About Time: Model-free Reinforcement Learning with Timed Reward Machines

Rajarshi Roy[1*], **Anirban Majumdar**[2], **Ritam Raha**[3], **David Parker**[1], **and Marta Kwiatkowska**[1]
[1]Department of Computer Science, University of Oxford, UK
[2]Tata Institute of Fundamental Research, India
[3]Max Planck Institute for Software Systems, Kaiserslautern, Germany

Abstract

Reward specification plays a central role in reinforcement learning (RL), guiding the agent's behavior. To express non-Markovian rewards, formalisms such as reward machines have been introduced to capture dependencies on histories. However, traditional reward machines lack the ability to model precise timing constraints, limiting their use in time-sensitive applications. In this paper, we propose timed reward machines (TRMs), which are an extension of reward machines that incorporate timing constraints into the reward structure. TRMs enable more expressive specifications with tunable reward logic, for example, imposing costs for delays and granting rewards for timely actions. We study model-free RL frameworks (i.e., tabular Q-learning) for learning optimal policies with TRMs under digital and real-time semantics. Our algorithms integrate the TRM into learning via abstractions of timed automata and employ counterfactual-imagining heuristics that exploit the TRM's structure to improve search. Experimentally, we demonstrate that our algorithm learns policies that achieve high rewards while satisfying the timing constraints specified by the TRM on popular RL benchmarks.

1 Introduction

Reinforcement Learning (RL) [Sutton and Barto, 2018] has become a foundational paradigm for sequential decision-making, enabling agents to learn optimal behavior through interactions with an environment. A crucial aspect of any RL problem is the reward specification, which defines the agent's learning objective. Traditionally, rewards are assumed to depend only on the current state and action, conforming to the *Markov* property. However, many real-world tasks require objectives that depend on the history of states and actions, such as completing a sequence of tasks or avoiding repeated errors. To address this, *non-Markovian* reward formalisms have been developed, with reward machines (RMs) emerging as a prominent approach.

Reward Machines [Icarte *et al.*, 2022] are finite-state automata that specify structured, history-dependent reward functions. They provide a compact and expressive way to encode high-level objectives and have been successfully integrated into RL frameworks, improving sample efficiency and interpretability. However, a critical limitation of existing RMs is their inability to express timing constraints, which are vital for time-sensitive applications such as robotics and autonomous driving [Mehdipour *et al.*, 2023; Sadigh *et al.*, 2018]. For instance, an AV might need to "slow down for 3 seconds to allow a pedestrian to cross" or "avoid an unsafe road for at least 10 seconds".

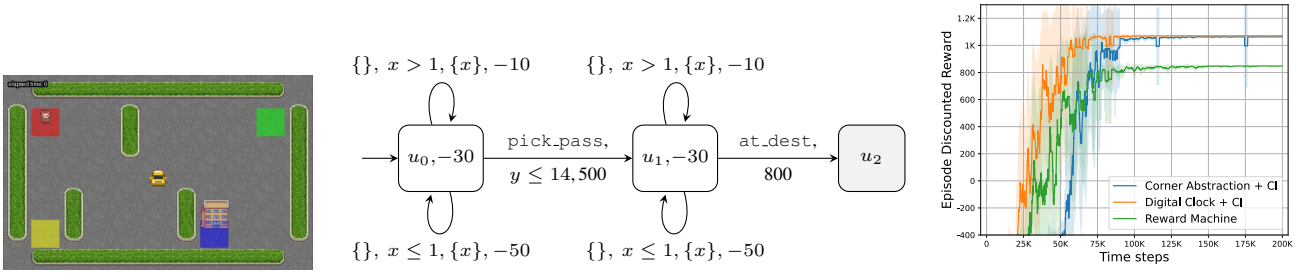
In this paper, we propose timed reward machines (TRMs) that enhance reward machines with fine-grained timing requirements. To this end, we augment reward machines with clocks that track elapsed time and use them to impose timing constraints, drawing inspiration from the timed automata (TA) literature [Alur and Dill, 1994]. TRMs thus allow reward functions to depend not only on the agent's history of states and actions, but also on the time intervals between events. Moreover, TRMs can assign costs and rewards to both states and transitions, incentivizing the agent to complete a task in a timely manner while optimizing the overall reward.

Example 1. *To illustrate our setting, we define a simple TRM (Fig. 1b) on the standard Taxi domain (Fig. 1a). The TRM encourages a taxi agent to drive slowly, e.g., due to heavy traffic, by providing a higher reward when it delays at each step (enforced by a self-loop with timing constraint $x > 1$). Additionally, it imposes a deadline for picking up the passenger (enforced by a transition with timing constraint $y \leq 14$). Finally, after pickup, the agent must reach the destination while driving slowly. Such time-sensitive objectives, involving delays and deadlines, can be naturally captured by TRMs.*

We interpret TRMs over Markov decision processes (MDPs) to model stochastic environments (Section 3). To express timing constraints, we augment the MDP action set with explicit delay actions. We then study two standard timing interpretations—digital-clock (Section 4) and real-time (Section 5)—also known as integer-clock and dense-time interpretations in TA literature [Henzinger *et al.*, 1992]. For each setting, we devise tabular Q-learning algorithms on product MDPs obtained by integrating the TRM in the environment.

In the digital-clock case, the product construction is straightforward: integer clock valuations are directly in-

[*]Part of this work was carried out while Rajarshi Roy was at the University of Liverpool, UK.



(a) Gym Taxi: Passenger is in loc red and destination is in loc blue

(b) A TRM that instructs the taxi to pick up a passenger and drop her at a destination, while moving slow.

(c) Rewards obtained using digital, real-time TRM, and standard reward machine.

Figure 1: An illustration of TRM on Gym Taxi domain.

cluded in the MDP state space. In the real-time case, we consider two approaches: (i) discretized time for continuous delays, and (ii) a corner-point abstraction based on region construction of timed automata, which encourages agents to pick delays that are very close to integers. We further enhance these algorithms with novel counterfactual imagining (CI), inspired by [Icarte *et al.*, 2022], to incorporate alternative clock valuations and timing delays. We also provide a detailed theoretical analysis of all the presented settings, including guarantees of convergence to (ε-)optimal policies.

Our algorithms based on TRMs, as opposed to standard reward machines (without delay actions), can enforce timing constraints and therefore obtain higher rewards that may depend on precise timing. We demonstrate this in Fig. 1c for our running example under both digital and real-time interpretations. Our detailed experiments, involving several TRMs over standard RL benchmarks (Section 6) reveal clear differences between the digital and real-time settings: in scenarios requiring substantial delays, the corner-point abstraction often yields better returns. Moreover, we consistently observe higher returns from our CI heuristics.

Related Work. We contrast our work with existing approaches, presenting them in order of relevance.

RL with Timed Specifications. To our knowledge, only a handful of works consider time-sensitive logical reward specifications for RL. [Xu and Topcu, 2019] optimize Metric Temporal Logic (MTL) objectives by translating them into (simple) timed automata, considering only the digital-clock setting. [Dole *et al.*, 2021; Dole *et al.*, 2023] study subclasses of Duration Calculus that compile to variants of timed automata. Both lines of work use timed automata as monitors (reward 1 for accept, 0 for reject) to maximize *satisfaction*.

Unlike declarative specifications, TRMs give designers fine-grained control over reward logic, including state-based delay costs and transition-based rewards, while retaining standard Markovian rewards on MDPs. We also study both digital-clock and real-time variants, together with several heuristics, theoretically and experimentally. Finally, sparse terminal rewards from monitors are often ineffective in long-horizon tasks, see [Roy *et al.*, 2026, Sec. 7.3] for justification.

RL with Non-Markovian Rewards. To specify non-Markovian rewards, the most widely used formalisms are temporal logics and finite-state machines (FSMs). Among temporal logics, there has been particular focus on Linear Temporal Logic (LTL) [Camacho *et al.*, 2019; Hasanbeig *et al.*, 2019; Bozkurt *et al.*, 2020; Jothimurugan *et al.*, 2021; Shao and Kwiatkowska, 2023]. Most of these approaches focus on translating formulas into FSMs, which are the operational structures used to guide learning.

FSMs are central to non-Markovian RL due to their compositional structure. Reward machines (RMs) [Icarte *et al.*, 2022] provide a general means of defining rewards in the FSM structure. Reward machines have been extended widely: stochastic transitions [Corazza *et al.*, 2022], ω-regular properties [Hahn *et al.*, 2023], partial observability [Icarte *et al.*, 2019], multi-agent settings [Neary *et al.*, 2021] and continuous-time MDP [Falah *et al.*, 2023; Falah *et al.*, 2025].

None of these works addresses fine-grained timing constraints as considered in this paper.

Timed Automata in Control and Planning. Timed automata [Alur and Dill, 1994] are a well-established formalism for modeling and verifying systems with time-dependent behavior [Bozga *et al.*, 1998; Larsen *et al.*, 1997]. Quantitative variants of TA, such as priced timed automata [Behrmann *et al.*, 2001; Bouyer *et al.*, 2004a] and weighted timed automata [Alur *et al.*, 2001], which are extensions with costs or rewards, have been used in strategy synthesis [Behrmann *et al.*, 2007; David *et al.*, 2015] and planning [Bouyer *et al.*, 2004b; Bouyer *et al.*, 2008; Tollund *et al.*, 2024].

Most of these works are model-based, using deductive methods (e.g., symbolic synthesis or game-theoretic techniques) to compute optimal strategies or plans. In contrast, we show that timed reward structures can be operated in a model-free RL setting using statistical methods.

2 Preliminaries and Background

Markov Decision Process. A (labeled) *Markov Decision Process* (MDP) [Sutton and Barto, 2018] is a tuple $\mathcal{M} = (S, A, T, R, \gamma, \mathrm{AP}, L)$, where S is a finite set of states, A is a finite set of actions, $T : S \times A \times S \to [0, 1]$ is a probabilistic transition function, $R : S \times A \times S \to \mathbb{R}$ is a reward function for the immediate rewards, $\gamma \in [0, 1)$ is a discount factor, and $L : S \times A \to 2^{\mathrm{AP}}$ is a labeling function that maps each state-action to a set of propositions AP.

A *policy* for an MDP is a mapping $\pi : (S \times A)^* S \to \Delta(A)$ that assigns a distribution over actions for a given history

of states and actions. For Markovian rewards, it suffices to consider deterministic and positional policies $\pi : S \to A$. The expected cumulative reward for a policy π is defined as: $V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s\right]$, where s_t is the state at time t, a_t is the action taken at time t.

Also, in this paper, we assume a fixed initial state s_{in} of the MDP, and denote the value of a policy as $V^\pi = V^\pi(s_{in})$.

Q-learning. Reinforcement learning (RL) learns policies that maximize discounted reward in MDPs. In model-free RL, the agent samples the environment without explicit transition or reward models. Q-learning, a standard model-free method, learns the optimal action-value function $Q : S \times A \to \mathbb{R}$, the expected return of taking a in s and then following the optimal policy. The Q-update rule is given by: $Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)]$, where α is the learning rate, $R(s,a)$ is the immediate reward received for action a in state s, and s' is the next state.

3 Problem formulation

3.1 Timed Reward Machine (TRM)

We extend classical timed automata with reward functions for the RL setting. For the reward machine formalism, we follow [Icarte *et al.*, 2018]. In our formalism, reward machines are augmented with a set of *clocks* X which can assume values in the time domain $\mathbb{T}$. We consider two different time settings: (1) a digital-clock setting, where $\mathbb{T} = \mathbb{N}$; and (2) a real-time setting, where $\mathbb{T} = \mathbb{R}_{\geq 0}$.

A *guard* is a conjunction of constraints of the form $\phi := x \bowtie c$, where $x \in X$ is a clock, $\bowtie \in \{<, \leq, =, \geq, >\}$ is a comparison operator, and $c \in \mathbb{N}$ is a non-negative constant. We denote the set of all guards over X by $\Phi(X)$. Given a set of clocks X, propositions AP, a TRM is a finite state machine defined as a tuple $\mathcal{A} = (U, u_0, F, \Delta_u, \Delta_r)$, where:

- U is a finite set of states, $u_0 \in U$ is the initial state, and $F \subset U$ is the set of terminal (sink) states;
- $\Delta_u \subseteq U \times 2^{\mathrm{AP}} \times \Phi(X) \times 2^X \times U$ is the transition relation defining the next state given the current state, active propositions in the current state, a guard over clocks, and the clocks to reset. We denote a transition using θ.
- $\Delta_r = \Delta_r^u \cup \Delta_r^\theta$, where $\Delta_r^u : U \to [S \to \mathbb{R}]$ is the state-based reward function and $\Delta_r^\theta : \Delta_u \to [S \times A \times S \to \mathbb{R}]$ is the transition-based reward function.

While our TRM formalism is inspired by priced timed automata, it is generalized for RL frameworks, allowing general Markovian reward functions over both states and transitions. In our setting, one can consider negative rewards to be costs.

A timed word $w = (d_0, l_0)(d_1, l_1)\ldots(d_n, l_n) \in (\mathbb{T} \times 2^{\mathrm{AP}})^*$ is a sequence in which $d_i \in \mathbb{T}$ is a time delay at position i and $l_i \in 2^{\mathrm{AP}}$ is the set of propositions observed after that delay at position i. A *run* $\mathcal{A}^w$ of a timed reward machine $\mathcal{A}$ on w is a sequence $(u_0, v_0) \xrightarrow{d_0,\theta_0,r_0} (u_1, v_1) \xrightarrow{d_1,\theta_1,r_1} \ldots \xrightarrow{d_n,\theta_n,r_n} (u_{n+1}, v_{n+1})$, where, for each i, $u_i \in U$ is the state of the TRM, $v_i \in \mathbb{T}^{|X|}$ is the clock valuation, r_i is the reward function, and θ_i is the transition at that position. The above run satisfies the following conditions:

- u_0 is the initial state, $v_0(x) = 0$ for all $x \in X$.
- let $\theta_i = (u_i, l_i, \phi_i, \rho_i, u_{i+1}) \in \Delta_u$, then the clock valuations satisfy the following condition: $v_i + d_i$ satisfies the guard ϕ_i, and $v_{i+1} = [\rho_i](v_i + d_i)$, where $[\rho_i]$ is the reset function that resets the clocks in ρ_i to 0 and keeps the others unchanged.
- $r_i[0] = \Delta_r^\theta(\theta_i)$ and $r_i[1] = \Delta_r^u(u_i)$ are the transition- and state-based reward functions at position i, resp.

A TRM $\mathcal{A}$ is *deterministic*, if for every state $u \in U$, every set of propositions $l \in 2^{\mathrm{AP}}$, and every pair of guards $\phi_1, \phi_2 \in \Phi(X)$ such that $(u, l, \phi_1, \rho, u') \in \Delta_u$ and $(u, l, \phi_2, \rho', u'') \in \Delta_u$, it holds that $\phi_1 \cap \phi_2 = \emptyset$. As is a common assumption in the literature on reward machines [Icarte *et al.*, 2018], we only consider deterministic TRMs.

Interpretation of TRM on MDPs. We model the environment as an MDP $\mathcal{M} = (S, A', T, \gamma, L, \mathrm{AP})$[1], which extends the standard labeled MDP definition by including timing delays in action space: $A' = \mathbb{T} \times A$. Here, the agent has the option to either act immediately or wait for a chosen amount of time, referred to as *delay*, before taking the next action. Augmenting the agent with delay actions is analogous to standard settings in timed control and games [Bouyer *et al.*, 2004b; Behrmann *et al.*, 2001].

In our setting, the agent's trajectory takes the form of a sequence $\zeta = s_0 \cdot (d_0, a_0) \cdot s_1 \cdot (d_1, a_1) \cdots (d_n, a_n) \cdot s_{n+1} \in (S \times (\mathbb{T} \times A))^* \times S$, where $s_i \in S$ is the MDP state, $d_i \in \mathbb{T}$ is the delay chosen, and $a_i \in A$ is the action taken. Since RL algorithms typically operate on sampled finite trajectories, we restrict attention to bounded-horizon trajectories in this work.

A trajectory ζ induces a timed word $w^\zeta = (d_0 + 1, L(s_0, a_0)) \ldots (d_n + 1, L(s_n, a_n))$, which serves as input to the TRM $\mathcal{A}$. The $+1$ offset in the delays represents the execution duration of an action in the environment. Our choice of a unit offset aligns better with the definitions of reward machines; setting all delays $d_i = 0$ yields a standard untimed word. One could, however, easily consider the action duration to be any constant $c \in \mathbb{R}$ offset. By doing so, we allow general real-time interpretation of MDPs.

On a timed word w^ζ, the TRM $\mathcal{A}$ produces a run $\mathcal{A}^\zeta : (u_0, v_0) \xrightarrow{d_0+1,\theta_0,r_0} (u_1, v_1) \xrightarrow{d_1+1,\theta_1,r_1} \ldots \xrightarrow{d_n+1,\theta_n,r_n} (u_{n+1}, v_{n+1})$. We now define the discounted cumulative reward for a trajectory ζ. This definition follows the standard treatment of discounting in decision processes with sojourn times, as presented in [Puterman, 1994, Eq 11.3.1].

Following that framework, each decision point occurring at time t_i contributes a discounted reward of $\gamma^{t_i} \cdot R_i$ to the total return, where R_i denotes the total reward accumulated during the transition from state s_i to s_{i+1} after taking action a_i. The reward R_i consists of two parts: the lump-sum reward $r_i^\theta(s_i, a_i, s_{i+1})$ obtained from the transition θ in $\mathcal{A}$ and the state-based reward $r_i^u(s_i)$ obtained from the state u in $\mathcal{A}$, accrued over the interval $[t_i, t_{i+1}]$. Formally, the total discounted cumulative reward is defined as:

$$G^\zeta = \sum_{i=0}^{n} \gamma^{t_i} \cdot [r_i^\theta(s_i, a_i, s_{i+1}) + r_i^u(s_i)], \text{where} \quad (1)$$

[1] We here drop the Markovian reward R as it is given by a TRM.

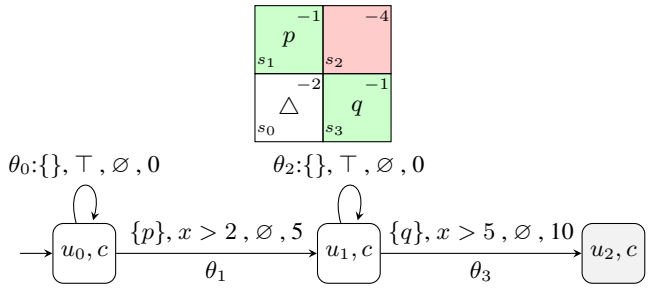


Figure 2: Environment (above) along with TRM objective (below). The cost function c is depicted in the top-right corner of each state.

- $t_i = \sum_{j=0}^{i-1}(d_j + 1)$ for $i \geq 1$, $t_0 = 0$.
- $r_i^\theta = \Delta_r^\theta(\theta_i)$ is the transition-based reward at point i.
- $r_i^u = \begin{cases} \sum_{t=0}^{d_i-1} \gamma^t \Delta_r^u(u_i) = \frac{1-\gamma^{d_i}}{1-\gamma}\Delta_r^u(u_i) & \text{for } \mathbb{T} = \mathbb{N}, \\ \int_0^{d_i} \gamma^t \Delta_r^u(u_i)\, dt = \frac{1-\gamma^{d_i}}{-\ln(\gamma)}\Delta_r^u(u_i) & \text{for } \mathbb{T} = \mathbb{R}_{\geq 0} \end{cases}$ is the state-based reward at point i.

We calculate the state-based reward r_i^u differently for the digital and real-time settings, following standard interpretations [Puterman, 1994]. In the digital clock setting, it is accumulated at each timestep during the delay period, whereas in the real-time setting, it is integrated over the delay interval.

Example 2. *To explain the definitions, consider the environment and the TRM in Fig. 2. The environment has two features, p and q, which denote moving into states s_1 and s_3, resp. Starting at s_0, the TRM $\mathcal{A}$ requires the agent first to observe p and then q, while satisfying simple clock constraints.*

We illustrate two trajectories, ζ_1 and ζ_2, for this example in Table 1. The figure also summarizes, for each trajectory, the induced timed words, the TRM runs, and the resulting discounted returns. Both trajectories use the same environment actions but differ in their delay choices, which leads to different behavior under the TRM. For instance, ζ_1 waits in a "good" state s_1, whereas ζ_2 waits in a "bad" state s_2, incurring a higher cost. Consequently, in the digital-clock setting with $\gamma = 0.9$, ζ_1 attains a higher discounted return ($G^{\zeta_1} \approx 6.4$) than ζ_2 ($G^{\zeta_2} \approx 5.1$). The exact ordering holds in real-time with the same delays (≈ 6.6 vs. ≈ 5.4).

Properties of TRM. We now make some observations about trajectories and rewards in TRMs. As in classical timed automata, a trajectory in TRMs can induce arbitrarily large clock values and delays. However, one can bound them to reason about the expected cumulative reward. As standard [Alur and Dill, 1994], we rely on the maximum constant M appearing in the guards of the TRM $\mathcal{A}$. Formally, $\overline{v}[x] = v[x]$ if $v[x] \leq M$ else $\overline{v}[x] = \infty$ for all $x \in X$, where ∞ denotes the clock values beyond M following the usual comparison semantics, e.g., $\infty > 2$ and $\infty \not\leq 3$. We also define bounded delays as $\overline{d} = d$ if $d < M$ else $\overline{d} = M$.

We extend this definition to trajectories: for a trajectory $\zeta = s_0 \cdot (d_0, a_0) \cdots (d_n, a_n) \cdot s_{n+1}$, we define a *delay-bounded* trajectory $\overline{\zeta} = s_0 \cdot (\overline{d}_0, a_0) \cdots (\overline{d}_n, a_n) \cdot s_{n+1}$. We can show that the delay-bounded trajectory induces a "similar" run in a TRM $\mathcal{A}$ to the original trajectory ζ. This follows from the fact that any clock value $v[x]$ beyond M has the same behavior with any guards on x; see [Roy *et al.*, 2026] for more details.

Moreover, under certain reasonable conditions, bounding the delays can improve the discounted reward. These include inducing costs for delaying in states rather than rewarding, and providing a high terminal reward for completing all tasks. Formally, the state–reward function always has negative values, i.e., $\Delta_r(u) < 0$ for every $u \in \mathcal{A}$, and we search for "good" trajectories ζ where $G_i^\zeta > 0$ for all decision points i, G_i^ζ being the discounted return from i onward. Trajectories as ζ can occur, e.g., when terminal rewards along ζ are sufficiently large. The following lemma presents this idea.

Lemma 1. *Let $\mathcal{A}$ be a TRM, ζ be a trajectory of $\mathcal{A}$ and $\overline{\zeta}$ be its delay-bounded trajectory. If (1) $\Delta_r^u(u) < 0$ for all $u \in U$; and (2) $G_i^\zeta > 0$ for all $0 \leq i \leq n$, then $G^{\overline{\zeta}} \geq G^\zeta$.*

Based on the assumptions of Lem. 1, we can bound the delay space to $\mathbb{D} = \mathbb{T} \cap [0, M]$. In our setting, policies are defined as $\pi : (S \times \mathbb{D} \times A)^* S \to \mathbb{D} \times A$, which map a trajectory to a bounded delay and an action. We call such policies *delay-discounted* policies, as the reward functions defined in Eq. 1 incorporate the discount factor to delays as well.

The expected cumulative reward of a policy π is then the expected discounted sum of rewards over all possible trajectories following π from state s: $V^\pi(s) = \mathbb{E}_{\zeta \sim \pi}[G^\zeta \mid \zeta[0] = s]$. A policy π is *optimal* (resp., ε-optimal) if $V^\pi = V^*$ (resp., $V^\pi \geq V^* - \varepsilon$), where $V^* = \sup_\pi V^\pi(s)$.

Problem 1. *Given a TRM $\mathcal{A}$ and a MDP $\mathcal{M}$, find a delay-discounted (ε-) optimal policy π^*.*

In the following sections, we propose algorithms for Problem 1, both in the digital-clock (dc) and real-time (rt) settings.

4 The Digital Clock Setting

Cross-product space. The most important aspect of our approach is the construction of a cross-product between the underlying MDP $\mathcal{M}$ and the TRM $\mathcal{A}$. The cross-product MDP $\mathcal{M}_{dc}^\otimes = \mathcal{M} \otimes \mathcal{A}$ is similar to what is done for classical reward machines, except that one needs to keep track of the clock values as well. For this, we again consider the maximum constant M appearing in the guards of clock $x \in X$ of $\mathcal{A}$ and use the symbol ∞ for clock values that go beyond M.

The cross-product $\mathcal{M}_{dc}^\otimes = (S^\otimes, A^\otimes, T^\otimes, R^\otimes)$ is defined as: $S^\otimes = S \times U \times V$, where S and U are the set of MDP and the TRM states, resp., and $V = (\{0, \ldots, M\} \cup \{\infty\})^{|X|}$ is the set of bounded clock values, $A^\otimes = \mathbb{D} \times A$ is the set of actions, $T^\otimes : S^\otimes \times A^\otimes \times S^\otimes \to [0, 1]$ and $R^\otimes : S^\otimes \times A^\otimes \times S^\otimes \to \mathbb{R}$ are the transition and the reward functions, resp., defined as:

$$T^\otimes((s,u,v),(d,a),(s',u',v')) = T(s,a,s'), \text{and}$$
$$R^\otimes((s,u,v),(d,a),(s',u',v')) = r^u(s) + r^\theta(s,a,s'), \text{if}$$
$$\exists \theta = (u, L(s,a), \phi, \rho, u'), \text{s.t. } v + d + 1 \models \phi, \text{ and}$$
$$v' = [\rho](v + d + 1), \text{ where}$$

for all $x \in X$, $(v+d+1)[x] = v[x]+d+1$ if $v[x]+d+1 \leq M$, otherwise ∞; $r^u = \frac{1-\gamma^d}{1-\gamma}\Delta_r^u(u)$, and $r^\theta = \Delta_r^\theta(\theta)$.

Theorem 1. *An optimal positional deterministic delay-discounted policy exists for the cross-product MDP $\mathcal{M}_{dc}^\otimes$.*

Trajectory	$\zeta_1\colon s_0\cdot(2,\uparrow)\cdot s_1\cdot(1,\rightarrow)\cdot s_2\cdot(0,\downarrow)\cdot s_3$	$\zeta_2\colon s_0\cdot(2,\uparrow)\cdot s_1\cdot(0,\rightarrow)\cdot s_2\cdot(1,\downarrow)\cdot s_3$
Timed word	$w^{\zeta_1}\colon (2{+}1,\{p\})(1{+}1,\emptyset)(0{+}1,\{q\})$	$w^{\zeta_2}\colon (2{+}1,\{p\})(0{+}1,\emptyset)(1{+}1,\{q\})$
TRM run	$\mathcal{A}^{\zeta_1}\colon (u_0,[0]) \xrightarrow[(5,-2)]{3,\theta_1} (u_1,[3]) \xrightarrow[(0,-1)]{2,\theta_2} (u_1,[5]) \xrightarrow[(10,-4)]{1,\theta_3} (u_2,[6])$	$\mathcal{A}^{\zeta_2}\colon (u_0,[0]) \xrightarrow[(5,-2)]{3,\theta_1} (u_1,[3]) \xrightarrow[(0,-1)]{1,\theta_2} (u_1,[4]) \xrightarrow[(10,-4)]{2,\theta_3} (u_2,[6])$
Digital case	$G^{\zeta_1}\colon [5+(-2)(1+\gamma^1)]+\gamma^3[0+(-1)]+\gamma^5[10+0]\approx 6.4$	$G^{\zeta_2}\colon [5+(-2)(1+\gamma^1)]+\gamma^3[0+0]+\gamma^4[10+(-4)]\approx 5.1$
Real case	$G^{\zeta_1}\colon [5+(-2)\int_0^2\gamma^t dt]+\gamma^3[0+(-1)\int_0^1\gamma^t dt]+\gamma^5[10+0]\approx 6.6$	$G^{\zeta_2}\colon [5+(-2)\int_0^2\gamma^t dt]+\gamma^3[0+0]+\gamma^4[10+(-4)\int_0^1\gamma^t dt]\approx 5.4$

Table 1: Description of trajectories ζ_1 and ζ_2 from Example 2 under digital and real-time semantics ($\gamma = 0.9$).

Due to finite state and action spaces of $\mathcal{M}^{\otimes}_{dc}$, the proof holds using standard results for MDPs [Puterman, 1994].

Q-learning on the cross-product. We adapt Q-learning to the cross-product space by modifying the Q-value updates:

$$Q((s,u,v),(d,a)) \leftarrow Q((s,u,v),(d,a)) + \alpha\big([R+ \gamma^{(d+1)} \max_{(d',a')} Q((s',u',v'),(d',a'))] - Q((s,u,v),(d,a))\big),$$

where R is the reward returned by the TRM for the transition from (s,u,v) to (s',u',v') under delay d and action a. Note that, unlike standard Q-learning, here in the update rule the discounting γ^{d+1} accounts for the delay.

The convergence follows from [Watkins and Dayan, 1992], since $\mathcal{M}^{\otimes}_{dc}$ is a valid finite MDP with probabilities $p((s',u',v')|(s,u,v),(d,a)) = p(s'|s,a)$.

Theorem 2. *Q-learning on the cross-product $\mathcal{M}^{\otimes}_{dc}$ converges to an optimal policy under standard assumptions: every state and action pair is visited infinitely often; and the learning rate α_t satisfies $\sum_{t=0}^{\infty}\alpha_t = \infty$ and $\sum_{t=0}^{\infty}\alpha_t^2 < \infty$.*

4.1 Counterfactual Imagining for Delays

The delay actions expand the action space, so we use *counterfactual imagining* to efficiently explore timing alternatives. In addition to varying the RM state as usual in an untimed setting [Icarte, 2022], we also vary clock values and delays.

Formally, during the Q-learning process, given a realized transition in the cross-product MDP $\mathcal{M}^{\otimes}_{dc}$ $\langle (s,u,v),(d,a),r,(s',u',v')\rangle$, we synthesize counterfactual experiences by varying the TRM states, clock values, and delays: $\big\langle (s,\bar{u},\bar{v}),\ (\bar{d},a),\ \bar{r},\ (s',\bar{u}',\bar{v}')\big\rangle$, where $(\bar{u},\bar{v}) \xrightarrow{\bar{d},\theta,\bar{r}} (\bar{u}',\bar{v}')$ is a single step based on the TRM.

Varying values $\bar{v}$ and delays $\bar{d}$ over all possibilities requires adding several alternatives due to potentially large clock range $\{0,\dots,M\}$. To keep the number of counterfactuals manageable, we restrict attention to promising alternatives. We consider only clock values $\bar{v}$ close to the realized value v, i.e., $\|\bar{v}-v\|_\infty \le r_{\mathsf{CI}}$ for a fixed radius r_{CI} (typically < 5). We further consider only delays $\bar{d}$ that enable satisfaction of guards in the alternative TRM state $\bar{u}$ under $\bar{v}$; specifically, we include delays $\bar{d}$ for all transitions $\theta = (\bar{u}, L(s,a), \phi, \rho, \bar{u}')$ such that $\bar{v}+\bar{d}+1 \models \phi$.

5 The Real-time Clock Setting

In the real-time case, clock values and delay actions can be real-valued, and delays can be chosen from the continuous range $[0,M]$, enabling more precise timing of actions. This allows for more possible policies, often leading to better rewards, which we illustrate through the following example.

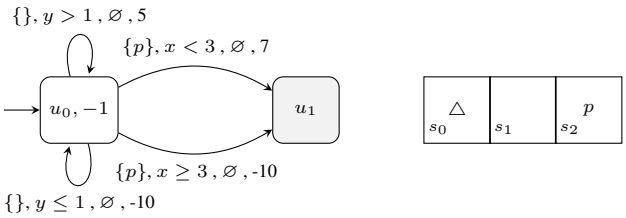


Figure 3: Example distinguishing digital and real-time settings.

Example 3. *Consider the example in Fig. 3. In the real-time setting, there exists a positive-valued policy, which can be seen from the trajectory: $\zeta_1 = s_0\cdot(0.1,\rightarrow)\cdot s_1\cdot(0,\rightarrow)\cdot s_2$, which achieves a discounted reward of $[5+(-1)(1-\gamma^{0.1})/-\ln(\gamma)]+\gamma^{1.1}[7]\approx 11.13$ for $\gamma = 0.9$. In contrast, there is no positive-valued policy in the digital-clock setting. For instance, similar trajectories in this setting, $\zeta_2 = s_0\cdot(0,\rightarrow)\cdot s_1\cdot(0,\rightarrow)\cdot s_2$ and $\zeta_3 = s_0\cdot(1,\rightarrow)\cdot s_1\cdot(0,\rightarrow)\cdot s_2$ will achieve discounted rewards of $-10+\gamma^1[7]\approx -3.7$ and $[5+(-1)(1-\gamma^1)/(1-\gamma)]+\gamma^2[-10]\approx -4.1$, resp.*

Moreover, in contrast to the digital-clock case, in the real-time setting, optimal policies may not exist, as stated below.

Theorem 3. *An optimal delay-discounted policy may not exist in the real-time setting.*

For instance, in Fig. 3, the best sequence $(d,\rightarrow)(0,\rightarrow)$, $0<d<1$, yields return $G^{\zeta} = [5+\frac{(1-\gamma^d)}{-\ln(\gamma)}(-1)]+\gamma^{(1+d)}[7]$. This achieves a supremum of 11.3 as $d\to 0^+$, but this is unattainable since $d=0$ violates the guard.

Therefore, our focus is on ε-optimal policies. However, the usual cross-product MDP $\mathcal{M}^{\otimes}_{rt}$ has infinite state and action spaces. Thus, to apply tabular RL, we design finite, discrete abstractions and focus on *deterministic positional* policies.

Uniform Discretization. A typical approach to approximating real-time is to uniformly discretize time using a step size $0<\frac{1}{\kappa}<1$, $\kappa>1\in\mathbb{N}$. In this setting, the clock valuations would be the set $V_\kappa = \{v\in([0,M]\cup\{\infty\})^{|X|} \mid v[x] = c/\kappa$ or $v[x]=\infty$ for $c=0,...,M\cdot\kappa$, for all $x\in X\}$ and the delay action space $\mathbb{D}_\kappa = \{c/\kappa\in[0,M] \mid c=0,...,M\cdot\kappa\}$. The corresponding finite cross-product MDP $\mathcal{M}^{\otimes}_{ud}$ is constructed as in the digital-clock case, with the modifications to the state and action spaces. Q-learning from Sec. 4 can then be applied, yielding similar convergence guarantees.

To improve the approximation quality, one would require a larger step size κ. However, this substantially increases the size of the valuation and delay spaces, $|V_\kappa| = (M\kappa+1)^{|X|}$ and $|\mathbb{D}_\kappa| = M\kappa+1$, limiting scalability.

5.1 Corner-Point Abstraction for TRMs

To address the challenges of discretizing real-time, we adopt principled abstractions of clock values from the timed automata literature. Specifically, we adapt the corner-point abstraction based on region abstraction [Alur *et al.*, 1992]. While previously used for priced timed automata [Bouyer *et al.*, 2004a], we adapt it to an RL setting, interpreting TRMs over MDPs with discounted rewards.

On an intuitive level, regions partition the infinite set of clock values into finitely many equivalence classes that behave identically w.r.t. guards. Region corners, on the other hand, are the integral boundary points of a region. Typically, an RL agent is incentivized to choose delays near region corners to obtain higher returns (as in the example from Fig. 3).

To formalize these ideas, let us briefly recall the region abstraction and define region corners. Given a set of clocks X and a max-constant M, a *region* is a tuple $(h, [X_0, \ldots, X_p])$, where $h : X \to \{0, \ldots, M\}$, and $(X_i)_{i=0}^{p}$ is a partition of X such that for all $i > 0$, $X_i \neq \emptyset$ and $h(x) = M$ implies $x \in X_0$. A valuation v is in a region if the following conditions hold: (i) $\forall x \in X, \lfloor v(x) \rfloor = h(x)$, (ii) $\forall x \in X, x \in X_0$ iff $\{v(x)\} = 0$ (i.e., $v(x) = h(x)$), and (iii) $\forall x, y \in X, \{v(x)\} \leq \{v(y)\} \iff x \in X_i, y \in X_j, i \leq j$, where $\lfloor c \rfloor$ and $\{c\}$ denote the integer and fractional parts of c, resp. For example, the valuation v with $v(x) = 1.2$, $v(y) = 0.5$ lies in the region $R = (\{x : 1, y : 0\}, [\{\}, \{x\}, \{y\}])$.

Two valuations $v, \overline{v}$ are *region-equivalent*, denoted $v \sim \overline{v}$, if they belong to the same region. For a valuation v, $[v]$ denotes the region to which it belongs. Region-equivalence can be naturally extended to trajectories and policies.

A corner point of a region R is a valuation $v \in \{0, \ldots, M\}^{|X|}$ with integral values for each clock and belongs to the (topological) closure of R, e.g., the corner points of the example region above are $(1, 0)$, $(1, 1)$, and $(2, 1)$.

We here exploit region corners to search for policies in $\mathcal{M}_{rt}^{\otimes}$ that yield higher rewards, namely *corner policies*. Intuitively, such policies choose delays so that the clock valuations encountered along each possible trajectory lie at region corners. The following lemma formalizes the benefits of corner policies. For a technical exposition, see [Roy *et al.*, 2026].

Lemma 2. *Given $\varepsilon > 0$ and any policy π in $\mathcal{M}_{rt}^{\otimes}$, there exists a region-equivalent corner policy $\hat{\pi}$ in $\mathcal{M}_{rt}^{\otimes}$ such that $V^{\hat{\pi}} > V^{\pi} - \varepsilon$ for some discounting $\gamma < 1$.*

This lemma restricts the search space to corner policies; however, it still remains infinite. To address this, instead of searching over the real-time cross-product MDP $\mathcal{M}_{rt}^{\otimes}$, we construct a 'finite' abstraction $\mathcal{M}_{ca}^{\otimes}$ utilizing region corners.

Cross-product for Corner-Point Abstraction. We first describe how a corner configuration (R, α) of the corner-abstraction of $\mathcal{A}$, where R is a region and α is its corner point, evolves under elapsing time. Here, the agent, in addition to a delay $d \in \mathbb{D} = \{0, \ldots, M\}$, chooses a region successor $\sigma \in \mathbb{S} = \{-2|X|, \ldots, 0, \ldots, 2|X|\}$ that assigns which region to move to associated with a corner[2]. Intuitively, applying a delay-successor tuple (d, σ) to a configuration (R, α) leads to a new configuration (R', α') obtained as follows: first shift both R and α by d time unit, and then choose the σ^{th} successor region associated with that corner. Formally, $(R, \alpha) \oplus (d, \sigma)$ is the new configuration (R', α') defined as: $\alpha' = \alpha + d$, $R''[h] = R[h] + d$ and R' is the σ^{th} successor region of R'' associated with α'.

Example 4. *Consider a corner configuration $(R = (\{x : 1, y : 0\}, [\{x\}, \{y\}]), \alpha = (1, 0))$. This region contains valuations such as $v(x) = 1$, $v(y) = 0.1$. Applying delay $(1, 0)$ leads to $(R_1 = (\{x : 2, y : 1\}, [\{x\}, \{y\}]), \alpha_1 = (2, 1))$, which is the same region and corner pair offset by $+1$. This region contains valuations such as $v(x) = 2$, $v(y) = 1.1$. Alternatively, applying delay $(1, 1)$ leads to $(R_2 = (\{x : 2, y : 1\}, [\{\}, \{x\}, \{y\}]), \alpha_2 = (2, 1))$, which is the region successor of R_1 associated with the same corner. This region contains valuations such as $v'(x) = 2.1$, $v'(y) = 1.2$. We graphically illustrate this example in [Roy* et al.*, 2026].*

We then define the cross-product MDP $\mathcal{M}_{ca}^{\otimes} = (S^{\otimes}, A^{\otimes}, T^{\otimes}, R^{\otimes})$ as follows: $S^{\otimes} = S \times U \times \mathcal{R} \times \mathcal{C}$, where $\mathcal{R}$ is the set of regions of $\mathcal{A}$, and $\mathcal{C}$ is the set of corner points associated with the regions; $A^{\otimes} = \mathbb{D} \times \mathbb{S} \times A$, where $\mathbb{D} = \{0, 1, \ldots, M\}$ and $\mathbb{S} = \{-2|X|, \ldots, 0, \ldots, 2|X|\}$; and $T^{\otimes} : S^{\otimes} \times A^{\otimes} \times S^{\otimes} \to [0, 1]$ and $R^{\otimes} : S^{\otimes} \times A^{\otimes} \times S^{\otimes} \to \mathbb{R}$ are defined as follows:

$$T^{\otimes}((s,u,R,\alpha),(d,\sigma,a),(s',u',R',\alpha')) = T(s,a,s'), \text{ and}$$

$$R^{\otimes}((s,u,R,\alpha),(d,\sigma,a),(s',u',R',\alpha')) = r^{u}(s) + r^{\theta}(s,a,s'),$$

if $\exists \theta = (u, L(s,a), \phi, \rho, u')$ and R'', s.t.

$$R'' \models \phi, \text{and } R' = [\rho](R''),$$

$$\text{where } (R'', \alpha'') = (R, \alpha) \oplus (d+1, \sigma), R'' \models \phi,$$

and $r^{u} = \frac{1-\gamma^{d}}{-\ln(\gamma)} \Delta_r^{u}(u)$ and $r^{\theta} = \Delta_r^{\theta}(\theta)$.

The required operations on regions for this construction are standard and can be computed efficiently [Alur *et al.*, 1992].

Hence, standard Q-learning can be applied on the corner-point abstraction MDP $\mathcal{M}_{ca}^{\otimes}$ to obtain an optimal policy. Moreover, we can prove that an optimal policy in $\mathcal{M}_{ca}^{\otimes}$ can be "lifted" to an ε-optimal region-equivalent corner policy in the original real-time MDP $\mathcal{M}_{rt}^{\otimes}$. Thus, we can show that:

Theorem 4. *Given $\varepsilon > 0$, our algorithm via Q-learning on the corner-point abstraction $\mathcal{M}_{ca}^{\otimes}$ returns an ε-optimal policy π^* for the real-time MDP $\mathcal{M}_{rt}^{\otimes}$ for some discounting $\gamma < 1$.*

We also design counterfactual imagining for the corner abstraction, analogous to the digital-clock setting (Section 4.1).

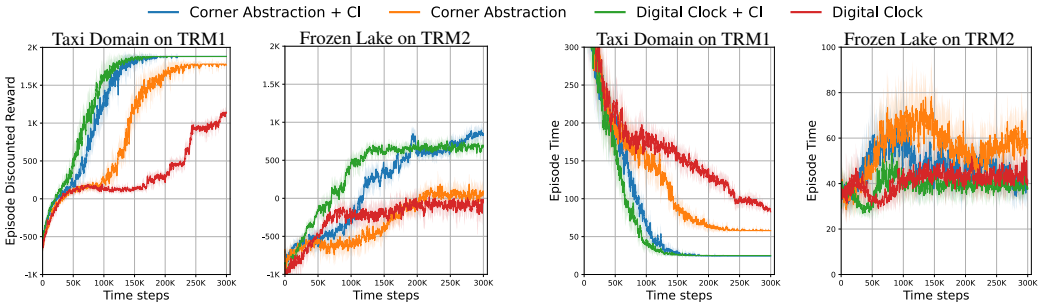
6 Evaluation

We implemented all proposed algorithms in Python 3[3] by extending [Icarte *et al.*, 2022]. All TRM-specific components, including corner-abstractions, were developed from scratch.

To improve learning performance, we employ several heuristics for interpreting TRMs. First, we reduce the valuation space V using clock-specific maximum constants M_x for each $x \in X$, as is standard in timed automata. Second, we bound the delay space $\mathbb{D}$ by the largest constant M_d appearing in guards of the form $x \bowtie c$ with $\bowtie \in \{>, \geq, =\}$. There is no loss of optimality, since delays beyond M_d only add cost.

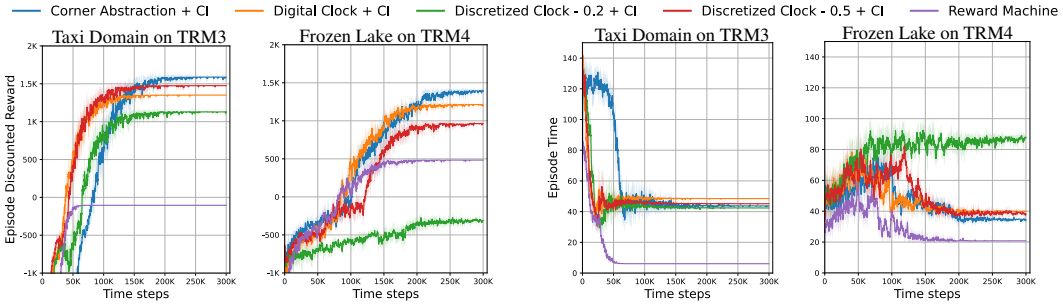
[2] Some successors may be invalid or not distinct for some (R, α).

[3] See https://github.com/ritamraha/Timed-Reward-Machines.



(a) Discounted reward comparison (b) Episode time comparison (including delays)

Figure 4: RQ1: Performance gain for counterfactual imagining for digital and real-time settings for two environments.



(a) Discounted reward comparison (b) Episode time comparison (including delays)

Figure 5: RQ2: Performance difference for various timed interpretations

Experimental Results. We address two key research questions: RQ1, the performance gains from counterfactual imagining; and RQ2, the performance differences across time interpretations. As TRMs are novel to the RL framework, there are no direct baselines to compare; we expand on the conceptual difference with other settings in [Roy *et al.*, 2026, Sec. 7].

Our evaluation uses standard Gym [Towers *et al.*, 2024] environments: (i) the *Taxi* domain, as in Fig. 1a, with propositions indicating colored pick-up locations and whether the passenger is in the taxi or at the destination; and (ii) *Frozen Lake* (default 8×8), augmented with three goals (a, b, c) and ten holes (h), with action success probability 0.8. Further details, including the precise TRMs, environments, and additional experimental statistics, are given in [Roy *et al.*, 2026].

We use Q-learning with per-episode parameter decay 0.999, initial rate $\alpha_0 = 0.9$, initial exploration $\varepsilon_0 = 0.9$, initial Q-values $Q_0 = 10$, $\gamma = 0.999$ and maximum global steps of 300 K. For counterfactual imagining (CI), we select the top 15 by rewards per transition. We averaged the results of each experiment over 10 independent runs.

RQ1. We demonstrate the gain of using CI on Taxi domain with TRM1 and on Frozen Lake with TRM2. TRM1 requires the Taxi agent to pick up a passenger, visit a green location, and drop them at the destination, while satisfying several timing constraints. TRM2 requires the Frozen Lake agent to satisfy three objectives, a, b, c, sequentially, while avoiding falling into the holes, and it must also move slowly.

Fig. 4a shows that CI yields significantly higher discounted rewards in both settings by exploring additional ways to satisfy timing constraints. Fig. 4b shows that CI also significantly reduces episode time, allowing agents to be faster.

RQ2. We evaluate the performance of different timing interpretations: (i) digital-clock setting, (ii) uniform discretization with $1/\kappa \in \{0.2, 0.5\}$, (iii) corner-point abstraction, and (iv) reward machines (RM). Note that RMs cannot choose delay actions, as it is not designed for timed specifications. We demonstrate this comparison on Taxi domain with TRM3 and Frozen Lake with TRM4. These TRMs are similar to those in the previous experiment, but with different timing constraints.

Fig. 5a shows that the corner-point abstraction consistently outperforms the other interpretations in terms of discounted return. This advantage is due to selecting delays that lie close to the guards. Fig. 5b shows that RMs lead to the shortest episodes; however, this behavior is undesirable, as completing episodes too fast violates several timing constraints.

7 Conclusion

We studied model-free RL for *timed reward machines* (TRMs), a formalism that augments reward machines with clocks for timing constraints. We interpreted TRMs over MDPs under digital and real-time semantics and devised abstractions for efficient RL. Our experiments show which abstractions and heuristics perform well for RL with TRMs.

This work is a step towards improving time-sensitive reward specification in RL. There are numerous avenues ahead, e.g., extending TRMs to CTMDPs [Falah *et al.*, 2025], adapting deep RL [Fujimoto *et al.*, 2018] to real-time, etc.

Acknowledgements. Rajarshi Roy, David Parker and Marta Kwiatkowska received funding from the ERC under the European Union's Horizon 2020 research and innovation programme (grant agreement No.834115, FUN2MODEL). Rajarshi Roy was also partly funded by the European Union (RobustifAI project, ID 101212818). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency (HADEA). Neither the European Union nor the granting authority can be held responsible for them. Anirban Majumdar was supported by the Department of Atomic Energy, Government of India, under project no. RTI4014. We also thank Anne-Kathrin Schmuck for insightful discussions.

References

[Alur and Dill, 1994] Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[Alur *et al.*, 1992] Rajeev Alur, Costas Courcoubetis, David L. Dill, Nicolas Halbwachs, and Howard Wong-Toi. An implementation of three algorithms for timing verification based on automata emptiness. In *RTSS*, pages 157–166. IEEE Computer Society, 1992.

[Alur *et al.*, 2001] Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. In *HSCC*, volume 2034 of *Lecture Notes in Computer Science*, pages 49–62. Springer, 2001.

[Behrmann *et al.*, 2001] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Guldstrand Larsen, Paul Pettersson, Judi Romijn, and Frits W. Vaandrager. Minimum-cost reachability for priced timed automata. In *HSCC*, volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2001.

[Behrmann *et al.*, 2007] Gerd Behrmann, Agnès Cougnard, Alexandre David, Emmanuel Fleury, Kim Guldstrand Larsen, and Didier Lime. Uppaal-tiga: Time for playing games! In *CAV*, volume 4590 of *Lecture Notes in Computer Science*, pages 121–125. Springer, 2007.

[Bouyer *et al.*, 2004a] Patricia Bouyer, Ed Brinksma, and Kim Guldstrand Larsen. Staying alive as cheaply as possible. In *HSCC*, volume 2993 of *Lecture Notes in Computer Science*, pages 203–218. Springer, 2004.

[Bouyer *et al.*, 2004b] Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim Guldstrand Larsen. Optimal strategies in priced timed game automata. In *FSTTCS*, volume 3328 of *Lecture Notes in Computer Science*, pages 148–160. Springer, 2004.

[Bouyer *et al.*, 2008] Patricia Bouyer, Ed Brinksma, and Kim Guldstrand Larsen. Optimal infinite scheduling for multi-priced timed automata. *Formal Methods Syst. Des.*, 32(1):3–23, 2008.

[Bozga *et al.*, 1998] Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. Kronos: A model-checking tool for real-time systems. In *CAV*, volume 1427 of *Lecture Notes in Computer Science*, pages 546–550. Springer, 1998.

[Bozkurt *et al.*, 2020] Alper Kamil Bozkurt, Yu Wang, Michael M. Zavlanos, and Miroslav Pajic. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*, pages 10349–10355. IEEE, 2020.

[Camacho *et al.*, 2019] 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 *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 6065–6073. ijcai.org, 2019.

[Corazza *et al.*, 2022] Jan Corazza, Ivan Gavran, and Daniel Neider. Reinforcement learning with stochastic reward machines. In *AAAI*, pages 6429–6436. AAAI Press, 2022.

[David *et al.*, 2015] Alexandre David, Peter Gjøl Jensen, Kim Guldstrand Larsen, Marius Mikucionis, and Jakob Haahr Taankvist. Uppaal stratego. In *TACAS*, volume 9035 of *Lecture Notes in Computer Science*, pages 206–211. Springer, 2015.

[Dole *et al.*, 2021] Kalyani Dole, Ashutosh Gupta, John Komp, Shankaranarayanan Krishna, and Ashutosh Trivedi. Event-triggered and time-triggered duration calculus for model-free reinforcement learning. In *RTSS*, pages 240–252. IEEE, 2021.

[Dole *et al.*, 2023] Kalyani Dole, Ashutosh Gupta, John Komp, Shankaranarayanan Krishna, and Ashutosh Trivedi. Correct-by-construction reinforcement learning of cardiac pacemakers from duration calculus requirements. In *AAAI*, pages 14792–14800. AAAI Press, 2023.

[Falah *et al.*, 2023] Amin Falah, Shibashis Guha, and Ashutosh Trivedi. Reinforcement learning for omega-regular specifications on continuous-time MDP. In *ICAPS*, pages 578–586. AAAI Press, 2023.

[Falah *et al.*, 2025] Amin Falah, Shibashis Guha, and Ashutosh Trivedi. Continuous-time reward machines. In *IJCAI*, pages 5056–5064. ijcai.org, 2025.

[Fujimoto *et al.*, 2018] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 1582–1591. PMLR, 2018.

[Hahn *et al.*, 2023] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Omega-regular reward machines. In *ECAI*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 972–979. IOS Press, 2023.

[Hasanbeig *et al.*, 2019] 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 *58th IEEE Conference on Decision and Control, CDC 2019, Nice, France, December 11-13, 2019*, pages 5338–5343. IEEE, 2019.

[Henzinger *et al.*, 1992] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. What good are digital clocks? In *ICALP*, volume 623 of *Lecture Notes in Computer Science*, pages 545–558. Springer, 1992.

[Icarte *et al.*, 2018] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Anthony Valenzano, and Sheila A. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 2112–2121. PMLR, 2018.

[Icarte *et al.*, 2019] Rodrigo Toro Icarte, Ethan Waldie, Toryn Q. Klassen, Richard Anthony Valenzano, Margarita P. Castro, and Sheila A. McIlraith. Learning reward machines for partially observable reinforcement learning. In *NeurIPS*, pages 15497–15508, 2019.

[Icarte *et al.*, 2022] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Anthony Valenzano, and Sheila A. McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *J. Artif. Intell. Res.*, 73:173–208, 2022.

[Icarte, 2022] Rodrigo Toro Icarte. *Reward Machines*. PhD thesis, University of Toronto, Canada, 2022.

[Jothimurugan *et al.*, 2021] Kishor Jothimurugan, Suguman Bansal, Osbert Bastani, and Rajeev Alur. Compositional reinforcement learning from logical specifications. In *NeurIPS*, pages 10026–10039, 2021.

[Larsen *et al.*, 1997] Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. *Int. J. Softw. Tools Technol. Transf.*, 1(1-2):134–152, 1997.

[Mehdipour *et al.*, 2023] Noushin Mehdipour, Matthias Althoff, Radboud Duintjer Tebbens, and Calin Belta. Formal methods to comply with rules of the road in autonomous driving: State of the art and grand challenges. *Automatica*, 152:110692, 2023.

[Neary *et al.*, 2021] Cyrus Neary, Zhe Xu, Bo Wu, and Ufuk Topcu. Reward machines for cooperative multi-agent reinforcement learning. In *AAMAS*, pages 934–942. ACM, 2021.

[Puterman, 1994] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.

[Roy *et al.*, 2026] Rajarshi Roy, Anirban Majumdar, Ritam Raha, David Parker, and Marta Kwiatkowska. About time: Model-free reinforcement learning with timed reward machines. *CoRR*, abs/2512.17637, 2026.

[Sadigh *et al.*, 2018] Dorsa Sadigh, Nick Landolfi, Shankar S. Sastry, Sanjit A. Seshia, and Anca D. Dragan. Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state. *Auton. Robots*, 42(7):1405–1426, 2018.

[Shao and Kwiatkowska, 2023] Daqian Shao and Marta Kwiatkowska. Sample efficient model-free reinforcement learning from LTL specifications with optimality guarantees. In *IJCAI*, pages 4180–4189. ijcai.org, 2023.

[Sutton and Barto, 2018] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction, 2nd Edition*. MIT Press, 2018.

[Tollund *et al.*, 2024] Rasmus G. Tollund, Nicklas S. Johansen, Kristian Ø. Nielsen, Álvaro Torralba, and Kim G. Larsen. Optimal infinite temporal planning: Cyclic plans for priced timed automata. In *ICAPS*, pages 588–596. AAAI Press, 2024.

[Towers *et al.*, 2024] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[Watkins and Dayan, 1992] Christopher J. C. H. Watkins and Peter Dayan. Technical note q-learning. *Mach. Learn.*, 8:279–292, 1992.

[Xu and Topcu, 2019] Zhe Xu and Ufuk Topcu. Transfer of temporal logic formulas in reinforcement learning. In *IJCAI*, pages 4010–4018. ijcai.org, 2019.