1)\varphi(x_1)$ for sharply bounded φ are in Σ_1^b .
- Formulae of the form $(\forall x_1 \leq s_1)(\exists x_2 \leq s_2)\varphi(x_1, x_2)$ for sharply bounded φ are in Π_2^b .

Bounded Arithmetic theories are defined by stating axioms defining the non-logical symbols plus some axiom scheme – the latter being responsible for the strength of the theory. Buss [12] has defined a set BASIC of open formulae defining non-logical symbols which can be extended to define also our additional non-logical symbols. The induction used in Bounded Arithmetic is given as a restriction of the usual induction given by

$$\varphi(0) \wedge (\forall x)(\varphi(x) \rightarrow \varphi(x+1)) \rightarrow (\forall x)\varphi(x)$$

For a set of formulae Φ , we use Φ -LIND to denote the set of formulae of the form

$$\varphi(0) \wedge (\forall x)(\varphi(x) \rightarrow \varphi(x+1)) \rightarrow (\forall x)\varphi(|x|)$$

where $\varphi \in \Phi$. The theories of Bounded Arithmetic are then built by picking a set of formulae and an induction scheme, and forming the theory BASIC + all instances of induction for formulae from the chosen set. From all possible choices, we will use the following:

$$S_2^1 = \text{BASIC} + \Sigma_1^b\text{-LIND}$$

$$S_2^2 = \text{BASIC} + \Sigma_2^b\text{-LIND}$$

In essence, theory S_2^1 expresses reasoning with polynomial-sized objects using induction on NP properties of polynomial length, while theory S_2^2 expresses reasoning with polynomial-sized objects using induction on NP^{NP} properties of polynomial length.

Buss [12] has also shown that certain different induction schemes are equivalent. In particular, we have the following:

Theorem 4.1 (Buss [12]). S_2^2 and BASIC + Π_2^b -LIND prove the same formulae.

4.1. Definable functions and search problems

Definable functions and search problems form an important notion for Bounded Arithmetic as they provide the link between theories and complexity classes. A function is said to be Σ_1^b -definable in theory T if there is a Σ_1^b -formula φ defining the graph of the function such that the unique existence of the value depending on its arguments can be proven in T . A predicate or formula is said to be Δ_1^b -definable in T if there is a Σ_1^b -formula and a Π_1^b -formula which both define the predicate and whose equivalence can be shown in T .

Theorem 4.2 (Buss [12]). *The Σ_1^b -definable functions in S_2^1 coincide with FP, the class of functions computable in polynomial time. The Δ_1^b -definable predicates in S_2^1 are exactly those in P, the class of predicates decidable in polynomial time.*

Let R be a total NP search problem. We say that R is Δ_1^b -definable in a theory T iff there exists a formula φ which is Δ_1^b -definable in T such that T proves the totality of R via φ , i.e. $T \vdash (\forall x)(\exists y)\varphi(x, y)$.

A multi-function is said to be Σ_1^b -definable in theory T if there is a Σ_1^b -formula φ defining the graph of the multi-function, such that the existence of the value depending on its arguments can be proven in T .

Theorem 4.3 (Buss, Krajíček [15]). *The Σ_1^b -definable multi-functions in S_2^2 are exactly the projection of problems in PLS.*

In terms of search problems, one can extract from the proof of this theorem that the Δ_1^b -definable total search problems in S_2^2 are exactly the problems in PLS.

5. PROVING MEMORYLESS DETERMINACY IN BOUNDED ARITHMETIC

In this section, we prove the main results of our paper. Our goal is to formalise a proof of Memoryless Determinacy in Bounded Arithmetic in order to obtain some new information about the complexity of the NP search problem **MEMDET**. For this, we first discuss how simple graph games and the definitions surrounding them can be formalised in Bounded Arithmetic. As we have sequence coding and decoding at hand as function symbols in Bounded Arithmetic, we can literally translate Definitions 2.1 and 3.1 to obtain the following formulae in \mathcal{L}_{BA} : $\text{GraphGame}(G)$ is the sharply bounded formula expressing that $\text{lh}(G) = 4$ and that $((G)_0, (G)_1, (G)_2)$ is a graph game of size $|(G)_3|$ according to Definition 2.1, where sets of vertices and edges are stored in some canonical way as sequences. We can also define the set of nodes and the size of a graph as terms in \mathcal{L}_{BA} by defining the nodes of G to be $(G)_0 ** (G)_1$, and the size of G , $\text{size}(G)$, to be $|(G)_3|$. We define $\text{last}(x)$ as the term $(x)_{\text{lh}(x)-1}$, and $\text{member}(x, y)$ as the formula $(\exists z < |y|) x = (y)_z$. Here, $|y|$ is sufficient as $\text{lh}(y) \leq |y|$ follows immediately from the axioms of Bounded Arithmetic.

“Partial maps from V to V ” can be formalised as sequences of pairs, and we immediately have the following sharply bounded formulae:

- $\text{PreStrategy}(G, \sigma)$ denotes that σ is a pre-strategy for G ;
- $\text{Strategy}(G, \sigma)$ denotes that σ is a strategy for G ;
- $\text{StrategyRestriction}(G, \sigma, i, \mu)$ denotes that $\mu = \sigma \upharpoonright_{V_i}$; and
- $\text{GraphRestriction}(G, \sigma, G')$ denotes that if $\text{PreStrategy}(G, \sigma)$ then $G' = G^\sigma$, otherwise $G' = \emptyset$.

It is obvious that S_2^1 can prove that formulae $\text{StrategyRestriction}(G, \sigma, i, \mu)$ and $\text{GraphRestriction}(G, \sigma, G')$ define graphs of functions, ie. that μ in the former and G' in the latter can be shown to exist uniquely in S_2^1 .

A bit more interesting is the formalisation of $\text{win}_i(\sigma, G, v)$ according to Definition 3.4. As we know that reachability in graphs is polynomial-time computable, it is easy to translate this Definition into Bounded Arithmetic. But this time we cannot simply translate it as a sharply bounded formula, but have to use the full polynomial-time expressivity available in S_2^1 . We proceed as follows:

- $\text{preReach}(G, v, L)$ denotes that if $\text{GraphGame}(G)$ then L is a sequence of length $\text{size}(G)+1$ with $(L)_0 = \langle v \rangle$ and $(L)_{i+1}$ is $(L)_i$ plus all nodes in G which can be reached in one step from $(L)_i$, and $L = \emptyset$ otherwise; and
- $\text{Reach}(G, v, w)$ denotes $(\exists L)(\text{preReach}(G, v, L) \wedge \text{member}(w, \text{last}(L)))$.

We are omitting here and in the following bounds to quantifiers if it is obvious how they can be defined. It is not hard to see that S_2^1 can prove that $\text{preReach}(G, v, L)$ defines the graph of a function computing L on input G and v . Thus, Reach is Δ_1^b in S_2^1 because

$$S_2^1 \vdash \text{Reach}(G, v, w) \leftrightarrow (\forall L)(\text{preReach}(G, v, L) \rightarrow \text{member}(w, \text{last}(L))) .$$

- $\text{preWin}_i(G, \sigma, S)$ denotes that if $\text{GraphGame}(G)$ and $\text{Strategy}(G, \sigma)$ then S is a sequence of length $\text{size}(G)$ and $(S)_j$ is G^{σ^i} with all nodes $< j$ eliminated, and $S = \emptyset$ otherwise; and
- $\text{win}_i(\sigma, G, v)$ denotes

$$(\exists S)(\text{preWin}_i(G, \sigma, S) \wedge (\forall x < |G|)(\text{member}(x, (G)_{1-i}) \wedge (\text{Reach}((S)_0, v, x) \vee v = x) \rightarrow \neg \text{Reach}((S)_x, x, x))) .$$

As before, S_2^1 can prove that $\text{preWin}_i(G, \sigma, S)$ defines the graph of a function computing S , and thus win_i is Δ_1^b in S_2^1 .

Using these formalisations we can try to carry out in Bounded Arithmetic the proof of Memoryless Determinacy as given by Björklund et al. [9]. But we immediately run into troubles, as this proof builds on general (memory-dependent) strategies which are exponential-sized objects in the size of the input graph. Therefore, this proof cannot be formalised in Bounded Arithmetic, and we have to give a new proof which avoids general strategies. We start by stating without proof two basic lemmas.

Lemma 5.1. *Playing according to a winning strategy never leaves the winning set. In other words, if σ is a winning strategy for P_i , then there are no edges leaving $W_i(G, \sigma)$ in G^{σ^i} .*

Lemma 5.2. *Let σ and σ' be strategies in G . For $v \in V$ and $i = 0, 1$ define*

$$(\sigma \triangleright_i \sigma')(v) := \begin{cases} \sigma(v) & \text{if } v \in W_i(G, \sigma) \cap V_i \\ \sigma'(v) & \text{otherwise} \end{cases}$$

Then

$$W_i(G, \sigma) \cup W_i(G, \sigma') \subseteq W_i(G, \sigma \triangleright_i \sigma')$$

That is, the strategy $\sigma \triangleright_i \sigma'$ derived from σ and σ' is at least as successful for P_i as either of σ and σ' .

With this we are ready to present our new proof of memoryless determinacy.

Theorem 5.3 (Memoryless Determinacy). $(\forall G)(\exists \sigma) \text{det}(G, \sigma)$

Proof. We prove the theorem by induction on $k \leq n^2$ over the formula

$$(\forall G \in \mathcal{G}_n) \left(\text{lh}(E) \leq k \rightarrow (\exists \sigma) \text{det}(G, \sigma) \right) .$$

Here, $\text{lh}(E)$ is a canonical way to express the number of edges in G . Let $G \in \mathcal{G}_n$ with $\text{lh}(E) = k$. By the induction hypothesis we know

$$(\forall G' \in \mathcal{G}_n) \left(\text{lh}(E') < k \rightarrow (\exists \sigma') \text{det}(G', \sigma') \right) .$$

Let $W_i := W_i(G)$. Using Lemma 5.2 (repeatedly) we can find one strategy σ such that $W_0 = W_0(G, \sigma)$ and $W_1 = W_1(G, \sigma)$, in the following way: By definition, for each $v \in W_i$ there is some strategy σ_v such that $\text{win}_i(\sigma_v, G, v)$. By repeatedly applying Lemma 5.2 we can combine these to obtain one strategy σ_0 such that $W_0 \subseteq W_0(G, \sigma_0)$. By definition of $W_0(G)$ we obviously have $W_0(G, \sigma_0) \subseteq W_0(G)$. Hence $W_0 = W_0(G, \sigma_0)$. Similar, we obtain some σ_1 such that $W_1 = W_1(G, \sigma_1)$. We now define σ as σ_0 on V_0 and σ_1 otherwise:

$$\sigma(v) := \begin{cases} \sigma_0(v) & \text{if } v \in V_0 \\ \sigma_1(v) & \text{otherwise} \end{cases}$$

Then we obviously have that $W_0 = W_0(G, \sigma)$ and $W_1 = W_1(G, \sigma)$, as by definition $W_i(G, \sigma)$ only depends on $\sigma|_{V_i}$.

Letting

$$U := V \setminus (W_0 \cup W_1)$$

what we have to show is that, in fact, $U = \emptyset$.

Assume for the sake of contradiction that $U \neq \emptyset$. We observe that by Lemma 5.1, there are no edges from $U \cap V_i$ to W_i . We also observe that there are no self-loops in U , that is, $u \rightarrow u \notin E$ for all $u \in U$.

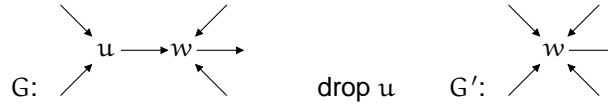


FIGURE 1: Case (I).

(I) Suppose there are $u, w \in U$, $u > w$ such that $u \rightarrow w$ is the only edge in G leaving u , as shown in Figure 1. In this case let us drop the edge $u \rightarrow w$, identify u with w and rename this identified node w . By the induction hypothesis there exists σ' with $\det(G', \sigma')$; we extend σ' to $\bar{\sigma}$ on G in the obvious way.

W.l.o.g., we assume that $w \in W_0(G', \sigma')$. As $w \notin W_0(G, \bar{\sigma})$, there must be a cycle reachable from w in $G^{\bar{\sigma}_0}$ won by P_1 . But essentially the same cycle is reachable from w in $G'^{\sigma'_0}$ and also won by P_1 – giving us our desired contradiction.

(II) Suppose now that for all $u, w \in U$ with $u > w$, if $u \rightarrow w$ is in G , then there is another edge $u \rightarrow w'$ in G with $w \neq w'$.

Let

$$U' := \{w \in U : (\exists u \in U)(u > w \wedge u \rightarrow w \in E)\}.$$

$u := \max U$ is not a winning position. Thus $u \rightarrow w \in E$ for some $w \in U$, hence $w \in U'$, and hence $U' \neq \emptyset$. Let $x := \min U'$. W.l.o.g., we assume that $x \in V_0$. Fix $y \in U$ with $y > x$ and $y \rightarrow x \in E$, and some $z \in V$ different from x with $y \rightarrow z \in E$. That is, we are in the situation displayed in Figure 2.



FIGURE 2: Case (II) general case.

(II.a) Assume $y \in V_1$. Consider, then, the graph G' where we remove the edge $y \rightarrow x$ from G , cf. Figure 3. By the induction hypothesis there exists some σ' such that $\det(G', \sigma')$. W.l.o.g. (by



FIGURE 3: Case (II.a)

Lemma 5.2) $\sigma'_i \upharpoonright_{W_i} = \sigma_i \upharpoonright_{W_i}$.

If P_1 wins in G' from x using σ' , then P_1 also wins in G from x using σ' , as $y \rightarrow x$ does not affect the choices of P_0 – contradiction.

Thus, P_0 wins in G' from x using σ' . As $x \in U$, P_0 does not win from x in G using σ' . Thus, there is a cycle reachable from x in $G^{\sigma'_0}$ won by P_1 . As this situation does not exist in G' , the cycle has to include $y \rightarrow x$. Also, it has to leave U , as otherwise the least node would be x and the cycle won by P_0 . It cannot reach W_0 , so it has to reach W_1 . But then W_1 is reachable from x in $G'^{\sigma'_0}$ – contradiction.

(II.b) Assume $y \in V_0$. Let G' be obtained from G by dropping the edge $y \rightarrow z$, cf. Figure 4. By the induction hypothesis there exists some σ' such that $\det(G', \sigma')$. W.l.o.g. (by Lemma 5.2)


FIGURE 4: Case (II.b)

$$\sigma'_i \upharpoonright W_i = \sigma_i \upharpoonright W_i.$$

If P_0 wins in G' from x using σ' , then P_0 also wins in G from x using σ' , as $y \rightarrow z$ does not affect the choices of P_1 – contradiction.

Thus, P_1 wins in G' from x using σ' . As $x \in U$, P_1 does not win from x in G using σ' . Thus, there is a cycle reachable from x in $G^{\sigma'_i}$ won by P_0 . Again, this cycle has to include $y \rightarrow z$. If the path from x to y would stay in U , adding $y \rightarrow x$ would give a win for P_0 in $G^{\sigma'_i}$. It cannot reach W_1 , so it has to reach W_0 . But then W_0 is reachable from x in $G^{\sigma'_i}$ – contradiction. \square

Theorem 5.4. $S_2^2 \vdash (\forall G)(\exists \sigma) \det(G, \sigma)$.

Proof. We argue informally in S_2^2 . Let G be a graph of size n . Let $\varphi(n, k)$ be the formula

$$(\forall G' \in \mathcal{G}_n) (\text{lh}(E') \leq k \rightarrow (\exists \sigma) \det(G', \sigma))$$

from the previous proof. Inspection of φ shows that it describes a coNP^{NP} property which can be equivalently expressed as a Π_2^b formula (provably in S_2^1). As $n^2 \leq |G\#G|$ (where “#” is one of the basic function symbols of Bounded Arithmetic), we obtain that

$$\varphi(n, 0) \wedge (\forall k < n^2) (\varphi(n, k) \rightarrow \varphi(n, k + 1)) \rightarrow \varphi(n, n^2)$$

is provable in S_2^2 by Theorem 4.1.

The main part in the proof of Theorem 5.3 is to show that $\varphi(n, k)$ implies $\varphi(n, k + 1)$. We observe that this only involves reasoning with polynomial-sized objects which can be formalised in weak theories of bounded arithmetic like S_2^1 —essentially we have to be able to construct from polynomial-sized objects (game graphs) other polynomial-sized objects which differ only in some very easy to describe properties. Thus we have, using that S_2^1 is a sub-theory of S_2^2 , that

$$(\forall k < n^2) (\varphi(n, k) \rightarrow \varphi(n, k + 1))$$

holds. It is easy to see that the base case $\varphi(n, 0)$ also holds, as there are no game graphs which satisfy $V' \neq \emptyset$ and $E' = \emptyset$. Putting things together we obtain $\varphi(n, n^2)$. Now, by assumption $G \in \mathcal{G}_n$ and $\text{lh}(E) \leq n^2$. Hence, $(\exists \sigma) \det(G, \sigma)$. \square

Applying Theorem 4.3 to the previous theorem shows the following corollary:

Corollary 5.5. **MEMDET** is in PLS. \square

6. CONCLUSION

We have studied the problem of determining who has the winning strategy in a parity game in terms of the corresponding problem of determining a memoryless winning strategy; we denoted this NP search problem by **MEMDET**. We have shown that **MEMDET** is in PLS, by giving a new proof of Memoryless Determinacy which can be formalised in Bounded Arithmetic S_2^2 – the theory which allows polynomially reasoning with NP^{NP} -induction of polynomial length.

Examining the proof of Theorem 4.3, one could describe a PLS algorithm solving **MEMDET** without mentioning Bounded Arithmetic. However, it is unclear to us whether this would be

beneficial, as the obtained algorithm would not directly deal with game graphs, but with proof-theoretic notions like witnessing formulas for sequents of Bounded Arithmetic formulas. Furthermore, this would hide any further observations that might be made about the complexity of parity games based on results from Bounded Arithmetic.

To the best of our knowledge, this is the first result which classifies the search problem **MEMDET** by relating it to PLS. Vöge and Jurdziński [16] present a discrete strategy improvement algorithm which is closest in spirit to a PLS algorithm for the problem. However, their algorithm is based on a non-deterministic operator `Improve`, which poses the most obvious of many challenges faced in considering whether or not this algorithm can be turned into a PLS-description.

The previous paragraph notwithstanding, one of the anonymous referees brought to our attention an unpublished report from 2004 [17] which, in a way not advertised by its title, announces the result that parity games lies inside PLS. Indeed, the further claim that parity games are *not* PLS-complete is made. We have not had the opportunity to verify the validity of the argument presented in this 78-page technical report (though we do not lack confidence in its authors). However, the techniques used there are vastly different to ours; even in light of this earlier yet unpublished announcement, there is still the merit in the present manuscript in introducing Bounded Arithmetic as a tool against the problem of establishing the complexity of parity games.

REFERENCES

- [1] Grädel, E., Thomas, W., and Wilke, T. (2002) *Automata, Logics and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer.
- [2] Stirling, C. (2001) *Modal and Temporal Properties of Processes*. Springer.
- [3] Martin, D.A. (1975) Borel determinacy. *Annals of Mathematics*, **102**, 363–371.
- [4] Emerson, E.A. and Jutla, C.S. (1991) Tree automata, mu-calculus and determinacy (extended abstract). *Proceedings of FoCS'91: Foundations of Computer Science*, pp. 368–377. IEEE Computer Society Press.
- [5] Emerson, E.A. (1985) Automata, tableaux, and temporal logics. *Proceedings of a Workshop on Logic of Programs*, volume 193 of *Lecture Notes in Computer Science*, pp. 79–87. Springer.
- [6] McNaughton, R. (1993) Infinite games played on finite graphs. *Ann. Pure and Appl. Logic*, **65**(2), 149–184.
- [7] Zielonka, W. (1998) Infinite games played on finitely coloured graphs with applications to automata on infinite trees. *J. Theo. Comp. Sci.*, **200**, 135–183.
- [8] Ehrenfeucht, A. and Mycielski, J. (1979) Positional strategies for mean payoff games. *J. Game Theory*, **8**, 109–113.
- [9] Björklund, H., Sandberg, S. and Vorobyov, S. (2004) Memoryless determinacy of parity and mean payoff games: a simple proof. *J. Theo. Comp. Sci.*, **310**, 365–378.
- [10] Jurdziński, M. (1998) Deciding the winner in parity games is in UP^{nco} -UP. *Information Processing Letters*, **68**(3), 119–124.
- [11] Jurdziński, M., Paterson, M. and Zwick, U. (2006) A deterministic subexponential algorithm for solving parity games. *Proceedings of SODA 2006: Symposium on Discrete Algorithms*, pp. 117–123. ACM/SIAM.
- [12] Buss, S.R. (1986) *Bounded arithmetic*, Bibliopolis, Naples.
- [13] Beame, P. et al (1998) The relative complexity of NP search problems. *J. Comput. System Sci.*, **57**(1), 3–19.
- [14] Johnson, D.S., Papadimitriou, C.H. and Yannakakis, M (1988) How easy is local search? *J. Comput. Syst. Sci.*, **37**(1), 79–100.
- [15] Buss, S.R. and Krajíček, J. (1994) An application of Boolean complexity to separation problems in bounded arithmetic. *Proc. London Math. Soc.* (3), **69**(1), 1–21.
- [16] Vöge, J. and Jurdziński, M. (2000) A discrete strategy improvement algorithm for solving parity games (Extended abstract). *Proceedings of CAV 2000: Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pp. 202–215, Springer-Verlag.
- [17] Björklund, H., Sandberg, S. and Vorobyov, S. (2004) Randomized subexponential algorithms for infinite games. Technical Report 2004-011, Uppsala University.