Alternating Good-for-MDPs Automata

Ernst Moritz Hahn[1], Mateo Perez[2], Sven Schewe[3], Fabio Somenzi[2], Ashutosh Trivedi[2], and Dominik Wojtczak[3(✉)]

[1] University of Twente, Enschede, The Netherlands
[2] University of Colorado Boulder, Boulder, USA
[3] University of Liverpool, Liverpool, UK
d.wojtczak@liverpool.ac.uk

Abstract. When omega-regular objectives were first proposed in model-free reinforcement learning (RL) for controlling MDPs, deterministic Rabin automata were used in an attempt to provide a direct translation from their transitions to scalar values. While these translations failed, it has turned out that it is possible to repair them by using good-for-MDPs (GFM) Büchi automata instead. These are nondeterministic Büchi automata with a restricted type of nondeterminism, albeit not as restricted as in good-for-games automata. Indeed, deterministic Rabin automata have a pretty straightforward translation to such GFM automata, which is bi-linear in the number of states and pairs. Interestingly, the same cannot be said for deterministic Streett automata: a translation to nondeterministic Rabin or Büchi automata comes at an exponential cost, even without requiring the target automaton to be good-for-MDPs. Do we have to pay more than that to obtain a good-for-MDPs automaton? The surprising answer is that we have to pay significantly less when we instead expand the good-for-MDPs property to alternating automata: like the nondeterministic GFM automata obtained from deterministic Rabin automata, the alternating good-for-MDPs automata we produce from deterministic Streett automata are bi-linear in the size of the deterministic automaton and its index. They can therefore be exponentially more succinct than the minimal nondeterministic Büchi automaton.

1 Introduction

Omega-automata [18,27] have found renewed interest—often as the result of translating a formula in LTL [20]—as specifications of qualitative objectives in reinforcement learning (RL) [26]. The acceptance condition of an ω-automaton determines the reward whose cumulative return the learning agent strives to maximise. The relation between the automaton and the reward signal should ensure that a strategy that maximises the expected return also maximises the probability to realise the objective. This so-called *faithfulness* requirement [10] restricts the type of ω-automaton that can be used to represent the objective, and this paper concerns how to find the right type of ω-automata.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements 864075 (CAESAR), and 956123 (FOCETA). This work is supported in part by the National Science Foundation grant 2009022, by a CU Boulder Research and Innovation Office grant, and by the EPSRC through grant EP/V026887/1.

A. Bouajjani et al. (Eds.): ATVA 2022, LNCS 13505, pp. 303–319, 2022.
https://doi.org/10.1007/978-3-031-19992-9_19

Deterministic automata with various types of acceptance conditions have been used in model checking and strategy synthesis [2]; notably, Büchi, parity, Rabin, and Streett. While deterministic Büchi automata—including generalised deterministic Büchi automata—do not accept all ω-regular languages, deterministic parity, Rabin, and Streett automata do; therefore, they are employed in the formulation of general solutions to synthesis problems. In addition, maximising the chance of meeting a parity and Rabin winning conditions in a game can be obtained using positional strategies, while Streett winning conditions require finite additional memory. This means that a positional strategy for a Markov decision process (MDP) or a stochastic game endowed with a parity or Rabin objective can be turned into a positional strategy to resolve nondeterministic choice. Strategy computation methods for both Rabin and Streett automata have been studied extensively [1,4,19]. These methods, however, are not applicable in RL. In order to apply RL to the computation of optimal strategies for ω-regular objectives, we have to devise a scheme for doling out rewards that, for generality, depend on the given ω-automaton, and perhaps on some hyperparameters of the learning algorithm, but not on the MDP—or the stochastic game—for which a control strategy is sought.

Two features of RL algorithms significantly affect the choice of translation from acceptance condition to rewards: 1) they require positional optimal strategies after the translation to rewards, as they learn values of states and transitions, and 2) the same transitions will always be optimal once a property is translated into scalar rewards. This appears to effectively exclude using Streett conditions directly [13], as optimal control requires memory for Streett objectives. This is regrettable, as Streett objectives do occur in practice. GR(1) [3] conditions, for example, translate smoothly into Streett objectives (in the pure original form, to one pair Streett objectives), such that a conjunction of GR(1) objectives will always have a natural representation as a deterministic Streett automaton. Likewise, each strong fairness requirement produces a Streett pair. Moreover, minimising the chance of satisfying a Rabin condition given as a deterministic Rabin automaton (DRA) is also equivalent to maximising the chance of satisfying the Streett condition given by its dual.

A natural way to move to simpler acceptance conditions requires some form of nondeterminism. Full recourse to nondeterminism, however, is not compatible with the computation of optimal strategies for MDPs or stochastic games. If we want to move away from using deterministic automata to describe the objective, we therefore need to impose restrictions on the automaton's nondeterminism. The precise nature of these restrictions depends on the type of environment interacting with the agent, whose control strategy we want to build. If the environment is a Markov decision process [21], the automaton needs to be good-for-MDPs (GFM) [11] (previously used automata [5,8,28] have this property) while, for stochastic games, with two strategic players, the stronger requirements of good-for-games automata [15] must be satisfied. GFM automata have the advantage that they can use simpler acceptance mechanisms. In particular, the GFM automata developed so far are nondeterministic Büchi automata, and being able to use a simple acceptance mechanism like Büchi is quite beneficial for RL [9,11]—though it is possible to use parity automata, using them comes at a cost [12].

When starting with a deterministic Streett automaton (DSA), a translation to a nondeterministic Büchi automaton (NBA) [23], or even to a nondeterministic Rabin

automaton [23], comes at the cost of an exponential blow-up, even without the restriction to GFM automata. This raises the question of whether or not there is a different way to efficiently translate the DSA into a suitable automaton. Our main result is that **alternating GFM automata can be exponentially more succinct than general nondeterministic Büchi automata**.

Intuitively, this should not be possible: the reason for the exponential blow-up from DSAs to NBAs (and even NRAs) is that one will either need some form of memory, such as a latest appearance record (LAR [6,7]), or a nondeterministic guess as to which way each Streett pair is satisfied in. Recall that a Streett pair consists of a green and a red set of states or transitions, and it is satisfied if no entry of the red set *or* some entry of the green set occurs infinitely often; a nondeterministic automaton can guess, for each pair, to validate either of these conditions. The intuitive effect of this blow-up would be that starting with DSAs is something that efficient RL approaches will struggle with.

The key message of this paper is that one can trade-off *alternation* for memory in computing an optimal strategy by moving to *alternating* GFM automata instead of traditional nondeterministic ones. Here is the interesting bit: while we do, unsurprisingly, need a latest appearance record in the control strategy we develop, the blow-up due to the LAR is not necessary while learning the optimal strategy!

2 Preliminaries

A *probability distribution* over a finite set S is a function $d\colon S \to [0,1]$ such that we have $\sum_{s\in S} d(s) = 1$. Let $\mathcal{D}(S)$ denote the set of all probability distributions over S. We say a distribution $d \in \mathcal{D}(S)$ is a *point distribution* if $d(s) = 1$ for some $s \in S$. For $d \in \mathcal{D}(S)$ we write $supp(d)$ for $\{s \in S\colon d(s) > 0\}$.

2.1 Stochastic Game Arenas and Markov Decision Processes

A *stochastic game arena* $\mathcal{G}$ is a tuple $(S, s_0, A, T, S_{\mathrm{Max}}, S_{\mathrm{Min}}, AP, L)$, where S is a finite set of states, $s_0 \in S$ is the initial state, A is a finite set of *actions*, $T\colon S \times A \rightharpoonup \mathcal{D}(S)$ is the *probabilistic transition (partial) function*, $\{S_{\mathrm{Max}}, S_{\mathrm{Min}}\}$ is a partition of the set of states S, AP is the set of *atomic propositions*, and $L\colon S \to 2^{AP}$ is the *labeling function*. For $s \in S$, $A(s)$ denotes the set of actions enabled in s. For states $s, s' \in S$ and $a \in A(s)$ we write $p(s'|s,a)$ for $T(s,a)(s')$.

A *run* of $\mathcal{G}$ is an ω-word $\langle s_0, a_1, s_1, \ldots\rangle \in S \times (A \times S)^\omega$ such that $p(s_{i+1}|s_i, a_{i+1}) > 0$ holds for all $i \geq 0$. A finite run is a finite such sequence, that is, a word in $S \times (A \times S)^*$. For a *run* $r = \langle s_0, a_1, s_1, \ldots\rangle$ we define the corresponding labeled run as $L(r) = \langle L(s_0), L(s_1), \ldots\rangle \in (2^{AP})^\omega$. We write $Runs^{\mathcal{G}}$ ($FRuns^{\mathcal{G}}$) for the set of runs (finite runs) of the SGA $\mathcal{G}$ and $Runs^{\mathcal{G}}(s)$ ($FRuns^{\mathcal{G}}(s)$) for the set of runs (finite runs) of the SGA $\mathcal{G}$ starting from state s. We write $last(r)$ for the last state of a finite run r.

A game on an SGA $\mathcal{G}$ is played between two players, Max and Min, by moving a token through the states of the arena. The game begins with a token in an *initial state* s_0; players Max and Min construct an infinite run by taking turns to choose enabled actions when the token is in a state controlled by them, and then moving the token to a successor state sampled from the selected distribution. A strategy of player Max in

$\mathcal{G}$ is a partial function $\pi\colon FRuns \rightharpoonup \mathcal{D}(A)$, defined for $r \in FRuns$ if, and only if, $last(r) \in S_{\mathrm{Max}}$, such that $supp(\sigma(r)) \subseteq A(last(r))$. A strategy σ of player Min is defined analogously. We drop the subscript $\mathcal{G}$ when the arena is clear from the context. Let $\Sigma_{\mathcal{G}}$ and $\Pi_{\mathcal{G}}$ be the sets of all strategies of player Max and player Min, respectively.

A memory structure for $\mathcal{G}$ is a tuple $\mathbf{M} = (M, m_0, \alpha_u)$ where M is a finite set of memory states, $m_0 \in M$ is the initial state, and $\alpha_u\colon M \times 2^{AP} \to M$ is the memory update function. The extended memory update $\hat{\alpha}_u\colon M \times (2^{AP})^* \to M$ can be defined in the usual manner. A finite memory strategy of player Max in $\mathcal{G}$ over a memory structure $\mathbf{M}$ is a Mealy machine $(\mathbf{M}, \alpha_x)$ where $\alpha_x\colon S_{\mathrm{Max}} \times M \to \mathcal{D}(A)$ is the *next action function* that suggests the next action based on the SGA and the memory state. The semantics of a finite memory strategy $(\mathbf{M}, \alpha_x)$ is given as a strategy $\sigma \in \Sigma_{\mathcal{G}}$ such that for every $r \in FRuns$ with $last(r) \in S_{\mathrm{Max}}$, we have that $\sigma(r) = \alpha_x(last(r), \hat{\alpha}_u(m_0, L(r)))$.

A strategy σ is *pure* if $\sigma(r)$ is a point distribution wherever it is defined; otherwise, σ is *mixed*. We say that σ is *stationary* if $last(r) = last(r')$ implies $\sigma(r) = \sigma(r')$ wherever σ is defined. A strategy is *positional* if it is both pure and stationary. We write $\overline{\Sigma}_{\mathcal{G}}$ and $\overline{\Pi}_{\mathcal{G}}$ for the sets of all *positional* strategies of player Max and player Min, respectively.

Let $Runs^{\mathcal{G}}_{\sigma,\pi}(s)$ denote the subset of runs $Runs^{\mathcal{G}}(s)$ starting from state s that are consistent with player Max and player Min following strategies σ and π, respectively. The behaviour of an SGA $\mathcal{G}$ under a strategy pair $(\sigma, \pi) \in \Sigma_{\mathcal{G}} \times \Pi_{\mathcal{G}}$ is defined on the probability space $(Runs^{\mathcal{G}}_{\sigma,\pi}(s), \mathcal{F}_{Runs^{\mathcal{G}}_{\sigma,\pi}(s)}, \Pr^{\mathcal{G}}_{\sigma,\pi}(s))$ over the set of infinite runs $Runs^{\mathcal{G}}_{\sigma,\pi}(s)$, where $\mathcal{F}_{Runs^{\mathcal{G}}_{\sigma,\pi}(s)}$ is the standard σ-algebra over them. Given a random variable $f\colon Runs^{\mathcal{G}} \to \mathbb{R}$ over the infinite runs of $\mathcal{G}$, we denote by $\mathbb{E}^{\mathcal{G}}_{\sigma,\pi}(s)\{f\}$ the expectation of f over the runs in the probability space $(Runs^{\mathcal{G}}_{\sigma,\pi}(s), \mathcal{F}_{Runs^{\mathcal{G}}_{\sigma,\pi}(s)}, \Pr^{\mathcal{G}}_{\sigma,\pi}(s))$.

We say that an SGA is a Markov decision process if $A(s)$ is a singleton for every $s \in S_{\mathrm{Min}}$ and is a Markov chain if $A(s)$ is singleton for every $s \in S$. To distinguish an MDP from an SGA, we denote an MDP by $\mathcal{M}$ and write its signature $\mathcal{M} = (S, s_0, A, T, AP, L)$ by assigning the (choiceless) states of player Min to player Max. The notions defined for SGAs naturally carry over to MDPs.

2.2 Omega-Automata

An alphabet is a finite set of letters. We write $\mathbb{B}$ for the binary alphabet $\{0, 1\}$. A finite word over an alphabet Σ is a finite concatenation of symbols from Σ. Similarly, an ω-word w over Σ is a function $w\colon \omega \to \Sigma$ from the natural numbers to Σ. We write Σ^* and Σ^ω for the set of finite and ω-words over Σ.

An ω-automaton $\mathcal{A} = (\Sigma, Q, q_0, \delta, \alpha)$ consists of a finite alphabet Σ, a finite set of states Q, an initial state $q_0 \in Q$, a transition function $\delta\colon Q \times \Sigma \to 2^Q$, and an acceptance condition $\alpha\colon Q^\omega \to \mathbb{B}$. A *deterministic* automaton is such that $\delta(q, \sigma)$ is a singleton for every state q and alphabet letter σ. For deterministic automata, we write $\delta(q, \sigma) = q'$ instead of $\delta(q, \sigma) = \{q'\}$.

A *run* of an automaton $\mathcal{A} = (\Sigma, Q, q_0, \delta, \alpha)$ on word $w \in \Sigma^\omega$ is a function $\rho\colon \omega \to Q$ such that $\rho(0) = q_0$ and $\rho(i+1) \in \delta(\rho(i), w(i))$. A run ρ is *accepting* if $\alpha(\rho) = 1$. A

word w is accepted by $\mathcal{A}$ if there exists an accepting run of $\mathcal{A}$ on w. The language of $\mathcal{A}$, written $\mathcal{L}(\mathcal{A})$, is the set of words accepted by $\mathcal{A}$. The set of states that appear infinitely often in ρ is written $\inf(\rho)$. A deterministic automaton $\mathcal{D}$ has exactly one run for each word in Σ^ω. We write $\inf^{\mathcal{D}}(w)$ for the set of states that appear infinitely often in the unique run of $\mathcal{D}$ on w; when clear from the context, we drop the superscript and simply write $\inf(w)$.

Several ways to give finite presentations of the acceptance conditions are in use. The ones relevant to this paper are listed below.

- A *Büchi* acceptance condition is specified by a set of states $F \subseteq Q$ such that

$$\alpha(\rho) = [\,\inf(\rho) \cap F \neq \emptyset\,].$$

- A *Rabin* acceptance condition of index k is specified by k pairs of sets of states, $\{(R_i, G_i)\}_{1 \leq i \leq k}$, and intuitively a run should visit at least one set of Red (ruinous) states finitely often and its corresponding Green (good) set of states infinitely often. Formally,

$$\alpha(\rho) = [\,\exists\, 1 \leq i \leq k.\, \inf(\rho) \cap R_i = \emptyset \text{ and } \inf(\rho) \cap G_i \neq \emptyset\,].$$

- A *Streett* acceptance condition of index k is specified by k pairs of sets of states, $\{(G_i, R_i)\}_{1 \leq i \leq k}$, and intuitively a run should visit each Red set of states finitely often or its corresponding Green set of states infinitely often. Formally,

$$\alpha(\rho) = [\,\forall\, 1 \leq i \leq k.\, \inf(\rho) \cap R_i = \emptyset \text{ or } \inf(\rho) \cap G_i \neq \emptyset\,].$$

We also allow for moving the acceptance condition from states to transitions. For a Büchi acceptance condition, this means defining a set $F \subseteq Q \times \Sigma \times Q$ of final transitions, where the i^{th} transition for a run ρ of a word w is $t(i) = \big(\rho(i), w(i), \rho(i+1)\big)$, $\inf_t(\rho, w)$ is the set of transitions that occur infinitely often, and a run ρ is accepting for w if $\inf_t(\rho, w) \cap F \neq \emptyset$.

2.3 Semantic Satisfaction: Optimal Strategies Against ω-Automata

Given an MDP $\mathcal{M} = (S, s_0, A, T, AP, L)$ and an ω-automaton $\mathcal{A} = (2^{AP}, Q, q_0, \delta, \alpha)$, we are interested in strategies that maximise the probability that the labels of a run of $\mathcal{M}$ form an ω-word in the language of $\mathcal{A}$. A strategy $\sigma \in \Sigma_{\mathcal{M}}$ and initial state $s \in S$ determine a sequence X_i of random variables denoting the i^{th} state of the MDP, where $X_0 = s$.

We define the optimal satisfaction probability $\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s)$ as

$$\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s) = \sup_{\sigma \in \Sigma_{\mathcal{M}}} \Pr{}^{\mathcal{M}}_{\sigma}(s)\,\{\langle L(X_0), L(X_1), \ldots\rangle \in \mathcal{L}(\mathcal{A})\}\,.$$

We say that a strategy $\sigma \in \Sigma_{\mathcal{M}}$ is optimal for $\mathcal{A}$ if

$$\Pr{}^{\mathcal{M}}_{\sigma}(s)\,\{\langle L(X_0), L(X_1), \ldots\rangle \in \mathcal{L}(\mathcal{A})\} = \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s)$$

for all $s \in S$.

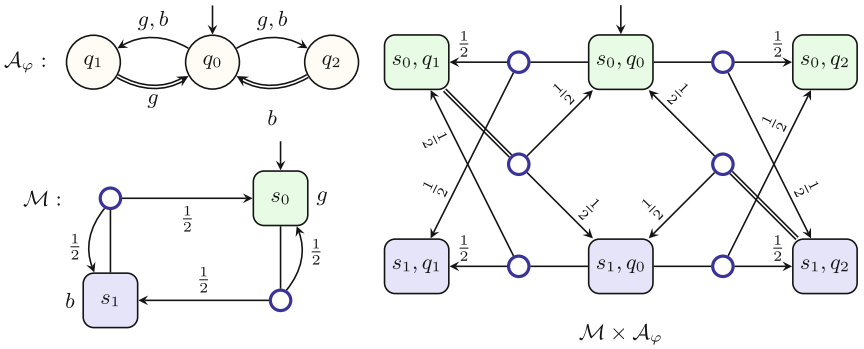

Fig. 1. Syntactic and Semantic Probabilities Differ. A nondeterministic Büchi automaton $\mathcal{A}_\varphi$ (top left) that recognises the language of infinitely many g's or infinitely many b's (which accepts all ω-words) and a Markov decision process $\mathcal{M}$, whose set of actions is a singleton (a Markov chain). Note that double edges mark accepting transitions. Notice that the MDP $\mathcal{M}$ satisfies the property with probability 1. Their product is shown on the right side, where there is no accepting end-component. Hence, the probability of reaching an accepting end-component (under any strategy) is 0.

2.4 Good-for-MDPs Automata

Given an MDP $\mathcal{M} = (S, s_0, A, T, AP, L)$ and automaton $\mathcal{A} = (2^{AP}, Q, q_0, \delta, \alpha)$, the *probabilistic model checking* problem is to find the optimal value $\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s_0)$ and an optimal strategy in $\Sigma_{\mathcal{M}}$. An intuitive way to compute $\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s_0)$ is to build the synchronous product $\mathcal{M} \times \mathcal{A}$ of $\mathcal{M}$ and $\mathcal{A}$ and compute the optimal probability to satisfy the acceptance condition in the product (we will name this syntactic satisfaction probability $\mathsf{PSyn}^{\mathcal{M}}_{\mathcal{A}}(s_0, q_0)$) and a strategy that maximises the probability of satisfying the acceptance condition. If these values $\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s)$ and $\mathsf{PSyn}^{\mathcal{M}}_{\mathcal{A}}(s)$ coincide for all possible $\mathcal{M}$, then the automaton is said to be *good-for-MDPs* [11].

The synchronous product is an MDP $\mathcal{M} \times \mathcal{A} = (S \times Q, (s_0, q_0), A \times Q, T^\times, AP, L)$, where

$$T^\times((s,q),(a,q'))(s',q'') = \begin{cases} T(s,a)(s') & \text{if } q' \in \delta(q, L(s)) \text{ and } q'' = q' \\ 0 & \text{if } q' \in \delta(q, L(s)) \text{ and } q'' \neq q' \\ \text{undefined} & \text{otherwise .} \end{cases}$$

A strategy $\sigma \in \Sigma_{\mathcal{M}\times\mathcal{A}}$ and initial state $(s, q) \in S \times Q$ determine a sequence (X_i, Q_i) of random variables denoting the i^{th} state of the product MDP, where $X_0 = s$ and $Q_0 = q$. The syntactic probability is defined to be

$$\mathsf{PSyn}^{\mathcal{M}}_{\mathcal{A}}(s,q) = \sup_{\sigma \in \Sigma_{\mathcal{M}\times\mathcal{A}}} \mathbb{E}^{\mathcal{M}\times\mathcal{A}}_{\sigma}(s)\left\{\alpha(\langle Q_0, Q_1, \ldots\rangle)\right\}.$$

An automaton is good-for-MDPs if $\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s, q_0) = \mathsf{PSyn}^{\mathcal{M}}_{\mathcal{A}}(s)$ for all MDPs $\mathcal{M}$ and states $s \in S$. Figure 1 shows an example of an ω-automaton with Büchi acceptance

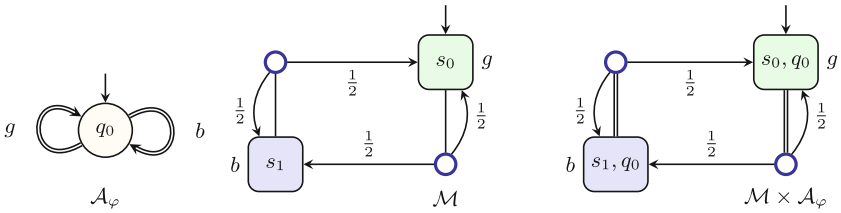

Fig. 2. Syntactic and Semantic Probabilities Agree. A deterministic Büchi automaton $\mathcal{A}_\varphi$ (left) that recognises the language of infinitely many g's or infinitely many b's (accepts all ω-words) and an MDP $\mathcal{M}$ (center), whose set of actions is singleton (a Markov chain). Again double edges mark accepting transitions. Notice that the MDP $\mathcal{M}$ satisfies the property with probability 1. Their product is shown on the right side where the whole MDP is one accepting end-component. Hence, the probability of reaching the end-component (under any strategy) is 1.

condition that is not GFM, while Fig. 2 shows an automaton that is GFM (since every deterministic ω-automaton is GFM.).

The advantage of being able to work on the syntactic product MDP is that the goal turns into reaching an accepting end-component (an end-component is a region of the MDP that, once entered, can be covered—each state visited infinitely often—almost surely while surely never leaving it; for Markov chains, these are the accepting leaf components) [11].

2.5 GFM Büchi Automata and Reinforcement Learning

The limit reachability technique [9] reduces the model checking problem for given MDP and GFM Büchi automaton to a reachability problem by slightly changing the structure of the product: one adds a target state t that can be reached with a given probability $1-\zeta$ whenever visiting an accepting transition of the original product MDP. This reduction avoids the identification of accepting end-components and thus allows a natural integration to a wide range of model-free RL approaches. Thus, while the proofs do lean on standard model checking properties that are based on identifying winning end-components, they serve as a justification not to consider them when running the learning algorithm.

For any $\zeta \in (0,1)$, the *augmented MDP* $\mathcal{M}^\zeta$ is an MDP obtained from $\mathcal{M} \times \mathcal{A}$ by adding a sink state t with a self-loop to the set of states of $\mathcal{M} \times \mathcal{A}$, and by making t a destination of each accepting transition τ of $\mathcal{M} \times \mathcal{A}$ with probability $1-\zeta$. The original probabilities of all other destinations of an accepting transition τ are multiplied by ζ. An example of an augmented MDP is shown in Fig. 3. With a slight abuse of notation, if σ is a strategy on the augmented MDP $\mathcal{M}^\zeta$, we denote by σ also the strategy on $\mathcal{M} \times \mathcal{A}$ obtained by removing t from the domain of σ. The following result shows the correctness of the construction.

Theorem 1 (Limit Reachability Theorem [9,11]). *If $\mathcal{A}$ is GFM, then there exists a threshold $\zeta' \in (0,1)$ such that, for all $\zeta > \zeta'$ and every state s, any strategy σ that maximises the probability of reaching the sink in $\mathcal{M}^\zeta$ is (1) an optimal strategy in*

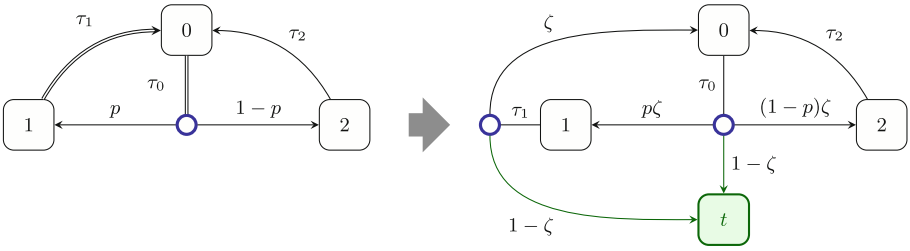

Fig. 3. Adding transitions to the target in the augmented product MDP.

$\mathcal{M} \times \mathcal{A}$ from s and (2) induces an optimal strategy for the original MDP $\mathcal{M}$ from s with the objective to produce a run in the language of $\mathcal{A}$. Moreover, $\mathcal{M}$ produces such a run almost surely if, and only if, the sink is almost surely reachable in $\mathcal{M}^\zeta$ for all $0 < \zeta < 1$.

Theorem 1 leads to a very simple model-free RL algorithm for GFM Büchi automata. The augmented product is not built by the RL algorithm, which does not know the transition structure of the environment MDP. Instead, the observations are used to drive the objective automaton. When the automaton reports an accepting transition, the interpreter tosses a biased coin to give the learner a reward with probability $1 - \zeta$. The interpreter also extracts the set of actions for the learner to choose from. If the automaton is not deterministic and it has not taken the one nondeterministic transition it needs to take yet, the set of actions the interpreter provides to the learner includes the choice of special "jump" actions that instruct the automaton to move to a chosen accepting component. When the automaton reports an accepting transition, the interpreter gives the learner a positive reward with probability $1 - \zeta$. When the learner actually receives a reward, the training episode terminates. Any RL algorithm that maximises this probabilistic reward is guaranteed to converge to a policy that maximises the probability of satisfaction of the objective.

3 Alternating GFM Automata

Before giving a translation from Deterministic Streett automaton (DSA) to a good-for-MDPs (GFM) alternating Büchi automaton (ABA), let us see how simple the translation is for its dual, a Deterministic Rabin Automaton (DRA). When we start with a DRA $\mathcal{R}$, the translation to a GFM Nondeterministic Büchi Automaton (NBA) [11] is straightforward as shown next.

Definition 1 (DRA to GFM NBA). *For a given deterministic Rabin automaton $\mathcal{R} = (\Sigma, Q, q_0, \delta, \{\langle R_i, G_i\rangle \mid i \in \{1, \ldots, k\}\})$, we construct a nondeterministic GFM automaton $\mathcal{B} = (\Sigma, Q \times \{0, \ldots, k\} \cup \{\bot\}, (q_0, 0), \delta', F)$ where:*

$$\delta'(\hat{q}, a) = \begin{cases} \{\delta(q,a)\} \times \{0, \ldots, k\} & \text{if } \hat{q} = (q, 0) \\ \{(\delta(q,a), i)\} & \text{if } \hat{q} = (q,i), i \neq 0, \text{ and } q \notin R_i \\ \bot & \text{otherwise.} \end{cases}$$

and $F = \{(q, i) \mid i \in \{1, \ldots, k\}$ *and* $q \in G_i\}$.

The resulting NBA makes only a single guess: it guesses when an accepting end-component is reached in the product MDP $\mathcal{M} \times \mathcal{R}$ (noting that the 0-copy is congruent to the original automaton), and then moves to a copy i, whose pair makes this end-component accepting. It is easy to see that this automaton is language equivalent to $\mathcal{R}$ and good-for-MDPs (e.g., it satisfies the simulation condition from [11]). For k pairs, this creates only $k+1$ copies, and thus a small overhead; and it allows one to then use standard reward translation techniques for Büchi acceptance conditions [9] in RL.

The question of how to maximise the probability to satisfy a Streett condition (or, likewise, how to minimise the probability to satisfy a Rabin condition) is more challenging. Broadly speaking, the translation of a Rabin acceptance condition is simplified by the fact that the nondeterministic choices of an NBA can easily handle the resolution of the disjunction of the acceptance condition on pairs, and resolving nondeterminism is something that always needs to be done when analysing an MDP. However, it is harder to accommodate for a conjunction of the acceptance condition on pairs, as in a Streett acceptance condition. As a consequence, the translation of a deterministic Streett automaton to a nondeterministic Rabin automaton (without a restriction to GFM) leads to a blow-up that results in $2^{\theta(n)}$ states [23], while a translation to an NBA requires $n2^{\theta(k)}$ states [23], even without the restriction to GFM.

Surprisingly, there is a way to exploit alternating good-for-MDPs automata with a small blow-up of $k+2$ *for* k *pairs.*

As Büchi games can be handled with similar techniques as for Büchi MDPs (Sect. 2.5) in model-free reinforcement learning (cf. [12]), the alternation itself does not create problems during learning; still, it is quite surprising that such a method works. This is partly because of the exponential memory requirement for Streett conditions, and partly because the acceptance player for the MDP would not have access to decisions the rejection player has made in the resulting game. However, while the automaton is small, the memory we infer from the winning strategy of this small automaton can be exponentially larger.

An optimal strategy in the resulting game does not in itself constitute a strategy for controlling the MDP for a given DSA. This is because different strategic choices of the antagonistic rejection player will lead to different positions in the game, and there is no guarantee that a consistent positional strategy for all of these positions exists. Moreover, the strategic choices of an antagonistic rejection player have no direct relation to the observable history. We show, however, that the history can be used to identify a state in the game, whose decisions the acceptance player should follow.

The need for memory is, therefore, not gone. Instead, the control strategy we construct in the correctness proof for the resulting alternating GFM Büchi automaton is only one part of the control strategy used for the MDP. The other is a latest appearance record (LAR), which is kept in addition to the constructed game. The LAR will determine the state, from a family of equivalent states, whose strategy will be followed.

3.1 Alternating GFM Automata

There is a number of mildly different definitions of alternating automata, and we can use the simplest one, where the states are partitioned into nondeterministic and universal states.

Definition 2. *An alternating ω-automaton $\mathcal{A} = (\Sigma, Q_n, Q_u, q_0, \delta, \alpha)$, with $Q = Q_n \cup Q_u$, is an automaton such that $(\Sigma, Q, q_0, \delta, \alpha)$ is a nondeterministic automaton, and Q_n and Q_u are disjoint sets of nondeterministic and universal states, respectively.*

A *run tree* of an alternating automaton $\mathcal{A} = (\Sigma, Q_n, Q_u, q_0, \delta, \alpha)$ on word $w \in \Sigma^\omega$ is a family of functions $\{\rho_j \colon \omega \to Q \mid j \in J\}$ for some non-empty index set J such that

- ρ_j is a run for all $j \in J$, and
- if ρ_j has a universal state q' at a position $i \in \omega$ ($\rho_j(i) = q' \in Q_u$), then, for all $q \in \delta\big(q', w(i)\big)$, there is a $j_q \in J$ such that $\rho_{j_q}(k) = \rho_j(k)$ for all $k \leq i$, and $\rho_{j_q}(i+1) = q$.

A run tree is accepting if all of the runs of $\{\rho_j \colon \omega \to Q \mid j \in J\}$ are accepting.

A minimal such family of runs can be viewed as a tree, where nondeterministic states have one successor, while universal states have many, namely all those defined by the local successor function. Alternatively, a family of runs can be viewed as a game, where an angelic acceptance player chooses the successor for a nondeterministic state, while an antagonistic rejection player selects the successor for a universal state. This way, they successively construct a run, and acceptance is decided by whether or not this run accepts.

We extend the product construction from Sect. 2.4 to produce a Büchi game from the product $\mathcal{M} \times \mathcal{A}$ of an MDP $\mathcal{M}$ with an alternating Büchi automaton $\mathcal{A}$, where the decisions of the rejection player are simply the decision to resolve the nondeterminism from the universal states, while resolving the nondeterminism from the MDP *and* resolving the nondeterminism from the nondeterministic automata states are left to the acceptance player. Both players have positional optimal strategies (where, for the rejection player, positionality includes the state and the choice made by the acceptance player[1]) in this game [17].

We refer to the probability, with which the acceptance player can win this product game from a product state (s, q) as

$$\mathsf{PSyn}^{\mathcal{M}}_{\mathcal{A}}(s,q) = \sup_{\sigma \in \Sigma_{\mathcal{M}\times\mathcal{A}}} \inf_{\pi \in \Pi_{\mathcal{M}\times\mathcal{A}}} \mathbb{E}^{\mathcal{M}\times\mathcal{A}}_{\sigma,\pi}(s,q)\left\{\alpha(\langle X_0, X_1, \ldots\rangle)\right\} \ ,$$

where α is the Büchi condition and X_i is the random variable corresponding to the state of the automaton at the i-th step.

Definition 3 (Alternating GFM Automata). *An alternating automaton $\mathcal{A}$ is* good-for-MDPs *if, for all MDPs $\mathcal{M}$, $\mathsf{PSyn}^{\mathcal{M}}_{\mathcal{A}}(s_0, q_0) = \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s_0)$ holds, where s_0 is the initial state of $\mathcal{M}$.*

[1] In a product of an MDP state with a universal automaton state, there needs to be a fixed order of who chooses first. It is common (and more natural) in RL to first resolve the choice of the action selected in the MDP.

3.2 Construction of the Alternating Büchi Automaton

The motivation for the translation of a deterministic Streett automaton to a GFM automaton is similar to that for Rabin: when having nondeterministic power, we can use it to guess when we have reached an accepting end-component that we plan to cover completely (i.e., we will almost surely visit every state and every transition in the end-component infinitely often).

While covering an accepting end-component may require memory (or randomisation), its properties with respect to the Streett condition are straightforward: for every Streett pair $\langle G, R\rangle$, if the end-component contains a red state $q \in R$ then it must also contain a green state $q' \in G$ from the same pair, which should (almost surely) be visited after every visit of q.

Definition 4 (DSA to Alternating GFM Büchi). *For a given deterministic Streett automaton $\mathcal{S} = (\Sigma, Q, q_0, \delta, \{\langle G_i, R_i\rangle \mid i \in \{1, \dots, k\}\})$, where we assume without loss of generality that $G_i \cap R_i = \emptyset$ for all $i = 1, \dots, k$, we construct an alternating Büchi automaton $\mathcal{A} = (\Sigma, Q, Q \times \{0, \dots, k\}, q_0, \delta', F)$ where:*

- *First, for every state $q \in Q$, we let $I_q = \{0\} \cup \{i \mid q \in R_i\}$.*
- *We now define, for every state $q \in Q$ and letter $a \in \Sigma$, where $q' = \delta(q, a)$:*
  - *$\delta'(q, a) = \{q', (q', 0)\}$ and,*
  - *for all $i = 0, \dots, k$, $\delta'((q, i), a) = \{q'\} \times (I_{q'} \setminus \{i\})$ if $q' \in G_i$ and $\delta'((q, i), a) = \{q'\} \times (I_{q'} \cup \{i\})$ if $q' \notin G_i$, using $G_0 = \emptyset$.*
- *Finally, we set the set of final transitions to $F = \{(q, i), a, (q', j) \mid i \neq j$ or $i = j = 0\}$.*

Note that the projection on the state of $\mathcal{S}$ is not affected by this translation.

The intuition for this translation is that the acceptance game starts in the original copy of the states—the nondeterministic states Q. From there, the acceptance player can *declare* when he has reached an accepting end-component, moving from the original copy to the 0-copy of the game. The rejection player can henceforth, whenever a state from the i^{th} red set R_i is seen, move from a j-copy to the i-copy, which can be viewed as a claim that the requirement on the i^{th} Streett pair is not fulfilled (finitely many R_i or infinitely many G_i states). She therefore *challenges* the acceptance player to visit a state from the i^{th} green set G_i (an i-challenge for short). When the game is in the j-copy, the game moves back to the 0-copy when no new challenge is made and a state in G_j is visited. Otherwise, the game stays in the j-copy.

The acceptance player wins if the rejection player makes infinitely many challenges (the $i \neq j$ part of the final transitions) or if the game stays infinitely often in the 0-copy (the $i = j = 0$ part of the final states). The rejection player wins if the acceptance player never declares, or if she makes only finitely many challenges, and her last challenge is never met.

To keep the definition simple, we have allowed the rejection player to always withdraw a challenge by moving back to the 0-copy without reason. This is never an attractive move for her (so long as she has other options), and can therefore be omitted in an implementation.

An illustrative example can be found in the full version of this paper [14].

Before proving that the resulting automaton is good-for-MDPs in the next section, we would like to point out that it is *not* good-for-games in general. An example for this is provided again in [14].

4 Correctness of the Construction

In order to prove that this alternating Büchi automaton is good-for-MDPs, we first show that using this automaton provides at least the same syntactic probability to win as using the deterministic Streett automaton $\mathcal{S}$.

Lemma 1. *Let $\mathcal{S}$ be a deterministic Streett automaton and $\mathcal{A}$ the alternating automaton from above constructed from $\mathcal{S}$. Then, for every MDP $\mathcal{M}$, $\mathcal{M} \times \mathcal{A}$ has at least the same winning probability as $\mathcal{M} \times \mathcal{S}$.*

Proof. We first observe that the acceptance player (as the Streett player in a finite state Streett game) has an optimal pure finite state memory strategy σ for $\mathcal{M} \times \mathcal{S}$. Let $(\mathcal{M} \times \mathcal{S})_\sigma$ be the Markov chain obtained by using this optimal control.

In $(\mathcal{M} \times \mathcal{S})_\sigma$, we will almost surely reach a leaf component, and the chance of winning is the chance of reaching an accepting leaf component (i.e., a leaf component where the Streett condition is almost surely satisfied).

For $\mathcal{M} \times \mathcal{A}$, we now define a pure finite state strategy τ for the acceptance player from σ and $(\mathcal{M} \times \mathcal{S})_\sigma$ as follows. Outside the accepting leaf components, we follow σ and stay in the original copy. When entering an accepting leaf component, we move to the 0-copy, but otherwise make the same decision as for σ. Henceforth, we make the same decision that σ would make on the history obtained by ignoring in which i-copy we are. (Note that the decision on making an i-challenge, and hence on which i-copy should be visited, rests with the rejection player.)

As this was an accepting leaf component in $(\mathcal{M} \times \mathcal{S})_\sigma$, if there is, for any pair $\langle G_i, R_i \rangle$, a (red) state in R_i in the leaf component, there is also a (green) state in G_i, and this state is almost surely visited infinitely often. Consequently, every challenge will, almost surely, eventually be met, and the acceptance player will win almost surely from these positions, regardless of how the rejection player plays. Thus, τ provides (at least) the same probability to win in $\mathcal{M} \times \mathcal{A}$ as σ provides for $\mathcal{M} \times \mathcal{S}$. □

Different to the case of nondeterministic good-for-MDPs automata originally suggested in [11], we also have to show that the probability of winning for $\mathcal{A}$ cannot exceed that for $\mathcal{S}$.

Lemma 2. *Let $\mathcal{S}$ be a deterministic Streett automaton and $\mathcal{A}$ the alternating automaton from above constructed from $\mathcal{S}$. Then, for every MDP $\mathcal{M}$, $\mathcal{M} \times \mathcal{S}$ has at least the same winning probability as $\mathcal{M} \times \mathcal{A}$.*

Before starting the proof, we define useful terminology, and make the assumption, for simplicity, that a positional optimal strategy σ for the acceptance player on $\mathcal{M} \times \mathcal{A}$ has been fixed.

We call two states of $\mathcal{M} \times \mathcal{A}$ *related*, if they refer to the same vertex of $\mathcal{M}$ and $\mathcal{S}$, but possibly to different copies of this state in $\mathcal{A}$. For such related states, it is obviously the

case that the probability to win from the 0-copy is at least as high as the probability to win from any other i-copy, as the acceptance player can just play as if he started in that i-copy until the time where the first challenge is made. (The only difference with respect to acceptance from the i-copy is then that paths where no challenge is made become winning, such that the probability to win can only go up.) We further observe that the probability to win from the original copy is always at least as high as the probability to win from the 0-copy, as the acceptance player can always declare.

We therefore coin the term "good copy" of a state: a copy of a state is *good* if, and only if, the probability of winning from this copy is as high as the probability of winning from the original copy. A good copy is called *reachable* if it is reachable in $(\mathcal{M} \times \mathcal{A})_\sigma$. The *oldest* reachable good copy of a state (relative to a history) is the good copy i, for which the last visit to G_i is longest ago, where the higher number is given preference as a tie breaker. The 0-copy is only the oldest reachable copy, when it is the only reachable good copy different to the original copy. If no other reachable copy is good, the original copy is the oldest reachable good copy. Naturally, all σ-successors of a reachable good copy are reachable good copies.

Note that the property of being the oldest reachable good copy is relative to the history; a latest appearance record (also known as index appearance record) [7,16,24,25] is a standard memory structure of size $k!$ for keeping track of all information required for determining the oldest copy for a given history. Let M_S be such a memory structure.

Proof. Let σ be an optimal positional strategy of the acceptance player in the Büchi game $\mathcal{M} \times \mathcal{A}$, and let $\mathcal{S}' = \mathcal{S} \times M_S$ be $\mathcal{S}$ equipped with a latest appearance record with $>$ as a tie breaker. We use this to construct the positional strategy τ for $\mathcal{M} \times \mathcal{S}'$ as the strategy that makes the same choice σ makes for the oldest reachable good copy of that state in the $\mathcal{S}$ projection of $\mathcal{S}'$.

It now suffices to show that the rejecting leaf components of $(\mathcal{M} \times \mathcal{S}')_\tau$ refer to states of $\mathcal{M} \times \mathcal{A}$, whose good copies have a winning probability of 0.

We first assume that there is a reachable leaf component that contains a state, where the oldest reachable good copy is the original copy. Note that this implies that the original copy is the only reachable good copy of that state. Naturally, the successor of a reachable good copy under σ is a reachable good copy, so every predecessor of the original copy, and by induction the complete leaf component, consists of states, where the original copy is the only good reachable copy. Thus, this leaf component in $(\mathcal{M} \times \mathcal{S})_\tau$ projects into an end-component in $(\mathcal{M} \times \mathcal{A})_\sigma$, where the rejection player has no decisions, and where no final transition occurs. The winning probability of all states in this end-component is 0.

We now assume that the rejecting leaf component contains only states with the same oldest reachable copy $i \geq 1$. Then the leaf component follows the positional strategy for the i-copy in $(\mathcal{M} \times \mathcal{A})_\sigma$; note that this entails that it does not contain a state in G_i. Therefore the rejection player surely wins in the i-copy of this end-component in $(\mathcal{M} \times \mathcal{A})_\sigma$ by never changing her challenge.

Let us finally turn to the case where a leaf component in $(\mathcal{M} \times \mathcal{S})_\tau$ contains only states, where all oldest reachable good copies are not the original copy, and that these copies are different, or all 0. We assume for contradiction that the leaf component is rejecting. Then there must be an index i such that there is a (red) state from R_i in the

leaf component, but not a (green) state from G_i. Moreover, there must be an i^* with this property where, in the given history, the last occurrence of G_{i^*} is longest ago, using $>$ as tie breaker. Further, let us consider a path through this leaf component that visits states from all (green) sets $G_{i'}$ represented in this leaf component.

Let us now consider a (red) state in R_{i^*} in the leaf component. If the j-copy is not the i^*-copy, then, as the rejection player can make an i^* challenge, the i^*-copy (as a viable successor under the optimal strategy) must be a reachable good copy of the state, too, and therefore, by our assumption, the oldest reachable good state. Thus, we move on to the i^*-copy, and henceforth never leave it, contradicting the assumption that we are in a leaf component that contains different copies, or only the 0-copy, as oldest reachable states.

We have shown that we almost surely reach a leaf component, where the probability of winning all related states is 0 in $\mathcal{M}\times\mathcal{A}$, or where the chance of winning is 1. Together with the local consistency of the probabilities, we get the claim. □

The two lemmas from this section imply that the syntactic and semantic probability to win are the same for all MDPs—in short, that $\mathcal{A}$ is good-for-MDPs. This in particular implies language equivalence on ultimately periodic words (which are a special case of Markov chains, where every state has only one successor), and therefore on all words, as two ω-automata that accept the same ultimately periodic words recognise the same language.

Moreover, we have provided a translation of an optimal strategy obtained for $\mathcal{M}\times\mathcal{A}$ into a strategy for $\mathcal{M}\times\mathcal{S}$ with (at least, and then with Lemma 1 precisely) the same optimal probability to win in the proof of Lemma 2.

Corollary 1. *The alternating Büchi automaton $\mathcal{A}$ that results from the construction of Sect. 3.2 from a DSA $\mathcal{S}$ is a good-for-MDPs automaton that recognises the same language as $\mathcal{S}$. Moreover, we can infer an optimal control strategy for the acceptance player for $\mathcal{M}\times\mathcal{S}$ from an optimal strategy of the acceptance player in $\mathcal{M}\times\mathcal{A}$.* □

An example of adding LAR memory can be found in [14].

We note that the memory we actually need is often smaller than the LAR we have mentioned, as the order can be mangled finitely often. That would, for example, allow us to only keep the order in some SCCs, namely those where we might get stuck in (with probability $\neq 0$)—and, of course, only for those indices that occur in states within these SCCs.

Note that the definition relative to reachability under σ is not required for correctness, but it provides the required connection to learning: when learning an optimal strategy in the game, the bit that is reachable under the optimal strategy we have learned is enough for constructing a pure finite state strategy.

Succinctness. Corollary 4 shows that the alternating Büchi automaton $\mathcal{A}$ that results from the construction of Sect. 3.2 from a DSA $\mathcal{S}$ is a good-for-MDPs automaton that recognises the same language as $\mathcal{S}$, and the number of states of $\mathcal{A}$ is merely $O(kn)$, where n and k are the number of states and Streett pairs of $\mathcal{S}$. At the same time, the translation of a deterministic Streett automaton to a nondeterministic Rabin automaton (without a restriction to GFM) leads to a blow-up that results in $2^{\theta(n)}$ states [23], while a translation to an NBA requires $n2^{\theta(k)}$ states [23], even without the restriction to GFM.

This immediately provides the following theorem.

Theorem 2. *Alternating GFM Büchi automata can be exponentially more succinct than (general) nondeterministic Büchi and Rabin automata.* □

5 Discussion

When ω-regular objectives were first used in model checking MDPs, deterministic Rabin automata were used to represent the objectives. The same has been attempted by the reinforcement learning community: when they first turned to ω-regular objectives, they tried the tested route through deterministic Rabin automata [22], but that translation fails as shown in [9]. Of course, with the current state of knowledge of good-for-MDPs automata, it is not hard to translate deterministic Rabin automata to nondeterministic Büchi automata that are good-for-MDPs, and then to analyse the product of such a Büchi automaton and the MDP in question.

While MDPs with Büchi conditions are a (relatively) easy target for RL methods (like Q-learning [9,11]), a similar translation of Streett automata (or for minimising the chance of meeting a Rabin objective) appears prohibitive. This is because *every* translation from DSAs to nondeterministic Büchi (or even to Rabin) automata incurs an exponential blow-up in the worst case. Surprisingly, we found a way to allow even this accepting condition to be efficiently used in reinforcement learning by generalising the property of being good-for-MDPs to alternating automata, and by constructing an equivalent good-for-MDPs alternating Büchi automaton with linear overhead.

References

1. de Alfaro, L.: Formal Verification of Probabilistic Systems. Ph.D. thesis, Stanford University (1998)
2. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press, Cambridge (2008)
3. Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of reactive(1) designs. J. Comput. Syst. Sci. **78**(3), 911–938 (2012). https://doi.org/10.1016/j.jcss.2011.08.007
4. Buhrke, N., Lescow, H., Vöge, J.: Strategy construction in infinite games with Streett and Rabin chain winning conditions. In: Margaria, T., Steffen, B. (eds.) TACAS 1996. LNCS, vol. 1055, pp. 207–224. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61042-1_46
5. Courcoubetis, C., Yannakakis, M.: The complexity of probabilistic verification. J. ACM **42**(4), 857–907 (1995)
6. Dziembowski, S., Jurdziński, M., Walukiewicz, I.: How much memory is needed to win infinite games? In: Symposium on Logic in Computer Science (LICS 1997), pp. 99–110 (1997)
7. Gurevich, Y., Harrington, L.: Trees, automata and games. In: Symposium on Theory of Computing (STOC 1982), pp. 60–65 (1982)
8. Hahn, E.M., Li, G., Schewe, S., Turrini, A., Zhang, L.: Lazy probabilistic model checking without determinisation. In: Concurrency Theory, pp. 354–367 (2015)

9. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Omega-regular objectives in model-free reinforcement learning. In: Vojnar, T., Zhang, L. (eds.) TACAS 2019. LNCS, vol. 11427, pp. 395–412. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17462-0_27
10. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Faithful and effective reward schemes for model-free reinforcement learning of omega-regular objectives. In: Hung, D.V., Sokolsky, O. (eds.) ATVA 2020. LNCS, vol. 12302, pp. 108–124. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59152-6_6
11. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In: TACAS 2020. LNCS, vol. 12078, pp. 306–323. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45190-5_17
12. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Model-free reinforcement learning for stochastic parity games. In: CONCUR: International Conference on Concurrency Theory, pp. 21:1–21:16. LIPIcs 171 (2020)
13. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: An impossibility result in automata-theoretic reinforcement learning. In: ATVA: Automated Technology for Verification and Analysis (2022). (to appear)
14. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Alternating good-for-MDP automata. arXiv preprint arXiv:2205.03243 (2022)
15. Henzinger, T.A., Piterman, N.: Solving games without determinization. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 395–410. Springer, Heidelberg (2006). https://doi.org/10.1007/11874683_26
16. Löding, C.: Methods for the transformation of ω-automata: complexity and connection to second order logic. Ph.D. thesis, Christian-Albrechts-University of Kiel (1998). Supervisor, Prof. Wolfgang Thomas
17. McIver, A.K., Morgan, C.C.: Games, probability, and the quantitative μ-calculus $qM\mu$. In: Baaz, M., Voronkov, A. (eds.) LPAR 2002. LNCS (LNAI), vol. 2514, pp. 292–310. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36078-6_20
18. Perrin, D., Pin, J.É.: Infinite Words: Automata, Semigroups, Logic and Games. Elsevier (2004)
19. Piterman, N., Pnueli, A.: Faster solutions of Rabin and Streett games. In: Symposium on Logic in Computer Science, pp. 275–284 (2006)
20. Pnueli, A.: The temporal logic of programs. In: IEEE Symposium on Foundations of Computer Science, pp. 46–57 (1977)
21. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, NY (1994)
22. Sadigh, D., Kim, E., Coogan, S., Sastry, S.S., Seshia, S.A.: A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications. In: Conference on Decision and Control (CDC), pp. 1091–1096 (2014)
23. Safra, S., Vardi, M.Y.: On ω-automata and temporal logic. In: Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing, pp. 127–137. STOC 1989, ACM, NY (1989). https://doi.org/10.1145/73007.73019
24. Safra, S.: Exponential determinization for omega-automata with strong-fairness acceptance condition (extended abstract). In: Proceedings of the 24th Annual ACM Symposium on Theory of Computing, 4–6 May 1992, Victoria, British Columbia, pp. 275–282. ACM (1992). https://doi.org/10.1145/129712.129739
25. Safra, S.: Exponential determinization for omega-automata with a strong fairness acceptance condition. SIAM J. Comput. **36**(3), 803–814 (2006). https://doi.org/10.1137/S0097539798332518

26. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. Second edn. MIT Press (2018)
27. Thomas, W.: Automata on infinite objects. In: Handbook of Theoretical Computer Science, pp. 133–191. The MIT Press/Elsevier (1990)
28. Vardi, M.Y.: Automatic verification of probabilistic concurrent finite state programs. In: Foundations of Computer Science, pp. 327–338 (1985)