Solving Parity Games by a Reduction to SAT

Martin Lange

Institut für Informatik, University of Munich, Germany

Abstract. This paper presents a reduction from the problem of solving parity games to the satisfiability problem for formulas of propositional logic in conjunctive normal form. It uses Jurdziński's characterisation of winning strategies via progress measures. The reduction is motivated by the apparent success that using SAT solvers has had in symbolic verification. The paper reports on a prototype implementation of the reduction and presents some runtime results.

1 Introduction

Solving a parity game is an intrinsic and interesting problem in theoretical computer science. It is equivalent to the model checking problem for the modal μ-calculus [7], and is closely related to the problem of solving other games like mean pay-off or stochastic games.

It is also one of the few inhabitants of the complexity class NP∩co-NP [8] – even UP∩co-UP [10] – and many people believe that it is in fact in PTIME. Although no-one has been able to show inclusion in PTIME so far, people have invented many algorithms for solving parity games.

Recursive methods like Zielonka's algorithm [21], etc. solve a game with at most p different priorities by referring several times to games with strictly less than p many different priorities. Consequently, their running time is exponential in the number of priorities in the game.

Strategy improvement as done by Jurdziński and Vöge's algorithm [12] based on Puri's [17] – and similar to Hoffman and Karp's [9] as well as Ludwig's [13] algorithms for stochastic games – uses the fact that strategies can be partially ordered with a winning strategy being maximal w.r.t. this order. This performs very well in practice but it is not known whether a polynomial number of iteration steps always suffices to find a winning strategy.

A randomised and also subexponential algorithm is due to Björklund, Petersson, Sandberg, and Vorobyov [16].

Every model checker for the full μ-calculus is in principal also an algorithm for solving parity games. Several of the former have emerged beginning with tableau-like methods [4, 20], automata-theoretic ones [7], equation solvers [1] and symbolic model checking procedures [2].

The algorithm with the currently best asymptotic complexity is Jurdziński's *small progress measures* procedure [11]. It is exponential in the number of odd priorities occurring in the game, i.e. in the half of the maximal priority. A similar asymptotic bound is achieved by Seidl's fixpoint iteration [19].

Opposing common undergraduate syllabi, polynomial time is not a synonym for efficiency. The famous SAT problem is NP-complete [5] and, hence, theoretically does not admit efficient algorithms. However, there are many SAT solvers that are astonishingly efficient in practice, for example CHAFF [15]. Such solvers are used successfully for example in bounded model checking [3].

Inspired by this we present a different approach to solving parity games: a reduction to SAT. Theoretically this is not too exciting since it is clear that such a reduction must exists. Furthermore, SAT is believed to be harder than PARITY. Again, clever heuristics implemented in nowadays SAT solvers can make up for this and result in an algorithm that is efficient in practice. Furthermore, there are fragments of SAT that can be solved in polynomial time. Hence, our reduction opens up a new possibility for showing inclusion of PARITY in PTIME.

The reduction is based on a comment by Emerson where he explains inclusion of the model checking problem for the modal μ-calculus in NP. He essentially writes "Guess a rank for each μ-subformula at each state in a transition system. Show that the lexicographic order on the tuples through the transition system is well-founded ." [6].

Note that an NP-algorithm is not the same as a reduction to SAT. What we want is a formula of propositional logic that is satisfiable iff the existential player has a winning strategy starting in a certain node of the parity game. We cannot let the SAT solver "guess" a strategy and verify that it is winning by a graph-theoretical method. The resulting propositional formula has to express already that the strategy is winning.

Following the idea about ranks consequently with the aim of a local characterisation of winning strategies we define the notion of a μ-annotation – effectively and unintentionally re-inventing Jurdziński progress measures [11]. We stick to the term μ-annotation because in this setting they are not dynamically updated and, hence, do not measure any progress. We hereby attribute the theory of μ-annotations to Jurdziński explicitly, but keep the corresponding results and proofs in this paper in order to put the reduction to SAT on a solid theoretical foundation.

The rest of the paper is organised as follows. Sections 2 and 3 recall definitions. Section 4 contains the aforementioned theory regarding μ-annotations, resp. progress measures. Sections 5 and 6 present two slightly different translations from PARITY to SAT based on the existence of such annotations. Section 7 presents experimental results of one of these translations, and Section 8 discusses further work.

2 Parity Games

A parity game is a tuple $G = (V, E, v_0, p)$ where (V, E) is a finite, directed graph and V is partitioned into two sets $V_\exists$ and $V_\forall$, $v_0 \in V$ is the starting node, and $p : V \to \mathbb{N}$ is a priority function. G is assumed to be total, i.e. for every $v \in V$ there is a $v' \in V$ with $(v, v') \in E$.

A play of G is a maximal path $\pi = v_0 v_1 v_2 \ldots$ through G starting in v_0. It is constructed in the following way. Given a node $v \in V_x$, player x chooses a $v' \in V$ with $(v, v') \in E$ and the construction of the play continues with v'.

Given a play $\pi = v_0 v_1 \ldots$ let $\inf \pi := \{v \in V \mid$ there are infinitely many $i \in \mathbb{N}$ s.t. $v = v_i\}$. Player $\exists$ wins the play $\pi = v_0 v_1 \ldots$ if $\min\{p(v) \mid v \in \inf \pi\}$ is even. If it is odd then player $\forall$ wins the play π.

A strategy for player x is a function $\sigma : V^* V_x \to V$ that tells player x which choice to make depending on the current construction of a play. A strategy is called *positional* if for all $\alpha, \beta \in V^*$ and all $v \in V_x$ we have $\sigma(\alpha v) = \sigma(\beta v)$. Hence, the choices made according to a positional strategy only depend on the last node visited. In such a case we will rather use $\sigma : V \to V$.

A play $\pi = v_0 v_1 \ldots$ is called *conforming* to a (positional) strategy σ for player x if for all $i \in \mathbb{N}$ if $v_i \in V_x$ then $v_{i+1} = \sigma(v_0 \ldots v_i)$. A strategy σ for player x is called a *winning strategy* if every play conforming to σ is won by player x.

Given a parity game G and a strategy σ for player $\exists$ we write $G|_\sigma$ for the parity game that is induced by σ on G. Formally, $G|_\sigma = (V, E \cap (V_\forall \times V \cup \{(v, \sigma(v)) \mid v \in V_\exists\}), v_0, p)$.

The problem of solving a parity game $G = (V, E, v_0, p)$ is to determine whether or not player $\exists$ has a winning strategy for G. Let PARITY $:= \{G \mid$ player $\exists$ has a winning strategy for $G\}$.

Theorem 1. [7] *Given a parity game G.*
(a) Player $\exists$ has a winning strategy for G iff player $\forall$ does not have a winning strategy for G.
(b) A player has a winning strategy for G iff she has a positional winning strategy for G.

Theorem 2. [8] *The problem of solving a parity game is in NP∩co-NP.*

Given a parity game G, the finite unraveling $\mathcal{R}(G)$ is informally obtained by unfolding the graph G to a tree with back-edges only to nodes of minimal priority on the thus created loop. Formally, it is defined as $\mathcal{R}(G) = (V', E', v_0, p')$ with

$$\begin{aligned}
V' &:= \{v_0 \ldots v_n \in V^+ \mid \forall i = 1, \ldots, n : (v_{i-1}, v_i) \in E \text{ and} \\
&\qquad \forall v \in V \text{ there are at most two } i, j \text{ s.t. } v_i = v_j = v\} \\
V'_\exists &:= V' \cap V^* V_\exists \\
V'_\forall &:= V' \cap V^* V_\forall \\
E' &:= \{(v_0 \ldots v_n, v_0 \ldots v_n v_{n+1}) \in V' \times V' \mid (v_n, v_{n+1}) \in E\} \cup \\
&\qquad \{(v_0 \ldots v_k \ldots v_n, v_0 \ldots v_k) \in V' \times V' \mid (v_n, v_k) \in E \text{ and} \\
&\qquad\qquad \forall i = k, \ldots, n : p(v_k) \leq p(v_i)\} \\
p'(v_0 \ldots v_n) &:= p(v_n)
\end{aligned}$$

For two nodes $v, v' \in V'$ we write $v \sim v'$ iff there is a $w \in V$ s.t. $v = v_0 v_1 \ldots w$ and $v' = v_0 v'_1 \ldots w$ for some $v_1, v'_1, \ldots$. Note that $\sim$ is an equivalence relation preserving priorities.

Lemma 1. *Player $\exists$ wins the parity game G iff she wins the parity game $\mathcal{R}(G)$.*

Proof. Suppose she has a winning strategy σ for G. It immediately carries over to a strategy σ' on $\mathcal{R}(G)$ via $\sigma'(v_0 \dots v_n) = v_0 \dots v_n \sigma(v_0 \dots v_n)$. It is not hard to see that σ' is a winning strategy. The converse direction follows from Theorem 1: if she does not have a winning strategy then player $\forall$ has a winning strategy which carries over to a strategy on $\mathcal{R}(G)$ in the same way. □

3 Propositional Logic and SAT

Given a set of propositional variables $V = \{X, Y, \dots\}$, formulas of propositional logic are built in the following way.

$$\Phi \ ::= \ X \mid \Phi \vee \Phi \mid \Phi \wedge \Phi \mid \neg\Phi$$

We use the usual abbreviations: $\varphi \to \psi := \neg\varphi \vee \psi$ and $\mathtt{tt} := X \vee \neg X$ for some $X \in V$.

A variable assignment is a mapping $\zeta : V \to \{0, 1\}$. A formula is called satisfiable iff there is an assignment to the variables in it that makes the formula true under the usual interpretations of the boolean connectives. Let SAT := $\{\Phi \mid \Phi$ is satisfiable $\}$.

A formula is said to be in conjunctive normal form, if it is of the form $\bigwedge_{i \in I} \bigvee_{j \in J_i} l_{i,j}$ where for all i, j: $l_{i,j}$ is a literal, i.e. a possibly negated variable. Let SAT-CNF := SAT $\cap$ $\{\Phi \mid \Phi$ is in conjunctive normal form $\}$.

Theorem 3. [5] *SAT and SAT-CNF are both NP-complete under polynomial time reductions.*

4 Characterising Winning Strategies Locally

Given a parity game $G = (V, E, v_0, p)$, let $p_0 \geq \max\{\ p(v) \mid v \in V\ \}$ be an odd upper bound on the priorities occurring in G. A μ-annotation for G is a tuple $\overline{a} = (a_1, a_3, \dots, a_{p_0}) \in \mathbb{N}^{\frac{p_0+1}{2}}$.

Given two μ-annotations $\overline{a} = (a_1, \dots, a_{p_0})$ and $\overline{b} = (b_1, \dots, b_{p_0})$ for G and a $p \leq p_0$, we define

$$\overline{a} \trianglelefteq_p \overline{b} \quad \text{iff} \quad \begin{cases} a_i \leq b_i & \text{for all } i = 1, \dots, p-1 \quad \text{if } p \text{ is even} \\ a_p < b_p \text{ and } a_i \leq b_i & \text{for all } i = 1, \dots, p-2 \quad \text{o.w.} \end{cases}$$

We write $\overline{a}^{(p)}$ for some odd $p \in \mathbb{N}$ to denote the p-component of $\overline{a}$.

A μ-annotation for G is an η that assigns to each $v \in V$ an annotation in the above sense. It is called successful, iff for all $v \in V$:

- if $v \in V_\forall$ then for all $w \in vE$: $\eta(w) \trianglelefteq_{p(w)} \eta(v)$,
- if $v \in V_\exists$ then there is a $w \in vE$: $\eta(w) \trianglelefteq_{p(w)} \eta(v)$.

Lemma 2. *Let G be a parity game, η be a μ-annotation for G, and $\pi = v_0 v_1 \ldots$ be a play of G. If for all $i \in \mathbb{N}$: $\eta(v_{i+1}) \trianglelefteq_{p(v_{i+1})} \eta(v_i)$ then the minimal priority occurring infinitely often in π is even.*

Proof. Suppose that the minimal priority p that occurs infinitely often in π is odd. Then there are infinitely many nodes $v_{i_1}, v_{i_2}, \ldots$ on π, s.t. $\eta(v_{i_1})^{(p)} > \eta(v_{i_2})^{(p)} > \ldots$ since eventually there is no lower even priority anymore that would allow $\eta(v_{i_j})^{(p)} < \eta(v_{i_{j+1}})^{(p)}$ for some $j \in \mathbb{N}$. But then we cannot have $\eta(v_{i+1}) \trianglelefteq_{p(v_{i+1})} \eta(v_i)$ for all $i \in \mathbb{N}$, because the natural numbers are Noetherian. □

Lemma 3. *Let G be a parity game and σ be a strategy for a player $\exists$. An η is a successful μ-annotation for G iff it is a successful μ-annotation for $G|_\sigma$.*

Proof. The "only if" part is trivial since $G|_\sigma$ is a substructure of G. For the "if" part note that G and $G|_\sigma$ have the same set of nodes. Hence, a successful μ-annotation η for $G|_\sigma$ is also a μ-annotation for G. It is easy to see that it is also successful. Let $G = (V_\exists, V_\forall, E, v_0, p)$ and $G|_\sigma = (V_\exists, V_\forall, E', v_0, p)$. Take any node $v \in V$. If $v \in V_\forall$ then $vE = vE'$, hence success immediately carries over to G on these nodes. If $v \in V_\exists$ then $vE' = \{w\}$ for some $w \in V$ with $\eta(w) \trianglelefteq_{p(w)} \eta(v)$. Now $vE \supseteq \{w\}$, i.e. there still is a w with $\eta(w) \trianglelefteq_{p(w)} \eta(v)$. □

Lemma 4. *Let G be a parity game. There is a successful μ-annotation for G iff there is a successful μ-annotation for $\mathcal{R}(G)$.*

Proof. Let V be the node set of G and V' be the node set of $\mathcal{R}(G)$. A successful μ-annotation η for G immediately carries over to a μ-annotation η' on $\mathcal{R}(G)$ via $\eta'(v_0 \ldots v_n) = \eta(v_n)$. It is not hard to see that η' is also successful.

Now suppose there is a successful μ-annotation η' for $\mathcal{R}(G)$. In a first step we construct a successful μ-annotation η_{min} with minimal values. This can be done by successively decreasing annotation values whilst preserving success in each step. For loops in $\mathcal{R}(G)$ this has to be done simultaneously for all nodes on this loop. Next, observe that an annotation at a node v only depends on the annotations at nodes reachable from v. Furthermore, for two nodes $v, v' \in V'$ the paths emerging from v and v' projected onto their last component are the same. Hence, we have $\eta_{min}(v) = \eta_{min}(v')$ if $v \sim v'$. Finally, this yields a well-defined μ-annotation for G via $\eta(w) = \eta_{min}(v')$ for some $v' = v_0 \ldots w$. Again, it is not hard to see that η is successful. □

Theorem 4. *Player $\exists$ wins the parity game G iff there is a successful μ-annotation for G.*

Proof. ($\Leftarrow$) Suppose there is a successful μ-annotation η for $G = (V, E, v_0, p)$. It immediately yields a positional strategy σ for player $\exists$ defined by $\sigma(v) = w$ only if $\eta(w) \trianglelefteq_{p(w)} \eta(v)$. By assumption, for every $v \in V_\exists$ there is at least one such w. If there is more than one satisfying the inequality then $\sigma(v)$ can simply be defined as any of them. It remains to be seen that σ is a winning strategy.

Let player $\forall$ play against σ with any other strategy σ'. The result is a play $\pi = v_0 v_1 \ldots$. Since η is a successful μ-annotation we have for all $i \in \mathbb{N}$: $\eta(v_{i+1}) \trianglelefteq_{p(v_{i+1})} \eta(v_i)$. Lemma 2 then shows that player $\exists$ wins the play. Since she does so for any play π which is conforming to σ, she wins the game G.

($\Rightarrow$) Suppose player $\exists$ has a winning strategy σ for G. According to Theorem 1 we can assume σ to be positional. We will use σ to construct a μ-annotation η for $\mathcal{R}(G|_\sigma)$. Let p_{max} be the maximal odd priority used in G. At the beginning set $\eta(v_0) = (2 \cdot |V|, \ldots, 2 \cdot |V|)$. Suppose a node v on level n of the quasi-tree $\mathcal{R}(G)$ has already been assigned an annotation $\eta(v) = (a_1, \ldots, a_{p_{max}})$. Then for every son w of v that is not a predecessor of v set

$$\eta(w) \;:=\; \begin{cases} (a_1, \ldots, a_{p(w)-2}, a_{p(w)} - 1, 2 \cdot |V|, \ldots, 2 \cdot |V|) & , \text{ if } p(w) \text{ is odd,} \\ (a_1, \ldots, a_{p(w)-1}, 2 \cdot |V|, \ldots, 2 \cdot |V|) & , \text{ o.w.} \end{cases}$$

Note that the depth of the quasi-tree $\mathcal{R}(G)$ is at most $2 \cdot |V|$ which is why the starting value for η at v_0 suffices.

It remains to be seen that η is successful. Clearly, if w is a son but not a predecessor of v in $\mathcal{R}(G|_\sigma)$ then $\eta(w) \trianglelefteq_{p(w)} \eta(v)$. Now suppose there is an edge from v to w in $\mathcal{R}(G|_\sigma)$ s.t. w is also a predecessor of v. According to Lemma 1, player $\exists$ also wins the parity game $\mathcal{R}(G|_\sigma)$. Hence, the minimal priority occurring on the path $w_0 \ldots w_n$, with $w = w_0$ and $v = w_n$, is even. Let p_0 be this priority. By the construction above we have $\eta(w_i)^{(p)} = \eta(w_j)^{(p)}$ for all $i, j = 0, \ldots, n$ and all $p < p_0$, in particular $i = 0$ and $j = n$. Furthermore, $\mathcal{R}(G|_\sigma)$ is constructed s.t. $p(w) = p_0 \leq p(w_i)$ for all $i = 0, \ldots, n$. Thus, we also have $\eta(w) \trianglelefteq_{p(w)} \eta(v)$.

Applying Lemmas 3 and 4 shows that there also is a successful μ-annotation for G which concludes the proof. □

Corollary 1. *Let G be a parity game with node set V. There is a successful μ-annotation for G iff there is a successful μ-annotation η for G s.t. for all $v \in V$: if $\eta(v) = (a_1, \ldots, a_p)$ then for all $i = 1, \ldots, p$: $0 \leq a_i < |V|$.*

Proof. Given a successful μ-annotation for G we can apply Theorem 4 twice in order to first obtain a positional winning strategy σ for player $\exists$ and then a successful μ-annotation η' for $\mathcal{R}(G|_\sigma)$ with node set V'. We have $\eta'(v)^{(p)} \leq 2 \cdot |V|$ for all $v \in V'$. The proof of Lemma 4 reduces this to a minimal successful μ-annotation η for $\mathcal{R}(G)$ which is also a successful μ-annotation for G. But then we must have $\eta(v)^{(p)} < |V|$ because the length of a maximal descending chain of natural numbers in any component of the annotations is bounded by the maximal number of transitions on a loop in $G|_\sigma$ which is $|V|$. Finally, the proof of Lemma 3 shows that a successful μ-annotation for $G|_\sigma$ is also a successful μ-annotation for G. □

5 Using Unary Encodings of Annotation Values

Theorems 4 and 1 yield a reduction from PARITY to SAT. Given a parity game $G = (V, E, v_0, p)$, let p_{max} be the maximal priority used in G. We build a propositional formula Φ^1_G, s.t. a satisfying variable assignment for Φ^1_G encodes a successful μ-annotation for G. Φ^1_G contains variables

- S_v for every $v \in V$. They are used to mark nodes that can be visited in a play that is conforming to σ.
- $T_{v,w}$ for every $v \in V$ and every $w \in vE$. They are used to mark edges that player $\exists$ moves along in a play conforming to σ.
- $Y^v_{p,a}$ for every $v \in V$, every odd priority p used in G and every $a \in \{0, \ldots, |V|-1\}$. Intuitively, $Y^v_{p,a}$ asserts that $\eta(v)^{(p)} = a$.

Part Ψ of Φ^1_G expresses that the variables S_v and $T_{v,w}$ represent a positional strategy for player $\exists$. Furthermore, Φ^1_G asserts that this strategy is a winning strategy, i.e. the corresponding annotation values form a successful annotation for G. Finally, Ψ' says that every p-component of every node v must have at least one value.

$$\Phi^1_G := \Psi \wedge \Psi' \wedge \bigwedge_{v \in V} \bigwedge_{w \in vE} T_{v,w} \rightarrow A(v,w)$$

$$A(v,w) := \bigwedge_{\substack{p < p(w) \\ p \text{ odd}}} E(v,w,p) \wedge \begin{cases} L(v,w) & \text{if } p(w) \text{ is odd} \\ 1 & \text{o.w.} \end{cases}$$

$$E(v,w,p) := \bigwedge_{a=0}^{|V|-1} \left(Y^v_{p,a} \rightarrow \bigvee_{b \leq a} Y^w_{p,b} \right)$$

$$L(v,w) := \bigwedge_{a=0}^{|V|-1} \left(Y^v_{p(w),a} \rightarrow \bigvee_{b < a} Y^w_{p(w),b} \right)$$

$$\Psi := S_{v_0} \wedge \bigwedge_{v \in V_\forall} \bigwedge_{w \in vE} (S_v \rightarrow T_{v,w}) \wedge \bigwedge_{v \in V_\exists} (S_v \rightarrow \bigvee_{w \in vE} T_{v,w}) \wedge \bigwedge_{(v,w) \in E} T_{v,w} \rightarrow S_w$$

$$\Psi' := \bigwedge_{v \in V} \bigwedge_{\substack{p \leq p_{max} \\ p \text{ odd}}} \bigvee_{a=0}^{|V|-1} Y^v_{p,a}$$

Formula $A(v,w)$ asserts that $\eta(w) \trianglelefteq_{p(w)} \eta(v)$ for an annotation η given as an assignment to the variables $Y^v_{p,a}$. Formula $E(v,w,p)$ says that $\eta(w)^{(p)} \leq \eta(v)^{(p)}$, and formula $L(v,w)$ says that $\eta(w)^{(p(w))} < \eta(v)^{(p(w))}$.

Theorem 5. *Player $\exists$ has a winning strategy for the parity game G iff Φ^1_G is satisfiable.*

Proof. Suppose player $\exists$ wins the parity game $G = (V, E, v_0, p)$. According to Theorem 1, she has a positional winning strategy σ. This immediately yields

a valuation for the variables S_v and $T_{v,w}$ fulfilling the subformula Ψ. Furthermore, according to Theorem 4, there is a successful μ-annotation η for G. This immediately yields an assignment for the variables $Y^v_{p,a}$ defined by $Y^v_{p,a} = 1$ iff $\eta(v)^{(p)} = a$. It is not hard to see that the remaining conjuncts are all fulfilled by this evaluation.

Now suppose that Φ^1_G is satisfiable. Then, the satisfying variable assignment defines a positional strategy σ for player $\exists$. Next we define an annotation η for G by $\eta(v)^{(p)} = a$ iff $a = \min \{ b \mid Y^v_{p,b} = 1 \}$. Again, it is not hard to see that η is successful. Note that Φ_G asserts that whenever $(v, w) \in E$ and both v and w belong to the strategy, then $\eta(w) \trianglelefteq_{p(w)} \eta(v)$. This implies success of η. □

Most SAT solvers expect their input to be in conjunctive normal form, and their performance depends on the input size as well as the number of variables occurring in it.

Proposition 1. *Given a parity game G with nodes V, edges E and maximal priority p_{max}, there is a Φ' in conjunctive normal form that is equivalent to Φ^1_G w.r.t. satisfiability s.t. $|\Phi'| = O(|V|^2 \cdot (|E| + \lceil \frac{p_{max}}{2} \rceil) + |V| \cdot |E| \cdot \lfloor \frac{p_{max}}{2} \rfloor)$ and the number of variables used in Φ' is $|V| + |E| + |V|^2 \cdot \lceil \frac{p_{max}}{2} \rceil$.*

Proof. In order to transform Φ^1_G to conjunctive normal form, one only needs the equivalence $\varphi \rightarrow (\psi_1 \wedge \psi_2) \equiv (\varphi \rightarrow \psi_1) \wedge (\varphi \rightarrow \psi_2)$. Note that this results in a linear blow-up only since the formulas on the left-hand side of the implication are of constant size. The result of this transformation applied to Φ^1_G is presented in Appendix A. □

6 Using Binary Encodings of Annotation Values

We present a second reduction from PARITY to SAT that proceeds along the same lines but encodes the annotation values binarily. This yields asymptotically smaller formulas and uses less variables. The disadvantage is their transformation into conjunctive normal form which is more complex and requires the introduction of additional variables. However, those variables are merely macros, hence, their values in an assignment are completely determined by the values of the variables used in the first place.

Given a parity game G with nodes V, edges E and maximal priority p_{max}, let $n := |V|$, $m := \lceil \log n \rceil - 1$. In addition to the variables S_v and $T_{v,w}$ for every $v \in V, w \in vE$ as introduced in Section 5, formula Φ^2_G uses variables $X^v_{p,i}$ for every $v \in V$, every odd priority $p \leq p_{max}$, and every $i = 0, \ldots, m$. Intuitively, $X^v_{p,i}$ represents the i-th bit of $\eta(v)^{(p)}$ for an annotation η of G.

$$\Phi^2_G \;:=\; \Psi \;\wedge\; \bigwedge_{v \in V} \bigwedge_{w \in vE} T_{v,w} \rightarrow A(v,w)$$

$$A(v,w) \;:=\; \bigwedge_{\substack{p < p(w) \\ p \text{ odd}}} E^m(v,w,p) \;\wedge\; \begin{cases} L^m(v,w) & \text{if } p(w) \text{ is odd} \\ 1 & \text{o.w.} \end{cases}$$

$$
\begin{aligned}
E^0(v,w,p) &:= X^w_{p,0} \rightarrow X^v_{p,0} \\
E^i(v,w,p) &:= (X^w_{p,i} \rightarrow X^v_{p,i}) \wedge ((X^w_{p,i} \vee \neg X^v_{p,i}) \rightarrow E^{i-1}(v,w,p)) \\
L^0(v,w) &:= \neg X^w_{p(w),0} \wedge X^v_{p(w),0} \\
L^i(v,w) &:= (X^w_{p(w),i} \rightarrow X^v_{p(w),i}) \wedge ((X^w_{p(w),i} \vee \neg X^v_{p(w),i}) \rightarrow L^{i-1}(v,w))
\end{aligned}
$$

where Ψ is as defined in Φ^1_G.

Formula $A(v,w)$ is the same as used in Φ^1_G. Formula $E^i(v,w,p)$ says that the i-th bit of $\eta(w)^{(p)}$ is less or equal than that of $\eta(v)^{(p)}$, and if they are equal then the same holds recursively for the next lower bit. Formula $L^i(v,w)$ does the same for the $p(w)$ components of $\eta(w)$ and $\eta(v)$ but requires them to differ at one bit at least.

The following is proved in just the same way as Theorem 5 with the values encoded binarily instead of unarily. Note that – for fixed v and p – any assignment to the variables $X^v_{p,i}$ defines a value between 0 and $|V|-1$. Thus, it is not necessary to explicitly require one to exist as it is done by Ψ' in Φ^1_G.

Theorem 6. *Player $\exists$ has a winning strategy for the parity game G iff Φ^2_G is satisfiable.*

Proposition 2. *For every parity game G with nodes V, edges E and maximal priority p_{max}, there is a Φ' in conjunctive normal form that is equivalent to Φ^2_G w.r.t. satisfiability s.t. $|\Phi'| = O(|E| \cdot \frac{p_{max}}{2} \cdot \log|V|)$ and the number of variables used in Φ' is*

$$
|V| \cdot (1 + \lceil \frac{p_{max}}{2} \rceil \cdot \lceil \log|V| \rceil) + |E| \cdot (1 + \lceil \log|V| \rceil \cdot (4 \cdot \lceil \frac{p_{max}}{2} \rceil + 2))
$$

Proof. Φ_G can be brought into conjunctive normal form using limited expansion: a subformula ψ that violates the conjunctive normal form is replaced by a new variable ψ, and top-level clauses are added that express $Z_\psi \leftrightarrow \psi$. This is done recursively on ψ's subformulas for as long as they are not literals. □

7 Experimental Results

We report on a prototype implementation of the translation from PARITY to SAT in Section 6. The programming language used for this implementation is the lazy functional language Haskell using the Glasgow Haskell Compiler. The tests were carried out on a machine with two Intel® Xeon™ 2.4 GHz processors and 4GB of RAM. The second processor remained unused. We do not report on the implementation using unary encodings, since its performance was worse than the binary one throughout all the tests.

The program takes parameters n, p and o, and creates a random graph with n nodes. The probability that $v \in V_\exists$ is 50% for each $v \in V$. Moreover, each v is

n	$p_{max} = 1$		$p_{max} = \lceil\sqrt{n}\rceil$	
	reduction	solving	reduction	solving
100	0.05s	0.03s - 0.05s	0.3s	0.1s - 0.2s
200	0.1s	0.06s - 0.1s	1s - 1.2s	0.7s - 11m54s
300	0.2s	0.2s	2.2s - 2.6s	1.1s - 2.1s
400	0.3s	0.2s - 0.3s	4s - 4.2s	1.4s - 4.4s
500	0.4s	0.2s - 0.3s	5.5s - 5.9s	3.3s - 37.3s
600	0.6s	0.4s	8.3s - 9.2s	4.8s - 1m4s
700	0.7s	0.3s - 0.5s	10.9s - 12.1s	3.8s - 1m25s
800	0.8s	0.6s	14s - 15.5s	6s - 1m44s
900	0.9s	0.6s - 0.7s	17s - 20s	9.2s - 2h6m
1000	1s	0.7s - 0.8s	20s - 23s	6.3s - 4m19s
1100	1.2s	0.9s - 1s	29s - 34s	9.8s - 1m11s
1200	1.4s	1s	40s	16s - 6m40s
1300	1.5s	1.1s - 1.5s	45s	18s - †
1400	1.6s	1.2s - 1.3s	52s	20s - 19m48s
1500	1.7s	1.3s	58s - 1m6s	16s - 2h28m
1600	1.8s	1.4s - 1.8s	1m12s	16s - †

Fig. 1. The running times based on binary coding.

assigned a priority that is chosen randomly from an even distribution between 0 and p inclusively. Finally, every v has exactly outdegree o. We have restricted the test runs to $o = 2$ in order to reduce dimensions for presenting the results. Note that every parity game can easily be transformed into an equivalent game with constant outdegree 2. The random graphs are directly translated into propositional formulas in conjunctive normal form and fed to the SAT solver zChaff [15].

Two sets of benchmarks were carried out on such random graphs of successively increasing size n: one with $p = 1$ which corresponds to model checking formulas of the modal μ-calculus with at most one alternation; the other with $p = \lceil\sqrt{n}\rceil$ which seems to be a reasonable choice for a principally unbounded but feasible number of priorities. Each input was processed 5 times. The minimal and maximal running times that occurred in this test series are shown in Figure 1. A value of "†" means that zChaff was aborted manually after running for more than 6h. For $n > 1600$ and $p_{max} = \lceil\sqrt{n}\rceil$ there is an increasing number of instances that do not terminate within 6h. For $p_{max} = 1$, the series shows a steady growth with more diverging running times. For example, checking games with $n = 20000$ takes approx. 1 min for the reduction and 30 sec to 5 min for the solving.

8 Conclusions

We have shown how the apparent power of recent SAT solvers can be used to solve parity games. The experimental results look promising: most parity games

are solved very quickly, but there are a few examples of formulas that could not be solved in reasonable time.

What remains to be done is to

- extend the translation in order to solve parity games globally, i.e. compute the entire winning region of one of the players rather than decide for a designated node only which winning region it belongs to;
- speed up the implementation and make it more memory-efficient by re-implementing it in a different language, for example C++;
- compare the optimised procedure with implementations of other algorithms for parity games, for instance *omega* [18], as well as model checkers for the modal μ-calculus like SMV [14], etc.
- check how this algorithms performs on families of parity games that are known to cause exponential behaviour of other algorithms;
- check whether the formulas created by one of the translations above can be solved in polynomial time.

References

1. G. Bhat and R. Cleaveland. Efficient model checking via the equational μ-calculus. In *Proc. 11th Symp. on Logic in Computer Science, LICS'96*, pages 304–312, New Brunswick, New Jersey, July 1996. IEEE.
2. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992.
3. E. M. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001.
4. R. Cleaveland. Tableau-based model checking in the propositional μ-calculus. *Acta Informatica*, 27(8):725–748, 1990.
5. S. A. Cook. The complexity of theorem-proving procedures. In *Conf. Rec. 3rd Annual ACM Symposium on Theory of Computing, STOC'71*, pages 151–158, Shaker Heights, OH, USA, 1971. ACM.
6. E. A. Emerson. Model checking and the μ-calculus. In N. Immerman and P. G. Kolaitis, editors, *Descriptive Complexity and Finite Models*, volume 31 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, chapter 6. AMS, 1997.
7. E. A. Emerson and C. S. Jutla. Tree automata, μ-calculus and determinacy. In *Proc. 32nd Symp. on Foundations of Computer Science*, pages 368–377, San Juan, Puerto Rico, October 1991. IEEE.
8. E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model checking for the μ-calculus and its fragments. *TCS*, 258(1–3):491–522, 2001.
9. A. Hoffman and R. M. Karp. On nonterminating stochastic games. *Management Science*, 12:359–370, 1966.
10. M. Jurdziński. Deciding the winner in parity games is in $UP \cap$co-UP. *Inf. Process. Lett.*, 68(3):119–124, 1998.
11. M. Jurdziński. Small progress measures for solving parity games. In H. Reichel and S. Tison, editors, *Proc. 17th Ann. Symp. on Theoretical Aspects of Computer Science, STACS'00*, volume 1770 of *LNCS*, pages 290–301. Springer, 2000.

12. M. Jurdziński and J. Vöge. A discrete strategy improvement algorithm for solving parity games. In E. A. Emerson and A. P. Sistla, editors, *Proc. 12th Int. Conf. on Computer Aided Verification, CAV'00*, volume 1855 of *LNCS*, pages 202–215. Springer, 2000.
13. W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117(1):151–155, 1995.
14. K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, Norwell Massachusetts, 1993.
15. M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proc. 38th Design Automation Conference, DAC'01*, June 2001.
16. V. Petersson and S. Vorobyov. A randomized subexponential algorithm for parity games. *Nordic Journal of Computing*, 8(3):324–345, 2001.
17. A. Puri. *Theory of hybrid systems and discrete event systems*. PhD thesis, University of California, Berkeley, 1995.
18. D. Schmitz and J. Vöge. Implementation of a strategy improvement algorithm for finite-state parity games. In S. Yu and A. Pun, editors, *Proc. Int. Conf. on Implementation and Application of Automata, CIAA'00*, volume 2088 of *LNCS*, pages 263–271. Springer, 2000.
19. H. Seidl. Fast and simple nested fixpoint. *Information Processing Letters*, 59(6):303–308, 1996.
20. C. Stirling. Local model checking games. In I. Lee and S. A. Smolka, editors, *Proc. 6th Conf. on Concurrency Theory, CONCUR'95*, volume 962 of *LNCS*, pages 1–11, Berlin, Germany, August 1995. Springer.
21. W. Zielonka. Notes on finite asynchronous automata. *R.A.I.R.O. – Informatique Théorique et Applications*, 21:99–135, 1987.

A Transformation of Φ^1_G to CNF

The following formula in conjunctive normal form is equivalent to Φ^1_G w.r.t. satisfiability.

$$
\begin{aligned}
\Phi' \;:=\; & S_{v_0} \;\wedge \bigwedge_{v\in V_\forall} \bigwedge_{w\in vE} (\neg S_v \vee T_{v,w}) \;\wedge \bigwedge_{v\in V_\exists} (\neg S_v \vee \bigvee_{w\in vE} T_{v,w}) \\
& \wedge \bigwedge_{v\in V} \Big(\bigwedge_{\substack{p \le p_{max} \\ p \text{ odd}}} \bigvee_{a=0}^{|V|-1} Y^v_{p,a} \\
& \qquad \wedge \bigwedge_{\substack{w \in vE \\ p(w) \text{ odd}}} \bigwedge_{a=0}^{|V|-1} \neg T_{v,w} \vee \neg Y^v_{p(w),a} \vee \bigvee_{b<a} Y^w_{p(w),b} \\
& \qquad \wedge \bigwedge_{w\in vE} \big((\neg T_{v,w} \vee S_w) \\
& \qquad\qquad \wedge \bigwedge_{\substack{p < p(w) \\ p \text{ odd}}} \bigwedge_{a=0}^{|V|-1} \neg T_{v,w} \vee \neg Y^v_{p,a} \vee \bigvee_{b\le a} Y^w_{p,b} \big) \Big)
\end{aligned}
$$

B Transformation of Φ^2_G to CNF

In addition to the variables S_v, $T_{v,w}$ and $X^v_{p,i}$ used in Φ^2_G we need variables

- $E^{v,w}_{p,i}$ to abbreviate $E^i(v,w,p)$,
- $L^{v,w}_i$ to abbreviate $L^i(v,w)$,
- $I^{v,w}_{p,i}$ to abbreviate $\neg X^w_{p,i} \vee X^v_{p,i}$,
- $J^{v,w}_{p,i}$ to abbreviate $\neg K^{v,w}_{p,i} \vee E^{v,w}_{p,i-1}$,
- $K^{v,w}_{p,i}$ to abbreviate $X^w_{p,i} \vee \neg X^v_{p,i}$,
- $H^{v,w}_i$ to abbreviate $\neg K^{v,w}_{p(w),i} \vee L^{v,w}_{i-1}$,

Then we can write Φ^2_G in conjunctive normal form as

$$
\begin{aligned}
\Phi' \;:=\; & S_{v_0} \wedge \bigwedge_{v\in V_\forall} \bigwedge_{w\in vE} (\neg S_v \vee T_{v,w}) \;\wedge\; \bigwedge_{v\in V_\exists} (\neg S_v \vee \bigvee_{w\in vE} T_{v,w}) \\
& \wedge \bigwedge_{v\in V} \Bigg(\bigwedge_{\substack{w\,\in\, vE \\ p(w) \text{ odd}}} (\neg T_{v,w} \vee L^{v,w}_m) \;\wedge\; D_L(v,w) \;\wedge\; D_H(v,w) \\
& \qquad \wedge \bigwedge_{w\in vE} \Big((\neg T_{v,w} \vee S_w) \\
& \qquad\qquad \wedge \bigwedge_{\substack{p \le p(w) \\ p \text{ odd}}} D_I(v,w,p) \wedge D_K(v,w,p) \\
& \qquad\qquad \wedge \bigwedge_{\substack{p < p(w) \\ p \text{ odd}}} \big((\neg T_{v,w} \vee E^{v,w}_{p,m}) \\
& \qquad\qquad\qquad \wedge D_E(v,w,p) \wedge D_J(v,w,p) \big) \Big) \Bigg)
\end{aligned}
$$

where

$$
\begin{aligned}
& D_E(v,w,p) \;:= \\
& \quad (\neg E^{v,w}_{p,0} \vee \neg X^w_{p,0} \vee X^v_{p,0}) \;\wedge\; (E^{v,w}_{p,0} \vee X^w_{p,0}) \;\wedge\; (E^{v,w}_{p,0} \vee \neg X^v_{p,0}) \wedge \\
& \quad \bigwedge_{i=1}^{m} (E^{v,w}_{p,i} \vee \neg I^{v,w}_{p,i} \vee \neg J^{v,w}_{p,i}) \;\wedge\; (\neg E^{v,w}_{p,i} \vee I^{v,w}_{p,i}) \;\wedge\; (\neg E^{v,w}_{p,i} \vee J^{v,w}_{p,i})
\end{aligned}
$$

$$
\begin{aligned}
& D_L(v,w) \;:= \\
& \quad (L^{v,w}_0 \vee X^w_{p(w),0} \vee \neg X^v_{p(w),0}) \wedge \\
& \quad (\neg L^{v,w}_0 \vee \neg X^w_{p(w),0}) \;\wedge\; (\neg L^{v,w}_0 \vee X^v_{p(w),0}) \wedge \\
& \quad \bigwedge_{i=1}^{m} (L^{v,w}_i \vee \neg I^{v,w}_{p(w),i} \vee \neg H^{v,w}_i) \wedge (\neg L^{v,w}_i \vee I^{v,w}_{p(w),i}) \wedge (\neg L^{v,w}_i \vee H^{v,w}_i)
\end{aligned}
$$

$$D_I(v,w,p) \;:=$$

$$\bigwedge_{i=1}^{m} \; (\neg I_{p,i}^{v,w} \vee \neg X_{p,i}^{w} \vee X_{p,i}^{v}) \;\wedge\; (I_{p,i}^{v,w} \vee X_{p,i}^{w}) \wedge (I_{p,i}^{v,w} \vee \neg X_{p,i}^{v})$$

$$D_K(v,w,p) \;:=$$

$$\bigwedge_{i=1}^{m} \; (\neg K_{p,i}^{v,w} \vee X_{p,i}^{w} \vee \neg X_{p,i}^{v}) \;\wedge\; (K_{p,i}^{v,w} \vee \neg X_{p,i}^{w}) \;\wedge\; (K_{p,i}^{v,w} \vee X_{p,i}^{v})$$

$$D_J(v,w,p) \;:=$$

$$\bigwedge_{i=1}^{m} \; (\neg J_{p,i}^{v,w} \vee E_{p,i-1}^{v,w} \vee \neg K_{p,i}^{v,w}) \;\wedge\; (J_{p,i}^{v,w} \vee \neg E_{p,i-1}^{v,w}) \;\wedge\; (J_{p,i}^{v,w} \vee K_{p,i}^{v,w})$$

$$D_H(v,w) \;:=$$

$$\bigwedge_{i=1}^{m} \; (\neg H_{i}^{v,w} \vee \neg K_{p(w),i}^{v,w} \vee L_{i-1}^{v,w}) \;\wedge\; (H_{i}^{v,w} \vee K_{p(w),i}^{v,w}) \;\wedge\; (H_{i}^{v,w} \vee \neg L_{i-1}^{v,w})$$