Omega-Regular Objectives in Model-Free Reinforcement Learning

Ernst Moritz Hahn[1,2], Mateo Perez[3], Sven Schewe[4], Fabio Somenzi[3], Ashutosh Trivedi[5(✉)], and Dominik Wojtczak[4]

[1] School of EEECS, Queen's University Belfast, Belfast, UK
[2] State Key Laboratory of Computer Science, Institute of Software, CAS, Beijing, People's Republic of China
[3] Department of ECEE, University of Colorado Boulder, Boulder, USA
[4] Department of Computer Science, University of Liverpool, Liverpool, UK
[5] Department of Computer Science, University of Colorado Boulder, Boulder, USA
ashutosh.trivedi@colorado.edu

Abstract. We provide the first solution for model-free reinforcement learning of ω-regular objectives for Markov decision processes (MDPs). We present a constructive reduction from the almost-sure satisfaction of ω-regular objectives to an almost-sure reachability problem, and extend this technique to learning how to control an unknown model so that the chance of satisfying the objective is maximized. We compile ω-regular properties into limit-deterministic Büchi automata instead of the traditional Rabin automata; this choice sidesteps difficulties that have marred previous proposals. Our approach allows us to apply model-free, off-the-shelf reinforcement learning algorithms to compute optimal strategies from the observations of the MDP. We present an experimental evaluation of our technique on benchmark learning problems.

1 Introduction

Reinforcement learning (RL) [3,37,40] is an approach to sequential decision making in which agents rely on reward signals to choose actions aimed at achieving prescribed objectives. Model-free RL refers to a class of techniques that are asymptotically space-efficient [36] because they do not construct a full model of the environment. These techniques include classic algorithms like Q-learning [37] as well as their extensions to deep neural networks [14,31]. Some objectives, like running a maze, are naturally expressed in terms of scalar rewards; in other cases the translation is less obvious. We solve the problem of *ω-regular rewards*, that

This work is supported in part by the Marie Skłodowska Curie Fellowship *Parametrised Verification and Control*, NSFC grants 61761136011 and 61532019, an ASIRT grant by the College of Engineering and Applied Sciences of CU Boulder, and EPSRC grants EP/M027287/1 and EP/P020909/1.

T. Vojnar and L. Zhang (Eds.): TACAS 2019, Part I, LNCS 11427, pp. 395–412, 2019.
https://doi.org/10.1007/978-3-030-17462-0_27

is, the problem of defining scalar rewards for the transitions of a Markov decision process (MDP) so that strategies that maximize the probability to satisfy an ω-regular objective may be computed by off-the-shelf, *model-free* RL algorithms.

Omega-regular languages [28,38] provide a rich formalism to unambiguously express qualitative safety and progress requirements of MDPs [2]. A common way to describe an ω-regular language is via a formula in Linear Time Logic (LTL); other specification mechanisms include extensions of LTL, various types of automata, and monadic second-order logic. A typical requirement that is naturally expressed as an ω-regular objective prescribes that the agent should eventually control the MDP to stay within a given set of states, while at all times avoiding another set of states. In LTL this would be written $(\mathsf{F}\,\mathsf{G}\,\texttt{goal}) \wedge (\mathsf{G}\,\neg\texttt{trap})$, where `goal` and `trap` are labels attached to the appropriate states, F stands for "finally," and G stands for "globally."

For verification or synthesis, an ω-regular objective is usually translated into an automaton that monitors the traces of the MDP [10]. Successful executions cause the automaton to take certain (accepting) transitions infinitely often, and ultimately avoid certain (rejecting) transitions. That is, ω-regular objectives are about the long-term behavior of an MDP; the frequency of reward collected is not what matters. A policy that guarantees no rejecting transitions and an accepting transition every ten steps, is better than a policy that promises an accepting transition at each step, but with probability 0.5 does not accept at all.

The problem of ω-regular rewards in the context of model-free RL was first tackled in 2014 by translating the objective into a deterministic Rabin automaton and deriving positive and negative rewards directly from the acceptance condition of the automaton [32]. In Sect. 3 we show that their algorithm, and the extension of [18] may fail to find optimal strategies, and may underestimate the probability of satisfaction of the objective. In [16,17] the use of limit-deterministic Büchi automata avoids the problems connected with the use of a Rabin acceptance condition. However, as shown in Sect. 3, that approach may still produce incorrect results.

We avoid the problems inherent in the use of deterministic Rabin automata for model-free RL by resorting to *limit-deterministic* Büchi automata, which, under mild restrictions, were shown by [8,15,33] to be suitable for both qualitative and quantitative analysis of MDPs under all ω-regular objectives. The Büchi acceptance condition, which, unlike the Rabin condition, does not use rejecting transitions, allows us to constructively reduce the almost-sure satisfaction of ω-regular objectives to an almost-sure reachability problem. It is also suitable for quantitative analysis: the value of a state converges to the maximum probability of satisfaction of the objective from that state as a parameter approaches 1.

We concentrate on model-free approaches and infinitary behaviors for finite MDPs. Related problems include model-based RL [13], RL for finite-horizon objectives [22,23], and learning for efficient verification [4].

This paper is organized as follows. Section 2 recalls definitions and notations. Section 3 shows the problems that arise when the reward of the RL algorithm is derived from the acceptance condition of a deterministic Rabin automaton. In Sect. 4 we prove the main results. Finally, Sect. 5 discusses our experiments.

2 Preliminaries

2.1 Markov Decision Processes

Let $\mathcal{D}(S)$ be the set of distributions over S. A *Markov decision process* $\mathcal{M}$ is a tuple (S, A, T, AP, L) where S is a finite set of states, A is a finite set of *actions*, $T\colon S \times A \rightharpoonup \mathcal{D}(S)$ is the probabilistic transition (partial) function, AP is the set of *atomic propositions*, and $L\colon S \to 2^{AP}$ is the *proposition labeling function*.

For any state $s \in S$, we let $A(s)$ denote the set of actions that can be selected in state s. For states $s, s' \in S$ and $a \in A(s)$, $T(s,a)(s')$ equals $p(s'|s,a)$. A *run* of $\mathcal{M}$ is an ω-word $\langle s_0, a_1, s_1, \ldots\rangle \in S \times (A \times S)^\omega$ such that $p(s_{i+1}|s_i, a_{i+1})>0$ for all $i \geq 0$. A finite run is a finite such sequence. 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 $Runs^{\mathcal{M}}(FRuns^{\mathcal{M}})$ for the set of runs (finite runs) of the MDP $\mathcal{M}$ and $Runs^{\mathcal{M}}(s)(FRuns^{\mathcal{M}}(s))$ for the set of runs (finite runs) of the MDP $\mathcal{M}$ starting from state s. We write $last(r)$ for the last state of a finite run r.

A strategy in $\mathcal{M}$ is a function $\sigma\colon FRuns \to \mathcal{D}(A)$ such that $supp(\sigma(r)) \subseteq A(last(r))$, where $supp(d)$ denotes the support of the distribution d. Let $Runs_\sigma^{\mathcal{M}}(s)$ denote the subset of runs $Runs^{\mathcal{M}}(s)$ that correspond to strategy σ with initial state s. Let $\Sigma_{\mathcal{M}}$ be the set of all strategies. A strategy σ is *pure* if $\sigma(r)$ is a point distribution for all runs $r \in FRuns^{\mathcal{M}}$ and we say that σ is *stationary* if $last(r) = last(r')$ implies $\sigma(r) = \sigma(r')$ for all runs $r, r' \in FRuns^{\mathcal{M}}$. A strategy that is not pure is *mixed*. A strategy is *positional* if it is both pure and stationary.

The behavior of an MDP $\mathcal{M}$ under a strategy σ is defined on a probability space $(Runs_\sigma^{\mathcal{M}}(s), \mathcal{F}_{Runs_\sigma^{\mathcal{M}}(s)}, \Pr_\sigma^{\mathcal{M}}(s))$ over the set of infinite runs of σ with starting state s. Given a real-valued random variable over the set of infinite runs $f\colon Runs^{\mathcal{M}} \to \mathbb{R}$, we denote by $\mathbb{E}_\sigma^{\mathcal{M}}(s)\{f\}$ the expectation of f over the runs of $\mathcal{M}$ originating at s that follow strategy σ.

Given an MDP $\mathcal{M} = (S, A, T, AP, L)$, we define its directed underlying graph $\mathcal{G}_{\mathcal{M}} = (V, E)$ where $V = S$ and $E \subseteq S \times S$ is such that $(s, s') \in E$ if $T(s,a)(s') > 0$ for some $a \in A(s)$. A sub-MDP of $\mathcal{M}$ is an MDP $\mathcal{M}' = (S', A', T', AP, L')$, where $S' \subset S$, $A' \subseteq A$ is such that $A'(s) \subseteq A(s)$ for every $s \in S'$, and T' and L' are analogous to T and L when restricted to S' and A'. In particular, $\mathcal{M}'$ is closed under probabilistic transitions, i.e. for all $s \in S'$ and $a \in A'$ we have that $T(s,a)(s') > 0$ implies that $s' \in S'$. An *end-component* [10] of an MDP $\mathcal{M}$ is a sub-MDP $\mathcal{M}'$ of $\mathcal{M}$ such that $\mathcal{G}_{\mathcal{M}'}$ is strongly connected.

Theorem 1 (End-Component Properties [10]). *Once an end-component C of an MDP is entered, there is a strategy that visits every state-action combination in C with probability 1 and stays in C forever. Moreover, for every strategy the union of the end-components is visited with probability 1.*

A Markov chain is an MDP whose set of actions is singleton. A *bottom strongly connected component* (BSCC) of a Markov chain is any of its end-components. A BSCC is accepting if it contains an accepting transition (see below) and otherwise it is rejecting. For any MDP $\mathcal{M}$ and positional strategy σ, let $\mathcal{M}_\sigma$ be the Markov chain resulting from resolving the nondeterminism in $\mathcal{M}$ using σ.

A *rewardful* MDP is a pair $(\mathcal{M}, \rho)$, where $\mathcal{M}$ is an MDP and $\rho\colon S \times A \to \mathbb{R}$ is a reward function assigning utility to state-action pairs. A rewardful MDP $(\mathcal{M}, \rho)$ under a strategy σ determines a sequence of random rewards $\rho(X_{i-1}, Y_i)_{i\geq 1}$, where X_i and Y_i are the random variables denoting the i-th state and action, respectively. Depending upon the problem of interest, different performance objectives may be of interest. The *reachability probability* objective $Reach(T)^{\mathcal{M}}_{\sigma}(s)$ (with $T \subseteq S$) is defined as $\Pr^{\mathcal{M}}_{\sigma}(s)\left\{\langle s, a_1, s_1, \ldots\rangle \in Runs^{\mathcal{M}}_{\sigma}(s)\colon \exists i\,.\, s_i \in T\right\}$. For a given discount factor $\lambda \in [0, 1[$, the *discounted reward* objective $Disct(\lambda)^{\mathcal{M}}_{\sigma}(s)$ is defined as $\lim_{N\to\infty} \mathbb{E}^{\mathcal{M}}_{\sigma}(s)\left\{\sum_{1\leq i\leq N} \lambda^{i-1}\rho(X_{i-1}, Y_i)\right\}$, while the *average reward* $Avg^{\mathcal{M}}_{\sigma}(s)$ is defined as $\limsup_{N\to\infty}(1/N)\mathbb{E}^{\mathcal{M}}_{\sigma}(s)\left\{\sum_{1\leq i\leq N} \rho(X_{i-1}, Y_i)\right\}$. For an objective $Reward^{\mathcal{M}} \in \{Reach(T)^{\mathcal{M}}, Disct(\lambda)^{\mathcal{M}}, Avg^{\mathcal{M}}\}$ and an initial state s, we define the optimal reward $Reward^{\mathcal{M}}_{*}(s)$ as $\sup_{\sigma\in\Sigma_{\mathcal{M}}} Reward^{\mathcal{M}}_{\sigma}(s)$. A strategy $\sigma \in \Sigma_{\mathcal{M}}$ is optimal for $Reward^{\mathcal{M}}$ if $Reward^{\mathcal{M}}_{\sigma}(s) = Reward^{\mathcal{M}}_{*}(s)$ for all $s \in S$.

2.2 ω-Regular Performance Objectives

A *nondeterministic ω-automaton* is a tuple $\mathcal{A} = (\Sigma, Q, q_0, \delta, \mathsf{Acc})$, where Σ is a finite *alphabet*, Q is a finite set of *states*, $q_0 \in Q$ is the *initial state*, $\delta\colon Q \times \Sigma \to 2^Q$ is the *transition function*, and Acc is the *acceptance condition*. A *run* r of $\mathcal{A}$ on $w \in \Sigma^\omega$ is an ω-word $r_0, w_0, r_1, w_1, \ldots$ in $(Q \cup \Sigma)^\omega$ such that $r_0 = q_0$ and, for $i > 0$, $r_i \in \delta(r_{i-1}, w_{i-1})$. Each triple (r_{i-1}, w_{i-1}, r_i) is a *transition* of $\mathcal{A}$.

We consider Büchi and Rabin acceptance conditions, which depend on the transitions that occur infinitely often in a run of an automaton. We write $\mathsf{inf}(r)$ for the set of transitions that appear infinitely often in the run r. The *Büchi* acceptance condition defined by $F \subseteq Q \times \Sigma \times Q$ is the set of runs $\{r \in (Q \cup \Sigma)^\omega\colon \mathsf{inf}(r) \cap F \neq \emptyset\}$. A *Rabin* acceptance condition is defined in terms of k pairs of subsets of $Q \times \Sigma \times Q$, $(B_0, G_0), \ldots, (B_{k-1}, G_{k-1})$, as the set $\{r \in (Q \cup \Sigma)^\omega\colon \exists i < k\,.\, \mathsf{inf}(r) \cap B_i = \emptyset \wedge \mathsf{inf}(r) \cap G_i \neq \emptyset\}$. The *index* of a Rabin condition is its number of pairs.

A run r of $\mathcal{A}$ is *accepting* if $r \in \mathsf{Acc}$. The *language*, $L_{\mathcal{A}}$, of $\mathcal{A}$ (or, *accepted* by $\mathcal{A}$) is the subset of words in Σ^ω that have accepting runs in $\mathcal{A}$. A language is *ω-regular* if it is accepted by an ω-automaton.

Given an MDP $\mathcal{M}$ and an ω-regular objective φ given as an ω-automaton $\mathcal{A}_\varphi = (\Sigma, Q, q_0, \delta, \mathsf{Acc})$, we are interested in computing an optimal strategy satisfying the objective. We define the satisfaction probability of a strategy σ from initial state s as: $\Pr^{\mathcal{M}}_{\sigma}(s \models \varphi) = \Pr^{\mathcal{M}}_{\sigma}(s)\left\{r \in Runs^{\mathcal{M}}_{\sigma}(s)\colon L(r) \in L_{\mathcal{A}}\right\}$. The optimal satisfaction probability $\Pr^{\mathcal{M}}_{*}(s \models \varphi)$ is defined as $\sup_{\sigma\in\Sigma_{\mathcal{M}}} \Pr^{\mathcal{M}}_{\sigma}(s \models \varphi)$ and we say that $\sigma \in \Sigma_{\mathcal{M}}$ is an optimal strategy for φ if $\Pr^{\mathcal{M}}_{*}(s \models \varphi) = \Pr^{\mathcal{M}}_{\sigma}(s \models \varphi)$.

An automaton $\mathcal{A} = (\Sigma, Q, q_0, \delta, \mathsf{Acc})$ is *deterministic* if $|\delta(q, \sigma)| \leq 1$ for all $q \in Q$ and all $\sigma \in \Sigma$. $\mathcal{A}$ is *complete* if $|\delta(q, \sigma)| \geq 1$. A word in Σ^ω has exactly one run in a deterministic, complete automaton. We use common three-letter abbreviations to distinguish types of automata. The first (D or N) tells whether

the automaton is deterministic; the second denotes the acceptance condition (B for Büchi and R for Rabin). The third letter (W) says that the automaton reads ω-words. For example, an NBW is a nondeterministic Büchi automaton, and a DRW is a deterministic Rabin automaton.

Every ω-regular language is accepted by some DRW and by some NBW. In contrast, there are ω-regular languages that are not accepted by any DBW. The *Rabin index* of a Rabin automaton [6,20] is the index of its acceptance condition. The Rabin index of an ω-regular language $\mathcal{L}$ is the minimum index among those of the DRWs that accept $\mathcal{L}$. For each $n \in \mathbb{N}$ there exist ω-regular languages of Rabin index n. The languages accepted by DBWs, however, form a proper subset of the languages of index 1.

2.3 The Product MDP

Given an MDP $\mathcal{M} = (S, A, T, AP, L)$ with a designated initial state $s_0 \in S$, and a deterministic ω-automaton $\mathcal{A} = (2^{AP}, Q, q_0, \delta, \mathsf{Acc})$, the *product* $\mathcal{M} \times \mathcal{A}$ is the tuple $(S \times Q, (s_0, q_0), A, T^\times, \mathsf{Acc}^\times)$. The probabilistic transition function $T^\times \colon (S \times Q) \times A \rightharpoonup \mathcal{D}(S \times Q)$ is such that $T^\times((s,q),a)((\hat{s},\hat{q})) = T(s,a)(\hat{s})$ if $\{\hat{q}\} = \delta(q, L(s))$ and is 0 otherwise. If $\mathcal{A}$ is a DBW, Acc is defined by $F \subseteq Q \times 2^{AP} \times Q$; then $F^\times \subseteq (S \times Q) \times A \times (S \times Q)$ defines $\mathsf{Acc}^\times$ as follows: $((s,q),a,(s',q')) \in F^\times$ if and only if $(q, L(s), q') \in F$ and $T(s,a)(s') \neq 0$. If $\mathcal{A}$ is a DRW of index k, $\mathsf{Acc}^\times = \{(B_0^\times, G_0^\times), \ldots, (B_{k-1}^\times, G_{k-1}^\times)\}$. To set B_i of Acc, there corresponds $B_i^\times$ of $\mathsf{Acc}^\times$ such that $((s,q),a,(s',q')) \in B_i^\times$ if and only if $(q, L(s), q') \in B_i$ and $T(s,a)(s') \neq 0$. Likewise for $G_i^\times$.

If $\mathcal{A}$ is a nondeterministic automaton, the actions in the product are enriched to identify both the actions of the original MDP and the choice of the successor state of the nondeterministic automaton.

End-components and runs are defined for products just like for MDPs. A run of $\mathcal{M} \times \mathcal{A}$ is accepting if it satisfies the product's acceptance condition. An *accepting end-component* of $\mathcal{M} \times \mathcal{A}$ is an end-component such that every run of the product MDP that eventually dwells in it is accepting.

In view of Theorem 1, satisfaction of an ω-regular objective φ by an MDP $\mathcal{M}$ can be formulated in terms of the accepting end-components of the product $\mathcal{M} \times \mathcal{A}_\varphi$, where $\mathcal{A}_\varphi$ is an automaton accepting φ. The maximum probability of satisfaction of φ by $\mathcal{M}$ is the maximum probability, over all strategies, that a run of the product $\mathcal{M} \times \mathcal{A}_\varphi$ eventually dwells in one of its accepting end-components.

It is customary to use DRWs instead of DBWs in the construction of the product, because the latter cannot express all ω-regular objectives. On the other hand, general NBWs are not used since causal strategies cannot optimally resolve nondeterministic choices because that requires access to future events [39].

2.4 Limit-Deterministic Büchi Automata

In spite of the large gap between DRWs and DBWs in terms of indices, even a very restricted form of nondeterminism is sufficient to make DBWs as expressive as DRWs. Broadly speaking, an LDBW behaves deterministically once it has seen an accepting transition. Formally, a *limit-deterministic* Büchi automaton

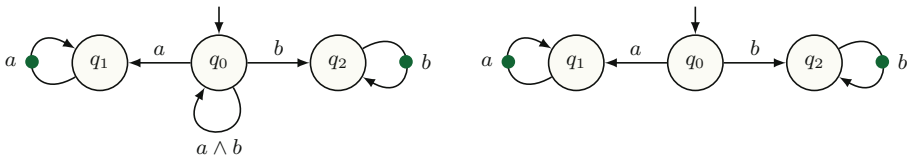

Fig. 1. Suitable (left) and unsuitable (right) LDBWs for the LTL formula $(\mathsf{G}\, a) \vee (\mathsf{G}\, b)$.

(LDBW) is an NBW $\mathcal{A} = (\Sigma, Q_i \cup Q_f, q_0, \delta, F)$ such that

- $Q_i \cap Q_f = \emptyset$, $F \subseteq Q_f \times \Sigma \times Q_f$;
- $|\delta(q, \sigma) \cap Q_i| \leq 1$ for all $q \in Q_i$ and $\sigma \in \Sigma$;
- $|\delta(q, \sigma)| \leq 1$ for all $q \in Q_f$ and $\sigma \in \Sigma$;
- $\delta(q, \sigma) \subseteq Q_f$ for all $q \in Q_f$ and $\sigma \in \Sigma$.

LDBWs are as expressive as general NBWs. Moreover, NBWs can be translated into LDBWs that can be used for the qualitative and quantitative analysis of MDPs [8,15,33,39]. We use the translation from [15], which uses LDBWs that consist of two parts: an initial deterministic automaton (without accepting transitions) obtained by a subset construction; and a final part produced by a breakpoint construction. They are connected by a single "guess", where the automaton guesses a singleton subset of the reachable states to start the breakpoint construction. Like in other constructions (e.g. [33]), one can compose the resulting automata with an MDP, such that the optimal control of the product defines a control on the MDP that maximizes the probability of obtaining a word from the language of the automaton. We refer to LDBWs with this property as *suitable* limit-deterministic automata (SLDBWs).

Definition 1 (Suitable LDBW). *An SLDBW $\mathcal{A}$ for property φ is an LDBW that recognizes φ and such that, for every finite MDP $\mathcal{M}$, there exists a positional strategy $\sigma \in \Sigma_{\mathcal{M}\times\mathcal{A}}$ such that the probability of satisfying the Büchi condition in the Markov chain $(\mathcal{M} \times \mathcal{A})_\sigma$ is $\Pr^{\mathcal{M}}_*(s \models \varphi)$.*

Although the construction of a suitable LDBW reaches back to the 80s [39], not all LDBWs are suitable. Broadly speaking, the nondeterministic decisions taken in the initial part may not depend on the future—though it may depend on the state of an MDP. The example LDBW from Fig. 1 (left) satisfies the property: it can try to delay to progress to one of the accepting states to when an end-component in an MDP is reached that always produces a's or b's, respectively. In contrast, the LDBW from Fig. 1 (right)—which recognizes the same language—will have to make the decision of seeing only a's or only b's immediately, without the option to wait for reaching an end-component. This makes it unsuitable for the use in MDPs.

Theorem 2 [8,15,33,39]**.** *Suitable limit-deterministic Büchi automata exist for all ω-regular languages.*

SLDBWs—and their properties described in Definition 1—are used in the qualitative and quantitative model checking algorithms in [8,15,33,39]. The accepting end-components of the product MDPs are all using only states from the final part of the SLDBW. Büchi acceptance then allows for using memoryless almost sure winning strategies in the accepting end-components, while outside of accepting end-components a memoryless strategy that maximizes the chance of reaching such an end-component can be used. The distinguishing property is the guarantee that they provide the correct probability, while using a product with a general NBW would only provide a value that cannot exceed it.

2.5 Linear Time Logic Objectives

LTL (Linear Time Logic) is a temporal logic whose formulae describe a subset of the ω-regular languages, which is often used to specify objectives in human-readable form. Translations exist from LTL to various forms of automata, including NBW, DRW, and SLDBW. Given a set of atomic propositions AP, a is an LTL formula for each $a \in AP$. Moreover, if φ and ψ are LTL formulae, so are $\neg\varphi, \varphi \vee \psi, \mathsf{X}\,\varphi, \psi\,\mathsf{U}\,\varphi$. Additional operators are defined as abbreviations: $\top \stackrel{\text{def}}{=} a \vee \neg a$; $\bot \stackrel{\text{def}}{=} \neg\top$; $\varphi \wedge \psi \stackrel{\text{def}}{=} \neg(\neg\varphi \vee \neg\psi)$; $\varphi \rightarrow \psi \stackrel{\text{def}}{=} \neg\varphi \vee \psi$; $\mathsf{F}\,\varphi \stackrel{\text{def}}{=} \top\,\mathsf{U}\,\varphi$; and $\mathsf{G}\,\varphi \stackrel{\text{def}}{=} \neg\,\mathsf{F}\,\neg\varphi$. We write $w \models \varphi$ if ω-word w over 2^{AP} satisfies LTL formula φ. The satisfaction relation is defined inductively [2,24].

2.6 Reinforcement Learning

For an MDP $\mathcal{M}$ and an objectives $Reward^{\mathcal{M}} \in \{Reach(T)^{\mathcal{M}}, Disct(\lambda)^{\mathcal{M}}, Avg^{\mathcal{M}}\}$, the optimal reward and an optimal strategy can be computed using value iteration, policy iteration, or, in polynomial time, using linear programming [12,30]. On the other hand, for ω-regular objectives (given as DRW, SLDBW, or LTL formulae) optimal satisfaction probabilities and strategies can be computed using graph-theoretic techniques (computing accepting end-component and then maximizing the probability to reach states in such components) over the product structure. However, when the MDP transition/reward structure is unknown, such techniques are not applicable.

For MDPs with unknown transition/reward structure, *reinforcement learning* [37] provides a framework to compute optimal strategies from repeated interactions with the environment. Of the two main approaches to reinforcement learning in MDPs, *model-free* approaches and *model-based* approaches the former, which is asymptotically space-efficient [36], has been demonstrated to scale well [14,25,35]. In a model-free approach such as Q-learning [31,37], the learner computes optimal strategies without explicitly estimating the transition probabilities and rewards. We focus on making it possible for model-free RL to learn a strategy that maximizes the probability of satisfying a given ω-regular objective.

3 Problem Statement and Motivation

Given MDP $\mathcal{M}$ with unknown transition structure and ω-regular objective φ, we seek a strategy that maximizes the probability that $\mathcal{M}$ satisfies φ.

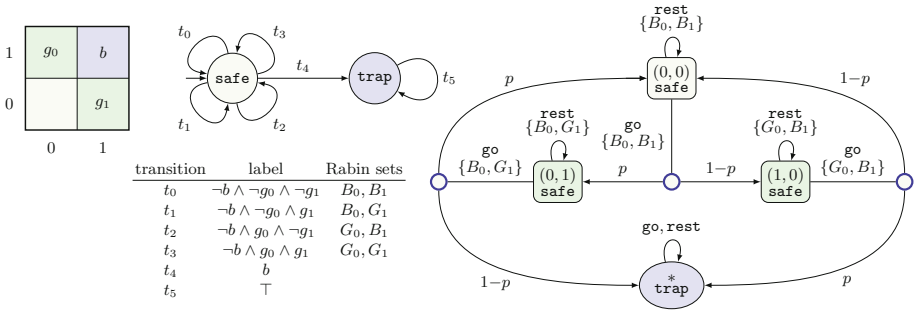

Fig. 2. A grid-world example (left), a Rabin automaton for $[(\mathsf{F}\,\mathsf{G}\,g_0) \vee (\mathsf{F}\,\mathsf{G}\,g_1)] \wedge \mathsf{G}\,\neg b$ (center), and product MDP (right).

To apply model-free RL algorithms to this task, one needs to define rewards that depend on the observations of the MDP and reflect the satisfaction of the objective. It is natural to use the product of the MDP and an automaton monitoring the satisfaction of the objective to assign suitable rewards to various actions chosen by the learning algorithm.

Sadigh *et al.* [32] were the first to apply model-free RL to a qualitative-version of this problem, i.e., to learn a strategy that satisfies the property with probability 1. For an MDP $\mathcal{M}$ and a DRW $\mathcal{A}_\varphi$ of index k, they formed the product MDP $\mathcal{M} \times \mathcal{A}_\varphi$ with k different "Rabin" reward functions $\rho_1, \ldots, \rho_k$. The function ρ_i corresponds to the Rabin pair $(B_i^\times, G_i^\times)$: it assigns a fixed negative reward $-R_- < 0$ to all edges in $B_i^\times$ and a fixed positive reward $R_+ > 0$ to all edges in $G_i^\times$. [32] claimed that if there exists a strategy satisfying an ω-regular objective φ with probability 1, then there exists a Rabin pair i, discount factor $\lambda_* \in [0, 1[$, and suitably high ratio R_*, such that for all $\lambda \in [\lambda_*, 1[$ and $R_-/R_+ \geq R_*$, any strategy maximizing λ-discounted reward for the MDP $(\mathcal{M} \times \mathcal{A}_\varphi, \rho_i)$ also satisfies the ω-regular objective φ with probability 1. Using Blackwell-optimality theorem [19], a paraphrase of this claim is that if there exists a strategy satisfying an ω-regular objective φ with probability 1, then there exists a Rabin pair i and suitably high ratio R_*, such that for all $R_-/R_+ \geq R_*$, any strategy maximizing expected average reward for the MDP $(\mathcal{M} \times \mathcal{A}_\varphi, \rho_i)$ also satisfies the ω-regular objective φ with probability 1. This approach has two faults, the second of which also affects approaches that replace DRWs with LDBWs [16,17].

1. We provide in Example 1 an MDP and an ω-regular objective φ with Rabin index 2, such that, although there is a strategy that satisfies the property with probability 1, optimal average strategies from any Rabin reward do not satisfy the objective with probability 1.
2. Even for an ω-regular objective with one Rabin pair (B, G) and $B{=}\emptyset$—i.e., one that can be specified by a DBW—we demonstrate in Example 2 that the problem of finding a strategy that satisfies the property with probability 1 may not be reduced to finding optimal average strategies.

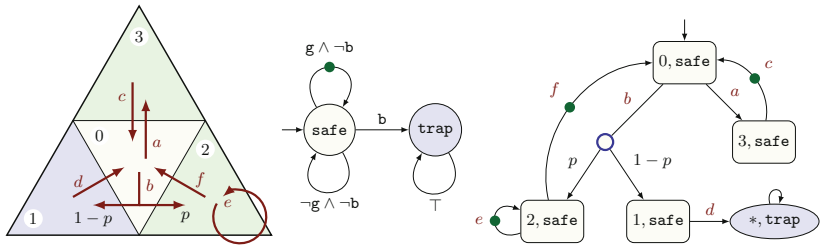

Fig. 3. A grid-world example. The arrows represent actions (left). When action b is performed, Cell 2 is reached with probability p and Cell 1 is reached with probability $1-p$, for $0 < p < 1$. Deterministic Büchi automaton for $\varphi = (\mathsf{G}\,\neg\mathsf{b}) \wedge (\mathsf{G}\,\mathsf{F}\,\mathsf{g})$ (center). The dotted transition is the only accepting transition. Product MDP (right).

Example 1 (Two Rabin Pairs). Consider the MDP given as a simple grid-world example shown in Fig. 2. Each cell (state) of the MDP is labeled with the atomic propositions that are true there. In each cell, there is a choice between two actions `rest` and `go`. With action `rest` the state of the MDP does not change. However, with action `go` the MDP moves to the other cell in the same row with probability p, or to the other cell in the same column with probability $1-p$. The initial cell is $(0, 0)$.

The specification is given by LTL formula $\varphi = [(\mathsf{F}\,\mathsf{G}\,g_0) \vee (\mathsf{F}\,\mathsf{G}\,g_1)] \wedge \mathsf{G}\,\neg b$. A DRW that accepts φ is shown in Fig. 2. The DRW has two accepting pairs: (B_0, G_0) and (B_1, G_1). The table besides the automaton gives, for each transition, its label and the B and G sets to which it belongs.

The optimal strategy that satisfies the objective φ with probability 1 chooses `go` in Cell $(0, 0)$ and chooses `rest` subsequently. However, for both Rabin pairs, the optimal strategy for expected average reward is to maximize the probability of reaching one of the $(0, 1), \texttt{safe}$ or $(1, 0), \texttt{safe}$ states of the product and stay there forever. For the first accepting pair the maximum probability of satisfaction is $\frac{1}{2-p}$, while for the second pair it is $\frac{1}{1+p}$.

Example 2 (DBW to Expected Average Reward Reduction). This counterexample demonstrates that even for deterministic Büchi objectives, the problem of finding an optimal strategy satisfying an objective may not be reduced to the problem of finding an optimal average strategy. Consider the simple grid-world example of Fig. 3 with the specification $\varphi = (\mathsf{G}\,\neg\mathsf{b}) \wedge (\mathsf{G}\,\mathsf{F}\,\mathsf{g})$, where atomic proposition `b` (blue) labels Cell 1 and atomic proposition `g` (green) labels Cells 2 and 3. Actions enabled in various cells and their probabilities are depicted in the figure.

The strategy from Cell 0 that chooses Action a guarantees satisfaction of φ with probability 1. An automaton with accepting transitions for φ is shown in Fig. 3; it is a DBW (or equivalently a DRW with one pair (B, G) and $B = \emptyset$).

The product MDP is shown at the bottom of Fig. 3. All states whose second component is `trap` have been merged. Notice that there is no negative reward since the set B is empty. If reward is positive and equal for all accepting transitions, and 0 for all other transitions, then when $p > 1/2$, the strategy that

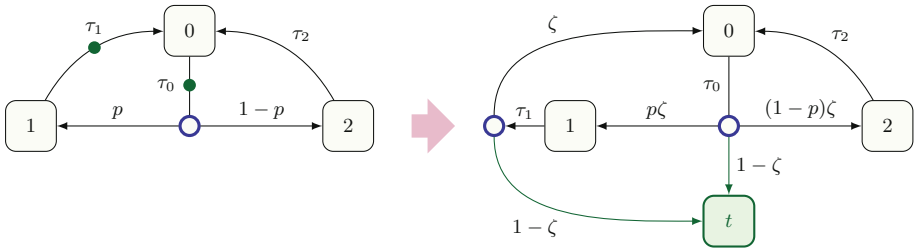

Fig. 4. Adding transitions to the target in the augmented product MDP.

maximizes expected average reward chooses Action b in the initial state and Action e from State $(2, \texttt{safe})$. Note that, for large values of λ, the optimal expected average reward strategies are also optimal strategies for the λ-discounted reward objective. However, these strategies are not optimal for ω-regular objectives.

Example 1 shows that one cannot select a pair from a Rabin acceptance condition ahead of time. This problem can be avoided by the use of Büchi acceptance conditions. While DBWs are not sufficiently expressive, SLDBWs express all ω-regular properties and are suitable for probabilistic model checking. In the next section, we show that they are also "the ticket" for model-free reinforcement learning, because they allow us to maximize the probability of satisfying an ω-regular specification by solving a reachability probability problem that can be solved efficiently by off-the-shelf RL algorithms.

4 Model-Free RL from Omega-Regular Rewards

We now reduce the model checking problem for a given MDP and SLDBW to a reachability problem by slightly changing the structure of the product: We add a target state t that can be reached with a given probability $1-\zeta$ whenever visiting an accepting transition of the original product MDP.

Our reduction avoids the identification of winning 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. In the rest of this section, we fix an MDP $\mathcal{M}$ and an SLDBW $\mathcal{A}$ for the ω-regular property φ.

Definition 2 (Augmented Product). *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 new 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. 4. 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.

We let $p_s^\sigma(\zeta)$ denote the probability of reaching t in $\mathcal{M}_\sigma^\zeta$ when starting at state s. Notice that we can encode this value as the expected average reward in the following rewardful MDP $(\mathcal{M}^\zeta, \rho)$, where we set the reward function $\rho(t,a) = 1$ for all $a \in A$ and $\rho(s,a) = 0$ otherwise. For any strategy σ, the probability $p_s^\sigma(\zeta)$ and the reward of σ from s in $(\mathcal{M}^\zeta, \rho)$ are the same. We also let a_s^σ be the probability that a run that starts from s in $(\mathcal{M} \times \mathcal{A})_\sigma$ is accepting.

Lemma 1. *If σ is a positional strategy on $\mathcal{M}^\zeta$, then, for every state s of the Markov chain $(\mathcal{M}\times\mathcal{A})_\sigma$, the following holds:*

1. *if the state s is in a rejecting BSCC of $(\mathcal{M}\times\mathcal{A})_\sigma$, then $p_s^\sigma(\zeta) = 0$;*
2. *if the state s is in an accepting BSCC of $(\mathcal{M}\times\mathcal{A})_\sigma$, then $p_s^\sigma(\zeta) = 1$;*
3. *the probability $p_s^\sigma(\zeta)$ of reaching t is greater than a_s^σ; and*
4. *if $p_s^\sigma(\zeta)$=1 then no rejecting BSCC is reachable from s in $(\mathcal{M}\times\mathcal{A})_\sigma$ and $a_s^\sigma = 1$.*

Proof. (1) holds as there are no accepting transition in a rejecting BSCC of $(\mathcal{M}\times\mathcal{A})_\sigma$, and so t cannot be reached when starting at s in $\mathcal{M}_\sigma^\zeta$. (2) holds because t (with its self-loop) is the only BSCC reachable from s in $\mathcal{M}^\zeta$. In other words, t (with its self-loop) and the rejecting BSCCs of $(\mathcal{M}\times\mathcal{A})_\sigma$ are the only BSCCs in $\mathcal{M}_\sigma^\zeta$. (3) then follows, because the same paths lead to a rejecting BSCCs in $(\mathcal{M}\times\mathcal{A})_\sigma$ and $\mathcal{M}_\sigma^\zeta$, while the probability of each such a path is no larger—and strictly smaller iff it contains an accepting transition—than in $\mathcal{M}_\sigma^\zeta$. (4) holds because, if $p_s^\sigma(\zeta) = 1$, then t (with its self-loop) is the only BSCC reachable from s in $\mathcal{M}_\sigma^\zeta$. Thus, there is no path to a rejecting BSCC in $\mathcal{M}_\sigma^\zeta$, and therefore no path to a rejecting BSCC in $(\mathcal{M}\times\mathcal{A})_\sigma$. □

Lemma 2. *Let σ be a positional strategy on $\mathcal{M}^\zeta$. For every state s of $\mathcal{M}^\zeta$, we have that $\lim_{\zeta\uparrow 1} p_s^\sigma(\zeta) = a_s^\sigma$.*

Proof. As shown in Lemma 1(3) for all ζ, we have $p_s^\sigma(\zeta) \geq a_s^\sigma$. For a coarse approximation of their difference, we recall that $(\mathcal{M}\times\mathcal{A})_\sigma$ is a finite Markov chain. The expected number of transitions taken before reaching a BSCC from s in $(\mathcal{M}\times\mathcal{A})_\sigma$ is therefore a finite number. Let us refer to the—no larger—expected number of accepting transitions taken before reaching a BSCC when starting at s in $(\mathcal{M}\times\mathcal{A})_\sigma$ as f_s^σ. We claim that $a_s^\sigma \geq p_s^\sigma(\zeta) - (1-\zeta)\cdot f_s^\sigma$. This is because the probability of reaching a rejecting BSCC in $(\mathcal{M}\times\mathcal{A})_\sigma$ is at most the probability of reaching a rejecting BSCC in $\mathcal{M}_\sigma^\zeta$, which is at most $1 - p_s^\sigma(\zeta)$, plus the probability of moving on to t from a state that is not in any BSCC in $(\mathcal{M}\times\mathcal{A})_\sigma$, which we are going to show next is at most $f_s^\sigma \cdot (1-\zeta)$.

First, a proof by induction shows that $(1-\zeta^k) \leq k(1-\zeta)$ for all $k \geq 0$. Let $P_s^\sigma(\zeta, k)$ be the probability of generating a path from s with k accepting transitions before t or a node in some BSCC of $(\mathcal{M}\times\mathcal{A})_\sigma$ is reached in $\mathcal{M}_\sigma^\zeta$. The probability of seeing k accepting transitions and not reaching t is at least ζ^k. Therefore, probability of moving to t from a state not in any BSCC is at most

$$\sum_k P_s^\sigma(\zeta,k)(1-\zeta^k) \leq \sum_k P_s^\sigma(\zeta,k)k\cdot(1-\zeta) \leq f_s^\sigma \cdot (1-\zeta).$$

The proof is now complete. □

This provides us with our main theorem.

Theorem 3. *There exists a threshold* $\zeta' \in]0,1[$ *such that, for all* $\zeta > \zeta'$ *and every state* s*, any strategy* σ *that maximizes the probability* $p_s^\sigma(\zeta)$ *of reaching the sink in* $\mathcal{M}^\zeta$ *is (1) an optimal strategy in* $\mathcal{M} \times \mathcal{A}$ *from* s *and (2) induces an optimal strategy for the original MDP* $\mathcal{M}$ *from* s *with objective* φ*.*

Proof. We use the fact that it suffices to study positional strategies, and there are only finitely many of them. Let σ_1 be an optimal strategy of $\mathcal{M} \times \mathcal{A}$, and let σ_2 be a strategy that has the highest likelihood of creating an accepting run among all non-optimal memoryless ones. (If σ_2 does not exist, then all strategies are equally good, and it does not matter which one is chosen.) Let $\delta = a_s^{\sigma_1} - a_s^{\sigma_2}$.

Let $f_{\max} = \max_\sigma \max_s f_s^\sigma$, where σ ranges over positional strategies only, and f_s^σ is defined as in Lemma 2. We claim that it suffices to pick $\zeta' \in]0,1[$ such that $(1-\zeta') \cdot f_{\max} < \delta$. Suppose that σ is a positional strategy that is optimal in $\mathcal{M}^\zeta$ for $\zeta > \zeta'$, but is not optimal in $\mathcal{M} \times \mathcal{A}$. We then have

$$a_s^\sigma \leq p_s^\sigma(\zeta) \leq a_s^\sigma + (1-\zeta) f_s^\sigma < a_s^\sigma + \delta \leq a_s^{\sigma_1} \leq p_s^{\sigma_1}(\zeta),$$

where these inequalities follow, respectively, from: Lemma 1(3), the proof of Lemma 2, the definition of ζ', the assumption that σ is not optimal and the definition of δ, and the last one from Lemma 1(3). This shows that $p_s^\sigma(\zeta) < p_s^{\sigma_1}(\zeta)$, i.e., σ is not optimal in $\mathcal{M}^\zeta$; a contradiction. Therefore, any positional strategy that is optimal in $\mathcal{M}^\zeta$ for $\zeta > \zeta'$ is also optimal in $\mathcal{M} \times \mathcal{A}$.

Now, suppose that σ is a positional strategy that is optimal in $\mathcal{M} \times \mathcal{A}$. Then the probability of satisfying φ in $\mathcal{M}$ when starting at s is at least[1] a_s^σ. At the same time, if there was a strategy for which the probability of satisfying φ in $\mathcal{M}$ is $> a_s^\sigma$, then the property of $\mathcal{A}$ to be an SLDBW (Definition 1) would guarantee the existence of strategy σ' for which $a_s^{\sigma'} > a_s^\sigma$; a contradiction with the assumption that σ is optimal. Therefore any positional strategy that is optimal in $\mathcal{M} \times \mathcal{A}$ induces an optimal strategy in $\mathcal{M}$ with objective φ. □

Corollary 1. *Due to Lemma 1(4),* $\mathcal{M}$ *satisfies* φ *almost surely if and only if the sink is almost surely reachable in* $\mathcal{M}^\zeta$ *for all* $0 < \zeta < 1$*.*

Theorem 3 leads to a very simple model-free RL algorithm. The augmented product is not built by the RL algorithm, which does not know the transition structure of the environment MDP. Instead, the observations of the MDP are used by an *interpreter* process to compute a run of the objective automaton. 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.

[1] This holds for all nondeterministic automata that recognize the models of φ: an accepting run establishes that the path was a model of φ.

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 maximizes this probabilistic reward is guaranteed to converge to a policy that maximizes the probability of satisfaction of the ω-regular objective.

5 Experimental Results

We implemented the construction described in the previous sections in a tool named Mungojerrie [11], which reads MDPs described in the PRISM language [21], and ω-regular automata written in the HOA format [1,9]. Mungojerrie builds the augmented product $\mathcal{M}^{\zeta}$, provides an interface for RL algorithms akin to that of [5] and supports probabilistic model checking. Our algorithm computes, for each pair (s, a) of state and action, the maximum probability of satisfying the given objective after choosing action a from state s by using off-the-shelf, temporal difference algorithms. Not all actions with maximum probability are part of positional optimal strategies—consider a product MDP with one state and two actions, a and b, such that a enables an accepting self-loop, and b enables a non-accepting one: both state/action pairs are assigned probability 1. In b's case, because choosing b once—or a finite number of times—does not preclude acceptance. Since the probability values alone do not identify a pure optimal strategy, Mungojerrie computes an optimal mixed strategy, uniformly choosing all maximum probability actions from a state.

The MDPs on which we tested our algorithms [26] are listed in Table 1. For each model, the numbers of decision states in the MDP, the automaton, and the product MDP are given. Next comes the probability of satisfaction of the objective for the strategy chosen by the RL algorithm as computed by the model checker (which has full access to the MDP). This is followed by the estimate of the probability of satisfaction of the objective computed by the RL algorithm and the time taken by learning. The last six columns report values of the hyperparameters when they deviate from the default values: ζ controls the probability of reward, ϵ is the exploration rate, α is the learning rate, and tol is the tolerance for probabilities to be considered different. Finally, ep-l controls the episode length (it is the maximum allowed length of a path in the MDP that does contain an accepting edge) and ep-n is the number of episodes. All performance data are the averages of three trials with Q-learning. Rewards are undiscounted, so that the value of a state-action pair computed by Q-learning is a direct estimate of the probability of satisfaction of the objective from that state when taking that action.

Models `twoPairs` and `riskReward` are from Examples 1 and 2, respectively. Model `deferred` is discussed later. Models `grid5x5` and `trafficNtk` are from [32]. The three "windy" MDPs are taken from [37]. The "frozen" examples are from [27]. Some ω-regular objectives are simple reachability requirements (e.g., `frozenSmall` and `frozenLarge`). The objective for the `othergrid` models is to collect three types of coupons, while incurring at most one of two types of penalties. In `doublegrid` two agents simultaneously move across the grid.

Table 1. Q-learning results. The default values of the learner hyperparameters are: $\zeta = 0.99$, $\epsilon = 0.1$, $\alpha = 0.1$, tol= 0.01, ep-l= 30, and ep-n= 20000. Times are in seconds.

Name	states	aut.	prod.	prob.	est.	time	ζ	ε	α	tol	ep-l	ep-n
`twoPairs`	4	4	16	1	1	0.26						
`riskReward`	4	2	8	1	1	1.47						
`deferred`	41	1	41	1	1	1.01						
`grid5x5`	25	3	75	1	1	10.82		0.01	0.2		400	30k
`trafficNtk`	122	13	462	1	1	2.89						
`windy`	123	2	240	1	1	12.35	0.95	0.001	0.05	0	900	200k
`windyKing`	130	2	256	1	1	14.34	0.95	0.02	0.2	0	300	120k
`windyStoch`	130	2	260	1	1	47.70	0.95	0.02	0.2	0	300	200k
`frozenSmall`	16	3	48	0.823	0.83	0.51			0.05	0	200	
`frozenLarge`	64	3	192	1	1	1.81			0.05	0	700	
`othergrid6`	36	25	352	1	1	10.80				0	300	75k
`othergrid20`	400	25	3601	1	1	78.00	0.9999	0.07	0.2	0	5k	
`othergrid40`	1600	25	14401	1	0.99	87.90	0.9999	0.05	0.2	0	14k	25k
`doublegrid8`	4096	3	12287	1	1	45.50				0	3k	100k
`doublegrid12`	20736	3	62207	1	1	717.6				0	20k	300k
`slalom`	36	5	84	1	1	0.98						
`rps1`	121	2	130	0.768	0.76	5.21		0.12	0.006	0		500k
`dpenny`	52	2	65	0.5	0.5	1.99		0.001	0.2	0	50	120k
`devious`	11	1	11	1	1	0.81						
`arbiter2`	32	3	72	1	1	5.16			0.5	0.02	200	
`knuthYao`	13	3	39	1	1	0.31					100	
`threeWayDuel`	10	3	13	0.397	0.42	0.08						
`mutual4-14`	27600	128	384386	1	1	2.74						
`mutual4-15`	27600	527	780504	1	1	3.61						

The objective for `slalom` is given by the LTL formula $\mathsf{G}(p \rightarrow \mathsf{X}\,\mathsf{G}\,\neg q) \wedge \mathsf{G}(q \rightarrow \mathsf{X}\,\mathsf{G}\,\neg p)$. For model `rps1` the strategy found by RL is (slightly) suboptimal. The difference in probability of 0.01 is explained by the existence of many strategies of nearly identical values. Model `mutual` [7,15,34] describes the mutual exclusion protocol of Pnueli and Zuck [29]. Though large, this model is easy for learning.

Figure 5 illustrates how increasing the parameter ζ makes the RL algorithm less sensitive to the presence of transient (not in an end-component) accepting transitions. Model `deferred` consists of two chains of states: one, which the agent choses with action a, has accepting transitions throughout, but leads to an end-component that is not accepting. The other chain, selected with action b, leads to an accepting end-component, but has no other accepting transitions. There are no other decisions in the model; hence only two strategies are possible, which we denote by a and b, depending on the action chosen.

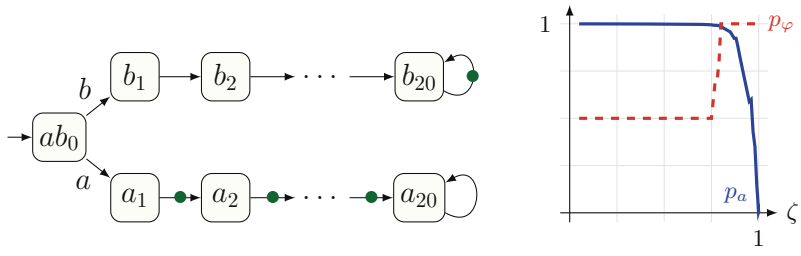

Fig. 5. Model `deferred` and effect of ζ on it.

The curve labeled p_a in Fig. 5 gives the probability of satisfaction under strategy a of the MDP's objective as a function of ζ as computed by Q-learning. The number of episodes is kept fixed at 20,000 and each episode has length 80. Each data point is the average of five experiments for the same value of ζ.

For values of ζ close to 0, the chance is high that the sink is reached directly from a transient state. Consequently, Q-learning considers strategies a and b equally good. For this reason, the probability of satisfaction of the objective, p_φ, according to the strategy that mixes a and b, is computed by Mungojerrie's model checker as 0.5. As ζ approaches 1, the importance of transient accepting transitions decreases, until the probability computed for strategy a is no longer considered to be approximately the same as the probability of strategy b. When that happens, p_φ abruptly goes from 0.5 to its true value of 1, because the pure strategy b is selected. The value of p_a continues to decline for larger values of ζ until it reaches its true value of 0 for $\zeta = 0.9999$. Probability p_b, not shown in the graph, is 1 throughout.

The change in value of p_φ does not contradict Theorem 3, which says that $p_b = 1 > p_a$ for all values of ζ. In practice a high value of ζ may be needed to reliably distinguish between transient and recurrent accepting transitions in numerical computation. Besides, Theorem 3 suggests that even in the almost-sure case there is a meaningful path to the target strategy where the likelihood of satisfying φ can be expected to grow. This is important, as it comes with the promise of a generally increasing quality of intermediate strategies.

6 Conclusion

We have reduced the problem of maximizing the satisfaction of an ω-regular objective in a MDP to reachability on a product graph augmented with a sink state. This change is so simple and elegant that it may surprise that it has not been used before. But the reason for this is equally simple: it does not help in a model checking context, as it does not remove any analysis step there. In a reinforcement learning context, however, it simplifies our task significantly. In previous attempts to use suitable LDBW [4], the complex part of the model checking problem—identifying the accepting end-components—is still present. Only after this step, which is expensive and requires knowledge of the structure

of the underlying MDP, can these methods reduce the search for optimal satisfaction to the problem of maximizing the chance to reach those components. Our reduction avoids the identification of accepting end-components entirely and thus allows a natural integration with a wide range of model-free RL approaches.

References

1. Babiak, T., et al.: The Hanoi omega-automata format. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9206, pp. 479–486. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21690-4_31
2. Baier, C., Katoen, J.-P.: Principles of Model Checking. MIT Press, Cambridge (2008)
3. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-Dynamic Programming. Athena Scientific, Belmont (1996)
4. Brázdil, T., et al.: Verification of Markov decision processes using learning algorithms. In: Cassez, F., Raskin, J.-F. (eds.) ATVA 2014. LNCS, vol. 8837, pp. 98–114. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11936-6_8
5. Brockman, G., et al.: OpenAI Gym. CoRR, abs/1606.01540 (2016)
6. Carton, O., Maceiras, R.: Computing the Rabin index of a parity automaton. Theoret. Inf. Appl. **33**, 495–505 (1999)
7. Chatterjee, K., Gaiser, A., Křetínský, J.: Automata with generalized Rabin pairs for probabilistic model checking and LTL synthesis. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 559–575. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_37
8. Courcoubetis, C., Yannakakis, M.: The complexity of probabilistic verification. J. ACM **42**(4), 857–907 (1995)
9. cpphoafparser (2016). https://automata.tools/hoa/cpphoafparser. Accessesd 05 Sept 2018
10. de Alfaro, L.: Formal Verification of Probabilistic Systems. Ph.D. thesis, Stanford University (1998)
11. Eliot, T.S.: Old Possum's Book of Practical Cats. Harcourt Brace Jovanovich, San Diego (1939)
12. Feinberg, E.A., Shwartz, A. (eds.): Handbook of Markov Decision Processes. Springer, New York (2002). https://doi.org/10.1007/978-1-4615-0805-2
13. Fu, J., Topcu, U.: Probably approximately correct MDP learning and control with temporal logic constraints. In: Robotics: Science and Systems, July 2014
14. Guez, A., et al.: An investigation of model-free planning. CoRR, abs/1901.03559 (2019)
15. Hahn, E.M., Li, G., Schewe, S., Turrini, A., Zhang, L.: Lazy probabilistic model checking without determinisation. In: Concurrency Theory (CONCUR), pp. 354–367 (2015)
16. Hasanbeig, M., Abate, A., Kroening, D.: Logically-correct reinforcement learning. CoRR, abs/1801.08099v1, January 2018
17. Hasanbeig, M., Abate, A., Kroening, D.: Certified reinforcement learning with logic guidance. arXiv e-prints, arXiv:1902.00778, February 2019
18. Hiromoto, M., Ushio, T.: Learning an optimal control policy for a Markov decision process under linear temporal logic specifications. In: Symposium Series on Computational Intelligence, pp. 548–555, December 2015

19. Hordijk, A., Yushkevich, A.A.: Blackwell optimality. In: Feinberg, E.A., Shwartz, A. (eds.) Handbook of Markov Decision Processes: Methods and Applications, pp. 231–267. Springer, Boston (2002). https://doi.org/10.1007/978-1-4615-0805-2_8
20. Krishnan, S.C., Puri, A., Brayton, R.K., Varaiya, P.P.: The Rabin index and chain automata, with applications to automata and games. In: Wolper, P. (ed.) CAV 1995. LNCS, vol. 939, pp. 253–266. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60045-0_55
21. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47
22. Lahijanian, M., Andersson, S.B., Belta, C.: Temporal logic motion planning and control with probabilistic satisfaction guarantees. IEEE Trans. Robot. **28**(2), 396–409 (2012)
23. Li, X., Vasile, C.I., Belta, C.: Reinforcement learning with temporal logic rewards. In: International Conference on Intelligent Robots and Systesm (IROS), pp. 3834–3839 (2017)
24. Manna, Z., Pnueli, A.: The Temporal Logic of Reactive and Concurrent Systems *Specification*. Springer, New York (1991). https://doi.org/10.1007/978-1-4612-0931-7
25. Mnih, V., et al.: Human-level control through reinforcement learning. Nature **518**, 529–533 (2015)
26. Mungojerrie ω-regular reinforcement learning benchmarks (2019). https://plv.colorado.edu/omega-regular-rl-benchmarks-2019
27. OpenAI Gym (2018). https://gym.openai.com. Accessed 05 Sept 2018
28. Perrin, D., Pin, J.É.: Infinite Words: Automata, Semigroups, Logic and Games. Elsevier, Amsterdam (2004)
29. Pnueli, A., Zuck, L.: Verification of multiprocess probabilistic protocols. Distrib. Comput. **1**, 53–72 (1986)
30. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York (1994)
31. Riedmiller, M.: Neural fitted Q iteration – first experiences with a data efficient neural reinforcement learning method. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) ECML 2005. LNCS (LNAI), vol. 3720, pp. 317–328. Springer, Heidelberg (2005). https://doi.org/10.1007/11564096_32
32. 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: IEEE Conference on Decision and Control (CDC), pp. 1091–1096, December 2014
33. Sickert, S., Esparza, J., Jaax, S., Křetínský, J.: Limit-deterministic Büchi automata for linear temporal logic. In: Chaudhuri, S., Farzan, A. (eds.) CAV 2016. LNCS, vol. 9780, pp. 312–332. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41540-6_17
34. Sickert, S., Křetínský, J.: MoChiBA: probabilistic LTL model checking using limit-deterministic Büchi automata. In: Artho, C., Legay, A., Peled, D. (eds.) ATVA 2016. LNCS, vol. 9938, pp. 130–137. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46520-3_9
35. Silver, D., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**, 484–489 (2016)

36. Strehl, A.L., Li, L., Wiewiora, E., Langford, J., Littman, M.L.: PAC model-free reinforcement learning. In: International Conference on Machine Learning ICML, pp. 881–888 (2006)
37. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, 2nd edn. MIT Press, Cambridge (2018)
38. Thomas, W.: Automata on infinite objects. In: Handbook of Theoretical Computer Science, pp. 133–191. The MIT Press/Elsevier, Cambridge (1990)
39. Vardi, M.Y.: Automatic verification of probabilistic concurrent finite state programs. In: Foundations of Computer Science, pp. 327–338 (1985)
40. Wiering, M., van Otterlo, M. (eds.): Reinforcement Learning: State of the Art. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27645-3