Algorithm and Strategy Construction for Sure-Almost-Sure Stochastic Parity Games

Laurent Doyen
CNRS & LMF, ENS Paris-Saclay, Gif-sur-Yvette, France

Shibashis Guha
Tata Institute of Fundamental Research, Mumbai, India

Abstract

We consider turn-based stochastic two-player games with a combination of a parity condition that must hold surely, that is in all possible outcomes, and of a parity condition that must hold almost-surely, that is with probability 1. The problem of deciding the existence of a winning strategy in such games is central in the framework of synthesis beyond worst-case where a hard requirement that must hold surely is combined with a softer requirement. Recent works showed that the problem is coNP-complete, and infinite-memory strategies are necessary in general, even in one-player games (i.e., Markov decision processes). However, memoryless strategies are sufficient for the opponent player. Despite these comprehensive results, the known algorithmic solution enumerates all memoryless strategies of the opponent, which is exponential in all cases, and does not construct a winning strategy when one exists.

We present a recursive algorithm, based on a characterisation of the winning region, that gives a deeper insight into the problem. In particular, we show how to construct a winning strategy to achieve the combination of sure and almost-sure parity, and we derive new complexity and memory bounds for special classes of the problem, defined by fixing the index of either of the two parity conditions.

2012 ACM Subject Classification Mathematics of computing → Stochastic processes

Keywords and phrases stochastic games, parity objectives, reactive synthesis

Digital Object Identifier 10.4230/LIPIcs.STACS.2026.34

Related Version *Full Version*: `https://arxiv.org/abs/2601.03381` [16]

Funding *Shibashis Guha*: Partially supported by the Department of Atomic Energy, Government of India, under project no. RTI4014.

1 Introduction

Turn-based games provide a simple model for the synthesis of system components satisfying a given specification. A component is viewed as a player trying to achieve a winning condition defined by the specification, regardless of the behaviour of other components. Abstracting the other components as a combination of an adversarial and stochastic environment gives rise to the standard model of two-player stochastic games played on graphs [19, 20], where the vertices of the graph are partitioned into system, environment, and stochastic vertices. Starting from an initial vertex, a play is an infinite path in which the successor of each vertex is chosen by the player owning that vertex (for system and environment vertices) or according to a probability distribution (for stochastic vertices). In general, strategies define the choice of the players, which may depend on the sequence of previously visited vertices, called the history. The synthesis of a component corresponds to constructing a strategy for the system that maximizes the probability to win (i.e., to satisfy the winning condition), regardless of the strategy of the environment.

© Laurent Doyen and Shibashis Guha;
licensed under Creative Commons License CC-BY 4.0

We consider specifications described by parity conditions, which are canonical to express all ω-regular requirements on plays [28, 15], with the property that memoryless optimal strategies exists for both players [12, Theorem 1.2], maximizing the probability to win without having to remember the history of the game. Deciding which player has a winning strategy in parity games lies in NP $\cap$ coNP [12, Corollary 1.1], but no polynomial-time algorithm is known for that problem, even in non-stochastic games. Improved bounds and quasi-polynomial algorithms exist though [22, 8, 23, 24].

In component synthesis, a natural requirement is to ensure that the winning condition holds with probability 1, called *almost-sure winning*. For example, to send a message over a network that may lose messages with a fixed probability p, a strategy is to keep sending the same message until delivery, which is almost-sure winning because the probability of delivery failure after k attempts is vanishing, as $p^k \to 0$ when $k \to \infty$. An even stronger requirement is that all possible outcomes, no matter their probability should, satisfy the specification, called *sure winning*. This corresponds to the worst-case scenario where the choices at both environment and stochastic vertices are purely antagonistic. Sure winning is typically used for critical requirements, while almost-sure winning is used to ensure good performance in a stochastic environment. Specifications that combine both a sure and almost-sure parity condition have been considered recently [5, 13], as a fundamental instance of the framework of synthesis beyond worst-case [7]. For such specifications, games are determined [13, Theorem 5] and the synthesis problem is coNP-complete [13, Corollary 7]; memoryless strategies are sufficient for the environment player [13, Theorem 6], and infinite memory is necessary for the system player [5, Example 1], even in MDPs (Markov decision processes, i.e., games with no environment vertices).

Despite these comprehensive results, the picture is not complete. The first issue is algorithmic. The coNP algorithm [13] is to guess a memoryless strategy for the environment player, and then to check that the system player is losing in the resulting MDP (using determinacy), which can be done in NP $\cap$ coNP [5, Theorem 15]. In practice, enumerating the memoryless strategies for the environment is always exponential, thus inefficient. Another related issue is the construction of a winning strategy for the system player, which is central in the context of synthesis. Only in the case of MDPs, the structure of (infinite-memory) winning strategies has been described [5, Definition 10]. However, the current solution of the synthesis problem for games gives no clue about how the system should play in order to satisfy the specification. We argue that these issues reflect the lack of a deeper insight into games with a combination of sure and almost-sure parity conditions.

Contribution. We outline the main contributions of the paper.

- We present a recursive algorithm for deciding the existence of a strategy that is both sure winning for a first parity condition, and almost-sure winning for a second parity condition. The algorithm computes the winning set of the system player in worst-case exponential time. The correctness is established using characterisations of the winning sets for each player, and gives a better insight into the structure and properties of such games.
- We obtain a precise description of the winning strategies for the system player, and how to construct them by switching between simple (memoryless) strategies, and repeat them for long enough using unbounded counters. Intuitively, a strategy that is only almost-sure winning for both parity conditions may have violating outcomes, but representing a mass of probability 0. Eventually switching to a sure-winning strategy for the first parity condition is necessary to ensure all outcomes satisfy that condition, however possibly

at the price of violating the second condition. The crux is to switch so rarely that those violating outcomes represent a mass of probability 0, maintaining the almost-sure satisfaction of the second parity condition. A similar idea works in MDPs, combining strategies that almost-surely visit certain end-components [5]. However, the extension to stochastic games is non-trivial as the notion of end-components no longer exists in games.

- As a consequence of the new algorithm and analysis, we get new bounds on the complexity and memory requirement of subclasses of the problem, involving the parity index (i.e., the number of priorities defining a parity condition). When either of the two parity conditions has a fixed index, we show that the synthesis problem becomes solvable in NP ∩ coNP; when both have a fixed index, the problem is in P. When the sure condition is coBüchi, memoryless strategies are sufficient for the system player, and when the almost-sure condition is Büchi, finite memory is sufficient. This gives a tight and complete picture, as it is known that the combination of a sure Büchi and almost-sure coBüchi condition requires infinite memory, already in MDPs [5, Example 1 & Theorem 18].
- A technical result of independent interest is the construction of a deterministic parity automaton (DPW) equivalent to the intersection of two DPW, with a blow up that is exponential only in the parity index of either of the two DPW. Beyond the product of the state spaces of the two automata, the blow up is by a factor $\sqrt{\min(d_1^{d_2}, d_2^{d_1})}$ where d_i $(i = 1, 2)$ is the parity index of the i-th automaton. Our solution is not particularly involved, but we could not find a construction with this property in the literature. The construction is crucial in proving that the synthesis problem is in NP ∩ coNP when one of the parity conditions has a fixed index.

Related work. Synthesis of strategies that satisfy a combination of multiple objectives that arises naturally in many applications, such as verification [1], planning [21], and AI [26]. In non-stochastic games and thus for sure winning, combinations of parity objectives [10] and of mean-payoff objectives [29] have been considered. In stochastic games, the quantitative analysis of multi-objective conditions saw very little progress, in particular for Boolean objectives (such as the parity condition), and the known positive results hold for reachability objectives or subclasses of games [14, 4]. As observed in [5], combinations of quantitative objectives is sometimes conceptually easier than Boolean objectives. Almost-sure winning for multiple quantitative objectives has been studied in the case of mean-payoff conditions [9]. The framework of beyond-worst-case synthesis was initially studied with quantitative objectives such as shortest path and mean-payoff [7].

We already discussed the results that are closest to our work [5, 13]. We note that the notion of limit-sure winning (i.e., winning with probability arbitrarily close to 1) has also been considered in combination with sure winning, and the synthesis problem is coNP-complete, where the coNP bound is obtained by guessing a memoryless strategy for the environment player, and then solving an MDP with sure-limit-sure parity condition, which is shown to be polynomial-time reducible to the sure-almost-sure problem in MDPs [13]. Boolean combinations of sure and almost-sure winning have been considered for MDPs [3]. A natural extension would be to consider arbitrary probability of satisfaction (instead of probability 1) in combination with sure winning. This question is solved for MDPs [5], and remains open for games.

Omitted proofs are available in the extended version [16].

2 Preliminaries

2.1 Basic definitions

A *probability distribution* on a finite set V is a function $d : V \to [0,1]$ such that $\sum_{v \in V} d(v) = 1$. The *support* of d is the set $\mathsf{Supp}(d) = \{v \in V \mid d(v) > 0\}$. We denote by $\mathcal{D}(V)$ the set of all probability distributions on V. Given $v \in V$, we denote by 1_v the *Dirac distribution* on v that assigns probability 1 to v.

A *stochastic game* $\mathcal{G} = \langle V, (V_1, V_2, V_\Diamond), E, \delta \rangle$ consists of a finite set V of vertices, mutually disjoint (but possibly empty) sets $V_1, V_2, V_\Diamond$ of respectively Player 1, Player 2, and probabilistic vertices such that $V = V_1 \cup V_2 \cup V_\Diamond$, a set $E \subseteq V \times V$ of edges, and a probabilistic transition function $\delta : V_\Diamond \to \mathcal{D}(V)$. We denote by $E(v) = \{v' \mid (v, v') \in E\}$ the set of all successors of a vertex $v \in V$, and we require that every vertex has at least one successor, $E(v) \neq \varnothing$ for all vertices $v \in V$, and that $\mathsf{Supp}(\delta(v)) = E(v)$ for all probabilistic vertices $v \in V_\Diamond$. The decision problems in this paper are independent of the transition probabilities, and therefore stochastic games can be specified by giving only the partition $(V_1, V_2, V_\Diamond)$ and the set E of edges. A *Markov decision process (MDP)* for Player i is the special case of a stochastic game where one of the players has no role, that is $V_{3-i} = \varnothing$.

The game is played in rounds, starting from an initial vertex v_0. In a round at vertex v, if $v \in V_i$ $(i = 1, 2)$, then Player i chooses a successor $v' \in E(v)$ (which is always possible since $E(v) \neq \varnothing$), and otherwise $v \in V_\Diamond$ and a successor $v' \in E(v)$ is chosen with probability $\delta(v)$. The next round starts at vertex v'. A *play* is an infinite path $v_0, v_1, \ldots$ such that $(v_j, v_{j+1}) \in E$ for all $j \geq 0$. A *history* is a finite prefix of a play. For $\tau = v_0, v_1, \ldots, v_k$ and $j \leq k$, let $|\tau| = k+1$ be the length of τ, define $\mathsf{Last}(\tau) = v_k$, and $\tau(j\triangleright) = v_j, \ldots, v_k$. Given a function $\Omega : V \to Z$ defined on vertices, define $\Omega(\tau) = \Omega(v_0), \Omega(v_1), \ldots, \Omega(v_k)$ the extension of Ω to histories.

A *strategy* for Player i $(i = 1, 2)$ in $\mathcal{G}$ is a function $\sigma_i : V^* V_i \to V$ such that $(v, \sigma_i(\tau v)) \in E$ is an edge for all histories $\tau \in V^*$ and vertices $v \in V_i$. A play $v_0, v_1, \ldots$ is an *outcome* of σ_i if $v_{j+1} = \sigma_i(v_0, v_1, \ldots, v_j)$ for all $j \geq 0$ such that $v_j \in V_i$. We denote by $\mathsf{Out}(v, \sigma)$ the set of all outcomes of a strategy σ with initial vertex v, and by $\mathsf{Out}(v, \sigma_1, \sigma_2)$ the set $\mathsf{Out}(v, \sigma_1) \cap \mathsf{Out}(v, \sigma_2)$. We denote by $\mathsf{Pr}_{v_0}^{\sigma_1, \sigma_2}$ (or simply $\mathsf{Pr}_{v_0}^{\sigma_i}$ in the case of MDPs for Player i) the standard probability measure on the sigma-algebra over the set of (infinite) plays with initial vertex v_0, generated by the cylinder sets spanned by the histories [2].

A strategy σ_i uses *finite memory* if there exists a right congruence $\approx$ of finite index (i.e., that can be generated by a finite-state transducer) over histories such that $\tau \approx \tau'$ implies $\sigma_i(\tau) = \sigma_i(\tau')$.

An objective is a measurable set $\psi \subseteq V^\omega$ of plays. Given a priority function $\Omega : V \to \mathbb{N}$, we consider the classical parity objective $\mathsf{Parity}(\Omega) = \{v_0, v_1, \cdots \in V^\omega \mid \limsup_{j \to \infty} \Omega(v_j) \text{ is even}\}$ and denote by $\mathsf{Parity}^{\mathsf{c}}(\Omega) = V^\omega \setminus \mathsf{Parity}(\Omega)$ the complement of that objective (which can itself be defined as a parity objective, using priority function Ω^{+1} where $\Omega^{+1}(v) = \Omega(v) + 1$). Büchi objectives are the special case of priority functions $\Omega : V \to \{1, 2\}$ that require visiting priority 2 infinitely often, while coBüchi objectives are the special case of priority functions $\Omega : V \to \{0, 1\}$ that require visiting priority 1 only finitely often. A key property of parity objectives is prefix-independence: $\theta \in \mathsf{Parity}(\Omega)$ if and only if $\tau\theta \in \mathsf{Parity}(\Omega)$, for all $\theta \in V^\omega$ and $\tau \in V^*$. Given a set $U \subseteq V$ and priority $p \in \mathbb{N}$, we denote by $U(\Omega, p) = \{v \in U \mid \Omega(v) = p\}$ the set of vertices of U with priority p, and by $\pi_{\Omega \geq p}(\tau)$ the sequence obtained from $\Omega(\tau)$ by removing all priorities smaller than p. For example, if $\tau = v_2 v_3 v_1 v_3 v_4 v_1 v_3$ and $\Omega(v_i) = i$, then $\pi_{\Omega \geq 3}(\tau) = 3343$.

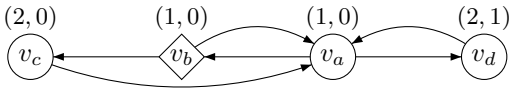

Figure 1 An MDP where the combination of sure Büchi and almost-sure coBüchi requires infinite memory [5]. Each node v is labeled by the pair of priorities $(\Omega_1(v), \Omega_2(v))$, inducing a (sure) Büchi condition to visit a vertex in $\{v_c, v_d\}$ infinitely often, and an (almost-sure) coBüchi condition to visit v_d finitely often.

A strategy σ_1 of Player 1 is *sure winning* for objective ψ from an initial vertex v if $\mathsf{Out}(v, \sigma_1) \subseteq \psi$, and *almost-sure winning* if $\mathsf{Pr}_v^{\sigma_1,\sigma_2}(\psi) = 1$ for all strategies σ_2 of Player 2. Note that sure winning for ψ immediately implies almost-sure winning for ψ (not the converse), and that a strategy that is almost-sure winning for ψ_1 and for ψ_2 is almost-sure winning for $\psi_1 \cap \psi_2$ (here the converse holds).

We consider the sure-almost-sure synthesis problem for parity objectives, which is to decide, given a stochastic game $\mathcal{G}$ with initial vertex v and two priority functions Ω_1, Ω_2, whether there exists a strategy σ_1 of Player 1 that is both sure winning for objective $\mathsf{Parity}(\Omega_1)$ and almost-sure winning for objective $\mathsf{Parity}(\Omega_2)$. We call such a strategy *sure-almost-sure* winning (assuming that the game $\mathcal{G}$, initial vertex v, and priority functions Ω_1, Ω_2 are clear from the context). We call the set of all initial vertices for which such a strategy exists the sure-almost-sure winning region, and we call the complement the sure-almost-sure spoiling region. Sure winning regions and almost-sure winning regions are defined analogously.

It is known that finite-memory strategies are not sufficient for sure-almost-sure winning, already in MDPs with a combined sure Büchi and almost-sure coBüchi objective [5], as illustrated in Figure 1 (in figures, Player-1 vertices are shown as circles, Player-2 vertices as boxes, and probabilistic vertices as diamonds). There is a strategic choice only in v_a. Surely visiting infinitely often a vertex in $\{v_c, v_d\}$, and at the same time almost-surely visiting only finitely often v_d can be done by choosing v_b from v_a more and more often as compared to choosing v_d, in a way such that choosing v_d infinitely often has probability 0, yet v_d is visited infinitely often in all outcomes where v_c is no longer visited from some point on.

2.2 Subgames and traps

Let $\mathsf{SurePre}_i : 2^V \to 2^V$ and $\mathsf{PosPre}_i : 2^V \to 2^V$ be respectively the sure and positive predecessor operator for Player i defined, for all $U \subseteq V$, by $\mathsf{SurePre}_i(U) = \{v \in V_i \mid E(v) \cap U \neq \varnothing\} \cup \{v \in V_{3-i} \cup V_\Diamond \mid E(v) \subseteq U\}$, and by $\mathsf{PosPre}_i(U) = \{v \in V_i \cup V_\Diamond \mid E(v) \cap U \neq \varnothing\} \cup \{v \in V_{3-i} \mid E(v) \subseteq U\}$.

Given a set $T \subseteq V$, the sure attractor $\mathsf{SureAttr}_i(T)$ for Player i is the least fixed point of the operator $X \mapsto \mathsf{SurePre}_i(X) \cup T$, thus $\mathsf{SureAttr}_i(T) = \bigcup_{j \geq 0} \mathsf{SurePre}_i^j(T)$ (where $\mathsf{SurePre}_i^0(T) = T$) and the positive attractor $\mathsf{PosAttr}_i(T)$ for Player i is the least fixed point of the operator $X \mapsto \mathsf{PosPre}_i(X) \cup T$.

Intuitively, from the vertices in $\mathsf{SureAttr}_i(T)$ Player i has a memoryless strategy to ensure eventually (in fact, within at most $|V|$ rounds) reaching a vertex in T regardless of the choices of Player $3-i$ and of the outcome of the probabilistic transitions; from the vertices in $\mathsf{PosAttr}_i(T)$ Player i has a memoryless strategy to ensure reaching with positive probability T regardless of the choices of Player $3-i$. We call such strategies the attractor strategies.

A set $U \subseteq V$ induces a *subgame* of $\mathcal{G}$, denoted by $\mathcal{G}_{\restriction U}$, if $E(v) \cap U \neq \varnothing$ for all $v \in U \cap (V_1 \cup V_2)$, and $E(v) \subseteq U$ for all $v \in U \cap V_\Diamond$, that is, the two players can cooperate to keep the play forever in U. Such a set U is a *trap* for Player i if, moreover, $E(v) \subseteq U$ for

all $v \in U \cap V_i$. For example, the set $V \setminus \mathsf{PosAttr}_i(T)$ is always a trap for Player i. A key property of traps is that if Player i has a sure (resp., almost-sure) winning strategy in $\mathcal{G}_{\restriction U}$ (for some objective ψ) from an initial vertex $v \in U$ where U is a trap for Player $3-i$, then Player i has a sure (resp., almost-sure) winning strategy in $\mathcal{G}$ from v for ψ.

Consider a set $U \subseteq V$ that only satisfies the condition $E(v) \cap U \neq \varnothing$ for all $v \in U \cap (V_1 \cup V_2)$, as it is the case for $U = V \setminus \mathsf{SureAttr}_i(T)$. Let $\mathsf{Bnd}(U) = \{v \in U \cap V_\Diamond \mid E(v) \not\subseteq U\}$ be the set of (boundary) probabilistic vertices in U with a successor outside U. Define the *subgame closure* $\lceil \mathcal{G} \rceil_U = \langle U', (U_1, U_2, U_\Diamond), E', \delta' \rangle$ by redirecting the (probabilistic) transitions leaving U to a sink vertex, formally (see also Figure 4):

- $U' = U \cup \{v_{\mathsf{sink}}\}$,
- $U_1 = U \cap V_1$; $U_2 = U \cap V_2$; $U_\Diamond = (U \cap V_\Diamond) \cup \{v_{\mathsf{sink}}\}$,
- $E' = E \cap (U \times U) \;\cup\; \mathsf{Bnd}(U) \times \{v_{\mathsf{sink}}\} \;\cup\; \{(v_{\mathsf{sink}}, v_{\mathsf{sink}})\}$,
- $\delta'(v)(v') = \delta(v)(v')$ for all $v \in U_\Diamond$ and $v' \in U$; $\delta'(v)(v_{\mathsf{sink}}) = \sum_{v' \in V \setminus U} \delta(v)(v')$ for all $v \in U_\Diamond$; and $\delta'(v_{\mathsf{sink}}) = 1_{v_{\mathsf{sink}}}$.

Note that if U induces a subgame of $\mathcal{G}$, then $\lceil \mathcal{G} \rceil_U$ is the disjoint union of $\mathcal{G}_{\restriction U}$ and $\{v_{\mathsf{sink}}\}$. We generally assign priority $\Omega_1(v_{\mathsf{sink}}) = \Omega_2(v_{\mathsf{sink}}) = 0$ to the sink vertex, making it winning for Player 1. Then the key property is that if Player 1 does not have a sure (resp., almost-sure) winning strategy in $\lceil \mathcal{G} \rceil_U$ (for some objective ψ) from an initial vertex $v \in U$, then Player 1 does not have a sure (resp., almost-sure) winning strategy in $\mathcal{G}$ from v for ψ.

3 Conjunction of Parity Conditions

It will be useful in the sequel to consider conjunctions of parity conditions: Our algorithm (Section 5) needs to compute the almost-sure winning region for the conjunction $\mathsf{Parity}(\Omega_1) \cap \mathsf{Parity}(\Omega_2)$ of two parity objectives.

We revisit the problem of transforming an objective specified as the conjunction of two parity conditions into an objective specified by a single parity condition. The transformation is always possible because the parity condition is a canonical way to express ω-regular objectives, and the class of ω-regular objectives is closed under Boolean operations [28]. However, we were unable to find a direct construction in the literature, in particular a construction that would incur only a polynomial blowup when either of the two priority functions has a fixed range (i.e., when the number of priorities is fixed in one of the parity conditions).

Such a construction being of independent interest (e.g., in formal languages and automata theory), we present it using automata models without referring to objectives and games. A *deterministic automaton* over a finite alphabet Σ is a tuple $\mathcal{A} = \langle Q, q_\varepsilon, \delta, \alpha \rangle$ consisting of a finite state space Q, an initial state $q_\varepsilon \in Q$, a transition function $\delta : Q \times \Sigma \to Q$, and an acceptance condition (which defines a subset of Q^ω), here a conjunction of parity conditions. A run of $\mathcal{A}$ on an infinite word $w = \sigma_0 \sigma_1 \dots$ is a sequence $q_0 q_1 \dots$ such that $q_0 = q_\varepsilon$ and $\delta(q_i, \sigma_i) = q_{i+1}$ for all $i \geq 0$, hence every word has a unique run. Given priority functions $\Omega_1, \Omega_2 : Q \to \mathbb{N}$, the set $L(\mathcal{A})$ of all words whose run satisfies the parity conditions $\mathsf{Parity}(\Omega_1)$ and $\mathsf{Parity}(\Omega_2)$ is called the *language* of $\mathcal{A}$. We denote such automata by D2PW, and in the special case where $\Omega_1 = \Omega_2$ by DPW, which is the classical deterministic parity automata.

The blowup in the transformation of D2PW into DPW is inevitably exponential, as shown in [6, Theorem 9] and confirmed by the fact that games with a conjunction of two parity objectives are coNP-complete [10], whereas with a single parity objective they are in NP $\cap$ coNP [18]. A natural path for this transformation is to view the parity condition as a special case of Streett conditions [6] which are closed under intersection, then to convert the

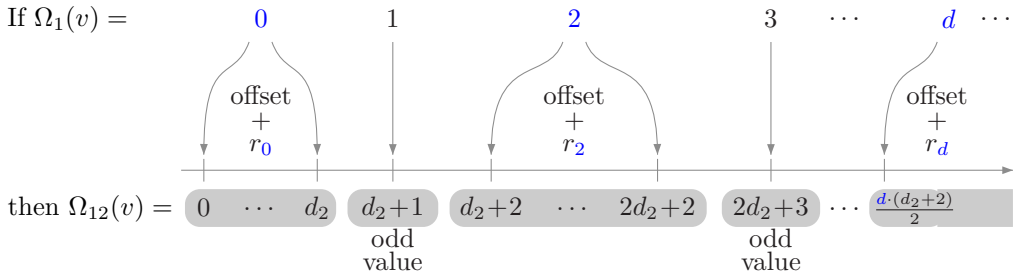

Figure 2 Construction of the priority function Ω_{12} for the conjunction of the parity condition defined by the priority functions Ω_1 and Ω_2. The offset for an even Ω_1-priority d is calculated as $\frac{d}{2} \cdot (d_2 + 2)$. The register r_d stores the largest Ω_2-priority since the last visit to an Ω_1-priority d.

Streett automaton (obtained from the D2PW) into a DPW. Priority functions with range $[d_1] = \{0, \ldots, d_1\}$ and $[d_2] = \{0, \ldots, d_2\}$ give $k = O(d_1 + d_2)$ Streett pairs, and the blowup in the conversion to DPW is by a factor in $O(k^k)$ [27]. A key observation is that this is an exponential blowup, even if one of d_1 or d_2 is constant. The approach "flattens" the parity conditions by "merging" them, which also happens when using (nondeterministic) Büchi automata[1] [6, Section 3.1].

In contrast, we present a direct construction of DPW from D2PW that is conceptually simple and with a blowup of at most $O(\max(d_1^{d_2/2}, d_2^{d_1/2}))$, thus polynomial if either of the priority functions has a fixed range.

The construction

Given a D2PW where the largest priority in the two parity conditions are d_1 and d_2, we construct an equivalent DPW where the largest priority is $O(d_1 \cdot d_2)$. We informally refer to priorities defined by a priority function Ω_i as Ω_i-priorities. It will be convenient to assume that d_2 is even, which is without loss of generality (we may increment d_2 if d_2 is odd).

Intuitively, given a finite run $\tau = q_0 q_1 \ldots q_k$ on some finite word w in the D2PW, where the priorities are $\Omega_1(\tau) = a_0 a_1 \ldots a_k$ and $\Omega_2(\tau) = b_0 b_1 \ldots b_k$, the run of our DPW on w has priority sequence $c_0 \ldots c_k$ with the following properties (see also Figure 2):

(P_1) if the Ω_1-priority a_k is odd, then c_k is odd and the value of c_k depends only on a_k;

(P_2) for all runs τ' on some finite word w' in the D2PW, with $\Omega_1(\tau') = a'_0 \ldots a'_k$, if $a_k < a'_k$ then $c_k < c'_k$ (where c'_k is the priority of the last state in the run of our DPW on w'); this is achieved by allocating disjoint segments where the value of c_k may lie. The segment associated with a_k has range $[d_2]$ if a_k is even, and it is a single odd value if a_k is odd;

(P_3) if the Ω_1-priority a_k is even, then the value of c_k is calculated as the sum of an offset $\nu(a_k)$ corresponding to the origin of the segment associated with a_k and the largest Ω_2-priority since the previous visit to the Ω_1-priority a_k (or since the beginning of the run if a_k is visited for the first time in τ). That is, $c_k = \nu(a_k) + \max\{b_i, b_{i+1}, \ldots, b_{k-1}\}$ where i is the largest index smaller than k such that $a_i = a_k$ (or, $i = 0$).

[1] There is a quadratic translation of DPW into nondeterministic Büchi automata, the intersection of Büchi automata gives a Büchi automaton with polynomial blowup, but going back to DPW is exponential [25]. This approach also "flattens" the parity condition in the sense that the exponential is in both d_1 and d_2.

In order to satisfy (P_3), the DPW automaton maintains, for each even Ω_1-priority d, a register r_d that contains the largest Ω_2-priority since the previous visit to a state with Ω_1-priority d (or since the beginning of the run).

Given a D2PW $\mathcal{A} = \langle Q, q_\varepsilon, \delta, \alpha\rangle$ where the acceptance condition α is a conjunction of parity conditions defined by two priority functions Ω_1 and Ω_2, let $\mathcal{A}' = \langle Q', q'_\varepsilon, \delta', \alpha'\rangle$ be the DPW defined as follows:

- $Q' = Q \times [d_2]^{[d_1]_{\text{even}}}$ where $[d_1]_{\text{even}} = \{0, 2, \ldots, 2\lfloor \frac{d_1}{2} \rfloor\}$; a state $(q, r_0, r_2, \ldots, r_{2\lfloor \frac{d_1}{2} \rfloor}) \in Q'$ consists of a state q of $\mathcal{A}$ and, for each even Ω_1-priority d, a register r_d with range $[d_2] = \{0, \ldots, d_2\}$;
- $q'_\varepsilon = (q_\varepsilon, 0, 0, \ldots, 0)$;
- for every state $(q, r_0, r_2, \ldots, r_{2\lfloor \frac{d_1}{2} \rfloor}) \in Q'$ and $\sigma \in \Sigma$, define $\delta((q, r_0, r_2, \ldots, r_{2\lfloor \frac{d_1}{2} \rfloor}), \sigma) = (q', r'_0, r'_2, \ldots, r'_{2\lfloor \frac{d_1}{2} \rfloor})$ where $q' = \delta(q, \sigma)$ and for every $d \in [d_1]_{\text{even}}$:

$$r'_d = \begin{cases} \max(r_d, \Omega_2(q)) & \text{if } \Omega_1(q) \neq d \\ \Omega_2(q) & \text{if } \Omega_1(q) = d \end{cases}$$

- the acceptance condition α' is a parity condition defined by the priority function Ω_{12} such that $\Omega_{12}(q, r_0, r_2, \ldots, r_{2\lfloor \frac{d_1}{2} \rfloor}) = \begin{cases} \frac{d\cdot(d_2+2)+d_2}{2} & \text{if } d = \Omega_1(q) \text{ is odd} \\ \frac{d\cdot(d_2+2)}{2} + r_d & \text{if } d = \Omega_1(q) \text{ is even} \end{cases}$

  The value $\nu(d) = \frac{d\cdot(d_2+2)}{2}$ is called the *offset* corresponding to even Ω_1-priority d (see Figure 2).

▶ **Example 1.** In Figure 3, on the left side, we have an input D2PW $\mathcal{A}$ with states Q and alphabet $\{a, b\}$. To accept a word, it must visit q_3 infinitely often. In the figure, above every state $q \in Q$, we write the priorities $\Omega_1(q)$ and $\Omega_2(q)$ in parentheses. Thus $\Omega_1(q_0) = \Omega_1(q_1) = 0$, $\Omega_1(q_2) = 2$, $\Omega_1(q_3) = 1$ and $\Omega_2(q_0) = 5$, $\Omega_2(q_1) = 3$, $\Omega_2(q_2) = 1$, $\Omega_2(q_3) = 6$.

In the figure on the right, we show the constructed automaton $\mathcal{A}'$ with states Q'. Note that the only even priorities that appear on the first dimension for automaton $\mathcal{A}$ are 0 and 2. As stated above, the initial state of $\mathcal{A}'$ is $q'_0 = (q_0, 0, 0)$. As the automaton $\mathcal{A}$ reads a letter and moves to state q_1 from q_0, then the maximum priority seen on the second dimension since the last visit to a state q with $\Omega_1(q) = 0$ is 5 and the maximum priority seen on the second dimension since the last visit to a state q with $\Omega_1(q) = 2$ is also 5. Hence $\mathcal{A}'$ moves to a state $(q_1, 5, 5)$ from the initial state $(q_0, 0, 0)$. The automaton $\mathcal{A}$ then reads another letter to move to q_2 from q_1. For $d = 0$, we have that q_1 is the last state q visited before the current visit to q_2 such that $\Omega_1(q) = 0$. Since $\Omega_2(q_1) = 3$, we now move to the state $(q_2, 3, 5)$ in $\mathcal{A}'$. From q_2, if $\mathcal{A}$ now reads a letter and moves to q_0, since $\Omega_1(q_2) = 2$ and $\Omega_2(q_2) = 1$, we move to the state $(q_0, 3, 1)$ in $\mathcal{A}'$. The other states in $\mathcal{A}'$ can be defined similarly.

Now we describe the priority function Ω_{12}. Note that $d_2 = 6$. Since $\Omega_1(q_0) = 0$ which is even, we have that $\Omega_{12}(q_0, 0, 0) = \frac{0\cdot 8}{2} + r_0 = 0 + 0 = 0$. Again, since $\Omega_1(q_1) = 0$, we have that $\Omega_{12}(q_1, 5, 5) = \frac{0\cdot 8}{2} + r_0 = 0 + 5 = 5$. For the state $(q_2, 3, 5)$, since $\Omega_1(q_2) = 2$, we have that $\Omega_{12}(q_2, 3, 5) = \frac{2\cdot 8}{2} + r_2 = 8 + 5 = 13$. For the state $(q_3, 3, 1)$, since $\Omega_1(q_3) = 1$ which is odd, we have that we have that $\Omega_{12}(q_3, 3, 1) = \frac{1\cdot 8+6}{2} = 7$. For the other states in Q', the parity function Ω_{12} can be defined analogously. The priority corresponding to each state in $\mathcal{A}'$ appears above the state in the figure. Further, for every transition from a state (q, r_2, r_4) to a state (q', r'_2, r'_4) in $\mathcal{A}'$, we have a corresponding transition from state q to q' in $\mathcal{A}$ and the letter labelling the transition in $\mathcal{A}'$ is the same as the letter labelling the transition in $\mathcal{A}$. To avoid clutter, we did not add these letters on the transitions in the figure describing $\mathcal{A}'$.

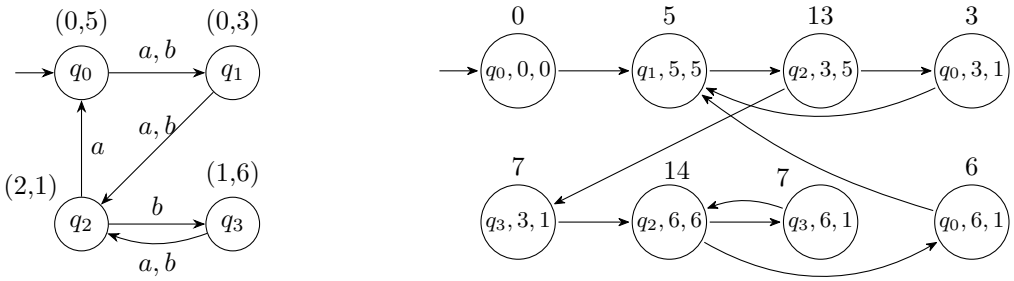

Figure 3 An example of a D2PW (left) and the DPW constructed (right).

The number of priorities in $\mathcal{A}'$ is in $O(d_1 \cdot d_2)$ and the blowup in the number of states is by a factor $d_2^{d_1/2}$. By exchanging the two priority functions in our construction, we obtain a blowup factor of $d_1^{d_2/2}$, and we can choose the construction with the smaller of the two factors.

▶ **Theorem 2.** *Given a D2PW $\mathcal{A}$, we can construct a DPW $\mathcal{A}'$ such that $L(\mathcal{A}) = L(\mathcal{A}')$, the number of priorities in $\mathcal{A}'$ is $O(d_1 \cdot d_2)$, and the blowup in the number of states (of $\mathcal{A}'$ as compared to $\mathcal{A}$) is by a factor $\sqrt{\min(d_1^{d_2}, d_2^{d_1})}$.*

As a corollary, the transformation in Theorem 2 is polynomial when either of the priority functions in the D2PW has a fixed (but arbitrary) range. In particular, it follows that the complexity of the synthesis problem for games with a conjunction of two sure (or almost-sure) parity conditions, which is coNP-complete in general, becomes NP ∩ coNP when either of the priority functions defining the parity conditions has a fixed range, as it reduces in polynomial time to a standard parity game.

Note that the same construction can be used to transform a disjunction of two parity conditions into a single parity condition. This follows since $\psi_1 \cup \psi_2 = (\psi_1^c \cap \psi_2^c)^c$ and because the complement of a parity condition defined by a priority function Ω is also a parity condition, defined by the priority function Ω^{+1}.

4 Winning Strategy and Winning Region

We describe informally a sure-almost-sure winning strategy for Player 1. We start by playing an almost-sure winning strategy for the conjunction of parity objectives $\mathsf{Parity}(\Omega_1) \cap \mathsf{Parity}(\Omega_2)$. Conjunction of parity objectives can be expressed as a single parity objective over an exponentially larger game graph (Theorem 2) and therefore we can assume that the almost-sure winning strategy uses finite memory.

To guarantee sure winning for the first objective, the crux is then to avoid outcomes that violate $\mathsf{Parity}(\Omega_1)$ (which is possible, though with probability 0) by switching from time to time to a strategy that ensures a visit to a large even priority for Ω_1, ignoring the second objective thus possibly at the price of visiting a large odd priority for Ω_2, then switching back to an almost-sure winning strategy for the conjunction of objectives. The switching should be so rare that the probability of switching infinitely often, thereby potentially violating the second objective $\mathsf{Parity}(\Omega_2)$, would be 0. On the other hand, the switching should still guarantee in all cases to eliminate the possibility of a single outcome violating $\mathsf{Parity}(\Omega_1)$.

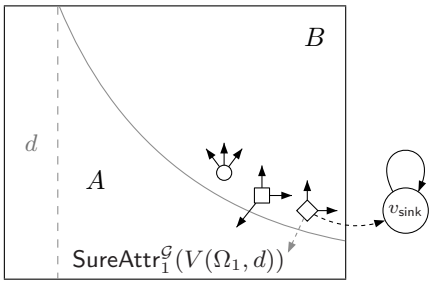

Figure 4 Decomposition of a winning region with an even largest Ω_1-priority d, into the sure attractor A (for Player 1) to d and the subgame closure of $B = V \setminus A$.

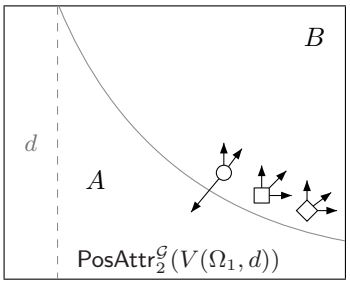

Figure 5 Decomposition of a winning region with an odd largest Ω_1-priority d, into the positive attractor A (for Player 2) to d and the subgame induced by $B = V \setminus A$.

Intuitively, a condition for switching is to have seen in the play a long sequence of odd priorities, without seeing any larger priority. Given odd priority p and integer N, let

$$\mathsf{Unlucky}(p, N) = \{\rho \in V^\omega \mid \text{the first } N \text{ elements of } \pi_{\Omega_1 \geq p}(\rho) \text{ are } p\}$$

be the event that at least N priorities $\geq p$ occur and the first N such priorities are p.

In the MDP obtained after fixing an almost-sure winning strategy for $\mathsf{Parity}(\Omega_1)$ in the game $\mathcal{G}$, it becomes unlikely to be unlucky (i.e., to observe $\mathsf{Unlucky}(p, N)$) as N grows to infinity.

▶ **Lemma 3.** *Consider an MDP with a priority function Ω_1 defining the parity objective $\psi = \mathsf{Parity}(\Omega_1)$, such that $\mathsf{Pr}_v^\sigma(\psi) = 1$ for all vertices $v \in V$ and all strategies σ. Let p be an odd priority in the range of Ω_1. Then for all $\varepsilon > 0$, there exists N_ε such that for all strategies σ and all vertices v, we have $\mathsf{Pr}_v^\sigma(\mathsf{Unlucky}(p, N_\varepsilon)) < \varepsilon$.*

Intuitively, since $\mathsf{Unlucky}(p, N)$ happens with positive probability, and if Player 1 switches from an almost-sure winning strategy whenever the event $\mathsf{Unlucky}(p, N)$ happens, using Borel-Cantelli lemma [17], Player 1 will then switch (even infinitely often) with probability 1. This holds for all fixed N even though the probability to be unlucky can be made arbitrarily small by choosing N appropriately (according to Lemma 3). It is undesirable to switch infinitely often, as this may cause the violation of the second parity objective. However, if we increase N upon seeing N priorities $\geq p$ (whether unlucky or not), the probability to switch infinitely often may be reduced to 0, which is acceptable.

For all $i > 0$, let $N_i = N_\varepsilon$ where N_ε is defined by Lemma 3 for $\varepsilon = 2^{-i}$. Using this sequence (N_i), we show that the probability to never be unlucky from some round i can be (lower) bounded by the expression $\prod_{j \geq i} \left(1 - 2^{-j}\right)$. Simple calculations show that this expression tends to 1 as $i \to \infty$ (e.g., see the proof of [5, Lemma 12]).

Characterisation of the winning region

We present two lemmas giving characterisations of (1) the sure-almost-sure winning region W (for Player 1) when its largest Ω_1-priority is even, and (2) the sure-almost-sure spoiling region $V \setminus W$ (for Player 2) when its largest Ω_1-priority is odd. These are the base cases required to establish the correctness of the recursive algorithm that we present in Section 5.

The symmetric cases, where the largest Ω_1-priority is odd in the winning region or is even in the spoiling region, are solved by recursive calls on subgames with a smaller number of Ω_1-priorities.

In Lemma 4, we show that the sure-almost-sure winning region W, which is a trap for Player 2 (inducing a subgame in which all vertices are winning), is characterised, when its largest Ω_1-priority d is even, by the existence of (see also Figure 4):

- an almost-sure winning strategy σ_{AS} for Player 1, for objective $\mathsf{Parity}(\Omega_1) \cap \mathsf{Parity}(\Omega_2)$;
- a sure-almost-sure winning strategy σ_{sub} for Player 1 for the parity objectives $\mathsf{Parity}(\Omega_1)$ and $\mathsf{Parity}(\Omega_2)$ in the subgame closure $\lceil \mathcal{G} \rceil_B$, where $B = V \setminus A$ and $A = \mathsf{SureAttr}_1(V(\Omega_1, d))$.

It is easy to show that these conditions are necessary for Player 1 to win in W, and the crux of Lemma 4 is to show that they are sufficient as well, by constructing a sure-almost-sure winning strategy σ_1 from σ_{AS}, σ_{sub}, and the attractor strategy σ_{Attr} in A. We describe σ_1 informally when the sure winning objective is a Büchi condition, namely the priorities are $\{1, 2\}$: a sure-almost-sure winning strategy first plays like σ_{AS}, and sets a horizon N by which priority 2 should be visited at least once. The larger is N, the more likely this would happen (Lemma 3). Nevertheless, in order to ensure the stronger objective of sure winning for $\mathsf{Parity}(\Omega_1)$, the strategy σ_1 deviates from σ_{AS} in the unlucky event that priority 2 is not visited within horizon N: as long as the history remains in B, we follow the sure-almost-sure winning strategy σ_{sub}, and whenever the history enters A we follow the attractor strategy σ_{Attr} until priority 2 is visited and then continue with σ_{AS}. The attractor strategy may endanger the (almost-sure) satisfaction of the second parity objective, for instance by visiting a large odd Ω_2-priority. We ensure that the probability that this happens infinitely often is 0 by increasing the horizon N as described after Lemma 3. The generalization of this construction to a sure parity objective is technical, requiring to track each Ω_1-priority, but conceptually analogous.

▶ **Lemma 4.** *Let $\mathcal{G}$ be a stochastic game with sure-almost-sure parity objective induced by the priority functions Ω_1, Ω_2, such that the largest Ω_1-priority d in $\mathcal{G}$ is even. Let $A = \mathsf{SureAttr}_1(V(\Omega_1, d))$ and let $B = V \setminus A$.*

All vertices in $\mathcal{G}$ are sure-almost-sure winning for Player 1 for the parity objectives $\mathsf{Parity}(\Omega_1)$ and $\mathsf{Parity}(\Omega_2)$ if and only if:

- *all vertices in $\mathcal{G}$ are almost-sure winning for Player 1 for the objective $\mathsf{Parity}(\Omega_1) \cap \mathsf{Parity}(\Omega_2)$, and*
- *all vertices in $\lceil \mathcal{G} \rceil_B$ are sure-almost-sure winning for Player 1 for the parity objectives $\mathsf{Parity}(\Omega_1)$ and $\mathsf{Parity}(\Omega_2)$.*

Proof. We first show that the conditions in the lemma are sufficient to ensure that all vertices in $\mathcal{G}$ are sure-almost-sure winning for Player 1.

Let σ_{AS} be an almost-sure winning strategy for Player 1 for the objective $\mathsf{Parity}(\Omega_1) \cap \mathsf{Parity}(\Omega_2)$, let σ_{Attr} be the sure-attractor strategy to $V(\Omega_1, d)$, and σ_{sub} be sure-almost-sure winning strategy for Player 1 for the parity objectives $\mathsf{Parity}(\Omega_1)$ and $\mathsf{Parity}(\Omega_2)$ in $\lceil \mathcal{G} \rceil_B$.

We define a strategy σ_1 for Player 1, and then show that it is sure-almost-sure winning. The strategy uses a Boolean flag $\mathsf{unlucky}$ and integer variables k_p, i_p for each odd priority p occurring in $\mathcal{G}$. The variable k_p is a pointer to a position in the history, and i_p is the phase number associated with priority p. Initially $\mathsf{unlucky} = \bot$ (false) and $k_p = i_p = 0$ for all odd p.

Given a history $\tau \in V^* V_1$, the value $\sigma_1(\tau)$ and the update of the variables are defined by executing the following instructions sequentially, and terminating when an instruction *play* is executed (where the sequence (N_i) was defined after Lemma 3):

1. if $\Omega_1(\mathsf{Last}(\tau)) = d$, then set $\mathsf{unlucky} = \bot$ and play $\sigma_{AS}(\tau)$;
2. if $\mathsf{unlucky} = \bot$,
   a. if $|\pi_{\Omega_1 \geq p}(\tau(k_p \triangleright))| < N_{i_p}$ for all odd p, then play $\sigma_{AS}(\tau)$;
   b. else, for each odd p such that $|\pi_{\Omega_1 \geq p}(\tau(k_p \triangleright))| \geq N_{i_p}$,
      i. if the first N_{i_p} elements of $\pi_{\Omega_1 \geq p}(\tau(k_p \triangleright))$ are all p's, then set $\mathsf{unlucky} = \top$;
      ii. increment i_p, and set $k_p = |\tau|$;
3. if $\mathsf{unlucky} = \bot$, then play $\sigma_{AS}(\tau)$;
4. if $\mathsf{unlucky} = \top$,
   a. if $\mathsf{Last}(\tau) \in B$, then play $\sigma_{sub}(\tau')$ where τ' is the longest suffix of τ that is entirely in B;
   b. else $\mathsf{Last}(\tau) \in A$, and play $\sigma_{Attr}(\tau)$.

Note that a play of $\mathcal{G}$ that remains in B is also a play in $\lceil \mathcal{G} \rceil_B$, hence using the strategy σ_{sub} in Condition 4a is well-defined.

We show that σ_1 is sure-almost-sure winning. First show that σ_1 is sure winning for objective $\mathsf{Parity}(\Omega_1)$. Consider an arbitrary outcome ρ of σ_1, let $p_{\max}$ be the largest priority (according to Ω_1) occurring infinitely often in ρ, and consider a position $\hat{k}$ in ρ after which no priority larger than $p_{\max}$ occurs. We show that $p_{\max}$ is even. We proceed by contradiction, assuming that $p_{\max}$ is odd (hence $p_{\max} < d$). Then along ρ, the variable $\mathsf{unlucky}$ must be infinitely often equal to $\top$, since condition 2(b)i holds infinitely often (after position $\hat{k}$, that is, when $k_{p_{\max}} > \hat{k}$). Condition 1 never holds after position $\hat{k}$, Condition 4 holds infinitely often. After position $\hat{k}$, Condition 4b does not hold, because otherwise priority $d > p_{\max}$ would occur. Hence Condition 4a always holds after position $\hat{k}$ and thus the suffix of ρ is an outcome of the sure-almost-sure winning strategy $\sigma_{sub}(\tau')$, hence $p_{\max}$ is even and not odd, which establishes the contradiction. Hence $p_{\max}$ is even and σ_1 is sure winning for objective $\mathsf{Parity}(\Omega_1)$.

Second, we show that σ_1 is almost-sure winning for objective $\mathsf{Parity}(\Omega_2)$. Note that, under the event that always $\mathsf{unlucky} = \bot$ from some point on, the strategy σ_1 plays like σ_{AS}, and thus the parity objective $\mathsf{Parity}(\Omega_2)$ holds with probability 1 thanks to prefix independence of parity objectives. To complete the proof, we show that the event that $\mathsf{unlucky} = \bot$ always holds from some point on has probability 1. By the choice of N_i (see Lemma 3), for all strategies of Player 2 the probability that $\mathsf{unlucky} = \bot$ always holds after round i is at least $\prod_{j \geq i} \left(1 - 2^{-j}\right)$, which tends to 1 as $i \to \infty$ (e.g., see the proof of [5, Lemma 12]).

We now prove the converse. Consider a game $\mathcal{G}$ where all vertices are sure-almost-sure winning for Player 1 for the parity objectives $\mathsf{Parity}(\Omega_1)$ and $\mathsf{Parity}(\Omega_2)$. First, since sure winning implies almost-sure winning, the first condition holds. Second, a sure-almost-sure winning strategy for Player 1 can be played in $\lceil \mathcal{G} \rceil_B$, and either the play always remains in B, or it eventually stays in the winning vertex v_{sink}. In both case, Player 1 is sure-almost-sure winning for the parity objectives, showing that the second condition holds. ◀

The sure-almost-sure spoiling region $V \setminus W$ may not induce a subgame in general, since probabilistic vertices may "leak" in the winning region. Lemma 5 gives a characterisation of the spoiling region in the case that it induces a subgame, which is sufficient for proving the correctness of our algorithm.

If Player 2 has a spoiling strategy (from every initial vertex) in the game obtained from a game $\mathcal{G}$ with an odd largest Ω_1-priority d by removing the positive attractor (for Player 2) to d, then he has a spoiling strategy in the original game $\mathcal{G}$ as well (see also Figure 5). The standard composition a spoiling strategy in the subgame with a (positive) attractor strategy gives a spoiling strategy in the original game, as shown in the proof of Lemma 5.

▶ **Lemma 5.** *Let $\mathcal{G}$ be a stochastic game with sure-almost-sure parity objective induced by the priority functions Ω_1, Ω_2, such that the largest Ω_1-priority d in $\mathcal{G}$ is odd. Let $A = \mathsf{PosAttr}_2^{\mathcal{G}}(V(\Omega_1, d))$ and let $B = V \setminus A$.*

The sure-almost-sure winning region for Player 1 in $\mathcal{G}$ for the parity objectives $\mathsf{Parity}(\Omega_1)$ and $\mathsf{Parity}(\Omega_2)$ is empty if and only if the sure-almost-sure winning region for Player 1 in $\mathcal{G}_{\restriction B}$ for the same objectives is empty.

5 Algorithm

We present an algorithm for the sure-almost-sure synthesis problem that computes the winning region of Player 1 by a recursive procedure in the style of Zielonka's algorithm [30]. The base case is when the game is empty or contains only the sink vertex v_{sink}. Then, two cases are possible:

- if the largest Ω_1-priority d in V is even, the first step is to compute the almost-sure winning region W_{AS} for the conjunction $\mathsf{Parity}(\Omega_1) \cap \mathsf{Parity}(\Omega_2)$ of parity objectives, which is a trap for Player 2 and an over-approximation of the sure-almost-sure winning region. This can be done by expressing the conjunction of parity objectives as a single parity objective over an exponentially larger game graph (Theorem 2), and using standard algorithms for almost-sure parity [11, 12]. In the subgame $\mathcal{G}_{\restriction W_{AS}}$ induced by $V = W_{AS}$, compute the sure attractor A for Player 1 to the vertices with priority d, then solve (recursively) the subgame closure $\lceil \mathcal{G} \rceil_{V \setminus A}$. If all vertices in the subgame closure are winning for Player 1 (line 9), then by the characterisation of Lemma 4 all vertices in $\mathcal{G}$ are also winning for Player 1. Otherwise, some set $B \subseteq V \setminus A$ of vertices in the subgame closure is not winning for Player 1, and by the key property of subgame closures, those vertices are also not winning in $\mathcal{G}$. Removing the positive attractor for Player 2 to B, it remains to solve (recursively) the subgame $\mathcal{G}_{\restriction V \setminus B}$ (line 13).
- if the largest Ω_1-priority d in V is odd, the algorithm proceeds analogously, computing the positive attractor A for Player 2 to the vertices with priority d, then solving (recursively) the subgame $\mathcal{G}_{\restriction V \setminus A}$, using the characterisation in Lemma 5 if no vertex is winning for Player 1 (line 20), or the key property of traps and subgame closures otherwise (line 24).

We prove the correctness of Algorithm 1 by induction on the size of the game, defined as the number $|V \setminus \{v_{\mathsf{sink}}\}|$ of vertices excluding the sink vertex. Consider a stochastic game $\mathcal{G} = \langle V, E, \delta \rangle$, and in the base case where there is no vertices other than v_{sink}, the winning region is V (line 1 of Algorithm 1).

Now let the number of vertices other then v_{sink} be $n + 1$ ($n \geq 0$) and assume Algorithm 1 computes the winning regions in all games of size at most n. Let d be the highest Ω_1-priority assigned to a vertex in V.

First, consider the case that d is even. The algorithm removes the vertices that are not almost-sure winning for the conjunction $\mathsf{Parity}(\Omega_1) \cap \mathsf{Parity}(\Omega_2)$ of parity objectives, as they cannot be sure-almost-sure winning. Then let $V = W_{AS}$ (line 5), let $Z \subseteq V$ be the set of all vertices with priority d, and let $A = \mathsf{SureAttr}_1(Z)$. Since $Z \neq \varnothing$ and $Z \subseteq A$, the set $V \setminus A$ has strictly fewer vertices (other than the sink) than V and by the induction hypothesis we get that W_1' (line 8) is the winning region of Player 1 in the subgame closure $\lceil \mathcal{G} \rceil_{V \setminus A}$) (and $W_2' = V \setminus W_1'$ that of Player 2). We now consider the following two cases.

1. if $W_2' = \varnothing$, then $W_1 = V$ by the characterisation of Lemma 4, that is Player 1 wins from every vertex in $\mathcal{G}$.

Algorithm 1 $\mathsf{solve}(\mathcal{G})$ to compute the sure-almost-sure winning region for parity objectives.

Input: Stochastic two-dimensional parity game given by $\mathcal{G} = (V_1, V_2, E)$ and (Ω_1, Ω_2)
Output: Sure-almost-sure winning region W_1, and losing region W_2, for the parity objectives $\mathsf{Parity}(\Omega_1)$ and $\mathsf{Parity}(\Omega_2)$.

1: **if** $V = \varnothing$ or $V = \{v_{\mathsf{sink}}\}$ **then return** V
2: $d = \max_{v \in V} \Omega_1(v)$
3: **if** d is even **then**
4: $\quad \mathcal{G} = \mathcal{G}_{\restriction W_{AS}}$ ▷ *W_{AS} is the almost-sure winning region for $Parity(\Omega_1) \cap \mathsf{Parity}(\Omega_2)$.*
5: $\quad V = W_{AS}$
6: $\quad Z = \{v \in V \mid \Omega_1(v) = d\}$ ▷ *Z may be empty.*
7: $\quad A = \mathsf{SureAttr}_1(Z)$
8: $\quad W'_1, W'_2 = \mathsf{solve}(\lceil \mathcal{G} \rceil_{V \setminus A})$
9: $\quad$ **if** $W'_2 = \varnothing$ **then**
10: $\quad\quad$ **return** $W_1 = V, W_2 = \varnothing$
11: $\quad$ **else**
12: $\quad\quad B = \mathsf{PosAttr}_2(W'_2)$
13: $\quad\quad W''_1, W''_2 = \mathsf{solve}(\mathcal{G}_{\restriction V \setminus B})$
14: $\quad\quad$ **return** $W_1 = W''_1, W_2 = W''_2 \cup B$
15: **else**
16: $\quad Z = \{v \in V \mid \Omega_1(v) = d\}$ ▷ *Z is nonempty.*
17: $\quad A = \mathsf{PosAttr}_2(Z)$
18: $\quad W'_1, W'_2 = solve(\mathcal{G}_{\restriction V \setminus A})$
19: $\quad$ **if** $W'_1 = \varnothing$ **then**
20: $\quad\quad$ **return** $W_1 = \varnothing, W_2 = V$
21: $\quad$ **else**
22: $\quad\quad B = \mathsf{SureAttr}_1(W'_1)$
23: $\quad\quad W''_1, W''_2 = \mathsf{solve}(\lceil \mathcal{G} \rceil_{V \setminus B})$
24: $\quad\quad$ **return** $W_1 = B \cup W''_1 \setminus \{v_{\mathsf{sink}}\}, W_2 = W''_2$

2. if $W'_2 \neq \varnothing$, then let $B = \mathsf{PosAttr}_2(W'_2)$ in $\mathcal{G}$, and since $v_{\mathsf{sink}} \notin W'_2$ we can apply the induction hypothesis to conclude that W''_1 is the winning region of Player 1 in the subgame $\mathcal{G}_{\restriction V \setminus B}$ (line 13). Since $V \setminus B$ is a trap for Player 2, the key property of traps (see Section 2.2) ensures that W''_1, returned by the algorithm, is also winning for Player 1 in the game $\mathcal{G}$.

We now consider the case that d is odd. Thus the set $Z \subseteq V$ all vertices with priority d is nonempty, and let $A = \mathsf{PosAttr}_2(Z) \neq \varnothing$. By the induction hypothesis, the set W'_1 (line 18) is the winning region of Player 1 in the subgame $\mathcal{G}_{\restriction V \setminus A}$. We now consider the following two cases.

1. if $W'_1 = \varnothing$, then $W_1 = \varnothing$ by the characterisation of Lemma 5, that is Player 1 wins from nowhere in $\mathcal{G}$.
2. if $W'_1 \neq \varnothing$, then let $B = \mathsf{SureAttr}_1(W'_1)$ in $\mathcal{G}$. We can apply the induction hypothesis to conclude that W''_1 is the winning region of Player 1 in the subgame closure $\lceil \mathcal{G} \rceil_{V \setminus B}$ (line 23). To show that $B \cup W''_1$ is the winning region of Player 1 in $\mathcal{G}$, note that W'_1 is a trap for Player 2 in the subgame $\mathcal{G}_{\restriction V \setminus A}$ and that B is a trap for Player 2 in the game $\mathcal{G}$. Consider the strategy of Player 1 defined as follows: (1) in W'_1, play like a sure-almost-sure winning strategy in the subgame $\mathcal{G}_{\restriction V \setminus A}$) (which exists and is also a sure-almost-sure winning strategy in $\mathcal{G}$); (2) in $B \setminus W'_1$, play the (sure) attractor strategy to reach W'_1;

(3) in W_1'', play like a sure-almost-sure winning strategy in the subgame $\mathcal{G}_{\restriction V \setminus A}$) (which is well-defined in the game $\mathcal{G}$). It is immediate that this strategy is sure-almost-sure winning from B and we show that it is also the case from W_1''. To show sure winning for the first parity objective, consider an outcome of the strategy from W_1'', and either the outcome remains in W_1'' and thus satisfies the parity condition by (3), or the outcome eventually leaves W_1'', thus reaching B from where it follows some path that satisfies the parity condition by (1) and (2). Prefix-independence ensures that the outcome from W_1'' also satisfies the parity condition. To show almost-sure winning for the second parity objective, the argument is similar, considering the conditional probability that the play eventually remains in W_1'' or leaves W_1'' and showing that in both cases, the parity objective is satisfied almost-surely.

The structure of Algorithm 1 is similar to Zielonka's algorithm for parity games [30], and so is the complexity analysis. The main difference is that we also need to solve stochastic games with a conjunction of almost-sure parity conditions (line 4). Using the construction in Section 3, and denoting by $\mathsf{ASPG}(N, d)$ the time complexity of solving almost-sure parity games with N vertices and largest priority d, this can be done in time $\mathsf{ASPG}(N, d_1 \cdot d_2)$ where $N = n \cdot \sqrt{\min(d_1^{d_2}, d_2^{d_1})}$ and n is the number of vertices in the game.

Let $T_n^{d_1}$ be the running time of Algorithm 1 on a game with n vertices and largest Ω_1-priority d_1. The running time also depends on the largest Ω_2-priority d_2, but we keep this parameter implicit, as it does not change through the recursive calls.

The crux of the analysis is to see that in the recursive calls $\lceil \mathcal{G} \rceil_{V \setminus A}$ (line 8 and 18), the largest Ω_1-priority in the game $\lceil \mathcal{G} \rceil_{V \setminus A}$ is at most $d_1 - 1$, and in the recursive calls $\mathsf{solve}(\lceil \mathcal{G} \rceil_{V \setminus B})$ (line 13 and 23), the number of vertices in the game $\lceil \mathcal{G} \rceil_{V \setminus B}$ is at most $n - 1$. Hence there is at most n recursive calls for a given largest Ω_1-priority d_1.

Besides the recursive calls and the solution of almost-sure parity games, the other computation can be done in polynomial time (such as computing attractors), thus bounded by $\mathsf{ASPG}(N, d)$. Hence we obtain the condition $T_n^{d_1} \leq n \cdot (T_{n-1}^{d_1} + \mathsf{ASPG}(N, d_1 \cdot d_2))$, which gives the (crude) bound $T_n^{d_1} \in O(n^{d_1} \cdot \mathsf{ASPG}(N, d_1 \cdot d_2))$.

6 Special Cases

We derive from Algorithm 1 several results for subclasses of the sure-almost-sure synthesis problem, giving a refined and more complete view of the problem. The results are presented in Table 1 where the rows (resp., columns) denote the sure (resp., almost-sure) satisfaction for Büchi, coBüchi, parity with fixed number of priorities ($\mathsf{Parity}^{\mathsf{f}}$), and parity condition. For each cell, we give (1) the smallest class of strategies that are sufficient for sure-almost-sure winning, according to the classification types of memoryless, finite-memory, and infinite-memory strategies, and (2) the complexity of solving the decision problem. All results are tight, in particular the complexity bounds: either the lower and upper bound match (completeness result), or the upper bound is the same as in a special case of the problem for which no matching lower bound is known, such as games with a (single) parity objective (with an NP $\cap$ coNP bound).

The new results in Table 1 appear in blue. The other results are either known from previous works, or trivially follow from the new results.

It is known that the general synthesis problem for sure-almost-sure winning is coNP-complete [13] and that infinite memory is necessary for the conjunction of sure Büchi and almost-sure coBüchi, already in MDPs [5]. It is immediate from the running-time analysis of Algorithm 1 that the problem is in P when both d_1 and d_2 are constant. The remaining results of Table 1 are established below.

Table 1 Memory requirement (0 stands for memoryless) and complexity bound for the special cases. New results appear in blue.

AS, ⟶ S, ↓	**Büchi**	**coBüchi**	**Parity^{f}**	**Parity**
Büchi	finite, P	∞ [5], P	∞, P	∞, NP ∩ coNP
coBüchi	0, P	0, P	0, P	0, NP ∩ coNP
Parity^{f}	finite, P	∞, P	∞, P	∞, NP ∩ coNP
Parity	finite, NP ∩ coNP	∞, NP ∩ coNP	∞, NP ∩ coNP	∞, coNP-C [13]

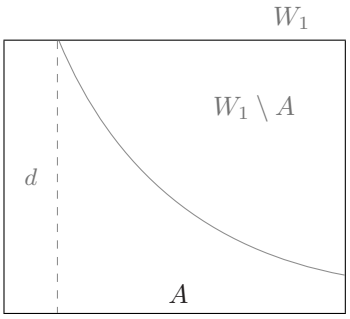

Figure 6 Decomposition in a certificate for winning region W_1 with largest priority even.

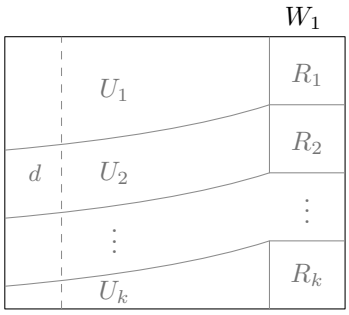

Figure 7 Decomposition in a certificate for winning region W_1 with largest priority odd.

6.1 One of the parity objectives has a fixed number of priorities

We show that the sure-almost-sure synthesis problem can be solved in NP when one of the two parity objectives has a fixed number d of priorities. Formally, an instance of the problem is now a stochastic game $\mathcal{G}$ and two priority functions $\Omega_1 : V \to \mathbb{N}$ and $\Omega_2 : V \to \{0, \dots, d\}$, or vice versa, that is, $\Omega_1 : V \to \{0, \dots, d\}$ and $\Omega_2 : V \to \mathbb{N}$. An NP algorithm is to guess the winning region W_1 of Player 1 and some of the sets that Algorithm 1 would compute when executed on $\lceil \mathcal{G} \rceil_{W_1}$ as a witness and verify that the guess is correct. Executing Algorithm 1 on a game where all vertices are winning sounds useless (we already know that all vertices are winning anyways), but it will be essential in order to define and prove the existence of certain sets (used by Algorithm 1 in intermediate computation) that we use to construct certificates in our NP algorithm.

We now give some intuition of how to define certificates of correctness for a guessed (winning) region W_1. The certificates are defined inductively for W_1 with largest Ω_1-priority d, based on certificates for subsets of W_1 with largest Ω_1-priority at most $d-1$ (following the recursive structure of Algorithm 1). The formal proof shows that the inductive certificates can be checked in polynomial time. First the region W_1 must be a trap for Player 2 (which can be checked in $O(E)$, and ensures that the subgame $\lceil \mathcal{G} \rceil_{W_1}$ is well defined).

1. If the largest Ω_1-priority d in W_1 is even (see Figure 6), the certificate consists of (1) a certificate that the region W_1 is almost-sure winning for the conjunction $\mathsf{Parity}(\Omega_1) \cap \mathsf{Parity}(\Omega_2)$ of parity objectives (such a certificate exists and can be verified in polynomial time by the remark at the end of Section 3), and (2) (inductively) a certificate that the region $W_1 \setminus A$ is sure-almost-sure winning, where $A = \mathsf{SureAttr}_1(\{v \in W_1 \mid \Omega_1(v) = d\})$ (see also lines 6,7 in Algorithm 1). The inductive certificate for $W_1 \setminus A$ exists by the characterisation given in Lemma 4.

2. If the largest Ω_1-priority d in W_1 is odd (see Figure 7), the certificate consists of (1) a partition $U_1, \ldots, U_k$ of W_1 into $k \leq |W_1|$ blocks; (2) nonempty sets $R_1, \ldots, R_k$ such that $R_i \subseteq U_i \setminus \{v \in W_1 \mid \Omega_1(v) = d\}$ is a trap for Player 2 in $\lceil \mathcal{G} \rceil_{W_1 \setminus (U_1 \cup \cdots \cup U_{i-1})}$ that contains no vertex with Ω_1-priority d and such that $U_i = \mathsf{SureAttr}_1(R_i)$ (defined in the subgame $\lceil \mathcal{G} \rceil_{W_1 \setminus (U_1 \cup \cdots \cup U_{i-1})}$), for all $1 \leq i \leq k$; and (3) (inductively) k certificates showing that each region R_i $(i = 1, \ldots, k)$ is sure-almost-sure winning in the game $\lceil \mathcal{G} \rceil_{W_1 \setminus (U_1 \cup \cdots \cup U_{i-1})}$. Here, the certificate corresponds to the sets computed by Algorithm 1 in the subgame $\lceil \mathcal{G} \rceil_{W_1}$ as follows: the set R_1 is the value of W_1' (line 18 of Algorithm 1), the set U_1 is the value of B (in the else-clause at line 22 of Algorithm 1), and the sets R_i and U_i for $i = 2, \ldots, k$ are obtained in a similar in way in the $(i-1)$-th recursive call (at line 23 of Algorithm 1). Note that the set U_1 is well defined because the then-clause at line 20 is never executed when the entire state space $V = W_1 \neq \varnothing$ is sure-almost-sure winning for Player 1, since if W_1' was empty, then W_1 would be empty by the characterisation given in Lemma 5.

▶ **Theorem 6.** *The sure-almost-sure synthesis problem can be solved in NP when one of the two parity objectives has a fixed number d of priorities.*

Now we analyze the amount of memory in winning strategies for relevant special cases of parity objectives.

6.2 No memory for sure coBüchi and almost-sure parity

For the conjunction of sure coBüchi and almost-sure parity objectives, memoryless strategies are sufficient for Player 1. To see this, consider the correctness proof of Algorithm 1 in the case the largest Ω_1-priority d is odd (which is the case for the coBüchi objective). The proof constructs a sure-almost-sure winning strategy in the winning region, defined over the three subregions W_1', $B \setminus W_1'$, W_1''. In W_1', the largest Ω_1-priority is 0 and the sure parity objective is therefore trivially satisfied. Hence an almost-sure winning strategy for the second parity objective is good enough, and thus can be chosen memoryless [12]. In the region $B \setminus W_1'$, a sure attractor strategy is used, which is memoryless. Finally, in W_1'' we can use an argument by induction over the number of vertices (memoryless strategies trivially suffice in empty games) to show that memoryless strategies are sufficient for Player 1.

▶ **Theorem 7.** *Memoryless strategies are sufficient for combined sure coBüchi and almost-sure parity objectives.*

6.3 Finite memory for sure parity and almost-sure Büchi

For the conjunction of sure parity and almost-sure Büchi objectives, we show that finite memory is sufficient for Player 1, namely memory of size $O(n \cdot d_1)$ where n is the number of vertices in the game and d_1 is the largest Ω_1-priority. The proof is by induction on d_1. The case $d_1 = 0$ reduces to just almost-sure parity (the sure parity objective is trivially satisfied), and the case $d_1 = 1$ corresponds to the conjunction of sure coBüchi and almost-sure parity (Büchi) objectives. In both cases, memoryless strategies are sufficient, thus memory $O(1)$ (see Section 6.2 for the latter).

We proceed by induction for $d_1 \geq 2$. First, if d_1 is even, then we construct a sure-almost-sure winning strategy as follows. Using the characterisation of Lemma 4, the winning region W_1 must be almost-sure winning for both the parity and Büchi objectives, and it can be decomposed into the sure attractor A to the vertices with Ω_1-priority d_1, and the complement $B = W_1 \setminus A$ that induces a sure-almost-sure winning subgame closure $\lceil \mathcal{G} \rceil_B$. By

induction hypothesis, there exists a sure-almost-sure winning strategy in $\lceil \mathcal{G} \rceil_B$ with memory $O(n \cdot (d_1 - 1))$. A sure-almost-sure winning strategy in W_1 is defined informally as follows: as long as the current vertex is in B, play according to the sure-almost-sure winning strategy in $\lceil \mathcal{G} \rceil_B$. Whenever the current vertex is in A, play the sure attractor strategy until reaching a vertex with Ω_1-priority d_1, and then play for $n = |W_1|$ rounds a (memoryless) almost-sure winning strategy for the Büchi objective. The memory of the strategy is $n + O(n \cdot (d_1 - 1))$ thus $O(n \cdot d_1)$ and we argue that the strategy is sure-almost-sure winning. To show that the (first) parity objective is satisfied surely, consider an arbitrary outcome of the strategy, and either it visits A infinitely often, and the also Ω_1-priority d_1 infinitely often, or it eventually remains in $B = W_1 \setminus A$ and thus follows a the sure-almost-sure winning strategy in $\lceil \mathcal{G} \rceil_B$. In both case the outcome satisfies the (first) parity objective. To show that the Büchi objective is satisfied almost-surely, notice that, (1) conditional to visiting A infinitely often, there is a bounded probability at least ν^n, where ν is the smallest non-zero probability in the game, to visit a Büchi vertex after each visit to A, and thus probability 1 to visit a Büchi vertex infinitely often; (2) conditional to visiting A finitely often, the play eventually remains in B where the Büchi objective is satisfied with probability 1. Hence in both cases, the Büchi objective is satisfied almost-surely.

Second, if d_1 is odd, then the strategy construction in the correctness proof of Algorithm 1 gives a sure-almost-sure winning strategy with memory $\sum_{i=1}^{k} O(n_i \cdot (d_1 - 1))$ where $\sum_{i=1}^{k} n_i = n$, and thus memory $O(n \cdot d_1)$ is sufficient.

▶ **Theorem 8.** *Finite-memory $O(n \cdot d_1)$ strategies are sufficient for combined sure parity and almost-sure Büchi objectives, where n is the number of vertices and d_1 is the largest priority in the sure parity objective.*

References

1. S. Almagor, O. Kupferman, and Y. Velner. Minimizing expected cost under hard Boolean constraints, with applications to quantitative synthesis. In *Proc. of CONCUR: Concurrency Theory*, volume 59 of *LIPIcs*, pages 9:1–9:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CONCUR.2016.9.
2. C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT, 2008.
3. R. Berthon, S. Guha, and J.-F. Raskin. Mixing probabilistic and non-probabilistic objectives in Markov decision processes. In *Proc. of LICS: Logic in Computer Science*, pages 195–208. ACM, 2020. doi:10.1145/3373718.3394805.
4. R. Berthon, J.-P. Katoen, and T. Winkler. Markov decision processes with sure parity and multiple reachability objectives. In *Proc. of RP: Reachability Problems*, LNCS 15050, pages 203–220. Springer, 2024. doi:10.1007/978-3-031-72621-7_14.
5. R. Berthon, M. Randour, and J.-F. Raskin. Threshold constraints with guarantees for parity objectives in Markov decision processes. In *Proc. of ICALP: International Colloquium on Automata, Languages, and Programming*, volume 80 of *LIPIcs*, pages 121:1–121:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.121.
6. U. Boker. Why these automata types? In *Proc. of LPAR-22.: 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning,*, volume 57 of *EPiC Series in Computing*, pages 143–163. EasyChair, 2018. doi:10.29007/C3BJ.
7. V. Bruyère, E. Filiot, M. Randour, and J.-F. Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. *Inf. Comput.*, 254:259–295, 2017. doi:10.1016/J.IC.2016.10.011.
8. C. S. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan. Deciding parity games in quasipolynomial time. In *Proc. of STOC: Symposium on Theory of Computing*, pages 252–263. ACM, 2017. doi:10.1145/3055399.3055409.

9 K. Chatterjee and L. Doyen. Perfect-information stochastic games with generalized mean-payoff objectives. In *Proc. of LICS: Logic in Computer Science*, pages 247–256. IEEE Computer Society Press, 2016.

10 K. Chatterjee, T. A. Henzinger, and N. Piterman. Generalized parity games. In *Proc. of FOSSACS: Foundations of Software Science and Computational Structures*, LNCS 4423, pages 153–167. Springer, 2007. doi:10.1007/978-3-540-71389-0_12.

11 K. Chatterjee, M. Jurdzinski, and T. A. Henzinger. Simple stochastic parity games. In *Proc. of CSL: Computer Science Logic*, LNCS 2803, pages 100–113. Springer, 2003. doi:10.1007/978-3-540-45220-1_11.

12 K. Chatterjee, M. Jurdzinski, and T. A. Henzinger. Quantitative stochastic parity games. In *Proc. of SODA: Symposium on Discrete Algorithms*, pages 121–130. SIAM, 2004. URL: http://dl.acm.org/citation.cfm?id=982792.982808.

13 K. Chatterjee and N. Piterman. Combinations of qualitative winning for stochastic parity games. In *Proc. of CONCUR: Concurrency Theory*, volume 140 of *LIPIcs*, pages 6:1–6:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.CONCUR.2019.6.

14 T. Chen, V. Forejt, M. Z. Kwiatkowska, A. Simaitis, and C. Wiltsche. On stochastic games with multiple objectives. In *Proc. of MFCS: Mathematical Foundations of Computer Science*, LNCS 8087, pages 266–277. Springer, 2013. doi:10.1007/978-3-642-40313-2_25.

15 T. Colcombet and D. Niwinski. On the positional determinacy of edge-labeled games. *Theor. Comput. Sci.*, 352(1-3):190–196, 2006. doi:10.1016/J.TCS.2005.10.046.

16 L. Doyen and S. Guha. Algorithm and strategy construction for sure-almost-sure stochastic parity games. *arXiv*, 2601.03381, 2026.

17 R. Durrett. *Probability: Theory and Examples*. Cambridge University Press, 2010.

18 E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *Proc. of FOCS: Foundations of Computer Science*, pages 368–377. IEEE Computer Society, 1991. doi:10.1109/SFCS.1991.185392.

19 J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer, 1997.

20 E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, LNCS 2500. Springer, 2002.

21 M. Guo and M. M. Zavlanos. Probabilistic motion planning under temporal tasks and soft constraints. *IEEE Trans. Autom. Control.*, 63(12):4051–4066, 2018. doi:10.1109/TAC.2018.2799561.

22 M. Jurdzinski. Deciding the winner in parity games is in UP ∩ coUP. *Inf. Process. Lett.*, 68(3):119–124, 1998.

23 M. Jurdzinski and R. Lazic. Succinct progress measures for solving parity games. In *Proc. of LICS: Symposium on Logic in Computer Science*, pages 1–9. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005092.

24 K. Lehtinen. A modal μ perspective on solving parity games in quasi-polynomial time. In A. Dawar and E. Grädel, editors, *Proc. of LICS: Symposium on Logic in Computer*, pages 639–648. ACM, 2018. doi:10.1145/3209108.3209115.

25 N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *Log. Methods Comput. Sci.*, 3(3), 2007. doi:10.2168/LMCS-3(3:5)2007.

26 D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *J. Artif. Intell. Res.*, 48:67–113, 2013. doi:10.1613/JAIR.3987.

27 S. Safra. Exponential determinization for omega-automata with strong-fairness acceptance condition (extended abstract). In *Proc. of STOC: Symposium on Theory of Computing*, pages 275–282. ACM, 1992. doi:10.1145/129712.129739.

28 W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, volume 3, Beyond Words, chapter 7, pages 389–455. Springer, 1997. doi:10.1007/978-3-642-59126-6_7.

29 Y. Velner, K. Chatterjee, L. Doyen, T. A. Henzinger, A. Rabinovich, and J.-F. Raskin. The complexity of multi-mean-payoff and multi-energy games. *Information and Computation*, 241:177–196, 2015. doi:10.1016/J.IC.2015.03.001.

30 W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998. doi:10.1016/S0304-3975(98)00009-7.