Generalized Parity Games

Krishnendu Chatterjee[1], Thomas A. Henzinger[1,2], and Nir Piterman[2]

[1] University of California, Berkeley, USA
[2] EPFL, Switzerland
c_krish@eecs.berkeley.edu, {tah,Nir.Piterman}@epfl.ch

Abstract. We consider games where the winning conditions are disjunctions (or dually, conjunctions) of parity conditions; we call them *generalized* parity games. These winning conditions, while ω-regular, arise naturally when considering fair simulation between parity automata, secure equilibria for parity conditions, and determinization of Rabin automata. We show that these games retain the computational complexity of Rabin and Streett conditions; i.e., they are NP-complete and co-NP-complete, respectively. The (co-)NP-hardness is proved for the special case of a conjunction/disjunction of two parity conditions, which is the case that arises in fair simulation and secure equilibria. However, considering these games as Rabin or Streett games is not optimal. We give an exposition of Zielonka's algorithm when specialized to this kind of games. The complexity of solving these games for k parity objectives with d parities, n states, and m edges is $O(n^{2kd} \cdot m) \cdot \frac{(k \cdot d)!}{d!^k}$, as compared to $O(n^{2kd} \cdot m) \cdot (k \cdot d)!$ when these games are solved as Rabin/Streett games. We also extend the subexponential algorithm for solving parity games recently introduced by Jurdziński, Patterson, and Zwick to generalized parity games. The resulting complexity of solving generalized parity games is $n^{O(\sqrt{n})} \cdot \frac{(k \cdot d)!}{d!^k}$. As a corollary we obtain an improved algorithm for Rabin and Streett games with d pairs, with time complexity $n^{O(\sqrt{n})} \cdot d!$.

1 Introduction

Games offer a natural framework for reasoning about systems. For example, *controller synthesis* is a natural framework where two-player games arise. We consider the controller that we wish to synthesize as a player in a game against an environment. The controller has to come up with a strategy that will allow it to decide on its action given environment inputs such that regardless of environment actions some goal is satisfied [17].

A *two-player game* is a finite or infinite directed graph where the vertices are partitioned between the two players. A *play* proceeds by moving a token between the vertices of the graph. If the token is found on a vertex of player 1, she chooses an outgoing edge and moves the token along that edge. If the token is found on a vertex of player 2, she gets to choose the outgoing edge. The result is an infinite sequence of vertices. In order to determine the winner in a play we consider the *infinity set*, the set of states occurring infinitely often in the play. Then, there are several methods to define acceptance conditions that determine which infinity sets are winning for which player. We *solve* a game by computing the set of states from which player 1 has a *strategy* to resolve her choices so that regardless of player 2's choices the play is winning, this is called the *winning set* of player 1. In the games considered here, the winning set of player 1 and the winning set of player 2 (defined dually) form a partition of the vertices of the game [12].

The class of *Rabin* [16] and *Streett* [20] winning conditions are cannonical forms to express all ω-regular winning conditions. Both conditions are defined using a set of pairs of subsets of the vertices of the graph. In order to win the Rabin condition over $\{\langle E_1, F_1\rangle, \ldots, \langle E_k, F_k\rangle\}$ the infinity set has to intersect E_i and not intersect F_i for some i. The Streett winning condition is the dual of the Rabin condition. In order to win the Streett condition over $\{\langle E_1, F_1\rangle, \ldots, \langle E_k, F_k\rangle\}$ the infinity set has to either be disjoint from E_i or to intersect F_i for every i. Rabin and Streett games with n vertices, m edges, and k pairs can be solved in time $O(mn^k k!)$ [15].

Another general acceptance condition is the *parity* acceptance condition [6]. In the parity condition, every vertex has a priority and a play is won if the maximal priority visited infinitely often is even. The parity condition is a special case of Rabin and Streett conditions that is closed under complement. While Rabin games are NP-complete (and Streett co-NP-complete) [5], parity games are in NP$\cap$co-NP [6]. Solving a parity game with m edges, n vertices, and $2k$ priorities can be done in time $O(m \cdot n^k)$ [10] or $n^{O(\sqrt{n})}$ [11].

In this paper, we are interested in games where the winning condition is a disjunction (dually conjunction) of parity conditions. That is, instead of considering one function assigning priorities to vertices we consider a set of such functions. A play is winning according to this definition if for one of the functions the maximal priority visited infinitely often is even. We call these winning conditions *generalized parity*.

Generalized parity winning conditions arise naturally in several scenarios. As mentioned, one of the main motivations for considering two-player games is controller synthesis. In the classical setting we consider the system playing against an arbitrary environment. Sometimes, it makes more sense to consider the case where the environment has a goal of its own. In such a case, we are searching for some equilibrium between the system and the environment in which both satisfy their requirements. This led to the introduction of *secure equilibria* [2]. When both players have parity winning conditions the solution of secure equilibria requires considering a game where the winning condition is the implication between two parity conditions. As parity objectives are closed under complement we can think about this as either the disjunction or the conjunction of two parity conditions.

Two-player games arise also in the context of simulation [13, 8]. Simulation is an important precondition for language containment between automata [3] and is also used in the context of minimization of automata [7, 19, 1]. Simulation between parity automata (automata whose acceptance condition is parity) is naturally framed as a game whose winning condition is again the implication of parity conditions. Finally, the disjunction of parity games also arises when considering determinization of Rabin and parity automata. Given a Rabin automaton with one pair, we know how to create an equivalent deterministic parity automaton [18, 14]. It follows that in order to determinize a Rabin automaton with k pairs we can consider the disjunction of deterministic parity automata. The acceptance condition of such an automaton is again a disjunction of parity conditions.

As explained, parity conditions are a special case of Rabin and Streett conditions. It follows that generalized parity conditions are again a special case of Rabin and Streett conditions. Indeed, every parity condition is in particular a Rabin condition and a disjunction of Rabin conditions is again a Rabin condition. Dually, every parity condition is a Streett condition and a conjunction of Streett conditions is again a Streett condition. On the other hand, they are also more general than Rabin and Streett conditions. Indeed a Rabin condition is a disjunction of parity conditions of index 3 and a Streett condition is a conjunction of parity conditions of index 3. It is an interesting question whether generalized parity conditions retain the computational hardness of Rabin and Streett conditions. We would also like to devise specialized algorithms for generalized parity conditions that outperform the natural reduction to Rabin and Streett conditions. These are the two questions considered in this paper.

We show that generalized parity conditions are NP and co-NP complete, suggesting that the computational complexity of Rabin and Streett conditions is retained. Our lower bound applies already for the special case of a disjunction/ conjunction of two parity conditions, which is the case that comes up in secure equilibria and fair-simulation.

We give specialized algorithms that outperform the reduction of generalized parity conditions to Rabin and Streett conditions. Specifically, Zielonka's algorithm [21] when specialized to a disjunction of k parity objectives with d parities works in time proportional to $O(m \cdot n^{2kd} \cdot \frac{(k \cdot d)!}{d!^k})$ (compared to $O(m \cdot n^{2kd} \cdot (k \cdot d)!)$ when these games are solved as Rabin or Streett games). We generalize the techniques of the subexponential algorithm for solving parity games [11] to generalized parity games. The resulting complexity of solving generalized parity games is $n^{O(\sqrt{n})} \cdot \frac{(k \cdot d)!}{d!^k}$. As a corollary we obtain an improved algorithm for Rabin and Streett games with k pairs, with time complexity $n^{O(\sqrt{n})} \cdot k!$, as compared to the previous best known algorithm with time complexity $O(m \cdot n^k \cdot k!)$ [15].

In the full version we also show how to extend the direct rank computation [10, 15] to generalized parity conditions. The resulting complexity of solving generalized parity games is $O(m \cdot n^{kd} \cdot \frac{(k \cdot d)!}{d!^k})$ (as compared to $O(m \cdot n^{kd} \cdot (k \cdot d)!)$).

2 Definitions

We consider turn-based deterministic games played by two-players with conjunction and disjunction of *parity* objectives; we call them *generalized* parity games. We define game graphs, plays, strategies, objectives and notion of winning below.

Game graphs. A *game graph* $G = ((S, E), (S_1, S_2))$ consists of a directed graph (S, E) with a finite state space S and a set E of edges, and a partition (S_1, S_2) of the state space S into two sets. The states in S_1 are player 1 states, and the states in S_2 are player 2 states. For a state $s \in S$, we write $E(s) = \{t \in S \mid (s, t) \in E\}$ for the set of successor states of s. We assume that every state has at least one out-going edge, i.e., $E(s)$ is non-empty for all states $s \in S$.

Plays. A game is played by two players: player 1 and player 2, who form an infinite path in the game graph by moving a token along edges. They start by placing the token on an initial state, and then they take moves indefinitely in the following way. If the token is on a state in S_1, then player 1 moves the token along one of the edges going out of the state. If the token is on a state in S_2, then player 2 does likewise. The result is an infinite path in the game graph; we refer to such infinite paths as plays. Formally, a *play* is an infinite sequence $\langle s_0, s_1, s_2, \ldots \rangle$ of states such that $(s_k, s_{k+1}) \in E$ for all $k \geq 0$. We write Ω for the set of all plays.

Strategies. A strategy for a player is a recipe that specifies how to extend plays. Formally, a *strategy* σ for player 1 is a function $\sigma\colon S^* \cdot S_1 \to S$ that, given a finite sequence of states (representing the history of the play so far) which ends in a player 1 state, chooses the next state. The strategy must choose only available successors, i.e., for all $w \in S^*$ and $s \in S_1$ we have $\sigma(w \cdot s) \in E(s)$. The strategies for player 2 are defined analogously. We write Σ and Π for the sets of all strategies for player 1 and player 2, respectively. Strategies in general require memory to remember the history of plays. An equivalent definition of strategies is as follows. Let M be a set called *memory.* A strategy with memory can be described as a pair of functions: (a) a *memory-update* function $\sigma_u\colon S \times M \to M$ that, given the memory and the current state, updates the memory; and (b) a *next-state* function $\sigma_n\colon S \times M \to S$ that, given the memory and the current state, specifies the successor state. The strategy is *finite-memory* if the memory M is finite. An important special class of strategies are *memoryless* strategies. The strategy is *memoryless* if the memory M is a singleton set. The memoryless strategies do not depend on the history of a play, but only on the current state. Each memoryless strategy for player 1 can be specified as a function $\sigma\colon S_1 \to S$

such that $\sigma(s) \in E(s)$ for all $s \in S_1$, and analogously for memoryless player 2 strategies. Given a starting state $s \in S$, a strategy $\sigma \in \Sigma$ for player 1, and a strategy $\pi \in \Pi$ for player 2, there is a unique play, denoted $\omega(s, \sigma, \pi) = \langle s_0, s_1, s_2, \ldots \rangle$, which is defined as follows: $s_0 = s$ and for all $k \geq 0$, if $s_k \in S_1$, then $\sigma(s_0, s_1, \ldots, s_k) = s_{k+1}$, and if $s_k \in S_2$, then $\pi(s_0, s_1, \ldots, s_k) = s_{k+1}$.

Conjunction and disjunction of parity objectives. We consider game graphs with a conjunction of parity objectives for player 1 and the complementary disjunction of parity objectives for player 2. For a play $\omega = \langle s_0, s_1, s_2, \ldots \rangle \in \Omega$, we define $\mathrm{Inf}(\omega) = \{s \in S \mid s_k = s \text{ for infinitely many } k \geq 0\}$ to be the set of states that occur infinitely often in ω. We also define reachability and safety objectives as they will be useful in the analysis of the algorithms.

1. *Reachability and safety objectives.* Given a set $T \subseteq S$ of states, the reachability objective $\mathrm{Reach}(T)$ requires that some state in T be visited, and dually, the safety objective $\mathrm{Safe}(F)$ requires that only states in F be visited. Formally, the sets of winning plays are $\mathrm{Reach}(T) = \{\langle s_0, s_1, s_2, \ldots \rangle \in \Omega \mid \exists k \geq 0.\ s_k \in T\}$ and $\mathrm{Safe}(F) = \{\langle s_0, s_1, s_2, \ldots \rangle \in \Omega \mid \forall k \geq 0.\ s_k \in F\}$. The reachability and safety objectives are dual in the sense that $\mathrm{Reach}(T) = \Omega \setminus \mathrm{Safe}(S \setminus T)$.
2. *Parity objectives, conjunction and disjunction.* For $d \in \mathbb{N}$, we let $[d] = \{0, 1, \ldots, d\}$ and $[d]_+ = \{1, 2, \ldots, d\}$. Let $p : S \to [d]$ be a function that assigns a *priority* $p(s)$ to every state $s \in S$. The parity objective requires that the maximal priority occurring infinitely often is *even*. Formally, the set of winning plays is $\mathrm{Parity}(p) = \{\omega \in \Omega \mid \max(\mathrm{Inf}(\omega)) \text{ is even}\}$. For a priority function $p : S \to [d]$ we denote by $\overline{p} : S \to [d+1]_+$ the priority function defined as follows: for $s \in S$ we have $\overline{p}(s) = p(s) + 1$. Then we have $\mathrm{Parity}(\overline{p}) = \Omega \setminus \mathrm{Parity}(p)$, i.e., parity objectives are closed under complementation. For $i = 1, 2, \ldots, k$, consider k priority functions $p_i : S \to [d_i]$. The objective $\mathrm{ConjParity}(p_1, p_2, \ldots, p_k)$ is the conjunction of the parity objectives defined by $\mathrm{Parity}(p_i)$, i.e., $\mathrm{ConjParity}(p_1, p_2, \ldots, p_k) = \bigcap_{i=1}^{k} \mathrm{Parity}(p_i)$. Similarly, the objective $\mathrm{DisjParity}(p_1, p_2, \ldots, p_k)$ is the disjunction of the parity objectives defined by $\mathrm{Parity}(p_i)$, i.e., $\mathrm{DisjParity}(p_1, p_2, \ldots, p_k) = \bigcup_{i=1}^{k} \mathrm{Parity}(p_i)$. The conjunction and disjunction of parity objectives are dual in the sense that $\mathrm{ConjParity}(p_1, p_2, \ldots, p_k) = \Omega \setminus \mathrm{DisjParity}(\overline{p}_1, \overline{p}_2, \ldots, \overline{p}_k)$. If all priority functions have range $[d]$ and there are k priority functions, then we refer to this class of conjunction and disjunction of parity objectives as $(\wedge, k, [d])$ and $(\vee, k, [d])$, respectively. Similarly, if all priority functions have range $[d]_+$ and there are k priority functions, then we refer to this class of conjunction and disjunction of parity objectives as $(\wedge, k, [d]_+)$ and $(\vee, k, [d]_+)$, respectively. We refer to parity objectives with priority functions with range $[1]$ as coBüchi objectives and with range $[2]_+$ as Büchi objectives.
3. *Rabin and Streett objectives.* A *Rabin specification* for the game graph G is a finite set $\mathcal{F} = \{\langle E_1, F_1 \rangle, \ldots, \langle E_d, F_d \rangle\}$ of pairs of sets of states, that is, $E_j \subseteq S$ and $F_j \subseteq S$ for all $1 \leq j \leq d$. The pairs in $\mathcal{F}$ are called Rabin pairs. The Rabin specification $\mathcal{F}$ requires that for some Rabin pair $1 \leq j \leq d$, all states in the left set E_j be visited finitely often, and some state in the right set F_j be visited infinitely often. Thus, the Rabin objective defined by $\mathcal{F}$ is the set $\mathrm{Rabin}(\mathcal{F}) = \{\omega \in \Omega \mid (\exists 1 \leq j \leq d)(\mathrm{Inf}(\omega) \cap E_j = \emptyset \wedge \mathrm{Inf}(\omega) \cap F_j \neq \emptyset)\}$ of winning paths. The complements of Rabin objectives are called Streett objectives. A *Streett specification* for G is likewise a set $\mathcal{F} = \{\langle E_1, F_1 \rangle, \ldots, \langle E_d, F_d \rangle\}$ of pairs of set of states $E_j \subseteq S$ and $F_j \subseteq S$. The pairs in $\mathcal{F}$ are called Streett pairs. The Streett specification $\mathcal{F}$ requires that for all Streett pairs $1 \leq j \leq d$, if some state in the left set F_j is visited infinitely often, then some state in the right set E_j is visited infinitely often. Formally, the Streett objective defined by W is the set $\mathrm{Streett}(\mathcal{F}) = \{\omega \in \Omega \mid (\forall 1 \leq j \leq d)(\mathrm{Inf}(\omega) \cap E_j \neq \emptyset \vee \mathrm{Inf}(\omega) \cap F_j = \emptyset)\}$ of winning paths. The Rabin and Streett objectives are dual in the sense that $\mathrm{Streett}(\mathcal{F}) = \Omega \setminus \mathrm{Rabin}(\mathcal{F})$. The parity objectives are a subclass of the Rabin objectives that are closed under complementation. It follows that every parity objective is both a Rabin objective and a Streett objective.

Relationship between objectives. It may be noted that given k priority functions $p_1, p_2, \ldots, p_k$ with range $[2d_1]$, $[2d_2] \ldots$, $[2d_k]$, the disjunction of the parity objectives can be expressed as a Rabin objective with $\sum_{i=1}^{k} d_i$ pairs, and the conjunction of parity objectives can be expressed as a Streett objective with $\sum_{i=1}^{k} d_i$ pairs. Conversely, a Rabin objective $\text{Rabin}(\mathcal{F})$ with k pairs can be expressed as an objective in $(\vee, k, [3]_+)$ as follows: for a pair $\langle E_i, F_i \rangle$ consider the priority function $p_i : S \to [3]_+$ such that $p_i(s) = 3$ if $s \in E_i$ and 2 if $s \in F_i \setminus E_i$ and 1 otherwise; then $\text{DisjParity}(p_1, p_2, \ldots, p_k) = \text{Rabin}(\mathcal{F})$. Similarly, a Streett objective $\text{Streett}(\mathcal{F})$ with k pairs can be expressed as an objective $(\wedge, k, [2])$ as follows: for a pair $\langle E_i, F_i \rangle$ consider the priority function $p_i : S \to [2]$ such that $p_i(s) = 2$ if $s \in E_i$, 1 if $s \in F_i \setminus E_i$ and 0 otherwise; then $\text{ConjParity}(p_1, p_2, \ldots, p_k) = \text{Streett}(\mathcal{F})$.

Winning strategies and sets. Given a game graph G and an objective $\Phi \subseteq \Omega$ for player 1, a strategy $\sigma \in \Sigma$ is a *winning strategy* for player 1 from a state s if for all player 2 strategies $\pi \in \Pi$ the play $\omega(s, \sigma, \pi)$ is winning, i.e., $\omega(s, \sigma, \pi) \in \Phi$. The winning strategies for player 2 are defined analogously. A state $s \in S$ is winning for player 1 with respect to the objective Φ if player 1 has a winning strategy from s. Formally, the set of *winning states* for player 1 with respect to the objective Φ in a game graph G is $W_1^G(\Phi) = \{s \in S \mid \exists \sigma \in \Sigma.\ \forall \pi \in \Pi.\ \omega(s, \sigma, \pi) \in \Phi\}$. Analogously, the set of winning states for player 2 with respect to an objective $\Psi \subseteq \Omega$ is $W_2^G(\Psi) = \{s \in S \mid \exists \pi \in \Pi.\ \forall \sigma \in \Sigma.\ \omega(s, \sigma, \pi) \in \Psi\}$. If the game graph is clear from the context we drop the game graph from the superscript. We say that there exists a memoryless winning strategy for player 1 with respect to the objective Φ if there exists such a strategy from all states in $W_1(\Phi)$; and similarly for player 2.

Theorem 1 (Determinacy and complexity [5]). *The following assertions hold.*

1. *For all game graphs $G = ((S, E), (S_1, S_2))$, for all Streett objectives Φ for player 1, and the complementary Rabin objective $\Psi = \Omega \setminus \Phi$ for player 2, the following assertions hold.*
   - *We have $W_1(\Phi) = S \setminus W_2(\Psi)$.*
   - *There exists a memoryless winning strategy for player 2 and finite-memory winning strategy for player 1.*
2. *Given a game graph G, a Streett objective Φ for player 1, and the complementary Rabin objective $\Psi = \Omega \setminus \Phi$ for player 2, the problem of deciding whether $s \in W_2(\Psi)$ is NP-complete and deciding whether $s \in W_1(\Phi)$ is coNP-complete.*

Observe that for Streett objective Φ and the Rabin objective $\Psi = \Omega \setminus \Phi$ by definition we have $S \setminus W_2(\Psi) = \{s \in S \mid \forall \pi \in \Pi.\ \exists \sigma \in \Sigma.\ \omega(s, \sigma, \pi) \in \Phi\}$. Theorem 1 states that $S \setminus W_2(\Psi) = \{s \in S \mid \exists \sigma \in \Sigma.\ \forall \pi \in \Pi.\ \omega(s, \sigma, \pi) \in \Phi\}$, i.e., the order of the universal and the existential quantifiers can be exchanged.

Closed sets and attractors. Some notions that will play key roles in the analysis of the algorithms are the notion of *closed sets* and *attractors*. We define them below.

Closed sets. A set $U \subseteq S$ of states is a *closed set* for player 1 if the following two conditions hold: (a) for all states $u \in (U \cap S_1)$, we have $E(u) \subseteq U$, i.e., all successors of player 1 states in U are again in U; and (b) for all $u \in (U \cap S_2)$, we have $E(u) \cap U \neq \emptyset$, i.e., every player 2 state in U has a successor in U. A player 1 closed set is also called a *trap* for player 1. The closed sets for player 2 are defined analogously. Every closed set U for player ℓ, for $\ell \in \{1, 2\}$, induces a sub-game graph, denoted $G \upharpoonright U$.

Proposition 1. *Consider a game graph G, and a closed set U for player 2. For every objective Φ for player 1, we have $W_1^{G \upharpoonright U}(\Phi) \subseteq W_1^G(\Phi)$.*

Attractors. Given a game graph G, a set $U \subseteq S$ of states, and a player $\ell \in \{1,2\}$, the set $Attr_\ell(U,G)$ contains the states from which player ℓ has a strategy to reach a state in U against all strategies of the other player; that is, $Attr_\ell(U,G) = W_\ell(\text{Reach}(U))$. The set $Attr_1(U,G)$ can be computed inductively as follows: let $R_0 = U$; let

$$R_{i+1} = R_i \cup \{s \in S_1 \mid E(s) \cap R_i \neq \emptyset\} \cup \{s \in S_2 \mid E(s) \subseteq R_i\} \qquad \text{for all } i \geq 0;$$

then $Attr_1(U,G) = \bigcup_{i\geq 0} R_i$. The inductive computation of $Attr_2(U,G)$ is analogous. For all states $s \in Attr_1(U,G)$, define $rank(s) = i$ if $s \in R_i \setminus R_{i-1}$, that is, $rank(s)$ denotes the least $i \geq 0$ such that s is included in R_i. Define a memoryless strategy $\sigma \in \Sigma$ for player 1 as follows: for each state $s \in (Attr_1(U,G) \cap S_1)$ with $rank(s) = i$, choose a successor $\sigma(s) \in (R_{i-1} \cap E(s))$ (such a successor exists by the inductive definition). It follows that for all states $s \in Attr_1(U,G)$ and all strategies $\pi \in \Pi$ for player 2, the play $\omega(s,\sigma,\pi)$ reaches U in at most $|Attr_1(U,G)|$ transitions.

Proposition 2. *For all game graphs G, all players $\ell \in \{1,2\}$, and all sets $U \subseteq S$ of states, the set $S \setminus Attr_\ell(U,G)$ is a closed set for player ℓ.*

Notation. For a game graph $G = ((S,E),(S_1,S_2))$, a set $U \subseteq S$ and $\ell \in \{1,2\}$, we write $G \setminus Attr_\ell(U,G)$ to denote the game graph $G \upharpoonright (S \setminus Attr_\ell(U,G))$.

3 Computational Complexity

In this section we study the computational complexity of generalized parity games. We will consider $(\vee, k, [d])$ and $(\wedge, k, [d])$ objectives and present complexity results varying both k and d. Observe that if both k and d are constants, then generalized parity games can be solved in polynomial time (by reduction to Rabin and Streett objectives with constant number of pairs). The next theorem completes the complexity analysis.

Theorem 2. *Given a game graph G the following assertions hold.*

1. *For objectives Ψ in $(\vee, k, [d])$ and Φ in $(\wedge, k, [d])$, and a state s: whether $s \in W_2(\Psi)$ can be decided in NP and whether $s \in W_1(\Phi)$ can be decided in coNP.*
2. *For objectives Ψ in $(\vee, k, [3]_+)$ and Φ in $(\wedge, k, [2])$, and a state s: (a) whether $s \in W_2(\Psi)$ is NP-hard and (b) whether $s \in W_1(\Phi)$ is coNP-hard.*
3. *For objectives Φ in $(\vee, k, [2]_+)$, or in $(\wedge, k, [2]_+)$, or in $(\vee, k, [1])$, or in $(\wedge, k, [1])$ and a state s: whether $s \in W_1(\Phi)$ (or $s \in W_2(\Phi)$) can be decided in PTIME.*
4. *For objectives Φ in $(\vee, 1, [d])$ or $(\wedge, 1, [d])$ and a state s: whether $s \in W_1(\Phi)$ (or $s \in W_2(\Phi)$) can be decided in NP $\cap$ co NP.*
5. *For objectives Ψ in $(\vee, 2, [d])$ and Φ in $(\wedge, 2, [d])$, and a state s: whether $s \in W_2(\Psi)$ is NP-hard and whether $s \in W_1(\Phi)$ is coNP-hard.*

Proof. We prove all the cases below.

1. The result follows from reduction to Rabin and Streett objectives, respectively.
2. Observe that by reduction of Rabin and Streett objectives to generalized parity objectives, we have Rabin objectives are subsumed in $(\vee, k, [3]_+)$ and Streett objectives are subsumed in $(\wedge, k, [2])$. Hence the NP-hardness and coNP-hardness follows from Theorem 1.
3. It may be noted that disjunction of Büchi objectives is a Büchi objective, the conjunction of coBüchi objectives is again a coBüchi objective, and the conjunction of Büchi objectives is a generalized Büchi objective. Games with Büchi, coBüchi and generalized Büchi objectives can be solved in polynomial time.

4. The result follows from the complexity of games with parity objectives.
5. We prove the result in Lemma 1. ∎

Lemma 1. *Given a game graph G, an objective Ψ in $(\vee, 2, [d])$, and a state s deciding whether $s \in W_2(\Psi)$ is NP-hard.*

Proof. We present a reduction from the SAT-problem. Consider a SAT formula ψ with clauses $C_0, C_1, \ldots, C_m$ and over boolean variables $x_0, x_1, \ldots, x_n$. We denote by C the set of all clauses and by X the set of all variables. A *literal* is a variable or its negation (i.e, x_i or $\neg x_i$). We denote by l a literal and by L the set of all literals. We now construct a game graph $G = ((S, E), (S_1, S_2))$ and an objective Ψ that is obtained as a disjunction of two parity objectives.

1. *State space and transitions.* We have

$$S_1 = \{s_0\}; \qquad S_2 = C \cup L;$$
$$E = \{(s_0, C_i) \mid C_i \in C\} \cup \{(C_i, l) \mid C_i \in C, l \text{ occurs in } C_i\} \cup \{(l, s_0) \mid l \in L\}$$

   Hence player 1 chooses between the clauses, and in each clause player 2 can choose a literal that makes the clause true, and from the literals the next state is the starting state s_0.

2. *Priority functions.* We specify priority functions $p_1 : S \to [2n]$ and $p_2 : S \to [2n]$ as follows:

$$p_1(s) = \begin{cases} 0 & s \in C; \text{ or } s = s_0; \\ 2k & s = x_k; \\ 2k+1 & s = \neg x_k; \end{cases} \qquad p_2(s) = \begin{cases} 0 & s \in C; \text{ or } s = s_0; \\ 2k & s = \neg x_k; \\ 2k+1 & s = x_k; \end{cases}$$

We analyze the game with objective $\Psi = \mathrm{DisjParity}(p_1, p_2)$ for player 2. Since the objective is a Rabin objective it suffices to analyze the memoryless strategies as candidate winning strategies for player 2. We analyze the following two cases.

1. *Satisfiability implies winning.* Let $A : X \to \{0, 1\}$ be a satisfying assignment for ψ. We define $\widehat{A} : X \to L$ as follows: for $x \in X$ we have $\widehat{A}(x) = x$ if $A(x) = 1$ and $\neg x$ otherwise. Fix a memoryless strategy $\pi : S_2 \to S$ as follows: for $C_i \in C$ pick a literal l_k that appear in C_i and $\widehat{A}(x_k) = l_k$ (such a literal exists since A is a satisfying assignment), and set $\pi(C_i) = l_k$. Now consider any strategy σ for player 1. Let l_j be the maximal literal that appear infinitely often along the play $\omega(s_0, \sigma, \pi)$. Observe that both x_j and $\neg x_j$ cannot appear infinitely often. If $l_j = x_j$, then $\mathrm{Parity}(p_1)$ is satisfied, and if $l_j = \neg x_j$, then $\mathrm{Parity}(p_2)$ is satisfied. Hence we can construct a winning strategy for player 2 from a satisfying assignment.
2. *Winning implies satisfiability.* Consider a pure memoryless strategy π for player 2. If there exists C_j, C_k such that $\pi(C_j) = x_i$ and $\pi(C_k) = \neg x_i$, then we show that π is not winning for player 2; otherwise, it is easy to construct a satisfying assignment from the memoryless strategy π. Consider C_j, C_k such that $\pi(C_j) = x_i$ and $\pi(C_k) = \neg x_i$, and the strategy σ for player 1 that alternates between C_j and C_k at s_0. Then we have $\max(p_\ell(\mathrm{Inf}(\omega(s, \sigma, \pi)))) = \max\{p_\ell(x_i), p_\ell(\neg x_i)\} = 2i + 1$, for $\ell \in \{1, 2\}$. It follows that π is not a winning strategy for player 2.

The result follows. ∎

4 The Classical Algorithm

We first present the classical style algorithm (Zielonka's algorithm) for games with conjunction and disjunction of parity objectives. We first present an informal description of the algorithm; and a formal description of the algorithm is given as Algorithm 1.

Notations. We will consider k priority functions $p_1 : S \to [2d_1], p_2 : S \to [2d_2], \ldots, p_k : S \to [2d_k]$. The objective Φ for player 1 is the conjunction of the parity objectives $\mathrm{ConjParity}(p_1, p_2, \ldots, p_k)$ and the objective for player 2 is the complementary objective $\Psi = \mathrm{DisjParity}(\overline{p}_1, \overline{p}_2, \ldots, \overline{p}_k)$. We will use the following notation: (a) for $p_i : S \to [2d_i]$ we denote by $\mathsf{MaxEven}(p_i) = p^{-1}(2d_i)$ the set of maximal even priority states, and if we consider a sub-game defined a by a subset S_j of states with $p_i : S_j \to [2\widehat{d_i}]$ with $\widehat{d_i} \leq d_i$, then in the sub-game we denote by $\mathsf{MaxEven}(p_i) = p^{-1}(2\widehat{d_i})$ the maximal even priority states in the sub-game; and (b) for $p_i : S \to [2d_i]$ we denote by $\mathsf{MaxOdd}(p_i) = p^{-1}(2d_i - 1)$ the set of maximal odd priority states, and if we consider a sub-game defined a by a subset S_j of states with $p_i : S_j \to [2\widehat{d_i}]$ with $\widehat{d_i} \leq d_i$, then in the sub-game we denote by $\mathsf{MaxOdd}(p_i) = p^{-1}(2\widehat{d_i} - 1)$ the maximal odd priority states in the sub-game.

Informal description of the classical algorithm. The algorithm computes the set of states winning for player 2 according to the disjunction of the parity conditions. We assume that all parity functions are to the range $[1..(2d+1)]$ for some d. If all parity conditions contain only states of priority 1, then obviously player 2 is losing. Indeed, every infinite play visits the maximal priority 1 according to all disjuncts. Suppose that no such void parity condition exists. The algorithm proceeds by choosing one of the disjuncts. Let d denote the maximal odd priority occurring in this disjunct. Then we compute the states from which player 2 wins by visiting priority d finitely often and visiting $d-1$ infinitely often, or eventually avoiding both of them and winning according to the lower priorities of this disjunct or one of the other disjuncts. In order to compute this region, we compute the set of states from which player 1 can force a visit to priority d, clearly we want to avoid these states so we consider the arena without these states. We now search for a trap of player 1 that is composed of two parts: First some states with priority $d-1$ and player 2's attractor to these states, and second, some states that are won with the simpler winning condition. When we find such a trap we conclude that it is winning for player 2, remove it from the arena and continue with the rest. If we do not find such a trap for every one of the disjuncts, we conclude that player 1 wins from all the region that remains.

Correctness and time complexity. The following theorem states the correctness and complexity of Algorithm 1. The correctness proofs in various forms are available in [4, 21, 9]. The exposition and proof here mostly resembles the one of Horn's paper [9].

Theorem 3 (Correctness and running time). *Given a game graph $G = ((S, E), (S_1, S_2))$ and priority functions $p_1 : S \to [2d_1], p_2 : S \to [2d_2], \ldots, p_k : S \to [2d_k]$ the following assertions hold:*

1. *we have*

$$W = W_2(\mathrm{DisjParity}(\overline{p}_1, \overline{p}_2, \ldots, \overline{p}_k)); \qquad S \setminus W = W_1(\mathrm{ConjParity}(p_1, p_2, \ldots, p_k)),$$

where W is the output of Algorithm 1;

2. *and the running time of Algorithm 1 is $O(m \cdot n^{2d}) \cdot \binom{d}{d_1, d_2, \ldots, d_k}$ where $n = |S|$, $m = |E|$ and $d = \sum_{i=1}^{k} d_i$.*

Proof. We first prove the correctness of the algorithm. Suppose that all the disjuncts have only priority 1. Then player 1 wins according to each of the disjuncts on the entire graph. Suppose that there exist some disjuncts where the maximal priority is at least three. We distinguish between two cases.

1. *Case 1.* Consider the case where the procedure returns through line 2.5.1. In this case the algorithm chooses a priority $\overline{p}_i$ and computes a sequence of regions $\overline{W}_1, \overline{W}_2, \ldots, \overline{W}_l$ such that $\overline{W}_l = \emptyset$ and $W_l = H_l$. Forall j the game H_j is a trap for player 1 in G_j and the game G_1 is a trap for player 1 in G. It follows that the only edges of player 1 that go outside $W_l = H_l$ are

Algorithm 1 **Classical algorithm for Disjunction of Parity Objectives**

Input : A 2-player game graph $G = ((S, E), (S_1, S_2))$ and
priority functions $p_1 : S \to [2d_1], p_2 : S \to [2d_2], \dots, p_k : S \to [2d_k]$.
Output: $W_2 \subseteq S$.
1. **return** DisjParityWin(G,$\overline{p}_1, \overline{p}_2, \dots, \overline{p}_k$);

Procedure DisjParityWin(G,$\overline{p}_1, \overline{p}_2, \dots, \overline{p}_k$)
1. **if** (for all $i = 1, 2, \dots, k$ we have $d_i = 0$)
   1.1 **return** $\emptyset$;
2. **foreach** $i = 1, 2, \dots, k$ such that $d_i \neq 0$
   2.1 $G_1 = G \setminus Attr_1(\mathsf{MaxOdd}(\overline{p}_i), G)$;
   2.2 $H_1 := G_1 \setminus Attr_2(\mathsf{MaxEven}(\overline{p}_i), G_1)$; $j := 0$;
   2.3 **repeat**
      2.3.1 $j := j + 1$;
      2.3.2 $W_j :=$ DisjParityWin(H_j, $\overline{p}_1, \overline{p}_2, \dots, \overline{p}_i : H_j \to [2d_i - 2], \dots, \overline{p}_k$);
      2.3.3 $\overline{W}_j = Attr_1(H_j \setminus W_j, G_j)$;
      2.3.4 $G_{j+1} := G_j \setminus \overline{W}_j$;
      2.3.5 $H_{j+1} := G_{j+1} \setminus Attr_2(\mathsf{MaxEven}(\overline{p}_i), G_{j+1})$;
   2.4 **until** $W_j = \emptyset$ or $W_j = H_j$
   2.5 **if** $(W_j = H_j)$
      2.5.1 **return** $Attr_2(G_j, G) \cup$ DisjParityWin($G \setminus Attr_2(G_j, G)$,$\overline{p}_1, \dots, \overline{p_k}$);
   end foreach
3. **return** $\emptyset$;

to G_l. Consider a play that reaches G_l. We partition G_l to the attractor of $\mathsf{MaxEven}(\overline{p}_i)$ and $H_l = W_l$. Player 2 plays in G_l according to the following strategy.
 – If the play is in the attractor to $\mathsf{MaxEven}(\overline{p}_i)$ player 2 attracts to $\mathsf{MaxEven}(\overline{p}_i)$.
 – If the play is in W_l player 2 applies the winning strategy in the subgame DisjParity($H_l, \overline{p}_1, \overline{p}_2, \dots, \overline{p}_i : H_j \to [2d_i - 2], \dots, \overline{p}_k$).

Consider an infinite play that in which player 2 follows this strategy. Clearly, player 1 cannot escape G_l, so the game stays indefinitely in G_l. There are two cases to consider.
 – Either the play gets to the attractor of $\mathsf{MaxEven}(\overline{p}_i)$ infinitely often, in which case $\mathsf{MaxEven}(\overline{p}_i)$ is visited infinitely often and $\mathsf{MaxOdd}(\overline{p}_i)$ finitely often and player 2 wins.
 – Or the play eventually stays in H_l and then player 2 wins according to the winning strategy in the subgame.

2. *Case 2.* Consider the case where the procedure returns through line 3. In this case for every one of the disjuncts $\overline{p}_i$ the algorithm computes a sequence of regions that are winning for player 1. We suggest the following strategy for player 1. The strategy uses the disjunct number i as memory. When playing according to disjunct i, player 1 applies the following strategy.
 – If the play is in the attractor to $\mathsf{MaxOdd}(\overline{p}_i)$ player 1 attracts to $\mathsf{MaxOdd}(\overline{p}_i)$.
 – If the play is in $\mathsf{MaxOdd}(\overline{p}_i)$ player 1 chooses some successor in G and updates her memory to $i + 1$.
 – Otherwise, the play is in some winning region $\overline{W}_j$ and player 1 applies the winning strategy in the appropriate subgame.

Consider an infinite play in which player 1 follows this strategy. If the memory value used by player 1 is changed infinitely often then forall priorities p_i we have the set $\mathsf{MaxOdd}(\overline{p}_i)$ is visited infinitely often and player 1 wins. If eventually the memory is constant then player 1 is playing according to some partition $\overline{W}_1, \dots, \overline{W}_k$. Each of the regions $\overline{W}_j$ is a trap for player 2.

It follows that from $\overline{W}_j$ player 2 can escape by going to some $\overline{W}_{j'}$ for $j' < j$. However, this can happen only a finite number of times. Eventually, the play remains in $\overline{W}_j$ for some constant j and player 1 wins according to the winning strategy in the subgame.

Finally, we consider the complexity of the algorithm. For a disjunct, the algorithm computes at most n times the winning region in a smaller region with one priority function with two less priorities. Then, at least one state is removed and the algorithm resumes on a smaller graph. In addition each attractor computation takes at most time proportional to the size of the transition $O(m)$. Denote the running time of the algorithm by $T(n, d_1, d_2, \ldots, d_k)$, then the following recurrences hold $T(n, d_1, d_2, \ldots, d_k)$.

$$T(n, d_1, d_2, \ldots, d_k) = O(m) + n \cdot \sum_{i=1}^{k} T(n-1, d_1, d_2, \ldots, d_i-1, \ldots, d_k) + T(n-1, k);$$
$$T(n, d_1, d_2, \ldots, d_k) = O(m) + n^2 \cdot \sum_{i=1}^{k} T(n-1, d_1, d_2, \ldots, d_i-1, \ldots, d_k)$$

The bound $T(n, d_1, d_2, \ldots, d_k) \leq O(m \cdot n^{2d}) \cdot \binom{d}{d_1, d_2, \ldots, d_k}$ follows from the second recurrence. ∎

Remarks. In the case of Rabin or Streett objectives the above algorithm is identical to the one in [4, 21, 9]. Indeed, every disjunct has 3 priorities, this means that forall i we have $d_i = 1$ and $\binom{d}{d_1, d_2, \ldots d_k}$ is $d!$. On the other hand, if we reduce the $\mathrm{ConjParity}(p_1, \ldots, p_k)$ to a Streett objective we get $d = \Sigma_{i=1}^{k} d_i$ pairs and the classical Streett algorithm would compute in time $O(m \cdot n^{2d} \cdot d!)$.

5 A New Algorithm

In this section we present a new algorithm for games with disjunction and conjunction of parity objectives. The algorithm is inspired by the algorithm of [11] for parity games. The algorithm is based on the notion of *dominions* and identifying small dominions cheaply. We now define dominions and the complexity to compute non-empty dominions (if they exist).

Dominions. Given a game graph $G = ((S, E), (S_1, S_2))$, with priority functions $p_1, p_2, \ldots, p_k$, we consider the objectives $\Phi = \mathrm{ConjParity}(p_1, p_2, \ldots, p_k)$ and $\Psi = \mathrm{DisjParity}(\overline{p}_1, \overline{p}_2, \ldots, \overline{p}_k)$ for player 1 and player 2, respectively. A set $U \subseteq S$ is a *dominion*

1. for player 1, if there exists a strategy σ for player 1 such that for all strategies π for player 2 and all states $s \in U$ we have $\omega(s, \sigma, \pi) \in \Phi \cap \mathrm{Safe}(U)$;
2. for player 2, if there exists a strategy π for player 2 such that for all strategies σ for player 1 and all states $s \in U$ we have $\omega(s, \sigma, \pi) \in \Psi \cap \mathrm{Safe}(U)$.

An equivalent characterization of dominions is as follows. A set $U \subseteq S$ is a dominion

1. for player 1, if U is a player 2 closed set and player 1 has a winning strategy σ for objective Φ from all states in U in the sub-game $G \upharpoonright U$;
2. for player 2, if U is a player 1 closed set and player 2 has a winning strategy π for objective Ψ from all states in U in the sub-game $G \upharpoonright U$.

Find dominion procedures. We show that given a set U of size ℓ, i.e., $|U| = \ell$, we can verify in time $\ell^{O(\ell)} \cdot O(d \cdot m)$ whether U is a dominion for a player, where $p_i : S \to [2d_i]$, $d = \sum_{i=1}^{k} d_i$, and m is the number of edges. We consider the following steps.

1. Whether U is a closed set for a player can be verified in time $O(m)$.
2. Given U is a closed set, we consider the sub-game $G \upharpoonright U$. The number of pure memoryless strategies for player 2 in this sub-game is at most ℓ^{ℓ}. By memoryless determinacy (Theorem 1) and reduction of disjunction of parity objectives to Rabin objectives, all states $s \in U$ are winning in $G \upharpoonright U$ for player 2 if there is a pure memoryless winning strategy for player 2 from all states in U; and all states $s \in U$ are winning for player 1 if player 1 can win from all $s \in U$ against all pure memoryless strategies for player 2.

3. Once a pure memoryless strategy for a player is fixed we obtain a graph. The winning sets for Rabin and Streett objectives in a graph can be computed in time $O(m \cdot k)$, where m is the number of edges and k is the number of pairs.

Our claim is a consequence of the above facts. We obtain the following lemma characterizing the computation of dominions.

Lemma 2. *Let G be a game graph with n states. Consider priority functions $p_1, p_2, \ldots, p_k$, and objectives $\Phi = \text{ConjParity}(p_1, p_2, \ldots, p_k)$ and $\Psi = \text{DisjParity}(\overline{p}_1, \overline{p}_2, \ldots, \overline{p}_k)$ for player 1 and player 2, respectively. Let $p_i : S \to [2d_i]$ and $d = \sum_{i=1}^{k} d_i$. A dominion for player 1 or player 2 of size at most ℓ, for $\ell \geq 1$, if one exists, can be computed in time $n^{O(\ell)} \cdot O(d)$.*

Proof. The number of possible subsets of size at most ℓ of n is $\sum_{i=1}^{\ell} \binom{n}{i} \leq n^{O(\ell)}$. We generate all possible subsets $U \subseteq S$ of size at most ℓ and then verify whether U is a dominion in time

$$n^{O(\ell)} \cdot O(d \cdot m) \leq n^{O(\ell)+2} \cdot O(d) = n^{O(\ell)} \cdot O(d).$$

The result follows. ∎

We will use the following notation in the sequel. Given a game graph $G = ((S, E), (S_1, S_2))$ with priority functions $p_1, p_2, \ldots, p_k$, and objectives $\Phi = \text{ConjParity}(p_1, p_2, \ldots, p_k)$ and $\Psi = \text{DisjParity}(\overline{p}_1, \overline{p}_2, \ldots, \overline{p}_k)$ for player 1 and player 2, respectively, we denote by *DisjParityDominion*$(G, \overline{p}_1, \overline{p}_2, \ldots, \overline{p}_k, \ell)$ a procedure that returns a dominion of size at most ℓ for player 2 (if one exists) and runs in time $|S|^{O(\ell)} \cdot O(d)$: if the procedure returns emptyset, then all dominions for player 2 have at least $\ell + 1$ states; and similarly, we denote *ConjParityDominion*$(G, p_1, p_2, \ldots, p_k, \ell)$ a procedure that returns a dominion of size at most ℓ for player 1 (if one exists) and runs in time $|S|^{O(\ell)} \cdot O(d)$: if the procedure returns emptyset, then all dominions for player 1 have at least $\ell + 1$ states.

The new algorithm. The new algorithm is based on the following simple observations about the sets obtained in the classical algorithm.

Fact 1. The set G_j obtained in step 2.5.1 of Algorithm 1 is a player 2 dominion in the game G.

Fact 2. The set $H_j \setminus W_j$ obtained in step 2.3.2 of Algorithm 1 is a player 1 dominion in the sub-game G_j.

With the above observations we obtain the new algorithm from the classical algorithm as follows. The formal description of algorithm is presented as Algorithm 2.

1. Before step 2 of the classical algorithm (which correspond to step 3 of Algorithm 2) we invoke *DisjParityDominion*$(G, \overline{p}_1, \overline{p}_2, \ldots, \overline{p}_k, \ell)$ with $\ell = \lceil \sqrt{|S|} \rceil$; and if a non-emptyset U is obtained, then we take U and its player 2 attractor out as a subset of the player 2 winning set and proceed on the sub-game; else we proceed as the classical algorithm.
2. Before step 2.3.2 of the classical algorithm (which correspond to step 3.3.3 of Algorithm 2) we invoke *ConjParityDominion*$(G, p_1, p_2, \ldots, p_k, \ell)$ with $\ell = \lceil \sqrt{|S|} \rceil$; and if a non-emptyset U is obtained, then we take U and its player 1 attractor out and proceed to step 2.3.4 (step 3.3.5 of Algorithm 2); else we proceed as the classical algorithm.

Correctness. The correctness of Algorithm 2 is immediate from the correctness of the classical algorithm and the observation that a dominion U for a player in a game graph G is a subset of the winning set of the player in G (Proposition 1).

Time complexity of Algorithm 2. We now analyze the time complexity of Algorithm 2. Let us denote by $T(n, d_1, d_2, \ldots, d_k)$ the running time of the algorithm on graphs with n states and priority functions $p_1, p_2, \ldots, p_k$ with $p_i : S \to [2d_i]$, for $i = 1, 2, \ldots, k$. Let $d = \sum_{i=1}^{k} d_i$. By

Algorithm 2 New algorithm for Disjunction of Parity Objectives

Input : A 2-player game graph $G = ((S, E), (S_1, S_2))$ and
priority functions $p_1 : S \to [2d_1], p_2 : S \to [2d_2], \dots, p_k : S \to [2d_k]$.
Output: $W_2 \subseteq S$.
1. **return** DisjParityWin(G,$\overline{p}_1, \overline{p}_2, \dots, \overline{p}_k$);

Procedure DisjParityWin(G,$\overline{p}_1, \overline{p}_2, \dots, \overline{p}_k$)
1. **if** (for all $i = 1, 2, \dots, k$ we have $d_i = 0$)
   1.1 **return** $\emptyset$;
2. $U =$ *DisjParityDominion*$(G, \overline{p}_1, \overline{p}_2, \dots, \overline{p}_k, \ell)$ for $\ell = \lceil \sqrt{|S|} \rceil$
   2.1 **if** $(U \neq \emptyset)$
   2.2 **return** $Attr_2(U, G) \cup$ DisjParityWin($G \setminus Attr_2(U, G)$,$\overline{p}_1, \overline{p}_2, \dots, \overline{p}_k$);
3. **foreach** $i = 1, 2, \dots, k$ such that $d_i \neq 0$
   3.1 $G_1 = G \setminus Attr_1(\mathsf{MaxOdd}(\overline{p}_i), G)$;
   3.2 $H_1 := G_1 \setminus Attr_2(\mathsf{MaxEven}(\overline{p}_i), G_1)$; $j := 0$;
   3.3 **repeat**
      3.3.1 $j := j + 1$;
      3.3.2 $U =$ *ConjParityDominion*$(G, p_1, p_2, \dots, p_k, \ell)$ for $\ell = \lceil \sqrt{|S|} \rceil$
         3.3.2.1 **if** $(U \neq \emptyset)$
         3.3.2.2 $\overline{W}_j = Attr_1(U, G_j)$; **goto** step 3.3.5
      3.3.3 $W_j :=$ DisjParityWin(H_j, $\overline{p}_1, \overline{p}_2, \dots, \overline{p}_i : H_j \to [2d_i - 2], \dots, \overline{p}_k$);
      3.3.4 $\overline{W}_j = Attr_1(H_j \setminus W_j, G_j)$;
      3.3.5 $G_{j+1} := G_j \setminus \overline{W}_j$;
      3.3.6 $H_{j+1} := G_{j+1} \setminus Attr_2(\mathsf{MaxEven}(\overline{p}_i), G_{j+1})$;
   3.4 **until** $W_j = \emptyset$ or $W_j = H_j$
   3.5 **if** $(W_j = H_j)$
      3.5.1 **return** $Attr_2(G_j, G) \cup$ DisjParityWin($G \setminus Attr_2(G_j, G)$,$\overline{p}_1, \dots, \overline{p_k}$);
   end foreach
3. **return** $\emptyset$;

Lemma 2 step 2 takes $n^{O(\sqrt{n})} \cdot O(d)$ time. For simplicity we will drop the $O(\cdot)$ from $O(d)$, the whole analysis can be easily carried out with $O(d)$. We now analyze the following cases.

1. If step 2 succeeds, then at least one state is removed and we need to solve a sub-game with one less state (which takes time $T(n-1, d_1, d_2, \dots, d_k)$).
2. If step 2 fails, then any dominion for player 1 in G must have size at least $\sqrt{n}$; hence the dominion G_j discovered at step 3.5.1 must be of size at least $\sqrt{n}$, (as otherwise it would have been discovered in step 2). Hence the DisjParityWin call at step 3.5.1 would require to solve a sub-game of size at most $n - \sqrt{n}$ and this would require time $T(n - \sqrt{n}, d_1, d_2, \dots, d_k)$. We now analyze the loop in step 3.3, and we analyze the work for a priority function and then sum it up for the k priority functions. For a fixed priority function p_i step 3.3.2 can get executed at most n times and by Lemma 2 each time it requires at most $n^{O(\sqrt{n})} \cdot d$ time. Hence the total work of step 3.3.2 requires at most $n \cdot n^{O(\sqrt{n})} \cdot d = n^{O(\sqrt{n})} \cdot d$ time. We now analyze step 3.3.3: since 3.3.3 is invoked upon failure of step 3.3.2, the set $H_j \setminus W_j$ discovered (which is a dominion) is of at least size $\sqrt{n}$. Hence this step gets executed $\sqrt{n}$ times, the first time it is called on a game graph with $n-1$ states and the range of the priority function p_i is $[2d_i - 2]$; and each subsequent time it is called with at most $n - \sqrt{n}$ states and with range of p_i as $[2d_i - 2]$. Hence

the total work of the loop for priority function p_i is

$$n^{O(\sqrt{n})} \cdot d + T(n-1, d_1, d_2, \ldots, d_i - 1, \ldots, d_k) + \sqrt{n} \cdot T(n - \sqrt{n}, d_1, d_2, \ldots, d_i - 1, \ldots, d_k).$$

Hence the total work when step 2 fails is obtained by summing over $i = 1$ to k and then adding $T(n - \sqrt{n}, d_1, d_2, \ldots, d_k)$ (the work after step 3.5.1 on the reduced game graph). That is we obtain the total work when step 2 fails is given by

$$\sum_{i=1}^{k} \Bigg(n^{O(\sqrt{n})} \cdot d + T(n-1, d_1, d_2, \ldots, d_i - 1, \ldots, d_k) \qquad (1)$$

$$+ \sqrt{n} \cdot T(n - \sqrt{n}, d_1, d_2, \ldots, d_i - 1, \ldots, d_k) \Bigg) + T(n - \sqrt{n}, d_1, d_2, \ldots, d_k).$$

Thus we obtain that $T(n, d_1, d_2, \ldots, d_k) = n^{O(\sqrt{n})} \cdot d + \max\{\mathsf{Term}_1, \mathsf{Term}_2\}$, where

$$\begin{array}{ll} \mathsf{Term}_1 = T(n-1, d_1, d_2, \ldots, d_k) & \text{(step 2 succeeds)} \\ \mathsf{Term}_2 = \text{Expression of (1)} & \text{(step 2 fails)} \end{array}$$

The max is taken over the two terms as step 2 may succeed or fail. If $T(n, d_1, d_2, \ldots, d_k) = n^{O(\sqrt{n})} \cdot d + T(n-1, d_1, d_2, \ldots, d_k)$, then easily we obtain that

$$T(n, d_1, d_2, \ldots, d_k) = n^{O(\sqrt{n})} \cdot d \cdot n = n^{O(\sqrt{n})} \cdot d.$$

We now analyze the recurrence $T(n, d_1, d_2, \ldots, d_k) = n^{O(\sqrt{n})} \cdot d + \mathsf{Term}_2$, where Term_2 correspond to the expression of (1). The following lemmas analyze the recurrence. Lemma 3 follows easily by induction.

Lemma 3. *Consider the following recurrence:*

$$T(n, d_1, d_2, \ldots, d_k) = \begin{cases} n^{O(\sqrt{n})} \cdot d + \mathsf{Term}_2 & \text{if } n \geq 2; \\ \binom{d}{d_1, d_2, \ldots, d_k} & \text{otherwise.} \end{cases}$$

Then we have $T(n, d_1, d_2, \ldots, d_k) \leq n^{O(\sqrt{n})} \cdot k \cdot d \cdot \binom{d}{d_1, d_2, \ldots, d_k} \cdot t(n)$, *where*

$$t(n) = \begin{cases} 1 + t(n-1) + (\sqrt{n} + 1) \cdot t(n - \sqrt{n}) & \text{if } n \geq 2; \\ 1 & \text{otherwise} \end{cases}$$

We will now show that the recurrence $t(n) = 1 + t(n-1) + (\sqrt{n}+1) \cdot t(n - \sqrt{n})$ satisfies the bound that $t(n) = n^{O(\sqrt{n})}$. In [11] a similar recurrence was analyzed. In [11] the recurrence $t(n) = 1 + t(n-1) + t(n - \sqrt{n})$ was proved to satisfy the bound $n^{O(\sqrt{n})}$. Our recurrence is more complex. In the next lemma we show that still the bound of [11] can be proved for the recurrence.

Lemma 4. *Consider the recurrence*

$$t(n) = \begin{cases} 1 + t(n-1) + (\sqrt{n} + 1) \cdot t(n - \sqrt{n}) & \text{if } n \geq 2; \\ 1 & \text{otherwise.} \end{cases}$$

Then we have $t(n) = n^{O(\sqrt{n})}$.

Proof. To bound $t(n)$ we will analyze the following tree:

1. there is a root labeled n (this correspond to the term 1 of the recurrence);
2. if $n > 1$, then it has a *left* child labeled $n-1$ and the sub-tree of $t(n-1)$ is attached to this child (this correspond to the term $t(n-1)$ of the recurrence);
3. if $n > 1$, then it has $(\lceil\sqrt{n}\rceil+1)$ *right* children labeled $n-\lfloor\sqrt{n}\rfloor$ and the sub-tree of $t(n-\lfloor\sqrt{n}\rfloor)$ is attached to each of the right children (this correspond to the term $(\sqrt{n}+1)\cdot t(n-\sqrt{n})$ of the recurrence). For simplicity we will drop the ceilings $\lceil\cdot\rceil$ and floors $\lfloor\cdot\rfloor$ below.

The number of nodes in the tree is a bound for our recurrence. We now bound the number of the nodes in the tree. A node in the tree with no sub-tree is referred as a *leaf.*

Length of a path. Any path in the tree from root down to a leaf has length at most n (as the label decrease by at least 1 at every step).

Right children in a path. We now bound the number right children on a path from the root down to a leaf. Consider a path from the root to a leaf and we consider the number of right children possible in a segment of the path between label k and $\frac{k}{2}$. Whenever a right children appear in this segment the label goes down by at least $\sqrt{\frac{k}{2}}$; and hence the number of possible right children in this segment is at most

$$\frac{\frac{k}{2}}{\sqrt{\frac{k}{2}}} = \sqrt{\frac{k}{2}}.$$

Hence the number of right children in a path from root to the leaf can be bounded by considering the bound on segments: n to $\frac{n}{2}$; then $\frac{n}{2}$ to $\frac{n}{4}$; then $\frac{n}{4}$ to $\frac{n}{8}$; and so on. This yields the bound

$$\sqrt{n}\cdot\left(\sum_{i=1}^{\infty}\frac{1}{\sqrt{2^i}}\right) = O(\sqrt{n}).$$

The number of paths. We now bound the number of paths in the tree. The length of a path is at most n; there are at most $O(\sqrt{n})$ right children; every choice of a left child in the path is unique and for every choice of a right children there are at most $(\sqrt{n}+1)$ choices (since any node can have at most $(\sqrt{n}+1)$ right children). Hence we obtain the following bound for the number distinct paths

$$\binom{n}{O(\sqrt{n})}\cdot(\sqrt{n}+1)^{O(\sqrt{n})} = n^{O(\sqrt{n})}.$$

Hence the desired result follows. ∎

Combining all the analysis of the recurrence and the correctness of Algorithm 2 we obtain the following result.

Theorem 4 (Correctness and running time). *Given a game graph $G = ((S,E),(S_1,S_2))$ and priority functions $p_1 : S \to [2d_1], p_2 : S \to [2d_2], \ldots, p_k : S \to [2d_k]$ the following assertions hold:*

1. *we have*

$$W{=}W_2(\mathrm{DisjParity}(\overline{p}_1,\overline{p}_2,\ldots,\overline{p}_k)); \qquad S\setminus W{=}W_1(\mathrm{ConjParity}(p_1,p_2,\ldots,p_k)),$$

where W is the output of Algorithm 2;

2. *and the running time of Algorithm 2 is $n^{O(\sqrt{n})}\cdot O(k\cdot d)\cdot\binom{d}{d_1,d_2,\ldots,d_k}$, where $n = |S|$ and $d = \sum_{i=1}^{k} d_i$.*

Remark. In the special case of Rabin and Streett objectives with k pairs the running time of Algorithm 2 is $n^{O(\sqrt{n})}\cdot O(k^2)\cdot k!$. For comparison, the algorithm in [15] works in time $O(mn^{k+1}kk!)$. We conclude that the algorithm presented above is of better complexity when the number of pairs is larger than $\sqrt{n}$.

References

1. D. Bustan and O. Grumberg. Simulation based minimization. In *Proc. of the 17th International Conference on Automated Deduction*, Pittsburgh, PA, June 2000.
2. K. Chatterjee, T.A. Henzinger, and M. Jurdziński. Games with secure equilibria. In *Proc. 19th IEEE Symposium on Logic in Computer Science*, pages 160–169. IEEE, 2004.
3. D.L. Dill, A.J. Hu, and H. Wong-Toi. Checking for language inclusion using simulation relations. In *Proc. 3rd Conference on Computer Aided Verification*, volume 575 of *Lecture Notes in Computer Science*, pages 255–265, Aalborg, July 1991. Springer-Verlag.
4. S. Dziembowski, M. Jurdziński, and I. Walukiewicz. How much memory is needed to win infinite games. In *Proc. 12th IEEE Symp. on Logic in Computer Science*, pages 99–110, 1997.
5. E.A. Emerson and C. Jutla. The complexity of tree automata and logics of programs. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 328–337, October 1988.
6. E.A. Emerson and C. Jutla. Tree automata, μ-calculus and determinacy. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 368–377, San Juan, October 1991.
7. K. Etessami and G. Holzmann. Optimizing büchi automata. In *11th International Conference on Concurrency Theory*, volume 1877 of *Lecture Notes in Computer Science*, pages 153–167. Springer-Verlag, 2000.
8. T.A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. In *Proc. 8th Conference on Concurrency Theory*, volume 1243 of *Lecture Notes in Computer Science*, pages 273–287, Warsaw, July 1997. Springer-Verlag.
9. F. Horn. Streett games on finite graphs. In *Proc. 2nd Workshop on Games in Design and Verification*, 2005.
10. M. Jurdziński. Small progress measures for solving parity games. In *17th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer-Verlag, 2000.
11. M. Jurdziński, M. Paterson, and U. Zwick. A deterministic subexponential algorithm for solving parity games. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*. ACM/SIAM, 2006.
12. D.A. Martin. Borel determinacy. *Annals of Mathematics*, 65:363–371, 1975.
13. R. Milner. An algebraic definition of simulation between programs. In *Second International Joint Conference on Artificial Intelligence*, pages 481–489. The British Computer Society, 1971.
14. N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *Proc. 25th Symposium on Logic in Computer Science*. IEEE, IEEE press, 2006. to appear.
15. N. Piterman and A. Pnueli. Faster solution of rabin and streett games. In *Proc. 21st Symposium on Logic in Computer Science*. IEEE, IEEE press, 2006.
16. M.O. Rabin. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS*, 141:1–35, 1969.
17. P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *IEEE Transactions on Control Theory*, 77:81–98, 1989.
18. S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, White Plains, October 1988.
19. F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In *Computer Aided Verification, Proc. 12th International Conference*, volume 1855 of *Lecture Notes in Computer Science*, pages 248–263. Springer-Verlag, 2000.
20. R.S. Streett. Propositional dynamic logic of looping and converse. *Information and Control*, 54:121–141, 1982.
21. W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1–2):135–183, 1998.