# Faithful and Effective Reward Schemes for Model-Free Reinforcement Learning of Omega-Regular Objectives

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

[1] University of Twente, The Netherlands
[2] University of Colorado Boulder, USA
[3] University of Liverpool, UK

**Abstract.** Omega-regular properties—specified using linear time temporal logic or various forms of omega-automata—find increasing use in specifying the objectives of reinforcement learning (RL). The key problem that arises is that of faithful and effective translation of the objective into a scalar reward for model-free RL. A recent approach exploits Büchi automata with restricted nondeterminism to reduce the search for an optimal policy for an $\omega$-regular property to that for a simple reachability objective. A possible drawback of this translation is that reachability rewards are sparse, being reaped only at the end of each episode. Another approach reduces the search for an optimal policy to an optimization problem with two interdependent discount parameters. While this approach provides denser rewards than the reduction to reachability, it is not easily mapped to off-the-shelf RL algorithms. We propose a reward scheme that reduces the search for an optimal policy to an optimization problem with a single discount parameter that produces dense rewards and is compatible with off-the-shelf RL algorithms. Finally, we report an experimental comparison of these and other reward schemes for model-free RL with omega-regular objectives.

## 1 Introduction

A significant challenge to widespread adoption of reinforcement learning (RL) is the faithful translation of designer's intent to the scalar reward signal required by RL algorithms [19]. Logic-based specifications help in two ways: by precisely capturing the intended objective, and by allowing its automatic translation to a reward function. Omega-regular objectives, such as those expressed in Linear Temporal Logic (LTL) [25] and by $\omega$-automata [28], have recently been proposed to specify learning objectives for both model-based [10,20] and model-free RL.

Model-free RL algorithms do not construct a model of the environment; hence, they often scale better than model-based algorithms. However, applying model-free RL to $\omega$-regular properties requires one to address separate concerns:

1. Finding the right automata representation to build product MDPs with $\omega$-regular acceptance conditions on the fly [13].

2. Translating the acceptance condition into a reward assignment that is appropriate for RL, such as reachability or maximizing an overall reward earned.
3. Computing policies that maximize expected reward with a RL technique, like Q-learning, which normally applies discounting to the rewards.

In addressing these concerns, one strives to achieve an overall translation that is *faithful* (maximizing reward means maximizing probability of achieving the objective) and *effective* (RL quickly converges to optimal strategies). In this paper, we focus on the second step, and explore its interplay with the third step when using off-the-shelf RL tools and techniques.

The first approach to learning for $\omega$-regular objectives used deterministic Rabin automata [30]. While the reduction used from Rabin automata to rewards does not have the required correctness properties [12]—there is still no direct translation from Rabin automata to rewards—this work opened the door to using reinforcement learning for temporal and $\omega$-regular properties.

The problems with handling Rabin automata suggest that one should use automata with simpler acceptance mechanisms, like Büchi automata. However, Büchi automata require nondeterminism to recognize all $\omega$-regular languages. Nondeterministic machines can use unbounded look-ahead to resolve nondeterministic choices. However, model checking and reinforcement learning (RL) for Markov Decision Process (MDPs [29]) have a game setting, which restricts the resolution of nondeterminism to be based on the past.

Being forced to resolve nondeterminism on the fly, an automaton may end up rejecting words it should accept, so that using it can lead to incorrect results. Due to this difficulty, initial solutions to game solving and probabilistic model checking have been based on deterministic automata—usually with Rabin or parity acceptance conditions. For two-player games, Henzinger and Piterman proposed the notion of *good-for-games (GFG)* automata [16]. These are nondeterministic automata that simulate [26,15,9] a deterministic automaton that recognizes the same language. The existence of a simulation strategy means that nondeterministic choices can be resolved without look-ahead.

On an MDP, however, the controller is not facing a strategic opponent who may take full advantage of the automaton's inability to resolve nondeterminism on the fly. Vardi was the first to note that probabilistic model checking is possible with Büchi automata only capable of restricted nondeterminism [37]. *Limit deterministic Büchi automata (LDBA)* [6,11,31] make no nondeterministic choice after seeing an accepting transition. They still recognize all $\omega$-regular languages and are, under mild restrictions [31], *suitable* for probabilistic model checking.

The second generation of methods for reinforcement learning therefore used such limit-deterministic Büchi automata [12,3,14]. These papers differ significantly in how they translate the Büchi condition into rewards. The first approach [12] reduces to reachability: in a nutshell, it translates traversing an accepting transition to reaching a fresh target state with a low probability $1 - \zeta$, and to continuing to traverse the product MDP with high probability $\zeta$. The second approach [3] assigns fixed rewards whenever passing an accepting transition, while using a complex discounting strategy: when passing an accepting transition, the reward is given and a discount factor of $\gamma_B \in ]0,1[$ is applied to the remaining rewards, whereas when traversing a non-accepting transition no reward is given, and a different discount factor $\gamma \in ]0,1[$ is applied. For the approach to be correct, it is required that $\gamma_B$ be a function of $\gamma$ with the property that, when $\gamma$ goes to 1, $\frac{1-\gamma}{1-\gamma_B(\gamma)}$ goes to 0. The advantage of this method is that rewards appear earlier, but at the cost of having two parameters (that are not independent of each other), and an overhead in the calculation. The third approach [14] uses a constant discount factor $\gamma \in ]0,1]$, which (while not technically correct [12,3]) usually works and provides good results in practice.

We use transformations on the reward structure from our reachability reduction in [12] to infer simple alternative total and discounted reward structures that favor the same strategies as the reachability reduction from [12] and therefore inherit the correctness from there. The total reward structure keeps the accepting sink, and simply provides a reward whenever an accepting transition is taken, regardless of whether or not the sink is reached. We show that this increases the expected payoff obtained from a strategy by a constant factor, thus preserving preferences between different strategies.

The discounted reward structure does not introduce the accepting sink, but works on the unadjusted product MDP. It uses a biased discount, where a discount is only made when an accepting transition is passed. This is closely related to [3], but keeps the vital separation of concerns that allows us to keep the proofs simple and the method easy to use and understand: We introduce a reduction that produces a faithful reward structure with a single variable $\zeta$. Coupled to a learning technique that uses discounted rewards, our approach is equivalent to that of [3] (though it suggests that $\gamma$ is really a function of $\gamma_B$, not the other way round), but with a clear separation of concerns: the smaller factor $\gamma_B$ (which corresponds to $\zeta \cdot \gamma$ in this setting) is the ingredient that makes the reward structure faithful, the larger discount factor $\gamma$ simply provides contraction.

A good reward scheme should promote fast learning by giving dense rewards with low variance. It should also be compatible with off-the-shelf RL algorithms, so that state-of-the-art algorithms may be used promptly and with little effort. The rewards produced by the scheme of [12] tend to be sparse because they are only possible at the end of an episode, when the target state is reached. On the other hand, the reachability-based rewards can be directly used with any off-the-shelf RL algorithm [17,32]. The total reward structure provides dense rewards and is straightforward to integrate with off-the-shelf RL algorithms. However, it is affected by the high variability of the return. Discounted rewards fix the problem with variability, but require the implementations of RL algorithms to accommodate state-dependent discounts.

While the reward transformations are ways to 'shape' rewards in the literal sense of arranging them in a way that they appear early, they are orthogonal to classic reward shaping techniques like adding potentials to MDP states [27,22]. As an orthogonal approach, it is a potentially helpful addition to all the reward schemes discussed above.

Reward machines [18,5] is a related notion of providing a formal structure to specify rewards. Reward machines are Mealy machines where the inputs are the observations of the MDP and the outputs are scalar rewards. The key difference of reward machines from ours is that reward machines interpret specification of finite traces (e.g. LTL on finite prefixes [8]). Moreover, they allow specification of arbitrary scalar rewards for various events, while in our work the reward is given strictly according to the formal specification.

This paper is organized as follows. After the preliminaries, we first introduce the novel total reward and then the new faithful total rewards based on biased discounts (Sections 3.2 and 3.3) for good-for-MDP automata. In Section 4, we discuss how to use Q-learning for this faithful reward scheme. In Section 5, we evaluate the impact of the contributions of the paper on reinforcement learning algorithms. Section 6 presents conclusions.

## 2 Preliminaries

A *nondeterministic Büchi automaton* is a tuple $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \Gamma \rangle$, where $\Sigma$ is a finite *alphabet*, $Q$ is a finite set of *states*, $q_0 \in Q$ is the *initial state*, $\Delta \subseteq Q \times \Sigma \times Q$ are transitions, and $\Gamma \subseteq Q \times \Sigma \times Q$ is the transition-based *acceptance condition*.

A *run* $r$ of $\mathcal{A}$ on $w \in \Sigma^\omega$ is an $\omega$-word $r_0, w_0, r_1, w_1, \ldots$ in $(Q \times \Sigma)^\omega$ such that $r_0 = q_0$ and, for $i > 0$, it is $(r_{i-1}, w_{i-1}, r_i) \in \Delta$. We write $\mathsf{inf}(r)$ for the set of transitions that appear infinitely often in the run $r$. A run $r$ of $\mathcal{A}$ is *accepting* if $\mathsf{inf}(r) \cap \Gamma \neq \emptyset$.

The *language*, $L_{\mathcal{A}}$, of $\mathcal{A}$ (or, *recognized* by $\mathcal{A}$) is the subset of words in $\Sigma^\omega$ that have accepting runs in $\mathcal{A}$. A language is *$\omega$-regular* if it is accepted by a Büchi automaton. An automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \Delta, \Gamma \rangle$ is *deterministic* if $(q, \sigma, q'), (q, \sigma, q'') \in \Delta$ implies $q' = q''$. $\mathcal{A}$ is *complete* if, for all $\sigma \in \Sigma$ and all $q \in Q$, there is a transition $(q, \sigma, q') \in \Delta$. A word in $\Sigma^\omega$ has exactly one run in a deterministic, complete automaton.

A *Markov decision process (MDP)* $\mathcal{M}$ is a tuple $\langle S, s_0, A, T, \Sigma, L \rangle$ where $S$ is a finite set of states, $s_0$ is a designated initial state, $A$ is a finite set of *actions*, $T : S \times A \to \mathcal{D}(S)$, where $\mathcal{D}(S)$ is the set of probability distributions over $S$, is the *probabilistic transition function*, $\Sigma$ is an alphabet, and $L : S \times A \times S \to \Sigma$ is the *labeling function* of the set of transitions. For a state $s \in S$, $A(s)$ denotes the set of actions available in $s$. For states $s, s' \in S$ and $a \in A(s)$, we have that $T(s,a)(s')$ equals $\Pr(s'|s,a)$.

A *run* of $\mathcal{M}$ is an $\omega$-word $s_0, a_1, \ldots \in S \times (A \times S)^\omega$ such that $\Pr(s_{i+1}|s_i, a_{i+1}) > 0$ for all $i \geq 0$. A finite run is a finite such sequence. For a *run* $r = s_0, a_1, s_1, \ldots$ we define the corresponding labeled run as $L(r) = L(s_0, a_1, s_1), L(s_1, a_2, s_2), \ldots \in \Sigma^\omega$. We write $\mathit{Runs}(\mathcal{M})$ ($\mathit{FRuns}(\mathcal{M})$) for the set of runs (finite runs) of $\mathcal{M}$ and $\mathit{Runs}_s(\mathcal{M})$ ($\mathit{FRuns}_s(\mathcal{M})$) for the set of runs (finite runs) of $\mathcal{M}$ starting from state $s$. When the MDP is clear from the context we drop the argument $\mathcal{M}$.

A strategy in $\mathcal{M}$ is a function $\mu : \mathit{FRuns} \to \mathcal{D}(A)$ such that for all finite runs $r$ we have $\mathit{supp}(\mu(r)) \subseteq A(\mathsf{last}(r))$, where $\mathit{supp}(d)$ is the support of $d$ and $\mathsf{last}(r)$

is the last state of $r$. Let $Runs^{\mu}_{s}(\mathcal{M})$ denote the subset of runs $Runs_s(\mathcal{M})$ that correspond to strategy $\mu$ and initial state $s$. Let $\Sigma_{\mathcal{M}}$ be the set of all strategies. A strategy $\mu$ is *pure* if $\mu(r)$ is a point distribution for all runs $r \in FRuns$ and we say that $\mu$ is *stationary* if $\mathsf{last}(r) = \mathsf{last}(r')$ implies $\mu(r) = \mu(r')$ for all runs $r, r' \in FRuns$. A strategy is *positional* if it is both pure and stationary.

The behavior of an MDP $\mathcal{M}$ under a strategy $\mu$ with starting state $s$ is defined on a probability space $(Runs^{\mu}_{s}, \mathcal{F}^{\mu}_{s}, \Pr^{\mu}_{s})$ over the set of infinite runs of $\mu$ from $s$. Given a random variable over the set of infinite runs $f : Runs \to \mathbb{R}$, we write $\mathbb{E}^{\mu}_{s}\{f\}$ for the expectation of $f$ over the runs of $\mathcal{M}$ from state $s$ that follow strategy $\mu$. A *Markov chain* is an MDP whose set of actions is singleton. For any MDP $\mathcal{M}$ and stationary strategy $\mu$, let $\mathcal{M}_{\mu}$ be the Markov chain resulting from choosing the actions in $\mathcal{M}$ according to $\mu$.

Given an MDP $\mathcal{M}$ and an automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \Gamma \rangle$, we want to compute an optimal strategy satisfying the objective that the run of $\mathcal{M}$ is in the language of $\mathcal{A}$. We define the *semantic satisfaction* probability for $\mathcal{A}$ and a strategy $\mu$ from state $s$ as:

$$\begin{aligned}\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s,\mu) &= \Pr{}^{\mu}_{s}\big\{r{\in}Runs^{\mu}_{s} : L(r){\in}L_{\mathcal{A}}\big\} \text{ and}\\ \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s) &= \sup_{\mu}\big(\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s,\mu)\big)\ .\end{aligned}$$

A strategy $\mu_*$ is optimal for $\mathcal{A}$ if $\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s,\mu_*) = \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s)$.

When using automata for the analysis of MDPs, we need a syntactic variant of the acceptance condition. Given an MDP $\mathcal{M} = \langle S, s_0, A, T, \Sigma, L \rangle$ and an automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \Delta, \Gamma \rangle$, the *product* $\mathcal{M} \times \mathcal{A} = \langle S \times Q, (s_0, q_0), A \times Q, T^{\times}, \Gamma^{\times} \rangle$ is an MDP augmented with an initial state $(s_0, q_0)$ and accepting transitions $\Gamma^{\times}$. The function $T^{\times} : (S \times Q) \times (A \times Q) \rightharpoonup \mathcal{D}(S \times Q)$ is defined by

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

Finally, $\Gamma^{\times} \subseteq (S{\times}Q){\times}(A{\times}Q){\times}(S{\times}Q)$ is defined by $((s,q),(a,q'),(s',q')) \in \Gamma^{\times}$ if, and only if, $(q, L(s,a,s'), q') \in \Gamma$ and $T(s,a)(s') > 0$. A strategy $\mu^{\times}$ on the product defines a strategy $\mu$ on the MDP with the same value, and vice versa. (For a stationary $\mu^{\times}$, $\mu$ may need memory.) We define the *syntactic satisfaction* probabilities as

$$\begin{aligned}\mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}((s,q),\mu^{\times}) &= \Pr{}^{\mu}_{s}\big\{r \in Runs^{\mu^{\times}}_{(s,q)}(\mathcal{M}\times\mathcal{A}) : \inf(r) \cap \Gamma^{\times} \neq \emptyset\big\}\\ \mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}(s) &= \sup_{\mu^{\times}}\big(\mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}((s,q_0),\mu^{\times})\big)\ .\end{aligned}$$

Note that $\mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}(s) = \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s)$ holds for a deterministic $\mathcal{A}$. In general, $\mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}(s) \leq \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s)$ holds, but equality is not guaranteed because the optimal resolution of nondeterministic choices may require access to future events.

An automaton $\mathcal{A}$ is *good for MDPs* (GFM), if $\mathsf{PSat}^{\mathcal{M}}_{\mathcal{A}}(s_0) = \mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s_0)$ holds for all MDPs $\mathcal{M}$ [13]. For an automaton to match $\mathsf{PSem}^{\mathcal{M}}_{\mathcal{A}}(s_0)$, its nondeterminism is restricted not to rely heavily on the future; rather, it must be possible to resolve the nondeterminism on-the-fly. In this paper we only consider GFM automata, which have this ability.

For $\omega$-regular objectives, optimal satisfaction probabilities and strategies can be computed using graph-theoretic techniques over the product structure. However, when the MDP transition structure is unknown, such techniques are not applicable. Model-free reinforcement learning overcomes this limitation.

## 3 Faithful Translation of Objectives to Rewards

The problem we address is the following:

> *Given MDP $\mathcal{M}$ with unknown transition structure and a GFM Büchi automaton $\mathcal{A}$ accepting an $\omega$-regular objective $\varphi$, compute a strategy optimal for $\mathcal{A}$, that is, a strategy that maximizes the probability that $\mathcal{M}$ satisfies $\varphi$.*

*Reinforcement learning* (RL) provides a framework to compute optimal strategies from repeated interactions with an MDPs with unknown transition structure. It consists of maximizing the expectation of a scalar reward. Of the two main approaches to RL, *model-free* and *model-based*, the former, which is asymptotically space-efficient [33], has been shown to scale well [35].

Bridging the gap between $\omega$-regular specifications and model-free RL requires a translation from specification to scalar reward such that a model-free RL algorithm maximizing scalar rewards produces a policy that maximizes the probability to satisfy the specification. We call this requirement *faithfulness*. Another key requirement on such a translation is *effectiveness*: the reward should be formulated to help mainstream RL algorithms (such as Q-learning [38]) to reliably and quickly learn such optimal policies. We next present three solutions to the faithfulness requirement. From the approach of [12] we derive two reward schemes that translate the maximization of satisfaction probability to total reward and discounted reward problems. In Sections 4 and 5 we discuss their effectiveness.

### 3.1 Reachability Rewards

The reduction from [12] (see Figure 1) was the first faithful translation of $\omega$-regular objectives to scalar rewards for model-free RL. Maximizing the chance to realize an $\omega$-regular objective given by an MDP Büchi automaton $\mathcal{A}$ for an MDP $\mathcal{M}$ is reduced to maximizing the chance to meet the reachability objective in the augmented MDP $\mathcal{R}^\zeta$ (for $\zeta \in ]0,1[$) obtained from $\mathcal{M} \times \mathcal{A}$ by

- adding a new target state $t$ (either as a sink with a self-loop or as a point where the computation stops; we choose here the latter view) and by
- making the target $t$ a destination of each accepting transition $\tau$ of $\mathcal{M} \times \mathcal{A}$ with probability $1 - \zeta$ and multiplying the original probabilities of all other destinations of an accepting transition $\tau$ by $\zeta$.

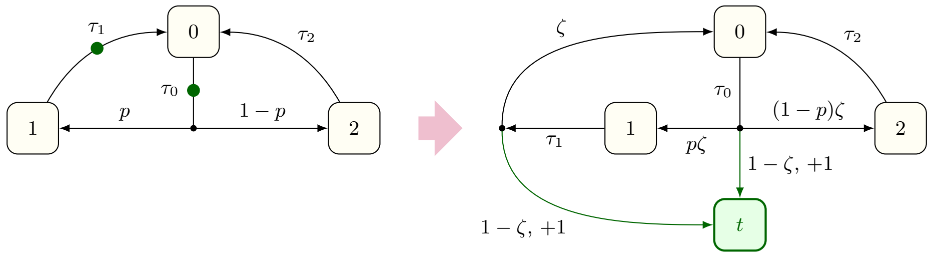

Fig. 1: Reachability reward scheme.

We define the probability to reach the sink $t$ in $\mathcal{R}^\zeta$ as

$$\mathsf{PSat}_t^{\mathcal{R}^\zeta}((s,q),\mu) = \Pr{}_s^\mu\{r \in \mathit{Runs}_{(s,q)}^\mu(\mathcal{R}^\zeta) : r \text{ reaches } t\}$$

$$\mathsf{PSat}_t^{\mathcal{R}^\zeta}(s) = \sup_\mu \big(\, \mathsf{PSat}_t^{\mathcal{R}^\zeta}((s,q_0),\mu)\big) \enspace .$$

**Theorem 1 ([12]).** *The following holds:*

1. *$\mathcal{R}^\zeta$ (for $\zeta \in ]0,1[$) and $\mathcal{M}\times\mathcal{A}$ have the same set of strategies.*
2. *For a positional strategy $\mu$, the chance of reaching the target $t$ in $\mathcal{R}^\zeta_\mu$ is 1 if, and only if, the chance of satisfying the Büchi objective in $(\mathcal{M}\times\mathcal{A})_\mu$ is 1: $\mathsf{PSat}_t^{\mathcal{R}^\zeta}((s_0,q_0),\mu) = 1 \;\Leftrightarrow\; \mathsf{PSat}_\mathcal{A}^\mathcal{M}((s_0,q_0),\mu) = 1$.*
3. *There is a $\zeta_0 \in ]0,1[$ such that, for all $\zeta \in [\zeta_0,1[$, an optimal reachability strategy $\mu$ for $\mathcal{R}^\zeta$ is an optimal strategy for the Büchi objective in $\mathcal{M}\times\mathcal{A}$: $\mathsf{PSat}_t^{\mathcal{R}^\zeta}((s_0,q_0),\mu) = \mathsf{PSat}_t^{\mathcal{R}^\zeta}(s_0) \;\Rightarrow\; \mathsf{PSat}_\mathcal{A}^\mathcal{M}((s_0,q_0),\mu) = \mathsf{PSat}_\mathcal{A}^\mathcal{M}(s_0)$.*

## 3.2 Total and Dense Rewards

Theorem 1 proves the faithfulness of the translation to reachability of [12], which, however, has a drawback. For $\zeta$ close to 1, the rewards occur late: they are *sparse*. Addressing this concern leads to our second translation, which produces *denser* rewards and reduces the problem to the maximization of *total* reward.

We build, for a GFM Büchi automaton $\mathcal{A}$ and an MDP $\mathcal{M}$, the augmented MDP $\mathcal{T}^\zeta$ (for $\zeta \in ]0,1[$) obtained from $\mathcal{M}\times\mathcal{A}$ in the same way as $\mathcal{R}^\zeta$, i.e., by

– adding a new sink state $t$ (as a sink where the computation stops) and
– by making the sink $t$ a destination of each accepting transition $\tau$ of $\mathcal{M}\times\mathcal{A}$ with probability $1-\zeta$ and by multiplying the original probabilities of all other destinations of an accepting transition $\tau$ by $\zeta$.

Unlike $\mathcal{R}^\zeta$, MDP $\mathcal{T}^\zeta$ is equipped with a total reward (also known as undiscounted reward) objective, where taking an accepting (in $\mathcal{M}\times\mathcal{A}$) transition $\tau$ provides a reward of 1, regardless of whether it leads to the sink $t$.

Let $N(r)$ be the number of accepting transitions in a run $r$ of $\mathcal{T}^\zeta$. Then,

$$\mathsf{ETotal}^{\mathcal{T}^\zeta}((s,q),\mu) = \mathbb{E}_{(s,q)}^\mu\{N(r) : r \in \mathit{Runs}_{(s,q)}^\mu(\mathcal{T}^\zeta)\}$$

$$\mathsf{ETotal}^{\mathcal{T}^\zeta}(s) = \sup_\mu \big(\mathsf{ETotal}^{\mathcal{T}^\zeta}((s,q_0),\mu)\big) \enspace .$$

Note that the set of runs with $N(r) = \infty$ has probability 0 in $\mathit{Runs}^{\mu}_{(s,q)}(\mathcal{T}^{\zeta})$: they are the runs that infinitely often do not move to $t$ on an accepting transition, where the chance that this happens at least $n$ times is $\zeta^n$ for all finite $n$.

**Theorem 2.** *The following holds:*

1. *MDP $\mathcal{T}^{\zeta}$ (for $\zeta \in ]0,1[$), MDP $\mathcal{R}^{\zeta}$ (for $\zeta \in ]0,1[$), and product MDP $\mathcal{M} \times \mathcal{A}$ have the same set of strategies.*
2. *For a positional strategy $\mu$, the expected reward for $\mathcal{T}^{\zeta}_{\mu}$ is $r$ if, and only if, the chance of reaching the target $t$ in $\mathcal{R}^{\zeta}_{\mu}$ is $r/(1-\zeta)$:*
   $\mathsf{PSat}^{\mathcal{R}^{\zeta}}_{t}((s_0,q_0),\mu) = (1-\zeta)\mathsf{ETotal}^{\mathcal{T}^{\zeta}}((s_0,q_0),\mu)$.
3. *The expected reward for $\mathcal{T}^{\zeta}_{\mu}$ is in $\big[0,(1-\zeta)^{-1}\big]$.*
4. *The chance of satisfying the Büchi objective in $(\mathcal{M} \times \mathcal{A})_{\mu}$ is 1 if, and only if, the expected reward for $\mathcal{T}^{\zeta}_{\mu}$ is $(1-\zeta)^{-1}$.*
5. *There is a $\zeta_0 \in ]0,1[$ such that, for all $\zeta \in [\zeta_0,1[$, a strategy $\mu$ that maximizes the reward for $\mathcal{T}^{\zeta}$ is an optimal strategy for the Büchi objective in $\mathcal{M} \times \mathcal{A}$.*

*Proof.* (1) Obvious, because all the states and their actions are the same apart from the sink state $t$ for which the strategy can be left undefined.

(2) The sink state $t$ can only be visited once along any run, so the expected number of times a run starting at $(s_0,q_0)$ while using $\mu$ is going to visit $t$ is the same as its probability of visiting $t$ , i.e., $\mathsf{PSat}^{\mathcal{R}^{\zeta}}_{t}((s_0,q_0),\mu)$. The only way $t$ can be reached is by traversing an accepting transition and this always happens with the same probability $(1-\zeta)$. So the expected number of visits to $t$ is the expected number of times an accepting transition is used, i.e., $\mathsf{ETotal}^{\mathcal{T}^{\zeta}}((s_0,q_0),\mu)$, multiplied by $(1-\zeta)$.

(3) follows from (2), because $\mathsf{PSat}^{\mathcal{R}^{\zeta}}_{t}((s_0,q_0),\mu)$ cannot be greater than 1.

(4) follows from (2) and Theorem 1 (2).

(5) follows from (2) and Theorem 1 (3). □

## 3.3 Discounted and Dense Rewards

The expected undiscounted reward for $\mathcal{T}^{\zeta}_{\mu}$ can be viewed as the expected total sum of dynamically discounted rewards for $(\mathcal{M} \times \mathcal{A})_{\mu}$, by giving a reward of $\zeta^i$ when passing through an accepting transition when $i$ *accepting* transitions have been used before. We call these *$\zeta$-biased discounted rewards.*

Let $\mathcal{D} = \mathcal{M} \times \mathcal{A}$ and, for a run $r$ with $N(r) = n$ accepting transitions, let the $\zeta$-biased discounted reward be $\mathsf{Disct}_{\zeta}(r) = \sum_{i=0}^{n-1} \zeta^i = \frac{1-\zeta^n}{1-\zeta}$ if $n < \infty$ or $\mathsf{Disct}_{\zeta}(r) = \sum_{i=0}^{\infty} \zeta^i = \frac{1}{1-\zeta}$ if $n = \infty$. Let

$$\mathsf{EDisct}^{\mathcal{D}}_{\zeta}((s,q),\mu) = \mathbb{E}^{\mu}_{(s,q)}\big\{\mathsf{Disct}_{\zeta}(r) : r \in \mathit{Runs}^{\mu}_{(s,q)}(\mathcal{D})\big\}$$
$$\mathsf{EDisct}^{\mathcal{D}}_{\zeta}(s) = \sup_{\mu}\big(\mathsf{EDisct}^{\mathcal{D}}_{\zeta}((s,q_0),\mu)\big) \ .$$

**Theorem 3.** *For every positional strategy $\mu$, the expected reward for $\mathcal{T}^{\zeta}_{\mu}$ is equal to the expected total $\zeta$-biased discounted reward for $\mathcal{D}_{\mu}$, i.e., for every start state $(s,q)$ we have:* $\mathsf{EDisct}^{\mathcal{D}}_{\zeta}((s,q),\mu) = \mathsf{ETotal}^{\mathcal{T}^{\zeta}}((s,q),\mu)$.

*Proof.* Note that for any start state $(s,q)$ and $n \geq 0$:

$$\Pr{}^{\mu}_{s}\big\{r \in \mathit{Runs}^{\mu}_{(s,q)}(\mathcal{T}^{\zeta}) : N(r) > n\big\} = \Pr{}^{\mu}_{s}\big\{r \in \mathit{Runs}^{\mu}_{(s,q)}(\mathcal{D}) : N(r) > n\big\} \cdot \zeta^{n} \ .$$

This is because the only transition-wise difference between $\mathcal{T}^{\zeta}$ and $\mathcal{D}$ is that every time an accepting transition is passed through in $\mathcal{T}^{\zeta}$, the process stops at the sink node with probability $1-\zeta$. Therefore, in order to use more than $n$ accepting transitions in $\mathcal{T}^{\zeta}$, the non-stopping option has to be chosen $n$ times in a row, each time with probability $\zeta$.

For any random variable $X : \Omega \to \mathbb{N} \cup \{\infty\}$ we have $\mathbb{E}X = \sum_{n\geq 0} \Pr(X > n)$. Now from the definition of $\mathsf{EDisct}^{\mathcal{D}}_{\zeta}((s,q),\mu)$ and $\mathsf{ETotal}^{\mathcal{T}^{\zeta}}((s,q),\mu)$ we get:

$$\begin{aligned}
\mathsf{EDisct}^{\mathcal{D}}_{\zeta}((s,q),\mu) &= \sum_{n\geq 1} \Pr{}^{\mu}_{(s,q)}\big\{r \in \mathit{Runs}^{\mu}_{(s,q)}(\mathcal{D}) : N(r) = n\big\} \cdot \sum_{0\leq i<n} \zeta^{i} \\
&\quad + \Pr{}^{\mu}_{(s,q)}\big\{r \in \mathit{Runs}^{\mu}_{(s,q)}(\mathcal{D}) : N(r) = \infty\big\} \cdot \sum_{i\geq 0} \zeta^{i} \\
&\overset{(*)}{=} \sum_{n\geq 0} \Pr{}^{\mu}_{(s,q)}\big\{r \in \mathit{Runs}^{\mu}_{(s,q)}(\mathcal{D}) : N(r) > n\big\} \cdot \zeta^{n} \\
&= \sum_{n\geq 0} \Pr{}^{\mu}_{(s,q)}\big\{r \in \mathit{Runs}^{\mu}_{(s,q)}(\mathcal{T}^{\zeta}) : N(r) > n\big\} \\
&= \mathsf{ETotal}^{\mathcal{T}^{\zeta}}((s,q),\mu) \ ,
\end{aligned}$$

where $(*)$ follows by expanding the products and joining up the terms that have a common factor of the form $\zeta^{n}$. □

This improves over [3] because it provides a clearer separation of concerns: the only discount factor represents the *translation to reachability.* The use of Q-learning [35] introduces two other parameters, the discount factor $\gamma$ and the learning rate $\alpha$, with $\gamma, \alpha \in ]0,1[$. For fixed parameters, Q-learning works in the limit when the parameters are chosen in the right order—e.g., $\lim_{\gamma\uparrow 1} \lim_{\alpha\downarrow 0}$ of the expected value works, while $\lim_{\alpha\downarrow 0} \lim_{\gamma\uparrow 1}$ does not—and when experimenting with different learning approaches, it is useful to separate concerns, rather then mixing parameters from the learning mechanism with those required for faithfulness.

Thus, using only one discount parameter, $\zeta$, instead of two (called $\gamma$ and $\gamma_B$ in [3]) parameters (that are not independent) to guarantee faithfulness provides a clean separation of concerns: Reinforcement learning will still use a discount for effectiveness, but the role of the two parameters is neatly separated. This formulation offers a simpler proof, and provides better intuition: discount whenever you have earned a reward. It also lends itself to implementation with convergent RL algorithms—as long as they support state-dependent discounts. The discount rate on accepting edges is multiplied by $\zeta$ instead of assigning a decaying reward of $\zeta^i$. This does not change the optimal strategies and the expected reward from the initial state remains the same.

## 4 Q-Learning and Effectiveness

We next discuss the applicability of Q-learning to the faithful reward schemes presented in Section 3. Recourse to Blackwell optimality allows us to deal also with undiscounted rewards, even though a naive use of Q-learning may produce incorrect results in this case.

Q-learning [38] is a well-studied model-free RL approach to compute an optimal strategy for discounted rewards. Q-learning computes so-called Q-values for every state-action pair. Intuitively, once Q-learning has converged to the fixed point, $Q(s, a)$ is the optimal reward the agent can get while performing action $a$ after starting at $s$. The Q-values can be initialized arbitrarily, but ideally they should be close to the actual values. Q-learning learns over a number of episodes, each consisting of a sequence of actions with bounded length. An episode can terminate early if a sink-state or another non-productive state is reached. Each episode starts at the designated initial state $s_0$. The Q-learning process moves from state to state of the MDP using one of its available actions and accumulates rewards along the way. Suppose that in the $i$-th step, the process has reached state $s_i$. It then either performs the currently (believed to be) optimal action $a_i = \max_a Q_i(s_{i+1}, a)$ (so-called *exploitation* option) or, with probability $\epsilon$, picks uniformly at random one of the actions available at $s_i$ (so-called *exploration* option). Either way, the Q-value is updated as follows:

$$Q_{i+1}(s_i, a_i) = (1 - \alpha_i)Q_i(s_i, a_i) + \alpha_i(r_i + \gamma \cdot \max_a Q_i(s_{i+1}, a)) \ ,$$

where $\alpha_i \in ]0, 1[$ is the learning rate and $\gamma \in ]0, 1]$ is the discount factor. Note the model-freeness: this update does not depend on the set of transitions nor their probabilities. For all other pairs $s, a$ we have $Q_{i+1}(s, a) = Q_i(s, a)$, i.e., they are left unchanged. Watkins and Dayan showed the convergence of $Q$-learning [38].

**Theorem 4 (Convergence).** *For $\lambda < 1$, bounded rewards $|r_i| \leq B$ and learning rates $0 \leq \alpha_i < 1$ satisfying: $\sum_{i=0}^{\infty} \alpha_i = \infty$ and $\sum_{i=0}^{\infty} \alpha_i^2 < \infty$, we have that $Q_i(x, a) \to Q(s, a)$ as $i \to \infty$ for all $s, a \in S \times A$ almost surely.*

However, in the total reward setting that corresponds to $Q$-learning with discount factor $\gamma = 1$, $Q$-learning may not converge, or converge to incorrect values as shown below.

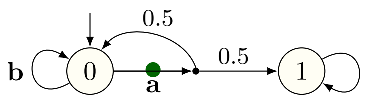

Fig. 2: Q-learning with total reward may converge to a wrong fixed point or not at all. The accepting transition is marked with a green dot.

*Example 1.* Consider the MDP in Figure 2 with reachability rewards (Section 3.1) and assume the following parameters: $\alpha = \zeta = 1/2$, $\epsilon > 0$, $\gamma = 1$. All $Q$-values are initialized to 0. It can be checked that after taking action **a** at state 0 and reaching the sink-state $t$ (with probability $1-\zeta$, not depicted) would result in setting $Q(0,\mathbf{a}) = (1-\alpha)\cdot 0+\alpha\cdot 1 = 1/2$. Repeating this $n$ times in a row (with probability $\geq (\epsilon/2)^n$) would lead to $Q(0,\mathbf{a}) = 1-1/2^n$. Taking then $m$ times action $\mathbf{b}$ (again with positive probability), would result in setting $Q(0,\mathbf{b})$ to $(1-1/2^m)(1-1/2^n)$, which tends to 1 as $n$ and $m$ increase. Note that the value of $Q(0,\mathbf{b})$ can never decrease as its update rule is $Q(0,\mathbf{b}) = \max_a Q(0,a)$. Therefore, even if Q-learning converges, $\max_a Q(0,a)$ can be far away from the actual value of state 0, which is clearly smaller than 3/4, as the dead-end node 2 with 0 reward is reached with probability $> \zeta/2 = 1/4$. The situation is even worse when we consider total and dense reward (Section 3.2). Following the same learning path (but never reaching the sink-state $t$) would result in $Q(0,\mathbf{b}) = n/2$ and $Q(0,\mathbf{a}) = (1-1/2^m)n/2$, and so $Q(0,a)$ will almost surely diverge to $\infty$ as the number of episodes increases.

To solve total-reward problems using $Q$-learning, we exploit the concept of Blackwell-optimal strategies. Given an MDP $\mathcal{M}$, we say that a strategy $\mu$ is Blackwell-optimal if there exists a $\lambda_0 \in ]0,1[$ such that $\mu$ is $\lambda$-discount optimal for all $\lambda \in ]\lambda_0, 1[$. Moreover, if $\mathcal{M}$ has $n$ states and all transition probabilities are rational with numerator and denominator bounded from above by $M$, then $\lambda_0$ is bounded from above by $1-((n!)^2 2^{2n+3} M^{2n^2})^{-1}$ [24,17,1]. The following theorem enables the application of $Q$-learning for discounted reward problem for total-reward when total rewards are bounded.

**Theorem 5 (Blackwell-Optimality [23]).** *Let $\mathcal{M}$ be an MDP and $\rho : S \times A \to \mathbb{R}$ be a reward function such that for every strategy $\mu$ of $\mathcal{M}$ expected total reward is finite, then every Blackwell-optimal strategy is total-reward optimal.*

All of the reward schemes introducted in the previous section can be reduced to total reward objectives with bounded expected total reward and hence $Q$-learning can be applied with discount factor left as a hyperparameter.

## 5 Experiments

We carried out our experiments in the tool Mungojerrie [12], which reads MDPs described in the PRISM language [21], and $\omega$-regular automata written in the HOA format [2,7]. Mungojerrie provides an interface for RL algorithms akin to that of [4] and supports probabilistic model checking.

We compared four reward schemes. Reachability reward (RR) is the scheme from [12]. Total reward (TR) is the scheme from Section 3.2. Discounted reward (DR) is the scheme from Section 3.3, which is equivalent to that of [3]. We will consider these methods the same in our analysis. Simple reward (SR) is a reward mechanism which provides +1 reward on accepting edges and 0 reward otherwise, similar to [30] restricted to Büchi objectives. Although this reward scheme is known not to be faithful [12], we compare it to see its practical value.

For our RL algorithm, we selected Q-learning [38] due to its widespread use and convergence guarantees. As with any RL algorithm, the performance of Q-learning depends on the hyperparameter values. We ran a grid search on the

Table 1: Q-learning results. Times are in seconds.

| Name | states | aut. | prod. | prob. | RR | TR | DR | SR |
|---|---|---|---|---|---|---|---|---|
| `twoPairs` | 4 | 4 | 16 | 1 | 0.04 | 0.04 | 0.06 | 1.23 |
| `riskReward` | 4 | 2 | 8 | 1 | **0.05**,0.05 | 0.30 | **0.07**,0.07 | 0.97 |
| `deferred` | 41 | 1 | 41 | 1 | 0.11 | **0.04**,0.23 | 0.12 | 1.43 |
| `grid5x5` | 25 | 3 | 75 | 1 | 1.56 | **6.97**,7.34 | 0.46 | 20.69 |
| `trafficNtk` | 122 | 13 | 462 | 1 | **0.09**,0.09 | 0.73 | **0.13**,0.14 | 2.11 |
| `windy` | 123 | 2 | 240 | 1 | **3.34**,3.64 | **29.37**,35.18 | 1.14 | 18.64 |
| `windyKing` | 130 | 2 | 256 | 1 | 1.02 | **20.55**,20.69 | 1.30 | 36.77 |
| `windyStoch` | 130 | 2 | 260 | 1 | **42.94**,52.28 | 56.57 | 4.28 | 67.55 |
| `frozenSmall` | 16 | 3 | 48 | 0.823 | **0.29**,0.48 | **0.83**,1.08 | 0.38 | 8.74 |
| `frozenLarge` | 64 | 3 | 192 | 1 | 0.94 | **6.47**,8.98 | 2.08 | 25.34 |
| `othergrid6` | 36 | 25 | 352 | 1 | **1.06**,1.17 | **5.31**,12.77 | 1.70 | 44.58 |

RMACC Summit supercomputer [34] across hyperparameter combinations for all examples and methods. The examples are taken from [12]. In the grid search, we varied $\zeta$ (equivalently, $\gamma_B$), the exploration rate $\epsilon$, the learning rate $\alpha$, the episode number, the episode length, and whether the learning rate decayed. The variations made to these parameters were selected by hand. Statistics for each grid point are based on three runs. The grid search required 207,900 runs and took over 100 days of CPU time. All methods require a sufficiently high discount factor. We used the very high value of $\gamma = 0.99999$ for all runs. A value so close to 1 is prone to cause, in practice, many of the problems that may occur in the undiscounted case. However, we also experimented with lower discount factors, and they provided very similar results.

The selection of the "best" parameters from our grid search makes use of two criteria. Criterion 1 is based on reward maximization. Given a reward scheme, an automaton, and an MDP, there is a maximum reward achievable. In the spirit of the model-free application of these methods, we estimated these maxima based on the recorded initial value in the Q-table. This can be determined without knowledge of the structure of the MDP and without additional computation. We then removed all sets of parameters which produced average values that were not within 5% of the maximum. If Q-learning has converged, we know that the value of the initial state of the Q-table is the value of the optimal strategy. However, if Q-learning has overestimated these values, then this criterion will select parameters that are the most prone to overestimation. Criterion 2 is based on using full knowledge of the MDP and access to the model checker. We fixed the strategies produced after learning completed and used the model checker to compute the probability of satisfaction of the property under these strategies. We then removed all sets of parameters which produced average values that were not within 1% of the maximum probability of satisfaction of the property.

In Table 1, we report the fastest time of all parameter values that remain after applying Criterion 1. Of these, we mark with bold red face those that fail Criterion 2 and report the fastest time of all parameter values that remain after applying both criteria.

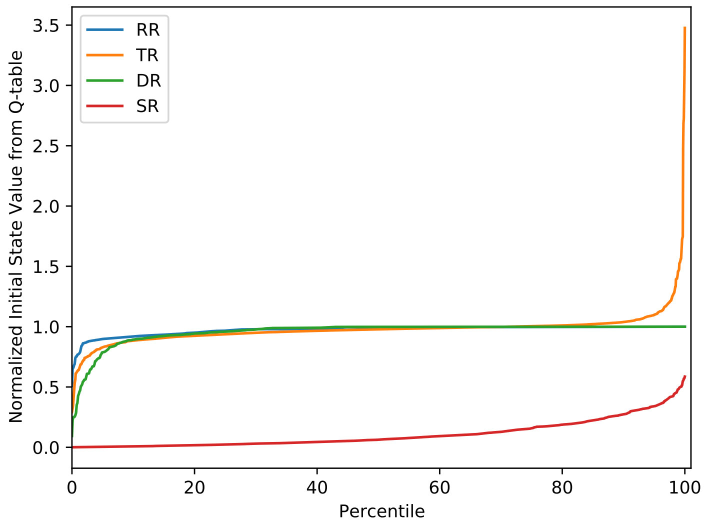

Fig. 3: Plot depicting inaccurate estimation of Q-values for TR and SR.

## 5.1 Inaccuracies in Estimation of Q-Values

In order to understand the performance of these reward schemes with Q-learning under our criteria, consider the example `riskReward`, which has two strategies: one leads to an accepting edge at every step with probability 0.9; the other leads to an accepting edge every other step with probability 1. This example shows that SR is not a faithful reward scheme [12] because the strategy that maximizes the probability of satisfaction does not maximize the SR reward. Since the strategy reported in Table 1 for SR is optimal for the satisfaction of the property, we know that Q-learning did not converge to a reward-maximizing strategy for the chosen values of the hyperparameters. From this, we can see that Criterion 1 may not rule out parameters that do not produce reward-maximizing strategies. We believe that this is because Criterion 1 selects based on Q-values, which may not be accurate estimates of the actual values obtained by optimal strategies.

In Figure 3, for each method we plot the values of the initial state from the Q-table of all grid points in increasing order. Each value in the figure was normalized such that an optimal reward maximization strategy for that method has a value of 1. Note that TR and SR do not have their values saturate like RR and DR. When we apply Criterion 1 to TR and SR, we remove all parameters that do not produce Q-values close to the top of the peaks that can be seen in the figure. As this filters out most of the runs, this offers an explanation for the longer running times of these methods. For SR, the Q-values do not converge. This is likely due to the fact that with a very large discount factor, SR needs more training than the other methods to converge to larger Q-values. For TR, we overestimate the value by up to about 3.5 times. We believe that this is due to Q-learning's tendency to overestimate the value of the optimal strategy.

As we discussed in Section 4, these methods rely on the contraction provided by discounting to be correct, and we have seen in Example 1 that in the undiscounted case the Q-value can go to infinity even in the absence of a strategy that wins with positive probability. While the contraction provided by the discount factor counters this effect, for a discount factor as high as the one chosen in the experiments the contraction is relatively weak. An extremely small learning rate would be needed to contain this effect. Additionally, the high variance reward of TR exacerbates the positive bias present in the estimator implemented by Q-learning. Double Q-learning [36] is a technique that mitigates this overestimation by using an estimator which is negatively biased. We believe that utilizing such techniques warrants further investigation.

In summary, Table 1 suggests that RR and DR perform similarly. The lower reward variance of DR explains why Criterion 1 selects optimal strategies for the satisfaction of the property more often than with RR. SR takes longer than the other methods under our criteria. On the other hand, TR is not well suited for Q-learning. While its denser reward may help to guide the learner, the inaccuracy in the Q-value estimates from this method negates this benefit. However, we do not know if these issues extend to other RL algorithms or if other algorithms may take better advantage of the denser reward of TR.

## 6 Discussion and Future Work

The three concerns to be addressed when applying model-free RL to $\omega$-regular properties are:

1. Finding the right automata representation.
2. Translating the acceptance condition into a faithful reward scheme.
3. Computing policies that maximize expected reward with an RL technique.

In [13], we addressed the first concern by introducing Good-for-MDP automata. This paper addresses the other two issues.

There have been two correct results for reinforcement learning of $\omega$-regular properties on MDPs. We have shown how to change the older of the two reward schemes, [12], to allow for earlier rewards in Section 3.2, and to obtain a biased discount scheme quite directly from there in Section 3.3. This biased discount scheme is significantly simpler than the scheme suggested in [3], which uses two entangled discount factors. Moreover, we have shown that the reward scheme from [3] can be viewed as the result of using the simple biased discount scheme from Section 3.3 embedded in a standard discounted Q-learning. We have therefore connected the known reward structures and provided simple intuitive explanations—in the form of a separation of concerns—for the reward structure used in [3]. Besides offering a simpler proof and new insights, it opens up an avenue of future work to see if other RL techniques will benefit from $\zeta$-biased discounted rewards without the common detour through discounting.

***Acknowledgment.*** This work utilized resources from the University of Colorado Boulder Research Computing Group, which is supported by the National Science Foundation (awards ACI-1532235 and ACI-1532236), the University of Colorado Boulder, and Colorado State University. This work was supported by the Engineering and Physical Sciences Research Council through grant EP/P020909/1 and by the National Science Foundation through grant 2009022.

## References

1. D. Andersson and Miltersen P. B. The complexity of solving stochastic games on graphs. In *Algorithms and Computation*, pages 112–121, 2009.
2. T. Babiak, F. Blahoudek, A. Duret-Lutz, J. Klein, J. Křetínský, D. Müller, D. Parker, and J. Strejček. The Hanoi omega-automata format. In *Computer Aided Verification*, pages 479–486, 2015. LNCS 9206.
3. A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. *CoRR*, abs/1909.07299, 2019.
4. G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *CoRR*, abs/1606.01540, 2016.
5. A. Camacho, R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith. LTL and beyond: Formal languages for reward function specification in reinforcement learning. In *Joint Conference on Artificial Intelligence*, pages 6065–6073, 2019.
6. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, July 1995.
7. cpphoafparser. `https://automata.tools/hoa/cpphoafparser`, 2016. Accesses: 2018-09-05.
8. G. De Giacomo and M. Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, pages 854–860, 2013.
9. K. Etessami, T. Wilke, and R. A. Schuller. Fair simulation relations, parity games, and state space reduction for Büchi automata. *SIAM J. Comput.*, 34(5):1159–1175, 2005.
10. J. Fu and U. Topcu. Probably approximately correct MDP learning and control with temporal logic constraints. In *Robotics: Science and Systems*, July 2014.
11. E. M. Hahn, G. Li, S. Schewe, A. Turrini, and L. Zhang. Lazy probabilistic model checking without determinisation. In *Concurrency Theory*, pages 354–367, 2015.
12. E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak. Omega-regular objectives in model-free reinforcement learning. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 395–412, 2019. LNCS 11427.
13. E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak. Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 306–323, 2020.
14. M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G. J. Pappas, and I. Lee. Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees. In *Conference on Decision and Control*, December 2019.
15. T. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. In *Concurrency Theory*, pages 273–287, 1997. LNCS 1243.
16. T. A. Henzinger and N. Piterman. Solving games without determinization. In *Computer Science Logic*, pages 394–409, September 2006. LNCS 4207.

17. A. Hordijk and A. A. Yushkevich. *Handbook of Markov Decision Processes: Methods and Applications*, chapter Blackwell Optimality, pages 231–267. Springer, 2002.
18. T. R. Icarte, T. Q. Klassen, R. A. Valenzano, and S. A. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *Conference on Machine Learning*, pages 2112–2121, July 2018.
19. Alex Irpan. Deep reinforcement learning doesn't work yet. `https://www.alexirpan.com/2018/02/14/rl-hard.html`, 2018.
20. J. Křetínský, G. A. Pérez, and J.-F. Raskin. Learning-based mean-payoff optimization in an unknown MDP under omega-regular constraints. In *CONCUR*, volume 118 of *LIPIcs*, pages 8:1–8:18, 2018.
21. M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, pages 585–591, July 2011. LNCS 6806.
22. A. Lavaei, F. Somenzi, S. Soudjani, A. Trivedi, and M. Zamani. Formal controller synthesis for unknown continuous-space MDPs via model-free reinforcement learning. In *International Conference on Cyber-Physical Systems*, April 2020.
23. M. E. Lewis. Bias optimality. In E. A. Feinberg and A. Shwartz, editors, *Handbook of Markov Decision Processes*, pages 89–111. Springer, 2002.
24. T. M. Liggett and S. A. Lippman. Short notes: Stochastic games with perfect information and time average payoff. *SIAM Review*, 11(4):604–607, 1969.
25. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems *Specification**. Springer, 1991.
26. R. Milner. An algebraic definition of simulation between programs. *Int. Joint Conf. on Artificial Intelligence*, pages 481–489, 1971.
27. A. Y. Ng, D. Harada, and S. J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, pages 278–287, 1999.
28. D. Perrin and J.-É. Pin. *Infinite Words: Automata, Semigroups, Logic and Games*. Elsevier, 2004.
29. M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York, NY, USA, 1994.
30. D. Sadigh, E. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia. A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications. In *CDC*, pages 1091–1096, December 2014.
31. S. Sickert, J. Esparza, S. Jaax, and J. Křetínský. Limit-deterministic Büchi automata for linear temporal logic. In *Computer Aided Verification*, pages 312–332, 2016. LNCS 9780.
32. F. Somenzi and A. Trivedi. Reinforcement learning and formal requirements. In *Numerical Software Verification - 12th International Workshop, NSV@CAV*, pages 26–41, July 2019. LNCS 11652.
33. A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman. PAC model-free reinforcement learning. In *International Conference on Machine Learning, ICML*, pages 881–888, 2006.
34. RMACC Summit Supercomputer. `https://rmacc.org/rmaccsummit`.
35. R. S. Sutton and A. G. Barto. *Reinforcement Learnging: An Introduction.* MIT Press, second edition, 2018.
36. H. van Hasselt. Double $Q$-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.
37. M. Y. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Foundations of Computer Science*, pages 327–338, 1985.
38. C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.