

# On the complexity of parity games

Arnold Beckmann and Faron Moller  
Department of Computer Science  
Swansea University  
UK

April 7, 2008

## Abstract

Parity games underlie the model checking problem for the modal  $\mu$ -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 which can be formulated 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  $\mu$ -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 of the problem to some other problem which is hard with respect to PLS, thus making the discovery of a polynomial-time solution unlikely according to current wisdom.

# 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. 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 maximal 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 [Sti01], 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 maximal 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 node with the maximum label which is encountered infinitely often. Finally, for consistency with [Sti01], we shall look to the minimum, rather than maximum, label that appears infinitely often. With little thought it can be realised that any parity games 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 [Mar75]. 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  $\mu$ -calculus [EJ91]. 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 easy to motivate intuitively: 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 intuitive “proof” is perfectly adequate for an undergraduate Computer Science lecture, particularly in light of the complexity of the formal proofs that have been devised. Emerson [Eme85] outlined the first such proof, with further proofs given by Emerson and Jutla [EJ91], McNaughton [McN93] and Zielonka [Zie98]. More recently, Björklund et al. [BSV04] have devised an elegant and elementary proof of memoryless determinacy of parity 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 minimum-labelled 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  $\mathbf{NP} \cap \mathbf{coNP}$  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 Jurziński [Jur98], is that the problem lies in  $\mathbf{UP} \cap \mathbf{coUP}$ .

One further observation can be usefully made at this point: the problem of determining who has the winning strategy in a parity game is equivalent with respect to polynomial-time reducibility to the 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 who it belongs to. For the other direction, we assume that we have a polynomial-time 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.

*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 [Bus86] 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 [BCE<sup>+</sup>98]. ■

An important class of search problems are total NP search problem in the sense of Beame et al. [BCE<sup>+</sup>98]. They are given by a binary, polynomial time computable relation  $R$  which is polynomially bounded, i.e.  $(\forall x)(\forall y) R(x, y) \Rightarrow |y| \leq |x|^{O(1)}$ , and total, i.e.  $(\forall x)(\exists y)R(x, y)$ . 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. Johnson et al. [JPY88] introduced, amongst others, one such class called Polynomial Local Search (PLS). PLS consists essentially of optimisation problems for which polynomial-time local-search heuristics exists.

We have already pointed out that the problem of determining who has the winning strategy in a parity game is equivalent with respect to polynomial-time reducibility to the problem of determining a memoryless winning strategy. 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, induction is the combinatorial principle which can be used directly to prove totality of search problems. 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 [Bus86] has shown that the **NP** search problems whose totality can be shown using induction of polynomial length on **NP** properties (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 is a polynomial-time problem. Buss and Krajíček [BK94] have (implicitly) shown that the **NP** search problems whose totality can be shown using induction of polynomial length on  $\mathbf{NP}^{\mathbf{NP}}$  properties (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  $S_2^2$ , to show that the problem of computing a positional winning strategy in a simple graph game is in **PLS**.

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

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

**Lemma 3.3.**  $P_i$  wins from  $v$  in  $G$  using strategy  $\sigma$  iff the least node in any cycle reachable from  $v$  in  $G^{\sigma_i}$  is owned by  $P_i$ .

*Proof.* If a cycle is reachable from  $v$  in  $G^{\sigma_i}$  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  $\sigma$ , 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  $\sigma$  is a strategy in  $G$ ,  $v \in V$ , and every cycle reachable from  $v$  in  $G^{\sigma_i}$  is won by  $P_i$  (ie, the least node on the cycle is owned by  $P_i$ ).

**Definition 3.5** (Winning Positions). Let

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

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

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

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

**Definition 3.6** (Winning Strategy). Let  $det(G, \sigma)$  denote that  $\sigma$  determines the game  $G$ , i.e., that  $\sigma$  is a strategy for  $G$  and that

$$(\forall v \in V)(win_0(\sigma, G, v) \vee win_1(\sigma, G, v))$$

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

**Theorem 3.7** (Memoryless Determinacy [Eme85]).  $(\forall G)(\exists \sigma) det(G, \sigma)$

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

*Proof.* By Memoryless Determinacy we know  $W_0(G) \cup W_1(G) = V$ , and that  $W_0(G) \cap W_1(G) = \emptyset$ . Hence:

$$\begin{aligned} v \in W_0(G) &\leftrightarrow (\exists \sigma) win_0(\sigma, G, v) \\ v \notin W_0(G) &\leftrightarrow (\exists \sigma) win_1(\sigma, G, v) \end{aligned}$$

The problem of determining  $win_i(\sigma, G, v)$ , 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 [Bus86].

Bounded Arithmetic as defined in [Bus86] 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:

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

They denote in their standard interpretation over the natural numbers  $\mathbb{N}$  the constant zero, 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  $2^{|x| \cdot |y|}$  produces polynomial growth rate, and finally 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 of no problem as our base theory can define *all* polynomial time computable function—Buss [Bus86] has shown that such

an extension of the language is conservative. The additional function symbols we will use are

$$(x)_y, \langle \rangle, x * y, x ** y, lh$$

Complete  
the list

We define  $last(x)$  as the term  $(x)_{lh(x)-1}$  and  $member(x, y)$  as the formula  $(\exists z < |y|)x = (y)_z$ .

wrong

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

place,  
shift

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:

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

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 [Bus86] has introduced classes of bounded formulae  $\Sigma_i^b$  and  $\Pi_i^b$  which correspond to the complexity classes  $\Sigma_i^p$  and  $\Pi_i^p$  in the polynomial time hierarchy. For an exact definition we refer the reader to Buss [Bus86, p.20]—we will only use the following observation:

- Formulae of the form  $(\exists x_1 \leq s_1)\varphi(x_1)$  for sharply bounded  $\varphi$  are in  $\Sigma_1^b$ .
- Formulae of the form  $(\forall x_1 \leq s_1)(\exists x_2 \leq s_2)\varphi(x_1, x_2)$  for sharply bounded  $\varphi$  are in  $\Pi_2^b$ .

Bounded Arithmetic theories are defined by stating axioms defining the non-logical symbols plus some axiom scheme—the latter is responsible for the strength of the theory. Buss [Bus86] 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  $\Phi$ , we use  $\Phi$ -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 set just picked. From all possible choices, we will use the following:

$$\begin{aligned} S_2^1 &= \text{BASIC} + \Sigma_1^b\text{-LIND} \\ S_2^2 &= \text{BASIC} + \Sigma_2^b\text{-LIND} \end{aligned}$$

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

Buss [Bus86] has also shown equivalences between different induction schemes. In particular, we have the following:

**Theorem 4.1** ([Bus86]).  $S_2^2$  and BASIC+ $\Pi_2^b$ -LIND prove the same formulae.

## Definable functions and search problems

Definable function and search problems form an important notion for Bounded Arithmetic as they provide the link between theories and complexity classes. A function is  $\Sigma_1^b$  definable in theory  $T$  if there is a  $\Sigma_1^b$ -formula  $\varphi$  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 is  $\Delta_1^b$  definable in  $T$  if there is a  $\Sigma_1^b$ -formula and a  $\Pi_1^b$ -formula which both define the predicate, and whose equivalence can be shown in  $T$ .

**Theorem 4.2** (Buss '86 [Bus86]). *The  $\Sigma_1^b$ -definable functions in  $S_2^1$  coincide with FP, the class of functions computable in polynomial time. The  $\Delta_1^b$ -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.  $R$  is  $\Delta_1^b$ -definable in a theory  $T$  iff there exists  $\varphi \in \Delta_1^b$  with respect to  $T$  such that  $T$  proves the totality of  $R$  via  $\varphi$ , i.e.  $T \vdash (\forall x)(\exists y)\varphi(x, y)$ .

**Theorem 4.3** (Buss, Krajíček'94 [BK94]). *The  $\Sigma_1^b$ -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  $\Delta_1^b$ -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  $\mathcal{L}_{BA}$ :  $GraphGame(G)$  is the sharply bounded formula expressing that  $lh(G) = 4$  and that  $((G)_0, (G)_1, (G)_2)$  is a graph game of size  $|(G)_3|$  according to Definition 2.1. 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  $G$  to be  $|(G)_3|$ .

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

$PreStrategy(G, \sigma)$  denotes that  $\sigma$  is a pre-strategy for  $G$ ,

$Strategy(G, \sigma)$  that  $\sigma$  is a strategy for  $G$ ,

$StrategyRestriction(G, \sigma, i, \mu)$  that  $\mu = \sigma \upharpoonright_{V_i}$ , and

$GraphRestriction(G, \sigma, G')$  that if  $PreStrategy(G, \sigma)$  then  $G' = G^\sigma$ , otherwise  $G' = \emptyset$ .

It is obvious that  $S_2^1$  can prove that  $StrategyRestriction(G, \sigma, i, \mu)$  and  $GraphRestriction(G, \sigma, G')$  define graphs of functions, i.e. that  $\mu$  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  $win_i(G, \sigma, v)$  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:

$preReach(G, v, L)$  denotes that if  $GraphGame(G)$  then  $L$  is a sequence of length  $size(G) + 1$  with  $(G)_0 = \langle v \rangle$  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.

$Reach(G, v, w)$  denotes  $(\exists L)(preReach(G, v, L) \wedge member(w, last(L)))$

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  $preReach(G, v, L)$  defines the graph of a function computing  $L$  on input  $G$  and  $v$ . Thus,  $Reach$  is  $\Delta_1^b$  in  $S_2^1$  because

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

$preWin(G, \sigma, S)$  denotes that if  $GraphGame(G)$  and  $Strategy(G, \sigma)$  then  $S$  is a sequence of length  $size(G)$  and  $(S)_i$  is  $G^\sigma$  with all nodes  $< i$  eliminated, and  $S = \emptyset$  otherwise.

$Win_i(G, \sigma, v)$  denotes

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

As before,  $S_2^1$  can prove that  $preWin(G, \sigma, S)$  defines the graph of a function computing  $S$ , and thus  $Win_i$  is  $\Delta_1^b$  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. [BSV04]. But we immediately run into troubles, as this proof builds on general strategies (with “memory”) 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 omits 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  $\sigma$  is a winning strategy for  $P_i$ , then there are no edges leaving  $W_i(G, \sigma)$  in  $G^{\sigma_i}$ .*

**Lemma 5.2.** *Let  $\sigma$  and  $\sigma'$  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  $\sigma$  and  $\sigma'$  is at least as successful for  $P_i$  as either of  $\sigma$  and  $\sigma'$ .*

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

**Theorem 5.3** (Memoryless Determinacy).  $(\forall G)(\exists \sigma) \det(G, \sigma)$

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

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

Let  $G \in \mathcal{G}_n$  with  $|E| = k$ . By the induction hypothesis we know

$$(\forall G' \in \mathcal{G}_n) (|E'| < k \rightarrow (\exists \sigma') \det(G', \sigma'))$$

Let  $W_i := W_i(G)$ . Using Lemma 5.2 (repeatedly) we can find one strategy  $\sigma$  such that  $W_0 = W_0(G, \sigma_0)$  and  $W_1 = W_1(G, \sigma_1)$ . 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  $W_i$ .

**(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  $\sigma'$  with  $\det(G', \sigma')$ ; we extend  $\sigma'$  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

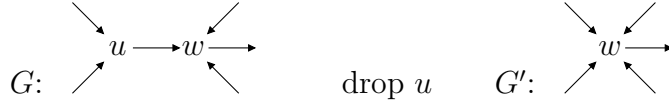


Figure 1: Case **(I)**.

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

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

$u := \max U$  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  $\bar{x} \in U$  with  $\bar{x} \geq x$  and  $\bar{x} \rightarrow x \in E$ , and some  $w \in V$  with  $\bar{x} \rightarrow w \in E$ . That is, we are in the situation displayed in Figure 2.



Figure 2: Case **(II)** general case.

**(II.a)** Assume  $\bar{x} \in V_1$ . Consider, then, the graph  $G'$  where we remove the edge  $\bar{x} \rightarrow x$  from  $G$ , cf. Figure 3. By the induction hypothesis there exists



Figure 3: Case **(II.a)**

some  $\sigma'$  such that  $\det(G', \sigma')$ . W.l.o.g. (by Lemma 5.2)  $\sigma'_i \upharpoonright_{w_i} = \sigma_i \upharpoonright_{w_i}$ .

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

Thus,  $P_0$  wins in  $G'$  from  $x$  using  $\sigma'$ . As  $x \in U$ ,  $P_0$  does not win from  $x$  in  $G$  using  $\sigma'$ . 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  $\bar{x} \rightarrow x$ . 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  $\bar{x} \in V_0$ . Let  $G'$  be obtained from  $G$  by dropping the edge  $\bar{x} \rightarrow w$ , cf. Figure 4. By the induction hypothesis there exists some  $\sigma'$  such



Figure 4: Case **(II.b)**

that  $\det(G', \sigma')$ . W.l.o.g. (by Lemma 5.2)  $\sigma'_i \upharpoonright_{w_i} = \sigma_i \upharpoonright_{w_i}$ .

If  $P_0$  wins in  $G'$  from  $x$  using  $\sigma'$ , then  $P_0$  also wins in  $G$  from  $x$  using  $\sigma'$ , as  $\bar{x} \rightarrow w$  does not affect the choices of  $P_1$  – contradiction.

Thus,  $P_1$  wins in  $G'$  from  $x$  using  $\sigma'$ . As  $x \in U$ ,  $P_1$  does not win from  $x$  in  $G$  using  $\sigma'$ . Thus, there is a cycle reachable from  $x$  in  $G^{\sigma'_1}$  won by  $P_0$ . Again, this cycle has to include  $\bar{x} \rightarrow w$ . If the path from  $x$  to  $\bar{x}$  would stay in  $U$ , adding  $\bar{x} \rightarrow x$  would give a win for  $P_0$  in  $G^{\sigma'_1}$ . It cannot reach  $W_1$ , so it has to reach  $W_0$ . But then  $W_0$  is reachable from  $x$  in  $G^{\sigma'_1}$  – contradiction.  $\square$

**Theorem 5.4.**

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

*Proof.* The induction used in the previous proof is of polynomial length and on a  $\text{coNP}^{\text{NP}} = \Pi_2^b$  property, which is available in  $S_2^2$ . Everything else in the proof involves reasoning with polynomial size objects and can be formalised in Bounded Arithmetic.  $\square$

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

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

## 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.

## References

- [BCE<sup>+</sup>98] Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. *J. Comput. System Sci.*, 57(1):3–19, 1998. 27th Annual ACM Symposium on the Theory of Computing (STOC’95) (Las Vegas, NV).
- [BK94] Samuel R. Buss and Jan Krajíček. An application of Boolean complexity to separation problems in bounded arithmetic. *Proc. London Math. Soc.* (3), 69(1):1–21, 1994.
- [BSV04] Henrik Björklund, Sven Sandberg, and Sergei Vorobyov. Memoryless determinacy of parity and mean payoff games: a simple proof. *J. Theo. Comp. Sci.*, 310:365–378, 2004.
- [Bus86] Samuel R. Buss. *Bounded arithmetic*, volume 3 of *Studies in Proof Theory. Lecture Notes*. Bibliopolis, Naples, 1986.
- [EJ91] E.A Emerson and C.S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *FoCS’91: The 32nd Annual Symposium on Foundations of Computer Science*, pages 368–377. IEEE Computer Society Press, 1991.
- [Eme85] E.A. Emerson. Automata, tableaux, and temporal logics. In *Proceedings of a Workshop on Logic of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 79–87. Springer, 1985.

- [JPY88] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *J. Comput. Syst. Sci.*, 37(1):79–100, 1988.
- [Jur98] Marcin Jurdziński. Deciding the winner in parity games is in  $UP \cap co-UP$ . *Information Processing Letters*, 68(3):119–124, 1998.
- [Mar75] Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102:363–371, 1975.
- [McN93] R. McNaughton. Infinite games played on finite graphs. *Ann. Pure and Appl. Logic*, 65(2):149–184, 1993.
- [Sti01] Colin Stirling. *Modal and Temporal Properties of Processes*. Texts in Computer Science. Springer, 2001.
- [Zie98] W. Zielonka. Infinite games played on finitely coloured graphs with applications to automata on infinite trees. *J. Theo. Comp. Sci.*, 200:135–183, 1998.