Expressive Temporal Specifications for Reward Monitoring

Omar Adalat[1], **Francesco Belardinelli**[1]

[1]Imperial College London
o.adalat24@imperial.ac.uk, francesco.belardinelli@imperial.ac.uk

Abstract

Specifying informative and dense reward functions remains a pivotal challenge in Reinforcement Learning, as it directly affects the efficiency of agent training. In this work, we harness the expressive power of *quantitative* Linear Temporal Logic on finite traces to synthesize *reward monitors* that generate a dense stream of rewards for *runtime*-observable state trajectories. By providing nuanced feedback during training, these monitors guide agents toward optimal behaviour and help mitigate the well-known issue of *sparse rewards* under long-horizon decision-making, which arises under the Boolean semantics dominating the current literature. Our framework is algorithm-agnostic and only relies on a state labelling function, and naturally accommodates specifying non-Markovian properties. Empirical results show that our quantitative monitors consistently subsume and, depending on the environment, outperform Boolean monitors in maximising a quantitative measure of task completion and in reducing convergence time.

Code — https://github.com/nightly/quantitative-reward-monitoring

Extended version — https://arxiv.org/abs/2511.12808

1 Introduction

Reinforcement Learning (RL) offers a powerful framework for sequential decision-making under uncertainty, leveraging sampling-based methods to iteratively refine policies to optimality (Sutton, Barto et al. 1998). Central to the success of any RL algorithm is the careful design of its reward function (Eschmann 2021). Ill-defined reward functions might lead to unintended behaviours (Leike et al. 2017), unforeseen consequences (Denison et al. 2024), and catastrophic failures (Festor et al. 2022). As such, the specification of reward functions is key to the development of safe and robust RL, including the AI alignment problem (Christian 2021).

Within this landscape, temporal logics (Baier and Katoen 2008) stand out as an elegant alternative to conventional reward specification approaches. By offering a formal, high-level language for expressing complex objectives and constraints, temporal logics enable more structured reward design and facilitate clearer reasoning about an agent's goals and behaviour. Consequently, temporal logic-based reward definitions can reduce the burden of manual engineering, while improving interpretability and maintainability, thus paving the way for more robust RL systems (Li et al. 2019; Hasanbeig, Kroening, and Abate 2020).

In the current literature, logical specifications in RL (including reward specifications) have mainly settled on qualitative, Boolean semantics, where a formula is evaluated as true or false (Sadigh et al. 2014). However, Boolean semantics inherently leads to *sparse rewards*, as feedback is often provided only upon complete satisfaction of the relevant formula. Sparse rewards pose significant challenges to RL algorithms, as they hinder effective credit assignment and increase sample complexity (Andrychowicz et al. 2017), limiting performance in long-horizon tasks.

To mitigate these issues, we investigate the use of quantitative Linear Temporal Logic on finite traces (denoted as $\text{LTL}_f[\mathcal{F}]$) to specify and synthesize *reward monitors* for providing richer feedback. Quantitative semantics enable assessing the degree to which a given formula is satisfied besides being just true or false.

Contributions. In this paper we provide a procedure to automatically construct *quantitative reward monitors* starting with rewards specified as formulas in $\text{LTL}_f[\mathcal{F}]$. Such monitors provide a reward in real-time by consuming a trace of states. The agent is ascribed a corresponding reward based on progress, using real-valued evaluation of a formula in *quantitative linear temporal logic interpreted on finite traces*. We evaluated the proposed approach empirically, investigating the efficacy of the approach in terms of convergence speed and a domain-specific task completion measure, against Boolean monitors and handcrafted reward functions. We demonstrate superior performance compared to Boolean monitors (in settings where quantitative information can be exploited) and, in some cases, manually-specified reward functions. Furthermore, we show how a quantitative reward monitor can be constructed with linear overhead in the number of transitions and states w.r.t. the formula size, and how non-Markovian properties are captured through temporally extended goals. Since quantitative semantics provide denser rewards, this leads to greater sample efficiency, ability to handle long-horizon tasks, and avoid sparse rewards.

2 Background

In this section we introduce the background required to build quantitative monitors from $\text{LTL}_f[\mathcal{F}]$ specifications.

Reinforcement Learning (RL) environments are typically modelled as *Markov decision processes* (MDPs), defined as tuples $\mathcal{M} = (\mathcal{S}, A, \Pr, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is a set of *states*; A is a set of *actions*; $\Pr : \mathcal{S} \times A \to \Delta(\mathcal{S})$ is the *transition kernel*, with $\Delta(\mathcal{S})$ being the set of probability distributions over $\mathcal{S}$, $\mathcal{R} : \mathcal{S} \times A \times \mathcal{S} \to \mathbb{R}$ is the (Markovian) *reward function*, and $\gamma \in [0, 1]$ is the *discount factor*.

In an MDP, a (stochastic) *policy* is a function $\pi : \mathcal{S} \to \Delta(A)$ and its performance is measured by the expected discounted return $V^\pi(s) = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_{t+1} \;\Big|\; \mathcal{S}_0 = s \Big]$, where the expectation is over the trajectory generated by following policy π from the initial state $\mathcal{S}_0 = s$. The goal of RL is to find an optimal policy $\pi^\star$ satisfying $\pi^\star \in \arg\max_\pi V^\pi(s)$ for every $s \in \mathcal{S}$. In this setting, the *Markov property* ensures that the next state and reward depend only on the current state and action. A *labelled Markov decision process* extends an MDP by adding a labelling function $\mathcal{L} : \mathcal{S} \to [0, 1]^{\mathcal{P}}$ from the set $\mathcal{S}$ of states to the set $\mathcal{P}$ of atoms.

Quantitative Linear Temporal Logic on finite traces. Linear Temporal Logic on finite traces (LTL_f) shares the syntax with Linear Temporal Logic (LTL), but has different semantics, due to being interpreted on traces of finite length (De Giacomo, Vardi et al. 2013). Naturally, episodes in RL have finite length, which is suitable for specifications interpreted on finite traces. Quantitative LTL_f ($\text{LTL}_f[\mathcal{F}]$) shares the same syntax as LTL_f, but instead of having a two-valued semantics, computations are assigned a real-valued scalar within the range $[0, 1]$, thus extending Zadeh's fuzzy propositional logic with linear temporal connectives (Lamine and Kabanza 2000; Faella, Legay, and Stoelinga 2008).

Definition 1 ($\text{LTL}_f[\mathcal{F}]$ syntax)**.** *Given a set $\mathcal{P}$ of atoms, the syntax of $\text{LTL}_f[\mathcal{F}]$ is defined as follows, where $p \in \mathcal{P}$:*

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid X\varphi \mid \varphi \,\mathcal{U}\, \varphi \mid \varphi \,\mathcal{R}\, \varphi$$

where $\top$ denotes true and $\bot \stackrel{def}{=} \neg\top$ denotes false.

Here X (Next), $\mathcal{U}$ (Until), and $\mathcal{R}$ (Release) are the standard LTL operators. The Boolean connectives, $\vee$ (or), $\rightarrow$ (implies), and $\leftrightarrow$ (iff) can be introduced by their standard abbreviations. The temporal operators F (eventually) and G (always) can be derived as $F\varphi \stackrel{def}{=} \top\mathcal{U}\varphi$ and $G\varphi \stackrel{def}{=} \bot\,\mathcal{R}\,\varphi$.

We also consider a fragment of LTL_f, termed *safe*-LTL_f, to syntactically identify whether a given specification is purely a safety objective (Sistla 1994).

Definition 2 (Safe-LTL_f)**.** *A safe-LTL_f formula φ is an until-free formula in negation normal form, such that its syntax comprises:*

$$\varphi \quad ::= \quad \top \mid p \mid \neg p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid X\varphi \mid \varphi_1 \,\mathcal{R}\, \varphi_2$$

The semantics for $\text{LTL}_f[\mathcal{F}]$ is provided as follows, where λ denotes a finite trace of states, and λ_i represents the i^{th} state in the trace.

Definition 3 (Semantics of $\text{LTL}_f[\mathcal{F}]$)**.** *The evaluation $[\![\varphi, i]\!](\lambda) \in [0, 1]$ of formula φ on the finite trace λ is defined inductively as follows:*

$$\begin{aligned}
[\![\top, i]\!](\lambda) &= 1 \\
[\![p, i]\!](\lambda) &= \mathcal{L}(\lambda_i)(p) \\
[\![\neg\varphi, i]\!](\lambda) &= 1 - [\![\varphi, i]\!](\lambda) \\
[\![\varphi_1 \wedge \varphi_2, i]\!](\lambda) &= \min\big([\![\varphi_1, i]\!](\lambda),\, [\![\varphi_2, i]\!](\lambda)\big) \\
[\![X\varphi, i]\!](\lambda) &= \begin{cases} [\![\varphi, i+1]\!](\lambda), & \text{if } i < |\lambda| \\ 0, & \text{otherwise,} \end{cases}
\end{aligned}$$

$$[\![\varphi_1 \,\mathcal{U}\, \varphi_2, i]\!](\lambda) = \max_{i \le j \le |\lambda|} \min_{i \le k < j} \Big([\![\varphi_2, j]\!](\lambda), [\![\varphi_1, k]\!](\lambda) \Big)$$

$$[\![\varphi_1 \,\mathcal{R}\, \varphi_2, i]\!](\lambda) = \min_{i \le j \le |\lambda|} \max_{i \le k < j} \Big([\![\varphi_2, j]\!](\lambda),\, [\![\varphi_1, k]\!](\lambda) \Big)$$

The truth value for derived logical connectives and temporal operators can be obtained from Def. 3.

Reward monitors are a type of transducer, which consumes a trace $\tau \stackrel{def}{=} (s_1, s_2, \ldots, s_n)$ of states and provides as output a sequence $r = (r_1, r_2, \ldots, r_n)$ of scalar rewards, one for each time step. We define a Boolean reward monitor as a specific type of transducer:

Definition 4 (BRM)**.** *A* Boolean reward monitor *is a Moore machine, defined as a tuple $\mathcal{B} = (Q, q_0, \Sigma, \Gamma, \delta, \theta)$, where Q is the set of the states, with initial state q_0. Σ is the input alphabet, Γ is the output alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $\theta : Q \to \mathbb{R}$ is the output (reward) function.*

Construction of a reward monitor is given by a set of specification-reward pairs, which are defined as follows:

Definition 5 (Specification-reward pairs)**.** *A specification-reward pair is a tuple (φ, ρ), where φ is an LTL_f or $\text{LTL}_f[\mathcal{F}]$ formula, and ρ is the associated scalar weight.*

In order to construct a Boolean reward monitor from an LTL_f formula, one can transform the formula into a deterministic finite automaton (DFA) (De Giacomo, Vardi et al. 2013), and then use it as a reward machine returning a reward of 0 on all states, except for accepting states outputting a reward of 1 (which can be multiplied by scalar ρ).

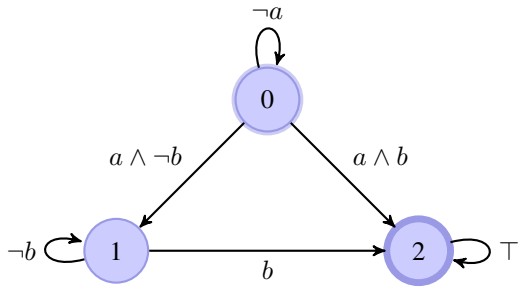
3 Quantitative Reward Monitors

In this section we provide the definition of quantitative reward monitors, as well as the procedure to build them from $\text{LTL}_f[\mathcal{F}]$ specifications.

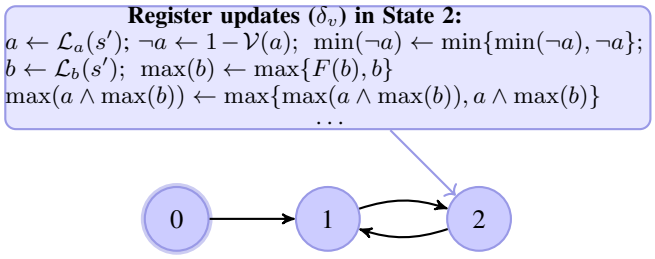
3.1 Monitor Definition

A *quantitative reward monitor* is a *finite state machine with registers*, defined as follows:

Definition 6 (QRM)**.** *A* Quantitative Reward Monitor *is defined as a tuple $\mathcal{A} = (Q, q_0, \delta_q, \mathcal{V}, \delta_v, \delta_r)$, where Q is the set of states, with $q_0 \in Q$ the initial state, a set $\mathcal{V} \subseteq T \times \mathbb{R}$ of register-value pairs (we write $\mathcal{V}(t)$ to retrieve the stored register value t) where T is the set of registers, $\delta_q : Q \to Q$*



(a) Boolean reward monitor (Moore machine) for the LTL$_f$ formula $\neg a \,\mathcal{U}\, (a \wedge Fb)$, where state 0 is the initial state, and state 2 can be viewed as the accepting state which outputs a reward.



(b) Quantitative reward monitor for the LTL$_f[\mathcal{F}]$ formula $\neg a \,\mathcal{U}\, (a \wedge Fb)$, where state 0 is the distinguished initial state. The register update function is partially displayed only for state 2 for brevity.

Figure 1: A boolean monitor with LTL$_f$ semantics (left-hand side) and quantitative monitor with LTL$_f[\mathcal{F}]$ semantics (right-hand side) constructed for the same formula: $\neg a \,\mathcal{U}\, (a \wedge Fb)$.

is the transition function, $\delta_v : Q \times T \to \mathcal{V}$ is the register update function, and $\delta_r = t_{reward} \times \rho$ is the reward function, where t_{reward} is the reward register and ρ is the scalar in the reward specification pair.

The registers update throughout the trace, and track aspects relevant to formula evaluation, for example the minimum and maximum of a subformula as required by the semantics of temporal operators F and G respectively. Intuitively, the reward register t_{reward} denotes the degree of satisfaction of a given specification's formula, hence providing quantitative feedback. In Figure 1 we demonstrate how both Boolean and quantitative monitors would be constructed for the same formula $\neg a \,\mathcal{U}\, (a \wedge Fb)$.

3.2 Safety Specifications

Reward monitors (both Boolean and quantitative) focus on two principal classes of objectives, namely *reachability* and *safety* objectives. From φ in a reward-specification pair, we can syntactically infer whether the specification comprises a safety objective (according to Def. 2), where we denote a safety formula (syntactically inferred) by φ_{safety}.

A *safety specification* requires that once φ_{safety} is violated at any time step, the monitor emits a fixed penalty for the remainder of the trace; that is, rewards are clamped to $\zeta \in \mathbb{R}$ with $\zeta \leq 0$ (typically $\zeta = 0$ or a negative constant). This covers both terminating failures, where the episode ends immediately, as well as non-terminating hazards, where the agent should not be able to recover to a positive return after an unsafe act (e.g. a self-driving car running a red light). Formally, if a safety formula φ_{safety} is violated at index i, then δ_r outputs ζ for all subsequent time indices $t \geq i$. Temporal credit assignment enables propagating the penalty back to the unsafe action.

3.3 Monitor Synthesis

Function CONSTRUCT for constructing a reward monitor $\mathcal{A}_\varphi$ is defined inductively on the LTL$_f[\mathcal{F}]$ formula φ, with the support of the memoization function SYNTH. As many sub-monitors are created recursively, storing them becomes useful to avoid repetitive computation, which is the purpose of Algorithm 1 SYNTH.

Let $\mathcal{L}_p(s)$ denote the value of the atomic proposition p in state s, where $s \in \mathcal{S}$ of the labelled MDP, the function CONSTRUCT works as follows:

- For $\varphi = \top$, the monitor is assigned states $Q \leftarrow \{q_0\}$, set of register-value pairs $\mathcal{V} \leftarrow \{(t_\top, 1)\}$, transition function $\delta_q \leftarrow \{(q_0, q_0)\}$, register update function $\delta_v \leftarrow \emptyset$, and reward function $\delta_r \leftarrow \mathcal{V}(t_\top)$.
- For $\varphi = p$ for $p \in \mathcal{P}$, the monitor is assigned states $Q \leftarrow \{q_0, q_1\}$, set of register-value pairs $\mathcal{V} \leftarrow \{(t_p, 0)\}$, transition function $\delta_q \leftarrow \{(q_0, q_1), (q_1, q_1)\}$, register update function $\delta_v(q_0, t_p) \leftarrow \mathcal{L}_p(s)$, and reward function $\delta_r \leftarrow \mathcal{V}(t_p)$.
- For negation $\varphi = \neg\psi$, let us create $\mathcal{A}^\psi$ from procedure SYNTH, where $Q \leftarrow Q^\psi$, $q_0 \leftarrow q_0^\psi$, $\mathcal{V} \leftarrow \mathcal{V}^\psi \cup \{(t_{\neg\psi}, 1 - \mathcal{V}^\psi(t_\psi))\}$, $\delta_q = \delta_q^\psi$, $\delta_v \leftarrow \delta_v^\psi$, and then $\delta_r \leftarrow \mathcal{V}(t_{\neg\psi})$. Finally, for all states $q \in Q$, assign $\delta_v(q, t_{\neg\psi}) = 1 - \mathcal{V}(t_\psi)$.
- For conjunction $\varphi = \varphi_1 \wedge \varphi_2$, let us create $\mathcal{A}^{\varphi_1}$ and $\mathcal{A}^{\varphi_2}$ from procedure SYNTH. Then we can take the Cartesian product of the two monitors, such that $Q = Q^{\varphi_1} \times Q^{\varphi_2} = \{(q^{\varphi_1}, q^{\varphi_2}) \mid q^{\varphi_1} \in Q^{\varphi_1}, q^{\varphi_2} \in Q^{\varphi_2}\}$, $q_0 \leftarrow (q_0^{\varphi_1}, q_0^{\varphi_2})$, and $\mathcal{V} \leftarrow \mathcal{V}^{\varphi_1} \cup \mathcal{V}^{\varphi_2} \cup \{(t_{\varphi_1 \wedge \varphi_2}, \min(\mathcal{V}(t_{\varphi_1}), \mathcal{V}(t_{\varphi_2})))\}$, and $\delta_q : Q^{\varphi_1} \times Q^{\varphi_2} \to Q^{\varphi_1} \times Q^{\varphi_2}$. Then, for all states $q \in Q$, we can assign $\delta_v \leftarrow \delta_v^{\varphi_1} \cup \delta_v^{\varphi_2}$ and $\delta_v(q, t_{\varphi_1 \wedge \varphi_2}) \leftarrow \min(\mathcal{V}(t_{\varphi_1}), \mathcal{V}(t_{\varphi_2}))$. Finally, let $\delta_r \leftarrow \mathcal{V}(t_{\varphi_1 \wedge \varphi_2})$.
- For temporal next $\varphi = X\psi$, the procedure is identical to an atomic proposition p, but the labelling of s' (as a result of taking action a in state s) is considered instead of s.
- For temporal until $\varphi = \varphi_1 \,\mathcal{U}\, \varphi_2$, let us start by obtaining $\mathcal{A}^{\varphi_1}$ and $\mathcal{A}^{\varphi_2}$ from procedure SYNTH. Let q_{-1} and q_{-2} denote the final state of a monitor and the penultimate state of a monitor respectively. We form a new monitor $\mathcal{A}^\varphi$ as follows, comprising a state space of $Q = Q^{\varphi_1} \times Q^{\varphi_2} = \{(q^{\varphi_1}, q^{\varphi_2}) \mid q^{\varphi_1} \in Q^{\varphi_1}, q^{\varphi_2} \in Q^{\varphi_2}\}$, and $q_0 \leftarrow (q_0^{\varphi_1}, q_0^{\varphi_2})$. Then,

we can state $\mathcal{V} \leftarrow \mathcal{V}^{\varphi_1} \cup \mathcal{V}^{\varphi_2} \cup \{(t_{\min \varphi_1}, 1)\} \cup \{(t_{\max \varphi_2}, 0)\} \cup \{(t_{\varphi_1 \mathcal{U} \varphi_2}, 0)\}$. Then, for all states $q \in Q$, we can assign $\delta_v(q, t_{\min \varphi_1}) \leftarrow \min(\mathcal{V}(t_{\min \varphi_1}), \mathcal{V}(t_{\varphi_1}))$, $\delta_v(q, t_{\max \varphi_2}) \leftarrow \max(\mathcal{V}(t_{\max \varphi_2}), \mathcal{V}(t_{\varphi_2}))$, and $\delta_v(q, t_{\varphi_1 \mathcal{U} \varphi_2}) \leftarrow \max(\mathcal{V}(t_{\varphi_1 \mathcal{U} \varphi_2}), \min(\mathcal{V}(t_{\min \varphi_1}), \mathcal{V}(t_{\max \varphi_2})))$. To support continuous updates, we assign $\delta_q(q_{-1}) \leftarrow q_{-2}$, $\delta_v(q_{-2}, t) \leftarrow \delta_v(q_{-1}, t)\ \forall t$, and all contained atomic registers are renamed (to avoid name clashing), and at initialisation are filled with the initial state labels $\mathcal{L}_p(\mathcal{S}_0)$, updatable with $\delta_v(q_0, t_p) \to \mathcal{L}_p(s')$. Finally, we can assign $\delta_r \leftarrow \mathcal{V}(t_{\varphi_1 \mathcal{U} \varphi_2})$.

- For temporal release $\varphi = \varphi_1 \mathcal{R} \varphi_2$, the same procedure as until $\mathcal{U}$ can be followed, substituting any instances of $\min$ with $\max$ and vice versa.

Monitors for temporal operators F (Eventually) and G (Always) can also be constructed primitively, with fewer registers as an implementation optimisation, only needing to track the max and min resp. of the associated subformula. Further details of optimisations and abbreviated connectives are provided in the extended paper's appendix.

Theorem 1. *The state and transition overhead of quantitative reward monitor construction from a* $\text{LTL}_f[\mathcal{F}]$ *formula* φ *is linear with respect to the size of* φ*.*

Theorem 1's proof is provided in the extended paper. Note that for LTL_f a monitor using alternating automata can also be constructed in linear time (De Giacomo, Vardi et al. 2015).

Algorithm 1: SYNTH. Monitor construction with memoization for an $\text{LTL}_f[\mathcal{F}]$ formula.

Input: $\text{LTL}_f[\mathcal{F}]$ formula φ, Cache
Output: A reward monitor $\mathcal{A} = (Q, q_0, \delta_q, \mathcal{V}, \delta_v, \delta_r)$
if φ ***is*** *in Cache* **then**
  return Cache[φ]
end
$\mathcal{A} \leftarrow$ CONSTRUCT(φ) `// Build inductively`
Cache[φ] $\leftarrow \mathcal{A}$
return $\mathcal{A}$

Lemma 2. *The procedure* SYNTH *returns a quantitative reward monitor for the provided* $\text{LTL}_f[\mathcal{F}]$ *formula.*

The proof of Lemma 2 appears in the extended version.

Theorem 3 (Correctness)**.** *Let* φ *be an* $\text{LTL}_f[\mathcal{F}]$ *formula,* $\mathcal{A}_\varphi$ *the QRM constructed from* φ *as per Alg. 1, and* λ *a finite trace of length* n*. Let* $\text{val}_{\mathcal{A}_\phi}(\varphi, i)$ *denote the value stored in the reward register at time index* i*. Then, for each index* $1 \leq i \leq n$*,*

$$\text{val}_{\mathcal{A}_\phi}(\varphi, i) = [\![\varphi, i]\!](\lambda)$$

We provide the details of the proof of Theorem 3 in the extended paper.

3.4 Composition and Learning

Learning involves taking the product between the quantitative monitor $\mathcal{A}$ and the MDP of the environment $\mathcal{M}$. We can define this product as the extended MDP $\mathcal{M} \otimes \mathcal{A}$ as follows:

Definition 7 (Extended MDP)**.** *Let* $\mathcal{M} = (\mathcal{S}, A, \Pr, \mathcal{R}, \gamma, \mathcal{L})$ *be a labelled MDP and* $\mathcal{A} = (Q, q_0, \delta_q, \mathcal{V}, \delta_v, \delta_r)$ *a quantitative reward monitor. Their synchronous product* $\mathcal{M} \otimes \mathcal{A} = (\mathcal{S}', A, \Pr', \mathcal{R}', \gamma, \mathcal{L})$ *is defined as follows, where* $q \in Q$*,* $a \in A$*, and* $s \in \mathcal{S}$*:*

- *State-space* $\mathcal{S}' \stackrel{def}{=} Q \times \mathcal{S}$
- *Transition kernel:* $\Pr'\big((q, s), a, (q', s')\big) \stackrel{def}{=}$
$$\begin{cases} \Pr(s, a, s') & \text{if } \delta_q\big((q, s)\big) = q', \\ 0 & \text{otherwise.} \end{cases}$$
- *Reward function:* $\mathcal{R}'((q, s), a, (q', s')) \stackrel{def}{=} \delta_r(\mathcal{V}' = \delta_v((q', s'), \mathcal{L}(s')), \rho)$
- *Remaining elements are inherited unchanged.*

The extended state-space including $q \in Q$ permits specifying non-Markovian (temporally extended) goals. However, a Markovian policy suffices when considering the product $\mathcal{M} \otimes \mathcal{A}$, as stated in Theorem 4.

Theorem 4. *A Markovian policy* π *in MDP* $\mathcal{M} \otimes \mathcal{A}$ *suffices to optimally capture the non-Markovian goals encoded by the monitor.*

A proof is provided in the extended version.

4 Experimental Evaluation

In this paper, we consider the notion of *reward convergence* with respect to the different reward mechanisms: Boolean monitors, quantitative monitors, and the pre-defined reward function, as provided in Definition 8, measuring reward stability to check convergence and agent performance (Dulac-Arnold et al. 2021; Machado et al. 2018; Brockman et al. 2016).

Definition 8 (Reward Convergence)**.** *Given episodic rewards* $\{\mathcal{R}_t\}_{t=1}^{T}$*, define the exponentially moving average* E_t *as*

$$E_t \stackrel{def}{=} \beta E_{t-1} + (1 - \beta)\mathcal{R}_t, \qquad \beta \stackrel{def}{=} 1 - \tfrac{2}{N+1},$$

with span N*. Define checkpoints every* N *steps:* $C_i \stackrel{def}{=} E_{iN}$*.*

An RL run converges (reward-wise) *if, given a tolerance* τ*, for the last* P *checkpoint pairs,*

$$|C_i - C_{i-1}| \leq \tau \quad \text{for all such } i.$$

To objectively measure learning performance, one method would be tracking *convergence*, although this can be hindered by the fact that an agent may still converge to a sub-optimal policy, particularly when using a non-informative reward function. Additionally, it is also the case that cumulative rewards themselves are not as useful as a metric when the quantitative monitor and handcrafted function (from the original environment implementation) are able to produce a reward at each time step (as contrasted to the Boolean monitor), and at arbitrary magnitudes of intermediate rewards.

Therefore, we use the notion of *task completion*, which is an evaluative (unobservable for the agent) *performance function* run at the terminal time point of the episode, and is universally suitable and uniform for all reward producers. This is similar to the performance function used in (Leike et al. 2017).

		Mean Episode Number			Mean Time (seconds)			Task Completion (%, mean ± 95% CI)		
		Base	**Boolean**	**Quant.**	**Base**	**Boolean**	**Quant.**	**Base**	**Boolean**	**Quant.**
Classic	Acrobot	**None**	600*	None	**None**	133.95*	None	**99.07%** ±0.09%	4.84% ±0.40%	94.77% ±0.30%
	Cartpole	**None**	None	None	**None**	None	None	**44.67%** ±0.42%	30.92% ±0.35%	33.34% ±0.37%
	Mountain Car	600*	600*	**None**	41.46*	42.98*	**None**	39.82% ±0.07%	37.04% ±0.06%	**42.28%** ±0.07%
	Pendulum	**None**	600*	None	**None**	181.17*	None	**74.39%** ±0.27%	45.03% ±0.18%	72.11% ±0.28%
Toy	Frozen Lake	216*	**45**	29*	0.04*	**0.03**	0.01*	58.86% ±0.10%	**62.16%** ±0.10%	58.96% ±0.10%
	Cliff Walking	**21**	25	2.3	**0.01**	0.04	0.003	**85.33%** ±0.05%	84.38% ±0.05%	84.95% ±0.05%
	Taxi	**11**	1301*	1221*	**0.01**	13.89*	4.26	**71.19%** ±0.07%	46.50% ±0.03%	52.76% ±0.03%
Box2D	Bipedal Walker	**None**	None	None	**None**	None	None	**16.78%** ±0.54%	7.43% ±0.15%	14.43% ±0.41%
	Lunar Lander	**None**	None	None	**None**	None	None	**57.45%** ±0.36%	43.44% ±0.17%	45.61% ±0.16%
Safety Gridworlds	Island Navigation	**1302**	71*	144*	**0.43**	0.05*	0.10*	**96.81%** ±0.02%	76.37% ±0.03%	76.72% ±0.03%
	Conveyor Belt	**191***	468*	518*	**0.15**	0.7*	0.7*	**60.96%** ±0.10%	27.1% ±0.09%	28.70% ±0.09%
	Sokoban	1129*	71*	**103**	0.47*	0.11*	**0.14**	50.23% ±0.01%	52.00% ±0.01%	**53.15%** ±0.02%

Table 1: Convergence data for all environments, split by base reward function, Boolean and quantitative monitors. Task completion entries are mean ± 95% confidence interval (half-width). (*) denotes suboptimal (comparatively) policy convergence.

Definition 9 (Task completion)**.** *A performance function which assigns a scalar in the range of* $[0, 1]$*, computed at the* terminal time step *(whether the episode ends by success, termination, or truncation). This signal is hidden from the agent during training, and is only used for evaluation.*

As for the scalars ρ in the specification-reward pair, we select them to be the same in the Boolean and quantitative monitors in cases where the formulas are equivalent. The full Boolean and quantitative specifications accompanying each environment are provided in the extended paper.

4.1 Environments

Environments in our experiments used the following groups:

- CLASSIC: low-dimensional continuous-control benchmarks based upon classical control theory problems (Sutton 1995; Barto, Sutton, and Anderson 1983; Moore 1990; Åström and Furuta 2000) that test stabilisation and swing-up behaviour.
- TOY: small grid-world domains with finite state-action spaces that permit exact dynamic-programming or tabular RL evaluation (Dietterich 2000; Brockman et al. 2016; Sutton, Barto et al. 1998), which illustrate path-planning under severe penalties as well as hierarchical goal decomposition.
- BOX2D: continuous-control problems with contact dynamics simulated in the Box2D engine (Brockman et al. 2016), requiring balance, locomotion, or soft landing.
- SAFETY GRIDWORLDS: a set of 2D gridworld environments provided by DeepMind (Leike et al. 2017), that evaluate the safety of RL agents against undesirable behaviours such as unsafe exploration, safe interruptibility, and irreversible side effects.

For each of these environments, we compare the manually specified reward function (e.g. provided by GYMNASIUM), a Boolean reward monitor, and a quantitative reward monitor. The complete descriptions and specification of all environments including all fluent definitions (both Boolean and quantitative) against the observation and action trajectories is detailed in the extended paper.

Specifying quantitative properties For each environment specified, we need to specify Boolean and quantitative properties for our labelled MDP. We also make use of a quantitative measure of task completion. Let us consider the environment of CARTPOLE in Example 1.

Example 1 (Cartpole's complete environmental specification)**.** *The relevant reward-specification pairs for* CARTPOLE *can be defined as the following for both the Boolean and quantitative settings:* $\big((F(G(reach\_goal)), 2),\ (G(balanced), 4)\big)$*, with the former expressing a persistence property and the latter a safety property. For the first specification, we want to ensure that we always remain in an intended goal position (not slipping backwards or forwards, here we can assume* $goal\_pos = 2$*). The second specification requires always remaining balanced, which is useful regardless of how far the agent is from the goal position.*

Below is the mathematical form of definitions for the labels used by the labelling function $\mathcal{L}(s)$ *of the labelled MDP, or in other words the quantitative properties:*

$$\textit{balanced} = \begin{cases} \frac{0.209 - |angle|}{0.209}, & \textit{if } |angle| \leq 0.209 \\ 0, & \textit{else} \end{cases}$$

$$\textit{reach\_goal} = \begin{cases} \frac{current\_pos}{goal\_pos}, & \textit{if } current\_pos > 0.01 \\ 0, & \textit{else} \end{cases}$$

In the case of Boolean atoms, $balanced$ *returns* $\top$ *iff* $angle \leq 0.209$*, else* $\bot$*. As for* $reach\_goal$*, it is true if the agent is within* 0.1 *of the* $goal\_pos$*. The task completion measure (performance function) is specified as an equally weighted mean of* $reach\_goal$ *and* $balanced$*.*

4.2 Reinforcement Learning Algorithms

The following section briefly describes the algorithms we use, including both tabular and policy gradient settings. *Tabular Q-learning* is used with TOY and SAFETY GRIDWORLDS environments, due to faring well with the relatively small discrete state-action spaces. *Proximal policy optimization* is used with CLASSIC and BOX2D environments to adequately handle continuous state and action spaces.

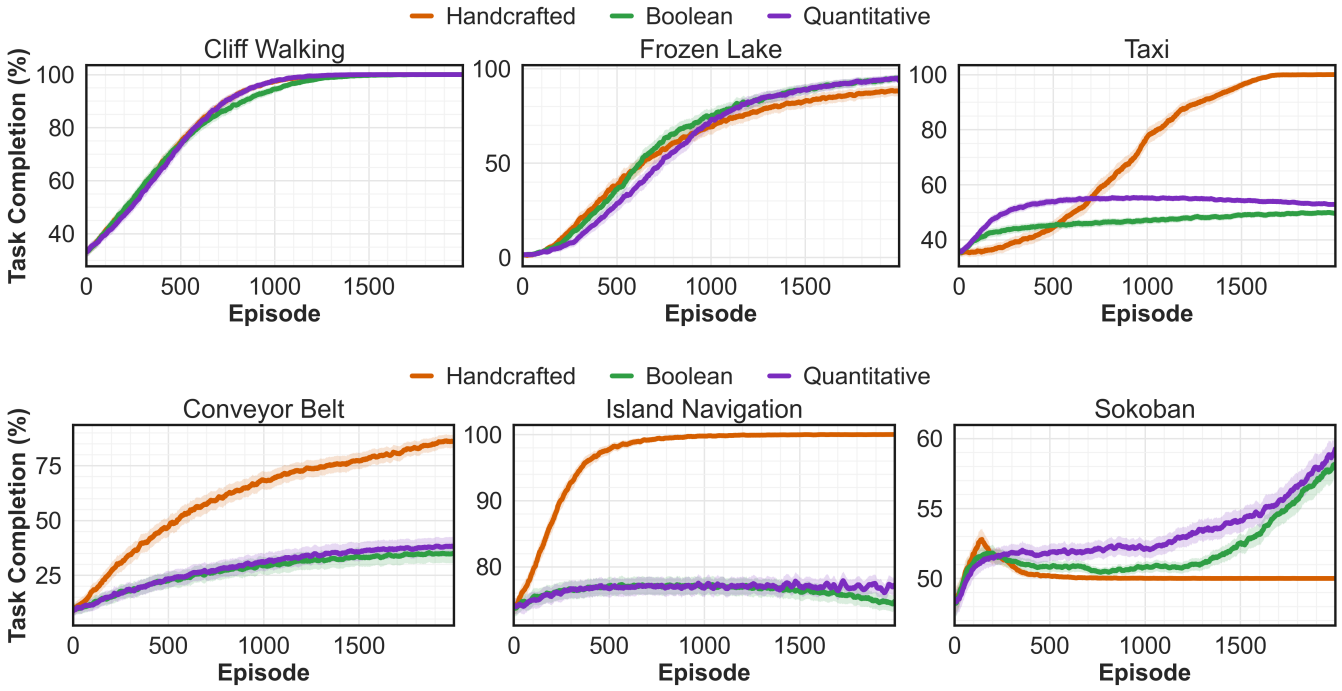


Figure 2: Task-completion percentages across SAFETY-GRIDWORLDS and TOY environments, ± 95% confidence interval.

Q-learning (Watkins and Dayan 1992) is a value-based method that iteratively updates the action-value function $Q(s, a)$ according to the Bellman optimality equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big),$$

where α is the learning rate, γ is the discount factor, r is the reward observed, (s, a) denotes the current state-action pair, and (s', a') denotes the next state-action pair.

Proximal policy optimization (PPO) (Schulman et al. 2017) is a policy-gradient method designed for continuous or high-dimensional state and action spaces. Instead of directly estimating action-values in a table, PPO maintains a parameterized policy $\pi_\theta(a|s)$ and optimizes it by maximizing a clipped surrogate objective:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right],$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the new and old policies, $\hat{A}_t$ is the advantage function estimate at time step t, and ϵ is a hyperparameter that bounds the ratio, helping to prevent destructive policy updates. Due to its on-policy nature, PPO is relatively stable.

4.3 Experimental Results

Table 1 presents our results in terms of reward convergence time and episode number, as well as our evaluation metric of quantitative task completion. The mean convergence time and episode number are taken from averaging a number of runs, with each run containing a fixed amount of training episode iterations. Likewise, the task completion metric is averaged over a number of runs, for each episode across the learning, where the terminal time step outputs the task completion progress. The number of episodes and runs used depends on the environment, where the number of episodes is shown in the graphs in Figure 2, and the remaining hyperparameters are specified in the extended paper.

Across environments, the quantitative monitor consistently matches or outperforms the Boolean monitor, and sometimes surpasses the manually specified reward function. Within simple and small gridworld-like environments, it was often simpler to use Boolean signals, rather than distance metrics such as Manhattan distance, as they do not take into account obstacles (thus encouraging exploring around non-optimal areas). Hence, the performance difference between Boolean and quantitative monitors is more pronounced in CLASSIC and BOX2D environments. We restricted ourselves to defining a quantitative specification using only the information known to the agent (observations and actions), which is not always the case where manually-specified reward functions are used. Our approach is easily extensible to incorporating such environmental signals by extending the scope of the labelling function. The learning performance gap of the quantitative and Boolean monitors is entirely determined by how well a quantitative measure can be defined per environment using the state trajectory, which is domain-dependent. However, in cases where writing a quantitative specification proves difficult, the performance should always be at least equal to using a Boolean monitor.

5 Related Work

Reward engineering is the practice of designing and tuning the reward structure to better align with the intended learning outcomes. Certain approaches exist specifically to counter *sparse* and *delayed* rewards, which remain compatible with QRMs and are useful for environments where feedback can only be provided towards the end of an episode. *Reward redistribution* techniques such as Align-RUDDER (Patil et al. 2022) shift terminal returns onto a few decisive actions, densifying feedback while preserving optimal policies. Causal variants (Zhang et al. 2023) relax alignment, but are susceptible to weak confounding. *Hindsight Experience Replay* (HER) (Andrychowicz et al. 2017) retroactively relabels failed roll-outs with achieved goals, generating synthetic dense rewards, though goal selection becomes intractable in large or partially observed spaces and may bias value estimates. *Reward shaping* augments each step-return with $F(s_t, s_{t+1}) = \gamma\Phi(s_{t+1}) - \Phi(s_t)$, guaranteeing policy invariance (Ng, Harada, and Russell 1999). Potentials Φ are either handcrafted (e.g., distance-to-goal) or learned (Grześ and Kudenko 2010), which must be informative yet low-variance.

Temporal logic rewards. A survey of reinforcement learning with temporal logic rewards is presented in (Liao 2020). In (Brafman, De Giacomo, and Patrizi 2018; De Giacomo et al. 2019), non-Markovian reward formulas expressed in LTL_f/LDL_f are translated into minimal DFA. The extended MDP is obtained as the synchronous product of the original MDP with these DFA components, guaranteeing minimality; because the DFA can be progressed symbol-by-symbol, the construction can also be carried out on-the-fly. The approach was refined in (De Giacomo et al. 2020), which merges the individual DFA into a single reward transducer, which can provide exponential savings. This compact transducer supports the four-valued runtime-monitoring semantics for LTL_f. Furthermore, our reward monitor can be constructed in linear time. Prior work has integrated GYM environments with reward monitors using the runtime monitoring language (Unniyankal et al. 2023; Hasanbeig, Kroening, and Abate 2020). Compared to these works, our approach has rewards that are densely provided with quantitative information, which is critical for addressing sparse rewards and sample efficiency.

Reward Machines (Icarte et al. 2022, 2018) represent non-Markovian rewards as Mealy machines whose edges are represented by modelled events. This makes them intuitive when crisp, symbolic events are available and supports automated potential-based shaping once the reward machine is given, although the effectiveness of the automated shaping is limited to simple monitors as the states and actions of the environment are not considered in the shaping, merely desired paths through the monitor provide further rewards. Recent works include extensions with counting automata (Bester et al. 2023), first order representations (Ardon et al. 2024), and learning of reward machines (Parac et al. 2024) for inverse RL.

Quantitative semantics. Li, Vasile, and Belta (2017) introduces quantitative semantics for robustness, but requires the whole trace to produce a robustness reward, therefore not being useful against sparse rewards and for long-horizon tasks. Balakrishnan and Deshmukh (2019) use quantitative Signal Temporal Logic (STL), for reward shaping, but assumes a bounded window length to keep complexity low and uses unnormalised atoms. Hamilton, Robinette, and Johnson (2022) outsources STL monitoring to an external tool making compositionality non-trivial, which is crucial for safety specifications and adequately handling non-Markovian properties, also again relying on unnormalised atoms. Jothimurugan, Alur, and Bastani (2019) use quantitative semantics, but the reward produced is not dense: a raw reward of $-\infty$ is given until the final monitor state, thus the method is based on reward shaping, and assumes bounds are supplied. There are scaling concerns, as each monitor state involves a separate policy head neural network.

Numeric reward machines are introduced by Levina et al. (2024) which retain Boolean transitions but introduce negative distance-to-goal formulas for providing denser rewards. However, quantitative temporal semantics are not used, nor is any synthesis algorithm provided to construct such machines from a formal specification.

Finally, all of the aforementioned papers overlook safety in the compositionality process, whereby safety properties are able to veto all rewards globally.

6 Conclusions and Future Work

In this paper, we have shown how to construct reward monitors using quantitative, linear time $\text{LTL}_f[\mathcal{F}]$ specifications, and then demonstrated the effectiveness of our approach against Boolean monitors and handcrafted reward functions, across various environments with differing reward characteristics. We empirically showed that our quantitative monitor surpasses or matches the performance of a Boolean monitor, and in some cases, surpasses the performance of the manually-specified reward function. Learning uses the product between the environment MDP and the composite reward monitor formed from all specification-reward pairs. Additionally, we syntactically identify which monitors target safety properties, which when violated, have the power to block rewards received from all other monitors indefinitely.

For future work, it would be interesting to consider discounted rewards which could form part of the specification tuple, or be used more directly through discounted LTL (Alur et al. 2023). Quantitative temporal operators could also be incorporated, such as in (Frigeri, Pasquale, and Spoletini 2012; Mu, Liu, and Li 2025), where F_t denotes eventually in the next t instants ("within"), similarly for always within the next t instants, as well as other more complex operators such as "almost always" and "soon", "nearly always", "gradually", which use fuzzy semantics, and may assist in learning convergence. Another aspect that could be investigated is robustness specifications (Anevlavis et al. 2022), which could help with other aspects such as curriculum learning (i.e., being confident that an agent has learnt a principal subtask prior to advancing to a more difficult task). Our approach would also be suitable for the multi-agent setting, where $\text{ATL}_f[\mathcal{F}]$ (Ferrando et al. 2024) could be used.

Acknowledgments

The research described in this paper was partially supported by the EPSRC (grant number EP/X015823/1) and the UKRI Centre for Doctoral Training in Safe and Trusted AI (grant number EP/S0233356/1). We also thank Jialue Xu and Alexander Philipp Rader for contributing some of the original code for the reward monitors.

References

Alur, R.; Bastani, O.; Jothimurugan, K.; Perez, M.; Somenzi, F.; and Trivedi, A. 2023. Policy synthesis and reinforcement learning for discounted LTL. In *International Conference on Computer Aided Verification*, 415–435. Springer.

Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Pieter Abbeel, O.; and Zaremba, W. 2017. Hindsight experience replay. *Advances in neural information processing systems*, 30.

Anevlavis, T.; Philippe, M.; Neider, D.; and Tabuada, P. 2022. Being correct is not enough: efficient verification using robust linear temporal logic. *ACM Transactions on Computational Logic (TOCL)*, 23(2): 1–39.

Ardon, L.; Furelos-Blanco, D.; Parac, R.; and Russo, A. 2024. FORM: Learning Expressive and Transferable First-Order Logic Reward Machines. *arXiv preprint arXiv:2501.00364*.

Åström, K. J.; and Furuta, K. 2000. Swinging up a pendulum by energy control. *Automatica*, 36(2): 287–295.

Baier, C.; and Katoen, J.-P. 2008. *Principles of model checking*. MIT press.

Balakrishnan, A.; and Deshmukh, J. V. 2019. Structured reward shaping using signal temporal logic specifications. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3481–3486. IEEE.

Barto, A. G.; Sutton, R. S.; and Anderson, C. W. 1983. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5): 834–846.

Bester, T.; Rosman, B.; James, S.; and Tasse, G. N. 2023. Counting Reward Automata: Sample Efficient Reinforcement Learning Through The Exploitation of Reward Function Structure. *arXiv preprint arXiv:2312.11364*.

Brafman, R.; De Giacomo, G.; and Patrizi, F. 2018. LTLf/LDLf non-markovian rewards. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. *arXiv preprint arXiv:1606.01540*.

Christian, B. 2021. *The alignment problem: How can machines learn human values?* Atlantic Books.

De Giacomo, G.; Favorito, M.; Iocchi, L.; Patrizi, F.; Ronca, A.; et al. 2020. Temporal logic monitoring rewards via transducers. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning*, 860–870.

De Giacomo, G.; Iocchi, L.; Favorito, M.; and Patrizi, F. 2019. Foundations for restraining bolts: Reinforcement learning with LTLf/LDLf restraining specifications. In *Proceedings of the international conference on automated planning and scheduling*, volume 29, 128–136.

De Giacomo, G.; Vardi, M. Y.; et al. 2013. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *Ijcai*, volume 13, 854–860.

De Giacomo, G.; Vardi, M. Y.; et al. 2015. Synthesis for LTL and LDL on finite traces. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015*, 1558–1564. AAAI Press.

Denison, C.; MacDiarmid, M.; Barez, F.; Duvenaud, D.; Kravec, S.; Marks, S.; Schiefer, N.; Soklaski, R.; Tamkin, A.; Kaplan, J.; et al. 2024. Sycophancy to subterfuge: Investigating reward-tampering in large language models. *arXiv preprint arXiv:2406.10162*.

Dietterich, T. G. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of artificial intelligence research*, 13: 227–303.

Dulac-Arnold, G.; Levine, N.; Mankowitz, D. J.; Li, J.; Paduraru, C.; Gowal, S.; and Hester, T. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9): 2419–2468.

Eschmann, J. 2021. Reward function design in reinforcement learning. *Reinforcement learning algorithms: Analysis and Applications*, 25–33.

Faella, M.; Legay, A.; and Stoelinga, M. I. A. 2008. Model checking quantitative linear time logic. In *6th Workshop on Quantitative Aspects of Programming Languages, QAPL 2008*, 61–77. Elsevier.

Ferrando, A.; Luongo, G.; Malvone, V.; and Murano, A. 2024. Theory and Practice of Quantitative ATL. In *International Conference on Principles and Practice of Multi-Agent Systems*, 231–247. Springer.

Festor, P.; Jia, Y.; Gordon, A. C.; Faisal, A. A.; Habli, I.; and Komorowski, M. 2022. Assuring the safety of AI-based clinical decision support systems: a case study of the AI Clinician for sepsis treatment. *BMJ health & care informatics*, 29(1): e100549.

Frigeri, A.; Pasquale, L.; and Spoletini, P. 2012. Fuzzy time in LTL. *arXiv preprint arXiv:1203.6278*.

Grześ, M.; and Kudenko, D. 2010. Online learning of shaping rewards in reinforcement learning. *Neural networks*, 23(4): 541–550.

Hamilton, N.; Robinette, P. K.; and Johnson, T. T. 2022. Training agents to satisfy timed and untimed signal temporal logic specifications with reinforcement learning. In *International Conference on Software Engineering and Formal Methods*, 190–206. Springer.

Hasanbeig, M.; Kroening, D.; and Abate, A. 2020. Deep reinforcement learning with temporal logics. In *Formal Modeling and Analysis of Timed Systems: 18th International Conference, FORMATS 2020, Vienna, Austria, September 1–3, 2020, Proceedings 18*, 1–22. Springer.

Icarte, R. T.; Klassen, T.; Valenzano, R.; and McIlraith, S. 2018. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, 2107–2116. PMLR.

Icarte, R. T.; Klassen, T. Q.; Valenzano, R.; and McIlraith, S. A. 2022. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73: 173–208.

Jothimurugan, K.; Alur, R.; and Bastani, O. 2019. A composable specification language for reinforcement learning tasks. *Advances in Neural Information Processing Systems*, 32.

Lamine, K. B.; and Kabanza, F. 2000. Using fuzzy temporal logic for monitoring behavior-based mobile robots. In *Proc. of IASTED Int. Conf. on Robotics and Applications*, 116–121.

Leike, J.; Martic, M.; Krakovna, V.; Ortega, P. A.; Everitt, T.; Lefrancq, A.; Orseau, L.; and Legg, S. 2017. AI safety gridworlds. *arXiv preprint arXiv:1711.09883*.

Levina, K.; Pappas, N.; Karapantelakis, A.; Feljan, A. V.; and Seipp, J. 2024. Numeric Reward Machines. *arXiv preprint arXiv:2404.19370*.

Li, X.; Serlin, Z.; Yang, G.; and Belta, C. 2019. A formal methods approach to interpretable reinforcement learning for robotic planning. *Science Robotics*, 4(37): eaay6276.

Li, X.; Vasile, C.-I.; and Belta, C. 2017. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3834–3839. IEEE.

Liao, H.-C. 2020. A survey of reinforcement learning with temporal logic rewards.

Machado, M. C.; Bellemare, M. G.; Talvitie, E.; Veness, J.; Hausknecht, M.; and Bowling, M. 2018. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61: 523–562.

Moore, A. W. 1990. Efficient memory-based learning for robot control. Technical report, University of Cambridge, Computer Laboratory.

Mu, C.; Liu, W.; and Li, Y. 2025. Checking with Fuzzy Temporal Logic Operators. In *Theoretical Computer Science: 42nd National Conference, NCTCS 2024, Qingdao, China, July 19–21, 2024, Revised Selected Papers*, volume 2354, 81. Springer Nature.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, 278–287. Citeseer.

Parac, R.; Nodari, L.; Ardon, L.; Furelos-Blanco, D.; Cerutti, F.; and Russo, A. 2024. Learning robust reward machines from noisy labels. *arXiv preprint arXiv:2408.14871*.

Patil, V.; Hofmarcher, M.; Dinu, M.-C.; Dorfer, M.; Blies, P. M.; Brandstetter, J.; Arjona-Medina, J.; and Hochreiter, S. 2022. Align-RUDDER: Learning From Few Demonstrations by Reward Redistribution. In *International Conference on Machine Learning*, 17531–17572. PMLR.

Sadigh, D.; Kim, E. S.; Coogan, S.; Sastry, S. S.; and Seshia, S. A. 2014. A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications. In *53rd IEEE Conference on Decision and Control*, 1091–1096. IEEE.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sistla, A. P. 1994. Safety, liveness and fairness in temporal logic. *Formal Aspects of Computing*, 6(5): 495–511.

Sutton, R. S. 1995. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, 8.

Sutton, R. S.; Barto, A. G.; et al. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Unniyankal, H.; Belardinelli, F.; Ferrando, A.; and Malvone, V. 2023. RMLGym: a Formal Reward Machine Framework for Reinforcement Learning. In *WOA*, 1–16.

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine learning*, 8: 279–292.

Zhang, Y.; Du, Y.; Huang, B.; Wang, Z.; Wang, J.; Fang, M.; and Pechenizkiy, M. 2023. Interpretable reward redistribution in reinforcement learning: A causal approach. *Advances in Neural Information Processing Systems*, 36: 20208–20229.