Solving Odd-Fair Parity Games

Irmak Sağlam
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Anne-Kathrin Schmuck
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

Abstract

This paper discusses the problem of efficiently solving parity games where player Odd has to obey an additional *strong transition fairness constraint* on its vertices – given that a player Odd vertex v is visited infinitely often, a particular subset of the outgoing edges (called *live edges*) of v has to be taken infinitely often. Such games, which we call *Odd-fair parity games*, naturally arise from abstractions of cyber-physical systems for planning and control. In this paper, we present a new Zielonka-type algorithm for solving Odd-fair parity games. This algorithm not only shares *the same worst-case time complexity* as Zielonka's algorithm for (normal) parity games but also preserves the algorithmic advantage Zielonka's algorithm possesses over other parity solvers with exponential time complexity.

We additionally introduce a formalization of Odd player winning strategies in such games, which were unexplored previous to this work. This formalization serves dual purposes: firstly, it enables us to prove our Zielonka-type algorithm; secondly, it stands as a noteworthy contribution in its own right, augmenting our understanding of additional fairness assumptions in two-player games.

2012 ACM Subject Classification Theory of computation → Solution concepts in game theory

Keywords and phrases parity games, strong transition fairness, algorithmic game theory

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.34

Related Version *Extended Version*: `https://arxiv.org/abs/2307.13396` [37]

Funding This work was partially supported by the DFG projects SCHM 3541/1-1 and 389792660 TRR 248–CPEC.

Acknowledgements We are grateful for the immense support provided by Munko Tsyrempilon for the experimental validation.

1 Introduction

Parity games are a canonical representation of ω-regular two-player games over finite graphs, which arise from many core computational problems in the context of correct-by-construction synthesis of reactive software or hardware. In particular, two player games on graphs have been extensively used in the context of cyber-physical system design [41, 6], showing their practical importance. *Fairness*, on the other hand, is a property that widely occurs in this context - both as a desired property to be enforced (e.g., requiring a synthesized scheduler to fairly serve its clients), as well as a common assumption on the behavior of other components (i.e., assuming the network to always eventually deliver a data packet). While *strong fairness* encoded by a Streett condition necessarily incurs a high additional cost in synthesis [14], it is known that the general reactivity(1) (GR(1)) fragment of linear temporal logic (LTL) [7] allows for efficient synthesis in the presence of very restricted fairness conditions. Due to its efficiency, it is extensively used in the context of cyber-physical system design, e.g. [45, 1, 30, 26, 27, 40].

Despite the omnipresence of fairness in such synthesis problems and the success of the GR(1) fragment, not much else is known about tractable fairness constraints in synthesis via two player games on graphs. A notable exception is the recent work by Banerjee et.

al. [5] which considers the sub-class of *strong* ***transition*** *fairness assumptions* [35, 15, 4] which require that whenever the environment player vertex v is visited infinitely often, a particular subset of the outgoing edges (called *live edges*) of v has to be taken infinitely often. In other words, *strong* ***transition*** *fairness assumptions* limit *strong fairness assumptions* to individual transitions. Despite their limited expressive power, such restricted fairness constrains do naturally arise in resource management [8], in abstractions of continuous-time physical processes for planning [9, 10, 34, 11, 36, 2] and controller synthesis [42, 32, 29], which makes them interesting to study.

Concretely, Banerjee et. al. [5] show that *parity games* with strong transition fairness assumptions on player Odd – which we call *Odd-fair parity games* – can be solved via a symbolic fixed-point algorithm in the μ-calculus with almost the same computational worst case complexity as the algorithm for the "normal" version of the same game. The existence of quasi-polynomial time solution algorithms for Odd-fair parity games then follows as a corollary of their nested fixed-point characterization [17, 3, 20]. Unfortunately, it is well known that symbolic fixed-point computations become cumbersome very fast for parity games, as the number of priorities in the game graph increases, leading to high computation times in practice. Given the known inefficiency of existing quasi-polynomial algorithms for parity games [44, 33], despite their theoretical advantages, they are not viable candidates for adoption in the development of efficient solution algorithms for Odd-fair parity games either. For (normal) parity games, computational tractability can be achieved by other algorithms, such as Zielonka's algorithm [46], tangle learning [43] or strategy-improvement [38], implemented in the state-of-the-art tool `oink` [44], with Zielonka's algorithm being widely recognized as the most prominent approach.

The **main contribution** of this paper is a Zielonka-type algorithm, referred to as "*Odd-fair Zielonka's algorithm*", for solving Odd-fair parity games. This novel algorithm meets the efficiency of Zielonka's algorithm while maintaining the same computational worst-case complexity (which is exponential just like the worst-case complexity of the fixed-point algorithm from [5]). Using a prototype implementation, we experimentally verify its efficiency, demonstrating that it matches Zielonka's algorithm in speed, thereby highlighting its comparable performance to fixed-point algorithms for classical parity games.

In contrast to the work by Banerjee et. al. [5], the adaptation and the correctness proof of Odd-fair *Zielonka's algorithm* requires the understanding of Odd player strategies, while [5] studies the solution of such games solely from the Even player's perspective. Unfortunately, Odd strategies are substantially more complex than Even strategies in such games, as they are not positional – while player Even strategies still are (see [5, Thm.3.10]). The **second contribution** of this paper is therefore the formalization of Odd player strategies in Odd-fair parity games, via so called *strategy templates*, which was unexplored prior to this work. We give a constructive proof for the existence of strategy templates winning for Odd from all vertices in the winning region of Odd. This serves dual purposes: firstly, it enables us to prove the correctness of the Odd-fair Zielonka's algorithm; secondly, it stands as a noteworthy contribution in its own right, augmenting our understanding of additional fairness assumptions in two-player games which are currently only unsatisfactorily adressed in various practically motivated synthesis problems.

2 Preliminaries

Notation. We use $\mathbb{N}$ to denote the set of natural numbers including zero and $\mathbb{N}^+$ to denote positive integers. Let Σ be a finite set. Then Σ^* and Σ^ω denote the sets of finite and infinite words over Σ, respectively.

Game graphs. A *game graph* is a tuple $G = (V, V^0, V^1, E)$ where (V, E) is a finite directed graph with *edges* E and *vertices* V partioned into player 0 and player 1 vertices, V^0 and V^1, respectively. Without loss of generality, we can assume that all nodes in V have at least one outgoing edge. Under this assumption, there exist plays from each vertex. A *play* originating at a vertex v_0 is an infinite sequence of vertices $\pi = v_0 v_1 \ldots \in V^\omega$. For $v \in V$, $E(v)$ denotes its successor set $\{w \mid (v, w) \in E\}$.

LTL winning conditions. Given a game graph G, we consider winning conditions specified using a formula Φ in *linear temporal logic* (LTL) over the vertex set V, that is, we consider LTL formulas whose atomic propositions are sets of vertices. In this case the set of desired infinite plays is given by the semantics of Φ which is an ω-regular language $\mathcal{L}(\Phi) \subseteq V^\omega$. The standard definitions of ω-regular languages and LTL are omitted for brevity and can be found in standard textbooks [4]. A game graph G under the winning condition Φ is written as $\langle G, \Phi \rangle$. A play π is winning for player 0 in $\langle G, \Phi \rangle$ if $\pi \in \mathcal{L}(\Phi)$, i.e. $\pi \models \Phi$.

Strategies. A *strategy* for player j over the game graph G is a function $\rho^j : V^* \cdot V^j \to V$ with the constraint that for all $u \cdot v \in V^* \cdot V^j$ it holds that $\rho^j(u \cdot v) \in E(v)$. A play $\pi = v_0 v_1 \ldots \in V^\omega$ is compliant with ρ^j if for all $i \in \mathbb{N}$ holds that $v_i \in V^j$ implies $v_{i+1} = \rho^j(v_0 \ldots v_i)$. A strategy ρ^j is winning from a subset V' of vertices of the game $\langle G, \Psi \rangle$ if all plays π in G that start at a vertex in V' and are compliant with ρ^j are winning w.r.t. Ψ. A strategy ρ is called *positional* iff for all $w_1, w_2 \in V^*$, $\rho(w_1 \cdot v) = \rho(w_2 \cdot v)$.

Parity Games. Parity games are particular two player games over a game graph G where the winning condition is given by a particular mapping of vertices. Formally, a parity game is a tuple $\mathcal{G} = \langle V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi \rangle$, where $(V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E)$ is a game graph and $\chi : V \to \mathbb{N}^+$ is a function which labels each vertex with an integer value, called a *priority*. The players 0 and 1 are called Even and Odd in a parity game and a play $\pi = v_1 v_2 \ldots$ is winning for Even iff $\max\{\inf(\pi)\}$ is *even*, where $\inf(\pi)$ is the set of vertices visited infinitely often in π. Otherwise the play is winning for Odd.

A node $v \in V$ is said to be won by Even, if Even has a (winning) strategy ρ such that all plays $\pi = v \cdot \pi'$ that are compliant with ρ are won by Even. The winning region of Even is the set of all nodes won by Even and is denoted by $\mathcal{W}_{\mathsf{Even}}$. The winning region of Odd, $\mathcal{W}_{\mathsf{Odd}}$, is defined similarly. It is well-known that parity games are determined, that is, all nodes are either in $\mathcal{W}_{Even}$ or in $\mathcal{W}_{Odd}$; and that both players have positional winning strategies from their respective winning regions [12].

Odd-Fair Parity Games. An *Odd-fair parity game* $\mathcal{G}^\ell$ is a tuple $\langle \mathcal{G}, E^\ell \rangle$, where $\mathcal{G} = \langle V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi \rangle$ is a parity game, $E^\ell \subseteq E$ is a set of *live edges* that originate from Odd player vertices and $V^\ell \subseteq V_{\mathsf{Odd}}$, the domain of the relation E^ℓ, is the set of *live vertices*. The live edges induce a *strong transition fairness constraint* – whenever a live vertex v is visited infinitely often, every outgoing live edge $(v, w') \in E^\ell$ needs to be taken infinitely often. Formally, a play π in $\mathcal{G}$ *complies* with E^ℓ if the LTL formula[1]

$$\alpha := \bigwedge_{(v,w) \in E^\ell} (\square \lozenge v \implies \square \lozenge (v \wedge \bigcirc w)) \tag{1}$$

[1] Here, $\square$, $\lozenge$ and $\bigcirc$ stand for the LTL operators 'always', 'eventually' and 'next'.

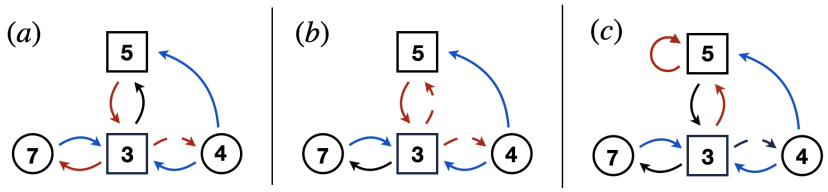

Figure 1 Odd-fair games with player even V_{Even} (circles) and player odd V_{Odd} (squares) vertices (labeled with their priorities). Live edges E^ℓ (dashed) originate from V_{Odd}. Colored player Odd (red) and player Even (blue) edges belong to player Odd's strategy template.

holds along π, i.e. $\pi \models \alpha$. A play π is winning for Even in $\mathcal{G}^\ell$ if and only if $\pi \models \neg\alpha$ or $\max\{\inf(\pi)\}$ is *even*. Dually, π is winning for Odd iff $\pi \models \alpha$ and $\max\{\inf(\pi)\}$ is odd. A strategy ρ over G is therefore winning for Even (resp. Odd) in $\mathcal{G}^\ell$ if all plays compliant with ρ are winning for Even (resp. Odd) in $\mathcal{G}^\ell$.

As the winning condition of a parity game can be equivalently modeled by a suitably defined LTL winning condition, we see that Odd-fair parity games are a special ω-regular game with perfect information. This implies that Odd-fair parity games are determined (by the Borel determinacy theorem [31]) and whenever there exists a winning strategy for Even/Odd in such a game, then there also exists one with *finite* memory [16].

3 Strategy Templates

In this section, we introduce a formalization of player Odd strategies in Odd-fair parity games via *strategy templates*. In contrast to player Even, player Odd winning strategies are no longer positional in Odd-fair parity games, as illustrated by the following example.

▶ **Example 1.** Consider the three different parity games depicted in Fig. 1. In all three games, Odd has a winning strategy from all vertices, i.e., $\mathcal{W}_{Odd} = V$. However, in order to win, the vertex 3 has to be seen infinitely often in game (a) and (b), which forces Odd to use its live edge\s infinitely often. This prevents the existence of a positional strategy for Odd in games (a) and (b): In (a) it needs to somehow alternate between (it's only) live edge to 4 and a "normal" edge to 7 (both indicated in red) in order to win, and in (b) it needs to somehow alternate between all its live edges (also indicated in red). In the game (c), Odd can win by 'escaping' its live vertex 3 to a "normal" vertex 5, and thereby has a positional strategy.

Now consider the subgraph of each game formed by all colored edges (red and blue), which include the strategy choices from V_{Odd} and *all* outgoing edges from V_{Even}. As we have seen that Odd needs to play all red edges repeatably, this subgraph represents the paths that *can* be seen in the game depending on the Even strategy. Hence, a node $v \in V^\ell \subseteq V_{\mathsf{Odd}}$ can be seen infinitely often in a play (compliant with Odd's strategy), if it lies on a cycle in this subgraph. We observe that, in games (a) and (b), node 3 lies on cycles in this subgraph, whereas in game (c), it does not. We further see that whenever a vertex $v \in V^\ell$ lies on a cycle, Odd needs to take all its outgoing live edges (as for vertex 3 in example (b)) and possibly one more edge (as for vertex 3 in example (a)), for all other vertices in V_{Odd} a positional strategy suffices (as for vertex 5 in all examples, and for vertex 3 in example (c)). This shows that Odd strategies are intuitively still "almost positional".

The intuitions conveyed by Ex. 1 are formalized by the following definitions.

▶ **Definition 2** (Odd Strategy Template). *Given an Odd-fair parity game $\mathcal{G}^\ell = \langle \mathcal{G}, E^\ell \rangle$ with $\mathcal{G} = \langle V, V_{Even}, V_{Odd}, E, \chi \rangle$, an Odd* strategy template *$\mathcal{S}$ over $\mathcal{G}^\ell$ is a subgraph of $\mathcal{G}$ given as follows: $\mathcal{S} := (V', E')$ where $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$ such that the following hold,*

- *if $v \in V_{Odd} \cap V'$ does not lie on a cycle in (V', E'), then $|E'(v)| = 1$,*
- *if $v \in V_{Odd} \cap V'$ lies on a cycle in (V', E') then $E^\ell(v) \subseteq E'(v)$ and $1 \leq |E'(v)| \leq |E^\ell(v)|+1$,*
- *if $v \in V_{Even} \cap V'$, then $E'(v) = E(v)$.*

▶ **Definition 3.** *Let $\mathcal{G}^\ell = \langle \mathcal{G}, E^\ell \rangle$ be an Odd-fair parity game with Odd strategy template $\mathcal{S} = (V', E')$, and $V'_{Odd} := V' \cap V_{Odd}$. Then an Odd strategy ρ is said to be* ***compliant*** *with $\mathcal{S}$ if it is a winning strategy in the game $\langle G, \alpha' \rangle$ where $G = (V, V_{Even}, V_{Odd}, E)$ and*

$$\alpha' := \bigwedge_{v \in V'_{Odd}} (\Box (v \implies \bigvee_{(v,w) \in E'} \bigcirc w)) \tag{2a}$$

$$\wedge \bigwedge_{v \in V'_{Odd}} (\Box \Diamond v \implies \bigwedge_{(v,w) \in E'} \Box \Diamond (v \wedge \bigcirc w)). \tag{2b}$$

Intuitively, for all Odd vertices in $\mathcal{S}$, the strategy ρ compliant with $\mathcal{S}$ takes only their outgoing edges in $\mathcal{S}$ (2a), and if a play visits an Odd node v infinitely often, then ρ takes each of v's outgoing edges in $\mathcal{S}$ infinitely often (2b). For an Odd strategy template $\mathcal{S}$, if $v \in V'_{\mathsf{Odd}}$ lies on a cycle in $\mathcal{S}$, then by Def. 2, $\mathcal{S}$ contains all live outgoing edges of v. By (2b) any Odd strategy ρ compliant with $\mathcal{S}$ satisfies the fairness condition in (1) for v. On the other hand, if $v \in V'_{\mathsf{Odd}}$ does not lie on a cycle in $\mathcal{S}$, then by (2a) any such ρ sees v at most once. Thus ρ trivially satisfies (1) for v. This observation is stated in the following proposition.

▶ **Proposition 4.** *Given the premisses of Def. 3 let π be a play starting from a node in V' that complies with ρ. Then $\pi \models \alpha$ where α is the LTL formula in* (1).

Next, we define Even strategy templates. Each Even strategy template encodes a unique Even positional strategy, which is known to exist in Odd-fair parity games [23], due to the lack of fair edges defined on Even vertices.

▶ **Definition 5.** *Given an Odd-fair parity game $\mathcal{G}^\ell = \langle \mathcal{G}, E^\ell \rangle$ with $\mathcal{G} = \langle V, V_{Even}, V_{Odd}, E, \chi \rangle$, an Even* strategy template *$\mathcal{S}$ over $\mathcal{G}^\ell$ is a subgraph of $\mathcal{G}$ given as $\mathcal{S} := (V', E')$ where $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$ such that,*

- *if $v \in V_{Even} \cap V'$, then $|E'(v)| = 1$,*
- *if $v \in V_{Odd} \cap V'$, then $E'(v) = E(v)$.*

An Even strategy ρ is compliant with the Even strategy template $\mathcal{S} = (V', E')$ if for all $v \in V'_{\mathsf{Even}}$, $\rho(v) = E'(v)$. In other words, ρ is the positional strategy defined by $\mathcal{S}$.

Let ρ be an Odd (Even) strategy, compliant with the Odd (Even) strategy template $\mathcal{S}$ and let π be a play compliant with ρ. Then we call π a play *compliant with* $\mathcal{S}$.

▶ **Definition 6.** *An Odd (Even) strategy template $\mathcal{S} = \langle V', E' \rangle$ is* winning *in the Odd-fair parity game $\mathcal{G}^\ell$ if all Odd (Even) strategies ρ compliant with $\mathcal{S}$ are winning for player Odd (Even) in $\mathcal{G}^\ell$ from V'. A winning Odd (Even) strategy template $\mathcal{S}$ is called* maximal *if $V' = \mathcal{W}_{Odd}$ ($\mathcal{W}_{Even}$).*

We note that maximal winning Odd (Even) strategy templates $\mathcal{S}$ immediately imply that for every vertex $v \in \mathcal{W}_{\mathsf{Odd}}$ ($\mathcal{W}_{\mathsf{Even}}$) there exists a winning strategy for player Odd (Even) from v that is compliant with $\mathcal{S}$. The existence of maximal winning Even strategy templates follows from the existence of positional Even strategies [23]. The first main contribution of this paper is a constructive proof showing the existence of maximal winning Odd strategy templates given in the next section. This result is then used in Sec. 5 to prove the correctness of Odd-fair Zielonka's algorithm, which is introduced there.

4 Existence of Maximal Winning Odd Strategy Templates

This section proves the existence of maximal winning Odd strategy templates[2] in Odd-fair parity games, formalized in the following theorem.

▶ **Theorem 7.** *Given an Odd-fair parity game $\mathcal{G}^\ell$, there exists a maximal winning Odd strategy template.*

We prove Thm. 7 by giving an algorithm which constructs $\mathcal{S}$ from a ranking function induced by a fixed-point algorithm in the μ-calculus which computes $\mathcal{W}_{\mathsf{Odd}}$. Towards this goal, Sec. 4.1 first introduces necessary preliminaries, Sec. 4.2 gives the fixed-point algorithm to compute $\mathcal{W}_{\mathsf{Odd}}$ and Sec. 4.3 formalizes how to extract a strategy template $\mathcal{S}$ from the ranking induced by this fixed-point and proves that $\mathcal{S}$ is indeed maximal and winning.

While this section uses fixed-point algorithms extensively to *construct* a maximal winning Odd strategy template towards a *proof* of Thm. 7, we note again that the proof of the new Zielonka's algorithm given in Sec. 5 only uses the *existence* of templates (i.e., the fact that Thm. 7 holds) and does not utilize their *construction* via the algorithm presented here.

4.1 Preliminaries on Fixed-Point Algorithms

This subsection contains the basic notation used in this section.

Set Transformers. Let $G = (V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E)$ be a game graph, $S, T \subseteq V$ and Λ be the player index.[3] Then we define the following predecessor operators:

$$\begin{aligned} \mathsf{Pre}^{\exists}_{\Lambda}(S) &:= \{v \in V_\Lambda \mid E(v) \cap S \neq \emptyset\} & \mathsf{Lpre}^{\exists}(S) &:= \{v \in V_{\mathsf{Odd}} \mid E^\ell(v) \cap S \neq \emptyset\} \\ \mathsf{Pre}^{\forall}_{\Lambda}(S) &:= \{v \in V_\Lambda \mid E(v) \subseteq S\} & \mathsf{Lpre}^{\forall}(S) &:= \{v \in V_{\mathsf{Odd}} \mid E^\ell(v) \subseteq S\} \end{aligned} \tag{3}$$

The predecessor operators $\mathsf{Pre}^{\exists}_{\Lambda}(S)$ and $\mathsf{Pre}^{\forall}_{\Lambda}(S)$ compute the sets of vertices with *at least one* successor and with *all* successors in S, respectively. The live predecessor operators $\mathsf{Lpre}^{\exists}(S)$ and $\mathsf{Lpre}^{\forall}(S)$ restrict this analysis to live edges. We see that

$$\neg\mathsf{Pre}^{\exists}_{\Lambda}(\neg S) = V_{\neg\Lambda} \cup \mathsf{Pre}^{\forall}_{\neg\Lambda}(S) \qquad \text{and} \qquad \neg\mathsf{Lpre}^{\exists}(\neg S) = V_{\mathsf{Even}} \cup \mathsf{Lpre}^{\forall}(S) \tag{4}$$

where for a set $X \subseteq V$, $\neg X$ stands for $V \setminus X$. We combine the pre-operators from (3) into the combined set:[4]

$$\mathsf{Cpre}_\Lambda(S) := \mathsf{Pre}^{\exists}_{\Lambda}(S) \cup \mathsf{Pre}^{\forall}_{\neg\Lambda}(S) \tag{5a}$$

$$\mathsf{Apre}(S,T) := \mathsf{Cpre}_{\mathsf{Even}}(T) \cup (\mathsf{Lpre}^{\exists}(T) \cap \mathsf{Pre}^{\forall}_{\mathsf{Odd}}(S)) \tag{5b}$$

$$\mathsf{Npre}(S,T) := \mathsf{Cpre}_{\mathsf{Odd}}(T) \cap (V_{\mathsf{Even}} \cup \mathsf{Lpre}^{\forall}(T) \cup \mathsf{Pre}^{\exists}_{\mathsf{Odd}}(S)) \tag{5c}$$

The *controllable predecessor operator* $\mathsf{Cpre}_\Lambda(S)$ computes the set of vertices from which player Λ can force visiting S in *one* step. It immediately follows that

$$\neg\mathsf{Cpre}_{\mathsf{Even}}(\neg S) := \mathsf{Cpre}_{\mathsf{Odd}}(S). \tag{6}$$

---

[2] In the rest of this section, we will sometimes call Odd strategy templates simply, *strategy templates*, since these are the only strategy templates we will be dealing with.

[3] $\Lambda \in \{\mathsf{Even}, \mathsf{Odd}\}$ where $\Lambda = \mathsf{Even}$ implies $\neg\Lambda = \mathsf{Odd}$, and vice versa.

[4] Note that $\mathsf{Apre}(S,T)$ and $\mathsf{Npre}(S,T)$ are meaningful only when $T \subseteq S$ and $S \subseteq T$, respectively. Otherwise they are equivalent to $\mathsf{Cpre}_{\mathsf{Even}}(T)$ and $\mathsf{Cpre}_{\mathsf{Odd}}(T)$. We note that these preconditions will always be satisfied in our calculations due to the monotonicity of fixed-point computations.

The *almost-sure controllable predecessor* operator $\mathsf{Apre}(S,T)$ computes the set of states that can be controlled by Player Even to stay in T (via $\mathsf{Cpre}_{\mathsf{Even}}(T)$) as well as all Player Odd states in V^ℓ that (a) will eventually make progress towards T if Player Odd obeys its fairness-assumptions (via $\mathsf{Lpre}^{\exists}$) and (b) will never leave S in the "meantime" (via $\mathsf{Pre}^{\forall}_{\mathsf{Odd}}(S)$)). Using (4) and (6) we have $\mathsf{Npre}(S,T) := \neg\mathsf{Apre}(\neg S, \neg T)$.

Fixed-point Algorithms in the μ-calculus. The μ-calculus offers a succinct representation of symbolic algorithms (i.e., algorithms manipulating sets of vertices instead of individual vertices) over a game graph G. We omit the (standard) syntax and semantics of μ-calculus formulas (see [25]) and only discuss their evaluation on an example fixed-point algorithm given by a 2-nested μ-calculus formula of the form $Z = \mu Y.\ \nu X.\ \phi(X,Y)$, where $X, Y \subseteq V$ are subsets of vertices and μ and ν denote, respectively, the least and the greatest fixed-point. ϕ is a formula composed from the *monotone set transformers* in (3) and (5).

Given this formula, first, both formal variables X and Y are initialized. As Y (resp. X) is preceded by μ (resp. ν) it is initialized with $Y^0 := \emptyset$ (resp. $X^0 := V$). Now we first keep Y at its initial value and iteratively compute $X^k = \phi(X^{k-1}, Y^0)$ until $X^{k+1} = X^k$. At this point X saturates, denoted by X^∞. We then "copy" X^∞, to Y, i.e., have $Y^1 := X^\infty$, reinitialize $X^0 := \emptyset$, and re-evaluate $X^k = \phi(X^{k-1}, Y^1)$ with the new value of Y. This calculation terminates if Y saturates, i.e., $Y^\infty = Y^{l+1} = X^l$ for some $l \geq 0$, and outputs $Z = Y^\infty$. In order to remember all intermediate values of X we use $X^{l,k}$ to denote the set computed in the k-th iteration over X during the computation of Y^l. I.e., $Y^l = X^{l,\infty}$.

Additional Notation. We will use the letters l, m and n exclusively to denote *even* positive integers. For $a \leq b \in \mathbb{N}$, we will use the regular set symbol $[a,b]$ to denote the set of all integers between a and b, i.e., $[a,b] := \{a, a+1, \ldots, b\}$; and $[\![a,b]\!]$ to denote all the *even* integers between a and b. E.g. $[\![2,7]\!] = \{2,4,6\}$. In addition, given an Odd-fair parity game $\mathcal{G}^\ell$, we define the sets $C_i := \{v \in V \mid \chi(v) = i\}$ and $\overline{C_i} := V \setminus C_i$ to ease notation. We say $\mathcal{G}^\ell$ has the least even upper bound l if $C_l \cup C_{l-1} \neq \emptyset$ and $C_i = \emptyset$ for all $i > l$.

4.2 A Fixed-Point Algorithm for $\mathcal{W}_{\mathsf{Odd}}$

Given an Odd-fair parity game $\mathcal{G}^\ell = \langle\langle V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi\rangle, E^\ell\rangle$ this section presents a fixed-point algorithm in the μ-calculus which computes the winning region $\mathcal{W}_{\mathsf{Odd}}$ of player Odd in Odd-fair parity games. It is obtained by negating the fixed-point formula computing $\mathcal{W}_{\mathsf{Even}}$ in [5], formalized in the following proposition and proven in the extended version of this work [37].

▶ **Proposition 8.** *Given an Odd-fair parity game $\mathcal{G}^\ell = (\langle V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi\rangle, E^\ell)$ with least even upper bound $l \geq 0$ it holds that $Z = \mathcal{W}_{\mathsf{Odd}}$, where*

$$Z := \mu Y_l.\ \nu X_{l-1}.\ \ldots \mu Y_2.\ \nu X_1.\ \bigcap_{j\in[\![2,l]\!]} \mathcal{B}_j[Y_j, X_{j-1}], \tag{7}$$

$$\text{where}\quad \mathcal{B}_j[\mathbf{Y},\mathbf{X}] := \left(\bigcup_{i\in[j+1,l]} C_i\right) \cup \left(\overline{C_j} \cap \mathsf{Npre}(\mathbf{Y},\mathbf{X})\right) \cup \left(C_j \cap \mathsf{Cpre}_{\mathsf{Odd}}(\mathbf{Y})\right).$$

Before utilizing (7) we illustrate its computations via an example.

▶ **Example 9.** Consider the Odd-fair parity game $\mathcal{G}^\ell$ depicted in Fig. 2 (left). Here, the name of the vertices coincide with their priorities, e.g., $C_2 = \{2a, 2b, 2c\}$. V_{Even} and V_{Odd} are indicated by circles and squares, respectively. Edges in E^ℓ are shown by dashed lines. As the least even upper bound in this example is $l = 4$,

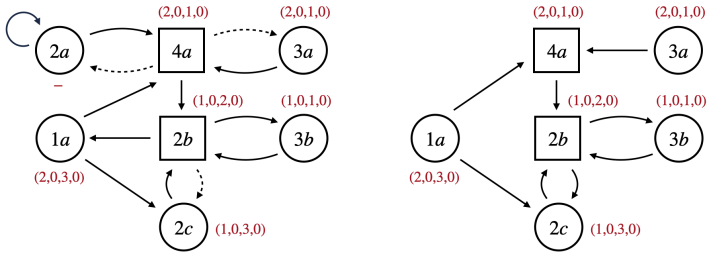

Figure 2 Odd-fair parity game $\mathcal{G}^\ell$ discussed in Ex. 9, 10, and 13 (left) and its corresponding minimum rank based maximal Odd strategy template $\mathcal{S}^{\mathcal{G}^\ell}$ as defined in Def. 12 (right).

$$Z = \mu Y_4.\ \nu X_3.\ \mu Y_2.\ \nu X_1.\ \Phi^{Y_4,X_3,Y_2,X_1} \quad \text{where} \tag{8}$$
$$\Phi^{Y_4,X_3,Y_2,X_1} := (\overline{C_4} \cap \mathsf{Npre}(Y_4, X_3)) \cup (C_4 \cap \mathsf{Cpre}_{\mathsf{Odd}}(Y_4))) \\ \cap (\overline{C_2} \cap \mathsf{Npre}(Y_2, X_1)) \cup (C_2 \cap \mathsf{Cpre}_{\mathsf{Odd}}(Y_2)) \cup C_4 \cup C_3).$$

Using the notation defined in Sec. 4.1, we initialize (8) by $Y_4^0 = \emptyset$, $X_3^{0,0} = V$, $Y_2^{0,0,0} = \emptyset$ and $X_1^{0,0,0,0} = V$ and observe from (5) that $\mathsf{Cpre}_{\mathsf{Odd}}(\emptyset) = \emptyset$ and $\mathsf{Npre}(\emptyset, V) = V$. We obtain

$$\begin{aligned} X_1^{0,0,0,1} &= \Phi^{Y_4^0,X_3^{0,0},Y_2^{0,0,0},X_1^{0,0,0,0}} = ((\overline{C_4} \cap \mathsf{Npre}(\emptyset, V)) \cup (C_4 \cap \mathsf{Cpre}_{\mathsf{Odd}}(\emptyset))) \cap \\ &\quad ((\overline{C_2} \cap \mathsf{Npre}(\emptyset, V)) \cup (C_2 \cap \mathsf{Cpre}_{\mathsf{Odd}}(\emptyset)) \cup C_4 \cup C_3) \\ &= (\overline{C_4}) \cap (\overline{C_2} \cup C_4 \cup C_3) = C_3 \cup C_1 \\ X_1^{0,0,0,2} &= \Phi^{Y_4^0,X_3^{0,0},Y_2^{0,0,0},X_1^{0,0,0,1}} \\ &= C_3 \cup (C_1 \cap \mathsf{Npre}(Y_2^{0,0,0}, X_1^{0,0,0,1})) = C_3 \cup (C_1 \cap \mathsf{Npre}(\emptyset, C_3 \cup C_1)) = C_3 \end{aligned}$$

where $\mathsf{Npre}(\emptyset, C_3 \cup C_1) = \emptyset$ as $v \in \mathsf{Npre}(\emptyset, C_3 \cup C_1)$ implies $v \in \mathsf{Cpre}_{\mathsf{Odd}}(C_3 \cup C_1) = \{2b, 4a\}$ and $v \in V_{\mathsf{Even}} \cup \mathsf{Lpre}^{\forall}(C_3 \cup C_1)$. However, $2b, 4a$ are Odd vertices with live outgoing edges to $2a, 2c \in (V \setminus (C_3 \cup C_1))$. In the next iteration, we again get $X_1^{0,0,0,3} = C_3$ and thus X_1 saturates with C_3. Therefore, $Y_2^{0,0,1} = C_3$. Now the next round of computations of Φ results in

$$\begin{aligned} X_1^{0,0,1,1} &= \Phi^{Y_4^0,X_3^{0,0},Y_2^{0,0,1},X_1^{0,0,1,0}} = C_3 \cup (C_1 \cap \mathsf{Npre}(Y_2^{0,0,1}, X_1^{0,0,1,0})) \cup (C_2 \cap \mathsf{Cpre}_{\mathsf{Odd}}(Y_2^{0,0,1})) \\ &= C_3 \cup (C_1 \cap \mathsf{Npre}(C_3, V)) \cup (C_2 \cap \mathsf{Cpre}_{\mathsf{Odd}}(C_3)) = C_3 \cup C_1 \cup \{2b\} \\ X_1^{0,0,1,2} &= \Phi^{Y_4^0,X_3^{0,0},Y_2^{0,0,1},X_1^{0,0,1,1}} = C_3 \cup \{2b\} = X_1^{0,0,1,3} \end{aligned}$$

Here C_1 and $\{2b\}$ get added in $X_1^{0,0,1,1}$ as $1a \in \mathsf{Npre}(C_3, V)$ trivially and $2b \in \mathsf{Cpre}_{\mathsf{Odd}}(C_3)$ due to the edge $(2b, 3b)$. C_1 is removed from $X_1^{0,0,1,2}$ since $1a$ cannot be forced by Odd to $C_1 \cup C_3 \cup \{2b\}$ in the next step. The fixed-point calculation proceeds in a similar fashion, until Y_4 reaches its saturation value $V \setminus \{2a\}$. The full computation of Z is given in [37].

4.3 Construction of a Rank-based Strategy Template

Given an Odd-fair parity game $\mathcal{G}^\ell$ with the least even priority upper bound $l \geq 0$, we define a ranking function $\mathsf{rank} : \mathcal{W}_{\mathsf{Odd}} \to \mathbb{N}^l$ first introduced in [39] and highly related to "progress measures" [24, 23, 22, 19]. Intuitively, $\mathsf{rank}(v)$ indicates in which iteration v was added to Z in (7) and never got removed from Z again, as illustrated by the following example.

▶ **Example 10.** Consider again the Odd-fair parity game depicted in Fig. 2. Here, $\mathsf{rank}(v)$ of each $v \in \mathcal{W}_{\mathsf{Odd}} = V \setminus \{2a\}$ is shown in red next to the node in the figure. Intuitively, the $4-$tuple is associated with the subscript Y_4, Y_3, Y_2, Y_1 of Φ in (8). For instance $\mathsf{rank}(3a) = (2,0,1,0)$ indicates that $3a$ was added to Z during the first iteration of Y_2 inside the second iteration of Y_4. More concretely, $3a \notin Y_4^0, 3a \notin Y_4^1, 3a \in Y_4^2$. So 2 is the first iteration of the Y_4 variable in which $3a$ got included in the variable. For Y_2, $3a \notin Y_2^{2,0,0}$ and $3a \in Y_2^{2,0,1}$, and therefore $\mathsf{rank}(3a) = (2,0,1,0)$.

The intuition of Ex. 10 is formalized in the following definition.

▶ **Definition 11** (rank). *Given an Odd-fair parity game $\mathcal{G}^\ell = (\langle V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi\rangle, E^\ell)$ with least even upper bound $l \geq 0$ and winning region $\mathcal{W}_{\mathsf{Odd}} \subseteq V$, we define the ranking function $\mathsf{rank} : \mathcal{W}_{\mathsf{Odd}} \to \mathbb{N}^l$ for $v \in \mathcal{W}_{\mathsf{Odd}}$ such that*

$$\mathsf{rank}(v) = (r_l, 0, r_{l-1}, 0 \ldots r_2, 0) \quad \text{if} \quad v \in \textstyle\bigcap_{j \in [\![2,l]\!]} Y_j^{r_l,0,\ldots,r_j} \setminus Y_j^{r_l,0,\ldots,r_j-1}. \tag{9}$$

where the valuations of the variables Y_j are obtained from the iterations of the fixed-point calculation in (7) as illustrated in Ex. 9.

A ranking function obtained from a fixed-point computation as in (9) naturally gives rise to a positional winning strategy for the respective player in (normal) ω-regular games that allow for positional strategies. The corresponding positional strategy is obtained by always choosing a *minimum ranked successor* in the winning region.[5] We use this insight to obtain a *candidate* maximal strategy template for player Odd (which we prove to be also *winning* in Prop. 14) as follows. We start with a subgraph on $\mathcal{W}_{\mathsf{Odd}}$ defining the minimum ranked successor strategy for Odd induced by the ranking in (9), and then iteratively add all live edges of nodes that lie on a cycle in the subgraph, to the subgraph. The saturated subgraph then defines a strategy template for Odd, as formalized next.

▶ **Definition 12** (Rank-based Strategy Template). *Given an Odd-fair parity game $\mathcal{G}^\ell = (\langle V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi\rangle, E^\ell)$ with least even upper bound $l \geq 0$ on the priorities of nodes, winning region $\mathcal{W}_{\mathsf{Odd}} \subseteq V$ and the ranking function $\mathsf{rank} : \mathcal{W}_{\mathsf{Odd}} \to \mathbb{N}^l$ from Defn. 11, we define a strategy template $\mathcal{S}^{\mathcal{G}^\ell} = (\mathcal{W}_{\mathsf{Odd}}, E')$ where E' is constructed as follows:*

(S1) *for all $v \in V_{\mathsf{Even}} \cap \mathcal{W}_{\mathsf{Odd}}$, add all $(v,w) \in E$ to E';*

(S2) *for all $v \in V_{\mathsf{Odd}} \cap \mathcal{W}_{\mathsf{Odd}}$, add $(v,w) \in E$ to E' for a w with $w = \mathit{argmin}_{w' \in E(v)} \mathsf{rank}(w')$ (w is arbitrarily picked amongst the successors with the mimimum ranking);*

(S3) *for all $v \in V^\ell \cap \mathcal{W}_{\mathsf{Odd}}$, add all $(v,w) \in E^\ell$ to E' if v lays on a cycle in $\mathcal{S}^{\mathcal{G}^\ell}$;*

(S4) *repeat item (S3) until no new edges are added.*

We call $\mathcal{S}^{\mathcal{G}^\ell}$ the minimum rank based maximal Odd strategy template *of $\mathcal{G}^\ell$.*

▶ **Example 13.** $\mathcal{S}^{\mathcal{G}^\ell}$ for $\mathcal{G}^\ell$ from Ex. 9 is depicted in Fig. 2 (right).

It is clear from the definition that $\mathcal{S}^{\mathcal{G}^\ell}$ is an Odd strategy template in $\mathcal{G}^\ell$. It is also maximal since each $v \in \mathcal{W}_{\mathsf{Odd}}$ is assigned a rank. It remains to show that it is winning:

▶ **Proposition 14.** *Every player Odd strategy compliant with $\mathcal{S}^{\mathcal{G}^\ell}$ is winning for Odd in $\mathcal{G}^\ell$.*

The full proof of Prop. 14 can be found in App. A.1 and we only give a proof-sketch here.

First, recall that $\mathcal{S}^{\mathcal{G}^\ell}$ is obtained by extending a minimum-rank based strategy as formalized in Def. 12. Based on this we call a play $v_1 v_2 \ldots$ in $\mathcal{S}^{\mathcal{G}^\ell}$ *minimal* if for all $v_i \in V_{\mathsf{Odd}}$, v_{i+1} is the minimum ranked successor of v_i. We further call a cycle minimal, if it is a section of

[5] See [5] for a similar construction of the positional winning strategy of Even in Odd-fair parity games

a minimal play. Now consider a play $\pi = v_0 v_1 \dots$ which is compliant with $\mathcal{S}^{\mathcal{G}^\ell}$ and $v_0 \in \mathcal{W}_{\mathsf{Odd}}$. Since π is compliant with an Odd strategy template, it obeys the fairness condition. It is left to show that π is Odd winning. We do this by a chain of three observations,

1. If $\mathcal{W}_{\mathsf{Odd}} \neq \emptyset$, there exists a non empty set $M := \{v \in \mathcal{W}_{\mathsf{Odd}} \mid \mathsf{rank}(v) = (1, 0, 1, 0, \dots, 1, 0)\}$ (see Prop. 21).
2. All cycles in $\mathcal{S}^{\mathcal{G}^\ell}$ that pass through a vertex in M are Odd winning (see Prop. 22).
3. All infinite minimal plays in $\mathcal{S}^{\mathcal{G}^\ell}$ visit M infinitely often (see Prop. 25).

While item 1 simply follows from the observation that $(1, 0, 1, 0, \dots, 1, 0)$ is the minimum rank the ranking function assigns to a vertex and the set of nodes with this rank cannot be empty due to the monotonicity of (7), the proofs for item 2 and 3 are rather technical.

With the observations in item 1-3 being proven, we are ready to show that π is Odd winning. Observe that $\pi = v_1 v_2 \dots$ "embeds" an infinite minimal play, that is, there exists a subsequence $\pi' = v_{j_1} v_{j_2} \dots$ of π where $j_1 < j_2 < \dots$ that is a minimal play. This is because whenever a $v \in V_{\mathsf{Odd}} \cap \mathcal{W}_{\mathsf{Odd}}$ is seen infinitely often in π, $(v, v_{\min})$ is seen infinitely often as well, where $v_{\min}$ is the minimum-rank successor of v in $\mathcal{S}^{\mathcal{G}^\ell}$. Since π' visits M infinitely often (from item 3), π does so too. Then due to pigeonhole principle, there exists an $x \in M$ that is visited infinitely often by π. Thus, a tail of π can be seen as consecutive cycles over x. Since all cycles that pass through M are Odd winning (from item 2), we conclude that π is Odd winning.

Thm. 7 now follows as a corollary of Prop. 14.

5 Zielonka's Algorithm for Odd-Fair Parity Games

In this section, we construct a Zielonka-like algorithm that solves Odd-fair parity games. We call this algorithm *Odd-fair Zielonka's algorithm*. We first recall Zielonka's original algorithm in Sec. 5.1 and outline the changes imposed for our new Odd-fair version in Sec. 5.2. We then discuss the correctness of this new algorithm in Sec. 5.4.

From now on we take $\mathcal{G}^\ell = \langle (V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi), E^\ell \rangle$ to be an Odd-fair parity game.

5.1 Zielonka's Original Algorithm

Intuitively, Zielonka's algorithm consists of two nested recursive functions, $\mathsf{SOLVE}_{\mathsf{Even}}(n, \mathcal{G})$ and $\mathsf{SOLVE}_{\mathsf{Odd}}(n, \mathcal{G})$ which compute $\mathcal{W}_{\mathsf{Even}}$ and $\mathcal{W}_{\mathsf{Odd}}$ in a given parity game $\mathcal{G}$ with, respectively, even or odd upper bound priority n. Both functions recursively call each other on a sequence of sub-games that is constructed during the run of the algorithm.

The main difference between Zielonka's original algorithm [46] and our new Odd-fair version in Alg. 1 is the computation of the safe reachability set, denoted by $\mathsf{SafeReach}^f_\Lambda$ within the algorithms. Intuitively, the safe reachability set of player Λ is the set of vertices from which Λ has a strategy to force the game into the reach set $R \subseteq V$, while staying in the safety set $S \subseteq V$. In a (normal) parity game $\mathcal{G}$ (without live edges), this set can be computed via the single-nested fixed-point formula

$$\mathcal{X}_\Lambda := \mu X \,.\, (S \cap (R \cup \mathsf{Cpre}_\Lambda(X))). \tag{10}$$

If one interpretes Alg. 1 over (normal) parity games $\mathcal{G}$, defines $\mathsf{SafeReach}^f_\Lambda$ via (10) for the respective player, and replaces $\mathsf{SafeReach}^f_{\mathsf{Odd}}(\cdot, X, \cdot)$ in the last return statement with X (so, the algorithm returns X for any Λ), one gets exactly Zielonka's algorithm for parity games.

Algorithm 1 Odd-Fair Zielonka's Algo.

procedure $\mathsf{SOLVE}_\Lambda(n, \mathcal{G}^\ell)$
  $X \leftarrow V$
  $Z_{\neg\Lambda} \leftarrow G$
  while $Z_{\neg\Lambda} \neq \emptyset$ **do**
    $N \leftarrow \{v \mid v \in X \text{ with } \chi(v) = n\}$
    $Z \leftarrow X \setminus \mathsf{SafeReach}^f_\Lambda(X, N, \mathcal{G}^\ell)$
    $Z_{\neg\Lambda} \leftarrow \mathsf{SOLVE}_{\neg\Lambda}(n-1, \mathcal{G}^\ell[Z])$
    $X \leftarrow X \setminus \mathsf{SafeReach}^f_{\neg\Lambda}(X, Z_{\neg\Lambda}, \mathcal{G}^\ell)$
  end while
  if $\Lambda = \mathsf{Even}$ **then return** X
  else return $\mathsf{SafeReach}^f_{\mathsf{Odd}}(V, X, \mathcal{G}^\ell)$
  end if
end procedure

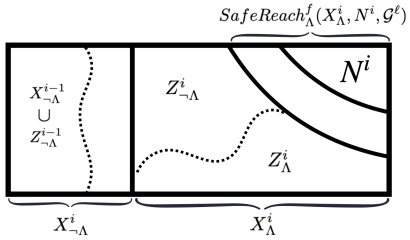

Figure 3 Visualization of the sets in Alg. 1.

5.2 The Odd-fair Zielonka's Algorithm

We are now considering an Odd-fair parity game $\mathcal{G}^\ell$. As discussed before, the main difference of the Odd-fair Zielonka's algorithm from the original one lies in the construction of the safe reachability sets denoted by $\mathsf{SafeReach}^f_\Lambda$ in Alg. 1. We therefore start by discussing its computation for both players.

The Odd Player. The first, somehow surprising, observation is that for player Odd in Odd-fair parity game $\mathcal{G}^\ell$, the safe reachability set $\mathcal{X}_{\mathsf{Odd}}$ can still be computed via (10). This is due to the fact that R only needs to be visited once, and Even vertices do not have live outgoing edges that might prevent player Odd from forcing a visit to R.

In addition, we can extract a *partial strategy template* for player Odd from the iterative computation of (10) via a similar, but much simpler ranking argument as used in Sec. 4. Here, $\mathsf{rank}(v) = 1$ for $v \in R$ and for the remaining vertices, $\mathsf{rank}(v)$ is the minimum integer j for which $v \in X^j := (S \cap (R \cup \mathsf{Cpre}_{\mathsf{Odd}}(X^{j-1})))$ where $X^0 = \emptyset$. The positional strategy of Λ is then to take the minimum ranked successor from each Odd node.

Another way to think about this strategy is in the form of an acyclic subgraph of $\mathcal{G}^\ell$ on $\mathcal{X}_{\mathsf{Odd}}$, where nodes in R have no outgoing edges, and for the remaining nodes, Odd nodes have one outgoing edge and Even nodes have all their outgoing edges. This is because if $v \in X^j \cap V_{\mathsf{Even}}$, all outgoing edges achieve positive progress towards R, i.e. for all $(v, w) \in E$, $w \in X^{j-1}$. Now it is easy to see that this subgraph almost defines a strategy template, i.e., on $\mathcal{X}_{\mathsf{Odd}} \setminus R$, Even nodes have all their outgoing edges in the subgraph, no Odd node lies on a cycle and all of them have one outgoing edge. However, vertices in R are dead-ends. We therefore call the strategy template induced by (10) *partial* and denote it by sr.

The Even Player. It follows from the results of Banerjee et. al. [5] that the safe reachability set $\mathcal{X}_{\mathsf{Even}}$ of player Even in Odd-fair parity games requires the 2-nested fixed-point formula $\nu Y.\mu X.S \cap (R \cup \mathsf{Apre}(Y, X))$, which (via the operators defined in Sec. 4.1) equals

$$\mathcal{X}_{\mathsf{Even}} := \nu Y \,.\, \mu X \,.\, S \cap (R \cup (\mathsf{Cpre}_{\mathsf{Even}}(X) \cup (\mathsf{Lpre}^{\exists}(X) \cap \mathsf{Pre}^{\forall}_{\mathsf{Odd}}(Y)))) \tag{11}$$

Intuitively, the necessity of a 2-nested formula arises from the following lack of information: we do not know in advance, which Odd nodes need to lie on a cycle on a strategy template required for Odd to win. If any positional strategy that lets Odd win (i.e., to avoid R or leave S) from a $v \in V^\ell$ requires v to lie on a cycle, then Odd has to take v's live outgoing edges as well, and thus, it can enter $\mathcal{X}_{\mathsf{Even}}$ and lose. The calculation of (11) starts with $Y^0 := V$, resulting in $\mathsf{Pre}^{\forall}_{\mathsf{Odd}}(V) = V$, hence

$$Y^1 := \mu X \ . \ S \cap (R \cup \mathsf{Cpre}_{\mathsf{Even}}(X) \cup \mathsf{Lpre}^{\exists}(X)). \tag{12}$$

Due to the disappearence of $\mathsf{Pre}^{\forall}_{\mathsf{Odd}}(Y)$ in this iteration, intuitively all $v \in V^\ell$ are treated as if they do not have any positional winning Odd strategy on them, so as if all Odd strategies have to take all the live edges in the game. Y^1 includes any Odd vertex that progresses towards R while staying in S with using either all its edges (due to $\mathsf{Cpre}_{\mathsf{Even}}(X)$) or through one live edge (due to $\mathsf{Lpre}^{\exists}(X)$). Thus, any vertex that manages to stay in $V \setminus Y^1$ does so due to being won by Odd even if Even could force all the live outgoing edges to be taken. Note that due to the monotonicity of fixed-point operators, for all j, $V \setminus Y^1 \subseteq V \setminus Y^j$.

Throughout the calculation, $V \setminus Y^j$ keeps track of the nodes that have managed to escape S or avoid R in the previous iteration, so are "already" won by Odd in the first j iterations. The inner fixed-point calculation in the $(j+1)^{th}$ iteration treats $V \setminus Y^j$ as a subset of Odd's winning region and it deems any node that can be forced by Odd to reach $V \setminus Y^j$, lost by Even. When the algorithm saturates, Y^∞ contains only those Odd nodes that cannot be forced by Odd to reach $V \setminus Y^\infty$, i.e., are won by Even. Here it is important to observe that, $V \setminus Y^\infty$ contains some Odd nodes that are not $V \setminus Y^1$. Since they are in Y^1, these nodes inductively reach Even winning vertices through live edges. This reveals that, all nodes in $V \setminus Y^j$ but not in $V \setminus Y^1$ win due to a positional Odd strategy that reaches $V \setminus Y^{j-1}$. Iteratively, this reveals that all such nodes have positional Odd strategies that make them reach $V \setminus Y^1$.

The above alternative interpretation of the computation of $\mathcal{X}_{\mathsf{Even}}$ in (11) is the key insight that we utilize to define our new Odd-fair Zielonka's algorithm, as discussed next.

The Odd-fair Zielonka's Algorithm. Following up on the previous discussion, we use the following insight within the construction of the Odd-fair Zielonka's algorithm. We assume the existence of a core subset $\mathcal{W}'_{\mathsf{Odd}} \subseteq \mathcal{W}_{\mathsf{Odd}}$ that player Odd can force all nodes in $\mathcal{W}_{\mathsf{Odd}}$ to, that is winning for Odd even under the assumption that Even can force all the live edges in the game to be taken. Since Zielonka's algorithm solves parity games by a sequence of nested safe-reachability calculations for alternating players, we apply the following trick: Instead of computing $\mathcal{X}_{\mathsf{Even}}$ via (11) in each recursive call of Alg. 1, we only compute Y^1 via (12) and use it as an *overapproximation* of $\mathcal{X}_{\mathsf{Even}}$ (which is indeed the case due to the monotonicity of (11) in Y). That is, while we take the Odd safe reachability set $\mathsf{SafeReach}^f_{\mathsf{Odd}}$ as the original (linear) Odd safe reachability computation known for these games (given in (10)), we do not take Even safe reachability formula $\mathsf{SafeReach}^f_{\mathsf{Even}}$ to be the (quadratic) Even safe reachability computation known for these games (given in (11)), but we instead take it as its (linear) subformula given in (12) and arrive at an overapproximation of the Even safe reachability region at the end of each $\mathsf{SafeReach}^f_{\mathsf{Even}}$ calculation. We finalize the recursive call $\mathsf{SOLVE}_{\mathsf{Odd}}$ by an extra call of $\mathsf{SafeReach}^f_{\mathsf{Odd}}$ applied to the (thus) underapproximated Odd winning region in the sub-game, therefore expanding the returned Odd winning region of the sub-game.

By this, it turns out that the recursive call of $\mathsf{SOLVE}_{\mathsf{Odd}}(n, \mathcal{G}^\ell)$ actually computes $\mathcal{W}'_{\mathsf{Odd}}$ as the set X and we ensure that $\mathcal{W}_{\mathsf{Odd}}$ is returned by the additional (linear) computation of $\mathsf{SafeReach}^f_{\mathsf{Odd}}$ over X in the last return statement of Alg. 1. This instantiation of the safe-reachability computations is formalized next.

▶ **Definition 15.** *Given an Odd-fair parity game* $\mathcal{G}^\ell = \langle (V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi), E^\ell \rangle$ *the safe-reachability procedures* $\mathsf{SafeReach}^f_{\mathsf{Odd}}(S, R, \mathcal{G}^\ell)$ *and* $\mathsf{SafeReach}^f_{\mathsf{Even}}(S, R, \mathcal{G}^\ell)$ *in Alg. 1 denote the iterative fixed-point computations in* (10) *for* Odd *and* (12) *for* Even.

5.3 Complexity of the Odd-fair Zielonka's Algorithm

The safe-reachability computations defined in Def. 15 have the same complexity as their computations via (10) in Zielonka's original algorithm. The only difference is in the number of calculated Pre operations: while $\mathsf{SafeReach}_{\mathsf{Even}}$ from Zielonka's original algorithm (10) require the calculation of only one Pre operator, $\mathsf{SafeReach}^{f}_{\mathsf{Even}}$ from (12) requires the calculation of 2 Pre operators. The additional final call of $\mathsf{SafeReach}^{f}_{\mathsf{Odd}}$ in $\mathsf{SOLVE}_{\mathsf{Odd}}$ procedure also has linear complexity and requires one Pre calculation. Therefore, not only the worst-case time complexity of Alg. 1 is equivalent to that of Zielonka's original algorithm (which would be the case even if we used the quadratic safe reachability formula from (11) for Even since the overall complexity of the algorithm is exponential) but we create almost no additional computational overhead in the algorithm by introducing the fairness assumptions.

We further remark that Alg. 1 is not a straight-forward interpretation of the nested fixed-point in (7), and its negation (see (14) in App. A.1 of [37]) in the form of Zielonka's algorithm. Firstly, such a straightforward approach is non-trivial due to Apre and Npre operators taking two variables from two different iterations of the fixed-point calculation. Furthermore, at each Even safe-reachability call of Alg. 1, as mentioned we compute 2 Pre operators (equation 12), whereas in each such corresponding step in the fixed-point iteration, we would have to compute 3 Pre operators due to the expansion of Apre (5b) and Npre (5c).

It remains to show that Odd-fair Zielonka's algorithm solves Odd-fair parity games.

5.4 Correctness of the Odd-fair Zielonka's Algorithm

We first recall that Odd-fair parity games are determined. Next, we prove the correctness of the algorithm by induction on n. Since in the base case $n = 0$ the calls correctly return $\emptyset$, it suffices to prove the correctness of each function, assuming the correctness of the other. This is formalized next.

▶ **Theorem 16** (Correctness of SOLVE_Λ, Alg. 1)**.** *Assume that for any Odd-fair parity game $\mathcal{G}'^{\ell}$ where $n' < n$ is an odd (resp. even) upper bound on the priorities of the game, $SOLVE_{Odd}(n', \mathcal{G}')^{\ell}$ correctly returns the Odd winning region (resp. $\mathsf{SOLVE}_{Even}(n', \mathcal{G}')^{\ell}$ correctly returns the Even winning region) in $\mathcal{G}'^{\ell}$. Then $SOLVE_{\Lambda}(n, \mathcal{G}^{\ell})$ correctly returns the winning region of player Λ where n is even if $\Lambda =$ Even and odd if $\Lambda =$ Odd.*

Notation. We follow the notation of Küsters' proof [28] of Zielonka's original algorithm [46]. Recall that $\mathcal{G}^{\ell}$ has no dead-ends. For some $X \subseteq V$, we call $\mathcal{G}^{\ell}[X] = \langle (X, X \cap V_{\mathsf{Even}}, X \cap V_{\mathsf{Odd}}, X \times X \subseteq E, \chi\,|_X), X \times X \subseteq E^{\ell} \rangle$ a *subgame* of $\mathcal{G}^{\ell}$ if it has no dead-ends. Here, $\chi\,|_X$ is the priority function $\chi : V \to \mathbb{N}$ restricted to domain X. Let n be an upper bound on the priorities in V. If the parity of n is even, set Λ to Even; if it's odd, set Λ to Odd.

Λ-trap and Λ-paradise. A Λ-trap is a subset $T \subseteq V$ for $\Lambda \in \{\mathsf{Even}, \mathsf{Odd}\}$ such that, $\forall v \in T \cap V_{\neg\Lambda}$, $\exists (v, w) \in E$ with $w \in T$ and $\forall v \in T \cap V_{\Lambda}$, $(v, w) \in E \implies w \in T$. A Λ-paradise in $\mathcal{G}^{\ell}$ is a subset $T \subseteq V$ which is a $\neg\Lambda$-trap in V and there exists a winning Λ strategy template (T, E') in $\mathcal{G}^{\ell}$.

The recursive calls of SOLVE_Λ and $\mathsf{SOLVE}_{\neg\Lambda}$ on subgames within Alg. 1 induce a characteristic partition of the game graph. For the correctness proof, we need to remember a series of these subgames that are constructed through previous recursive calls. The partition of these subsets is illustrated in Fig. 3 and formalized as follows.

$$X^i_\Lambda := V \setminus X^i_{\neg\Lambda} \qquad N^i := \{v \in X^i_\Lambda \mid \chi(v) = n\} \tag{13}$$
$$Z^i := X^i_\Lambda \setminus \mathsf{SafeReach}^f_\Lambda(X^i_\Lambda, N^i, \mathcal{G}^\ell) \qquad X^{i+1}_{\neg\Lambda} := \mathsf{SafeReach}^f_{\neg\Lambda}(V, X^i_{\neg\Lambda} \cup Z^i_{\neg\Lambda}, \mathcal{G}^\ell)$$

here, in addition Z^i_Λ is the Λ winning region in the subgame $\mathcal{G}^\ell[Z^i]$. Intuitively, the sets constructed in (13) correspond to the sets with the same name within Alg. 1.

We collect the following observations on these sets, which are proven in [37].

1. **([37] – Obs. 37)** $X^i_{\neg\Lambda}$ is an Λ-trap, X^i_Λ, Z^i and Z^i_Λ are $\neg\Lambda$-traps in V. Z^i is in $\neg\Lambda$-trap in X_Λ and $Z^i_{\neg\Lambda}, Z^i_\Lambda$ are Λ- and $\neg\Lambda$-traps in Z^i, respectively. Therefore, $\mathcal{G}^\ell[Y]$ is a subgame of $\mathcal{G}^\ell$ with Y being any of these sets.
2. **([37] – Lem. 38)** $X^i_{\neg\Lambda} \cup \mathsf{SafeReach}^f_{\neg\Lambda}(X^i_\Lambda, Z^i_{\neg\Lambda}, \mathcal{G}^\ell) = \mathsf{SafeReach}^f_{\neg\Lambda}(V, X^i_{\neg\Lambda} \cup Z^i_{\neg\Lambda}, \mathcal{G}^\ell)$.
3. **([37] – Cor. 39)** As a consequence of the previous item, $\{X^i_{\neg\Lambda}\}_{i\in\mathbb{N}}$ is an increasing sequence. Consequently, $\{X^i_\Lambda\}_{i\in\mathbb{N}}$ is a decreasing sequence. As V is finite, this immediately implies that these sequences reach a saturation value for some, and in fact the same, k.
4. **([37] – Lem. 34)** If $R \subseteq V$ is an Odd-paradise in $\mathcal{G}^\ell$, then $\mathsf{SafeReach}^f_{\mathsf{Odd}}(V, R, \mathcal{G}^\ell)$ is also an Odd-paradise in $\mathcal{G}^\ell$.
5. **([37] – Lem. 31)** The set $U \setminus \mathsf{SafeReach}_\Lambda(U, R, \mathcal{G}^\ell)$ is a Λ-trap in U.

In contrast to Zielonka's original algorithm, the proof of the procedures $\mathsf{SOLVE}_{\mathsf{Even}}$ and $\mathsf{SOLVE}_{\mathsf{Odd}}$ is not identical in Odd-fair Zielonka's algorithm. This is due to the different safe-reachability set constructions used. Next we sketch the correctness proof of Thm. 16 for $\Lambda := \mathsf{Odd}$, corresponding to the correctness of procedure $\mathsf{SOLVE}_{\mathsf{Odd}}$. The proof for $\Lambda := \mathsf{Even}$ is left to the appendix, as it resembles the proof Zielonka's original algorithm more.

▶ **Proposition 17.** *Given the premisses of Thm. 16 for $\Lambda = \mathsf{Odd}$, if $Z^k_{\mathsf{Even}} = \emptyset$ then $\mathsf{SafeReach}^f_{\mathsf{Odd}}(V, X^k_{\mathsf{Odd}}, \mathcal{G}^\ell)$ is an Odd-paradise and $V \setminus \mathsf{SafeReach}^f_{\mathsf{Odd}}(V, X^k_{\mathsf{Odd}}, \mathcal{G}^\ell)$ is an Even-paradise in $\mathcal{G}^\ell$.*

Within Prop. 17, the fact that $Z^k_{\mathsf{Even}} = \emptyset$ refers to the termination of the recursive call in Alg. 1 which results in the saturation of the sequence $\{X^i_{\mathsf{Odd}}\}_{i\in\mathbb{N}}$ with X^k_{Odd}. This implies that $\mathsf{SOLVE}_{\mathsf{Odd}}$ returns $T := \mathsf{SafeReach}^f_{\mathsf{Odd}}(V, X^k_{\mathsf{Odd}}, \mathcal{G}^\ell)$, which is an Odd-paradise and $V \setminus T$ an Even-paradise. With this, Thm. 16 follows from Prop. 17 for $\Lambda = \mathsf{Odd}$. We now give a proof sketch of Prop. 17.

We first recall from observation 1 that T and $V \setminus T$ are Even- and Odd-traps in V, respectively. In order to prove Prop. 17, it remains to show that there exists an Odd (resp. Even) strategy template which is winning in $\mathcal{G}^\ell$ and maximal on T (resp. $V \setminus T$). We next give the construction of these templates and a high-level intuition on why they are actually *winning*.

Winning Odd Strategy Templates. As X^k_{Odd} is known to be an Even-trap, it can be proven to be an Odd-paradise by constructing a winning maximal strategy template on it. It then follows from observation 4 that T is also an Odd-paradise.

Towards a construction of a maximal winning Odd strategy template on X_{Odd}, we first observe that $X^k_{\mathsf{Odd}} = Z^k_{\mathsf{Odd}} \cup \mathsf{SafeReach}^f_{\mathsf{Odd}}(X^k_{\mathsf{Odd}}, N^k, \mathcal{G}^\ell)$ (as $Z^k_{\mathsf{Even}} = \emptyset$). Then there exists a maximal winning Odd strategy template z on $Z^k = Z^k_{\mathsf{Odd}}$ in game $\mathcal{G}^\ell[Z^k]$. Any play π compliant with z that starts and stays in Z^k is clearly Odd winning. However, z is not necessarily an Odd strategy template in $\mathcal{G}^\ell$ since there are possibly some $(v, w) \in E$ with $v \in Z^k \cap V_{\mathsf{Even}}$ and $w \notin Z^k$. For all such edges, $w \in \mathsf{SafeReach}^f_{\mathsf{Odd}}(X^k_{\mathsf{Odd}}, N^k, \mathcal{G}^\ell)$ since X^k_{Odd} is an Even-trap in V. For the state set $\mathcal{X}_{\mathsf{Odd}} := \mathsf{SafeReach}^f_{\mathsf{Odd}}(X^k_{\mathsf{Odd}}, N^k, \mathcal{G}^\ell)$, recall from Sec. 5.2 that there exists partial strategy template sr defined on $\mathcal{X}_{\mathsf{Odd}}$ with dead ends in N^k.

Using the templates z and sr, we can construct a maximal candidate Odd strategy template on X^k_{Odd}. Following the intuition behind the construction of $\mathcal{S}^{\mathcal{G}^\ell}$ in Def. 12, we first define a base subgraph (X^k_{Odd}, E') with $E' \subseteq E$ s.t. $(v,w) \in E$ is in E' if either (i) $(v,w) \in z \cup sr$, (ii) $v \in V_{\mathsf{Even}} \cap X^k_{\mathsf{Odd}}$, or (iii) $v \in N^k \cap V_{\mathsf{Odd}}$ and $w = v_r$ where v_r is a random fixed successor of v, that is in X^k_{Odd}. Such a successor is guaranteed to exist since X^k_{Odd} is an Even-trap. We now extend the subgraph (X^k_{Odd}, E') to an Odd strategy template by adding all live edges originating in vertices $X^k_{\mathsf{Odd}} \cap V^\ell$ that lie on a cycle in E', similar to Def. 12 (S3)-(S4). This results in a subgraph $\mathcal{S} = (X^k_{\mathsf{Odd}}, \overline{E'})$ that is a maximal Odd strategy template. The underlying idea behind $\mathcal{S}$ being winning is the following: Any play that starts in X^k_{Odd} either stays in Z^k after some point and is won by $\mathcal{S}$ collapsing to z, or sees a newly added cycle (one that is not in $z \cup sr$) infinitely often. All such cycles contain a newly added edge. An analysis of newly added edges reveal that, all of them – when seen infinitely often – eventually drag a play towards N^i. Thus, every play that sees a new cycle infinitely often sees n infinitely often, and thus won by Odd.

Winning Even Strategy Templates. Here we show that $V \setminus T$ is an Even-paradise in $\mathcal{G}^\ell$. We first define $\mathcal{X}^i_{\mathsf{Even}} := \mathsf{SafeReach}^f_{\mathsf{Even}}(X^i_{\mathsf{Odd}}, Z^i_{\mathsf{Even}}, \mathcal{G}^\ell)$ and denote by sr^i the partial Even strategy template defined on $\mathcal{X}^i_{\mathsf{Even}}$. We further denote the winning Even strategy on Z^i_{Even} in game $\mathcal{G}^\ell[Z^i]$ by z^i. We can now construct the Even strategy template $\mathcal{S} = (V \setminus T, E')$ where E' is the combination of edges in $sr^i \cup z^i$ with $\{(v,w) \in E \mid v \in V_{\mathsf{Odd}} \cap (V \setminus T)\}$. Since $V \setminus T$ is an Odd-trap by observation 5, the edge set E' stays within $V \setminus T$, i.e. $E' \subseteq V \setminus T \times V \setminus T$. Then clearly, $\mathcal{S}$ is an Even strategy template. To see $\mathcal{S}$ is winning we first observe that each $v \in V \setminus T$ there exists a unique $i < k$ such that $v \in \mathcal{X}^i_{\mathsf{Even}}$. Let $\pi = v_1 v_2 \dots$ be a play compliant with $\mathcal{S}$ and let $s = \mathcal{X}_1 \mathcal{X}_2 \dots$ be the sequence such that $v_i \in \mathcal{X}$. (1) If $v_j \in Z^i_{\mathsf{Even}}$, $v_{j+1} \in Z^i_{\mathsf{Even}} \cup \{\mathcal{X}^r_{\mathsf{Even}} \mid r < i\}$. This follows from Z^i_{Even} being an Odd-trap in X^i_{Odd}. (2) If π visits $v \in \mathcal{X}^i$ infinitely often, π visits Z^i_{Even} infinitely often: This is because π visits the (v,w) in $\mathcal{S}$ that makes positive progress towards Z^i_{Even} infinitely often as well. Let i be the minimum index such that $\mathcal{X}^i_{\mathsf{Even}}$ is seen infinitely often in s. By (1), π visits Z^i_{Even} infinitely often and by (1) and the minimality of i, it should eventually stay in Z^i_{Even}. Thus $\mathcal{S}$ eventually collapses to z^i_{Even} on π and the play is won by Even.

5.5 Experimental Results

We conducted an experimental study to empirically validate the claim that our new Odd-fair Zielonka's algorithm retains its efficiency in practice (see App. A.2 for details).

We generated Odd-fair parity instances manipulating 286 benchmark instances of the dataset PGAME_Synth_2021 of the SYNTCOMP benchmark suite [18] and 51 instances of PGSolver dataset of Keiren's benchmark suite [21] by adding live edges to the given (normal) parity games. We empirically compared the (non-optimized[6]) C++-based implementations of (i) the Odd-fair Zielonka's algorithm (`OF-ZL`) from Alg. 1, (ii) the "normal" Zielonka's algorithm (`N-ZL`) from [46], (iii) the fixed-point algorithm for Odd-fair parity games (`OF-FP`) from [5] implementing (7), and (iv) the "normal" fixed-point algorithm (`N-FP`) for "normal" parity games from [13]. On the *SYNTCOMP benchmarks*, the time-out rates are: 82 instances for `OF-FP`, 58 for `OF-ZL`; 73 for `N-FP` and 47 for `N-ZL`. On the 204 instances that neither of the algorithms time out the average computation times are: 122.7 seconds for `OF-FP`, 4.6

[6] While optimized version of `N-ZL` and `N-FP` are available in `oink` [44] our goal is a conceptual comparison, which is better achieved by similar (non-optimized) implementations for all algorithms.

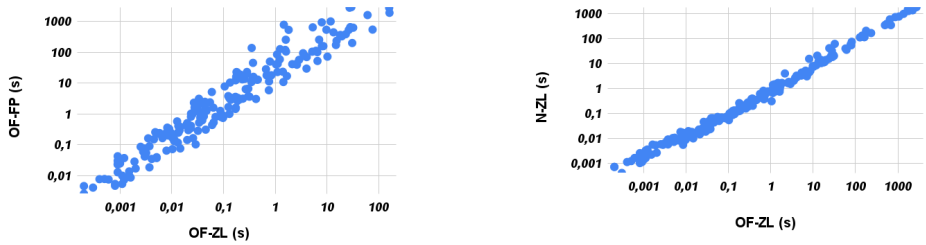

Figure 4 Scatter plot for a comparative evaluation of `OF-ZL` vs. `OF-FP` (left), and `OF-ZL` vs. `N-ZL` (right). Both plots show computation times in seconds using logarithmic scaling.

seconds for `OF-ZL`, 45.2 seconds for `N-FP` and 3.6 seconds for `N-ZL`. For all instances that did not time out for all four algorithms, Fig. 4 shows scatter plots comparing the computation times of `OF-ZL` with `OF-FP` (left) and `OF-ZL` with `N-ZL` (right) using logarithmic scaling. The diagonal shows instances with similar computation times. Points above the diagonal show superior performance of `OF-ZL`. For the *PGSolver dataset* `OF-FP` timed out on all generated instances, whereas `OF-ZL` took 24.9 seconds on average to terminate.

We clearly see that `OF-ZL` performs up to one order of magnitude better than `OF-FP` in many instances while `OF-ZL` and `N-ZL` perform very similar on the given benchmark instances. In addition, we observe that `OF-FP` starts timing out as soon as the examples became more complex. These outcomes match the known comparison results between the naive fixed-point calculation versus Zielonka's algorithm, on normal parity games.

References

1 Rajeev Alur, Salar Moarref, and Ufuk Topcu. Counter-strategy guided refinement of GR(1) temporal logic specifications. In *Formal Methods in Computer-Aided Design, FMCAD 2013, Portland, OR, USA, October 20-23, 2013*, pages 26–33. IEEE, 2013.

2 Benjamin Aminof, Giuseppe De Giacomo, and Sasha Rubin. Stochastic fairness and language-theoretic fairness in planning in nondeterministic domains. In J. Christopher Beck, Olivier Buffet, Jörg Hoffmann, Erez Karpas, and Shirin Sohrabi, editors, *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020*, pages 20–28. AAAI Press, 2020.

3 André Arnold, Damian Niwiński, and Paweł Parys. A quasi-polynomial black-box algorithm for fixed point evaluation. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic, CSL 2021, January 25-28, 2021, Ljubljana, Slovenia (Virtual Conference)*, volume 183 of *LIPIcs*, pages 9:1–9:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

4 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

5 Tamajit Banerjee, Rupak Majumdar, Kaushik Mallik, Anne-Kathrin Schmuck, and Sadegh Soudjani. Fast symbolic algorithms for omega-regular games under strong transition fairness. *TheoretiCS*, 2, 2023.

6 Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. *Formal methods for discrete-time dynamical systems*, volume 15. Springer, 2017.

7 Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.*, 78(3):911–938, 2012.

8 Krishnendu Chatterjee, Luca de Alfaro, Marco Faella, Rupak Majumdar, and Vishwanath Raman. Code aware resource management. *Formal Methods Syst. Des.*, 42(2):146–174, 2013.

9 Alessandro Cimatti, Marco Pistore, Marco Roveri, and Paolo Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artif. Intell.*, 147(1-2):35–84, 2003.

10 Marco Daniele, Paolo Traverso, and Moshe Y. Vardi. Strong cyclic planning revisited. In Susanne Biundo and Maria Fox, editors, *Recent Advances in AI Planning, 5th European Conference on Planning, ECP'99, Durham, UK, September 8-10, 1999, Proceedings*, volume 1809 of *Lecture Notes in Computer Science*, pages 35–48. Springer, 1999.

11 Nicolás D'Ippolito, Natalia Rodríguez, and Sebastian Sardiña. Fully observable non-deterministic planning as assumption-based reactive synthesis. *J. Artif. Intell. Res.*, 61:593–621, 2018.

12 E. Allen Emerson and Charanjit S. Jutla. On simultaneously determinizing and complementing omega-automata (extended abstract). In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89), Pacific Grove, California, USA, June 5-8, 1989*, pages 333–342. IEEE Computer Society, 1989.

13 E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 368–377. IEEE Computer Society, 1991.

14 E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs. *SIAM J. Comput.*, 29(1):132–158, 1999.

15 Nissim Francez. *Fairness*. Springer-Verlag, Berlin, Heidelberg, 1986.

16 Yuri Gurevich and Leo Harrington. Trees, automata, and games. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 60–65. ACM, 1982.

17 Daniel Hausmann and Lutz Schröder. Quasipolynomial computation of nested fixpoints. In Jan Friso Groote and Kim Guldstrand Larsen, editors, *Tools and Algorithms for the Construction and Analysis of Systems – 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 – April 1, 2021, Proceedings, Part I*, volume 12651 of *Lecture Notes in Computer Science*, pages 38–56. Springer, 2021.

18 Swen Jacobs, Guillermo A. Pérez, Remco Abraham, Véronique Bruyère, Michaël Cadilhac, Maximilien Colange, Charly Delfosse, Tom van Dijk, Alexandre Duret-Lutz, Peter Faymonville, Bernd Finkbeiner, Ayrat Khalimov, Felix Klein, Michael Luttenberger, Klara J. Meyer, Thibaud Michaud, Adrien Pommellet, Florian Renkin, Philipp Schlehuber-Caissier, Mouhammad Sakr, Salomon Sickert, Gaëtan Staquet, Clement Tamines, Leander Tentrup, and Adam Walker. The reactive synthesis competition (SYNTCOMP): 2018-2021. *CoRR*, abs/2206.00251, 2022. doi:10.48550/arXiv.2206.00251.

19 Marcin Jurdzinski. Small progress measures for solving parity games. In Horst Reichel and Sophie Tison, editors, *STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Lille, France, February 2000, Proceedings*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer, 2000.

20 Marcin Jurdzinski, Rémi Morvan, and K. S. Thejaswini. Universal algorithms for parity games and nested fixpoints. In Jean-François Raskin, Krishnendu Chatterjee, Laurent Doyen, and Rupak Majumdar, editors, *Principles of Systems Design – Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday*, volume 13660 of *Lecture Notes in Computer Science*, pages 252–271. Springer, 2022.

21 Jeroen J. A. Keiren. Benchmarks for parity games. In Mehdi Dastani and Marjan Sirjani, editors, *Fundamentals of Software Engineering – 6th International Conference, FSEN 2015 Tehran, Iran, April 22-24, 2015, Revised Selected Papers*, volume 9392 of *Lecture Notes in Computer Science*, pages 127–142. Springer, 2015.

22 Nils Klarlund. *Progress Measures and Finite Arguments for Infinite Computations*. PhD thesis, Cornell University, USA, 1990.

23 Nils Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Ann. Pure Appl. Log.*, 69(2-3):243–268, 1994.

24 Nils Klarlund and Dexter Kozen. Rabin measures and their applications to fairness and automata theory. In *Proceedings of the Sixth Annual Symposium on Logic in Computer Science (LICS '91), Amsterdam, The Netherlands, July 15-18, 1991*, pages 256–265. IEEE Computer Society, 1991.

25 Dexter Kozen. Results on the propositional mu-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

26 Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. Where's waldo? sensor-based temporal logic motion planning. In *2007 IEEE International Conference on Robotics and Automation, ICRA 2007, 10-14 April 2007, Roma, Italy*, pages 3116–3121. IEEE, 2007.

27 Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Trans. Robotics*, 25(6):1370–1381, 2009.

28 Ralf Küsters. Memoryless determinacy of parity games. In Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors, *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, volume 2500 of *Lecture Notes in Computer Science*, pages 95–106. Springer, 2001.

29 Rupak Majumdar, Kaushik Mallik, Anne-Kathrin Schmuck, and Sadegh Soudjani. Symbolic control for stochastic systems via parity games. *CoRR*, abs/2101.00834, 2021.

30 Shahar Maoz and Jan Oliver Ringert. Synthesizing a lego forklift controller in GR(1): A case study. In Pavol Cerný, Viktor Kuncak, and Parthasarathy Madhusudan, editors, *Proceedings Fourth Workshop on Synthesis, SYNT 2015, San Francisco, CA, USA, 18th July 2015*, volume 202 of *EPTCS*, pages 58–72, 2015.

31 Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.

32 Petter Nilsson, Necmiye Ozay, and Jun Liu. Augmented finite transition systems as abstractions for control synthesis. *Discret. Event Dyn. Syst.*, 27(2):301–340, 2017.

33 Paweł Parys. Parity games: Zielonka's algorithm in quasi-polynomial time. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 10:1–10:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

34 Marco Pistore and Paolo Traverso. Planning as model checking for extended goals in non-deterministic domains. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 479–486. Morgan Kaufmann, 2001.

35 Jean-Pierre Queille and Joseph Sifakis. Fairness and related properties in transition systems – A temporal logic to deal with fairness. *Acta Informatica*, 19:195–220, 1983.

36 Miquel Ramírez and Sebastian Sardiña. Directed fixed-point regression-based planning for non-deterministic domains. In Steve A. Chien, Minh Binh Do, Alan Fern, and Wheeler Ruml, editors, *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*. AAAI, 2014.

37 Irmak Sağlam and Anne-Kathrin Schmuck. Solving odd-fair parity games, 2023. `arXiv:2307.13396`.

38 Sven Schewe. An optimal strategy improvement algorithm for solving parity and payoff games. In Michael Kaminski and Simone Martini, editors, *Computer Science Logic, 22nd International Workshop, CSL 2008, 17th Annual Conference of the EACSL, Bertinoro, Italy, September 16-19, 2008. Proceedings*, volume 5213 of *Lecture Notes in Computer Science*, pages 369–384. Springer, 2008.

39 Robert S. Streett and E. Allen Emerson. The propositional mu-calculus is elementary. In Jan Paredaens, editor, *Automata, Languages and Programming, 11th Colloquium, Antwerp, Belgium, July 16-20, 1984, Proceedings*, volume 172 of *Lecture Notes in Computer Science*, pages 465–472. Springer, 1984.

40 María Svorenová, Jan Kretínský, Martin Chmelik, Krishnendu Chatterjee, Ivana Cerná, and Calin Belta. Temporal logic control for stochastic linear systems using abstraction refinement of probabilistic games. In Antoine Girard and Sriram Sankaranarayanan, editors, *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, HSCC'15, Seattle, WA, USA, April 14-16, 2015*, pages 259–268. ACM, 2015.

41 Paulo Tabuada. *Verification and Control of Hybrid Systems – A Symbolic Approach*. Springer, 2009.

42 John G Thistle and RP Malhamé. Control of ω-automata under state fairness assumptions. *Systems & control letters*, 33(4):265–274, 1998.

43 Tom van Dijk. Attracting tangles to solve parity games. In Hana Chockler and Georg Weissenbacher, editors, *Computer Aided Verification – 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part II*, volume 10982 of *Lecture Notes in Computer Science*, pages 198–215. Springer, 2018.

44 Tom van Dijk. Oink: An implementation and evaluation of modern parity game solvers. In Dirk Beyer and Marieke Huisman, editors, *Tools and Algorithms for the Construction and Analysis of Systems – 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part I*, volume 10805 of *Lecture Notes in Computer Science*, pages 291–308. Springer, 2018.

45 Kai Weng Wong, Rüdiger Ehlers, and Hadas Kress-Gazit. Resilient, provably-correct, and high-level robot behaviors. *IEEE Trans. Robotics*, 34(4):936–952, 2018.

46 Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998.

A Appendix

A.1 Proof of Prop. 14

We will restate the fixed-point formula that calculates the Odd winning region and the main proposition for the sake of self-containment.

▶ **Proposition 7.** *Given an Odd-fair parity game $\mathcal{G}^\ell = (\langle V, V_{\mathsf{Even}}, V_{\mathsf{Odd}}, E, \chi\rangle, E^\ell)$ with least even upper bound $l \geq 0$ it holds that $Z = \mathcal{W}_{\mathsf{Odd}}$, where*

$$Z := \mu Y_l.\ \nu X_{l-1}.\ \ldots \mu Y_2.\ \nu X_1.\ \bigcap_{j\in[\![2,l]\!]} \mathcal{B}_j[Y_j, X_{j-1}], \tag{14}$$
$$\text{where}\quad \mathcal{B}_j[\mathbf{Y},\mathbf{X}] := \left(\bigcup_{i\in[j+1,l]} C_i\right) \cup \left(\overline{C_j} \cap \mathsf{Npre}(\mathbf{Y},\mathbf{X})\right) \cup \left(C_j \cap \mathsf{Cpre}_{\mathsf{Odd}}(\mathbf{Y})\right).$$

then $Z = \mathcal{W}_{\mathsf{Odd}}$. Further, it takes $\mathcal{O}(n^{l+1})$ symbolic steps to compute Z.

▶ **Proposition 14.** *Every player Odd strategy compliant with $\mathcal{S}^{\mathcal{G}^\ell}$ is winning for Odd in $\mathcal{G}^\ell$.*

The main observation behind the proof of Prop. 14 is similar to the main observation in Sec. 5, leading to the proof of Alg. 1. That is, there exists a core subset of the Odd winning region $\mathcal{W}'_{\mathsf{Odd}} \subseteq \mathcal{W}_{\mathsf{Odd}}$, that is added to Z in the first iteration of the fixed-point calculation in (7), to which each $v \in \mathcal{W}_{\mathsf{Odd}}$ can be made to reach by Odd. Here in particular, we show that any Odd strategy compliant with $\mathcal{S}^{\mathcal{G}^\ell}$ reaches $\mathcal{W}'_{\mathsf{Odd}}$ (infinitely often) while obeying the fairness condition, and is thus winning for Odd.

The proof of Prop. 14 consists of 3 main propositions. Before we present them, we will gather some observations from the fixed-point formula (7) and present them as lemmas.

According to our previous definitions, $Y_m^{r_l,r_{l-1},\dots,r_m}$ denotes the value of Y_m variable after the r_m^{th} iteration on it, while Y_i, X_i variables for $i > m$ are in their $r_i + 1^{th}$ iterations. If we flatten this formula we get the following equality: $Y_m^{r_l,r_{l-1},\dots,r_m} =$

$$\nu X_{m-1} \dots \mu Y_2 \nu X_1. \bigcap_{j\in[\![m+2,l]\!]} \mathcal{B}_j[Y_j^{r_j}, X_{j-1}^{r_{j-1}}] \cap \mathcal{B}_m[Y_m^{r_m-1}, X_{m-1}] \cap \bigcap_{j\in[\![2,m-2]\!]} \mathcal{B}_j[Y_j, X_{j-1}]$$

Observe that when the fixed-point above is calculated, all X_j, Y_j values for $j < m$ will saturate at the same value, which is the final result of the computation. That is,

▶ **Lemma 18.**

$$Y_m^{r_l,\dots,r_m} = \bigcap_{j\in[\![m+2,l]\!]} \mathcal{B}_j[Y_j^{r_j}, X_{j-1}^{r_{j-1}}] \cap \mathcal{B}_m[Y_m^{r_m-1}, Y_m^{r_l,\dots,r_m}] \cap \bigcap_{j\in[\![2,m-2]\!]} \mathcal{B}_j[Y_m^{r_l,\dots,r_m}, Y_m^{r_l,\dots,r_m}]$$

▶ **Lemma 19.** *For all $v \in \mathcal{W}_{Odd}$ with $\mathsf{rank}(v) = (r_l, 0, \dots, r_2, 0)$. Then,*

$$v \in \bigcap_{j\in[\![2,l]\!]} Y_j^{r_l-1,0,r_{l-2}-1,0,\dots,r_{j-2}-1,0,r_j}$$

This is similar to our previous observation. $\mathsf{rank}(v) = (r_l, 0, \dots, r_2, 0)$ implies v was added to the formula while Y_j variable was on it's r_j^{th} iteration for all $j \in [\![2, l]\!]$. Since $X_{j-1}^0 = V$, the iteration values of X variables can be safely ignored.

▶ **Lemma 20.** *if $v \in V_{Even}$,* $\forall (v, w) \in E, \mathsf{rank}(v) \geq_{l+1-\chi(v)} \mathsf{rank}(w)$

if $v \in V_{Odd}$, $\exists (v, w) \in E, \mathsf{rank}(v) \geq_{l+1-\chi(v)} \mathsf{rank}(w)$

where $\mathsf{rank}(v) \geq_b \mathsf{rank}(w)$ denotes the $\geq$ relation in the lexicographic ordering, restricted to the first b elements of the tuple. If $\chi(v)$ is even, the inequalities are strict.

Proof. Consider a v with $\chi(v) \in \{m-1, m\}$ for an even m and let $\mathsf{rank}(v) = (r_l, 0, \dots, r_2, 0)$. By Lem. 19, $v \in Y_m^{r_l-1,0,\dots,r_{m-2}-1,0,r_m}$. If we look at the flattening of this formula in Lem.18, v is in particular, inside the middle term of this formula. That is,
$v \in \mathcal{B}_m[Y_m^{r_l-1,\dots,r_m-1}, Y_m^{r_l-1,\dots,r_m}]$. If we go through the definition of this term we get,

$$\Big(\bigcup_{i\in[m+1,l]} C_i\Big) \cup (\overline{C_m} \cap \mathsf{Npre}(Y_m^{r_l-1,\dots,r_m-1}, Y_m^{r_l-1,0,\dots,r_m})) \cup (C_m \cap \mathsf{Cpre}_{\mathsf{Odd}}(Y_m^{r_l-1,0,\dots,r_m-1}))$$

That gives us,
$$\begin{array}{ll} \text{if } \chi(v) = m, & v \in \mathsf{Cpre}_{\mathsf{Odd}}(Y_m^{r_l-1,0,\dots,r_m-1}) \\ \text{if } \chi(v) = m-1, & v \in \mathsf{Npre}(Y_m^{r_l-1,0,\dots,r_m-1}, Y_m^{r_l-1,0,\dots,r_m}) \end{array}$$

By the definition of Npre we get, if $\chi(v) = m - 1$ then $v \in \mathsf{Cpre}_{\mathsf{Odd}}(Y_m^{r_l-1,0,\dots,r_m})$. Since odd indices get 0-ranks, the claim of the lemma follows from the definition of $\mathsf{Cpre}_{\mathsf{Odd}}$ together with the observation $\mathsf{rank}(v) \geq_{l+1-m} \mathsf{rank}(w) \Leftrightarrow \mathsf{rank}(v) \geq_{l+1-(m-1)} \mathsf{rank}(w)$. ◀

Now we are ready to introduce the first of our three main propositions:

▶ **Proposition 21.** *If $\mathcal{W}_{Odd} \neq \emptyset$, there exists a non empty set $M := \{v \in \mathcal{W}_{Odd} \mid \mathsf{rank}(v) = (1, 0, 1, 0, \dots, 1, 0)\}$. Furthermore, for all $v \in M$, $\chi(v)$ is odd.*

Observe that $(1, 0, 1, 0, \dots, 1, 0)$ is the smallest rank possible. Therefore, $v \in M$ are the vertices that were added to Z in (7) in the first iteration of the fixed-point calculation and were never removed. The first part of the proposition follows from the monotonicity of fixed-point calculation. That is, if M was empty Z would be empty as well.

For the second part, observe that in the first iteration of the formula, for all j, $Y_j = \emptyset$. Also, $\mathsf{Cpre}_{\mathsf{Odd}}(\emptyset) = \emptyset$. Then from (7), Z does not contain any v with even priority.

▶ **Proposition 22.** *All cycles in $\mathcal{S}^{\mathcal{G}^\ell}$ that pass through a vertex in M are Odd winning.*

To see why Prop. 22 holds, we make an observation. For an even $m \leq l$, let $Y_m^{\mathbb{I}}$ denote the value of Y_m after the first ever iteration over it is completed, during the computation of 7. I.e. $Y_m^{\mathbb{I}} = Y^{0,0,\dots,0,1}$. Since for all j, $Y_j^0 = \emptyset$ and $X_{j-1}^0 = V$, Lem. 18 gives,

$$Y_m^{\mathbb{I}} = \bigcap_{j \in [\![m+2,l]\!]} \mathcal{B}_j[\emptyset, V] \cap \mathcal{B}_m[\emptyset, Y_m^{\mathbb{I}}] \cap \bigcap_{j \in [\![2,m-2]\!]} \mathcal{B}_j[Y_m^{\mathbb{I}} Y_m^{\mathbb{I}}] \tag{15}$$

If we go through the definition of $\mathcal{B}_j$ we see that: the first term of this formula adds or deletes $v \in C_j$ with $j > m$. It adds all the ones with odd j and removes all the ones with even j.The last term adds and removes $v \in C_j$ for $j \leq m-2$. It adds the ones in $\mathsf{Cpre}_{\mathsf{Odd}}(Y_m^{\mathbb{I}})$ and removes the ones that are not. The middle term eliminates C_m and all $v \in C_j \cap \neg \mathsf{Npre}(\emptyset, Y_m^{\mathbb{I}})$ for $j < m$, and adds $v \in C_{m-1} \cap \mathsf{Npre}(\emptyset, Y_m^{\mathbb{I}})$. If we go through the definition of Npre, we see that $\mathsf{Npre}(\emptyset, Y_m^{\mathbb{I}}) = \mathsf{Cpre}_{\mathsf{Odd}}(Y_m^{\mathbb{I}}) \cap (V_{\mathsf{Even}} \cup \mathsf{Lpre}^{\forall}(Y_m^{\mathbb{I}}))$. This gives,

$$v \in Y_m^{\mathbb{I}} \iff \chi(v) > m \text{ and is odd, or } \chi(v) < m \text{ and } v \in \mathsf{Npre}(\emptyset, Y_m^{\mathbb{I}}) \tag{16}$$

Then for all $v \in M$, $v \in Y_m^{\mathbb{I}}$ for each even $m \leq l$. In particular, $v \in Y_n^{\mathbb{I}}$ where n is such that $\chi(v) = n-1$. It follows that $v \in \mathcal{B}_n[\emptyset, Y_n^{\mathbb{I}}]$. Then, $v \in \mathsf{Cpre}_{\mathsf{Odd}}(Y_n^{\mathbb{I}}) \cap (V_{\mathsf{Even}} \cup \mathsf{Lpre}^{\forall}(Y_n^{\mathbb{I}}))$. Since all live outgoing edges of v are in $Y_n^{\mathbb{I}}$, for all (v, w) in $\mathcal{S}^{\mathcal{G}^\ell}$, $w \in Y_n^{\mathbb{I}}$.

By our previous observation w either has an odd priority larger than n, or is in $\mathsf{Cpre}_{\mathsf{Odd}}(Y_n^{\mathbb{I}}) \cap (V_{\mathsf{Even}} \cup \mathsf{Lpre}^{\forall}(Y_n^{\mathbb{I}}))$. If $\chi(w) > n$ is odd, then $w \in Y_{\chi(w)+1}^{\mathbb{I}}$, and we repeat the same argument to conclude the highest priority seen is always odd.

▶ **Definition 23.** *We call a play $\pi = v_1 v_2 \dots$ in $\mathcal{S}^{\mathcal{G}^\ell}$* minimal *if for all $v_i \in V_{\mathsf{Odd}}$, v_{i+1} is the minimum ranked successor of v_i. A minimal cycle is a section of a minimal play.*

▶ **Lemma 24.** *Every minimal play is Odd winning.*

A minimal play only sees minimal cycles. Let $\delta = w_1 w_2 \dots w_1$ be such a cycle. δ cannot be an Even winning cycle: Assume $b := \max\{\chi(w) \mid w \in \delta\}$ is even. Let $w_i \in \delta$ have priority b. By Obs. 20, $\mathsf{rank}(w_i) >_{l+1-b} \mathsf{rank}(w_{i+1}) \geq_{l+1-\chi(w_{i+1})} \dots \geq_{l+1-\chi(w_{i-1})} \mathsf{rank}(w_i)$. Since for all $w_j \in \delta$, $\chi(w_j) \leq b$, the inequality yields $\mathsf{rank}(w_i) >_{l+1-b} \mathsf{rank}(w_i)$, which is a contradiction.

▶ **Proposition 25.** *Any minimal play compliant with $\mathcal{S}^{\mathcal{G}^\ell}$ visits M infinitely often.*

Let $\delta = w_1 w_2 \dots w_1$ be a minimal cycle and w_k its vertex with maximum priority. We will show that $w_k \in M$. Since $\pi = \delta\delta\dots$ is a minimal play, by Lemma. 24 we know $\chi(w_k)$ is odd. Furthermore, we have observed in 16 that $w_k \in Y_m^{\mathbb{I}}$ for all $m > \chi(w_k)$. If we can show that $w_k \in Y_m^{\mathbb{I}}$ also for $m < \chi(w_k)$, then we have $w_k \in M$. We will now show this.

Assume to the contrary that $w_k \notin M$ and let j be the largest non-trivial index of $\mathsf{rank}(w_k)$. That is $j < l$ is the largest even integer such that $w_k \notin Y_j^{\mathbb{I}}$. Let t be the value of this index, i.e. $w_k \in Y_j^{0,\dots,0,t} \setminus Y_j^{0,\dots,0,t-1}$. Let us denote $Y_j^{0,\dots,0,t}$ by $Y_j^{\mathbf{t}}$ for short.

Since δ is minimal, Lem. 20 gives $\mathsf{rank}(w_i) \geq_{l+1-\chi(w_i)} \mathsf{rank}(w_{i+1})$ for all $w_i \in \delta$. Since $\chi(w_i) \leq \chi(w_k)$ for all i and $\chi(w_k) < j$; $\mathsf{rank}(w_i) \geq_{l+1-j} \mathsf{rank}(w_{i+1})$ for all $w_i \in \delta$. This implies $\mathsf{rank}(w) =_{l+1-j} \mathsf{rank}(w')$ for all $w, w' \in \delta$. It follows that for all $w \in \delta$, $w \in Y_j^{\mathbf{t}} \setminus Y_j^{\mathbf{t-1}}$.

Once more by Lem. 18 we get that for all $w \in \delta$,

$$w \in \mathcal{B}_j[Y_j^{\mathbf{t-1}}, Y_j^{\mathbf{t}}] = (\bigcup_{i \in [j+1,l]} C_i) \cup (\overline{C_j} \cap \mathsf{Npre}(Y_j^{\mathbf{t-1}}, Y_j^{\mathbf{t}}) \cup (C_j \cap \mathsf{Cpre}_{\mathsf{Odd}}(Y_j^{\mathbf{t-1}})))$$

Since $\chi(w) < j$, this implies

$$w \in \mathsf{Npre}(Y_j^{\mathbf{t-1}}, Y_j^{\mathbf{t}}) = \mathsf{Cpre}_{\mathsf{Odd}}(Y_j^{\mathbf{t}}) \cap (V_{\mathsf{Even}} \cup \mathsf{Lpre}^{\forall}(Y_j^{\mathbf{t}}) \cap \mathsf{Pre}^{\exists}_{\mathsf{Odd}}(Y_j^{\mathbf{t-1}}))$$

Now consider the set $Y_j^{\mathbf{t}} \setminus Y_j^{\mathbf{t-1}}$, which is initially empty. Then the first term in δ that gets in $Y_j^{\mathbf{t}} \setminus Y_j^{\mathbf{t-1}}$ has to be in $\mathsf{Pre}^{\exists}_{\mathsf{Odd}}(Y_j^{\mathbf{t-1}})$. This contradicts our assumption that all $w_i \in Y_j^{\mathbf{t}} \setminus Y_j^{\mathbf{t-1}}$ and proves that $w_k \in M$. We are now ready to prove the main theorem.

Proof of Thm. 14. Let $\pi = v_0 v_1 \dots$ be a play compliant with $\mathcal{S}^{\mathcal{G}^\ell}$ with $v_0 \in \mathcal{W}_{\mathsf{Odd}}$. Since π is compliant with an Odd strategy template, it is a fair play. For a node $v \in \mathcal{W}_{\mathsf{Odd}}$, let $v_{\min}$ be the minimum ranked successor of v. Since π is fair, for all v that is visited infinitely often in π, $v_{\min}$ is visited infinitely often as well. This gives us an infinite subsequence of π that is minimal. Since all minimal plays visit M infinitely often (Prop. 25), π visists M infinitely often. Then there must exist an $x \in M$ that π visits infinitely often. Then a tail of π is consisted of consecutive cycles over x. Since all cycles that pass through M are Odd winning (Prop. 22), π is Odd-winning. ◀

A.2 Details on Experimental Results

We conducted an experimental study to empirically validate the claim that our new Odd-fair Zielonka's algorithm retains its efficiency in practice. For this, we implemented the following algorithms (non-optimized) in C++:

- `OF-ZL`: Odd-fair Zielonka's algorithm (Alg. 1),
- `N-ZL`: "normal" Zielonka's algorithm from [46] (i.e., Alg. 1 with the simplifications described in Sec. 5.1),
- `OF-FP`: the fixed-point algorithm for Odd-fair parity games implementing (7) ,
- `N-FP`: the fixed-point algorithm for "normal" parity games from [13].

Of course, for both `N-ZL` and `N-FP` there exist optimized implementations (e.g. `oink` [44]). However, the goal of this section is to show a conceptual comparison, rather than evaluating best computation times. We believe this is better achieved using similar (non-optimized) implementations for all algorithms. In particular, by our experiments we show:

1. `OF-ZL`: significantly outperforms `OF-FP` on almost all benchmarks (Fig. 5 (right))
2. the performance of `OF-ZL` and `N-ZL` on the given benchmark set is very similar (Fig. 8),
3. the comparative performance of `OF-ZL` and `N-ZL` w.r.t. their respective fixed-point versions `OF-FP` and `N-FP`), respectively, is very similar (see Fig. 7).

All experiments where run on a large benchmark suite explained in Sec. A.2.1. To perform our experiments we used a machine equipped with Intel(R) Core(TM) i5-6600 CPU @ 3.30GHz and 8GB RAM. We declare a timeout when the calculation of an example exceeds 1 hour.

A.2.1 Benchmark

We generated Odd-fair parity game instances manipulating 286 benchmark instances of PGAME_Synth_2021 dataset of the SYNTCOMP benchmark suite [18] and 51 benchmark instances of the PGSolver dataset of Keiren's benchmark suite [21]. Within the latter, we restricted ourselves to instances with ≤ 5000 nodes. Both datasets contain examples of

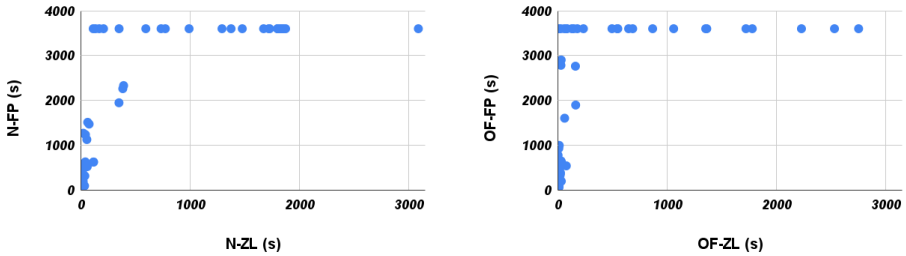

Figure 5 (Zoomed out version) A comparison of `N-FP` vs. `N-ZL` in regular parity games (left), and `OF-FP` vs. `OF-ZL` on fair parity games (right).

normal parity games. For each selected example, we generate Odd-fair parity game instances for a particular liveness percentage α. For a α%-liveness instant, we fix α% of the Odd nodes in the game, and turn α% of each of their outgoing edges to live edges.

On the Odd-fair instances with 50%−liveness generated from the SYNTCOMP benchmark suite, there are 204 instances where neither of the algorithms `OF-FP`, `OF-ZL`, `N-FP` or `N-ZL` timed out. On these instances, `OF-ZL` gives an average computation time of 4.6 seconds while `OF-FP` took 122.7 seconds on average. On the same examples, `N-ZL` takes on average 3.6 seconds to compute while `N-FP` takes 45.2 seconds. For the PGSolver dataset `OF-FP` timed out on all generated instances, whereas `OF-ZL` took 24.9 seconds on average to terminate.

A.2.2 Comparative Evaluation

In order to validate the computational advantage of `OF-ZL` over `OF-FP`, we ran both algorithms on all 50%-liveness instances generated from the SYNTCOMP benchmark dataset. On 58 of these instances, both algorithms time out. The run-times for all other instances are depicted in Fig. 5 (right), 6 (right) and 7 (right). The left plots in Fig. 5-7 show the same comparison for the "normal" parity algorithms `N-ZL` and `N-FP`. In both cases, Fig. 6 shows the zoomed-in version of the respective plot in Fig. 5. Fig. 7 shows the data-points from the respective plot in Fig. 6 as a scatter plot in log-scale. The examples on which only `x-FP` times out, can be seen as the dots on the ceiling of the plots in Fig. 5. In all plots, points above the diagonal correspond to instances where Zielonka's algorithm outperforms the fixed-point algorithm.

We clearly see in Fig. 5-7 that Zielonka's algorithm performs significantly better than the fixed-point version, both in the Odd-fair (right) and in the normal (left) case. More importantly, the overall performance comparison between `OF-ZL` over `OF-FP` (right plots) mimics the comparison between `N-ZL` over `N-FP`. This allows us to conclude that our new Odd-fair Zielonka's algorithm retains the computational advantages of Zielonka's algorithm.

A closer look at runtimes (see [37]) reveals `OF-ZL` has almost the same runtime as `N-ZL`, implying our changes in the algorithm incur almost no computational disadvantages over the original algorithm. This allows us to handle transition fairness for almost free in practice.

Conclusion. The results show that `OF-ZL` is significantly faster in solving Odd-fair parity games compared to `OF-FP`. The outcomes match the known comparison results between fixed-point calculation versus Zielonka's algorithm, on normal parity games.

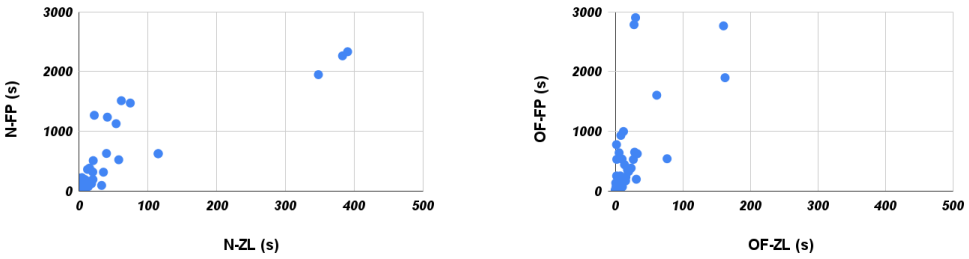

Figure 6 (Zoomed in version) A comparison of `N-FP` vs. `N-ZL` in regular parity games (left), and `OF-FP` vs. `OF-ZL` on fair parity games (right).

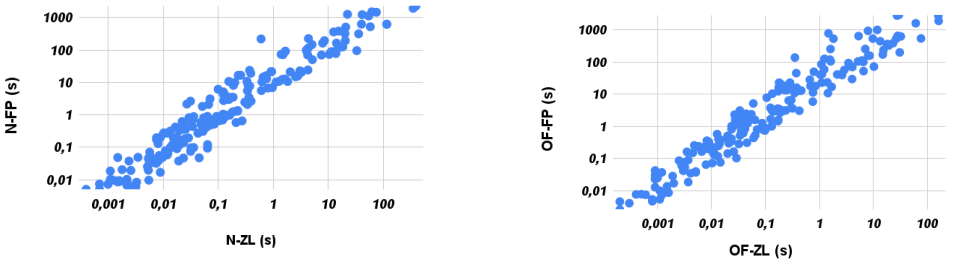

Figure 7 A comparison of `N-FP` vs. `N-ZL` in regular parity games (left), and `OF-FP` vs. `OF-ZL` on fair parity games (right) in terms of log-scale plots where the timeouts are removed.

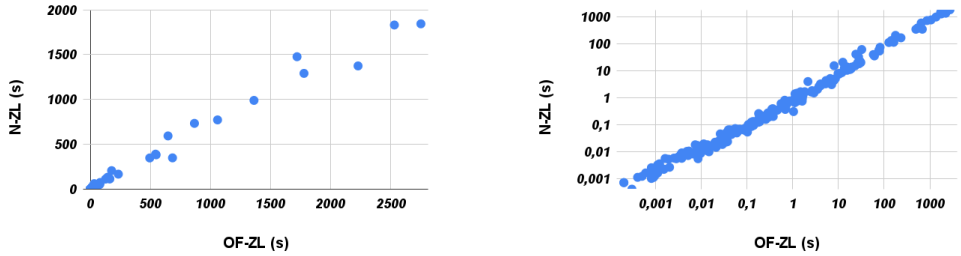

Figure 8 A comparison of `N-ZL` vs. `OF-ZL` over examples that do not timeout on both. Right hand side plot visualizes the same data in logscale.