ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230368

Omega-Regular Reward Machines

Ernst Moritz Hahn[a], Mateo Perez[b], Sven Schewe[c], Fabio Somenzi[b], Ashutosh Trivedi[b;*] and Dominik Wojtczak[c]

[a]University of Twente, NL
[b]University of Colorado Boulder, USA
[c]University of Liverpool, UK
ORCiD ID: Ernst Moritz Hahn https://orcid.org/0000-0002-9348-7684,
Mateo Perez https://orcid.org/0000-0003-4220-3212, Sven Schewe https://orcid.org/0000-0002-9093-9518,
Fabio Somenzi https://orcid.org/0000-0002-2085-2003, Ashutosh Trivedi https://orcid.org/0000-0001-9346-0126,
Dominik Wojtczak https://orcid.org/0000-0001-5560-0546

Abstract. Reinforcement learning (RL) is a powerful approach for training agents to perform tasks, but designing an appropriate reward mechanism is critical to its success. However, in many cases, the complexity of the learning objectives goes beyond the capabilities of the Markovian assumption, necessitating a more sophisticated reward mechanism. Reward machines and ω-regular languages are two formalisms used to express non-Markovian rewards for quantitative and qualitative objectives, respectively. This paper introduces ω-regular reward machines, which integrate reward machines with ω-regular languages to enable an expressive and effective reward mechanism for RL. We present a model-free RL algorithm to compute ε-optimal strategies against ω-regular reward machines and evaluate the effectiveness of the proposed algorithm through experiments.

1 Introduction

Reinforcement learning (RL) [23] is a powerful learning-based synthesis paradigm that relies on providing rewards and punishment signals to reinforce or diminish behaviours. This is based on the principle that behaviours that are repeatedly rewarded tend to become habitual, while behaviours that are punished tend to diminish with experience. Therefore, translating high-level objectives into reward and punishment signals is critical to successful RL applications.

Simple objectives, such as cost-optimal reachability or safety, can be intuitively encoded into Markovian reward signals. However, more complex objectives require a stateful reward mechanism. Two formalisms that have been used to express non-Markovian reward signals are formal specifications (ω-regular languages and linear temporal logic)[21, 12, 5, 11] and reward machines (Mealy machines based monitors with scalar rewards as outputs)[18, 4]. The former is used to express long-run logical constraints, or "qualitative" specifications, while the latter is used for "quantitative" objectives, such as the discounted sum of rewards[1].

This paper argues for the need to optimise quantitative rewards under logical constraints over infinite horizons and proposes a model to conveniently express such learning objectives, which we refer to as *ω-regular reward machines*. These models integrate the two formalisms, allowing for the optimisation of quantitative rewards while also enforcing logical constraints over infinite horizons. The proposed ω-regular reward machines provide a powerful and efficient way to specify complex reward structures for RL, enabling the effective and efficient training of RL agents.

Reward Programming in RL. Formal specifications, such as linear temporal logic (LTL), ω-regular languages, and their generalisations [1], provide unambiguous and intuitive languages to express infinite-horizon requirements. However, manually designing rewards from higher-level specifications is tedious and error-prone. To address this challenge, researchers have proposed automatic translations from formal specifications to reward signals, providing a programmable, transparent, explainable, and trustworthy RL.

Sadigh et al. [21] initiated the study of model-free RL, where learning objectives were expressed in LTL. They used LTL to ω-automaton reduction [1] to design a scalar reward signal, with the hope that maximising the discounted objective maximises the probability of satisfaction of the LTL objective. However, the work of Hahn *et al.* [12] revealed challenges in translating formal specifications to reward machines, and proposed a correct translation from more general ω-automata based requirements to reward machines. Since then, several formally correct reward schemes [14, 13, 3, 20] have been proposed to automate ω-regular reward translation.

Icarte et al. [18, 19] advocated for the need of non-Markovian reward signals and popularised the use of Mealy machines to express such rewards. Reward machines provide an imperative language to program reward signals, allowing designers to better tune the reward logic by expressing their domain-specific expertise in the form of scalar rewards. However, we argue that—since reward machines encode a finite-horizon, albeit discounted, view of the environment—they fail to capture intuitive specifications and give rise to unintended and undesirable behaviours. To support this claim, we adapt the counterexample given by Hahn et al. [12] for the translation scheme of Sadigh et al. [21] to show how reward machines fail to capture intuitive specifications.

* Corresponding Author. Email: ashutosh.trivedi@colorado.edu.

[1] The terms *qualitative* and *quantitative* are used in this paper to differentiate between logic-based specifications over infinite behaviour and reward-based optimisation objectives. However, it's important to note that these terms can be misleading since maximising the probability of satisfying a logical property is technically a quantitative requirement. Similarly, finding a policy that provides a reward greater than a given budget can be considered a qualitative requirement.

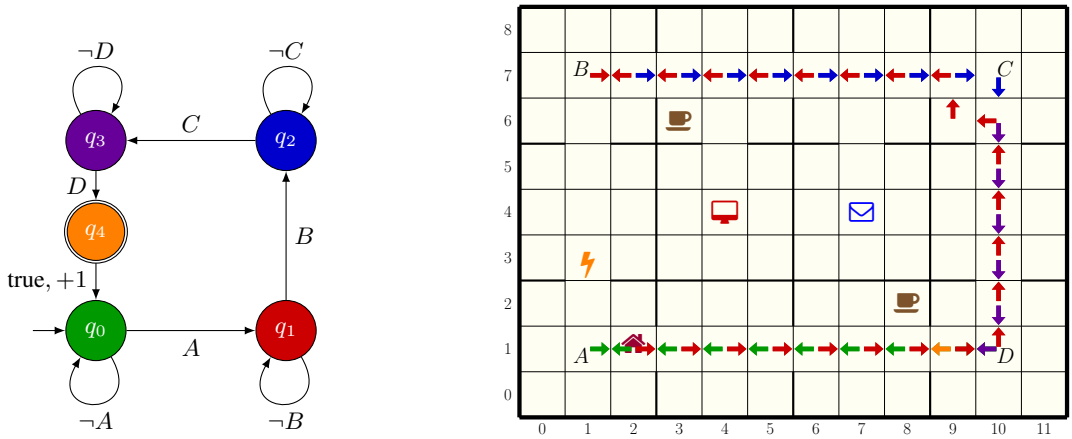

Figure 1. An ω-regular reward machine and the corresponding learned strategy in the office patrol example. The arrow colour corresponds to the ω-regular reward machine state colours. The strategy was learned with Q-learning, as described in Section 4. Rewards are zero unless otherwise specified.

Example 1. Counter-intuitive Office Grid-World. Figure 1 shows a grid-world example adapted from [19]. A robot patrols the four corner rooms of an office complex. However, unlike in [19], an electrical wire dangles from the ceiling along the path connecting rooms A and B, blocking safe passage. The robot can avoid the hazard by reaching B from A via D and C and then retracing its steps. Alternatively, it can try to fix the dangling wire but, in so doing, it may be damaged and put out of commission with probability $1/5$. If successful, the robot can then follow the shorter route that connects the corners in a simple cycle. Doling out reward every time the robot completes one round does not guarantee that the robot will follow the *safe* strategy that avoids the dangling wire. This is because the *risky* strategy, where the robot attempts to fix the dangling wire, incurs a risk that is offset by the reduction in path length, resulting in a higher expected discounted reward. The key problem is that maximising the expectation of the cumulative reward is different from maximising the probability that the reward is positive. We transform the reward machine into an ω-regular reward machine by marking the state q_4 as *accepting* (or, equivalently, mark its outgoing transitions as accepting). The overall objective is now modified so that visiting this accepting state infinitely with maximum probability takes precedence over the discounted reward. Using the technique described in this paper, we learn a strategy with Q-learning that satisfies the ω-regular reward machine and patrols while avoiding the hazard.

At the same time, ω-regular language-based RL is not sufficient to express simple quantitative preferences as shown next.

Example 2. Office Grid-World with Preference. In the environment of Figure 1, the robot may be tasked with picking up mail and coffee for the people who work in the "office" room. The robot may be free to collect the mail before getting the coffee, or vice versa but, *preferably* it should get mail first to prevent the coffee from getting cold. This kind of preference is naturally expressed as rewards on the transitions of the reward machine, while the satisfaction of the main objective (delivery of mail and coffee to the office) is guaranteed by imposing an ω-regular objective.

While there are some efforts to combine quantitative and qualitative formalism to express learning objective, they have been limited in expressing ω-regular languages [18, 19, 2]. To the best of our knowledge, there is no prior work that can handle general ω-regular objectives with discounted rewards. We propose ω-regular reward machines to fill this gap.

The ω-regular reward machines (ω-RMs) are defined as nondeterministic Büchi automata equipped with a scalar reward function. In RL, ω-RMs can act as interpreters that observe the sequence of actions taken by the learning agent and the corresponding sequence of observations from the environment and provide a sequence of scalar rewards. Unlike reward machines [19], ω-RMs may be nondeterministic, and the resolution of these choices is delegated to the RL agent. The goal of the RL agent is primarily to visit the accepting states infinitely often and then maximise the discounted sum of rewards. Besides the examples above, there are multiple scenarios that call for expressing such combinations in RL.

- **Specification gaming.** The term *specification gaming* refers to the behaviour of a learning agent that satisfies the literal specification, often in the terms of the reward signal, but not the intended one. While it is impossible to eliminate instances of specification gaming beforehand, detecting such behaviour can provide clues to some underspecified constraints.
  Although it is easy to explicitly express the constraints, designing reward signals that integrate such constraints can be challenging. For instance, consider the coastrunner example described in [8]. The environment provides a positive reward for target hitting, assuming that the agent naturally wants to finish the boat race as soon as possible. However, since this assumption is not backed by any explicit reinforcement, the learned behavior may not align with the desired one. One possible reward mechanism to address this issue is ω-regular reward machines that provide the discounted sum of rewards predicated upon the satisfaction of the ω-regular objective that the agent eventually terminates the boat race. Note that this requirement cannot be expressed directly as a reward machine or in formal logic.
- **Relative Preference over Accepting States.** Büchi automata [1] generalise finite automata to accept infinite behaviours that cause the accepting transitions to be visited infinitely often. ω-regular reward machines generalise Büchi automata by allowing the designer to express relative preference over various accepting states.
- **Repair Machines.** Another scenario where ω-regular reward machines can be useful is in repair machines. Suppose we have an RL problem where the learning objective is expressed as a Büchi automaton, and the RL agent can rewrite some of the observations of the environment before they are evaluated by the interpreter. The space of these repairs is given by a repair machine [9], which is defined as a weighted nondeterministic transducer, where the weight

corresponds to the cost of the rewrite action. In this case, the goal of the RL agent is to satisfy the objective given by the Büchi automaton while minimising a discounted sum of the costs associated with the repairs. The composition of the Büchi automaton-based specification and the repair machine can be expressed as an ω-regular reward machines. By leveraging ω-regular reward machines, we can ensure that the learning agent satisfies the intended specifications while optimising the cumulative reward and minimising the costs associated with the repairs.

- **Ulysses Contract.** Another application of ω-regular reward machines is in modelling Ulysses contracts. A Ulysses contract is a decision made by an agent to restrict potentially tempting but irrational choices by a future version of itself. This form of self-binding contract is named after the Greek hero Ulysses who, in the Odyssey, has his crew tie him to the mast to safely enjoy the sirens' song. With ω-regular RMs, the requirement to visit certain states infinitely often encodes the Ulysses contract, while the individual rewards encode various immediate rewards. This model allows the RL agent to maximise rewards without violating the specification. By using ω-regular reward machines to express Ulysses contracts, we can ensure that the learning agent follows a long-term plan of action that aligns with the desired objectives, even in the presence of potentially tempting but irrational choices.

Contributions. The paper provides an expressive framework for designing RL agents that can satisfy complex temporal specifications while optimising the cumulative reward. We introduce ω-regular reward machines, which can express complex objectives involving both quantitative and qualitative aspects. We then provide a convergent RL algorithm that approximates the optimal value for the ω-regular objective. In the case of a known model, we show the tractability of computing optimal value and ε-optimal policies. We also implement the proposed algorithm as an open-source tool and provide experimental results demonstrating its effectiveness.

Related Work. So far, we have cited several related works on reward machines [18, 19] and formal specifications [21, 17, 12, 14, 20, 3] in model-free reinforcement learning. There has been substantial work on lexicographic objectives in optimisation and RL, including lexicographic discounted objectives [22, 7, 6], lexicographic ω-regular objectives [15], and a combination of safety and discounted objectives [2]. However, to the best of our knowledge, this is the first work to consider the general class of ω-regular objectives with discounted rewards in model-free RL.

Organisation. We begin by providing a brief overview of Markov decision processes (MDPs) and ω-regular specifications. In Section 3, we introduce our reward mechanism, ω-regular reward machines, and provide details on how they can be used to express complex objectives involving both quantitative and qualitative aspects. We then discuss results of probabilistic model checking and reinforcement learning using ω-regular reward machines. In Section 4, we present experimental results that demonstrate the effectiveness of the proposed approach, followed by some concluding remarks.

2 Preliminaries

An alphabet Σ is a finite set of letters. A finite string (resp. ω-string) over Σ is defined as a finite sequence (resp. an infinite ω-sequence) of letters from Σ. We denote the empty string by ε. We write Σ^* and Σ^ω for the set of finite and ω-strings over Σ. A language (resp. ω-language) L over an alphabet Σ is defined as a set of finite strings (resp. ω-strings).

2.1 Markov Decision Processes

Let $\mathcal{D}(S)$ denote the set of all discrete distributions over S. A Markov decision process (MDP) $\mathcal{M}$ is a tuple (S, s_0, A, T, AP, L), where S is a finite set of states, $s_0 \in S$ is the initial state, A is a finite set of *actions*, $T\colon S \times A \to \mathcal{D}(S)$ is the probabilistic transition function, AP is the set of *atomic propositions* (observations), and $L\colon S \to 2^{AP}$ is the *labelling function*.

For any state $s \in S$, we let $A(s)$ denote the set of actions that can be selected in state s. An MDP is a Markov chain if $A(s)$ is singleton for all $s \in S$. For states $s, s' \in S$ and $a \in A(s)$, $T(s,a)(s')$ equals $\Pr(s'|s,a)$. 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 $\Pr(s_{i+1}|s_i, a_{i+1})>0$ for all $i \geq 0$. A finite run is a finite such sequence. For a *run* $r = \langle s_0, a_1, s_1, \ldots\rangle$ we define the corresponding labelled 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 the 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. A memory skeleton is a tuple $M = (M, m_0, \alpha_u)$ where M is a finite set of memory states, m_0 is the initial state, and $\alpha_u : M \times \Sigma \to M$ is the memory update function. We define the extended memory update function $\hat{\alpha}_u : M \times \Sigma^* \to M$ in a straightforward way. A finite memory strategy for $\mathcal{M}$ over a memory skeleton M is a Mealy machine (M, α_x) where $\alpha_x : S \times M \to \mathcal{D}(A)$ is the *next action function* that suggests the next action based on the MDP and memory state. The semantics of a finite memory strategy (M, α_x) is given as a strategy $\sigma : FRuns \to \mathcal{D}(A)$ such that for every $r \in FRuns$ we have that $\sigma(r) = \alpha_x(last(r), \hat{\alpha}_u(m_0, L(r)))$.

A strategy σ is *pure* if $\sigma(r)$ is a point distribution for all runs $r \in FRuns^{\mathcal{M}}$ and is *mixed* (short for strictly mixed) if $supp(\sigma(r)) = A(last(r))$ for all runs $r \in FRuns^{\mathcal{M}}$. Let $Runs^{\mathcal{M}}_\sigma(s)$ 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. We say that σ is *stationary* if $last(r) = last(r')$ implies $\sigma(r) = \sigma(r')$ for all finite runs $r, r' \in FRuns^{\mathcal{M}}$. A stationary strategy can be given as a function $\sigma : S \to \mathcal{D}(A)$. A strategy is *positional* if it is both pure and stationary.

An MDP $\mathcal{M}$ under a strategy σ results in a Markov chain $\mathcal{M}_\sigma$. If σ is a finite memory strategy, then $\mathcal{M}_\sigma$ is a finite-state Markov chain. The behaviour of an MDP $\mathcal{M}$ under a strategy σ and starting state $s \in S$ is defined on a probability space

$$(Runs^{\mathcal{M}}_\sigma(s), \mathcal{F}_{Runs^{\mathcal{M}}_\sigma(s)}, \Pr^{\mathcal{M}}_\sigma(s))$$

over the set of infinite runs of σ with starting state s. Given a random variable $f\colon Runs^{\mathcal{M}} \to \mathbb{R}$, we denote by $\mathbb{E}^{\mathcal{M}}_\sigma(s)\,\{f\}$ the expectation of f over the runs of $\mathcal{M}$ originating at s that follow σ.

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'. Moreover $\mathcal{M}'$ is closed under probabilistic transitions. An *end-component* [10] of an MDP $\mathcal{M}$ is a sub-MDP $\mathcal{M}'$ such that for every state pair $s, s' \in S'$ there is a strategy that can reach s' from s with positive probability. A maximal end-component is an end-component that is maximal under set-inclusion. Every state s of an MDP $\mathcal{M}$ belongs to at most one maximal end-component.

2.2 Discounted Reward Objectives

The learning objective over MDPs in RL is often typically expressed using a Markovian reward function, i.e. a function $\rho\colon S \times$

$A \times S \to \mathbb{R}$ assigning utility to transitions. A *rewardful* MDP is a tuple $\mathcal{M} = (S, s_0, A, T, \rho)$ where S, s_0, A, and T are defined in a similar way as for MDP, and ρ is a Markovian reward function. A rewardful MDP $\mathcal{M}$ under a strategy σ determines a sequence of random rewards $\rho(X_{i-1}, Y_i, X_i)_{i\geq 1}$, where X_i and Y_i are the random variables denoting the i-th state and action, respectively. For $\lambda \in [0,1[$, the *discounted reward* $\mathsf{Disct}(\lambda)^{\mathcal{M}}_{\sigma}(s)$ is defined as $\lim_{N\to\infty} \mathbb{E}^{\mathcal{M}}_{\sigma}(s)\Big\{\sum_{1\leq i\leq N} \lambda^{i-1}\rho(X_{i-1}, Y_i, X_i)\Big\}$. We define the optimal discounted reward $\mathsf{Disct}^{\mathcal{M}}_{*}(s)$ for a state $s \in S$ as $\mathsf{Disct}^{\mathcal{M}}_{*}(s) \stackrel{\text{def}}{=} \sup_{\sigma\in\Sigma_{\mathcal{M}}} \mathsf{Disct}^{\mathcal{M}}_{\sigma}(s)$. A strategy σ is discount-optimal if $\mathsf{Disct}^{\mathcal{M}}_{\sigma}(s) = \mathsf{Disct}^{\mathcal{M}}_{*}(s)$ for all $s \in S$.

Often, complex learning objectives cannot be expressed using Markovian reward signals. A recent trend is to express learning objectives using finite-state reward machines [19]. A (nondeterministic) reward machine is a tuple $\mathcal{R} = (\Sigma, U, u_0, \delta, \rho)$ where U is a finite set of states, $u_0 \in U$ is the starting state, $\delta\colon U\times\Sigma \to 2^U$ is the transition relation, and $\rho\colon U \times \Sigma \times U \to \mathbb{R}$ is the reward function.

Given an MDP $\mathcal{M} = (S, s_0, A, T, AP, L)$ and a reward machine $\mathcal{R} = (2^{AP}, U, u_0, \delta, \rho)$, their product $\mathcal{M}\times\mathcal{R} = (S\times U, (s_0, u_0), (A\times U), T^{\times}, \rho^{\times})$ is a rewardful MDP where $T^{\times}\colon (S\times U) \times (A\times U) \to \mathcal{D}(S\times U)$ is such that

$$((s,u),(a,u'))((s',u')) \mapsto \begin{cases} T(s,a)(s') & \text{if } u' \in \delta(u, L(s)) \\ 0 & \text{otherwise.} \end{cases}$$

and $\rho^{\times}\colon (S\times U) \times (A\times U) \times (S\times U) \to \mathbb{R}$ is defined such that $\rho^{\times}((s,u),(a,u'),(s',u'))$ equals $\rho(u, L(s), u')$ if $(u, L(s), u') \in \delta$. For discounted reward objective, the optimal strategy of $\mathcal{M}\times\mathcal{R}$ are positional on $\mathcal{M}\times\mathcal{R}$. Moreover, these positional strategies characterise a finite memory strategy (with memory skeleton based on the states of $\mathcal{R}$ and the next-action function based on the positional strategy) over $\mathcal{M}$ maximising the learning objective given by $\mathcal{R}$.

In our reductions, we make use of total reward objective $\mathsf{ETotal}^{\mathcal{M}}_{*}(s)$ defined in a similar fashion as the discounted objective when the discount factor λ is equal to 1. The concepts of expected total reward and optimal strategy is defined in an analogous manner.

2.3 Omega-Regular Specifications

A *Büchi automaton* is a tuple $\mathcal{A} = (\Sigma, Q, q_0, \delta, F)$, 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 $F \subseteq Q \times \Sigma \times Q$ is the set of *accepting transitions*.

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 $(Q \times \Sigma)^{\omega}$ 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 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 $\mathsf{inf}(r) \cap F \neq \emptyset$. The *language* $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$ 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 a Büchi automaton.

Given an MDP $\mathcal{M} = (S, s_0, A, T, AP, L)$ and a Büchi automaton $\mathcal{A} = (2^{AP}, Q, q_0, \delta, F)$, their *product* $\mathcal{M} \times \mathcal{A} = (S\times Q, (s_0, q_0), A\times Q, T^{\times}, F^{\times})$ is an MDP with accepting transitions $F^{\times}$ where $T^{\times}\colon (S\times Q) \times (A\times Q) \to \mathcal{D}(S \times Q)$ is such that

$$((s,q),(a,q'))((s',q')) \mapsto \begin{cases} T(s,a)(s') & \text{if } (q, L(s,a,s'), q') \in \delta \\ 0 & \text{otherwise.} \end{cases}$$

The set of accepting transitions $F^{\times} \subseteq (S \times Q) \times (A \times Q) \times (S \times Q)$ is defined by $((s,q),(a,q'),(s',q')) \in F^{\times}$ if, and only if, $(q, L(s,a,s'), q') \in F$ and $T(s,a)(s') > 0$.

A strategy σ on the product defines a strategy σ' on the MDP with the same value, and vice versa. Note that for a stationary σ on the product, the strategy σ on the MDP may need memory. End-components and runs of the product MDP are defined just like for MDPs. An accepting end-component is an end-component that contains an accepting transition. A run of $\mathcal{M}\times\mathcal{A}$ is accepting if $\inf(r) \cap F^{\times} \neq \emptyset$. We define the Büchi satisfaction probability $\mathsf{BSat}_{\sigma}(s)$ of a strategy σ as the probability of this strategy generating an accepting run, i.e.

$$\Pr^{\mathcal{M}\times\mathcal{A}}_{\sigma}\big\{r \in \mathit{Runs}^{\mathcal{M}\times\mathcal{A}}_{\sigma}(s, q_0) : \inf(r) \cap F^{\times} \neq \emptyset\big\}.$$

Similarly, $\mathsf{BSat}(s)$ is the optimal satisfaction probability over the product, i.e. $\mathsf{BSat}(s) = \sup_{\sigma} \mathsf{BSat}_{\sigma}(s, q_0)$. We say that a strategy σ_* is *Büchi-optimal* from $s \in S$ if $\mathsf{BSat}(s) = \mathsf{BSat}_{\sigma_*}(s, q_0)$.

3 Omega-Regular Reward Machines

Our definition of ω-regular reward machine integrates the definitions of reward machines and Büchi automata. The notion of product with an MDP is defined in a similar fashion. The optimisation objective is to compute optimal discounted reward over all Büchi-optimal strategies and near-optimal strategies achieving this reward.

Definition 1 (ω-Regular Reward Machine (ω-RM).)**.** *An ω-RM is a tuple $\mathcal{R} = (\Sigma, U, u_0, \delta, \rho, F)$ where U is a finite set of states, $u_0 \in U$ is the starting state, $\delta\colon U \times \Sigma \to 2^U$ is the transition relation, $\rho\colon U \times \Sigma \times U \to \mathbb{R}$ is the reward function, and $F \subseteq U \times \Sigma \times U$ is the set of* accepting transitions.

Given an MDP $\mathcal{M} = (S, s_0, A, T, AP, L)$ and an automaton $\mathcal{R} = (2^{AP}, U, u_0, \delta, \rho, F)$, their *product* $\mathcal{P} = \mathcal{M} \times \mathcal{R} = (S\times U, (s_0, u_0), A\times U, T^{\times}, \rho^{\times}, F^{\times})$ is an MDP with initial state (s_0, u_0) and accepting transitions $F^{\times}$ where $T^{\times}\colon (S\times U) \times ((A\times U)) \to \mathcal{D}(S\times U)$ is such that:

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

The set of accepting transitions $F^{\times} \subseteq (S \times U) \times (A \times U) \times (S \times U)$ is defined by $((s,q),(a,q'),(s',q')) \in F^{\times}$ if, and only if, $(q, L(s), q') \in F$ and $T(s,a)(s') > 0$.

Let us fix the product MDP $\mathcal{P} = \mathcal{M} \times \mathcal{R}$ as the tuple (Q, q_0, A, T, ρ, F) for the rest of this section. To define the optimisation objective for ω-RM, we need to define the following concepts over the product MDP $\mathcal{P}$.

- The *Büchi satisfaction probability* $\mathsf{BSat}_{\sigma}(s)$ is the probability to satisfy the Büchi objective by a strategy σ from a given state s and is defined similar to that for Büchi automata (Section 2.3). The *optimal satisfaction probability* $\mathsf{BSat}(s)$ and a Büchi-optimal strategy σ^* that achieves these, i.e., $\mathsf{BSat}(s_0) = \mathsf{BSat}_{\sigma^*}(s_0)$ is defined similar to that for Büchi automata (Section 2.3).
- The *optimal Büchi-discounted value* Bval is the optimal discounted reward among Büchi optimal strategies defined as:

$$\mathsf{Bval} : q \mapsto \sup_{\sigma} \big\{\mathsf{value}_{\sigma}(q) \mid \mathsf{BSat}(q) = \mathsf{BSat}_{\sigma}(q)\big\}$$

  where $\mathsf{value}_{\sigma} : q \mapsto \mathsf{Disct}(\lambda)^{\mathcal{M}}_{\sigma}(q)$ is the discounted value of σ.
- An *optimal Büchi-discounted strategy* is a strategy σ that attains optimal Büchi-discounted value, i.e., $\mathsf{BSat}(q_0) = \mathsf{BSat}_{\sigma}(q_0)$ and, for a given $\varepsilon > 0$ an ε-optimal (near optimal) Büchi-discounted strategy is such that $\mathsf{Bval}_{\sigma}(q_0) > \mathsf{value}_{\sigma}(q_0) - \varepsilon$.

We seek near-optimal strategies that maximise the chance of satisfying the Büchi objective, but will only be arbitrarily close to satisfying the discounted reward objective due to the following observation.

Lemma 1. *The optimal Büchi-discounted strategies may not exist.*

Proof. Consider an MDP where one can freely choose the next letter from an alphabet $\{a, b\}$ and have a reward of 1 for a and 0 for b, as well as a primary Büchi objective to see infinitely many b's, then we cannot achieve an expected reward of $\frac{1}{1-\lambda}$ while satisfying the Büchi objective. We can, however, get arbitrarily close, e.g., by producing a's until a reward $> \frac{1}{1-\lambda} - \varepsilon$ is collected for any given $\varepsilon > 0$, and henceforth produce b's. While the optimal Büchi-discounted value is $\frac{1}{1-\lambda}$, no (finite or infinite memory) strategy can attain it. □

3.1 Known MDP: Probabilistic Model Checking

Consider the problem to compute the optimal Büchi-discounted value and near-optimal strategies when $\mathcal{M}$ and $\mathcal{R}$, and therefore their product $\mathcal{P}$, are known. A possible first step is to model check the MDP $\mathcal{P}$ against the Büchi objective. This provides the probability of achieving the Büchi objective from every state together with a positional strategy σ^* of how to achieve it.

Model checking the product MDP $\mathcal{P} = (Q, q_0, A, T, \rho, F)$ is a standard operation [1, 10]. One would typically start with qualitative model checking, which consists of two intertwined procedures:

1. Remove all states in Q from which no accepting transition is reachable with positive probability. If any states were removed, go to step 2.
2. Recursively remove state-action pairs (q, a) where $T(q, a)(q') > 0$ for any state q' that has been previously removed, and remove states q such that all of its state-action pairs (q, a) have already been removed. If any state was removed at the end of this procedure, go back to step 1.

Both steps work in time linear in the transition graph of $\mathcal{P}$, and a fixed point is reached in at most $|Q|$ steps, because a new procedure call is made only if at least one state was removed. The remaining states, $Q_1 \subseteq Q$, are those, for which we can satisfy the Büchi objective almost surely, and the last application of (1) provides such a strategy.

To extend this method to quantitative model checking (computing the optimal probability of satisfying the Büchi objective), we can simply add, for all states q removed during the procedure above (whose set will be denoted by $Q_{<1} = Q \setminus Q_1$) and all actions a, variables $p_{(q,a)}$ and p_q that represent the probability to win when taking the state-action pair (q, a) and when starting at q, respectively. To calculate the correct probabilities, we define the following linear program that these probabilities have to satisfy.

For all $q \in Q_{<1}$ and $a \in A$:

$$p_{(q,a)} = \sum_{q' \in Q_{<1}} T(q, a)(q') p_q + \sum_{q' \in Q_1} T(q, a)(q')$$
$$p_q \geq 0 \qquad \text{and} \qquad p_q \geq p_{(q,a)}$$

with the objective to *Minimise* $\sum_{q \in Q_{<1}} p_q$.

Note that we can achieve p_q value when starting at q by playing a safe strategy that at any state, q', only uses an action, a, such that $p_{q'} = p_{(q',a)}$ and such that there is a positive probability to visit an accepting transition as done in (1).

Compute Near-Optimal Strategies. To find a strategy that is near-optimal with respect to the discounted reward without sacrificing the probability to satisfy the Büchi objective we do the following. First, we remove all state-action pairs (q, a) such that $p_q \neq p_{(q,a)}$ from $\mathcal{P}$, because taking any such action would reduce the probability of generating an accepting run. Next, we remove all states that have no actions left. Note that, in the example discussed in Lemma 1, we would neither remove any states nor any state-action pairs.

Now, for the remaining states, $q \in Q'$, and state-actions pairs, (q, a), we introduce variables ρ_q and $\rho_{(q,a)}$, respectively. We find the optimal discounted reward values and a positional strategy τ that realises them using the following linear program.

For all $q \in Q'$ and $a \in A(q)$:

$$\rho_q \geq \rho_{(q,a)}$$
$$\rho_{(q,a)} = \sum_{q' \in Q'} T(q, a)(q') \big(\rho(q, a, q') + \lambda \rho_{q'}\big) ,$$

with the objective to *Minimize* $\sum_{q \in Q'} \rho_q$.

In the example shown in Lemma 1, this would be to always produce a's—which would not satisfy the Büchi objective. However, any strategy obtained this way can either be followed long enough that the value of the tail is marginal, or we can initially follow this strategy, and switch to a strategy σ^* that pursues the Büchi objective with a small probability in every step. Both approaches will lead to a strategy that maximises the probability to satisfy the Büchi objective (in our example, with probability 1) while providing an expected payoff that is near-optimal among the strategies that maximise satisfying the Büchi objective.

3.2 Reinforcement Learning: Reward Translation

When the MDP is not known, we use model-free RL to approximate optimal value and learn a near optimal strategy. We present a reduction from the product MDP $\mathcal{P} = (Q, q_0, A, T, \rho, F)$ to a related MDP $\mathcal{P}_\lambda$ such that optimal values from $\mathcal{P}_\lambda$ can be used to compute the optimal value in $\mathcal{P}$.

Definition 2 (Reward Translation)**.** *For a product MDP $\mathcal{P} = (Q, q_0, A, T, \rho, F)$ with discount factor λ, consider a related MDP $\mathcal{P}_\lambda = (Q', (q_0, 0), A, T', \rho, F')$, where:*

- $Q' = (Q \times \{0, 1\}) \cup \{t\}$ *is the state space, where t is a fresh trap state including the initial state $(q_0, 0)$,*
- $T' \colon Q' \times A \times Q' \to [0, 1]$ *is the transition function where*

$$T'((q, 0), a, (q', 0)) = (1 - \lambda) T(q, a, q')$$
$$T'((q, 0), a, (q', 1)) = \lambda T(q, a, q')$$
$$T'((q, 1), a, (q', 1)) = \begin{cases} \zeta T(q, a, q') & \text{if } (q, a, q') \in F \\ T(q, a, q') & \text{otherwise} \end{cases}$$
$$T'((q, 1), a, t) = (1 - \zeta) \sum_{\{q' \mid (q,a,q') \in F\}} T(q, a, q')$$
$$T'(t, \sigma, t) = 1$$

where $\zeta \in (0, 1)$ is a parameter,

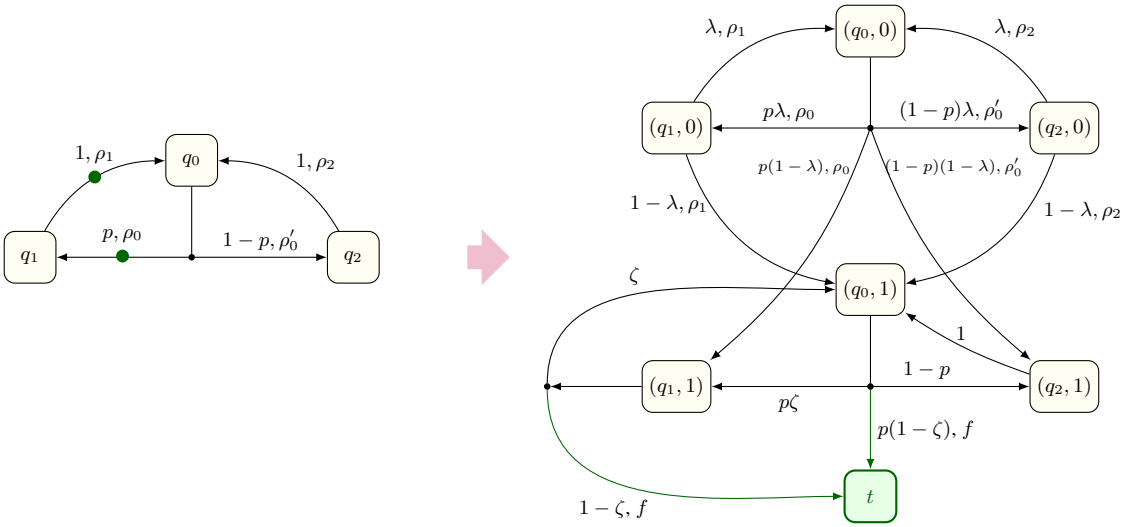

Figure 2. A translation of a product MDP with a combined Büchi-discounted reward objective with a discount factor λ on the left hand side to an MDP with total reward objective on the right hand side. Each transition is labelled with its probability followed by its reward. Rewards with value 0 are omitted. Intuitively, the new MDP consists of two phases: the maximisation of the discounted reward (states labelled $(q_i, 0)$) followed by maximisation of the Büchi objective (states labelled $(q_i, 1)$). We move from the first phase to the second with probability $1 - \lambda$ in each step. Any original accepting transition is marked with a green dot and is replaced in the new MDP with a transition that leads with probability $p(1 - \zeta)$ to the new trap state t and with probability $p\zeta$ to the original target state, where p is the original transition probability.

- *and* $\rho' \colon Q' \times A \times Q' \to \mathbb{R}$ *is the reward function where*

$$\begin{aligned} \rho'((q,0),a,(q',0)) &= \rho(q,a,q') \\ \rho'((q,0),a,(q',1)) &= \rho(q,a,q') \\ \rho'((q,1),a,(q',1)) &= 0 \\ \rho'((q,1),a,t) &= f \\ \rho'(t,a,t) &= 0 \end{aligned}$$

  where $f \geq 1$ *is a (usually large) parameter.*

An example of this translation is shown in Figure 2. Note that while the MDP $\mathcal{P}_\lambda$ has two additional parameters, f and ζ, it only has numerical rewards and is therefore simpler to analyse.

The parameter ζ has a similar meaning as in [12]: with probability $1 - \zeta$ one declares that accepting edges will be seen forever after seeing a single accepting edge. Increasing the value of ζ means that the agent needs to see a larger number of accepting edges to obtain the same probability of declaring that accepting edges will be seen infinitely often. As in [12], if the Büchi objective is satisfied with probability 1 any $0 < \zeta < 1$ is correct in our reduction. The parameter f is used to weight the value of the Büchi objective relative to the discounted reward objective. As we will show, for large enough f, the two objectives are ordered lexicographically. For $\mathcal{P}_\lambda$, we want to learn the optimal expected reward. For this, we define

- $\mathsf{value}'(r) = \sum_{i=0}^{\infty} \rho'(r_i, a_i, r_{i+1})$ for any run r and,
- $\mathsf{value}'_\sigma \colon q \mapsto \mathbb{E}\big\{\mathsf{value}'(r) \mid r \in \mathit{Runs}^\sigma_q(\mathcal{P}_\lambda)\big\}$, and
- $\mathsf{value}' \colon q \mapsto \sup_\sigma \mathsf{value}'_\sigma(q)$, as well as a strategy σ with $\mathsf{value}'_\sigma(q_0) = \mathsf{value}'(q_0)$.

We say that a positional strategy σ is ε-consistent with value function value' on a set S if $|\mathsf{value}'_\sigma(q) - \mathsf{value}'(q)| < \varepsilon$ for all $q \in S$.

Let us start with an established learning approach for Büchi objectives, which is a special case for the approach from [12].

Theorem 1 (Limit Reachability [12])**.** *For a product MDP* $\mathcal{P}$ *and a given parameter* $f \geq 1$*, there exists* $\varepsilon > 0$ *and* $\zeta^* \in (0,1)$ *such that, for all* $\zeta \in (\zeta^*, 1)$ *and all* $q \in Q$*,*

$$\mathsf{value}'\big((q,1)\big) \in [f \cdot \mathsf{BSat}(q), f \cdot \mathsf{BSat}(q) + \varepsilon],$$

and all positional strategies that are ε*-consistent with* value' *from* $Q \times 1$ *are optimal strategies for the Büchi objective.*

Theorem 2. *For a given product MDP* $\mathcal{P}$ *and* $\varepsilon \in (0,1)$*, there is a parameter* $f^* \geq 1$ *such that, for all* $f \geq f^*$*, there is a* $\zeta^* \in (0,1)$ *such that, for all* $\zeta \in (\zeta^*, 1)$*,*

$$\mathsf{value}'\big((q,0)\big) \in [\mathsf{Bval}(q) + f \cdot \mathsf{BSat}(q), \mathsf{Bval}(q) + f \cdot \mathsf{BSat}(q) + \varepsilon].$$

Proof. We start with a number of simple observations.

1. For every set of parameters, $\mathcal{P}_\lambda$ is a total payoff MDP and therefore has positional optimal strategies.
2. For every strategy that provides the optimal probability $\mathsf{BSat}(q_0)$ to satisfy the Büchi objective, the following holds: for all $k \in \mathbb{N}$, under the assumption that the MDP moves to the 1-copy after k steps, the expected chance of satisfying the Büchi objective is $\mathsf{BSat}(q_0)$. Moreover, this value is $\leq \mathsf{BSat}(q_0)$ for every strategy, and, for some $k' \leq |Q|$ and all $k \geq k'$, it is $< \mathsf{BSat}(q_0)$ for a non-optimal positional strategy. Thus, for any positional non-optimal strategy, there is a probability $p < \mathsf{BSat}(q_0)$ such that the chance of meeting the Büchi objective is $\leq p$. Let p^* be the maximal such value. (Note that the set of positional strategies is finite.)
3. Let d be the maximal difference in expected discounted payoff between two strategies. (d can be bounded by maximum value of ρ divided by $1 - \lambda$). Then we choose $f^* \geq \frac{d}{\gamma(\mathsf{BSat}(q_0) - p^*)} + 1$.

With these observations, we choose f^* as in (3) and, for $f \geq f^*$, ζ^* as in Theorem 1. This provides correct strategies for obtaining the Büchi objective with maximal probability from all positions in the 1-copy, and values that approximate $f \cdot \mathsf{BSat}\big((q,1)\big)$ with ε precision.

Let us now consider two positional strategies for $\mathcal{R}_\gamma$: σ, which maximises the chance of satisfying the Büchi objective, and τ, which

Name	states	time (s)	f	ζ	γ	α	ϵ	init	ep-l	ep-n
cheapest	6	0.56					0.50			3k
promises	16	0.71					0.50			3k
robot 4x4	64	0.30					0.20			1k
virus	859	61.28		0.95					50	150k
busyRing2	169	45.38				0.005	0.75			175k
twoWECs	6	4.80	500			0.005	0.20			30k
officeZapPatrol	876	165.05		0.5				7	100	350k
officePreferences	1248	77.27	100	0		0.002		100	150	200k
officeZapPreferences	1254	111.01	100	0		0.005	0.01	100	1500	250k

Table 1. Q-learning results. Blank entries indicate that default parameters are used. The default parameters are $f = 10$, $\zeta = 0.99$, $\gamma = 0.999$, $\alpha = 0.01$, $\epsilon = 0.1$, init $= 0$, ep-l $= 20$, and ep-n $= 20k$. The same external discount factor of $\lambda = 0.99$ was used for all experiments.

does not. By (3) and Theorem 1, this implies that the expected reward of σ is at least $1 - \varepsilon$ better than the expected reward of τ. Thus, a positional strategy with optimal reward will also maximize the chance of satisfying the Büchi objective.

Let us now consider two positional strategies for $\mathcal{P}_\lambda$ that are both ε-consistent with value on $Q \times \{1\}$ (and thus optimal w.r.t. the Büchi objective there), σ and τ, such that the expected reward in the 0-copy of σ is at least ε higher than that of τ. Then, using Theorem 1, the expected reward of σ in $\mathcal{P}_\lambda$ is higher than the reward by τ.

Putting these two together, we get that the optimal solution for $\mathcal{R}_\gamma$ provides that, for all $k \in \mathbb{N}$, the probability of satisfying the Büchi objective after k steps is $\mathsf{BSat}(q_0)$ while, among the strategies with this property, the expected discounted reward is ε-optimal. □

Learning. For given parameters, $\mathcal{P}_\lambda$ is simply an MDP with total rewards *and* contraction on $Q \times \{0\}$ and a reachability objective in $Q \times \{1\} \cup \{t\}$. The values and strategies to obtain them can be learned with standard techniques, such as Q-learning. Note that the parameter ε can be selected after learning is complete since the strategy for $\mathcal{P}_\lambda$ is independent of ε.

Theorem 3. *Given $\mathcal{P}_\lambda$ and parameters f and ζ, we can use Q-learning to find value' and an optimal strategy σ.*

These three theorems provide us with a way to robustly infer a near-optimal strategy for $\mathcal{M}$ from $\mathcal{R}$ for appropriate parameters f and ζ: To obtain a 2ε-optimal strategy for $\mathcal{M}$, we can simply find or approximate an optimal strategy for $\mathcal{R}$. To transfer this strategy to $\mathcal{M}$, we can follow the 0-copy ($Q \times \{0\}$) long enough; say k steps, so that the contribution of all but the first k transitions is smaller than ε. We can then move on to follow the strategy for the 1-copy ($Q \times \{1\}$).

Corollary 1. *For a product MDP $\mathcal{P}$, a $\varepsilon > 0$ and appropriate parameter $f \geq 1$ and $\zeta \in (0, 1)$, we can infer a near optimal strategy for $\mathcal{P}$ from a near optimal strategy for $\mathcal{P}_\lambda$ with parameters f and ζ.*

4 Experimental Results

We implemented the construction described in Section 3 in the tool MUNGOJERRIE [16]. The construction is implemented on-the-fly, where the states of the MDP and the ω-regular reward machine are kept independent and concatenated at each time step. We ran tabular Q-learning on multiple case studies, as seen in Table 1. Table 1 shows the name of the case study, the number of states in the product MDP, the time taken for learning in seconds, the value of f, ζ, and algorithm discount factor γ. The table also shows the learning rate α, the ϵ-greedy exploration rate ϵ, the value the Q-table was initialised to, the episode reset length, and the number of training episodes. The episode reset length is the number of time steps between accepting edges in the second layer that needs to be exceeded for the episode to be reset. After learning is complete, we verify that the values of the learned strategy matches the values computed by the linear programs described in Section 3. This ensures that the learned strategy can be transformed into an ε-optimal strategy for any $\varepsilon > 0$. For all of our experiments, we use the same external discount factor $\lambda = 0.99$.

Example *cheapest* shows how quantitative rewards may supplement an ω-regular specification to steer the agent toward a path of minimum *cost* rather than a path of minimum *length*. Example *promises* illustrates the use of quantitative rewards to model advances that an agent may get in return for a promise to fulfil an obligation encoded as an ω-regular objective. Examples *robot 4x4* and *virus* were used in [15] in the context of multiple ω-regular properties lexicographically combined. Their use here demonstrates the wider set of specifications allowed by ω-regular reward machines. In *busyRing2*, quantitative rewards are used to measure the fraction of time spent in negotiating the asynchronous arbitration protocol. In *twoWECs*, quantitative rewards indicate preference between sets of states that satisfy the ω-regular specification. The last three examples are based on the office grid-world discussed in the introduction. Since these examples require deep exploration in order to discover the near-optimal strategy, we initialised the Q-table optimistically to aid exploration.

Overall, our experimental results demonstrate the effectiveness and versatility of ω-regular reward machines in solving a wide range of RL problems with complex specifications. By leveraging the power of ω-regular objectives and discounted rewards, we can specify and learn complex behaviours and preferences that would be difficult to express with traditional RL techniques. Moreover, our open-source implementation may allow researchers and practitioners to easily apply these techniques to their own problems.

5 Conclusion

Reward machines and formal specifications are two leading reward programming languages that roughly correspond to imperative (how to give rewards?) and declarative (what to give rewards for?) ways of expressing programmer's intent. This paper presents a hybrid approach that combines the declarative and imperative specifications. The ω-regular RMs can be constructed by expressing declarative specification in LTL and the imperative reward specification as reward machines. The ω-RMs strictly generalise both reward machines (from finite-horizon behaviour to infinite-horizon) and ω-regular languages (from qualitative to quantitative satisfaction) in a natural framework. We presented a parametric reduction from the optimisation problem over ω-RM to an optimal reward reachability problem that can be constructed and learned in a model-free manner.

Acknowledgements

This work was supported in part by the EPSRC through grants EP/X017796/1 and EP/X03688X/1, the NSF through grant CCF-2009022 and the NSF CAREER award CCF-2146563; and the EU's Horizon 2020 research and innovation programme under grant agreements No 864075 (CAESAR).

References

[1] Christel Baier and Joost-Pieter Katoen, *Principles of Model Checking*, MIT Press, 2008.

[2] Alper Kamil Bozkurt, Yu Wang, and Miroslav Pajic, 'Model-free learning of safe yet effective controllers', in *2021 60th IEEE Conference on Decision and Control (CDC), Austin, TX, USA, December 14-17, 2021*, pp. 6560–6565. IEEE, (2021).

[3] Alper Kamil Bozkurt, Yu Wang, Michael M. Zavlanos, and Miroslav Pajic, 'Control synthesis from linear temporal logic specifications using model-free reinforcement learning', in *International Conference on Robotics and Automation (ICRA)*, pp. 10349–10355, (2020).

[4] Alberto Camacho, Oscar Chen, Scott Sanner, and Sheila A McIlraith, 'Non-markovian rewards expressed in LTL: Guiding search via reward shaping (extended version)', in *GoalsRL, a workshop collocated with ICML/IJCAI/AAMAS*, (2018).

[5] Alberto Camacho, Rodrigo Toro Icarte, Toryn Q Klassen, Richard Anthony Valenzano, and Sheila A McIlraith, 'LTL and beyond: Formal languages for reward function specification in reinforcement learning.', in *IJCAI*, volume 19, pp. 6065–6073, (2019).

[6] Krishnendu Chatterjee, 'Markov decision processes with multiple long-run average objectives', in *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science*, eds., V. Arvind and Sanjiva Prasad, pp. 473–484, Berlin, Heidelberg, (2007). Springer Berlin Heidelberg.

[7] Krishnendu Chatterjee, Rupak Majumdar, and Thomas A Henzinger, 'Markov decision processes with multiple objectives', in *Annual symposium on theoretical aspects of computer science*, pp. 325–336. Springer, (2006).

[8] Jack Clark and Dario Amodei. Faulty Reward Functions in the Wild. https://openai.com/blog/faulty-reward-functions/, 2016. Accessed on: 01/18/2023.

[9] Vrunda Dave, Shankara Narayanan Krishna, Vishnu Murali, and Ashutosh Trivedi, 'Optimal repair for omega-regular properties', in *Automated Technology for Verification and Analysis, ATVA 2022*, eds., Ahmed Bouajjani, Lukás Holík, and Zhilin Wu, volume 13505 of *Lecture Notes in Computer Science*, pp. 354–370. Springer, (2022).

[10] Luca De Alfaro, *Formal verification of probabilistic systems*, Ph.D. dissertation, Stanford University, 1998.

[11] Maor Gaon and Ronen Brafman, 'Reinforcement learning with non-markovian rewards', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34-04, pp. 3980–3987, (2020).

[12] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak, 'Omega-regular objectives in model-free reinforcement learning', in *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 395–412, (2019). LNCS 11427.

[13] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak, 'Faithful and effective reward schemes for model-free reinforcement learning of omega-regular objectives', in *ATVA: Automated Technology for Verification and Analysis*, pp. 108–124, (2020). LNCS 12302.

[14] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak, 'Model-Free Reinforcement Learning for Stochastic Parity Games', in *CONCUR: International Conference on Concurrency Theory*, pp. 21:1–21:16, (2020). LIPIcs 171.

[15] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak, 'Model-free reinforcement learning for lexicographic omega-regular objectives', in *Formal Methods, FM 2021*, pp. 142–159, (2021). LNCS 13047.

[16] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak, 'Mungojerrie: Linear-time objectives in model-free reinforcement learning', in *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 527–545, (2023). LNCS 13993.

[17] Mohammadhosein Hasanbeig, Yiannis Kantaros, Alessandro Abate, Daniel Kroening, George J Pappas, and Insup Lee, 'Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees', in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 5338–5343. IEEE, (2019).

[18] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith, 'Using reward machines for high-level task specification and decomposition in reinforcement learning', in *International Conference on Machine Learning*, pp. 2107–2116, (2018).

[19] Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith, 'Reward machines: Exploiting reward function structure in reinforcement learning', *Journal of Artificial Intelligence Research*, **73**, 173–208, (2022).

[20] Ryohei Oura, Ami Sakakibara, and Toshimitsu Ushio, 'Reinforcement learning of control policy for linear temporal logic specifications using limit-deterministic generalized Büchi automata', *IEEE Control. Syst. Lett.*, **4**(3), 761–766, (2020).

[21] Dorsa Sadigh, Eric S Kim, Samuel Coogan, S Shankar Sastry, and Sanjit A Seshia, 'A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications', in *Conference on Decision and Control (CDC)*, pp. 1091–1096, (December 2014).

[22] Joar Skalse, Lewis Hammond, Charlie Griffin, and Alessandro Abate, 'Lexicographic multi-objective reinforcement learning', *arXiv preprint arXiv:2212.13769*, (2022).

[23] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, second edn., 2018.