Formal Methods in System Design (2024) 63:40–80
https://doi.org/10.1007/s10703-023-00411-4

Stochastic games with lexicographic objectives

Krishnendu Chatterjee[1] · **Joost-Pieter Katoen**[2] · **Stefanie Mohr**[3] · **Maximilian Weininger**[3] · **Tobias Winkler**[2]

Received: 3 January 2022 / Accepted: 25 January 2023 / Published online: 8 March 2023

Abstract
We study turn-based stochastic zero-sum games with lexicographic preferences over objectives. Stochastic games are standard models in control, verification, and synthesis of stochastic reactive systems that exhibit both randomness as well as controllable and adversarial non-determinism. Lexicographic order allows one to consider multiple objectives with a strict preference order. To the best of our knowledge, stochastic games with lexicographic objectives have not been studied before. For a mixture of reachability and safety objectives, we show that deterministic lexicographically optimal strategies exist and memory is only required to remember the already satisfied and violated objectives. For a constant number of objectives, we show that the relevant decision problem is in $\mathsf{NP} \cap \mathsf{coNP}$, matching the current known bound for single objectives; and in general the decision problem is PSPACE-hard and can be solved in $\mathsf{NEXPTIME} \cap \mathsf{coNEXPTIME}$. We present an algorithm that computes the lexicographically optimal strategies via a reduction to the computation of optimal strategies in a sequence of single-objectives games. For omega-regular objectives, we restrict our analysis to one-player games, also known as Markov decision processes. We show that lexicographically optimal strategies exist and need either randomization or finite memory. We present an algorithm that solves the relevant decision problem in polynomial time. We have implemented our algorithms and report experimental results on various case studies.

Keywords Probabilistic verification · Stochastic games · Markov decision process · Multiple objectives · Lexicographic preferences · Reachability · Safety · Omega-regular

1 Introduction

Simple stochastic games (SG) [1] are zero-sum turn-based stochastic games played over a finite state space by two adversarial players, Maximizer and Minimizer, along with randomness in the transition function. These games allow the interaction of angelic and demonic non-determinism as well as stochastic uncertainty. They generalize classical models such as *Markov decision processes* (MDP) [2] which have only one player and stochastic uncertainty. An objective specifies the desired set of trajectories of the game, and the goal of the Maximizer is to maximize the probability of satisfying the objective against all choices

✉ Tobias Winkler
tobias.winkler@cs.rwth-aachen.de

Extended author information available on the last page of the article

of the Minimizer. The basic decision problem is to determine whether the Maximizer can ensure the satisfaction of the objective with a given probability threshold. This is among the rare and intriguing problems that are $\mathsf{NP} \cap \mathsf{coNP}$, and whether it belongs to P is a major and long-standing open problem. Besides the theoretical interest, SG are a standard model in control and verification of stochastic reactive systems [2–5], and they serve as robust abstractions of MDP when precise transition probabilities are unknown [6, 7].

Multi-objective optimization problems are relevant in the analysis of systems with multiple – potentially conflicting – goals where trade-offs must be considered. While such trade-off analyses have been extensively studied for MDP with various classes of objectives, see e.g. [2, 8–11], the problem is notoriously hard for SG. In fact, even for multiple reachability objectives, such games are not determined [12] and their decidability is still open.

This work considers SG with multiple objectives equipped with a total lexicographic preference order. That is, if there are, say, two objectives Ω_1 and Ω_2 where Ω_1 comes first in the preference order, then Maximizer should pick a strategy that is (i) optimal for Ω_1, and (ii) optimal for Ω_2 among all strategies that satisfy condition (i). This lexicographic optimization scheme easily generalizes to an arbitrary number of objectives. Lexicographic objectives are useful in many scenarios. For example, an autonomous vehicle might have a primary objective to avoid clashes and a secondary objective to optimize performance; or a robot saving lives during a fire in a building might have a primary objective to save as many lives as possible, and a secondary objective to minimize energy consumption. Thus studying reactive systems with lexicographic objectives is a very relevant problem that has been considered in many different contexts [13, 14]. In particular, non-stochastic games with lexicographic objectives [15, 16] and MDP with lexicographic objectives [17, 18] have been considered, but to the best of our knowledge SG with lexicographic objectives have not been studied.

1.1 Our contribution

The contribution of this paper is twofold.

(I) We present theoretical and practical results for SG with lexicographic reachability and safety objectives. These are the same as in the conference version of this paper [19], but we improved the presentation and included the full proofs. The main contributions are as follows.

- *Determinacy* In contrast to SG with multiple objectives that are not determined, we establish determinacy of SG with a lexicographic combination of reachability and safety objectives.
- *Computational complexity* For the associated decision problem we establish the following: (a) if the number of objectives is constant, then the decision problem lies in $\mathsf{NP} \cap \mathsf{coNP}$, matching the current known bound for SG with a single objective; (b) in general, the decision problem is PSPACE-hard and can be solved in $\mathsf{NEXPTIME} \cap \mathsf{coNEXPTIME}$.
- *Strategy complexity* We show that there exist lexicographically optimal strategies that are deterministic and use finite memory. We also show that memory is only needed in order to remember the already satisfied and violated objectives, i.e., the memory is *arena-independent* [20].

- *Algorithm* We present an algorithm that computes the unique lexicographic value and the witness lexicographically optimal strategies via a reduction to the computation of optimal strategies in a sequence of *single-objectives* games.
- *Experimental results* We have implemented the algorithm and present experimental results on several case studies.

Technical contribution The key idea is that, given the lexicographic order of the objectives, we can consider them sequentially. After every objective, we remove all actions that are not optimal, thereby forcing all following computations to consider only locally optimal actions. The main complication is that local optimality of actions does not imply global optimality when interleaving reachability and safety, as the latter objective can use locally optimal actions to stay in the safe region without reaching the more important target. We introduce quantified reachability objectives as a means to solve this problem.

(II) We present new results for MDP with lexicographic ω-regular objectives. These results extend the conference version [19] of this paper. We discuss the reason for only addressing MDP and not SG in Sect. 4.3, also sketching the possible directions for dealing with the more general model of games, as well as a solution for a subclass of SG. The main contributions are as follows.

- *Algorithm* We present an algorithm that computes the unique lexicographic value and witness lexicographically optimal strategies for MDP with a tuple of lexicographic Streett objectives in polynomial time.
- *Computational complexity* We establish that the associated decision problem is in P. This result relies on the objectives being given as Streett-conditions. If they are in a different form, e.g. LTL formulae, we get the corresponding doubly exponential blow-up of transforming LTL to Streett.
- *Strategy complexity* We show that there exist lexicographically optimal strategies. They need either randomization or finite memory, but only in a limited part of the state space, the so-called *end components*. In the rest of the MDP, memoryless deterministic strategies suffice.
- *Experimental results* We have implemented the algorithm and present experimental results on several case studies, comparing our deterministic algorithm to the one from [18], which is based on reinforcement learning. Surprisingly, our algorithm often outperforms the learning-based approach.

Technical contribution The key problem is that satisfaction of general ω-regular objectives depends on the game's infinite runs. This is different from reachability (and safety) where satisfaction (violation, resp.) can be witnessed after a finite number of steps. Hence, we have to analyze the part of the state space where a play can remain forever, i.e., the end components. In MDP, we can compute the lexicographic value vector for all states in end components and then reduce the problem to reachability. In SG, the analysis of the end components is more complicated and we leave it for future work. A more detailed discussion of the complications and the possible solutions is in Sect. 4.3.

1.2 Related work

We discuss related works on (a) MDP with multiple objectives, (b) SG with multiple objectives, (c) lexicographic objectives in related models, and (d) existing tool support.

(a) MDP with multiple objectives have been widely studied over a long time [2, 8]. In the context of verifying MDP with multiple objectives, both qualitative objectives such as reachability and LTL [21], as well as quantitative objectives, such as mean payoff [9, 22], discounted sum [23], total reward [24] and combinations thereof [11, 25] have been considered. Besides multiple objectives with expectation criteria, other criteria have also been considered, e.g. combinations with variance [26], or multiple percentile (threshold) queries [22, 27–29]. Practical applications of MDP with multiple objectives are described in [30–32].

(b) More recently, SG with multiple objectives have been considered, but the results are more limited [33]. Multiple mean-payoff objectives were first examined in [34] and the qualitative problems are coNP-complete [35]. Some special classes of SG (namely stopping SG) have been solved for total-reward objectives [12, 36] and applied to autonomous driving [37]. However, in the general case, decidability of SG with multiple objective remains open, even in the simple case of reachability objectives. On the positive side, it is known that the Pareto curve of achievable thresholds can be approximated to arbitrary precision [38].

(c) The study of lexicographic objectives has been considered in many different contexts [13, 14]. Non-stochastic games with lexicographic mean-payoff objectives and parity conditions have been studied in [15] for the synthesis of reactive systems with performance guarantees. Non-stochastic games with multiple ω-regular objectives equipped with a monotonic preorder, which subsumes lexicographic order, have been studied in [39]. Moreover, the beyond worst-case analysis problems studied in [40] also considers primary and secondary objectives, which has a lexicographic flavor. MDP with lexicographic discounted-sum objectives have been studied in [17], and have been extended with partial-observability in [41]. Very recently, a reinforcement learning-based algorithm for MDP with lexicographic ω-regular objectives was presented in [18]. This solution relies on the choice of suitable parameters for the learning and, in the absence of mechanisms to guess them, does not provide a formal guarantee about the correctness of the result. In contrast, our algorithm is deterministic and guarantees to find the lexicographic value and optimal strategies.

(d) PRISM-Games [42] provides tool support for several multi-player multi- objective settings. MultiGain [43] is limited to generalized mean-payoff MDP. Storm [44] can, among numerous single-objective problems, solve Markov automata with multiple timed reachability or expected cost objectives [45], multi-cost bounded reachability MDP [46], and it can provide simple strategies for multiple expected reward objectives in MDP [10]. The recent tool TEMPEST [47] includes a solver for SG with safety and mean-payoff objectives. It can also solve multiple mean-payoff objectives by mixing them into a single one.

1.3 Organization of this paper

After recalling preliminaries and defining the problem in Section 2, we first, in Sect. 3.1, consider SG with reachability and safety objectives with the following restriction: all target sets are absorbing. Then, in Sect. 3.2 we extend our insights to general games, yielding the full algorithm and the theoretical results. Section 4 considers the lexicographic ω-regular objectives for MDP. After providing a solution for end components in Sect. 4.1, the general algorithm is given in Sect. 4.2. In Sect. 4.3 we discuss the complications for SG. Finally, Section 5 describes the prototypical implementation and experimental evaluation of both algorithms, and Sect. 6 concludes.

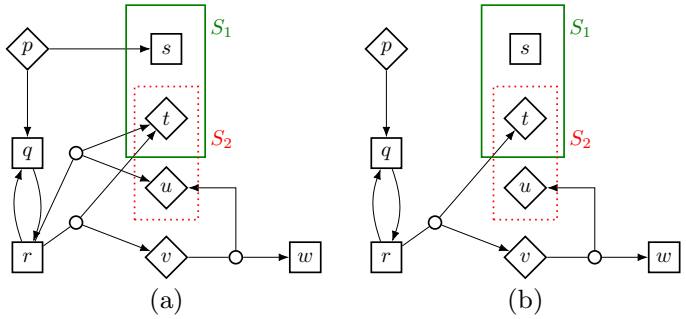


Fig. 1 **a** An example of a stochastic game (action labels omitted). Max-states are rendered as squares □ and Min-states as rhombs ◇. Probabilistic choices correspond to small circles. All probabilities are equal to 1/2. The absorbing lex-objective $\mathbf{\Omega} = \{\mathsf{Reach}(S_1), \mathsf{Safe}(S_2)\}$ is indicated by the thick green line around $S_1 = \{s, t\}$ and the dotted red line around $S_2 = \{t, u\}$. Self-loops in sinks are omitted. **b** Restriction of the game to lex-optimal actions only

2 Preliminaries

We fix some general notation first. A *probability distribution* on a finite set A is a function $f : A \to [0, 1]$ such that $\sum_{x \in A} f(x) = 1$. We denote the set of all probability distributions on A by $\mathcal{D}(A)$. A distribution f is *Dirac* if $f(a) = 1$ for some $a \in A$. Vectors $\mathbf{x} \in B^n$ where B is an arbitrary set are denoted in a bold font. For $1 \leq i \leq n$ we write $\mathbf{x}_i$ to refer to the i-th component of $\mathbf{x}$. Moreover, we use $\mathbf{x}_{<i}$ to denote the (possibly empty) vector $(\mathbf{x}_1, \ldots, \mathbf{x}_{i-1})$.

2.1 Games, strategies, and basic objectives

In this section, we formally introduce the stochastic models used in this paper, most importantly *stochastic games* and *Markov decision processes*. We also define strategies and basic objectives.

2.1.1 Stochastic games and Markov decision processes

In this paper, we consider *(simple) stochastic games* [1], which are defined as follows (see Fig. 1 on page 1 for examples):

Definition 1 (SG: Stochastic game) An SG is a tuple $\mathcal{G} = (S_\Box, S_\Diamond, L, \mathsf{Act}, P)$ with $S := S_\Box \uplus S_\Diamond \neq \emptyset$ a finite set of states, L a finite set of action labels, $\mathsf{Act} : S \to 2^L \setminus \{\emptyset\}$ a set of actions available at every state, and $P : S \times L \dashrightarrow \mathcal{D}(S)$ a (partial) probabilistic transition function. $P(s, a)$ is defined for $s \in S$ and $a \in L$ iff $a \in \mathsf{Act}(s)$.

We write $P(s, a, s')$ instead of $P(s, a)(s')$ for all $s, s' \in S$, $a \in \mathsf{Act}(s)$. A state $s \in S$ is called *absorbing* (or sink) if $P(s, a, s) = 1$ for all $a \in \mathsf{Act}(s)$. $\mathsf{Sinks}(\mathcal{G})$ denotes the set of all absorbing states of $\mathcal{G}$. We refer to the two players of the game as Max and Min, and the sets $S_\Box$ and $S_\Diamond$ as Max- and Min-states, respectively. As the game is *turn-based*, these sets partition the state space S: in each state, it is either Max's or Min's turn.

Intuitively, the semantics of an SG is as follows: In every turn, the corresponding player picks one of the finitely many available actions $a \in \mathsf{Act}(s)$ in the current state s. The game

then transitions to the next state according to the probability distribution $P(s, a)$. The winning conditions and the initial state are not part of the game and need to be specified externally.

We also consider *Markov decision processes* in this paper:

Definition 2 (MDP: Markov decision process) An MDP is the special case of an SG where either $S_{\Diamond} = \emptyset$ or $S_{\Box} = \emptyset$. In other words, an MDP is a *1-player SG*.

2.1.2 Strategies

We define the formal semantics of SG (and MDP) by means of paths and strategies. Let $\mathcal{G} = (S_{\Box}, S_{\Diamond}, L, \mathsf{Act}, P)$ be an SG. An *infinite path* π is an infinite sequence $\pi = s_0 a_0 s_1 a_1 \cdots \in (S \times L)^{\omega}$, such that for every $i \geq 0$, we have $a_i \in \mathsf{Act}(s_i)$ and $s_{i+1} \in \{s' \mid P(s_i, a_i, s') > 0\}$. *Finite* paths are defined analogously as elements of $(S \times L)^* \times S$. Omitting the action labels in a path yields a sequence of states. We call such state sequences *trajectories*.

A *strategy* of player Max is a function $\sigma : (S \times L)^* \times S_{\Box} \to \mathcal{D}(L)$ where $\sigma(\pi s)(s') > 0$ only if $s \in \mathsf{Act}(s)$. The strategy σ is *memoryless* if $\sigma(\pi s) = \sigma(\pi' s)$ for all $\pi, \pi' \in (S \times L)^*$ and $s \in S_{\Box}$. More generally, σ has memory of class-size at most m if the set $(S \times L)^*$ can be partitioned into m classes $M_1, \ldots, M_m \subseteq (S \times L)^*$ such that $\sigma(\pi s) = \sigma(\pi' s)$ for all $1 \leq i \leq m$, $\pi, \pi' \in M_i$ and $s \in S_{\Box}$. A memory of class-size m can be represented with $\lceil \log m \rceil$ bits.

A strategy σ of Max is *deterministic* if $\sigma(\pi s)$ is Dirac for all πs. Strategies that are both memoryless and deterministic are called *MD* and can be identified as functions $\sigma : S_{\Box} \to L$. Note that there are at most $|L|^{|S_{\Box}|}$ different MD strategies, i.e., exponentially many in $S_{\Box}$. MD strategies play an important role in this paper because they are sufficient in order to play optimally w.r.t. to many basic objectives, see below.

Strategies τ of player Min are defined analogously, with $S_{\Box}$ replaced by $S_{\Diamond}$ in the above definitions. The set of all strategies of player Max is denoted with Σ_{Max}, and its set of all MD strategies with $\Sigma^{\mathsf{MD}}_{\mathsf{Max}}$. Similarly, we use the notation Σ_{Min} and $\Sigma^{\mathsf{MD}}_{\mathsf{Min}}$ for the corresponding strategy sets of player Min.

2.1.3 Markov chains and probability measures

A *Markov chain* (MC) is the special case of an SG with $|\mathsf{Act}(s)| = 1$ for all game states s. Fixing strategies σ, τ of both players in an SG $\mathcal{G}$ yields the *induced* MC $\mathcal{G}^{\sigma,\tau}$. Formally, $\mathcal{G}^{\sigma,\tau}$ has infinitely many states $S^{\sigma,\tau} = (S \times L)^* \times S$ and the transition probabilities are given as

$$P^{\sigma,\tau}(\pi s, \pi s a s') = \begin{cases} \sum_{a \in \mathsf{Act}(s)} \sigma(\pi s)(a) \cdot P(s, a, s') & \text{if } s \in S_{\Box}, \\ \sum_{a \in \mathsf{Act}(s)} \tau(\pi s)(a) \cdot P(s, a, s') & \text{if } s \in S_{\Diamond}, \end{cases}$$

for all $\pi \in (S \times L)^*$, states $s, s' \in S$ and $a \in \mathsf{Act}(s)$. Note that the term $\sigma(\pi s)(a)$ is the probability assigned by Max to action a when the game reaches state s with history π. It is also possible to fix just one player's strategy σ which results in the *induced MDP* $\mathcal{G}^{\sigma}$.

Even though induced Markov chains (and MDP) are generally infinite, it is well-known that if the strategies σ, τ are finite-memory, then $\mathcal{G}^{\sigma,\tau}$ is equivalent (more precisely: bisimilar) to a *finite* MC. In particular, if the strategies are MD, then the induced MC is simply obtained from keeping only the actions selected by the MD strategies at each state.

Unlike SG and MDP, Markov chains are purely probabilistic systems, and it is in fact possible to define a probability measure over certain *events* that may occur in an MC.

Ultimately, it is this probability measure that assigns meaning to statements like "Max can win the game with probability 1/2". The construction of such a probability measure is standard, see e.g. [3, Ch. 10]. Given a Markov chain with (countable) state space S, we define the *cylinder set* of a finite trajectory $\pi \in S^*$ as $\pi S^\omega = \{\pi\pi' \mid \pi' \in S^\omega\}$. We obtain a *sigma-algebra* $(S^\omega, \mathcal{F})$ where $\mathcal{F} \subseteq 2^{S^\omega}$ is the smallest set that contains all the cylinder sets and that is closed under complement and countable union. The sets in $\mathcal{F}$ are called *measurable events*. Each measurable event $E \in \mathcal{F}$ can be assigned a probability $\mathbb{P}_{s_0}(E) \in [0,1]$ given some MC state $s_0 \in S$. This *probability measure* is constructed as follows: First, the probability of a cylinder set is defined as

$$\mathbb{P}_{s_0}(\pi S^\omega) = \begin{cases} 1 & \text{if } \pi \text{ is the empty trajectory,} \\ 0 & \text{if } \pi = s\pi' \text{ for some } s \neq s_0, \pi' \in S^*, \\ \prod_{i=0}^{|\pi|-1} P(\pi_i, \pi_{i+1}) & \text{otherwise,} \end{cases}$$

where $|\pi|$ is the length of a finite trajectory. Second, the probabilities of countably many *pairwise disjoint* events $(E_i)_{i\geq 0}$ satisfy

$$\mathbb{P}_{s_0}\left(\bigcup_{i\geq 0} E_i\right) \quad = \quad \sum_{i\geq 0} \mathbb{P}_{s_0}(E_i)\,.$$

It turns out that these axioms induce a unique probability measure $\mathbb{P}_{s_0} : \mathcal{F} \to [0,1]$. The resulting structure $(S^\omega, \mathcal{F}, \mathbb{P}_{s_0})$ is a *probability space*. The probability measure of an induced Markov chain $\mathcal{G}^{\sigma,\tau}$ with initial state s_0 is denoted $\mathbb{P}^{\sigma,\tau}_{s_0}$.

2.1.4 Reachability and safety objectives

In our setting, an *objective* is a measurable event $\Omega \subseteq S^\omega$ of infinite trajectories in an SG $\mathcal{G}$. Note that on the level of an induced MC $\mathcal{G}^{\sigma,\tau}$, where σ, τ are arbitrary strategies, the event Ω must be identified with the (measurable) event

$$\Omega' = \{\pi_1 s_1, \pi_2 s_2, \ldots \in (S^{\sigma,\tau})^\omega \mid s_1 s_2 \ldots \in \Omega,\ \forall i \geq 0 : P^{\sigma,\tau}(\pi_i s_i, \pi_{i+1} s_{i+1}) > 0\}\,,$$

i.e., the trajectories in Ω' are like the ones in Ω but each state also carries a possible history. To simplify the notation, we do not distinguish between Ω and Ω'.

The *reachability objective* $\mathsf{Reach}(T)$ with *target set* $T \subseteq S$ is the objective

$$\mathsf{Reach}(T) \quad = \quad \{\, s_0 s_1 \ldots \in S^\omega \mid \exists\, i \geq 0 \;:\; s_i \in T \,\}\,.$$

The set $\mathsf{Safe}(T) = S^\omega \setminus \mathsf{Reach}(T)$ is called a *safety objective*; alternatively,

$$\mathsf{Safe}(T) \quad = \quad \{\, s_0 s_1 \ldots \in S^\omega \mid \forall\, i \geq 0 \;:\; s_i \notin T \,\}\,.$$

In other words, a safety objective consists of avoiding the *unsafe set* T forever (we remark that some other authors specify safety objectives in terms of a *safe* set that should never be left). Further, for sets $T_1, T_2 \subseteq S$ we define the *until objective*

$$T_1 \,\mathsf{U}\, T_2 \quad = \quad \{\, s_0 s_1 \ldots \in S^\omega \mid \exists\, i \geq 0 \;:\; s_i \in T_2 \;\wedge\; \forall j < i : s_j \in T_1 \,\}\,.$$

Reachability, safety, and until objectives are among the simplest examples of measurable events [3]. A reachability or safety objective where the set T satisfies $T \subseteq \mathsf{Sinks}(\mathcal{G})$

is called *absorbing*. For the safety probabilities in an (induced) MC, it holds that $\mathbb{P}_{s_0}(\mathsf{Safe}(T)) = 1 - \mathbb{P}_{s_0}(\mathsf{Reach}(T))$.

2.1.5 ω-regular objectives and streett conditions

Reachability, safety, and until objectives as discussed above are all special cases of *ω-regular objectives*. In general, the class of ω-regular objectives can be characterized as follows: An objective $\Omega \subseteq S^\omega$ is ω-regular iff there exists a *deterministic Streett*[1] *automaton* (DSA) with input alphabet S that accepts exactly the trajectories in Ω. In order to reason about optimal strategies for Ω, one can construct the automata-theoretic product of the game with the DSA defining Ω. This product is an SG with a *Streett objective*, and optimal strategies for Ω can be found by optimizing the probability to satisfy the Streett condition in the product. More formally, DSA and products of DSA and SG are defined as follows:

Definition 3 (DSA and product contruction) A *deterministic Streett automaton* (DSA) is a finite automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, (\mathsf{F}_j, \mathsf{I}_j)_{1 \le j \le m})$, where $Q \neq \emptyset$ is a finite set of control states, $\Sigma \neq \emptyset$ is a finite input alphabet, $\delta : Q \times \Sigma \to Q$ is a deterministic transition function, $q_0 \in Q$ is an initial state, and for all $1 \le j \le m$, $(\mathsf{F}_j, \mathsf{I}_j) \subseteq Q \times Q$ is a tuple of states called *Streett pair*.

Consider an SG $\mathcal{G} = (S_\Box, S_\Diamond, L, \mathsf{Act}, P)$ and $\mathcal{A} = (Q, S, \delta, q_0, (\mathsf{F}_j, \mathsf{I}_j)_{1 \le j \le m})$.

We define the *product* of $\mathcal{G}$ and $\mathcal{A}$ as the SG $\mathcal{G} \times \mathcal{A} = (S_\Box \times Q, S_\Diamond \times Q, L, \mathsf{Act}', P')$ where for all $s \in S$ and $q \in Q$, $\mathsf{Act}'(s,q) = \mathsf{Act}(s)$, and $P' : (S \times Q) \times L \dashrightarrow \mathcal{D}(S \times Q)$ such that for all $a \in \mathsf{Act}(s)$, $P'((s,q),a)$ is a distribution $d_{s,q,a}$ over $S \times Q$ with $d_{s,q,a}(s',q') = P(s,a,s')$ if $q' = \delta(q,s)$, and $d_{s,q,a}(s',q') = 0$ otherwise.

The tuples $(S \times \mathsf{F}_j, S \times \mathsf{I}_j)_{1 \le j \le m}$ are a *Streett condition* for $\mathcal{G} \times \mathcal{A}$.

Such product constructions are standard in probabilistic model checking, see e.g. [3, Sec. 10.6.4] for further details. In the rest of the paper we assume that we are already given a (product) SG equipped with a Streett condition when considering ω-regular objectives, i.e., we do not consider the product explicitly.

Let $\mathcal{G}$ be an SG with state space S. The *semantics* of a *Streett condition* $(\mathsf{F}_j, \mathsf{I}_j)_{1 \le j \le m}$, $\forall 1 \le j \le m : \mathsf{F}_j, \mathsf{I}_j \subseteq S$, for $\mathcal{G}$ is defined in terms of the objective $\Omega = \mathsf{Streett}((\mathsf{F}_j, \mathsf{I}_j)_{1 \le j \le m})$. An infinite path $s_0 s_1 \ldots \in S^\omega$ is contained in Ω iff

$$\forall\, 1 \le j \le m \;:\; \{\, i \ge 0 \mid s_i \in \mathsf{F}_j \,\} \text{ is finite} \quad \text{or} \quad \{\, i \ge 0 \mid s_i \in \mathsf{I}_j \,\} \text{ is infinite} .$$

It is well-known that ω-regular sets, and thus, in particular, Streett objectives, are measurable events, see e.g. [3, Sec. 10.3].

2.2 Lexicographic objectives

The *lexicographic* order on $\mathbb{R}^n$ is defined as $\mathbf{x} \le_{\mathsf{lex}} \mathbf{y}$ iff $\mathbf{x}_i \le \mathbf{y}_i$ where $i \le n$ is the greatest position such that for all $j < i$ it holds that $\mathbf{x}_j = \mathbf{y}_j$. The position i is called *tiebreaker*. Notice that for arbitrary sets $X \subseteq [0,1]^n$, suprema and infima exist in the lexicographic order.

[1] *For technical reasons (discussed in Sect. 4.1), we prefer Streett over the equally expressive Rabin, Parity, or Muller conditions.*

Definition 4 (Lex-objectives and lex-values) Let $\mathcal{G}$ be an SG with state space S. A *lex(-icographic) objective* for $\mathcal{G}$ is a vector $\mathbf{\Omega} = (\Omega_1, \dots, \Omega_n)$ where $\Omega_i \subseteq S^\omega$ is an objective in $\mathcal{G}$ for all $1 \leq i \leq n$. For all $s \in S$, the *lex(-icographic) value function* ${}^{\mathbf{\Omega}}\mathbf{v}^{\mathsf{lex}} : S \to [0,1]^n$ is defined as:

$$ {}^{\mathbf{\Omega}}\mathbf{v}^{\mathsf{lex}}(s) \quad = \quad \sup_{\sigma \in \Sigma_{\mathsf{Max}}} \inf_{\tau \in \Sigma_{\mathsf{Min}}} \mathbb{P}_s^{\sigma,\tau}(\mathbf{\Omega}) \tag{1} $$

where $\mathbb{P}_s^{\sigma,\tau}(\mathbf{\Omega})$ denotes the *vector* $(\mathbb{P}_s^{\sigma,\tau}(\Omega_1), \dots, \mathbb{P}_s^{\sigma,\tau}(\Omega_n)) \in [0,1]^n$ and the suprema and infima are taken with respect to the order $\leq_{\mathsf{lex}}$ on $[0,1]^n$.

Thus the lex-value at state s is the lexicographically supremal vector of probabilities that Max can ensure against all possible behaviors of Min when the game starts in s. We prove in Sect. 3.2.3 that the supremum and infimum in (1) can be exchanged in the case of (possibly) mixed reachability and safety objectives; this property is called *determinacy*. We omit the superscript $\mathbf{\Omega}$ in ${}^{\mathbf{\Omega}}\mathbf{v}^{\mathsf{lex}}$ if it is clear from the context.

Example 1 (SG and lex-values) Consider the SG sketched in Fig. 1a with the lex-objective $\mathbf{\Omega} = \{\mathsf{Reach}(S_1), \mathsf{Safe}(S_2)\}$. Player Max must thus maximize the probability to reach S_1 and, moreover, among all optimal strategies for $\mathsf{Reach}(S_1)$, it must choose one that maximizes the probability to avoid S_2 forever.

2.2.1 Lex-value of actions and lex-optimal actions

We extend the notion of value to actions. Let $s \in S$. The *lex-value of an action* $a \in \mathsf{Act}(s)$ is defined as $\mathbf{v}^{\mathsf{lex}}(s,a) = \sum_{s'} P(s,a,s') \cdot \mathbf{v}^{\mathsf{lex}}(s')$. If $s \in S_{\Box}$, then action a is called *lex-optimal* if $\mathbf{v}^{\mathsf{lex}}(s,a) = \max_{b \in \mathsf{Act}(s)} \mathbf{v}^{\mathsf{lex}}(s,b)$. Similarly, if $s \in S_{\Diamond}$, then a is called lex-optimal if $\mathbf{v}^{\mathsf{lex}}(s,a) = \min_{b \in \mathsf{Act}(s)} \mathbf{v}^{\mathsf{lex}}(s,b)$. There exists at least one lex-optimal action because $\mathsf{Act}(s)$ is finite by definition.

Example 2 (Lex-values of actions) We now intuitively explain the lex-values of all states in Fig. 1a. The lex-value of sink states s, t, u and w is determined by their membership in the sets S_1 and S_2. E.g., $\mathbf{v}^{\mathsf{lex}}(s) = (1,1)$, as it is part of the set S_1 that should be reached and not part of the set S_2 that should be avoided. Similarly we get the lex-values of t, u and w as $(1,0)$, $(0,0)$ and $(0,1)$ respectively. State v has a single action that yields $(0,0)$ or $(0,1)$ each with probability $1/2$, thus $\mathbf{v}^{\mathsf{lex}}(v) = (0, 1/2)$.

State p has one action going to s, which would yield $(1,1)$. However, as p is a Min-state, its best strategy is to avoid giving such a high value. Thus, it uses the action going downwards and $\mathbf{v}^{\mathsf{lex}}(p) = \mathbf{v}^{\mathsf{lex}}(q)$. State q only has a single action going to r, so $\mathbf{v}^{\mathsf{lex}}(q) = \mathbf{v}^{\mathsf{lex}}(r)$.

State r has three choices: (i) Going back to q, which results in an infinite loop between q and r, and thus never reaches S_1. So a strategy that commits to this action will not achieve the optimal value. (ii) Going to t or u each with probability $1/2$. In this case, the safety objective is definitely violated, but the reachability objective is achieved with $1/2$. (iii) Going to t or v each with probability $1/2$. Similarly to (ii), the probability to reach S_1 is $1/2$, but additionally, there is a $1/2 \cdot 1/2$ chance to avoid S_2. Thus, since r is a Max-state, its lex-optimal choice is the action leading to t or v and we get $\mathbf{v}^{\mathsf{lex}}(r) = (1/2, 1/4)$.

2.2.2 Lex-optimal strategies

Definition 5 (Lex-optimal strategies) A strategy $\sigma \in \Sigma_{\mathsf{Max}}$ is *lex-optimal* for $\mathbf{\Omega}$ if for all $s \in S$, $\mathbf{v}^{\mathsf{lex}}(s) = \inf_{\tau' \in \Sigma_{\mathsf{Min}}} \mathbb{P}_s^{\sigma,\tau'}(\mathbf{\Omega})$. A strategy $\tau \in \Sigma_{\mathsf{Min}}$ is a *lex-optimal counter-strategy* against σ if $\mathbb{P}_s^{\sigma,\tau}(\mathbf{\Omega}) = \inf_{\tau' \in \Sigma_{\mathsf{Min}}} \mathbb{P}_s^{\sigma,\tau'}(\mathbf{\Omega})$.

Further, a strategy σ of Max (Min, resp.) is called *locally lex-optimal* if for all $\pi \in (S \times L)^*$, $s \in S_\Box$ ($s \in S_\Diamond$, resp.) and $a \in \mathsf{Act}(s)$, we have $\sigma(\pi s)(a) > 0$ implies that action a is lex-optimal. Thus, locally lex-optimal strategies only assign positive probability to lex-optimal actions. Locally lex-optimal strategies are not necessarily (globally) lex-optimal, see Example 4.

We stress that in general, counter-strategies of Min may depend on the strategy chosen by Max; this is because of the quantification order "sup inf".

3 Lexicographic 2-player stochastic games

In this section, we derive properties and algorithms for SG with lexicographic reachability and safety objectives. Formally, a *lexicographic reachability-safety objective* (*reach-safe lex-objective*, for short) for a game $\mathcal{G}$ with state space S is a vector $\mathbf{\Omega} = (\Omega_1, \ldots, \Omega_n)$ such that $\Omega_i \in \{\mathsf{Reach}(S_i), \mathsf{Safe}(S_i)\}$ with $S_i \subseteq S$ for all $1 \leq i \leq n$. Note that arbitrary alternations of reachability and safety objectives are allowed. We call $\mathbf{\Omega}$ *absorbing* if $S_i \subseteq \mathsf{Sinks}(\mathcal{G})$ for all $1 \leq i \leq n$. Intuitively, games with absorbing objectives terminate once a single reachability or safety objective in $\mathbf{\Omega}$ has been satisfied or violated, respectively, which somewhat simplifies the analysis.

The rest of this section is structured as follows. In Sect. 3.1, we treat absorbing reach-safe lex-objectives, and we reduce the non-absorbing case to the absorbing setting in Sect. 3.2.

3.1 Lexicographic reach-safe SG with absorbing targets

This section deals with computing the values and optimal strategies of SG with absorbing reach-safe lex-objectives. In Sect. 3.1.1, we prove a structural result (Theorem 1) about optimal strategies which implies in particular that MD optimal strategies exist (Theorem 2). The subsequent Sect. 3.1.2 presents our algorithm. The main technical difficulty arises from interleaving reachability and safety objectives.

3.1.1 Characterizing optimal strategies

This first subsection derives a characterization of lex-optimal strategies in terms of local optimality and an additional reachability condition (Theorem 1 further below). It is one of the key ingredients for the correctness of the algorithm presented later and also gives rise to a (non-constructive) proof of the existence of MD lex-optimal strategies in the absorbing case.

We begin with the following lemma that summarizes some straightforward facts that we will frequently use. Recall that a strategy is *locally lex-optimal* if it only selects actions with optimal lex-value.

Lemma 1 *The following statements hold for every SG $\mathcal{G}$ with absorbing reach-safe lex-objective $\mathbf{\Omega}$:*

(a) *If $\sigma \in \Sigma^{\mathsf{MD}}_{\mathsf{Max}}$ is lex-optimal and $\tau \in \Sigma^{\mathsf{MD}}_{\mathsf{Min}}$ is a lex-optimal counter strategy against σ, then σ and τ are both locally lex-optimal. We do not (yet) claim that MD optimal strategies σ, τ exist in general.*
(b) *Let $\widetilde{\mathcal{G}}$ be the subgame obtained by removing all actions (of both players) that are not locally lex-optimal in $\mathcal{G}$. Let $\widetilde{\mathbf{v}}^{\mathsf{lex}}$ be the lex-values in $\widetilde{\mathcal{G}}$. Then $\widetilde{\mathbf{v}}^{\mathsf{lex}} = \mathbf{v}^{\mathsf{lex}}$.*

Proof Both claims follow from the definitions of lex-value and lex-optimal strategy. For (b) in particular, a strategy using actions that are not lex-optimal can be transformed into a strategy that achieves a greater (lower, resp.) value. Thus removing the non-lex-optimal actions does not affect the lex-value. □

Example 3 (Modified game $\widetilde{\mathcal{G}}$) Consider again the SG from Fig. 1a. Recall the lex-values from Example 2. Now we remove the actions that are not locally lex-optimal. This means we drop the action that leads from p to s and the action that leads from r to t or u (Fig. 1b). Since these actions were not used by the lex-optimal strategies, the value in the modified SG is the same as that of the original game.

Example 4 (Locally lex-optimal does not imply globally lex-optimal) Note that in the subgame in Fig. 1, we do not drop the action that leads from r to q, because $\mathbf{v}^{\mathsf{lex}}(r) = \mathbf{v}^{\mathsf{lex}}(q)$, so this action is locally lex-optimal. In fact, a lex-optimal strategy for Max can use it arbitrarily many times without reducing the lex-value, as long as it eventually picks the action leading from r to t or v. However, if Max only played the action leading to q, the lex-value would be reduced to $(0, 1)$ as we would not reach S_1, but would also avoid S_2.

We stress the following consequence of this: Playing a locally lex-optimal strategy is not necessarily globally lex-optimal. It is not sufficient to just restrict the game to locally lex-optimal actions of the previous objectives and then solve the current one. Note that in fact the optimal strategy for the second objective $\mathsf{Safe}(S_2)$ would be to remain in $\{q, r\}$; however, Max must not pick this safety strategy before it has not "tried everything" for all previous reachability objectives, in this case reaching S_1.

The final set

This idea of "trying everything" for an objective $\mathsf{Reach}(S_i)$ is equivalent to the following: *either reach the target set S_i, or reach a set of states from which S_i cannot be reached anymore*. Formally, let $\mathsf{Zero}_i = \{s \in S \mid \mathbf{v}_i^{\mathsf{lex}}(s) = 0\}$ be the set of states where Max cannot enforce reaching S_i with positive probability if Max plays optimally w.r.t. to more important targets $j < i$. Note that Zero_i indeed depends on the lex-value, and not on the single-objective value of reaching S_i. This is important as the single-objective value could be greater than 0, but a more important objective has to be sacrificed to achieve it.

We define the set of states where we have "tried everything" for all reachability objectives as follows:

Definition 6 (Final set) For absorbing $\mathbf{\Omega} = (\Omega_1, \ldots, \Omega_n)$ and $1 \leq i \leq n{+}1$, let $R_{<i} = \{j < i \mid \Omega_j = \mathsf{Reach}(S_j)\}$. We define the *Final Set*

$$F_{<i} \quad = \quad \bigcup_{k \in R_{<i}} S_k \;\cup\; \bigcap_{k \in R_{<i}} \mathsf{Zero}_k$$

with the convention that $F_{<i} = S$ if $R_{<i} = \emptyset$. We also let $F = F_{<n+1}$.

In other words, the Final Set F contains all target states as well as the states whose lex-value vector has zero entries at *all* positions corresponding to reachability objectives. The latter is necessary because as long as a state still has a positive lex-value w.r.t. at least one reachability objective, the optimal behaviour of Max is trying to reach that. Otherwise, Max would not have "tried everything".

Example 5 (Final set) For the game in Fig. 1 with objectives $\Omega_1 = \mathsf{Reach}(S_1), \Omega_2 = \mathsf{Safe}(S_2)$, we have $\mathsf{Zero}_1 = \{u, v, w\}$ and thus $F = \mathsf{Zero}_1 \cup S_1 = \{s, t, u, v, w\}$. An MD lex-optimal strategy of Max must almost-surely reach this set against any strategy of Min; only then it has "tried everything".

We can now characterize MD lex-optimal strategies in terms of local lex-optimality and the Final Set.

Theorem 1 *Let $\mathbf{\Omega}$ be an absorbing reach-safe lex-objective and $\sigma \in \Sigma^{\mathsf{MD}}_{\mathsf{Max}}$. Then σ is lex-optimal for $\mathbf{\Omega}$ if and only if σ is locally lex-optimal and for all $s \in S$ we have*

$$\forall \tau \in \Sigma^{\mathsf{MD}}_{\mathsf{Min}} \;:\; \mathbb{P}^{\sigma,\tau}_s(\mathsf{Reach}(F)) \;=\; 1\,, \tag{2}$$

where F is the Final Set from Definition 6.

Proof sketch The "*if*"-direction is shown by induction on the number n of targets. We make a case distinction according to the type of Ω_n: If it is safety, then we prove that local lex-optimality is already sufficient for global lex-optimality. If Ω_n is reachability, then intuitively, the additional condition (2) ensures that the strategy σ indeed "tries everything" and either reaches the target S_n or eventually a state in Zero_n where the opponent Min can make sure that Max cannot escape. The technical details of these assertions rely on a fixed point characterization of the reachability probabilities and the classical Knaster-Tarski fixed point theorem [48].

For the "*only if*"-direction recall that lex-optimal strategies are necessarily locally lex-optimal by Lemma 1 (a). Further, let i be such that $\Omega_i = \mathsf{Reach}(S_i)$ and assume for contradiction that σ remains forever within $S \setminus (S_i \cup \mathsf{Zero}_i)$ with positive probability against some strategy of Min. But then σ visits states with positive lex-value for Ω_i infinitely often without ever reaching S_i. Thus σ is not globally lex-optimal, contradiction. □

Full Proof of Theorem 1

We first prove the following lemma about the special case of MDP:

Lemma 2 *Let $\mathcal{G}$ be an MDP (i.e., $S_\Box = \emptyset$ or $S_\Diamond = \emptyset$) and let $\mathbf{\Omega}$ be an absorbing lex-objective. Then there exists an MD lex-optimal strategy for $\mathbf{\Omega}$ for the respective player.*

Proof We assume w.l.o.g. that $S_\Diamond = \emptyset$; otherwise we can exchange all $\mathsf{Reach}(S_i)$ for $\mathsf{Safe}(S_i)$ in $\mathbf{\Omega}$ and swap the roles of Min and Max. Fix a state $s \in S$. It is known that the set of points $\mathbf{x} \in [0,1]^n$ such that there exists a strategy $\sigma \in \Sigma_{\mathsf{Max}}$ with

$$\left(\mathbb{P}^{\sigma}_{s}(\Omega_1), \ldots, \mathbb{P}^{\sigma}_{s}(\Omega_n)\right) \dot{\geq} \mathbf{x}$$

where $\dot{\geq}$ denotes point-wise inequality is a *closed convex polyhedron* $\mathfrak{P}$ [21, 49] which is contained in $[0,1]^n$. Therefore $\mathfrak{P}$ contains a maximum $\mathbf{x}^*$ in the order $\leq_{\mathsf{lex}}$. Moreover, $\mathbf{x}^*$ is a vertex of $\mathfrak{P}$, i.e., a point contained in $\mathfrak{P}$ which is not a proper convex combination of two *different* points of $\mathfrak{P}$. If not, then $\mathbf{x}^* = \alpha\mathbf{y} + (1-\alpha)\mathbf{z}$ for $\mathbf{y} \neq \mathbf{z} \in \mathfrak{P}$ and $0 < \alpha < 1$. Let i be the tiebraker position of $\mathbf{y}$ and $\mathbf{z}$. We can assume w.l.o.g. that $\mathbf{y}_i > \mathbf{z}_i$. But then it follows immediately that $\mathbf{y} >_{\mathsf{lex}} \mathbf{x}^*$, contradicting the fact that $\mathbf{x}^*$ was maximal in $\mathfrak{P}$. The claim follows because in MDP the vertices of $\mathfrak{P}$ are achieved by MD strategies [49]. □

Before proving Theorem 1, we show the following intermediate result, where we use $\mathsf{Reach}\left(S_i \cup \mathsf{Zero}_i\right)$ (for all $1 \leq i \leq n$ such that $\Omega_i = \mathsf{Reach}\left(S_i\right)$) instead of $\mathsf{Reach}(F)$.

Lemma 3 *Let $\sigma \in \Sigma^{\mathsf{MD}}_{\mathsf{Max}}$ and let $\mathbf{\Omega}$ be an absorbing reach-safe lex-objective. Then σ is lex-optimal for $\mathbf{\Omega}$ if and only if σ is locally lex-optimal and for all $1 \leq i \leq n$ such that $\Omega_i = \mathsf{Reach}\left(S_i\right)$ and all $s \in S$ it holds that*

$$\forall \tau \in \Sigma^{\mathsf{MD}}_{\mathsf{Min}} : \mathbb{P}^{\sigma,\tau}_{s}(\mathsf{Reach}\left(S_i \cup \mathsf{Zero}_i\right)) = 1. \tag{3}$$

Proof We show the two directions of the "if and only if" statement. Recall that an MC can be simplified to a tuple $\mathcal{M} = (S, P)$ such that $P : S \to \mathcal{D}(S)$.

"*if*": We use the following characterization of the reachability probabilities in any (not necessarily finite) Markov chain: The probabilities $\mathbb{P}_s(\mathsf{Reach}\left(S'\right))$ constitute the least fixed point $x(s)$ of the operator

$$\mathcal{R} : [0,1]^S \to [0,1]^S,\ \mathcal{R}(x)(s) = \begin{cases} 1 & \text{if } s \in S' \\ \sum_{s'} P(s,s') \cdot x(s') & \text{else} \end{cases} \tag{4}$$

which is monotonic on the complete lattice $[0,1]^S$ (that is, the set of all mappings from S to $[0,1]$) [3]. In a finite MC, the fixed point of $\mathcal{R}$ can be made unique by requiring additionally that $\mathcal{R}(x)(s) = 0$ if there is no path from s to S' in the MC.

We now prove the "*if*"-direction by induction on n. We first show the inductive step and then argue that the base case $n = 1$ follows with a similar, slightly simpler argument. Let $n > 1$. Moreover, let $\sigma \in \Sigma^{\mathsf{MD}}_{\mathsf{Max}}$ be locally lex-optimal and assume that (3) holds. To prove that σ is lex-optimal, we let $\tau \in \Sigma_{\mathsf{Min}}$ be a lex-optimal-counter strategy against σ in the induced MDP $\mathcal{G}^{\sigma}$ and show that $\mathbb{P}^{\sigma,\tau}_{s}(\Omega_i) = \mathbf{v}^{\mathsf{lex}}_i(s)$ for all $1 \leq i \leq n$. By Lemma 2, we can assume that τ is MD. By the I.H., σ is already lex-optimal for $\mathbf{\Omega}_{<n} = (\Omega_1, \ldots, \Omega_{n-1})$. Next observe that since τ is a lex-optimal counter-strategy against σ, it holds that

$$\mathbb{P}^{\sigma,\tau}_{s}(\Omega_n) \leq \mathbf{v}^{\mathsf{lex}}_n(s). \tag{5}$$

Thus we only need to prove the other inequality "$\geq$" in (5). Therefore we make a case distinction according to the type of Ω_n:

- $\Omega_n = \mathsf{Safe}\left(S_n\right)$. Consider the MC $\mathcal{G}^{\sigma,\tau}$. Since σ, τ are both MD, this MC has the same finite state space S as the game and its transition probability function is defined as $P^{\sigma,\tau}(s,s') = P(s,\sigma(s),s')$ if $s \in S_{\Box}$ and $P^{\sigma,\tau}(s,s') = P(s,\tau(s),s')$ if $s \in S_{\Diamond}$. In $\mathcal{G}^{\sigma,\tau}$, the safety probabilities $\mathbb{P}_s(\Omega_n) = \mathbb{P}_s(\mathsf{Safe}\left(S_n\right))$ constitute the greatest fixed point of the operator

$$\mathcal{S} : [0,1]^S \to [0,1]^S,\ \mathcal{S}(x)(s) = \begin{cases} 0 & \text{if } s \in S_n \\ \sum_{s'} P^{\sigma,\tau}(s,s') \cdot x(s') & \text{else} \end{cases}$$

which is obtained from the operator $\mathcal{R}$ for reachability using the relation $\mathbb{P}_s(\mathsf{Safe}(S_n)) = 1 - \mathbb{P}_s(\mathsf{Reach}(S_n))$. Just like $\mathcal{R}$, the operator $\mathcal{S}$ is also monotonic on the complete lattice $[0,1]^S$ and we can apply the well-known Theorem of Knaster & Tarski: If we can prove that for all $s \in S$

$$\mathbf{v}_n^{\mathsf{lex}}(s) \leq \mathcal{S}(\mathbf{v}_n^{\mathsf{lex}})(s) \tag{6}$$

then this implies $\mathbf{v}_n^{\mathsf{lex}}(s) \leq (\mathtt{gfp}\ \mathcal{S})(s) = \mathbb{P}_s^{\sigma,\tau}(\Omega_n)$, where $\mathtt{gfp}\ \mathcal{S}$ denotes the greatest fixed point of $\mathcal{S}$. To prove (6), we let $s \in S$ and make another case distinction:

- $s \in S_n$. In this case, $\mathbf{v}_n^{\mathsf{lex}}(s) = 0 \leq \mathcal{S}(\mathbf{v}_n^{\mathsf{lex}})(s)$ holds trivially.
- $s \in S_\Box \setminus S_n$. Then

$$\begin{aligned} \mathbf{v}^{\mathsf{lex}}(s) &= \max_{a \in \mathsf{Act}(s)} \sum_{s'} P(s,a,s') \cdot \mathbf{v}^{\mathsf{lex}}(s') && \text{(by Lemma 1(b))} \\ &= \sum_{s'} P(s,\sigma(s),s') \cdot \mathbf{v}^{\mathsf{lex}}(s') && (\sigma \text{ is locally lex-optimal}) \end{aligned}$$

  and thus in particular $\mathbf{v}_n^{\mathsf{lex}}(s) = \mathcal{S}(\mathbf{v}_n^{\mathsf{lex}})(s)$.
- $s \in S_\Diamond \setminus S_n$. Let $\mathbf{v}_{<n}^{\mathsf{lex}}(s)$ be the lex-value with respect to the first $n-1$ objectives $\mathbf{\Omega}_{<n}$. Since σ is lex-optimal for $\mathbf{\Omega}_{<n}$ and τ is a lex-optimal counter-strategy against σ, we have that

$$\begin{aligned} \mathbf{v}_{<n}^{\mathsf{lex}}(s) &= \min_{a \in \mathsf{Act}(s)} \sum_{s'} P(s,a,s') \cdot \mathbf{v}_{<n}^{\mathsf{lex}}(s') && \text{(by Lemma 1(a))} \\ &= \sum_{s'} P(s,\tau(s),s') \cdot \mathbf{v}_{<n}^{\mathsf{lex}}(s') \end{aligned}$$

  Let $\mathsf{Act}_{<n}(s)$ be the lex-optimal actions available in s with respect to $\mathbf{\Omega}_{<n}$. By the previous equation, $\tau(s) \in \mathsf{Act}_{<n}(s)$. Therefore,

$$\begin{aligned} \mathbf{v}_n^{\mathsf{lex}}(s) &= \min_{a \in \mathsf{Act}_{<n}(s)} \sum_{s'} P(s,a,s') \cdot \mathbf{v}_n^{\mathsf{lex}}(s') \\ &\leq \sum_{s'} P(s,\tau(s),s') \cdot \mathbf{v}_n^{\mathsf{lex}}(s') = \mathcal{S}(\mathbf{v}_n^{\mathsf{lex}})(s). \end{aligned}$$

  Thus together with (5) we have $\mathbf{v}_n^{\mathsf{lex}}(s) = \mathbb{P}_s^{\sigma,\tau}(\Omega_n)$ and σ is lex-optimal for $\mathbf{\Omega} = (\Omega_1, \ldots, \Omega_n)$.
- $\Omega_n = \mathsf{Reach}(S_n)$. This case is a similar but slightly more involved than the previous case. As mentioned earlier, in $\mathcal{G}^{\sigma,\tau}$ the probabilities $\mathbb{P}_s(\Omega_n)$ constitute the *unique* fixed point of the following monotonic operator:

$$\mathcal{R} : [0,1]^S \to [0,1]^S,\ \mathcal{R}(x)(s) = \begin{cases} 1 & \text{if } s \in S_n \\ 0 & \text{if } s \text{ cannot reach } S_n \\ \sum_{s'} P(s,s') \cdot x(s') & \text{else} \end{cases}$$

where the transition probability function P of the Markov chain $\mathcal{G}^{\sigma,\tau}$ is defined as before. As in the other case, we prove that $\mathbf{v}_n^{\mathsf{lex}}(s) \leq \mathcal{R}(\mathbf{v}_n^{\mathsf{lex}})(s)$ for all $s \in S$, which

implies $\mathbf{v}_n^{\mathsf{lex}}(s) \leq (\mathtt{gfp}\,\mathcal{R})(s) = \mathbb{P}_s^{\sigma,\tau}(\Omega_n)$. Notice that the greatest fixed point $\mathtt{gfp}\,\mathcal{R}$ is equal to the unique fixed point of $\mathcal{R}$. Let $s \in S$ and let us again make a case distinction to prove $\mathbf{v}_n^{\mathsf{lex}}(s) \leq \mathcal{R}(\mathbf{v}_n^{\mathsf{lex}})(s)$ for all s:

- If $s \in S_n$, then $\mathbf{v}_n^{\mathsf{lex}}(s) = 1 = \mathcal{R}(\mathbf{v}_n^{\mathsf{lex}})(s)$.
- The cases where s can reach S_n but $s \notin S_n$ can be shown exactly as in the previous case where Ω_n was safety.
- Now suppose s *cannot* reach S_n in $\mathcal{G}^{\sigma,\tau}$, i.e., $\mathbb{P}_s^{\sigma,\tau}(\mathsf{Reach}(S_n)) = 0$. In this case we need to show that $\mathbf{v}_n^{\mathsf{lex}}(s) = 0$, or equivalently, $s \in \mathsf{Zero}_n$. By condition (3), we have for all $t \in S$ that

$$\begin{aligned} & 1 \\ = \;& \mathbb{P}_t^{\sigma,\tau}(\mathsf{Reach}(S_n \cup \mathsf{Zero}_n)) \\ = \;& \mathbb{P}_t^{\sigma,\tau}(\mathsf{Reach}(S_n)) + \mathbb{P}_t^{\sigma,\tau}(\mathsf{Reach}(\mathsf{Zero}_n)) \\ = \;& \mathbb{P}_t^{\sigma,\tau}(\mathsf{Reach}(S_n)) + 1 - \mathbb{P}_t^{\sigma,\tau}(\mathsf{Safe}(\mathsf{Zero}_n)) \end{aligned}$$

  and thus $\mathbb{P}_t^{\sigma,\tau}(\mathsf{Reach}(S_n)) = \mathbb{P}_t^{\sigma,\tau}(\mathsf{Safe}(\mathsf{Zero}_n))$. Therefore, σ is also locally lex-optimal for the objective $(\Omega_1, \Omega_2, \ldots, \mathsf{Safe}(\mathsf{Zero}_n))$. But then we can show exactly as in the previous case that $\mathbf{v}_n^{\mathsf{lex}}(t) \leq \mathcal{S}(\mathbf{v}_n^{\mathsf{lex}})(t)$ where $\mathcal{S}$ is the fixed point operator for safety probabilities associated to the objective $\mathsf{Safe}(\mathsf{Zero}_n)$. Thus $\mathbf{v}_n^{\mathsf{lex}}(s) \leq (\mathtt{gfp}\,\mathcal{S})(s) = \mathbb{P}_s^{\sigma,\tau}(\mathsf{Safe}(\mathsf{Zero}_n)) = 0$.

Finally, for the base case $n = 1$ observe that the same reasoning applies with the simplification that we do not need to care about previous targets. In particular, we do not need to apply the I.H.

"*only if*": Let $\sigma \in \Sigma_{\mathsf{Max}}^{\mathsf{MD}}$ be lex-optimal. First observe that σ is also locally lex-optimal by Lemma 1(a). Now let i be such that $\Omega_i = \mathsf{Reach}(S_i)$, let $s \in S$ be any state and let $\tau \in \Sigma_{\mathsf{Min}}^{\mathsf{MD}}$. It remains to show (3). Assume for contradiction that $\mathbb{P}_s^{\sigma,\tau}(\mathsf{Reach}(S_i \cup \mathsf{Zero}_i)) < 1$. This means that in the *finite* Markov chain $\mathcal{G}^{\sigma,\tau}$, there exists a reachable bottom strongly connected component (BSCC, see [3, Ch. 10]) $B \subseteq S$ such that $B \cap (S_i \cup \mathsf{Zero}_i) = \emptyset$. Thus for every state $t \in B$, we have $\mathbf{v}_i^{\mathsf{lex}}(t) > 0$. Further it holds that $\mathbb{P}_t^{\sigma,\tau}(\mathsf{Reach}(S_i)) = 0$ because s can only reach states inside B, but $B \cap S_i = \emptyset$. However, this is a contradiction to the lex-optimality of σ. □

We now conclude the proof of Theorem 1:

Proof of Theorem 1 Let σ be locally lex-optimal, let $s \in S$ and let $\tau \in \Sigma_{\mathsf{Min}}^{\mathsf{MD}}$. We show the following equivalence:

$$\mathbb{P}_s^{\sigma,\tau}(\mathsf{Reach}(F)) \;=\; 1 \quad \Longleftrightarrow \quad \forall i \in R \;:\; \mathbb{P}_s^{\sigma,\tau}(\mathsf{Reach}(S_i \cup \mathsf{Zero}_i)) \;=\; 1,$$

where $R = \{i \leq n \mid \Omega_i = \mathsf{Reach}(S_i)\}$. The equivalence states that conditions (3) and (2) are equivalent and thus Lemma 3 is equivalent to Theorem 1. For $R = \emptyset$ there is nothing to show, so we let $R \neq \emptyset$.

To show direction "⇒", assume for contradiction that the left-hand side holds but $\mathbb{P}_s^{\sigma,\tau}(\mathsf{Reach}(S_i \cup \mathsf{Zero}_i)) < 1$ for some $i \in R$. Then in the finite MC $\mathcal{G}^{\sigma,\tau}$ there exists a BSCC B which is reachable from s with positive probability and $B \cap (S_i \cup \mathsf{Zero}_i) = \emptyset$. Thus if $t \in B$, then $t \notin S_i$ and $t \notin \mathsf{Zero}_i$. Thus $t \notin F$, contradiction because t is reachable from s with positive probability.

For direction "⇐", the argument is similar. Suppose that the right-hand side holds but $\mathbb{P}_s^{\sigma,\tau}(\mathsf{Reach}(F)) < 1$. Then in the finite MC $\mathcal{G}^{\sigma,\tau}$ there exists a BSCC B which is reachable from s with positive probability and $B \cap F = \emptyset$. Let $t \in B$. Then since $t \notin F$, we have by definition that $t \notin S_i$ for all $i \in R$ and $t \notin \mathsf{Zero}_j$ for some $j \in R$. But this is a contradiction to $\mathbb{P}_s^{\sigma,\tau}(\mathsf{Reach}\big(S_j \cup \mathsf{Zero}_j\big)) = 1$ because t is reachable from s with positive probability. □

Existence of MD lex-optimal strategies

The characterization from Theorem 1 also allows us to prove that MD lex-optimal strategies exist for absorbing reach-safe lex-objectives.

Theorem 2 *In every SG with absorbing reach-safe lex-objective* $\mathbf{\Omega}$*, there exist MD lex-optimal strategies for both players.*

Proof sketch We consider the subgame $\widetilde{\mathcal{G}}$ obtained by removing lex-sub-optimal actions for both players and then show that the (single-objective) value of $\mathsf{Reach}(F)$ in $\widetilde{\mathcal{G}}$ equals 1. An optimal MD strategy for $\mathsf{Reach}(F)$ exists [1]; further, it is locally lex-optimal, because we are in $\widetilde{\mathcal{G}}$, and it reaches F almost surely. Thus, it is lex-optimal for $\mathbf{\Omega}$ by the "*if*"-direction of Theorem 1. □

Full proof Let $\widetilde{\mathcal{G}}$ be the game obtained by removing lex-sub-optimal actions in $\mathcal{G}$ for both players. Let $v(s)$ be the value of state $s \in S$ for the objective $\mathsf{Reach}(F)$ in the modified game $\widetilde{\mathcal{G}}$, where F is the Final Set like in Theorem 1 (we can assume that $R \neq \emptyset$). We show that $v(s) = 1$ for all $s \in S$. Assume towards contradiction that there exists a state s with $v(s) < 1$.

- If $s \in \mathsf{Sinks}(\mathcal{G})$, then either $s \in S_i$ for some $i \in R$, or otherwise s is a sink which is not contained in any of the S_i with $i \in R$ and thus $s \in \mathsf{Zero}_i$ for all $i \in R$. Thus $s \in F$ by definition of F and $v(s) = 1$, contradiction.
- Let $s \notin \mathsf{Sinks}(\mathcal{G})$. Let σ be an MD optimal strategy for $\mathsf{Reach}(F)$ in $\widetilde{\mathcal{G}}$ and let τ be an MD optimal counter-strategy. Notice that such strategies exist because we are only considering a single objective [1]. As usual, we consider the finite MC $\widetilde{\mathcal{G}}^{\sigma,\tau}$. Since $v(s) < 1$, we have $\mathbb{P}_s^{\sigma,\tau}(\mathsf{Reach}(F)) < 1$ which means that there is a BSCC $B \subseteq S$ in $\widetilde{\mathcal{G}}^{\sigma,\tau}$ such that $B \cap F = \emptyset$ and $\mathbb{P}_s^{\sigma,\tau}(\mathsf{Reach}(B)) > 0$. Let $t \in B$ be any state in the BSCC. Then clearly, $\mathbb{P}_t^{\sigma,\tau}(\mathsf{Reach}(F)) = 0$ and thus $v(t) = 0$ because σ is optimal for $\mathsf{Reach}(F)$. But since $t \notin F$, we have by definition of F that $\exists i \in R : t \notin \mathsf{Zero}_i$, which means that $\mathbf{v}_i^{\mathsf{lex}}(t) > 0$. Notice that here, $\mathbf{v}^{\mathsf{lex}}$ are the lex-values in the *original* game $\mathcal{G}$, however by Lemma 1(b), they coincide with the lex-values in $\widetilde{\mathcal{G}}$. Thus since $\mathbf{v}_i^{\mathsf{lex}}(t) > 0$, there is a strategy of Max in $\widetilde{\mathcal{G}}$ that reaches S_i with positive probability against all counter-strategies of Min and thus also reaches F with positive probability because $S_i \subseteq F$. This is a contradiction to $v(t) = 0$.

□

3.1.2 Algorithm for SG with absorbing targets

Theorem 2 is not constructive because it relies on the values $\mathbf{v}^{\mathsf{lex}}$ without showing how to compute them. Computing the values and constructing an optimal strategy for Max in the case of an absorbing reach-safe lex-objective is the topic of this subsection.

Definition 7 (QRO) Consider an SG with state space S and a function $q : S' \to [0,1]$ for some $S' \subseteq S$. The *quantified reachability objective* (QRO) $\mathsf{Reach}(q)$ is defined as follows: For all strategies σ and τ,

$$\mathbb{P}_s^{\sigma,\tau}(\mathsf{Reach}(q)) \quad = \quad \sum_{t \in S'} \mathbb{P}_s^{\sigma,\tau}((S \setminus S') \cup t) \cdot q(t).$$

Intuitively, a QRO generalizes its standard Boolean counterpart by additionally assigning a [0, 1]-valued weight – or *reward* – to the states in the target set S'. The *probability* of a QRO in some (induced) MC with a given initial state s is defined as the expected value of the reward received when reaching S'. Clearly, this number is in [0, 1]. Note that it does not depend on whatever happens after reaching S'; in fact, it is unaffected by making all states in S' absorbing.

In Sect. 3.2, we also need the dual notion of *quantified safety objectives*, whose probability is defined as $\mathbb{P}_s^{\sigma,\tau}(\mathsf{Safe}(q)) = 1 - \mathbb{P}_s^{\sigma,\tau}(\mathsf{Reach}(q))$. Intuitively, maximizing a quantified safety objective is equivalent to minimizing the probability of the dual QRO.

Remark 1 The standard Boolean reachability objective $\mathsf{Reach}(S')$ is a special case of a QRO with $q(s) = 1$ for all $s \in S'$. Vice versa, a QRO can be easily reduced to a standard reachability objective $\mathsf{Reach}(S')$: Convert all states $t \in S'$ into sinks, then for each such t prepend a new state t' with a single action a and $P(t', a, t) = q(t)$ and $P(t', a, \bot) = 1 - q(t)$ where $\bot$ is a sink state. Finally, redirect all transitions leading into t to t'. Despite this equivalence, it turns out to be convenient and natural to use QROs.

Example 6 (QRO) Example 4 illustrated that solving a safety objective after a reachability objective can lead to problems, as the optimal strategy for $\mathsf{Safe}(S_2)$ did not use the action that actually reached S_1. In Example 5, we indicated that the Final Set $F = \{s, t, u, v, w\}$ has to be reached with probability 1, and among the states of F the ones with the highest safety values should be preferred. This requirement can be encoded in a QRO as follows: Compute the values for the $\mathsf{Safe}(S_2)$ objective for the states in F. Then construct the function $q_2 : F \to [0,1]$ that maps all states in F to their safety value, i.e., $q_2 : \{ s \mapsto 1,\ t \mapsto 0,\ u \mapsto 0,\ v \mapsto 1/2,\ w \mapsto 1 \}$.

With QRO, we can effectively reduce (interleaved) safety objectives to quantified *reachability* objectives:

Lemma 4 (Reduction Safe → Reach) *Let* $\mathbf{\Omega} = (\Omega_1, \ldots, \Omega_n)$ *be an absorbing reach-safe lex-objective with* $\Omega_n = \mathsf{Safe}(S_n)$. *Define the QRO* $q_n : F \to [0,1]$ *as* $q_n(t) = \mathbf{v}_n^{\mathsf{lex}}(t)$ *for all* $t \in F$ *where* F *is the Final Set* (*Definition* 6), *and define* $\mathbf{\Omega'} = (\Omega_1, \ldots, \Omega_{n-1}, \mathsf{Reach}(q_n))$. *Then it holds that* ${}^{\mathbf{\Omega}}\mathbf{v}^{\mathsf{lex}} = {}^{\mathbf{\Omega'}}\mathbf{v}^{\mathsf{lex}}$.

Proof (Proof sketch) By definition, ${}^{\mathbf{\Omega}}\mathbf{v}^{\mathsf{lex}}(s) = {}^{\mathbf{\Omega}'}\mathbf{v}^{\mathsf{lex}}(s)$ for all $s \in F$, so we only need to consider the states in $S \setminus F$. Since any lex-optimal strategy for $\mathbf{\Omega}$ or $\mathbf{\Omega}'$ must also be lex-optimal for $\mathbf{\Omega}_{<n}$, we know by Theorem 1 that such a strategy reaches $F_{<n}$ almost-surely. Note that we have $F_{<n} = F$, as the n-th objective, either the QRO or the safety objective, does not add any new states to F. The reachability objective $\mathsf{Reach}(q_n)$ weighs the states in F with their lexicographic safety values $\mathbf{v}_n^{\mathsf{lex}}$. Thus we additionally ensure that in order to reach F, we use those actions that give us the best safety probability afterwards. In this way we obtain the correct lex-values $\mathbf{v}_n^{\mathsf{lex}}$ for states in $S \setminus F$. □

Proof (Full proof) Let $\sigma \in \Sigma^{\mathsf{MD}}_{\mathsf{Max}}$ be lex-optimal for $\mathbf{\Omega}$ (such a σ exists by Theorem 2). Clearly, σ is in particular lex-optimal for the first $n-1$ objectives $\mathbf{\Omega}_{<n}$. Let us denote by $\Sigma^{<n}_{\mathsf{Max}}$ the set of all MD lex-optimal strategies for player Max with respect to $\mathbf{\Omega}_{<n}$. We have $\sigma \in \Sigma^{<n}_{\mathsf{Max}}$.

We know by Theorem 1 that σ reaches $F_{<n} = F$ almost surely against all $\tau \in \Sigma^{\mathsf{MD}}_{\mathsf{Min}}$. Now fix an optimal counter strategy $\tau \in \Sigma^{\mathsf{MD}}_{\mathsf{Min}}$ of Min against σ w.r.t. the full objective $\mathbf{\Omega}$. For an arbitrary MD strategy $\sigma' \in \Sigma^{\mathsf{MD}}_{\mathsf{Max}}$ of Max, let $\Sigma^{<n}_{\mathsf{Min}}(\sigma')$ denote the set of all MD lex-optimal counter strategies against σ' w.r.t. $\mathbf{\Omega}_{<n}$. Clearly, $\tau \in \Sigma^{<n}_{\mathsf{Min}}(\sigma)$.

For all $s \in S$, it holds that

$$
\begin{aligned}
\mathbb{P}_s^{\sigma,\tau}(\mathsf{Safe}(S_n)) &= \sum_{t \in F} \mathbb{P}_s^{\sigma,\tau}((S \setminus F) \cup t) \cdot \mathbb{P}_t^{\sigma,\tau}(\mathsf{Safe}(S_n)) \\
&\qquad \text{(as } F \text{ is reached a.s. in } \mathcal{G}^{\sigma,\tau}\text{, and } (S \setminus F) \cap S_n = \emptyset \text{ as } \mathbf{\Omega} \text{ is absorbing)} \\
\implies \quad \mathbf{v}_n^{\mathsf{lex}}(s) &= \sum_{t \in F} \mathbb{P}_s^{\sigma,\tau}((S \setminus F) \cup t) \cdot \mathbf{v}_n^{\mathsf{lex}}(t) \qquad (\sigma, \tau \text{ are lex-optimal w.r.t. } \mathbf{\Omega}) \\
&= \sup_{\sigma' \in \Sigma^{<n}_{\mathsf{Max}}} \inf_{\tau' \in \Sigma^{<n}_{\mathsf{Min}}(\sigma')} \sum_{t \in F} \mathbb{P}_s^{\sigma',\tau'}((S \setminus F) \cup t) \cdot \mathbf{v}_n^{\mathsf{lex}}(t) \qquad (\text{as } \sigma \in \Sigma^{<n}_{\mathsf{Max}}) \\
&= \sup_{\sigma' \in \Sigma^{<n}_{\mathsf{Max}}} \inf_{\tau' \in \Sigma^{<n}_{\mathsf{Min}}(\sigma')} \mathbb{P}_s^{\sigma',\tau'}(\mathsf{Reach}(q_n)) \quad \text{(Definition 7)} \\
&= \text{the n-th component of the vector } \sup_{\sigma' \in \Sigma^{\mathsf{MD}}_{\mathsf{Max}}} \inf_{\tau' \in \Sigma^{\mathsf{MD}}_{\mathsf{Min}}} \mathbb{P}_s^{\sigma',\tau'}(\mathbf{\Omega}') \\
&= {}^{\mathbf{\Omega}'}\mathbf{v}_n^{\mathsf{lex}}(s)\,.
\end{aligned}
$$

This proves the claim. □

Example 7 (Reduction Safe → Reach) Recall Example 6. By Lemma 4, computing $\sup_\sigma \inf_\tau \mathbb{P}_s^{\sigma,\tau}(\mathsf{Reach}(S_1), \mathsf{Reach}(q_2))$ yields the correct lex-value $\mathbf{v}^{\mathsf{lex}}(s)$ for all $s \in S$. Consider for instance state r: The action leading to q is clearly suboptimal for $\mathsf{Reach}(q_2)$ as it does not reach F. Both other actions surely reach F. However, since $q_2(t) = q_2(u) = 0$ while $q_2(v) = 1/2$, the action leading to u and v is preferred over that leading to t and u, as it ensures the higher safety probability after reaching F.

Algorithm 1 Solve absorbing reach-safe lex-objective

Input: SG $\mathcal{G}$, absorbing reach-safe lex-objective $\mathbf{\Omega} = (\Omega_1, \ldots, \Omega_n)$
Output: Lex-value vectors $\mathbf{v}^{\mathsf{lex}}$, MD lex-optimal strategy σ for Max

1: **procedure** `SolveAbsorbingRS`$(\mathcal{G}, \mathbf{\Omega})$
2: initialize $\mathbf{v}^{\mathsf{lex}}$ and σ arbitrarily
3: $\widetilde{\mathcal{G}} \leftarrow \mathcal{G}$ ▷ Consider whole game in the beginning.
4: **for** $1 \leq i \leq n$ **do**
5: $(v, \widetilde{\sigma}) \leftarrow$ `SolveSingleObj`$(\widetilde{\mathcal{G}}, \Omega_i)$ ▷ Black box
6: **if** $\Omega_i = \mathsf{Safe}\,(S_i)$ **then**
7: $F_{<i} \leftarrow$ Final Set with respect to $\widetilde{\mathcal{G}}$ and $\mathbf{\Omega}_{<i}$ ▷ see Def. 6
8: $q_i(s) \leftarrow v(s)$ for all $s \in F_{<i}$ ▷ see Def. 7
9: $(v, \sigma_Q) \leftarrow$ `SolveSingleObj`$(\widetilde{\mathcal{G}}, \mathsf{Reach}\,(q_i))$ ▷ Black box
10: **end if**
11: $\widetilde{\mathcal{G}} \leftarrow$ restriction of $\widetilde{\mathcal{G}}$ to optimal actions w.r.t. v
12: $\mathbf{v}_i^{\mathsf{lex}} \leftarrow v$
13: **for** $s \in S$ **do**
14: **if** $(\Omega_i{=}\mathsf{Reach}\,(S_i) \wedge v(s){>}0)$ or $(\Omega_i{=}\mathsf{Safe}\,(S_i) \wedge s{\in}F_{<i})$ **then**
15: $\sigma(s) \leftarrow \widetilde{\sigma}(s)$ ▷ Strategy improvement
16: **else if** $\Omega_i{=}\mathsf{Safe}\,(S_i) \wedge s{\notin}F_{<i}$
17: $\sigma(s) \leftarrow \sigma_Q(s)$
18: **end if**
19: **end for**
20: **end for**
return $(\mathbf{v}^{\mathsf{lex}}, \sigma)$
21: **end procedure**

We now explain the basic structure of Algorithm 1. More technical details are explained in the proof sketch of Theorem 3. The idea of Algorithm 1 is, as sketched in Sect. 3.1.1, to consider the objectives sequentially in the order of importance, i.e., starting with Ω_1. The i-th objective is solved (Lines 5-10) and the game is restricted to only the locally optimal actions (Line 11). This way, in the i-th iteration of the main loop, only actions that are locally lex-optimal for objectives 1 through $(i{-}1)$ are considered. Finally, we construct the optimal strategy and update the result variables (Lines 12-13).

Theorem 3 *Given an SG $\mathcal{G}$ and an absorbing reach-safe lex-objective $\mathbf{\Omega} = (\Omega_1, \ldots, \Omega_n)$, Algorithm 1 computes the vector of lex-values $\mathbf{v}^{\mathsf{lex}}$ and an MD lex-optimal strategy σ for player Max. It needs $n + m$ calls to a solver for single reachability, where m is the number of safety objectives in $\mathbf{\Omega}$.*

Proof sketch We explain the intuition of the algorithm and highlight the most interesting details.

- *$\widetilde{\mathcal{G}}$-invariant* For $i \geq 1$, in the i-th iteration of the loop, $\widetilde{\mathcal{G}}$ is the original SG restricted to only those actions that are locally lex-optimal for the targets 1 to $(i{-}1)$; this is the case because Line 11 was executed for all previous targets.

- *Single-objective case* The single-objective that is solved in Line 5 can be either reachability or safety. We can use any (precise) single-objective solver as a black box, e.g. strategy iteration [1]. By Remark 1, it is no problem to call a single-objective solver with a QRO since there is a trivial reduction.
- *QRO for safety* If an objective is of type reachability, no further steps need to be taken; if on the other hand, it is safety, we need to ensure that the problem explained in Example 4 does not occur. Thus we compute the Final Set $F_{<i}$ for the i-th target and then construct and solve the QRO as in Lemma 4.
- *Resulting strategy* When storing the resulting strategy, we again need to avoid errors induced by the fact that locally lex-optimal actions need not be globally lex-optimal. This is why for a reachability objective, we only update the strategy in states that have a positive value for the current objective; if the value is 0, the current strategy does not have any preference, and we need to keep the old strategy. For safety objectives, we need to update the strategy in two ways: for all states in the Final Set $F_{<i}$, we set it to the safety strategy $\widetilde{\sigma}$ (from Line 5) as within $F_{<i}$ we do not have to consider the previous reachability objectives and therefore must follow an optimal safety strategy. For all states in $S \setminus F_{<i}$, we set it to the reachability strategy from the QRO σ_Q (from Line 9). This is correct, as σ_Q ensures almost-sure reachability of $F_{<i}$ which is necessary to satisfy all preceding reachability objectives; moreover σ_Q prefers those states in $F_{<i}$ that have a higher safety value (cf. Lemma 4).

□

Full proof The proof is by induction on n. For $n = 1$, the algorithm is correct by the assumption that `SolveSingleObj` is correct in the single-objective case. Next, we show the inductive step $n > 1$ for reachability and then for safety objectives.

- *Case 1* $\Omega_n = \mathsf{Reach}(S_n)$. By the I.H., $\widetilde{\mathcal{G}}$ is the correct restriction of $\mathcal{G}$ to lex-optimal actions for both players with respect to the first $n-1$ objectives $\mathbf{\Omega}_{<n} = (\Omega_1, \dots, \Omega_{n-1})$ and σ is a lex-optimal MD strategy in $\mathcal{G}$ with respect to $\mathbf{\Omega}_{<n}$. The algorithm computes an MD optimal strategy $\widetilde{\sigma}$ in $\widetilde{\mathcal{G}}$ with respect to the single-objective $\mathsf{Reach}(S_n)$, and the single-objective values $v(s)$ of this objective in $\widetilde{\mathcal{G}}$ by calling `SolveSingleObj` (line 5). The strategy $\sigma \in \Sigma^{\mathsf{MD}}_{\mathsf{Max}}$ is then updated as follows:

$$\sigma(s) \quad = \quad \begin{cases} \widetilde{\sigma}(s) & \text{if } \widetilde{v}(s) > 0 \\ \sigma_{old}(s) & \text{if } \widetilde{v}(s) = 0. \end{cases}$$

  We claim that σ is lex-optimal for the whole lex-objective $\mathbf{\Omega} = (\Omega_1, \dots, \Omega_n)$.

  - We first show that σ remains lex-optimal for the first $n-1$ objectives $\mathbf{\Omega}_{<n}$ by applying the "*if*"-direction of Lemma 3: First observe that by definition, σ is locally lex-optimal with respect to $\mathbf{\Omega}_{<n}$. Therefore it only remains to show condition (3) in Lemma 3. Let $i < n$ such that $\Omega_i = \mathsf{Reach}(S_i)$, let $s \in S$ and let $\tau \in \Sigma^{\mathsf{MD}}_{\mathsf{Min}}$ be a counter-strategy against σ. If $v(s) = 0$, then $\mathbb{P}^{\sigma,\tau}_s(\mathsf{Reach}(S_i \cup \mathsf{Zero}_i)) = 1$ because from initial state s, σ behaves like σ_{old} which is lex-optimal for $\mathbf{\Omega}_{<n}$. Thus let $v(s) > 0$. From the "*only if*"-direction of Lemma 3 applied to $\widetilde{\sigma}$, we know that $\mathbb{P}^{\sigma,\tau}_s(\mathsf{Reach}(S_n \cup \{s \in S \mid v(s) = 0\})) = 1$. Thus for a play π that starts in s and is consistent with σ, τ, almost-surely one of the following two cases occurs:

    * If π reaches a state $t \in S_n$, then since t is a sink, we have $t \in S_i \cup \mathsf{Zero}_i$.
    * If t with $v(t) = 0$ is reached in π, then since we play according to σ_{old} from t, we either reach S_i or Zero_i by the "*only if*"-direction of Lemma 1 applied to σ_{old}.

Thus $\mathbb{P}_s^{\sigma,\tau}(\mathsf{Reach}(S_i \cup \mathsf{Zero}_i)) = 1$ and σ remains lex-optimal for $\mathbf{\Omega}_{<n}$.

- To complete the proof that σ is lex-optimal for $\mathbf{\Omega}$, notice that $v(s) = \mathbf{v}_n^{\mathsf{lex}}(s)$ for all $s \in S$ by Lemma 1.

- *Case 2* $\Omega_n = \mathsf{Safe}(S_n)$. Since by the I.H., the values $\mathbf{v}_1^{\mathsf{lex}}, \dots, \mathbf{v}_{n-1}^{\mathsf{lex}}$ are the correct lex-values with respect to $\mathbf{\Omega}_{<n}$, the algorithm computes the final set $F_{<n} = F_{<n+1} = F$.
  Next observe that $v(s) = \mathbf{v}_n^{\mathsf{lex}}(s)$ for all $s \in F$ because of the following:

  - First, $\widetilde{\sigma}$ is locally lex-optimal w.r.t. $\mathbf{\Omega}_{<n}$ because it is defined in the subgame $\widetilde{\mathcal{G}}$. Therefore by Lemma 1, σ is already (globally) lex-optimal for $\mathbf{\Omega}_{<n}$ from all $s \in F$ because condition 2 is satisfied trivially. Thus $v(s) \leq \mathbf{v}_n^{\mathsf{lex}}(s)$.
  - Second, by the same argument, an MD lex-optimal strategy for $\mathbf{\Omega}$ is necessarily locally lex-optimal w.r.t. $\mathbf{\Omega}_{<n}$. The strategy $\widetilde{\sigma}$ is locally lex-optimal w.r.t. $\mathbf{\Omega}_{<n}$ and moreover optimal for Ω_n in the subgame $\widetilde{\mathcal{G}}$. Thus $v(s) \geq \mathbf{v}_n^{\mathsf{lex}}(s)$.

  Notice that it is very well possible that $v(s) > \mathbf{v}_n^{\mathsf{lex}}(s)$ for $s \notin F$ because the strategy $\widetilde{\sigma}$ does not necessarily reach F from $s \notin F$. Applying Lemma 4 concludes the proof: The quantified reachability objective q_n constructed by the algorithm indeed satisfies $q_n(s) = v(s) = \mathbf{v}_n^{\mathsf{lex}}(s)$ for all $s \in F$ as we have just shown. The result strategy σ defined by the algorithm is

$$\sigma(s) \quad = \quad \begin{cases} \widetilde{\sigma}(s) & \text{if } s \in F \\ \sigma_Q(s) & \text{if } s \notin F \end{cases}$$

  where $\sigma_Q$is a lex-optimal strategy for $\mathbf{\Omega}' = (\Omega_1, \dots, \Omega_{n-1}, \mathsf{Reach}(q_n))$. Thus with Lemma 4 and the above discussion, σ is lex-optimal from *all* states.

□

3.2 General lexicographic reach-safe SG

We now consider SG with general reach-safe lex-objectives that are *not* necessarily absorbing. In Section 3.2.1, we describe a reduction to the absorbing case. The resulting algorithm is given in Section 3.2.2. Finally, in Section 3.2.3, we discuss complexity and determinacy results that follow easily from our algorithms.

3.2.1 Reduction to absorbing reach-safe lex-objectives

In general lexicographic SG, strategies need memory. This is because they have to remember which reachability and safety objectives have already been satisfied and violated, respectively, and behave accordingly. We formalize the solution of such games by means of *stages*. Intuitively, one can think of a stage as a copy of the game with fewer objectives, or as the *sub-game* that is played after visiting one of the targets or unsafe sets for the first time. This approach is standard for dealing with multiple reachability or safety objectives [12, 50, 51].

Definition 8 (Stage) Given a general reach-safe lex-objective $\mathbf{\Omega} = (\mathbf{\Omega}_1, \dots, \mathbf{\Omega}_n)$ and a set $I \subseteq \{i \leq n\}$, a *stage* $\mathbf{\Omega}(I)$ is the objective vector where the objectives $\mathbf{\Omega}_i$ are *removed* for all $i \in I$. For every state $s \in S$, let $\mathbf{\Omega}(s)$ denote the stage $\mathbf{\Omega}(\{i \mid s \in S_i\})$. If a stage contains only one objective, we call it *simple*.

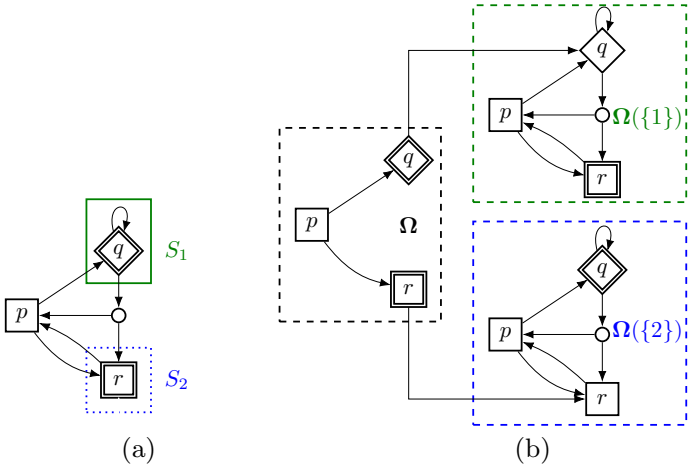


Fig. 2 **a** SG with non-absorbing lex-objective $\mathbf{\Omega} = (\mathsf{Reach}(S_1), \mathsf{Reach}(S_2))$. **b** The three stages identified by the sub-objectives $\mathbf{\Omega}, \mathbf{\Omega}(\{1\}) = \mathsf{Reach}(S_2)$, and $\mathbf{\Omega}(\{2\}) = \mathsf{Reach}(S_1)$. The latter two stages are *simple*

Example 8 (Stages) Consider the SG in Fig. 2a on Page 24. There are two (reachability) objectives and thus $2^2 = 4$ four possible stages: The one where we consider the original objective (the region denoted with $\mathbf{\Omega}$ in Fig. 2b), the *simple* ones where we consider only one of the objectives (regions $\mathbf{\Omega}(\{1\})$ and $\mathbf{\Omega}(\{2\})$), and the one where both objectives have been visited. This final stage is trivial since there are no more objectives left, hence we do not depict it and do not have to consider it. Note that the actions of q and r are omitted in the $\mathbf{\Omega}$-stage, this is because a new stage begins once we visit these states for the first time.

Consider the two simple stages first: In stage $\mathbf{\Omega}(\{1\})$, state q has value 0 as it belongs to player Min who may use the self-loop to avoid reaching $r \in S_2$. In stage $\mathbf{\Omega}(\{2\})$, both p and r have value 1 because Max can easily reach the target state $q \in S_1$ from both of them. By combining this knowledge, we can assemble an optimal strategy for every possible initial state in the principal stage $\mathbf{\Omega}$. In particular, note that an optimal Max-strategy for state p needs memory: First go to r and thereby reach stage $\mathbf{\Omega}(\{2\})$. Afterwards, go back from r to p, and then use the other action in p to reach q and the final stage $\mathbf{\Omega}(\{1, 2\})$.

Note that this example reveals another interesting fact about lexicographic games: Optimal strategies may try to satisfy less important objectives first.

In Example 8, we combined our knowledge about the sub-objectives in the corresponding stages to find the lex-values for the overall objective. In general, the lex-values of the stages are vectors in [0, 1]. We will reuse the idea of *quantified* reachability and safety objectives from Definition 7 (Page 18) in order to find an optimal strategy for some stage $\mathbf{\Omega}$ *given the values of its substages*: For all $1 \leq i \leq n$, let $q_i : \bigcup_{j \leq n} S_j \to [0, 1]$ be defined as follows:

$$q_i(s) \quad = \quad \begin{cases} 1 & \text{if } s \in S_i \text{ and otherwise:} \\ {}^{\mathbf{\Omega}(s)}\mathbf{v}_i^{\mathsf{lex}}(s) & \text{if } \Omega_i \text{ is reachability} \\ 1 - {}^{\mathbf{\Omega}(s)}\mathbf{v}_i^{\mathsf{lex}}(s) & \text{if } \Omega_i \text{ is safety.} \end{cases} \tag{7}$$

Recall that $\mathbf{\Omega}(s)$ denotes the stage $\mathbf{\Omega}(\{i \mid s \in S_i\})$. Further, we define

$$\mathsf{q}\mathbf{\Omega} \quad = \quad (\,\mathsf{type}_1(q_1),\, \ldots,\, \mathsf{type}_n(q_n)\,) \tag{8}$$

where for all $1 \leq i \leq n$, $\mathsf{type}_i = \mathsf{Reach}$ if $\Omega_i = \mathsf{Reach}(S_i)$, and $\mathsf{type}_i = \mathsf{Safe}$ if $\Omega_i = \mathsf{Safe}(S_i)$. Thus we have reduced a general reach-safe lex-objective $\boldsymbol{\Omega}$ to a vector of *quantitative* reachability or safety objectives $\mathsf{q}\boldsymbol{\Omega}$. This reduction preserves the values:

Lemma 5 *For every reach-safe lex-objective* $\boldsymbol{\Omega}$*, it holds that* ${}^{\boldsymbol{\Omega}}\mathbf{v}^{\mathsf{lex}} = {}^{\mathsf{q}\boldsymbol{\Omega}}\mathbf{v}^{\mathsf{lex}}$*, where* $\mathsf{q}\boldsymbol{\Omega}$ *is the quantified reach-safe lex-objective defined in* (7) *and* (8)*.*

Proof In this proof, we write $\mathfrak{S} = \bigcup_{i \leq n} S_i$ for the sake of readability. For the case $s \in \mathfrak{S}$, a straightforward induction shows that $q_i(s) = {}^{\boldsymbol{\Omega}}\mathbf{v}_i^{\mathsf{lex}}(s) = {}^{\mathsf{q}\boldsymbol{\Omega}}\mathbf{v}_i^{\mathsf{lex}}(s)$ for all $1 \leq i \leq n$.

Now let $s \notin \mathfrak{S}$. Let σ, τ be arbitrary strategies of Max and Min, respectively. Further, suppose that $\Omega_i = \mathsf{Reach}(S_i)$. In the induced Markov chain $\mathcal{G}^{\sigma,\tau}$, the following holds (*all infima and suprema are taken over lex-optimal (counter-)strategies with respect to* $\boldsymbol{\Omega}_{<i}$):

$$
\begin{aligned}
\mathbb{P}_s^{\sigma,\tau}(\Omega_i) &= \sum_{\pi t \in Paths_{fin}(\mathfrak{S})} \mathbb{P}_s^{\sigma,\tau}(\pi t) \cdot \mathbb{P}_{\pi t}^{\sigma,\tau}(\Omega_i) && \text{(because } \pi \text{ visits n state in } \mathfrak{S}) \\
&= \sum_{\pi t \in Paths_{fin}(\mathfrak{S})} \mathbb{P}_s^{\sigma,\tau}(\pi t) \cdot \mathbb{P}_t^{\sigma(\pi),\tau(\pi)}(\Omega_i) && \text{(where } \sigma(\pi) \text{ behaves like } \sigma \text{ after seeing } \pi) \\
\Longrightarrow \quad \mathbf{v}_i^{\mathsf{lex}}(s) &= \sup_{\sigma} \inf_{\tau} \sum_{\pi t \in Paths_{fin}(\mathfrak{S})} \mathbb{P}_s^{\sigma,\tau}(\pi t) \cdot \sup_{\sigma(\pi)} \inf_{\tau(\pi)} \mathbb{P}_t^{\sigma(\pi),\tau(\pi)}(\Omega_i) \\
&\quad \text{(because behavior of } \sigma, \tau \text{ after } \pi \text{ is independent of the behavior before } \pi) \\
&= \sup_{\sigma} \inf_{\tau} \sum_{\pi t \in Paths_{fin}(\mathfrak{S})} \mathbb{P}_s^{\sigma,\tau}(\pi t) \cdot {}^{\boldsymbol{\Omega}}\mathbf{v}_i^{\mathsf{lex}}(t) && \text{(by definition of the lex-value } {}^{\boldsymbol{\Omega}}\mathbf{v}^{\mathsf{lex}}(t)) \\
&= \sup_{\sigma} \inf_{\tau} \sum_{\pi t \in Paths_{fin}(\mathfrak{S})} \mathbb{P}_s^{\sigma,\tau}(\pi t) \cdot q_i(t) && \text{(because } t \in \mathfrak{S}) \\
&= \sup_{\sigma} \inf_{\tau} \sum_{t \in \mathfrak{S}} \mathbb{P}_s^{\sigma,\tau}((S \setminus \mathfrak{S}) \,\mathsf{U}\, t) \cdot q_i(t) && \text{(by definition of the until property)} \\
&= \sup_{\sigma} \inf_{\tau} \mathbb{P}_s^{\sigma,\tau}(\mathsf{Reach}(q_i)) = {}^{\mathsf{q}\boldsymbol{\Omega}}\mathbf{v}_i^{\mathsf{lex}}(s).
\end{aligned}
$$

where we used the notation $Paths_{fin}(\mathfrak{S}) = \{\pi t \in (S \setminus \mathfrak{S})^* \times S \mid t \in \mathfrak{S}\}$ for the set of all finite paths to a state in $\mathfrak{S}$ in $\mathcal{G}^{\sigma,\tau}$ and $\mathbb{P}_s^{\sigma,\tau}(\pi t)$ denotes the probability of such a path when the Markov chain $\mathcal{G}^{\sigma,\tau}$ starts in s.

If Ω_i is a safety objective instead, then the claim follows from the above using the relationship $\mathbb{P}_s^{\sigma,\tau}(\mathsf{Safe}(S_i)) = 1 - \mathbb{P}_s^{\sigma,\tau}(\mathsf{Reach}(S_i))$. □

The reward functions q_i involved in $\mathsf{q}\boldsymbol{\Omega}$ all have the same domain $\bigcup_{j \leq n} S_j$. Hence we can, as mentioned below Definition 7, consider $\mathsf{q}\boldsymbol{\Omega}$ on the game where all states in $\bigcup_{j \leq n} S_j$ are sinks without changing the lex-value. This is precisely the definition of an absorbing game, and hence we can compute ${}^{\mathsf{q}\boldsymbol{\Omega}}\mathbf{v}^{\mathsf{lex}}$ using Algorithm 1 from Sect. 3.1.2.

3.2.2 Algorithm for general SG

Algorithm 2 computes the lex-value ${}^{\boldsymbol{\Omega}}\mathbf{v}^{\mathsf{lex}}$ for a given lexicographic objective $\boldsymbol{\Omega}$ and an arbitrary SG $\mathcal{G}$. We highlight the following technical details:

- *Reduction to absorbing case* We just have seen that once we have the quantitative objective vector $\mathsf{q}\boldsymbol{\Omega}$, we can use the algorithm for absorbing SG (Line 13).

- *Computing the quantitative objective vector* To compute $\mathsf{q}\boldsymbol{\Omega}$, the algorithm calls itself recursively on all states in the union of all target sets (Line 5-7). We annotated this recursive call "With dynamic programming", as we can reuse the results of the computations. In the worst case, we have to solve all $2^n - 1$ possible non-empty stages. Finally, given the values ${}^{\boldsymbol{\Omega}(s)}\mathbf{v}^{\mathsf{lex}}$ for all $s \in \bigcup_{j \leq n} S_j$, we can construct the quantitative objective (Line 10 and 12) that is used for the call to `SolveAbsorbingRS`.
- *Termination* Since there are finitely many objectives in $\boldsymbol{\Omega}$ and in every recursive call at least one objective is removed from consideration, eventually we have a *simple* objective that can be solved by `SolveSingleObj` (Line 3).
- *Resulting strategy* The resulting strategy is composed in Line 14: It adheres to the strategy for the quantitative query ${}^{\mathsf{q}\boldsymbol{\Omega}}\sigma$ until some $s \in \bigcup_{j \leq n} S_j$ is reached. Then, to achieve the values promised by $q_i(s)$ for all i with $s \notin S_i$, it adheres to ${}^{\boldsymbol{\Omega}(s)}\sigma$, the optimal strategy for stage $\boldsymbol{\Omega}(s)$ obtained by the recursive call.

Algorithm 2 Solve non-absorbing reach-safe lex-objective

Input: SG $\mathcal{G}$, reach-safe lex-objective $\boldsymbol{\Omega} = (\Omega_1, \ldots, \Omega_n)$
Output: Lex-values ${}^{\boldsymbol{\Omega}}\mathbf{v}^{\mathsf{lex}}$, lex-optimal $\sigma \in \Sigma_{\mathsf{Max}}$ with at most $2^n - 1$ memory

1: **procedure** `SolveGeneralRS`$(\mathcal{G}, \boldsymbol{\Omega})$
2: **if** $\boldsymbol{\Omega} = (\Omega_1)$ is just a single objective **then** ▷ Base case
3: **return** `SolveSingleObj`$(\mathcal{G}, \Omega_1)$ ▷ Blackbox
4: **end if**
▷ Recursive call with dynamic programming
5: **for** $s \in \bigcup_{j \leq n} S_j$ **do**
6: $\left({}^{\boldsymbol{\Omega}(s)}\mathbf{v}^{\mathsf{lex}},\ {}^{\boldsymbol{\Omega}(s)}\sigma\right) \leftarrow$ `SolveGeneralRS`$(\mathcal{G}, \boldsymbol{\Omega}(s))$
7: **end for**
▷ Assemble quantified objective as in (7) and (8)
8: **for** $1 \leq i \leq n$ **do**
9: Let $q_i \colon \bigcup_{j \leq n} S_j \to [0, 1]$,
10: $q_i(s) \leftarrow \begin{cases} 1 & \text{if } s \in S_i \text{ and else:} \\ {}^{\boldsymbol{\Omega}(s)}\mathbf{v}_i^{\mathsf{lex}}(s) & \text{if } \mathsf{type}(\Omega_i) = \mathsf{Reach} \\ 1 - {}^{\boldsymbol{\Omega}(s)}\mathbf{v}_i^{\mathsf{lex}}(s) & \text{if } \mathsf{type}(\Omega_i) = \mathsf{Safe} \end{cases}$
11: **end for**
12: $\mathsf{q}\boldsymbol{\Omega} \leftarrow (\mathsf{type}_1(q_1), \ldots, \mathsf{type}_n(q_n))$
13: $\left({}^{\mathsf{q}\boldsymbol{\Omega}}\mathbf{v}^{\mathsf{lex}},\ {}^{\mathsf{q}\boldsymbol{\Omega}}\sigma\right) \leftarrow$ `SolveAbsorbingRS`$(\mathcal{G}, \mathsf{q}\boldsymbol{\Omega})$ ▷ Algorithm 1
▷ Resulting strategy; note that ${}^{\boldsymbol{\Omega}(s)}\sigma$ was computed in Line 6
14: $\sigma \leftarrow$ adhere to ${}^{\mathsf{q}\boldsymbol{\Omega}}\sigma$ until reaching some $s \in \bigcup_{j \leq n} S_j$, then to ${}^{\boldsymbol{\Omega}(s)}\sigma$
15: **return** $({}^{\mathsf{q}\boldsymbol{\Omega}}\mathbf{v}^{\mathsf{lex}}, \sigma)$
16: **end procedure**

Theorem 4 *Given an SG $\mathcal{G}$ and a reach-safe lex-objective $\mathbf{\Omega} = (\Omega_1, \ldots, \Omega_n)$, Algorithm 2 correctly computes the vector of lex-values $\mathbf{v}^{\mathsf{lex}}$ and a deterministic lex-optimal strategy σ of player* Max *which uses memory of class-size at most 2^n-1. The algorithm needs at most 2^n-1 calls to* `SolveAbsorbingRS` *or* `SolveSingleObj`.

Proof Correctness of the algorithm and termination follows from the discussion of the algorithm, Lemma 5 and Theorem 3. □

3.2.3 Determinacy and complexity

Theorem 5 below states that lexicographic games are *determined* for arbitrary lex-objectives $\mathbf{\Omega}$. Intuitively, this means that the lex-value is independent of the player who fixes their strategy first. This property does not hold for non-lexicographic multi-reachability/safety objectives [12].

Theorem 5 (Determinacy) *For all SG $\mathcal{G}$ and reach-safe lex-objectives $\mathbf{\Omega}$, it holds for all $s \in S$ that*:

$$\mathbf{v}^{\mathsf{lex}}(s) \quad = \quad \sup_{\sigma \in \Sigma_{\mathsf{Max}}} \inf_{\tau \in \Sigma_{\mathsf{Min}}} \mathbb{P}_s^{\sigma,\tau}(\mathbf{\Omega}) \quad = \quad \inf_{\tau \in \Sigma_{\mathsf{Min}}} \sup_{\sigma \in \Sigma_{\mathsf{Max}}} \mathbb{P}_s^{\sigma,\tau}(\mathbf{\Omega}) \,.$$

Proof This statement follows because single-objective games are determined [1] and Algorithm 2 obtains all values by either solving single-objective instances directly (Line 3) or calling Algorithm 1, which also reduces everything to the single-objective case (Line 5 of Algorithm 1). Thus the sup-inf values $\mathbf{v}^{\mathsf{lex}}$ returned by the algorithm are in fact equal to the inf-sup values. □

By analyzing Algorithm 2, we also get the following complexity results:

Theorem 6 (Complexity) *For any SG $\mathcal{G}$ and reach-safe lex-objective $\mathbf{\Omega}$ of size n*:

1. *Strategy complexity: Deterministic strategies with $2^n - 1$ memory-classes (i.e., bit-size n) are sufficient and necessary for lex-optimal strategies.*
2. *Computational complexity: The decision problem $\mathbf{v}^{\mathsf{lex}}(s_0) \geq^{?}_{\mathsf{lex}} \mathbf{x}$ is* PSPACE*-hard and can be solved in* NEXPTIME ∩ coNEXPTIME. *If n is a constant or $\mathbf{\Omega}$ is absorbing, then it is contained in* NP ∩ coNP.

Proof

1. For each stage, Algorithm 2 computes an MD strategy for the quantitative objective. These strategies are then concatenated whenever a new stage is entered. Equivalently, every stage has an MD strategy for every state, so as there are at most $2^n - 1$ stages (since there are n objectives), the strategy needs at most $2^n - 1$ states of memory; these can be represented with n bits. Intuitively, we save for every target set whether it has been visited. The memory lower bound already holds in non-stochastic reachability games where all n targets have to be visited with certainty [50].

2. The work of [28] shows that in MDP, it is PSPACE-hard to decide if n targets can be visited almost-surely. This problem trivially reduces to ours. For the NP upper bound, observe that there are at most $2^n - 1$ stages, i.e., a constant amount if n is assumed to be constant (or even just one stage if $\mathbf{\Omega}$ is absorbing). Thus we can guess an MD strategy for player Max in every stage. The guessed overall strategy can then be checked by analyzing the induced MDP in polynomial time [21]. The same procedure works for player Min and since the game is determined, we have membership in coNP. In the same way, we obtain the NEXPTIME ∩ coNEXPTIME upper bound in the general case where n is arbitrary.

□

We leave the question of whether PSPACE is also an upper bound open. The main obstacle towards proving PSPACE-membership is that it is unclear if the lex-value – being dependent on the value of *exponentially* many stages in the worst-case – may actually have exponential bit-complexity.

4 MDP with lexicographic ω-regular objectives

In this section, we study one-player SG, also known as MDP, with ω-regular lexicographic objectives. Similarly to the previous section, we first consider a simpler problem in Sect. 4.1: We show how to solve lexicographic ω-regular objectives in *end components* – fragments of the state space where the player can remain forever if it chooses to do so. Then, using this, we solve arbitrary MDP in Sect. 4.2. We provide our motivation for restricting attention to MDP instead of the more general SG in the final Sect. 4.3.

Intuitively, the reason for first considering end components is that ω-regular objectives depend on the infinite suffix of a path, and such an infinite suffix can only occur with positive probability inside an end component. Thus, a standard approach for dealing with ω-regular objectives is to analyze the end components first. Afterwards, we consider the problem of reaching the end components where the objectives can be satisfied. The end components that are ranked higher in the lexicographic order are preferred. In this way we reduce the problem to lexicographic reachability.

There exist several acceptance conditions that capture exactly the class of ω-regular objectives; here, we will use *Streett* acceptance for technical reasons. In practice, ω-regular conditions are often specified as formulas in *Linear Temporal Logic* (LTL). Such formulas can be transformed to deterministic Streett automata (DSA) with a doubly exponential space blow-up, see e.g. [52, 53]. Model checking an MDP against an LTL formula reduces to checking Streett acceptance of the product of the MDP with the corresponding DSA (see Definition 3). It is also possible to iterate this product construction for several LTL formulas or DSA. We assume in the following that we are already given a "product" MDP with n Streett conditions.

4.1 Lexicographic streett in end components

Intuitively, an *end component* of an MDP is a subset of states where the player (i) *can* remain forever if it wants to, and (ii) can visit all states in the subset infinitely often with probability 1. We now give a formal definition. Let $\mathcal{M}$ be an MDP with state space S, and let $E \subseteq S$. For a state $s \in E$ and an action $a \in \mathsf{Act}(s)$, we say that *a exits E* if $P(s, a, s') > 0$ for some $s' \notin E$.

Definition 9 (End component) Let $\mathcal{M} = (S, \mathsf{Act}, P)$ be an MDP. A set $E \subseteq S$ is called an *end component* (EC) of $\mathcal{M}$ if there exists $A \subseteq \bigcup_{s \in E} \mathsf{Act}(s)$ such that

1. for all $a \in A$, it holds that a does not exit E, and
2. for all $s, s' \in E$, there exists a finite path $sa_0 \dots a_n s' \in (E \times A)^* \times E$.

An EC that is not contained in another EC is called *maximal* (MEC).

The set $\mathcal{E}$ of all MEC of an MDP can be computed in polynomial time, see e.g. [54]. Further, recall that a Streett objective Ω for an MDP with state space S is defined in terms of a set of pairs $\Omega = \mathsf{Streett}((\mathsf{F}_j, \mathsf{I}_j)_{1 \le j \le m})$ such that $\mathsf{F}_j, \mathsf{I}_j \subseteq S$ for all $1 \le j \le m$. An infinite trajectory $s_0 s_1 \dots \in S^\omega$ satisfies Ω iff

$$\forall\, 1 \le j \le m \;:\; \{\, i \ge 0 \mid s_i \in \mathsf{F}_j \,\} \text{ is finite} \quad \text{or} \quad \{\, i \ge 0 \mid s_i \in \mathsf{I}_j \,\} \text{ is infinite}\,.$$

4.1.1 Solving single streett in end components

We first recall a polynomial-time algorithm from [55, Sec. 4] for deciding whether a given EC of some MDP is *good* or *bad* for a Streett objective. Here, good means that the player can satisfy the Streett objective with probability 1 without leaving the EC; a bad EC is one that is not good. Note that remaining in a bad EC forever would violate the Streett objective. Hence the Street value of all states in an EC is either 0 or 1.

Let E be an EC of some MDP $\mathcal{M}$, and let $(\mathsf{F}_j, \mathsf{I}_j)_{1 \le j \le m}$ be a Streett condition on $\mathcal{M}$. The algorithm proceeds in an alternating fashion. It first identifies the *bad states* of E. A state $s \in E$ is bad if $s \in \mathsf{F}_j$ and $\mathsf{I}_j \cap E = \emptyset$ for some $1 \le j \le m$. Intuitively, s is bad because the player may visit s at most finitely many times; thus, in order to satisfy the Streett condition, s must be eventually avoided forever with probability 1. The algorithm removes all bad states from E which results in a sub-MDP $\widetilde{E}$. In general, $\widetilde{E}$ is not an end component due to the state removal. However, if $\widetilde{E}$ is an EC, then we can label it as good. Indeed, since $\widetilde{E}$ contains no bad states, it suffices to visit all states in $\widetilde{E}$ infinitely often with probability one[2]. The original EC E is then also good; the player can satisfy the Streett objective by simply navigating to $\widetilde{E}$ which is possible with an MD strategy. If $\widetilde{E}$ is not an EC, then the algorithm computes a MEC decomposition of $\widetilde{E}$, calls itself recursively on all the resulting MEC, and labels E as good iff it finds that at least one of the MEC is good. In summary:

Lemma 6 (from [55]) *There is a polynomial-time algorithm* `SingleStreettSat` *which takes as input an EC E of some MDP $\mathcal{M}$ together with a Streett condition $(\mathsf{F}_j, \mathsf{I}_j)_{1 \le j \le m}$ on $\mathcal{M}$, and outputs* `true` *iff E is good. In case E is good, the algorithm additionally outputs an optimal strategy for the states inside the EC.*

[2] Visiting every state in an EC infinitely often with probability 1 is achieved by a memoryless randomized strategy that picks actions uniformly at random, or, alternatively, by a deterministic strategy using memory constructed as in [56, Lem. 5.4].

4.1.2 Algorithm for lexicographic streett in end components

We now present an algorithm that computes the lex-value of a Street lex-objective for an EC of some MDP. Note that this lex-value is a vector in $\{0,1\}^n$ which is moreover independent of the initial state. The key idea of our algorithm is the observation that Streett objectives are closed under conjunction. Indeed, for Streett conditions $\Omega_1 = \mathsf{Streett}((\mathsf{F}^1_j, \mathsf{I}^1_j)_{1\le j\le m_1})$ and $\Omega_2 = \mathsf{Streett}((\mathsf{F}^2_j, \mathsf{I}^2_j)_{1\le j\le m_2})$ it holds that a given path satisfies both Ω_1 and Ω_2 iff it satisfies the *single* Streett condition

$$\Omega_1 \wedge \Omega_2 \;=\; \mathsf{Streett}((\mathsf{F}^1_1, \mathsf{I}^1_1), \ldots, (\mathsf{F}^1_{m_1}, \mathsf{I}^1_{m_1}), (\mathsf{F}^2_1, \mathsf{I}^2_1), \ldots, (\mathsf{F}^2_{m_2}, \mathsf{I}^2_{m_2}))\,. \tag{9}$$

With this observation and `SingleStreettSat` as described above, our algorithm is easy to implement and runs in polynomial time.

Algorithm 3 `LexStreettEC`: Lexicographic Streett in EC of an MDP

Input: EC E of some MDP, lex-objective $\boldsymbol{\Omega} = (\Omega_1, \ldots, \Omega_n)$ where for all $1 \le i \le n$, we have $\Omega_i = \mathsf{Streett}((\mathsf{F}^i_j, \mathsf{I}^i_j)_{1\le j\le m_i})$

Output: Lex-value ${}^{\boldsymbol{\Omega}}\mathbf{v}^{\mathsf{lex}}$ of E w.r.t. $\boldsymbol{\Omega}$ (is independent of initial state), and a lex-optimal strategy ${}^{\boldsymbol{\Omega}}\sigma$

procedure `LexStreettEC`$(E, \boldsymbol{\Omega})$
  $\mathbf{v} \leftarrow (0, \ldots, 0)$ ▷ Will contain the resulting lex-value
  $\Omega_{curr} \leftarrow \mathsf{Streett}(\,(\,)\,)$ ▷ Initially empty Streett objective
  for i from 1 to n **do**
    $\Omega_{test} \leftarrow \Omega_{curr} \wedge \Omega_i$ ▷ As in (9)
    if `SingleStreettSat`(E, Ω_{test}) **then** ▷ As in Lemma 6
      $\mathbf{v}_i = 1$
      $\Omega_{curr} \leftarrow \Omega_{test}$
    end if
  end for
  $\sigma \leftarrow$ strategy computed by last successful call to `SingleStreettSat`
  return $(\mathbf{v}, \sigma)$
end procedure

Lemma 7 *Algorithm* 3 *computes the correct lex-values of an EC in polynomial time.*

Proof The idea of the algorithm relies on the following observation: Let $1 \le i \le n$, and let

$$\Omega_{<i} \quad = \quad \bigwedge_{j<i,\, {}^{\Omega}\mathbf{v}_j^{\mathsf{lex}}=1} \Omega_j \,.$$

Then it holds that ${}^{\Omega}\mathbf{v}_i^{\mathsf{lex}} = 1$ iff the *single* Streett objective $\Omega_{<i} \wedge \Omega_i$ can be achieved with probability 1 in E, and otherwise ${}^{\Omega}\mathbf{v}_i^{\mathsf{lex}} = 0$. This is because in an EC, a Streett objective is either achievable with probability 1 or 0. Our algorithm computes $\Omega_{<i}$ iteratively; $\Omega_{<i}$ is called Ω_{curr} in Algorithm 3.

Formally, we show the following invariant by induction: For all $0 \leq i \leq n$ and $j \leq i$, it holds that after i iterations of the loop in Line 4, the objective Ω_{curr} contains Ω_j iff ${}^{\Omega}\mathbf{v}_j^{\mathsf{lex}} = 1$. This trivially holds for $i = 0$, i.e., if the loop has not been executed yet. For $i > 0$, observe that by the induction hypothesis we have after $i - 1$ iterations that ${}^{\Omega}\mathbf{v}_i^{\mathsf{lex}} = 1$ iff $\Omega_{test} = \Omega_{curr} \wedge \Omega_i$ can be achieved with probability 1 in E (Line 6).

If so, then after the i iterations, Ω_{curr} will contain Ω_i, and otherwise Ω_{curr} will not contain Ω_i, which establishes the invariant again. Overall, correctness of the algorithm follows because after $i = n$ iterations, we have for all $1 \leq j \leq i$ that $\boldsymbol{v}_j = 1$ iff Ω_j is contained in Ω_{curr} iff ${}^{\Omega}\mathbf{v}_j^{\mathsf{lex}} = 1$. □

4.2 Lexicographic streett in general MDP

We now show how to compute the value of a lexicographic Streett objective in general MDP, i.e., not just in end components as in the previous subsection. We use the standard approach of first computing a MEC decomposition of the MDP, analysing the MEC in isolation, and then reduce the overall question to a reachability problem. In particular, in our case, we obtain a lexicographic reachability problem, and we can reuse the algorithm from Sect. 3. We state our algorithm formally as Algorithm 4.

Concretely, we construct a modified MDP $\widetilde{\mathcal{M}}$ (Line 2, 7 and 8), where for every MEC E we add a new sink state s_E that can be reached from every state in the MEC (Line 7 and 8). This sink intuitively corresponds to remaining in the MEC forever, using the optimal strategy σ_E of the MEC as computed by Algorithm 3 (Line 6). Thus, choosing to remain in E achieves the lex-value $\mathbf{v}_E^{\mathsf{lex}}$ of the MEC E. We remodel our objective as a reachability lex-objective Ω_{reach} by having each target set T_i contain exactly those s_E where $\mathbf{v}_{E,i}^{\mathsf{lex}} = 1$ (Line 3, 9 and 11). Now, by solving this absorbing reachability objective Ω_{reach} in the modified MDP $\widetilde{\mathcal{M}}$ (Line 12), we compute the lex-value for the Streett objective in the original MDP $\mathcal{M}$. Intuitively, we compute the optimal probability to reach a good MEC E and then, by going to the sink s_E, use the optimal strategy in the MEC to indeed satisfy the best combination of Streett objectives. Thus, the resulting strategy is a combination of the strategy σ_{reach} to reach the MEC E and the strategy σ_E to remain in the MEC E (Line 13).

Algorithm 4 `LexStreettMDP`: Lexicographic Streett in general MDP

Input: MDP $\mathcal{M}$, Streett lex-objective $\mathbf{\Omega} = (\Omega_1, \ldots, \Omega_n)$
Output: Lex-values ${}^{\mathbf{\Omega}}\mathbf{v}^{\mathsf{lex}}$ (one for each state), lex-optimal strategy σ

1: **procedure** `LexStreettMDP`($\mathcal{M}$, $\mathbf{\Omega}$)
2: $\widetilde{\mathcal{M}} \leftarrow \mathcal{M}$ ▷ $\widetilde{\mathcal{M}}$ will be a modified version of $\mathcal{M}$
3: $T_1, \ldots, T_n \leftarrow \emptyset$ ▷ Initialize target sets for reachability lex-objective
4: $\mathcal{E} \leftarrow$ `ComputeMEC`($\mathcal{M}$) ▷ e.g. [54]
5: **for** $E \in \mathcal{E}$ **do**
6: $(\mathbf{v}_E^{\mathsf{lex}}, \sigma_E) \leftarrow$ `LexStreettEC`($E, \mathbf{\Omega}$) ▷ Algorithm 3
7: Add new sink state s_E to $\widetilde{\mathcal{M}}$.
8: For all $s \in E$, add the action $s \to s_E$ to $\widetilde{\mathcal{M}}$.
9: For all $1 \le i \le n$, if $\mathbf{v}_{E,i}^{\mathsf{lex}} = 1$, then $T_i \leftarrow T_i \cup \{s_E\}$.
10: **end for**
11: $\Omega_{reach} \leftarrow (\mathsf{Reach}(T_1), \ldots, \mathsf{Reach}(T_n))$
12: $(\mathbf{v}, \sigma_{reach}) \leftarrow$ `SolveAbsorbingRS`($\widetilde{\mathcal{M}}, \Omega_{reach}$) ▷ Algorithm 1, Sec. 3.1
13: $\sigma \leftarrow$ Adhere to σ_{reach} until it takes an action $s \to s_E$, for some $E \in \mathcal{E}$, then adhere to σ_E.
14: **return** $(\mathbf{v}, \sigma)$
15: **end procedure**

Theorem 7 *Given an MDP $\mathcal{M}$ and a Streett lex-objective $\mathbf{\Omega}$, Algorithm 4 computes the lex-values ${}^{\mathbf{\Omega}}\mathbf{v}^{\mathsf{lex}} : S \to [0,1]^n$ and a lex-optimal strategy σ in polynomial time.*

Proof The proof relies on the following standard observation: Independent of the chosen strategy, the player eventually reaches an EC of $\mathcal{M}$ with probability 1 and stays there forever. That is, there may exist strategies such that the induced MC has infinite paths that never reach an EC; however, their total probability mass is zero. Moreover, whether or not a Streett objective is satisfied on a given infinite path that reaches and (stays in) some EC depends only on that EC, but not on whatever happened before reaching it. This discussion implies that the player should prefer reaching EC with higher lex-value. More formally, for a given initial MDP state $s \in S$, the player should choose a strategy σ that (lexicographically) maximizes the following quantity:

$$\sum_{E \in \mathcal{E}} \mathbb{P}_s^{\sigma}(\mathsf{Reach}(E)) \cdot \mathbf{v}_E^{\mathsf{lex}} \tag{10}$$

The supremum (and maximum) of this quantity is indeed the lex-value $\mathbf{v}^{\mathsf{lex}}(s)$. The strategy $\sigma = \sigma_{reach}$ computed in Line 12 maximizes (10) by construction of Ω_{reach} and due to the additional actions $s \to s_E$ that we have added for all EC E and $s \in E$ in Line 8; recall that these actions simulate staying forever in E, thereby "earning" the lex-value of E.

For the runtime, observe that `ComputeMEC` can be implemented in polynomial time [54], and there are at most $|S|$ many MEC. The algorithm `LexStreettEC` can also be implemented in polynomial time as discussed in Sect. 4.1. Finally, note that `SolveAbsorbingRS` can be implemented in polynomial time for MDP due to Algorithm 1 in Sect. 3.1 and the fact that single-objective MDP reachability can be encoded as a linear program [2]. □

Theorem 8 (Complexity) *The following holds for all MDP $\mathcal{M}$ with state space S and Streett lex-objective $\mathbf{\Omega}$.*

1. *Strategy complexity*: *Memoryless* (*but randomized*), *or alternatively*, *deterministic strategies with at most $|S|$ memory classes are sufficient to play optimally in $\mathcal{M}$ w.r.t. to $\mathbf{\Omega}$.*
2. *Computational complexity*: *The decision problem ${}^{\mathbf{\Omega}}\mathbf{v}^{\mathsf{lex}}(s_0) \geq^{?}_{\mathsf{lex}} \mathbf{x}$ is in P for all $s_0 \in S$.*

Proof

1. The strategy that Algorithm 4 returns is a combination of the strategy σ_{reach} for the absorbing reachability lex-objective and the strategies σ_E inside the maximal end components $E \in \mathcal{E}$. The former is MD by Algorithm 1, but the latter either needs randomization or memory, see Sect. 4.1. The construction from [56, Lem. 5.4] shows that at most $|E|$ memory classes are sufficient for a deterministic σ_E. Therefore, the overall strategy needs at most $\sum_{E \in \mathcal{E}} |E| \leq |S|$ memory since the MEC are pairwise disjoint.
2. Algorithm 4 solves the decision problem in polynomial time.

□

We stress that the complexity results from Theorem 8 crucially rely on the Streett objectives being directly defined on $\mathcal{M}$. If we had instead defined $\mathbf{\Omega}$ through n deterministic Streett automata, then we would first have to construct the product of $\mathcal{M}$ with all the automata, causing an exponential space blowup.

Now that we have established how to solve MDP with lexicographic ω-regular objectives, we discuss the case of SG with such objectives.

4.3 On SG with lexicographic ω-regular objectives

The techniques we introduced in Sect. 3 for solving reachability and safety are not applicable to ω-regular objectives. This is because they rely on reachability and safety being achieved and violated in finite time, respectively. Every stage in the solution corresponds to a subset of objectives being achieved/violated, and this decomposition into stages is vital for the reduction to the single-dimensional case.

In contrast, for ω-regular objectives, satisfaction or violation of an objective depends *only on the infinite suffix of the path*. This is why in MDP we analyze end components (EC), the parts of the state space where the infinite behaviour can occur. For SG, the concept of EC is not strong enough to fully understand the infinite behaviours. It still captures parts of the state space where a path can remain forever, but the standard definition assumes that the players cooperate to stay. However, the players are antagonistic, and Min will try to violate the objective. Thus, not all states in an EC will have the same value (as they do in MDP), and we need a finer decomposition of EC.

There are similar problems in single-dimensional reachability in SG [57]. There, the concepts of *bloated* and *simple* EC were used for a more fine-grained analysis. Equivalently, in deterministic games with parity objectives, the concept of *tangle* [58] captures parts of an EC where one player can certainly win.

We conjecture that, inspired by these concepts, it is possible to decompose EC in SG such that an algorithm similar to our Algorithm 4 solves the problem. However, even mimicking the first step of the development of simple EC [59] is not obvious: Consider for instance SG with one-player EC, i.e., for every EC T it holds that either $T \subseteq S_{\Box}$ or $T \subseteq S_{\Diamond}$. In this case, it still holds that all states in an EC have the same value, so the general algorithm is the same as in MDP. The difference is that we have to compute the lexicographic value in end components belonging to Min. Min wants to violate a Street objective, which is equivalent to achieving the Rabin objective with the same pairs [60]. But for minimizing the lexicographic value, we want to violate all the given Street objectives, which means achieving a conjunction of Rabin objectives. This, however, calls for a more involved solution, because we mix conjunction (between the objectives) and disjunction (between the pairs of objectives). In contrast, our solution relies on the fact that conjunction of Street objectives is again a Street objective. Thus, already in SG with one-player EC a solution needs more complex automata theoretic tools, e.g. Emerson-Lei automata [61] for representing the mixed Boolean combination of requirements on infinite paths.

We decided to focus on MDP with ω-regular objectives, where an elegant and easy-to-implement solution is possible by using the lexicographic reachability approach from Section 3.1 and the existing results about Streett objectives in MDP. We leave the development of new graph-theoretic concepts for SG with ω-regular objectives as future work.

5 Experimental evaluation

In this section, we report the results of a series of experiments made with implementations of the algorithms in Sect. 3 and 4.

5.1 Reach-safe lex-objective in SG: experiments

5.1.1 Implementation

We have implemented Algorithms 1 and 2 as prototypes within PRISM-games [42]. The code is available online[3]. Since PRISM-games does not provide an *exact* algorithm to solve SG, we used the available value iteration approach to implement the single-objective black-box `SolveSingleObj`. Note that since value iteration is not exact for single-objective SG, PRISM-games cannot compute the exact lex-values. Nevertheless, we focus on measuring the overhead introduced by our algorithm compared to a single-objective solver. In our implementation, value iteration stops if the values do not increase by more than 10^{-8} in one iteration, which is PRISM's default configuration.

[3] https://doi.org/10.5281/zenodo.5798108

5.1.2 Case studies

For our experiments, we have collected four case studies from the literature where lexicographic objectives seem particularly useful:

Dice This example is shipped with PRISM-games [42] and models a simple dice game between two players. The number of throws in this game is a configurable parameter, which we instantiate with 10, 20 and 50. The game has three possible outcomes: Player Max wins, Player Min wins or draw. A natural lex-objective is thus to maximize the winning probability and then the probability of a draw.
Charlton This case study [37] is also included in PRISM-games. It models an autonomous car navigating through a road network. A natural lex-objective is to minimize the probability of an accident (possibly damaging human life) and then maximize the probability to reach the destination.
Hallway (HW) This instance is based on the Hallway example which is standard in the AI literature [62, 63]. A robot can move north, east, south or west in a known environment, but each move only succeeds with a certain probability, and otherwise rotates or moves the robot in an undesired direction. We extend the example by a target wandering around based on a mixture of probabilistic and demonic non-deterministic behavior, thereby obtaining a stochastic game that models, for instance, a panicking person in a building on fire. Moreover, we assume a 0.01 probability of damaging the robot when executing certain movements; a damaged robot's actions succeed with even smaller probability. The primary objective is to save the human, and the secondary objective is to avoid damaging the robot. We use grid-worlds of sizes 5×5, 8×8, and 10×10.
Avoid the observer (AV) This case study is inspired by a similar example in [64]. It models a game between an intruder and an observer in a grid-world. The grid can have different sizes as in HW, and we use 10×10, 15×15, and 20×20. The most important objective of the intruder is to avoid the observer, and its secondary objective is to exit the grid. We assume that the observer can only detect the intruder within a certain distance and otherwise makes random moves. At every position, the intruder moreover has the option to stay and search for a treasure. In our example, a treasure is found with probability 0.1 each time the intruder decides to search for it. Collecting a treasure is the ternary (reachability) objective.

Note that the above case studies (and in fact also the ones from Sect. 5.2 further below) have just 2-3 objectives. We are currently not aware of any examples where a significantly larger amount of objectives is useful or natural. In any case, this might not be computationally feasible as the runtime of our algorithm is exponential in the number of objectives in the worst case.

5.1.3 Experimental results

The experiments were conducted on a 2.4 GHz Quad-Core Intel© Core^{TM}i5 processor, with 4GB of RAM available to the Java VM. The results are reported in Table 1. We only recorded the run time of the actual algorithms; the time needed to parse and build the model

Table 1 Experimental results for SG with reach-safe lex-objectives

Model	$\|S\|$	Time [s]			Average #actions		#Stages
		Lex	First	All	$\mathcal{G}$	$\widetilde{\mathcal{G}}$	
R – R							
Dice[10]	4,855	<1	<1	<1	1.42	1.41	1/3
Dice[20]	16,915	<1	<1	<1	1.45	1.45	1/3
Dice[50]	96,295	3	2	2	1.48	1.48	1/3
S – R							
Charlton	502	<1	<1	<1	1.56	1.07	3/3
R – S							
HW[5×5]	25,000	10	7.15	7	2.44	1.02	3/3
HW[8×8]	163,840	152	117	117	2.50	1.01	3/3
HW[10×10]	400,000	548	435	435	2.52	1.01	3/3
S – R – R							
AV[10×10]	106,524	15	<1	10	2.17	1.55, 1.36	4/7
AV[15×15]	480,464	85	<1	50	2.14	1.52, 1.36	4/7
AV[20×20]	1,436,404	281	3	172	2.13	1.51, 1.37	4/7

is excluded. All numbers are rounded to full seconds. All instances (even those with state spaces of order 10^6) could be solved within a few minutes.

The two leftmost columns of Table 1 show the type of the lex-objective, the name of the case study with scaling parameters (if applicable), and the number of states in the model. The next three columns contain the verification times (excluding time to parse and build the model), rounded to full seconds. The columns labeled $\mathcal{G}$ and $\widetilde{\mathcal{G}}$ provide the average number of actions per state, i.e., the value $|S|^{-1} \sum_{s \in S} |\mathsf{Act}(s)|$ in the original SG $\mathcal{G}$ as well as in all subgames $\widetilde{\mathcal{G}}$ (which result from removing lex-suboptimal actions, see Algorithm 1) considered in the main stage. The rightmost column reports on the fraction of stages that had to be analyzed, i.e., the stages solved by the algorithm compared to the theoretically maximal possible number of stages ($2^n - 1$).

We compare the time of our algorithm on the lexicographic objective (column Lex) to the time for checking just the first single-objective (First) and the sum of checking all single-objectives individually (All). The runtimes of our algorithm and checking all single-objectives individually are always in the same order of magnitude. This shows that our algorithm works well in practice and that the overhead is often small. Even on SG of non-trivial size (HW[10×10] and AV[20×20]), our algorithm returns the result within a few minutes.

Regarding the average number of actions, we see that the decrease in the number of actions in the sub-games $\widetilde{\mathcal{G}}$ obtained by restricting the input game to optimal actions varies: For example, very few actions are removed in the Dice instances, in AV we have a moderate decrease, and in HW a significant decrease and almost all the non-determinism is eliminated after the first objective. We conjecture that the less actions are removed, the higher is the overhead compared to the individual single-objective solutions (column All). Consider the AV and HW examples: While for AV[20×20], computing the lexicographic solution takes

1.7 times as long as all the single-objective solutions, it took only about 25% longer for HW[10×10]; this is the case because in HW, only very few actions remain after the first objective. In AV, on the other hand, lots of choices have to be considered even for the second and third objective. Note that the first objective sometimes (HW), but not always (AV) needs the majority of the runtime.

We also see that the algorithm does not always have to explore all possible stages. For example, for Dice we always just need a single stage, because the SG is absorbing. For Charlton and HW all stages are relevant for the lex-objective, while for AV only 4 of 7 need to be considered.

5.2 MDP with lexicographic LTL: experiments

5.2.1 Implementation

We implemented Algorithms 3 and 4 as an extension of Storm [44]. This allows us to reuse existing efficient implementations for basic algorithms, such as finding end components and solving single-objective reachability. The code is available online[4].

Our implementation accepts as input an MDP and an ordered list of LTL formulas. We transform each LTL formula into a deterministic Streett automaton using the tool Spot [65], and then construct the standard product automaton of the MDP and all the automata. This results in a (larger) MDP with several Streett objectives. We then proceed as described in the algorithms in Sect. 4.

5.2.2 Case studies

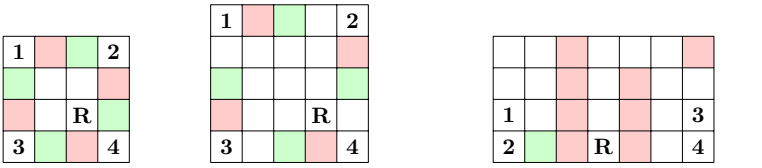
We evaluate our implementation on the benchmarks from [18] and on one additional new case study, the CleaningRobot. In summary, we consider the following case studies for our experiments:

Cleaningrobot A robot cleans a house with two floors. It starts in the upper floor. At some point, it can decide to try to clean the stairs, but with a probability of 0.5 it will fall down and not clean the stairs. After this, it can only clean the ground floor, and never go up to the first floor again. The LTL objectives that we check are (in this order) (1) $\mathbf{GF}\ clean_{first}$, (2) $\mathbf{F}\ clean_{stairs}$, and (3) $\mathbf{GF}\ clean_{ground}$.

Gridworld This example models a robot that moves around in a grid world, has to visit several outposts, avoid dangerous zones, and recharge every now and then. It can only move left, right, up and down. The uncertainty comes into play when the robot chooses to go into one direction: With probability of 0.5, it will move one step, and with 0.5 probability it will move two steps. A visualization can be found in Fig. 3, together with the objectives that we checked.

Virus This example models a computer virus that is trying to breach a system. The system consists of a grid of nodes, 3×3 in our case. If a node is infected, it can choose to attack its neighboring nodes. This will be detected with a probability of 0.5. After a node was successfully attacked, it will get infected with a probability of 0.5, as well. This model stems originally from [66]. The objectives to be verified are:

[4] https://doi.org/10.5281/zenodo.7233605



(a) 4×4 grid (b) 5×5 grid (c) 4×7 grid

Fig. 3 Robot Gridworld: A robot (R) has to visit the outposts (**1**,**2**,**3**,**4**), recharge (green states) and avoid dangerous zones (red states). The graphics are taken from [18]

1. $\mathbf{F}s_{3,2} = 2 \ \wedge \ \mathbf{G}((s_{3,2} = 2 \wedge s_{3,1} \neq 2) \implies \mathbf{XX}s_{3,1} = 2)$, and
2. $\mathbf{F}s_{1,1} = 2 \ \wedge \ \mathbf{G}(s_{2,2} \neq 2 \wedge s_{3,2} \neq 2)$.

The first one makes sure that node (3, 2) will eventually be infected, and once it is infected, node (3, 1) should be infected in at most two steps. The second objective defines a specific path to infect node (1, 1), without infecting nodes (2, 2) and (3, 2).

UAV This model considers an unmanned aerial vehicle (UAV) that moves on a network of roads, originally presented in [67]. There are waypoints that should be visited (modelled by the first objective), and restricted operation zones that should be avoided

Table 2 Experimental Results. The first two columns show the model (with scaling parameters) and the respective number of states. The third column describes the lexicographic order of the objectives, and the fourth column contains the lex-value vector. The two rightmost columns show the running time of our algorithm and of the reinforcement learning algorithm of [18]

Model	|S|	|MEC|	Order	Result	Time [s]	
					Ours	[18]
CleaningRobot	5	4	1, 2, 3	1, 0, 0	<1	2
CleaningRobot	5	4	2, 1, 3	0.5, 0.5, 0.5	<1	3
CleaningRobot	5	4	3, 2, 1	1, 0.5, 0	<1	2
Gridworld[4×4]	16	14	1, 2, 3	1, 0, 1	<1	13
Gridworld[4×4]	16	14	2, 1, 3	1, 0, 1	<1	4
Gridworld[5×5]	25	2	1, 2, 3	1, 1, 1	<1	26
Gridworld[4×7]	28	46	1, 2, 3	1, 0.75, 0.25	<1	44
Gridworld[4×7]	28	46	2, 1, 3	1, 0.5, 0	<1	24
Gridworld[10×10]	100	2	1, 2, 3	1, 1, 1	<1	5
Gridworld[20×20]	400	2	1, 2, 3	1, 1, 1	<1	7
Gridworld[30×30]	900	2	1, 2, 3	1, 1, 1	<1	9
Virus	809	1042	1, 2	1, 0	28	87
Virus	809	1042	2, 1	1, 0.25	28	405
UAV	11,448	6578	1, 2	1, 1	1367	213

(modelled by the second objective). Formally, the objectives are (1) $\bigwedge_i \mathbf{G}\neg roz_i$, and (2) $\mathbf{F}w_1 \wedge \mathbf{F}w_2 \wedge \mathbf{F}w_6$.

5.2.3 Experimental results

The experiments were conducted a server running Ubuntu 20.04.2 with 251 GB RAM and a Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz. The results are reported in Table 2. We only recorded the run time of the actual algorithms; the time needed to parse and build the model is excluded because they were all below one second. All numbers are rounded to full seconds.

For each model of the case study, we report the number of states, the number of MEC, and the lexicographic preference order of the objectives. As a result we show the probabilities of each objective, and the runtime of our approach, as well as the runtime of [18], which solves the same problem using Reinforcement Learning. Apart from the existing benchmarks we also added some more models.

We see that the runtime of our approach is significantly faster on most models, e.g., less than one second on a 4×7-grid in contrast to 44 seconds. However, on models with a larger state space, and especially a large number of MEC, the learning procedure of [18] scales better, e.g., it only takes 213s for the UAV-model, whereas our approaches takes 1367s. This scaling is a known advantage and the goal of learning procedures. However, for this they they sacrifice guarantees on the resulting probabilities, which can be arbitrarily far off. In contrast, after termination, our algorithm outputs the *exact* lex-value and *provably optimal* strategies.

6 Conclusion and future work

In this work, we considered simple stochastic games with lexicographic objectives. Simple stochastic games are a standard model in reactive synthesis of stochastic systems, and lexicographic objectives allow for an analysis with multiple objectives with an order of preference. We first focused on the most basic objectives: safety and reachability. While simple stochastic games with lexicographic objectives have not been studied before, we have presented determinacy, strategy complexity, computational complexity, and algorithms for these games. Moreover, we have shown how these games can model different case studies and presented experimental results. Afterwards, we discussed stochastic games with ω-regular objectives and gave a simple solution for the subclass of Markov decision processes with lexicographic ω-regular objectives represented as Streett conditions.

There are several directions for future work. First, for lexicographic reachability-safety objectives in SG, closing the complexity gap ($\mathsf{NEXPTIME} \cap \mathsf{coNEXPTIME}$ upper bound and PSPACE lower bound, see Theorem 6) is an open question. Second, the problem of solving stochastic games with lexicographic ω-regular objectives remains open. Combinations of ω-regular objectives in SG are a difficult open problem in general. For instance, even qualitative combinations for more than two objectives as well as quantitative combinations have not been solved yet [68]. Finally, one can consider lexicographic combinations

of quantitative objectives such as mean payoff or total reward, which allow for modeling further practical applications.

Funding Tobias Winkler and Joost-Pieter Katoen are supported by the DFG RTG 2236 UnRAVeL and the innovation programme under the Marie Skłodowska-Curie grant agreement No. 101008233 (Mission). Krishnendu Chatterjee is supported by the ERC CoG 863818 (ForM-SMArt) and the Vienna Science and Technology Fund (WWTF) Project ICT15-003. Maximilian Weininger is supported by the DFG projects 383882557 Statistical Unbounded Verification (SUV) and 427755713 Group-By Objectives in Probabilistic Verification (GOPro). Stefanie Mohr is supported by the DFG RTG 2428 CONVEY. Open Access funding enabled and organized by Projekt DEAL.

Data availibility The software and data used to generate the experimental results of this paper are available online(See https://doi.org/10.5281/zenodo.5798108 and https://doi.org/10.5281/zenodo.7233605 for the experiments described in Sect. 5.1 and Sect. 5.2, respectively.).

References

1. Condon A (1992) The complexity of stochastic games. Inf Comput 96(2):203–224. https://doi.org/10.1016/0890-5401(92)90048-K
2. Puterman ML (2014) Markov decision processes: discrete stochastic dynamic programming. Wiley, New Jersey
3. Baier C, Katoen J-P (2008) Principles of model checking. MIT Press, Massachusetts
4. Filar J, Vrieze K (1997) Competitive markov decision processes. Springer, Switzerland
5. Chatterjee K, Henzinger TA (2012) A survey of stochastic ω-regular games. J Comput Syst Sci 78(2):394–413. https://doi.org/10.1016/j.jcss.2011.05.002
6. Chatterjee K, Sen K, Henzinger TA (2008) Model-checking omega-regular properties of interval Markov chains. In: FoSSaCS. Lecture notes in computer science, vol 4962, Springer, Switzerland, pp 302–317
7. Weininger M, Meggendorfer T, Kretínský J (2019) Satisfiability bounds for ω-regular properties in bounded-parameter Markov decision processes. In: CDC, IEEE, New York, pp, 2284–2291, doi: https://doi.org/10.1109/CDC40024.2019.9029460
8. Altman E (1999) Constrained Markov decision processes. CRC Press, Florida
9. Chatterjee K (2007) Markov decision processes with multiple long-run average objectives. In: FSTTCS. Lecture notes in computer science, vol 4855, Springer, Switzerland, pp 473–484, doi: https://doi.org/10.1007/978-3-540-77050-3_39
10. Delgrange F, Katoen J, Quatmann T, Randour M (2020) Simple strategies in multi-objective MDPs. In: TACAS (1). Lecture notes in computer science, vol 12078, Springer, Switzerland, pp 346–364, doi: https://doi.org/10.1007/978-3-030-45190-5_19
11. Berthon R, Guha S, Raskin J (2020) Mixing probabilistic and non-probabilistic objectives in markov decision processes. In: LICS, ACM, New York, pp 195–208, doi: https://doi.org/10.1145/3373718.3394805
12. Chen T, Forejt V, Kwiatkowska MZ, Simaitis A, Wiltsche C (2013) On stochastic games with multiple objectives. In: MFCS. Lecture notes in computer science, vol 8087, Springer, Switzerland, pp 266–277, doi: https://doi.org/10.1007/978-3-642-40313-2_25
13. Fishburn PC (1974) Exceptional paper - lexicographic orders, utilities and decision rules: a survey. Manage Sci 20(11):1442–1471. https://doi.org/10.1287/mnsc.20.11.1442

14. Blume L, Brandenburger A, Dekel E (1991) Lexicographic probabilities and choice under uncertainty. Econom: J Econom Soc. https://doi.org/10.2307/2938240
15. Bloem R, Chatterjee K, Henzinger TA, Jobstmann B (2009) Better quality in synthesis through quantitative objectives. In: CAV. Lecture notes in computer science, vol 5643, Springer, Switzerland, pp 140–156, doi: https://doi.org/10.1007/978-3-642-02658-4_14
16. Colcombet T, Jurdzinski M, Lazic R, Schmitz S (2017) Perfect half space games. In: Logic in Computer Science, LICS 2017, IEEE Computer Society, Washington, DC, pp 1–11, doi: https://doi.org/10.1109/LICS.2017.8005105
17. Wray KH, Zilberstein S, Mouaddib A (2015) Multi-objective MDPs with conditional lexicographic reward preferences. In: AAAI, AAAI Press, California, pp 3418–3424, doi: https://doi.org/10.5555/2888116.2888191
18. Hahn EM, Perez M, Schewe S, Somenzi F, Trivedi A, Wojtczak D (2021) Model-free reinforcement learning for lexicographic omega-regular objectives. In: FM. Lecture Notes in Computer Science, vol 13047, Springer, Switzerland , pp 142–159, doi: https://doi.org/10.1007/978-3-030-90870-6_8
19. Chatterjee K, Katoen J, Weininger M, Winkler T (2020) Stochastic games with lexicographic reachability-safety objectives. In: CAV (2). Lecture notes in computer science, vol 12225, Springer, Switzerland, pp 398–420, doi: https://doi.org/10.1007/978-3-030-53291-8_21
20. Bouyer P, Roux SL, Oualhadj Y, Randour M, Vandenhove P (2022) Games where you can play optimally with arena-independent finite memory. Log Methods Comput Sci. https://doi.org/10.46298/lmcs-18(1:11)2022
21. Etessami K, Kwiatkowska MZ, Vardi MY, Yannakakis M (2008) Multi-objective model checking of Markov decision processes. LMCS. https://doi.org/10.2168/LMCS-4(4:8)2008
22. Brázdil T, Brozek V, Chatterjee K, Forejt V, Kucera A (2014) Two views on multiple mean-payoff objectives in Markov decision processes. LMCS. https://doi.org/10.2168/LMCS-10(1:13)2014
23. Chatterjee K, Forejt V, Wojtczak D (2013) Multi-objective discounted reward verification in graphs and mdps. In: LPAR. Lecture notes in computer science, vol 8312, Springer, Switzerland, pp 228–242, https://doi.org/10.1007/978-3-642-45221-5_17
24. Forejt V, Kwiatkowska MZ, Norman G, Parker D, Qu H (2011) Quantitative multi-objective verification for probabilistic systems. In: TACAS, pp 112–127 . doi: https://doi.org/10.1007/978-3-642-19835-9_11
25. Quatmann T, Katoen J (2021) Multi-objective optimization of long-run average and total rewards. In: TACAS (1). Lecture notes in computer science, vol 12651, Springer, Switzerland, pp 230–249, doi: https://doi.org/10.1007/978-3-030-72016-2_13
26. Brázdil T, Chatterjee K, Forejt V, Kucera A (2017) Trading performance for stability in markov decision processes. J Comput Syst Sci 84:144–170. https://doi.org/10.1016/j.jcss.2016.09.009
27. Filar JA, Krass D, Ross KW (1995) Percentile performance criteria for limiting average Markov decision processes. IEEE Trans Autom Control 40(1):2–10
28. Randour M, Raskin J, Sankur O (2017) Percentile queries in multi-dimensional Markov decision processes. Formal Methods Syst Des 50(2–3):207–248. https://doi.org/10.1007/s10703-016-0262-7
29. Chatterjee K, Kretínská Z, Kretínský J (2017) Unifying two views on multiple mean-payoff objectives in Markov decision processes. LMCS. https://doi.org/10.23638/LMCS-13(2:15)2017
30. Baier C, Dubslaff C, Klüppelholz S (2014) Trade-off analysis meets probabilistic model checking. In: CSL-LICS, pp 1–1110, doi: https://doi.org/10.1145/2603088.2603089
31. Baier C, Dubslaff C, Klüppelholz S, Daum M, Klein J, Märcker S, Wunderlich S (2014) Probabilistic model checking and non-standard multi-objective reasoning. In: FASE. Lecture notes in computer science, vol 8411, Springer, Switzerland, pp 1–16, doi: https://doi.org/10.1007/978-3-642-54804-8_1
32. Roijers DM, Whiteson S (2017) Multi-objective decision making. Synth Lect Artif Intell Mach Learn. https://doi.org/10.2200/S00765ED1V01Y201704AIM034
33. Svorenová M, Kwiatkowska M (2016) Quantitative verification and strategy synthesis for stochastic games. Eur J Control 30:15–30. https://doi.org/10.1016/j.ejcon.2016.04.009
34. Basset N, Kwiatkowska MZ, Topcu U, Wiltsche C (2015) Strategy synthesis for stochastic games with multiple long-run objectives. In: TACAS. Lecture notes in computer science, vol 9035, Springer, Switzerland, pp 256–271, doi: https://doi.org/10.1007/978-3-662-46681-0_22
35. Chatterjee K, Doyen L (2016) Perfect-information stochastic games with generalized mean-payoff objectives. In: LICS, ACM, New York, pp 247–256, doi: https://doi.org/10.1145/2933575.2934513
36. Brenguier R, Forejt V (2016) Decidability results for multi-objective stochastic games. In: ATVA. Lecture notes in computer science, vol 9938, pp 227–243, doi: https://doi.org/10.1007/978-3-319-46520-3_15

37. Chen T, Kwiatkowska MZ, Simaitis A, Wiltsche C (2013) Synthesis for multi-objective stochastic games: An application to autonomous urban driving. In: QEST, pp 322–337, doi: https://doi.org/10.1007/978-3-642-40196-1_28
38. Ashok P, Chatterjee K, Kretínský J, Weininger M, Winkler T (2020) Approximating values of generalized-reachability stochastic games. In: LICS, ACM, New York, pp 102–115, doi: https://doi.org/10.1145/3373718.3394761
39. Bruyère V, Hautem Q, Raskin J (2018) Parameterized complexity of games with monotonically ordered omega-regular objectives. In: CONCUR. LIPIcs, vol 118, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Wadern . pp 29–12916, doi: https://doi.org/10.4230/LIPIcs.CONCUR.2018.29
40. Bruyère V, Filiot E, Randour M, Raskin J (2017) Meet your expectations with guarantees: beyond worst-case synthesis in quantitative games. Inf Comput 254:259–295. https://doi.org/10.1016/j.ic.2016.10.011
41. Wray KH, Zilberstein S (2015) Multi-objective POMDPs with lexicographic reward preferences. In: IJCAI, AAAI Press, California, pp 1719–1725, http://ijcai.org/Abstract/15/245
42. Kwiatkowska M, Parker D, Wiltsche C (2018) PRISM-games: verification and strategy synthesis for stochastic multi-player games with multiple objectives. STTT 20(2):195–210. https://doi.org/10.1007/s10009-017-0476-z
43. Brázdil T, Chatterjee K, Forejt V, Kucera A (2015) MultiGain: a controller synthesis tool for MDPs with multiple mean-payoff objectives. In: TACAS. lecture notes in computer science, vol 9035, Springer, Switzerland, pp 181–187, doi: https://doi.org/10.1007/978-3-662-46681-0_12
44. Dehnert C, Junges S, Katoen J, Volk M (2017) A storm is coming: A modern probabilistic model checker. In: CAV (2). Lecture notes in computer science, vol 10427, Springer, Switzerland pp. 592–600, doi: https://doi.org/10.1007/978-3-319-63390-9_31
45. Quatmann T, Junges S, Katoen J (2017) Markov automata with multiple objectives. In: CAV (1). Lecture Notes in Computer Science, vol 10426, Springer, Switzerland, pp 140–159, doi: https://doi.org/10.1007/978-3-319-63387-9_7
46. Hartmanns A, Junges S, Katoen J, Quatmann T (2020) Multi-cost bounded tradeoff analysis in MDP 64:1483–1522. https://doi.org/10.1007/s10817-020-09574-9
47. Pranger S, Könighofer B, Posch L, Bloem R (2021) TEMPEST - synthesis tool for reactive systems and shields in probabilistic environments. In: ATVA. Lecture notes in computer science, vol 12971, Springer, Switzerland, pp 222–228, doi: https://doi.org/10.1007/978-3-030-88885-5_15
48. Tarski A (1955) A lattice-theoretical fixpoint theorem and its applications. Pacific J Math 5(2):285–309. https://doi.org/10.2140/pjm.1955.5.285
49. Forejt V, Kwiatkowska MZ, Parker D (2012) Pareto curves for probabilistic model checking. In: ATVA. Lecture notes in computer science, vol 7561, Springer, Switzerland, pp. 317–332, doi: https://doi.org/10.1007/978-3-642-33386-6_25
50. Fijalkow N, Horn F (2010) The surprizing complexity of reachability games. CoRR **abs/1010.2420**arxiv:1010.2420
51. Winkler T, Weininger M (2021) Stochastic games with disjunctions of multiple objectives. In: GandALF. EPTCS, vol 346, pp 83–100, doi: https://doi.org/10.4204/EPTCS.346.6
52. Kupferman O, Vardi MY (1998)Freedom, weakness, and determinism: From linear-time to branching-time. In: LICS, IEEE Computer Society, Washington, DC, pp 81–92, doi: https://doi.org/10.1109/LICS.1998.705645
53. Sickert S (2019) A unified translation of linear temporal logic to ω-automata. PhD thesis, Technical University of Munich, Germany https://nbn-resolving.org/urn:nbn:de:bvb:91-diss-20190801-1484932-1-4
54. Chatterjee K, Henzinger M (2011) Faster and dynamic algorithms for maximal end-component decomposition and related graph problems in probabilistic verification. In: SODA, SIAM, Philadelphia, pp 1318–1336, doi: https://doi.org/10.1137/1.9781611973082.101
55. Chatterjee K, Dvorák W, Henzinger M, Loitzenbauer V (2016) Model and objective separation with conditional lower bounds: disjunction is harder than conjunction. In: LICS, ACM, New York, pp197–206, doi: https://doi.org/10.1145/2933575.2935304
56. Chatterjee K, Dvorák W, Henzinger M, Loitzenbauer V (2016) Model and objective separation with conditional lower bounds: disjunction is harder than conjunction. CoRR **abs/1602.02670**arxiv:1602.02670
57. Kelmendi E, Krämer J, Kretínský J, Weininger M (2018) Value iteration for simple stochastic games: stopping criterion and learning algorithm. In: CAV (1). Lecture notes in computer science, vol 10981, Springer, Switzerland, pp. 623–642, doi: https://doi.org/10.1007/978-3-319-96145-3_36

58. van Dijk T (2018) Attracting tangles to solve parity games. In: CAV (2). Lecture notes in computer science, vol 10982, Springer, Switzerland, pp 198–215, doi: https://doi.org/10.1007/978-3-319-96142-2_14
59. Ujma M (2015) On verification and controller synthesis for probabilistic systems at runtime. PhD thesis, University of Oxford, UK http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.711811
60. Löding C (1999) Optimal bounds for transformations of omega-automata. In: FSTTCS. Lecture notes in computer science, vol 1738, Springer, Switzerland, pp 97–109, doi: https://doi.org/10.1007/3-540-46691-6_8
61. Müller D, Sickert S (2017) LTL to deterministic emerson-lei automata. In: GandALF. EPTCS, vol 256, pp 180–194, doi: https://doi.org/10.4204/EPTCS.256.13
62. Littman ML, Cassandra AR, Kaelbling LP (1995) Learning policies for partially observable environments: Scaling up. In: ICML, Morgan Kaufmann, Massachusetts, pp 362–370, doi: https://doi.org/10.1016/b978-1-55860-377-6.50052-9
63. Chatterjee K, Chmelik M, Gupta R, Kanodia A (2016) Optimal cost almost-sure reachability in POMDPs. Artif Intell 234:26–48. https://doi.org/10.1016/j.artint.2016.01.007
64. Chatterjee K, Chmelík M (2015) POMDPs under probabilistic semantics. Artif Intell 221:46–72. https://doi.org/10.1016/j.artint.2014.12.009
65. Duret-Lutz A, Lewkowicz A, Fauchille A, Michaud T, Renault E, Xu L (2016) Spot 2.0 - A framework for LTL and ω-automata manipulation. In: ATVA. Lecture notes in computer science, vol 9938, pp 122–129, doi: https://doi.org/10.1007/978-3-319-46520-3_8
66. Kwiatkowska M, Norman G, Parker D, Vigliotti MG (2009) Probabilistic mobile ambients. Theor Comput Sci 410(12):1272–1303. https://doi.org/10.1016/j.tcs.2008.12.058
67. Feng L, Wiltsche C, Humphrey LR, Topcu U (2015) Controller synthesis for autonomous systems interacting with human operators. In: ICCPS, ACM, New York, pp 70–79, doi: https://doi.org/10.1145/2735960.2735973
68. Chatterjee K, Piterman N (2019) Combinations of qualitative winning for stochastic parity games. In: CONCUR. LIPIcs, vol 140, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Wadern, pp 6–1617, doi: 0.4230/LIPIcs.CONCUR.2019.6

Authors and Affiliations

Krishnendu Chatterjee[1] · Joost-Pieter Katoen[2] · Stefanie Mohr[3] · Maximilian Weininger[3] · Tobias Winkler[2]

Krishnendu Chatterjee
krishnendu.chatterjee@ist.ac.at

Joost-Pieter Katoen
katoen@cs.rwth-aachen.de

Stefanie Mohr
mohr@in.tum.de

Maximilian Weininger
maxi.weininger@tum.de

1 IST Austria, Klosterneuburg, Austria

2 RWTH Aachen University, Aachen, Germany

3 Technical University of Munich, Munich, Germany