A Framework for Transforming Specifications in Reinforcement Learning

Rajeev Alur, Suguman Bansal, Osbert Bastani, and Kishor Jothimurugan

University of Pennsylvania

Abstract. Reactive synthesis algorithms allow automatic construction of policies to control an environment modeled as a Markov Decision Process (MDP) that are optimal with respect to high-level temporal logic specifications assuming the MDP model is known a priori. Reinforcement learning algorithms, in contrast, are designed to learn an optimal policy when the transition probabilities of the MDP are unknown, but require the user to associate local rewards with transitions. The appeal of high-level temporal logic specifications has motivated research to develop RL algorithms for synthesis of policies from specifications. To understand the techniques, and nuanced variations in their theoretical guarantees, in the growing body of resulting literature, we develop a formal framework for defining transformations among RL tasks with different forms of objectives. We define the notion of sampling-based reduction to relate two MDPs whose transition probabilities can be learnt by sampling, followed by formalization of preservation of optimal policies, convergence, and robustness. We then use our framework to restate known results, establish new results to fill in some gaps, and identify open problems.

Keywords: Reinforcement learning · Reactive synthesis · Temporal logic specifications.

1 Introduction

In reactive synthesis for probabilistic systems, the system is typically modeled as a (finite-state) Markov Decision Process (MDP), and the desired behavior of the system is given as a logical specification. For example, in robot motion planning, the model captures the physical environment in which the robot is operating and how the robot updates its position in response to the available control commands; and the logical requirement can specify that the robot should always avoid obstacles and eventually reach all of the specified targets. The synthesis algorithm then needs to compute a control policy that maximizes the probability that an infinite execution of the system under the policy satisfies the logical specification. There is a well developed theory of reactive synthesis for MDPs with respect to temporal logic specifications, accompanied by tools optimized with heuristics for improving scalability and practical applications (for instance, see [4] for a survey).

Reactive synthesis algorithms assume that the transition probabilities in the MDP modeling the environment are known a priori. In many practical settings, these probabilities are not known, and the model needs to be learnt by exploration or sampling-based simulation. Reinforcement learning (RL) has emerged to be an effective paradigm for synthesis of control policies in this scenario. The optimization criterion for policy synthesis for RL algorithms is typically specified by associating a local reward with each transition and aggregating the sequence of local rewards along an infinite execution using discounted-sum or limit-average operators. Since an RL algorithm is learning the model using sampling, it needs to compute approximations to the optimal policy with guaranteed convergence. Furthermore, ideally, the algorithm should give a PAC (Probably Approximately Correct) guarantee regarding the number of samples needed to ensure that the value of the policy it computes is within a specified bound of that of the optimal policy with a probability greater than a specified threshold. RL algorithms with convergence and efficient PAC guarantees are known for discounted-sum as well as limit-average rewards [20].

RL algorithms such as Q-learning [28] are increasingly used in robotic motion planning in real-world complex scenarios, and seem to work well in practice even when the assumptions necessary for theoretical guarantees do not hold. However, a key shortcoming is that the user must manually encode the desired behavior by associating rewards with system transitions. An appealing alternative is to instead have the user provide a high-level logical specification encoding the task. First, it is more natural to specify the desired properties of the global behavior, such as "always avoid obstacles and reach targets in this specified order", in a logical formalism such as temporal logic. Second, logical specifications facilitate testing and verifiability since it can be checked independently whether the synthesized policy satisfies the logical requirement. Finally, we can assume that the learning algorithm knows the logical specification in advance, unlike the local rewards learnt during model exploration, thereby opening up the possibility of design of specification-aware learning algorithms. For example, the structure of a temporal-logic formula can be exploited for hierarchical and compositional learning to reduce the sample complexity in practice [15, 18].

This has motivated many researchers to design RL algorithms for logical specifications [2, 6, 9, 13, 22, 12, 30, 11, 29, 16, 21, 14]. The natural approach is to (i) translate the logical specification to an automaton that accepts executions that satisfy the specification, (ii) define an MDP that is the product of the MDP being controlled and the specification automaton, (3) associate rewards with the transitions of the product MDP so that either discounted-sum or limit-average aggregation (roughly) captures acceptance by the automaton, and (4) apply an off-the-shelf RL algorithm such as Q-learning to synthesize the optimal policy. While this approach is typical to all the papers in this rapidly growing body of research, and many of the proposed techniques have been shown to work well empirically, there are many nuanced variations in terms of their theoretical guarantees. Instead of attempting to systematically survey the literature, let us make some general observations: some consider only finite executions with a known time horizon; some provide convergence guarantees only when the optimal policy satisfies the specification almost surely; some include a parameterized reduction—the parameter being the discount factor, for instance, establishing correctness for some value of the parameter without specifying how to compute it. The bottom line is that there are no known RL algorithms with convergence and/or PAC guarantees to synthesize a policy to maximize the satisfaction of a temporal logic specification (or an impossibility result that such algorithms cannot exist).

In this paper, we propose a formal framework for defining transformations among RL tasks. We define an RL task to consist of an MDP $\mathcal{M}$ together with a specification ϕ for the desired policy. The MDP is given by its states, actions, a function to reset the state to its initial state, and a step-function that allows sampling of its transitions. Possible forms of specifications include transition-based rewards to be aggregated as discounted sum or limit average, reward machines [14], safety, reachability, and linear temporal logic formulas. We then define *sampling-based reduction* to formalize transforming one RL task $(\mathcal{M}, \phi)$ to another $(\bar{\mathcal{M}}, \phi')$. While the relationship between the transformed model $\bar{\mathcal{M}}$ and the original model $\mathcal{M}$ is inspired by the classical definitions of simulation maps over (probabilistic) transition systems, the main challenge is that the transition probabilities cannot be directly defined in terms of the unknown transition probabilities of $\mathcal{M}$. Intuitively, the step-function to sample transitions of $\bar{\mathcal{M}}$ is definable in terms of the step-function of $\mathcal{M}$ used as a black-box, and our formalization allows this.

The notion of reduction among RL tasks naturally leads to formalization of preservation of optimal policies, convergence, and robustness (that is, policies close to optimal in one get mapped to ones close to optimal in the other). We use this framework to revisit existing results, fill in some gaps, and identify open problems.

2 Preliminaries

Markov Decision Process. A Markov Decision Process (MDP) is a tuple $\mathcal{M} = (S, A, s_0, P)$, where S is a finite set of states, s_0 is the initial state,[1] A is a finite set of actions, and $P : S \times A \times S \to [0, 1]$ is the transition probability function, with $\sum_{s' \in S} P(s, a, s') = 1$ for all $s \in S$ and $a \in A$.

An *infinite run* $\zeta \in (S \times A)^\omega$ is a sequence $\zeta = s_0 a_0 s_1 a_1 \ldots$, where $s_i \in S$ and $a_i \in A$ for all $i \in \mathbb{N}$. Similarly, a *finite run* $\zeta \in (S \times A)^* \times S$ is a finite sequence $\zeta = s_0 a_0 s_1 a_1 \ldots a_{t-1} s_t$. For any run ζ of length at least j and any $i \leq j$, we let $\zeta_{i:j}$ denote the subsequence $s_i a_i s_{i+1} a_{i+1} \ldots a_{j-1} s_j$. We use $\texttt{Runs}(S, A) = (S \times A)^\omega$ and $\texttt{Runs}_f(S, A) = (S \times A)^* \times S$ to denote the set of infinite and finite runs, respectively.

Let $\mathcal{D}(A) = \{\Delta : A \to [0, 1] \mid \sum_{a \in A} \Delta(a) = 1\}$ denote the set of all distributions over actions. A policy $\pi : \texttt{Runs}_f(S, A) \to \mathcal{D}(A)$ maps a finite run $\zeta \in \texttt{Runs}_f(S, A)$ to a distribution $\pi(\zeta)$ over actions. We denote by $\Pi(S, A)$ the set of all such policies. A policy π is *positional* if $\pi(\zeta) = \pi(\zeta')$ for all

[1] A distribution η over initial states can be modeled by adding a new state s_0 from which taking any action leads to a state sampled from η.

$\zeta, \zeta' \in \texttt{Runs}_f(S, A)$ with $\texttt{last}(\zeta) = \texttt{last}(\zeta')$ where $\texttt{last}(\zeta)$ denotes the last state in the run ζ. A policy π is deterministic if, for all finite runs $\zeta \in \texttt{Runs}_f(S, A)$, there is an action $a \in A$ with $\pi(\zeta)(a) = 1$.

Given a finite run $\zeta = s_0 a_0 \dots a_{t-1} s_t$, the *cylinder* of ζ, denoted by $\texttt{Cyl}(\zeta)$, is the set of all infinite runs starting with prefix ζ. Given an MDP $\mathcal{M}$ and a policy $\pi \in \Pi(S, A)$, we define the probability of the cylinder set by $\mathcal{D}_\pi^{\mathcal{M}}(\texttt{Cyl}(\zeta)) = \prod_{i=0}^{t-1} \pi(\zeta_{0:i})(a_i) P(s_i, a_i, s_{i+1})$. It is known that $\mathcal{D}_\pi^{\mathcal{M}}$ can be uniquely extended to a probability measure over the σ-algebra generated by all cylinder sets.

Simulator. In reinforcement learning, the standard assumption is that the set of states S, the set of actions A, and the initial state s_0 are known but the transition probability function P is unknown. The learning algorithm has access to a simulator $\mathbb{S}$ which can be used to sample runs of the system $\zeta \sim \mathcal{D}_\pi^{\mathcal{M}}$ using any policy π. The simulator can also be the real system, such as a robot, that $\mathcal{M}$ represents. Internally, the simulator stores the current state of the MDP which is denoted by $\mathbb{S}.\texttt{state}$. It makes the following functions available to the learning algorithm.

$\mathbb{S}.\texttt{reset()}$: This function sets $\mathbb{S}.\texttt{state}$ to the initial state s_0.

$\mathbb{S}.\texttt{step}(a)$: Given as input an action a, this function samples a state $s' \in S$ according to the transition probability function P—i.e., the probability that a state s' is sampled is $P(s, a, s')$ where $s = \mathbb{S}.\texttt{state}$. It then updates $\mathbb{S}.\texttt{state}$ to the newly sampled state s' and returns s'.

3 Task Specification

In this section, we present many different ways in which one can specify the reinforcement learning objective. We define a *reinforcement learning task* to be a pair $(\mathcal{M}, \phi)$ where $\mathcal{M}$ is an MDP and ϕ is a specification for $\mathcal{M}$. In general, a specification ϕ for $\mathcal{M} = (S, A, s_0, P)$ defines a function $J_\phi^{\mathcal{M}} : \Pi(S, A) \to \mathbb{R}$ and the reinforcement learning objective is to compute a policy π that maximizes $J_\phi^{\mathcal{M}}(\pi)$. Let $\mathcal{J}^*(\mathcal{M}, \phi) = \sup_\pi J_\phi^{\mathcal{M}}(\pi)$ denote the maximum value of $J_\phi^{\mathcal{M}}$. We let $\Pi_{\texttt{opt}}(\mathcal{M}, \phi)$ denote the set of all optimal policies in $\mathcal{M}$ w.r.t. ϕ—i.e., $\Pi_{\texttt{opt}}(\mathcal{M}, \phi) = \{\pi \mid J_\phi^{\mathcal{M}}(\pi) = \mathcal{J}^*(\mathcal{M}, \phi)\}$. In many cases, it is sufficient to compute an ε-optimal policy $\tilde{\pi}$ with $J_\phi^{\mathcal{M}}(\tilde{\pi}) \geq \mathcal{J}^*(\mathcal{M}, \phi) - \varepsilon$; we let $\Pi_{\texttt{opt}}^\varepsilon(\mathcal{M}, \phi)$ denote the set of all ε-optimal policies in $\mathcal{M}$ w.r.t. ϕ.

3.1 Rewards

The most common kind of specification used in reinforcement learning is reward functions that map transitions in $\mathcal{M}$ to real values. There are two main ways to define the objective function $J_\phi^{\mathcal{M}}$:

– *Discounted Sum.* The specification consists of a reward function $R : S \times A \times S \to \mathbb{R}$ and a discount factor $\gamma \in]0, 1[$—i.e., $\phi = (R, \gamma)$. The value of a policy

π is

$$J_\phi^{\mathcal{M}}(\pi) = \mathbb{E}_{\zeta \sim \mathcal{D}_\pi^{\mathcal{M}}}\Big[\sum_{i=0}^{\infty} \gamma^i R(s_i, a_i, s_{i+1})\Big],$$

where s_i and a_i denote the state and the action at the i^{th} step of ζ, respectively. Though less standard, one can use different discount factors in different states of $\mathcal{M}$, in which case we have $\gamma : S \to]0, 1[$ and

$$J_\phi^{\mathcal{M}}(\pi) = \mathbb{E}_{\zeta \sim \mathcal{D}_\pi^{\mathcal{M}}}\Big[\sum_{i=0}^{\infty} \Big(\prod_{j=0}^{i-1} \gamma(s_j)\Big) R(s_i, a_i, s_{i+1})\Big].$$

– *Limit Average.* Here $\phi = R : S \times A \times S \to \mathbb{R}$. The value of a policy π is

$$J_\phi^{\mathcal{M}}(\pi) = \liminf_{t \to \infty} \mathbb{E}_{\zeta \sim \mathcal{D}_\pi^{\mathcal{M}}}\Big[\frac{1}{t}\sum_{i=0}^{t-1} R(s_i, a_i, s_{i+1})\Big].$$

Reward Machines. Reward Machines [14] extend simple transition-based reward functions to history-dependent ones by using an automaton model. Formally, a reward machine for an MDP $\mathcal{M} = (S, A, s_0, P)$ is a tuple $\mathcal{R} = (U, u_0, \delta_u, \delta_r)$, where U is a finite set of states, u_0 is the initial state, $\delta_u : U \times S \to U$ is the state transition function, and $\delta_r : U \to [S \times A \times S \to \mathbb{R}]$ is the reward function. Given an infinite run $\zeta = s_0 a_0 s_1 a_1 \ldots$, we can construct an infinite sequence of reward machine states $\rho_{\mathcal{R}}(\zeta) = u_0 u_1, \ldots$ defined by $u_{i+1} = \delta_u(u_i, s_{i+1})$. Then, we can assign either a discounted sum or a limit average reward to ζ:

– *Discounted Sum.* Given a discount factor $\gamma \in]0, 1[$, the full specification is $\phi = (\mathcal{R}, \gamma)$ and we have

$$\mathcal{R}_\gamma(\zeta) = \sum_{i=0}^{\infty} \gamma^i \delta_r(u_i)(s_i, a_i, s_{i+1}).$$

The value of a policy π is $J_\phi^{\mathcal{M}}(\pi) = \mathbb{E}_{\zeta \sim \mathcal{D}_\pi^{\mathcal{M}}}[\mathcal{R}_\gamma(\zeta)]$. One could also use state-dependent discount factors in this case.

– *Limit Average.* The specification is just the reward machine $\phi = \mathcal{R}$. The t-step average reward of the run ζ is

$$\mathcal{R}_{\text{avg}}^t(\zeta) = \frac{1}{t}\sum_{i=0}^{t-1} \delta_r(u_i)(s_i, a_i, s_{i+1}).$$

The value of a policy π is $J_\phi^{\mathcal{M}}(\pi) = \liminf_{t \to \infty} \mathbb{E}_{\zeta \sim \mathcal{D}_\pi^{\mathcal{M}}}[\mathcal{R}_{\text{avg}}^t(\zeta)]$.

3.2 Abstract Specifications

The above specifications are defined w.r.t. a given set of states S and actions A, and can only be interpreted over MDPs with the same state and action spaces.

In this section, we look at *abstract specifications*, which are defined independent of S and A. To achieve this, a common assumption is that there is a fixed set of propositions $\mathcal{P}$, and the simulator provides access to a labeling function $L : S \to 2^{\mathcal{P}}$ denoting which propositions are true in any given state. Given a run $\zeta = s_0 a_0 s_1 a_1 \ldots$, we let $L(\zeta)$ denote the corresponding sequence of labels $L(\zeta) = L(s_0)L(s_1)\ldots$. A *labeled MDP* is a tuple $\mathcal{M} = (S, A, s_0, P, L)$. WLOG, we only consider labeled MDPs in the rest of the paper.

Abstract Reward Machines. Reward machines can be adapted to the abstract setting quite naturally. An *abstract reward machine* (ARM) is similar to a reward machine except δ_u and δ_r are independent of S and A—i.e., $\delta_u : U \times 2^{\mathcal{P}} \to U$ and $\delta_r : U \to [2^{\mathcal{P}} \to \mathbb{R}]$. Given current ARM state u_i and next MDP state s_{i+1}, the next ARM state is given by $u_{i+1} = \delta_u(u_i, L(s_{i+1}))$, and the reward is given by $\delta_r(u_i)(L(s_{i+1}))$.

Languages. Formal languages can be used to specify qualitative properties about runs of the system. A language specification $\phi = \mathcal{L} \subseteq (2^{\mathcal{P}})^\omega$ is a set of "desirable" sequences of labels. The value of a policy π is the probability of generating a sequence in $\mathcal{L}$—i.e.,

$$J_\phi^{\mathcal{M}}(\pi) = \mathcal{D}_\pi^{\mathcal{M}}\big(\{\zeta \in \texttt{Runs}(S, A) \mid L(\zeta) \in \mathcal{L}\}\big).$$

Some common ways to define languages are as follows.

- *Reachability.* Given an accepting set of propositions $X \in 2^{\mathcal{P}}$, we have $\mathcal{L}_{\texttt{reach}}(X) = \{w \in (2^{\mathcal{P}})^\omega \mid \exists i.\ w_i \cap X \neq \emptyset\}$.
- *Safety.* Given a safe set of propositions $X \in 2^{\mathcal{P}}$, we have $\mathcal{L}_{\texttt{safe}}(X) = \{w \in (2^{\mathcal{P}})^\omega \mid \forall i.\ w_i \subseteq X\}$.
- *Linear Temporal Logic.* Linear Temporal Logic [24] over propositions $\mathcal{P}$ is defined by the grammar

  $$\varphi := b \in \mathcal{P} \mid \varphi \vee \varphi \mid \neg\varphi \mid X\varphi \mid \varphi\, \mathcal{U}\, \varphi$$

  where X denotes the "Next" operator and $\mathcal{U}$ denotes the "Until" operator. We refer the reader to [25] for more details on the semantics of LTL specifications. We use $\Diamond$ and $\Box$ to denote the derived "Eventually" and "Always" operators, respectively. Given an LTL specification φ over propositions $\mathcal{P}$, we have $\mathcal{L}_{\text{LTL}}(\varphi) = \{w \in (2^{\mathcal{P}})^\omega \mid w \models \varphi\}$.

3.3 Learning Algorithms

A learning algorithm $\mathcal{A}$ is an iterative process that in each iteration (i) either resets the simulator or takes a step in $\mathcal{M}$, and (ii) outputs its current estimate of an optimal policy π. A learning algorithm $\mathcal{A}$ induces a random sequence of output policies $\{\pi_n\}_{n=1}^\infty$. We consider two common kinds of learning algorithms. First, we consider algorithms that converge in the limit almost surely.

Definition 1. *A learning algorithm $\mathcal{A}$ is said to converge in the limit for a class of specifications $\mathbb{C}$ if, for any RL task $(\mathcal{M}, \phi)$ with $\phi \in \mathbb{C}$,*

$$J_{\phi}^{\mathcal{M}}(\pi_n) \to \mathcal{J}^*(\mathcal{M}, \phi) \text{ as } n \to \infty \quad \text{almost surely}$$

Q-learning [28], which we briefly discuss below, is an example of a learning algorithm that converges in the limit for discounted sum rewards. There are variants of Q-learning for limit average rewards [1] which have been shown to converge in the limit under some assumptions on the MDP $\mathcal{M}$. The second kind of algorithms is *Probably Approximately Correct* (PAC-MDP) [19] algorithms which are defined as follows.

Definition 2. *A learning algorithm $\mathcal{A}$ is said to be PAC-MDP for a class of specifications $\mathbb{C}$ if, for any $p > 0$, $\varepsilon > 0$, and any RL task $(\mathcal{M}, \phi)$ with $\mathcal{M} = (S, A, s_0, P)$ and $\phi \in \mathbb{C}$, there is an $N = h(|S|, |A|, |\phi|, \frac{1}{p}, \frac{1}{\varepsilon})$ such that, with probability at least $1 - p$, we have*

$$\left|\left\{n \mid \pi_n \notin \Pi_{opt}^{\varepsilon}(\mathcal{M}, \phi)\right\}\right| \leq N.$$

We say a PAC-MDP algorithm is *efficient* if the function h is polynomial in $|S|, |A|, \frac{1}{p}, \frac{1}{\varepsilon}$. There are efficient PAC-MDP algorithms for both discounted rewards [20, 26] as well as limit average rewards [7, 20]. Below, we briefly describe one such algorithm, the E^3 algorithm from [20].

Q-learning. For discounted sum rewards ($\phi = (R, \gamma)$), a *q-function* $q^{\pi}(s, a)$ under a policy π is the expected future reward of taking an action a in a state s and then following the policy π. Any optimal policy $\pi^* \in \Pi_{\mathtt{opt}}(\mathcal{M}, \phi)$ is known [27] to satisfy the Bellman equation

$$q^{\pi^*}(s, a) = \sum_{s' \in S} P(s, a, s')\Big(R(s, a, s') + \gamma \cdot \max_{a' \in A} q^{\pi^*}(s', a')\Big)$$

for all states $s \in S$ and actions $a \in A$ (and vice-versa). Observe that, if q^{π^*} is known, a deterministic positional optimal policy can be obtained by choosing the action with the highest q^{π^*} value from each state. When the transition probabilities are known, q^{π^*} can be computed using value-iteration. In reinforcement learning, the transition probabilities are assumed to be unknown. In this case, *Q-learning* is used to estimate q^{π^*} [28]. First, the q-function is initialized randomly. We denote this q-function estimate as $\tilde{q}(s, a)$. In every iteration, the RL-agent observes the current state $s \in S$ and chooses an action $a \in A$ according to some exploratory policy. A commonly used exploratory policy is the ε-greedy policy, which selects a random action from A with probability ε, and $\arg\max_{a' \in A} \tilde{q}(s, a')$ with probability $1 - \varepsilon$. Upon taking a step from s to s' using a in the environment, $\tilde{q}(s, a)$ is updated using

$$\tilde{q}(s, a) \leftarrow (1 - \alpha) \cdot \tilde{q}(s, a) + \alpha \cdot \big(R(s, a, s') + \gamma \cdot \max_{a' \in A} \tilde{q}(s', a')\big)$$

where hyperparameter α is the *learning-rate*. Q-learning is guaranteed to converge to the optimal q-function q^{π^*} in the limit if every state-action pair is visited infinitely often. Q-learning is a *model-free* algorithm since it does not explicitly learn the transition probabilities of the MDP.

Explicit Explore and Exploit (E^3). The E^3 algorithm [20] estimates the MDP, i.e., estimates the transition probabilities explicitly. Once the MDP is estimated *sufficiently well*, the optimal policy in the estimated MDP is guaranteed to be a near-optimal policy in the original MDP with high probability. E^3 is a *model-based* algorithm since it explicitly learns the transition probabilities of the MDP.

To estimate the transition probabilities sufficiently well, the algorithm performs *balanced wandering* while exploring the MDP. Initially, all states are marked as *unknown*. If the current state is unknown, then the algorithm chooses an action that has been taken the fewest number of times, breaking ties randomly. A state becomes *known* if it has been visited sufficiently many times; intuitively, if a state is known, then with high probability, the algorithm has learned the probability distribution from the state with sufficient accuracy. The algorithm terminates exploration when all states are known. The central result of [20] is that the MDP can be estimated with an accuracy of $\mathcal{O}(\varepsilon)$ with $1-p$ confidence within a polynomial number of steps in $|S|, |A|, \frac{1}{p}, \frac{1}{\varepsilon}$. For discounted-sum rewards, the number of steps is also polynomial in $\frac{1}{1-\gamma}$. This gives rise to an efficient PAC-MDP learning algorithm for discounted sum and limit average rewards.

4 Reductions

There has been a lot of research on RL algorithms for reward-based specifications. The most common approach for language-based specifications is to transform the given specification into a reward function and apply algorithms that maximize the expected reward. In such cases, it is important to ensure that maximizing the expected reward corresponds to maximizing the probability of satisfying the specification. In this section, we study such reductions and formalize a general notion of a *sampling-based reduction* in the RL setting—i.e., the transition probabilities are unknown and only a simulator of $\mathcal{M}$ is available.

4.1 Specification Translations

We first consider the simplest form of reduction, which involves translating the given specification into another one. Given a specification ϕ for MDP $\mathcal{M} = (S, A, s_0, P, L)$ we want to construct another specification ϕ' such that for any $\pi \in \Pi_{\texttt{opt}}(\mathcal{M}, \phi')$, we also have $\pi \in \Pi_{\texttt{opt}}(\mathcal{M}, \phi)$. This ensures that ϕ' can be used to compute a policy that maximizes the objective of ϕ. Note that since the transition probabilities P are not known, the translation has to be independent of P and furthermore the above *optimality preservation* criterion must hold for all P.

Definition 3. *An optimality preserving specification translation is a computable function $\mathcal{F}$ that maps the tuple (S, A, s_0, L, ϕ) to a specification ϕ' such that for all transition probability functions P, letting $\mathcal{M} = (S, A, s_0, P, L)$, we have $\Pi_{opt}(\mathcal{M}, \phi') \subseteq \Pi_{opt}(\mathcal{M}, \phi)$.*

A first attempt at a reinforcement learning algorithm for language-based specifications is to translate the given specification to a reward machine (either discounted sum or limit average). However there are some limitations to this approach. First, we show that it is not possible to reduce reachability and safety objectives to reward machines with discounted rewards (using a single discount factor).

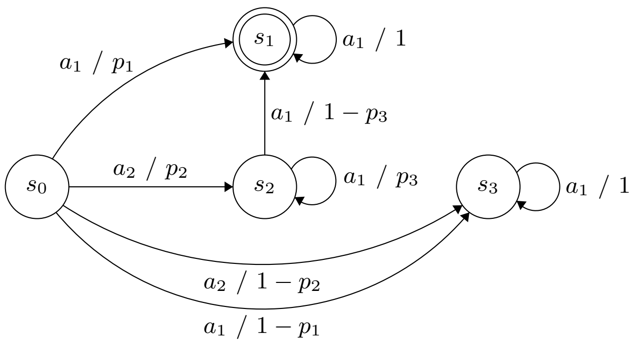

Fig. 1: Counterexample for reducing reachability specification to discounted sum rewards.

Theorem 1. *Let $\mathcal{P} = \{b\}$ and $\phi = \mathcal{L}_{reach}(\{b\})$. There exists S, A, s_0, L such that for any reward machine specification with a discount factor $\phi' = (\mathcal{R}, \gamma)$, there is a transition probability function P such that for $\mathcal{M} = (S, A, s_0, P, L)$, we have $\Pi_{opt}(\mathcal{M}, \phi') \not\subseteq \Pi_{opt}(\mathcal{M}, \phi)$.*

Proof. Consider the MDP depicted in Figure 1, which has states $S = \{s_0, s_1, s_2, s_3\}$, actions $A = \{a_1, a_2\}$, and labeling function L given by $L(s_1) = \{b\}$ (marked with double circles) and $L(s_0) = L(s_2) = L(s_3) = \emptyset$. Each edge denotes a state transition and is labeled by an action followed by the transition probability; the latter are parameterized by p_1, p_2, and p_3. At states s_1, s_2, and s_3 the only action available is a_1.[2] There are only two deterministic policies π_1 and π_2 in $\mathcal{M}$; π_1 always chooses a_1, whereas π_2 first chooses a_2 followed by a_1 afterwards.

For the sake of contradiction, suppose there is a $\phi' = (\mathcal{R}, \gamma)$ that preserves optimality w.r.t. ϕ for all values of p_1, p_2, and p_3. WLOG, we assume that

[2] This can be modeled by adding an additional dead state that is reached upon taking action a_2 in these states.

the rewards are normalized—i.e. $\delta_r : U \to [S \times A \times S \to [0, 1]]$. If $p_1 = p_2 = p_3 = 1$, then taking action a_1 in s_0 achieves reach probability of 1, whereas taking action a_2 in s_0 leads to a reach probability of 0. Hence, we must have that $\mathcal{R}_\gamma(s_0a_1(s_1a_1)^\omega) \geq \mathcal{R}_\gamma(s_0a_2(s_2a_1)^\omega) + \varepsilon$ for some $\varepsilon > 0$, as otherwise, π_2 maximizes $J^{\mathcal{M}}_{\phi'}$ but does not maximize $J^{\mathcal{M}}_{\phi}$.

For any finite run $\zeta \in \texttt{Runs}_f(S, A)$, let $\mathcal{R}_\gamma(\zeta)$ denote the finite discounted sum reward of ζ. Let t be such that $\frac{\gamma^t}{1-\gamma} \leq \frac{\varepsilon}{2}$. Then, for any $\zeta \in \texttt{Runs}(S, A)$, we have

$$\begin{aligned}\mathcal{R}_\gamma(s_0a_1(s_1a_1)^\omega) &\geq \mathcal{R}_\gamma(s_0a_2(s_2a_1)^\omega) + \varepsilon\\ &\geq \mathcal{R}_\gamma(s_0a_2(s_2a_1)^t s_2) + \varepsilon\\ &\geq \mathcal{R}_\gamma(s_0a_2(s_2a_1)^t s_2) + \frac{\gamma^t}{1-\gamma} + \frac{\varepsilon}{2}.\end{aligned}$$

Since $\lim_{p_3\to 1} p_3^t = 1$, there exists $p_3 < 1$ such that $1 - p_3^t \leq \frac{\varepsilon}{8}(1-\gamma)$. Let $p_1 < 1$ be such that $p_1 \cdot \mathcal{R}_\gamma(s_0a_1(s_1a_1)^\omega) \geq \mathcal{R}_\gamma(s_0a_2(s_2a_1)^t s_2) + \frac{\gamma^t}{1-\gamma} + \frac{\varepsilon}{4}$ and let $p_2 = 1$. Then, we have

$$\begin{aligned}J^{\mathcal{M}}_{\phi'}(\pi_1) &\geq p_1 \cdot \mathcal{R}_\gamma(s_0a_1(s_1a_1)^\omega)\\ &\geq \mathcal{R}_\gamma(s_0a_2(s_2a_1)^t s_2) + \frac{\gamma^t}{1-\gamma} + \frac{\varepsilon}{4}\\ &\geq p_3^t \cdot \Big(\mathcal{R}_\gamma(s_0a_2(s_2a_1)^t s_2) + \frac{\gamma^t}{1-\gamma}\Big) + \frac{\varepsilon}{4}\\ &\geq p_3^t \cdot \Big(\mathcal{R}_\gamma(s_0a_2(s_2a_1)^t s_2) + \frac{\gamma^t}{1-\gamma}\Big) + (1-p_3^t)\cdot\Big(\frac{1}{1-\gamma}\Big) + \frac{\varepsilon}{8}\\ &> J^{\mathcal{M}}_{\phi'}(\pi_2),\end{aligned}$$

where the last inequality followed from the fact that when using π_2, the system stays in state s_2 for at least t steps with probability p_3^t, and the reward of such trajectories is bounded from above by $\mathcal{R}_\gamma(s_0a_2(s_2a_1)^t s_2) + \frac{\gamma^t}{1-\gamma}$, along with the fact that the reward of all other trajectories is bounded by $\frac{1}{1-\gamma}$. This leads to a contradiction since π_1 maximizes $J^{\mathcal{M}}_{\phi'}$ but $J^{\mathcal{M}}_{\phi}(\pi_1) = p_1 < 1 = J^{\mathcal{M}}_{\phi}(\pi_2)$. □

Note that we did not use the fact that the reward machine is finite state; therefore, the above result applies to general non-Markovian reward functions of the form $R : \texttt{Runs}_f(S, A) \to [0, 1]$ with γ-discounted sum reward defined by $R_\gamma(\zeta) = \sum_{i=0}^{\infty} \gamma^i R(\zeta_{0:i})$. The proof can be easily modified to show the result for safety specifications as well.

The main issue in translating to discounted sum rewards is the fact that the rewards vanish over time and the overall reward only depends on the first few steps. This issue can be partly overcome by using limit average rewards. In fact, there exist abstract reward machines for reachability and safety specifications. A reward machine for the specification $\phi = \mathcal{L}_{\texttt{reach}}(\{b\})$ is shown in Figure 2. Each transition is labeled with a Boolean formula over $\mathcal{P}$ followed by the reward. It is

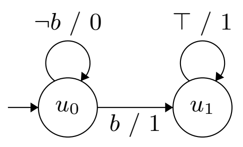

Fig. 2: ARM for $\phi = \mathcal{L}_{\texttt{reach}}(\{b\})$.

easy to see that for any MDP $\mathcal{M}$ and any policy π of $\mathcal{M}$, we have $J^{\mathcal{M}}_{\mathcal{R}}(\pi) = J^{\mathcal{M}}_{\phi}(\pi)$. The reward machine for $\mathcal{L}_{\texttt{safe}}(\mathcal{P} \setminus \{b\})$ is obtained by replacing the reward value r by $1-r$ on all transitions. However, there does not exist an ARM translation for the specification $\phi = \mathcal{L}_{\text{LTL}}(\Box\Diamond b)$, which requires the proposition b to be true infinitely often.

Theorem 2. *Let $\mathcal{P} = \{b\}$ and $\phi = \mathcal{L}_{\text{LTL}}(\Box\Diamond b)$. For any ARM specification $\phi' = \mathcal{R}$ with limit average rewards, there exists an MDP $\mathcal{M} = (S, A, s_0, P, L)$ such that $\Pi_{opt}(\mathcal{M}, \phi') \not\subseteq \Pi_{opt}(\mathcal{M}, \phi)$.*

Proof. For the sake of contradiction, let $\phi' = \mathcal{R} = (U, u_0, \delta_u, \delta_r)$ be an ARM that preserves optimality w.r.t ϕ for all MDPs. The extended state transition function $\delta_u : U \times (2^{\mathcal{P}})^* \to U$ is defined naturally. WLOG, we assume that all states in $\mathcal{R}$ are reachable from the initial state and that the rewards are normalized—i.e., $\delta_r : U \to [2^{\mathcal{P}} \to [0,1]]$.

A *cycle* in $\mathcal{R}$ is a sequence $C = u_1\ell_1u_2\ell_2\ldots\ell_ku_{k+1}$ where $u_i \in U$, $\ell_i \in 2^{\mathcal{P}}$, $u_{i+1} = \delta_u(u_i, \ell_i)$ for all i, $u_{k+1} = u_1$, and the states $u_1, \ldots, u_k$ are distinct. A cycle is *negative* if $\ell_i = \emptyset$ for all i, and *positive* if $\ell_i = \{b\}$ for all i. The average reward of a cycle C is given by $\mathcal{R}_{\text{avg}}(C) = \frac{1}{k}\sum_{i=1}^{k}\delta_r(u_i)(\ell_i)$. For any cycle $C = u_1\ell_1\ldots\ell_ku_{k+1}$ we can construct a deterministic MDP $\mathcal{M}_C$ with a single action that first generates a sequence of labels σ such that $\delta_u(u_0, \sigma) = u_1$, and then repeatedly generates the sequence of labels $\ell_1\ldots\ell_k$. The limit average reward of the only policy π in $\mathcal{M}_C$ is $J^{\mathcal{M}_C}_{\mathcal{R}}(\pi) = \mathcal{R}_{\text{avg}}(C)$ since the cycle C repeats indefinitely.

Now, given any positive cycle C_+ and any negative cycle C_-, we claim that $\mathcal{R}_{\text{avg}}(C_+) > \mathcal{R}_{\text{avg}}(C_-)$. To show this, consider an MDP $\mathcal{M}$ with two actions a_1 and a_2 such that taking action a_1 in the initial state s_0 leads to $\mathcal{M}_{C_+}$, and taking action a_2 in s_0 leads to $\mathcal{M}_{C_-}$. The policy π_1 that takes action a_1 in s_0 achieves a satisfaction probability of $J^{\mathcal{M}}_{\phi}(\pi_1) = 1$, whereas the policy π_2 taking action a_2 in s_0 achieves $J^{\mathcal{M}}_{\phi}(\pi_2) = 0$. Since $J^{\mathcal{M}}_{\mathcal{R}}(\pi_1) = \mathcal{R}_{\text{avg}}(C_+)$ and $J^{\mathcal{M}}_{\mathcal{R}}(\pi_2) = \mathcal{R}_{\text{avg}}(C_-)$, we must have that $\mathcal{R}_{\text{avg}}(C_+) > \mathcal{R}_{\text{avg}}(C_-)$ to preserve optimality w.r.t. ϕ. Since there are only finitely many cycles in $\mathcal{R}$, there exists an $\varepsilon > 0$ such that for any positive cycle C_+ and any negative cycle C_- we have $\mathcal{R}_{\text{avg}}(C_+) \geq \mathcal{R}_{\text{avg}}(C_-) + \varepsilon$.

Consider a bottom strongly connected component (SCC) of the graph of $\mathcal{R}$. We show that this component contains a negative cycle $C_- = u_1\ell_1\ldots\ell_ku_{k+1}$ along with a second cycle $C = u'_1\ell'_1\ldots\ell'_{k'}u'_{k'+1}$ such that $u'_1 = u_1$ and $\ell'_1 = \{b\}$. To construct C_-, from any state in the bottom SCC of $\mathcal{R}$, we can follow edges labeled $\ell = \emptyset$ until we repeat a state, and let this state be u_1. Then, to construct

C, from $u_1' = u_1$, we can follow the edge labeled $\ell_1' = \{b\}$ to reach $u_2' = \delta(u_1', \ell_1')$; since we are in the bottom SCC, there exists a path from u_2' back to u_1'. Now, consider a sequence of the form $C_m = C_-^m C$, where $m \in \mathbb{N}$. We have

$$\begin{aligned}\mathcal{R}_{\text{avg}}(C_m) &= \frac{mk\mathcal{R}_{\text{avg}}(C_-) + k'\mathcal{R}_{\text{avg}}(C)}{mk + k'} \\ &\leq \frac{mk}{mk + k'}\mathcal{R}_{\text{avg}}(C_-) + \frac{k'}{mk + k'} \\ &\leq \mathcal{R}_{\text{avg}}(C_-) + \frac{k'}{mk + k'}.\end{aligned}$$

Let m be such that $\frac{k'}{mk+k'} \leq \frac{\varepsilon}{2}$ and C_+ be any positive cycle. Then, we have $\mathcal{R}_{\text{avg}}(C_+) \geq \mathcal{R}_{\text{avg}}(C_m) + \frac{\varepsilon}{2}$; therefore, there exists $p < 1$ such that $p \cdot \mathcal{R}_{\text{avg}}(C_+) \geq \mathcal{R}_{\text{avg}}(C_m) + \frac{\varepsilon}{4}$. Now, we can construct an MDP $\mathcal{M}$ in which (i) taking action a_1 in initial state s_0 leads to $\mathcal{M}_{C_+}$ with probability p, and to a dead state (where b does not hold) with probability $1 - p$, and (ii) taking action a_2 in initial state s_0 leads to a deterministic single-action component that forces $\mathcal{R}$ to reach u_1 (recall that WLOG, all states in $\mathcal{R}$ are assumed to reachable from u_0), and then generates the sequence of labels in C_m indefinitely. Let π_1 and π_2 be policies that select a_1 and a_2 in s_0, respectively. Then, we have

$$J_{\mathcal{R}}^{\mathcal{M}}(\pi_1) \geq p \cdot \mathcal{R}_{\text{avg}}(C_+) \geq \mathcal{R}_{\text{avg}}(C_m) + \frac{\varepsilon}{4} = J_{\mathcal{R}}^{\mathcal{M}}(\pi_2) + \frac{\varepsilon}{4}.$$

However $J_{\phi}^{\mathcal{M}}(\pi_1) = p < 1 = J_{\phi}^{\mathcal{M}}(\pi_2)$, a contradiction. $\square$

Note that the result in the above theorem only claims non-existence of *abstract* reward machines for the LTL specification $\Box\Diamond b$, whereas Theorem 1 holds for arbitrary reward machines and history dependent reward functions.

4.2 Sampling-based Reduction

The previous section suggests that keeping the MDP $\mathcal{M}$ fixed might be insufficient for reducing LTL specifications to reward-based ones. In this section, we formalize the notion of a *sampling-based reduction* where we are allowed to modify the MDP $\mathcal{M}$ in a way that makes it possible to simulate the modified MDP $\bar{\mathcal{M}}$ using a simulator for $\mathcal{M}$ without the knowledge of the transition probabilities of $\mathcal{M}$.

Given an RL task $(\mathcal{M}, \phi)$ we want to construct another RL task $(\bar{\mathcal{M}}, \phi')$ and a function f that maps policies in $\bar{\mathcal{M}}$ to policies in $\mathcal{M}$ such that for any policy $\bar{\pi} \in \Pi_{\texttt{opt}}(\bar{\mathcal{M}}, \phi')$, we have $f(\bar{\pi}) \in \Pi_{\texttt{opt}}(\mathcal{M}, \phi)$. Since it should be possible to simulate $\bar{\mathcal{M}}$ without the knowledge of the transition probability function P of $\mathcal{M}$, we impose several constraints on $\bar{\mathcal{M}}$.

Let $\mathcal{M} = (S, A, s_0, P, L)$ and $\bar{\mathcal{M}} = (\bar{S}, \bar{A}, \bar{s}_0, \bar{P}, \bar{L})$. First, it must be the case that $\bar{S}$, $\bar{A}$, $\bar{s}_0$, $\bar{L}$ and f are independent of P. Second, since the simulator of $\bar{\mathcal{M}}$ uses the simulator of $\mathcal{M}$ we can assume that at any time, the state of the simulator of $\bar{\mathcal{M}}$ includes the state of the simulator of $\mathcal{M}$. Formally, there is a map $\beta : \bar{S} \to S$ such that for any $\bar{s}$, $\beta(\bar{s})$ is the state of $\mathcal{M}$ stored in $\bar{s}$. Since it is only possible to simulate $\mathcal{M}$ starting from s_0 we must have $\beta(\bar{s}_0) = s_0$. Next, when taking a step in $\bar{\mathcal{M}}$, a step in $\mathcal{M}$ may or may not occur, but the probability that a transition is sampled from $\mathcal{M}$ should be independent of P. Given these desired properties, we are ready to define a step-wise sampling-based reduction.

Definition 4. *A step-wise sampling-based reduction is a computable function $\mathcal{F}$ that maps the tuple (S, A, s_0, L, ϕ) to a tuple $(\bar{S}, \bar{A}, \bar{s}_0, \bar{L}, f, \beta, \alpha, q_1, q_2, \phi')$ where $f : \Pi(\bar{S}, \bar{A}) \to \Pi(S, A)$, $\beta : \bar{S} \to S$, $\alpha : \bar{S} \times \bar{A} \to \mathcal{D}(A)$, $q_1 : \bar{S} \times \bar{A} \times \bar{S} \to [0, 1]$, $q_2 : \bar{S} \times \bar{A} \times A \times \bar{S} \to [0, 1]$ and ϕ' is a specification such that*

- *$\beta(\bar{s}_0) = s_0$,*
- *$q_1(\bar{s}, \bar{a}, \bar{s}') = 0$ if $\beta(\bar{s}) \neq \beta(\bar{s}')$ and,*
- *for any $\bar{s} \in \bar{S}$, $\bar{a} \in \bar{A}$, $a \in A$, and $s' \in S$ we have*

$$\sum_{\bar{s}' \in \beta^{-1}(s')} q_2(\bar{s}, \bar{a}, a, \bar{s}') = 1 - \sum_{\bar{s}' \in \bar{S}} q_1(\bar{s}, \bar{a}, \bar{s}'). \tag{1}$$

For any transition probability function $P : S \times A \times S \to [0, 1]$, the new transition probability function $\bar{P} : \bar{S} \times \bar{A} \times \bar{S} \to [0, 1]$ is defined by

$$\bar{P}(\bar{s}, \bar{a}, \bar{s}') = q_1(\bar{s}, \bar{a}, \bar{s}') + \mathbb{E}_{a \sim \alpha(\bar{s}, \bar{a})}[q_2(\bar{s}, \bar{a}, a, \bar{s}') P(\beta(\bar{s}), a, \beta(\bar{s}'))]. \tag{2}$$

In Equation 2, $q_1(\bar{s}, \bar{a}, \bar{s}')$ denotes the probability with which $\bar{\mathcal{M}}$ steps to $\bar{s}'$ without sampling a transition from $\mathcal{M}$. In the event that a step in $\mathcal{M}$ does occur, $\alpha(\bar{s}, \bar{a})(a)$ gives the probability of the action a taken in $\mathcal{M}$ and $q_2(\bar{s}, \bar{a}, a, \bar{s}')$ is the (unnormalized) probability with which $\bar{\mathcal{M}}$ transitions to $\bar{s}'$ given that action a in $\mathcal{M}$ caused a transition to $\beta(\bar{s}')$. We now prove that, for any P, $\bar{P}$ defined in Equation 2 is a valid transition probability function.

Lemma 1. *Given a step-wise sampling-based reduction $\mathcal{F}$, for any MDP $\mathcal{M} = (S, A, s_0, P, L)$ and specification ϕ, the function $\bar{P}$ defined by $\mathcal{F}$ is a valid transition probability function.*

Proof. It is easy to see that $\bar{P}(\bar{s}, \bar{a}, \bar{s}') \geq 0$ for all $\bar{s}, \bar{s}' \in \bar{S}$ and $\bar{a} \in \bar{A}$. Now for any $\bar{s} \in \bar{S}$ and $\bar{a} \in \bar{A}$, letting $\sum_{\bar{s}'} q_1(\bar{s}, \bar{a}, \bar{s}') = p(\bar{s}, \bar{a})$, we have

$$\begin{aligned}
\sum_{\bar{s}' \in \bar{S}} P(\bar{s}, \bar{a}, \bar{s}') &= p(\bar{s}, \bar{a}) + \sum_{\bar{s}' \in \bar{S}} \mathbb{E}_{a \sim \alpha(\bar{s}, \bar{a})}[q_2(\bar{s}, \bar{a}, a, \bar{s}') P(\beta(\bar{s}), a, \beta(\bar{s}'))] \\
&= p(\bar{s}, \bar{a}) + \mathbb{E}_{a \sim \alpha(\bar{s}, \bar{a})}\Big[\sum_{\bar{s}' \in \bar{S}} q_2(\bar{s}, \bar{a}, a, \bar{s}') P(\beta(\bar{s}), a, \beta(\bar{s}'))\Big] \\
&= p(\bar{s}, \bar{a}) + \mathbb{E}_{a \sim \alpha(\bar{s}, \bar{a})}\Big[\sum_{s' \in S} \sum_{\bar{s}' \in \beta^{-1}(s')} q_2(\bar{s}, \bar{a}, a, \bar{s}') P(\beta(\bar{s}), a, \beta(\bar{s}'))\Big] \\
&= p(\bar{s}, \bar{a}) + \mathbb{E}_{a \sim \alpha(\bar{s}, \bar{a})}\Big[\sum_{s' \in S} P(\beta(\bar{s}), a, s') \sum_{\bar{s}' \in \beta^{-1}(s')} q_2(\bar{s}, \bar{a}, a, \bar{s}'))\Big] \\
&= p(\bar{s}, \bar{a}) + \mathbb{E}_{a \sim \alpha(\bar{s}, \bar{a})}\Big[\sum_{s' \in S} P(\beta(\bar{s}), a, s')(1 - p(\bar{s}, \bar{a}))\Big] \\
&= 1
\end{aligned}$$

where the penultimate step followed from Equation 1. □

Algorithm 1 Step function of the simulator $\bar{\mathbb{S}}$ of $\bar{\mathcal{M}}$ given β, α, q_1, q_2 and a simulator $\mathbb{S}$ of $\mathcal{M}$.

```
function $\bar{\mathbb{S}}$.step($\bar{a}$)
  $\bar{s} \leftarrow \bar{\mathbb{S}}$.state
  $p \leftarrow \sum_{\bar{s}'} q_1(\bar{s}, \bar{a}, \bar{s}')$
  $x \sim$ Uniform$(0, 1)$
  if $x \leq p$ then
    $\bar{\mathbb{S}}$.state $\leftarrow \bar{s}' \sim \dfrac{q_1(\bar{s}, \bar{a}, \bar{s}')}{p}$
  else
    $a \sim \alpha(\bar{s}, \bar{a})$
    $s' \leftarrow \mathbb{S}$.step($a$)
    $\bar{\mathbb{S}}$.state $\leftarrow \bar{s}' \sim \dfrac{q_2(\bar{s}, \bar{a}, a, \bar{s}')\mathbb{1}(\beta(\bar{s}') = s')}{1 - p}$        {Ensures $\beta(\bar{s}') = s'$}
  return $\bar{\mathbb{S}}$.state
```

Example 1. A simple example of a step-wise sampling-based reduction is the product construction used to translate reward machines to regular reward functions [14]. Let $\mathcal{R} = (U, u_0, \delta_u, \delta_r)$. Then, we have $\bar{S} = S \times U$, $\bar{A} = A$, $\bar{s}_0 = (s_0, u_0)$, $\bar{L}(s, u) = L(s)$, $\beta(s, u) = s$, $\alpha(a)(a') = \mathbb{1}(a' = a)$, $q_1 = 0$, and $q_2((s, u), a, a', (s', u')) = \mathbb{1}(u' = \delta_u(u, s'))$. The specification ϕ' is a reward function given by $R((s, u), a, (s', u')) = \delta_r(u)(s, a, s')$, and $f(\bar{\pi})$ is a policy that keeps track of the reward machine state and acts according to $\bar{\pi}$. □

Given an MDP $\mathcal{M} = (S, A, s_0, P, L)$ and a specification ϕ, the reduction $\mathcal{F}$ defines a unique triplet $(\bar{\mathcal{M}}, \phi', f)$ with $\bar{\mathcal{M}} = (\bar{S}, \bar{A}, \bar{s}_0, \bar{P}, \bar{L})$, where $\bar{S}, \bar{A}, \bar{s}_0, \bar{L}, f$ and ϕ' are obtained by applying $\mathcal{F}$ to (S, A, s_0, L, ϕ) and $\bar{P}$ is defined by Equation 2. We let $\mathcal{F}(\mathcal{M}, \phi)$ denote the triplet $(\bar{\mathcal{M}}, \phi', f)$. Given a simulator $\mathbb{S}$ of $\mathcal{M}$, we can construct a simulator $\bar{\mathbb{S}}$ of $\bar{\mathcal{M}}$ as follows.

$\bar{\mathbb{S}}$.`reset()`: This function internally sets the current state of the MDP to $\bar{s}_0$ and calls the reset function of $\mathcal{M}$.

$\bar{\mathbb{S}}$.`step(`$\bar{a}$`)`: This function is outlined in Algorithm 1. We use $\bar{s}' \sim \Delta(\bar{s}')$ to denote that $\bar{s}'$ is sampled from the distribution defined by Δ. It takes a step without calling $\mathbb{S}$.`step` with probability p. Otherwise, it samples an action a according to $\alpha(\bar{s}, \bar{a})$, calls $\mathbb{S}$.`step(`a`)` to get next state s' of $\mathcal{M}$ and then samples an $\bar{s}'$ satisfying $\beta(\bar{s}') = s'$ based on q_2. Equation 1 ensures that $\frac{q_2}{1-p}$ defines a valid distribution over $\beta^{-1}(s')$.

We call the reduction step-wise since at most one transition of $\mathcal{M}$ can occur during a transition of $\bar{\mathcal{M}}$. Under this assumption, we justify the general form of $\bar{P}$. Let $\bar{s}$ and $\bar{a}$ be fixed. Let $X_{\bar{S}}$ be a random variable denoting the next state in $\bar{\mathcal{M}}$ and X_A be a random variable denoting the action taking in $\mathcal{M}$ (it takes a dummy value $\perp \notin A$ when no step in $\mathcal{M}$ is taken). Then, for any $\bar{s}' \in \bar{S}$, we have

$$\Pr[X_{\bar{S}} = \bar{s}'] = \Pr[X_{\bar{S}} = \bar{s}' \wedge X_A = \perp] + \sum_{a \in A} \Pr[X_{\bar{S}} = \bar{s}' \wedge X_A = a].$$

Now, we have

$$\begin{aligned}&\Pr[X_{\bar{S}} = \bar{s}' \wedge X_A = a]\\&= \Pr[X_A = a] \Pr[X_{\bar{S}} = \bar{s}' \mid X_A = a]\\&= \Pr[X_A = a] \Pr[\beta(X_{\bar{S}}) = \beta(\bar{s}') \mid X_A = a] \Pr[X_{\bar{S}} = \bar{s}' \mid X_A = a, \beta(X_{\bar{S}}) = \beta(\bar{s}')]\\&= P(\beta(\bar{s}), a, \beta(\bar{s}')) \cdot \Pr[X_A = a] \Pr[X_{\bar{S}} = \bar{s}' \mid X_A = a, \beta(X_{\bar{S}}) = \beta(\bar{s}')].\end{aligned}$$

Taking $q_1(\bar{s}, \bar{a}, \bar{s}') = \Pr[X_{\bar{S}} = \bar{s}' \wedge X_A = \perp]$, $\alpha(\bar{s}, \bar{a})(a) = \Pr[X_A = a] / \Pr[X_A \neq \perp]$, and $q_2(\bar{s}, \bar{a}, a, \bar{s}') = \Pr[X_{\bar{S}} = \bar{s}' \mid X_A = a, \beta(X_{\bar{S}}) = \beta(\bar{s}')] \cdot \Pr[X_A \neq \perp]$, we obtain the form of $\bar{P}$ in Definition 4. Note that Equation 1 holds since both sides evaluate to $\Pr[X_A \neq \perp]$.

To be precise, it is also possible to reset the MDP $\mathcal{M}$ to s_0 in the middle of a run of $\bar{\mathcal{M}}$. This can be modeled by taking $\alpha(\bar{s}, \bar{a})$ to be a distribution over $A \times \{0, 1\}$, where $(a, 0)$ represents taking action a in the current state $\beta(\bar{s})$ and $(a, 1)$ represents taking action a in s_0 after a reset. We would also have $q_2 : \bar{S} \times \bar{A} \times A \times \{0, 1\} \times \bar{S} \to [0, 1]$ and furthermore $q_1(\bar{s}, \bar{a}, \bar{s}')$ can be nonzero if $\beta(\bar{s}') = s_0$. For simplicity, we use Definition 4 without considering resets in $\mathcal{M}$ during a step of $\bar{\mathcal{M}}$. However, the discussions in the rest of the paper apply to the general case as well. Now we define the *optimality preservation* criterion for sampling-based reductions.

Definition 5. *A step-wise sampling-based reduction $\mathcal{F}$ is optimality preserving if for any RL task $(\mathcal{M}, \phi)$ letting $(\bar{\mathcal{M}}, \phi', f) = \mathcal{F}(\mathcal{M}, \phi)$ we have $f(\Pi_{opt}(\bar{\mathcal{M}}, \phi')) \subseteq \Pi_{opt}(\mathcal{M}, \phi)$ where $f(\Pi) = \{f(\pi) \mid \pi \in \Pi\}$ for a set of policies Π.*

It is easy to see that the reduction in Example 1 is optimality preserving for both discounted sum and limit average rewards since $J^{\bar{\mathcal{M}}}_{\phi'}(\bar{\pi}) = J^{\mathcal{M}}_{\phi}(f(\bar{\pi}))$ for any policy $\bar{\pi} \in \Pi(\bar{S}, \bar{A})$. Another interesting observation is that we can reduce discounted sum rewards with multiple discount factors $\gamma : S \to]0, 1[$ to the usual case with a single discount factor.

Theorem 3. *There is an optimality preserving step-wise sampling-based reduction $\mathcal{F}$ such that for any $\mathcal{M} = (S, A, s_0, P)$ and $\phi = (R, \gamma)$, where $R : S \times A \times S \to \mathbb{R}$ and $\gamma : S \to]0, 1[$, we have $f(\mathcal{M}, \phi) = (\bar{\mathcal{M}}, \phi', f)$, where $\phi' = (R', \gamma')$, with $R' : \bar{S} \times \bar{A} \times \bar{S} \to \mathbb{R}$ and $\gamma' \in]0, 1[$.*

Proof. Let $\bar{S} = S \sqcup \{s_\perp\}$, where $s_\perp$ is a new sink state, $\bar{A} = A$, and $\bar{s}_0 = s_0$. We set $\gamma' = \gamma_{\max} = \max_{s \in S} \gamma(s)$, and define R' by $R'(s, a, s') = \frac{\gamma_{\max}}{\gamma(s)} R(s, a, s')$ if $s, s' \in S$ and 0 otherwise. We define $\bar{P}(s_\perp, a, s_\perp) = 1$ for all $a \in A$. For any $s \in S$, we have $\bar{P}(s, a, s') = \frac{\gamma(s)}{\gamma_{\max}} P(s, a, s')$ if $s' \in S$ and $\bar{P}(s, a, s_\perp) = 1 - \frac{\gamma(s)}{\gamma_{\max}}$. Intuitively, $\bar{\mathcal{M}}$ transitions to the sink state $s_\perp$ with probability $1 - \frac{\gamma(s)}{\gamma_{\max}}$ from any state s on taking any action a which has the effect of reducing the discount factor from $\gamma_{\max}$ to $\gamma(s)$ in state s since all future rewards are 0 after transitioning to $s_\perp$. Although we explicitly defined $\bar{P}$, note that it has the general form of Equation 2 and can be sampled from without knowing P. Now, we take $\phi' = (R', \gamma')$, and $f(\bar{\pi})$ to be $\bar{\pi}$ restricted to $\texttt{Runs}_f(S, A)$. It is easy to see that for any $\bar{\pi} \in \Pi(\bar{S}, A)$, we have $J^{\bar{\mathcal{M}}}_{\phi'}(\bar{\pi}) = J^{\mathcal{M}}_{\phi}(f(\bar{\pi}))$; therefore, this reduction preserves optimality. □

4.3 Reductions from Temporal Logic Specifications

A number of strategies have been recently proposed for learning policies from temporal specifications by reducing them to reward-based specifications. For instance, [2] proposes a reduction from Signal Temporal Logic (STL) specifications to rewards in the finite horizon setting—i.e., the specification φ is evaluated over a fixed T_φ-length prefix of the rollout ζ.

The authors of [12, 13] propose a reduction from LTL specifications to discounted rewards which proceeds by first constructing a product of the MDP $\mathcal{M}$ with a Limit Deterministic Büchi automaton (LDBA) $\mathcal{A}_\varphi$ derived from the LTL formula φ and then generates transition-based rewards in the product MDP. The strategy is to assign a fixed positive reward of r when an accepting state in $\mathcal{A}_\varphi$ is reached and 0 otherwise. As shown in [11], this strategy does not always preserve optimality if the discount factor γ is required to be strictly less that one. However, authors of [12] show that in certain cases, the discount factor can be taken to be $\gamma = 1$. Similar approaches are proposed in [30, 17, 8], though they do not provide optimality preservation guarantees.

A recent paper [11] presents a step-wise sampling-based reduction from LTL specifications to limit average rewards. It first constructs an LDBA $\mathcal{A}_\varphi$ from the LTL formula φ and then considers a product $\mathcal{M} \otimes \mathcal{A}_\varphi$ of the MDP $\mathcal{M}$ with $\mathcal{A}_\varphi$ in which the nondeterminism of $\mathcal{A}_\varphi$ is handled by adding additional actions that represent the choice of the possible transitions in $\mathcal{A}_\varphi$ that can be taken. Now, the reduced MDP $\bar{\mathcal{M}}$ is obtained by adding an additional sink state $\bar{s}_\perp$ with the property that whenever an accepting state in $\mathcal{A}_\varphi$ is reached in $\bar{\mathcal{M}}$, there is a $(1-\lambda)$ probability of transitioning to $\bar{s}_\perp$ during the next transition in $\bar{\mathcal{M}}$. They show that for a large enough value of λ, any policy maximizing the probability of reaching $\bar{s}_\perp$ in $\bar{\mathcal{M}}$ can be used to construct a policy that maximizes $J^{\mathcal{M}}_{\mathcal{L}_{\text{LTL}}(\varphi)}$. As shown before, this reachability property in $\bar{\mathcal{M}}$ can be translated to limit average rewards. The main drawback of this approach is that the lower bound on λ for preserving optimality depends on the transition probability function P; hence, it is not possible to correctly pick the value of λ without knowledge of P. A heuristic used in practice is to assign a default large value to λ. Their result can be summarized as follows.

Theorem 4 ([11]). *There is a family of step-wise sampling-based reductions $\{\mathcal{F}_\lambda\}_{\lambda\in]0,1[}$ such that for any MDP $\mathcal{M}$ and LTL specification $\phi = \mathcal{L}_{\text{LTL}}(\varphi)$, there exists a $\lambda_{\mathcal{M},\phi} \in]0,1[$ such that for all $\lambda \geq \lambda_{\mathcal{M},\phi}$, letting $(\bar{\mathcal{M}}_\lambda, \phi'_\lambda, f_\lambda) = \mathcal{F}_\lambda(\mathcal{M}, \phi)$, we have $f_\lambda(\Pi_{opt}(\bar{\mathcal{M}}_\lambda, \phi'_\lambda)) \subseteq \Pi_{opt}(\mathcal{M}, \phi)$ and $\phi'_\lambda = R_\lambda : S \times A \times S \to \mathbb{R}$ is a limit average reward specification.*

Another approach [5] with an optimality preservation guarantee reduces LTL specifications to discounted rewards with two discount factors $\gamma_1 < \gamma_2$ which are applied in different sets of states. This approach involves taking the product $\mathcal{M} \times \mathcal{A}_\varphi$ as $\bar{\mathcal{M}}$ and assigns a reward of $1 - \gamma_1$ to accepting states (where discount factor γ_1 is applied) and 0 elsewhere (discount factor γ_2 is applied). Applying Theorem 3 we get the following result as a corollary of the optimality preservation guarantee of this approach.

Theorem 5 ([5]). *There is a family of step-wise sampling-based reductions* $\{\mathcal{F}_\gamma\}_{\gamma \in]0,1[}$ *such that for any MDP* $\mathcal{M}$ *and LTL specification* $\phi = \mathcal{L}_{\mathrm{LTL}}(\varphi)$*, there exists* $\gamma_{\mathcal{M},\phi} \in]0,1[$ *such that for all* $\gamma \geq \gamma_{\mathcal{M},\phi}$*, letting* $(\bar{\mathcal{M}}_\gamma, \phi'_\gamma, f_\gamma) = \mathcal{F}_\gamma(\mathcal{M}, \phi)$*, we have* $f_\gamma(\Pi_{\mathit{opt}}(\bar{\mathcal{M}}_\gamma, \phi'_\gamma)) \subseteq \Pi_{\mathit{opt}}(\mathcal{M}, \phi)$ *and* $\phi'_\gamma = (R_\gamma, \gamma)$ *is a discounted sum reward specification.*

Similar to [11], the optimality preservation guarantee only applies to large enough γ, and the lower bound on γ depends on the transition probability function P.

To the best of our knowledge, it is unknown if there exists an optimality preserving step-wise sampling-based reduction from LTL specifications to reward-based specifications that is completely independent of P.

Open Problem 1 *Does there exist an optimality preserving step-wise sampling-based reduction* $\mathcal{F}$ *such that for any RL task* $(\mathcal{M}, \phi)$ *where* ϕ *is an LTL specification, letting* $(\bar{\mathcal{M}}, \phi', f) = \mathcal{F}(\mathcal{M}, \phi)$*, we have that* ϕ' *is a reward-based specification (either limit average or discounted sum)?*

5 Robustness

A key property of reward-based specifications that is exploited by RL algorithms is *robustness*. In this section, we discuss the concept of robustness for specifications as well as reductions. We show that robust reductions from LTL specifications to reward-based specifications are not possible because of the fact that LTL specifications are not robust.

5.1 Robust Specifications

A specification ϕ is said to be *robust* [23] if an optimal policy for ϕ in an estimate $\mathcal{M}'$ of the MDP $\mathcal{M}$ achieves close to optimal performance in $\mathcal{M}$. Formally, an MDP $\mathcal{M} = (S, A, s_0, P, L)$ is said to be δ*-close* to another MDP $\mathcal{M}' = (S, A, s_0, P', L)$ if their states, actions, initial state, and labeling function are identical and their transition probabilities differ by at most a δ amount—i.e.,

$$|P(s,a,s') - P'(s,a,s')| \leq \delta$$

for all $s, s' \in S$ and $a \in A$.

Definition 6. *A specification* ϕ *is* robust *if for any MDP* $\mathcal{M}$ *for which* ϕ *is a valid specification and* $\varepsilon > 0$*, there exists a* $\delta_{\mathcal{M},\varepsilon} > 0$ *such that if MDP* $\mathcal{M}'$ *is* $\delta_{\mathcal{M},\varepsilon}$*-close to* $\mathcal{M}$*, then an optimal policy in* $\mathcal{M}'$ *is an* ε*-optimal policy in* $\mathcal{M}$*—i.e.,* $\Pi_{\mathit{opt}}(\mathcal{M}', \phi) \subseteq \Pi^{\varepsilon}_{\mathit{opt}}(\mathcal{M}, \phi)$.

The simulation lemma in [20] proves that both discounted-sum and limit-average specifications are robust. On the other hand, [23] shows that language-based specifications, even safety specifications, are not robust. Here, we give a slightly modified example (that we will use later) to show that the specification $\phi = \mathcal{L}_{\mathtt{safe}}(\{b\})$ is not robust.

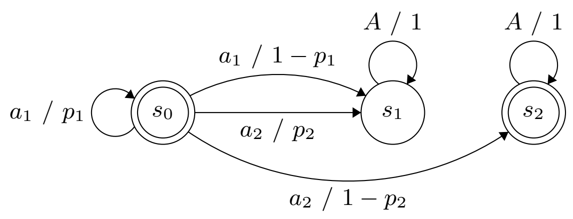

Fig. 3: Example showing non-robustness of $\mathcal{L}_{\texttt{safe}}(\{b\})$.

Theorem 6 ([23]). *There exists an MDP $\mathcal{M}$ and a safety specification ϕ such that, for any $\delta > 0$, there is an MDP $\mathcal{M}_\delta$ that is δ-close to $\mathcal{M}$ which satisfies $\Pi_{opt}(\mathcal{M}_\delta, \phi) \cap \Pi^{\varepsilon}_{opt}(\mathcal{M}, \phi) = \emptyset$ for all $\varepsilon < 1$.*

Proof. Consider the MDP $\mathcal{M}$ in Figure 3 with $p_1 = p_2 = 1$; the double circles denote states where b holds. Then, an optimal policy for $\phi = \mathcal{L}_{\texttt{safe}}(\{b\})$ always selects action a_1 and achieves a satisfaction probability of 1. Now let $\mathcal{M}_\delta$ denote the same MDP with $p_1 = p_2 = 1 - \delta$. Then, any optimal policy for ϕ in $\mathcal{M}_\delta$ must select a_2 almost surely, which is not optimal for $\mathcal{M}$. In fact, such a policy achieves a satisfaction probability of 0 in $\mathcal{M}$. Therefore, we have $\Pi_{\texttt{opt}}(\mathcal{M}_\delta, \phi) \cap \Pi^{\varepsilon}_{\texttt{opt}}(\mathcal{M}, \phi) = \emptyset$ for any $\delta > 0$ and any $\varepsilon < 1$. □

5.2 Robust Reductions

In our discussion of reductions, we were interested in optimality preserving sampling-based reductions mapping an RL task $(\mathcal{M}, \phi)$ to another task $(\bar{\mathcal{M}}, \phi')$. However, in the learning setting, if we use a PAC-MDP algorithm to compute a policy $\bar{\pi}$ for $(\bar{\mathcal{M}}, \phi')$, it might be the case that $\bar{\pi} \notin \Pi_{\texttt{opt}}(\bar{\mathcal{M}}, \phi')$. Therefore, we cannot conclude anything useful about the optimality of the corresponding policy $f(\bar{\pi})$ in $\mathcal{M}$ w.r.t. ϕ. Ideally, we would like to ensure that for any $\varepsilon > 0$ there is a $\varepsilon' > 0$ such that an ε'-optimal policy for $(\bar{\mathcal{M}}, \phi')$ is a corresponding ε-optimal policy for $(\mathcal{M}, \phi)$.

Definition 7. *A step-wise sampling-based reduction $\mathcal{F}$ is robust if for any RL task $(\mathcal{M}, \phi)$ with $(\bar{\mathcal{M}}, \phi', f) = \mathcal{F}(\mathcal{M}, \phi)$ and any $\varepsilon > 0$, there is an $\varepsilon' > 0$ such that $f(\Pi^{\varepsilon'}_{opt}(\bar{\mathcal{M}}, \phi')) \subseteq \Pi^{\varepsilon}_{opt}(\mathcal{M}, \phi)$.*

Observe that for any optimal policy $\bar{\pi} \in \Pi_{\texttt{opt}}(\bar{\mathcal{M}}, \phi')$ for $\bar{\mathcal{M}}$ and ϕ', we have $f(\bar{\pi}) \in \bigcap_{\varepsilon > 0} \Pi^{\varepsilon}_{\texttt{opt}}(\mathcal{M}, \phi) = \Pi_{\texttt{opt}}(\mathcal{M}, \phi)$; hence, a robust reduction is also optimality preserving. Although a robust reduction is preferred when translating LTL specifications to reward-based ones, the following theorem shows that such a reduction is not possible.

Theorem 7. *Let $\mathcal{P} = \{b\}$ and $\phi = \mathcal{L}_{safe}(\{b\})$. Then, there does not exist a robust step-wise sampling-based reduction $\mathcal{F}$ with the property that for any given $\mathcal{M}$, if $(\bar{\mathcal{M}}, \phi', f) = \mathcal{F}(\mathcal{M}, \phi)$, then ϕ' is a robust specification and $\Pi_{opt}(\bar{\mathcal{M}}, \phi') \neq \emptyset$.*

Proof. Consider the MDP $\mathcal{M} = (S, A, s_0, P, L)$ in Figure 3 with $p_1 = p_2 = 1$, and consider any $\varepsilon < 1$. From Theorem 6, we know that for any $\delta > 0$ there is an MDP $\mathcal{M}_\delta = (S, A, s_0, P_\delta, L)$ that is δ-close to $\mathcal{M}$ such that $\Pi_{\mathtt{opt}}(\mathcal{M}_\delta, \phi) \cap \Pi^{\varepsilon}_{\mathtt{opt}}(\mathcal{M}, \phi) = \emptyset$. For the sake of contradiction, suppose that such a reduction exists. Then, since $\mathcal{M}$ and $\mathcal{M}_\delta$ represent the same input (S, A, s_0, L, ϕ), the reduction outputs the same tuple $(\bar{S}, \bar{A}, \bar{s}_0, \bar{L}, f, \beta, \alpha, q_1, q_2, \phi')$ in both cases. Furthermore, from Equation 2 it follows, that the new transition probability functions $\bar{P}$ and $\bar{P}_\delta$ corresponding to P and P_δ differ by at most a δ amount—i.e., $|\bar{P}(\bar{s}, \bar{a}, \bar{s}') - \bar{P}_\delta(\bar{s}, \bar{a}, \bar{s}')| \leq \delta$ for all $\bar{s}, \bar{s}' \in \bar{S}$ and $\bar{a} \in \bar{A}$. Let $\bar{\mathcal{M}}$ and $\bar{\mathcal{M}}_\delta$ be the MDPs corresponding to $\bar{P}$ and $\bar{P}_\delta$.

Let $\varepsilon' > 0$ be such that $f(\Pi^{\varepsilon'}_{\mathtt{opt}}(\bar{\mathcal{M}}, \phi')) \subseteq \Pi^{\varepsilon}_{\mathtt{opt}}(\mathcal{M}, \phi)$. Since the specification ϕ' is robust, there is a $\delta = \delta_{\bar{\mathcal{M}}, \varepsilon'} > 0$ such that $\Pi_{\mathtt{opt}}(\bar{\mathcal{M}}_\delta, \phi') \subseteq \Pi^{\varepsilon'}_{\mathtt{opt}}(\bar{\mathcal{M}}, \phi')$. Let $\bar{\pi} \in \Pi_{\mathtt{opt}}(\bar{\mathcal{M}}_\delta, \phi')$ be an optimal policy for $\bar{\mathcal{M}}_\delta$ w.r.t. ϕ'. Now, since the reduction is optimality preserving, we have $f(\bar{\pi}) \in \Pi_{\mathtt{opt}}(\mathcal{M}_\delta, \phi)$. But then, we also have $f(\bar{\pi}) \in \Pi^{\varepsilon}_{\mathtt{opt}}(\mathcal{M}, \phi)$, which contradicts our assumption on $\mathcal{M}_\delta$. □

We observe that the above result holds when the reduction is only allowed to take at most one step in $\mathcal{M}$ during a step in $\bar{\mathcal{M}}$ (and can be generalized to a bounded number of steps). This leads to the following open problem.

Open Problem 2 *Does there exist a robust sampling-based reduction $\mathcal{F}$ such that for any RL task $(\mathcal{M}, \phi)$, where ϕ is an LTL specification, letting $(\bar{\mathcal{M}}, \phi', f) = \mathcal{F}(\mathcal{M}, \phi)$, we have that ϕ' is a reward-based specification (allowing $\bar{\mathcal{M}}$ to take unbounded number of steps in $\mathcal{M}$ per transition)?*

Note that even if such a reduction is possible, simulating $\bar{\mathcal{M}}$ would computationally hard since there might be no bound on the time it takes for a step in $\bar{\mathcal{M}}$ to occur.

6 Reinforcement Learning from LTL Specifications

We formalized a notion of sampling-based reduction for MDPs with unknown transition probabilities. Although reducing LTL specifications to reward-based ones is a natural approach towards obtaining learning algorithms for LTL specifications, we showed that step-wise sampling-based reductions are insufficient to obtain learning algorithms with guarantees. This leads us to the natural question of whether it is possible to design learning algorithms for LTL specifications with guarantees. Unfortunately, it turns out that it is not possible to obtain PAC-MDP algorithms for safety specifications.

Theorem 8. *There does not exist a PAC-MDP algorithm for the class of safety specifications.*

Theorem 7 shows that it is not possible to obtain a PAC-MDP algorithm for safety specifications by simply applying a step-wise sampling-based reduction followed by a PAC-MDP algorithm for reward-based specifications. Also, Theorem 7 does not follow from Theorem 8 because, the definition of a robust reduction allows the maximum value of ε' that satisfies $f(\Pi^{\varepsilon'}_{\texttt{opt}}(\bar{\mathcal{M}}, \phi')) \subseteq \Pi^{\varepsilon}_{\texttt{opt}}(\bar{\mathcal{M}}, \phi)$ to depend on the transition probability function P of $\mathcal{M}$. However the sample complexity function h of a PAC algorithm (Definition 2) should be independent of P. We now give a proof of Theorem 8.

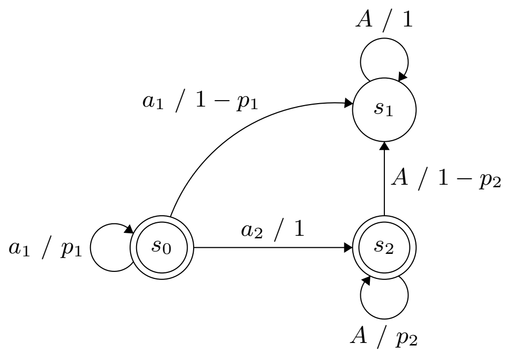

Fig. 4: A class of MDPs for showing no PAC-MDP algorithm exists for safety specifications.

Proof. Suppose there is a PAC-MDP algorithm $\mathcal{A}$ for the class of safety specifications. Consider $\mathcal{P} = \{b\}$ and the family of MDPs shown in Figure 4 where double circles denote states at which b holds. Let $\phi = \mathcal{L}_{\texttt{safe}}(\{b\})$ and $0 < \varepsilon < \frac{1}{2}$. For any $\delta > 0$, we use $\mathcal{M}^1_\delta$ to denote the MDP with $p_1 = 1$ and $p_2 = 1 - \delta$, and $\mathcal{M}^2_\delta$ to denote the MDP with $p_1 = 1 - \delta$ and $p_2 = 1$. Finally, let $\mathcal{M}$ denote the MDP with $p_1 = p_2 = 1$. We first show the following.

Lemma 2. *For any $\delta \in]0, 1[$, we have $\Pi^{\varepsilon}_{opt}(\mathcal{M}^1_\delta, \phi) \cap \Pi^{\varepsilon}_{opt}(\mathcal{M}^2_\delta, \phi) = \emptyset$.*

Proof. Suppose $\pi \in \Pi^{\varepsilon}_{\texttt{opt}}(\mathcal{M}^1_\delta, \phi)$ is an ε-optimal policy for $\mathcal{M}^1_\delta$ w.r.t. ϕ. Let $x_i = \pi((s_0 a_1)^i s_0)(a_1)$ denote the probability that π chooses a_1 after i self-loops in s_0. Then $J^{\mathcal{M}^1_\delta}_\phi(\pi) = \lim_{t\to\infty} \prod_{i=0}^{t} x_i$ since choosing a_2 in s_0 leads to eventual violation of the safety specification. The policy π^*_1 that always chooses a_1 achieves a value of $J^{\mathcal{M}^1_\delta}_\phi(\pi^*_1) = \mathcal{J}^*(\mathcal{M}^1_\delta, \phi) = 1$. Since $\pi \in \Pi^{\varepsilon}_{\texttt{opt}}(\mathcal{M}^1_\delta, \phi)$ we have $\lim_{t\to\infty} \prod_{i=0}^{t} x_i \geq 1 - \varepsilon$. Therefore $\prod_{i=0}^{t} x_i \geq 1 - \varepsilon$ for all $t \in \mathbb{N}$ since $z_t = \prod_{i=0}^{t} x_i$ is a non-increasing sequence.

Now let $E_t = \texttt{Cyl}((s_0 a_1)^t s_1)$ denote the set of all runs that reach s_1 after exactly t steps while staying in s_0 until then. We have

$$\mathcal{D}^{\mathcal{M}^2_\delta}_\pi(E_t) = (1 - p_1) p_1^{t-1} \prod_{i=0}^{t-1} x_i \geq \delta(1-\delta)^{t-1}(1-\varepsilon).$$

Since $\{E_t\}_{t=1}^{\infty}$ are pairwise disjoint sets, letting $E = \bigcup_{t=1}^{\infty} E_t$, we have

$$\mathcal{D}_{\pi}^{\mathcal{M}_{\delta}^{2}}(E) = \sum_{t=1}^{\infty} \mathcal{D}_{\pi}^{\mathcal{M}_{\delta}^{2}}(E_t) \geq \sum_{t=1}^{\infty} \delta(1-\delta)^{t-1}(1-\varepsilon) = 1-\varepsilon.$$

But we have that $E \subseteq B = \{\zeta \in \mathtt{Runs}(S,A) \mid L(\zeta) \notin \mathcal{L}_{\mathtt{safe}}(\{b\})\}$ and hence $J_{\phi}^{\mathcal{M}_{\delta}^{2}}(\pi) = 1 - \mathcal{D}_{\pi}^{\mathcal{M}_{\delta}^{2}}(B) \leq 1 - \mathcal{D}_{\pi}^{\mathcal{M}_{\delta}^{2}}(E) \leq \varepsilon$. Any policy π_2^* that picks a_2 in the first step achieves $J_{\phi}^{\mathcal{M}_{\delta}^{2}}(\pi_2^*) = \mathcal{J}^*(\mathcal{M}_{\delta}^2, \phi) = 1$. Since $\varepsilon < \frac{1}{2}$, we have $J_{\phi}^{\mathcal{M}_{\delta}^{2}}(\pi) \leq \varepsilon < \frac{1}{2} < 1 - \varepsilon = \mathcal{J}^*(\mathcal{M}_{\delta}^2, \phi) - \varepsilon$ which implies $\pi \notin \Pi_{\mathtt{opt}}^{\varepsilon}(\mathcal{M}_{\delta}^2, \phi)$. Therefore $\Pi_{\mathtt{opt}}^{\varepsilon}(\mathcal{M}_{\delta}^1, \phi) \cap \Pi_{\mathtt{opt}}^{\varepsilon}(\mathcal{M}_{\delta}^2, \phi) = \emptyset$ for all $\delta \in]0,1[$. □

Now let h be the sample complexity function of $\mathcal{A}$ as in Definition 2. We let $p = 0.1$ and $N = h(|S|, |A|, |\phi|, \frac{1}{p}, \frac{1}{\varepsilon})$. We let $K = 2N+1$ and choose $\delta \in]0,1[$ such that $(1-\delta)^K \geq 0.9$. Let $\{\pi_n\}_{n=1}^{\infty}$ denote the sequence of output policies of $\mathcal{A}$ when run on $\mathcal{M}$ with the precision $\varepsilon < \frac{1}{2}$ and $p = 0.1$. For $j \in \{1,2\}$, let E_j denote the event that at most N out of the first K policies $\{\pi_n\}_{n=1}^{K}$ are *not* ε-optimal for $\mathcal{M}_{\delta}^{j}$ (when $\mathcal{A}$ is run on $\mathcal{M}$). Then we have $\Pr_{\mathcal{A}}^{\mathcal{M}}(E_1) + \Pr_{\mathcal{A}}^{\mathcal{M}}(E_2) \leq 1$ because E_1 and E_2 are disjoint events (due to Lemma 2).

For $j \in \{1,2\}$, we let $\{\pi_n^j\}_{n=1}^{\infty}$ be the sequence of output policies of $\mathcal{A}$ when run on $\mathcal{M}_{\delta}^{j}$ with the same precision ε and $p = 0.1$. Let F_j denote the event that at most N out of the first K policies $\{\pi_n^j\}_{n=1}^{K}$ are *not* ε-optimal for $\mathcal{M}_{\delta}^{j}$ (when $\mathcal{A}$ is run on $\mathcal{M}_{\delta}^{j}$). Then PAC-MDP guarantee of $\mathcal{A}$ gives us that $\Pr_{\mathcal{A}}^{\mathcal{M}_{\delta}^{j}}(F_j) \geq 0.9$ for $j \in \{1,2\}$. Now let G_j denote the event that the the first K samples from $\mathcal{M}_{\delta}^{j}$ correspond to the deterministic transitions in $\mathcal{M}$—i.e., taking a_1 in s_0 leads to s_0 and taking any action in s_2 leads to s_2. We have that $\Pr_{\mathcal{A}}^{\mathcal{M}_{\delta}^{j}}(G_j) \geq (1-\delta)^K \geq 0.9$ for $j \in \{1,2\}$.

Applying union bound, we get that $\Pr_{\mathcal{A}}^{\mathcal{M}_{\delta}^{j}}(F_j \wedge G_j) \geq 0.8$ for $j \in \{1,2\}$. The probability of any execution (sequence of output policies, actions taken, resets performed and transitions observed) of $\mathcal{A}$ on $\mathcal{M}_{\delta}^{j}$ that satisfies the conditions of F_j and G_j is less than or equal to the probability of obtaining the same execution when $\mathcal{A}$ is run on $\mathcal{M}$ and furthermore such an execution also satisfies the conditions of E_j. Therefore, we have $\Pr_{\mathcal{A}}^{\mathcal{M}}(E_j) \geq \Pr_{\mathcal{A}}^{\mathcal{M}_{\delta}^{j}}(F_j \wedge G_j) \geq 0.8$ for $j \in \{1,2\}$. But this contradicts the fact that $\Pr_{\mathcal{A}}^{\mathcal{M}}(E_1) + \Pr_{\mathcal{A}}^{\mathcal{M}}(E_2) \leq 1$. □

A recent paper [10] claims to provide a PAC-MDP algorithm for LTL specifications. Unfortunately, the analysis is incorrect because the authors assume that for two δ-close MDPs the respective *accepting end states* [3] corresponding to the same specification are the same (in proof of Lemma 1 of the paper) which is not true since two δ-close MDPs can have very different graph connectivity properties as demonstrated in the above proof. However, we believe that their algorithm can be proven to be PAC-MDP under some assumptions on the MDP $\mathcal{M}$; for example, when all transition probabilities are bounded away from 0 and 1.

Next, to the best of our knowledge, it is unknown if there is a learning algorithm that converges in the limit for the class of LTL specifications.

Open Problem 3 *Does there exist a learning algorithm that converges in the limit for the class of LTL specifications?*

Observe that algorithms that converge in the limit do not necessarily have a bound on the number of samples needed to learn an ε-optimal policy; instead, they only guarantee that the values of the policies $\{J_{\phi}^{\mathcal{M}}(\pi_n)\}_{n=1}^{\infty}$ converge to the optimal value $\mathcal{J}^*(\mathcal{M}, \phi)$ almost surely. Therefore, the rate of convergence can be arbitrarily small and can depend on the transition probability function P.

7 Concluding Remarks

We have proposed a formal framework for sampling-based reductions of RL tasks. Given an RL task (an MDP and a specification), the goal is to generate another RL task such that the transformation preserves optimal solutions and is (optionally) robust. A key challenge is that the transformation must be defined without the knowledge of the transition probabilities.

This framework offers a unified view of the literature on RL from logical specifications, in which an RL task with a logical specification is transformed to one with a reward-based specification. We define optimality preserving as well as robust sampling-based reductions for RL tasks. Specification translations are special forms of sampling-based reductions in which the underlying MDP is not altered. We show that specification translations from LTL to reward machines with discounted-sum objectives and to abstract reward machines with limit-average objectives do not preserve optimal solutions. This motivates the need for transformations in which the underlying MDP may be altered. By revisiting such transformations from existing literature within our framework, we expose the nuances in their theoretical guarantees about optimality preservation. Specifically, known transformations from LTL specifications to rewards are not strictly optimality preserving sampling-based reductions since they depend on parameters which are not available in the RL setting such as some information about the transition probabilities of the MDP. We show that LTL specifications, which are non-robust, cannot be robustly transformed to a robust specification, such as discounted-sum or limit-average rewards. We wrap up with proving that LTL specifications do not have a PAC-MDP learning algorithm even for simple safety specifications.

Finally, we are left with multiple open problems. Notably, it is unknown whether there exists a learning algorithm for LTL that converges in the limit which does not depend on any unavailable information about the MDP. However, existing algorithms for learning from LTL specifications have been demonstrated to be effective in practice, even for continuous state MDPs. This shows that there is a gap between the theory and practice suggesting that we need better measures for theoretical analysis of such algorithms; for instance, realistic MDPs may have additional structure that makes learning possible.

Bibliography

[1] Abounadi, J., Bertsekas, D., Borkar, V.S.: Learning algorithms for Markov decision processes with average cost. SIAM Journal on Control and Optimization **40**(3), 681–698 (2001)

[2] Aksaray, D., Jones, A., Kong, Z., Schwager, M., Belta, C.: Q-learning for robust satisfaction of signal temporal logic specifications. In: Conference on Decision and Control (CDC). pp. 6565–6570. IEEE (2016)

[3] de Alfaro, L.: Formal verification of probabilistic systems. Ph.D. thesis, Stanford University (1997)

[4] Baier, C., de Alfaro, L., Forejt, V., Kwiatkowska, M.: Model checking probabilistic systems. In: Handbook of Model Checking, pp. 963–999. Springer (2018)

[5] Bozkurt, A.K., Wang, Y., Zavlanos, M.M., Pajic, M.: Control synthesis from linear temporal logic specifications using model-free reinforcement learning. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 10349–10355. IEEE (2020)

[6] Brafman, R., De Giacomo, G., Patrizi, F.: Ltlf/ldlf non-markovian rewards. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)

[7] Brafman, R.I., Tennenholtz, M.: R-max - a general polynomial time algorithm for near-optimal reinforcement learning. Journal of Machine Learning Research **3** (2003)

[8] Camacho, A., Toro Icarte, R., Klassen, T.Q., Valenzano, R., McIlraith, S.A.: LTL and beyond: Formal languages for reward function specification in reinforcement learning. In: International Joint Conference on Artificial Intelligence. pp. 6065–6073 (7 2019)

[9] De Giacomo, G., Iocchi, L., Favorito, M., Patrizi, F.: Foundations for restraining bolts: Reinforcement learning with ltlf/ldlf restraining specifications. In: Proceedings of the International Conference on Automated Planning and Scheduling. vol. 29, pp. 128–136 (2019)

[10] Fu, J., Topcu, U.: Probably approximately correct MDP learning and control with temporal logic constraints. In: Robotics: Science and Systems (2014)

[11] Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Omega-regular objectives in model-free reinforcement learning. In: Tools and Algorithms for the Construction and Analysis of Systems. pp. 395–412 (2019)

[12] Hasanbeig, M., Kantaros, Y., Abate, A., Kroening, D., Pappas, G.J., Lee, I.: Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees. In: Conference on Decision and Control (CDC). pp. 5338–5343 (2019)

[13] Hasanbeig, M., Abate, A., Kroening, D.: Logically-constrained reinforcement learning. arXiv preprint arXiv:1801.08099 (2018)

[14] Icarte, R.T., Klassen, T., Valenzano, R., McIlraith, S.: Using reward machines for high-level task specification and decomposition in reinforcement learning. In: International Conference on Machine Learning. pp. 2107–2116. PMLR (2018)

[15] Icarte, R.T., Klassen, T.Q., Valenzano, R., McIlraith, S.A.: Reward machines: Exploiting reward function structure in reinforcement learning. arXiv preprint arXiv:2010.03950 (2020)

[16] Jiang, Y., Bharadwaj, S., Wu, B., Shah, R., Topcu, U., Stone, P.: Temporal-logic-based reward shaping for continuing learning tasks (2020)

[17] Jothimurugan, K., Alur, R., Bastani, O.: A composable specification language for reinforcement learning tasks. In: Advances in Neural Information Processing Systems. vol. 32, pp. 13041–13051 (2019)

[18] Jothimurugan, K., Bansal, S., Bastani, O., Alur, R.: Compositional reinforcement learning from logical specifications. In: Advances in Neural Information Processing Systems (To appear) (2021)

[19] Kakade, S.M.: On the sample complexity of reinforcement learning. University of London, University College London (United Kingdom) (2003)

[20] Kearns, M., Singh, S.: Near-optimal reinforcement learning in polynomial time. Machine learning **49**(2), 209–232 (2002)

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

[22] Littman, M.L., Topcu, U., Fu, J., Isbell, C., Wen, M., MacGlashan, J.: Environment-independent task specifications via GLTL (2017)

[23] Littman, M.L., Topcu, U., Fu, J., Isbell, C., Wen, M., MacGlashan, J.: Environment-independent task specifications via GLTL. arXiv preprint arXiv:1704.04341 (2017)

[24] Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science. pp. 46–57. IEEE (1977)

[25] Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. Journal of the ACM (JACM) **32**(3), 733–749 (1985)

[26] Strehl, A.L., Li, L., Wiewiora, E., Langford, J., Littman, M.L.: PAC model-free reinforcement learning. In: Proceedings of the 23rd international conference on Machine learning. pp. 881–888 (2006)

[27] Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)

[28] Watkins, C.J., Dayan, P.: Q-learning. Machine learning **8**(3-4), 279–292 (1992)

[29] Xu, Z., Topcu, U.: Transfer of temporal logic formulas in reinforcement learning. In: International Joint Conference on Artificial Intelligence. pp. 4010–4018 (7 2019)

[30] Yuan, L.Z., Hasanbeig, M., Abate, A., Kroening, D.: Modular deep reinforcement learning with temporal logic specifications. arXiv preprint arXiv:1909.11591 (2019)