2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Model-free Reinforcement Learning for Spatiotemporal Tasks using Symbolic Automata

Anand Balakrishnan[*] Stefan Jakšić[†] Edgar A. Aguilar[†] Dejan Ničković[†] Jyotirmoy V. Deshmukh[*]

Abstract— **Reinforcement learning (RL) is a popular paradigm for synthesizing controllers in environments modeled as Markov Decision Processes (MDPs). The RL formulation assumes that users define local rewards that depend only on the current state (and action), and learning algorithms seek to find control policies that maximize cumulative rewards along system trajectories. An implicit assumption in RL is that policies that maximize cumulative rewards are desirable as they meet the intended control objectives. However, most control objectives are *global* properties of system trajectories, and meeting them with local rewards requires tedious, manual and error-prone process of hand-crafting the rewards. We propose a new algorithm for automatically inferring local rewards from high-level task objectives expressed in the form of *symbolic automata* (SA); a symbolic automaton is a finite state machine where edges are labeled with symbolic predicates over the MDP states. SA subsume many popular formalisms for expressing task objectives, such as discrete-time versions of Signal Temporal Logic (STL). We assume that a model-free RL setting, i.e., we assume *no prior knowledge* of the system dynamics. We give theoretical results that establish that an optimal policy learned using our shaped rewards also maximizes the probability of satisfying the given SA-based control objective. We empirically compare our approach with other RL methods that try to learn policies enforcing temporal logic and automata-based control objective. We demonstrate that our approach outperforms these methods both in terms of the number of iterations required for convergence and the probability that the learned policy satisfies the SA-based objectives.**

I. Introduction

Real-world systems operate in highly uncertain environments, and it is challenging to design precise symbolic models that capture system dynamics and environment uncertainty. A popular abstraction to describe such real-world stochastic systems is that of (discrete-time) Markov decision processes (MDPs). Such MDPs capture the Markovian property that the probability distribution of a state at any time $t + 1$ depends only on its state and control action at time t. A control policy for an MDP is a function that maps an MDP state to the action that should be taken in that state. Reinforcement learning (RL) refers to frameworks that learn the control policies for an MDP by repeated interaction with its environment [1], [2], [3]. The typical RL formulation assumes that each state (and action) in the MDP is associated with a local (Markovian) reward, and RL algorithms seek to find policies that maximize the cumulative reward on the induced system trajectories. Modern RL approaches do not assume any prior knowledge of the system dynamics, and learn a control policy *solely* through interactions with the environment.

A challenge for RL is that local rewards require careful hand-crafting; poorly defined rewards can lead to the RL agent learning policies that maximize cumulative rewards but induce system trajectories that are undesirable or unsafe [4]. An underlying reason for such phenomena is that several kinds of desired temporal behaviors are inherently non-Markovian. For example, consider the property that the system should first reach some region R_1 in the state space, then enter region R_2. A local reward that attempts to move the system state towards R_1 or R_2 would need to know if the system trajectory has visited R_1. Other such examples include patrolling tasks (repeatedly visiting regions over some finite horizon), and sequential tasks (sequentially visiting regions R_1, then R_2, and then R_3). Such tasks are commonly encountered when designing control policies for robot motion.

In contrast, early works on reactive synthesis to satisfy such temporal objectives have been successful in the context of motion planning and controller synthesis [5], [6]. However, these methods require fairly accurate environment models and, additionally, face scalability challenges when the dimension of the state-space increases.

To address these challenges, prior work has explored expressing non-Markovian objectives using temporal logic or automata and augmenting the state space of the MDP with a specification state. Then, a common approach is to propose a (local) reward function that guarantees that the optimal policies (learned using these rewards) satisfy the given automata/logic-based objectives with high probability. Specifically, this line of work includes the use ω-regular automata for infinite horizon behavior [7], [8], [9], [10], finite horizon behavior using deterministic finite automata (DFA) [11], [12], and the use of reward machines [13], [14]. However, several of these methods suffer from the inferred reward function from being sparse; most actions in the augmented MDP may not give the agent a reward, and such infrequent rewardful transitions in the specification automaton can cause poor training performance. We demonstrate this empirically in our experimental results.

Quantitative semantics of temporal logics [15], [16] such as Metric Temporal Logic (MTL) and Signal Temporal Logic (STL) map a given system behavior to a real-valued scalar that measures the spatiotemporal *robustness* of a trajectory. Prior work has also investigated heuristic ways to combine such robustness semantics with RL algorithms [17], [18], [19]. A key limitation of these approaches is that they do not give any guarantees that optimal policies will satisfy the

[*]University of Southern California, Los Angeles, California, USA. Email: {anandbal,jdeshmuk}@usc.edu
[†]AIT Austrian Institute of Technology GmbH, Vienna, Austria. Email: {stefan.jaksic, edgar.aguilar, dejan.nickovic}@ait.ac.at

specification.

In this paper, we propose the use of *symbolic automata* (SA) [20] for specifying control objectives. SA subsume discrete-time variants of STL and MTL while permitting quantitative semantics [21]. Our main contribution is a reward inference procedure that uses the transition structure of the SA to define Markovian rewards such that a policy that maximizes the cumulative local rewards also maximizes the probability of satisfying the SA objective. We empirically demonstrate the efficacy of our reward functions in reducing the training time to find an optimal policy, and show that even suboptimal policies have a higher probability of satisfying the objective compared to similar approaches.

The layout of the paper is as follows. In Section II we introduce the basic terminology, and in Section III we formally introduce the problem. Section IV introduces the reward functions that we infer and we empirically demonstrate their efficacy in Section V. Finally, we conclude with a discussion on related work in Section VI.

II. Preliminaries

A. Symbolic Automata

Let $X = \{x_1, \ldots, x_n\}$ be a set of variables, where each x_i takes values in some compact space D. Let $v : X \to D$ be a *valuation* function (or just valuation) that maps a variable $x \in X$ to the value of x. Given the set of variables X, we abuse notation and use $v(X)$ to denote $(v(x_1), \ldots, v(x_n))$, i.e., $v(X) \in D^n$.

Definition 1 (Predicate): A *predicate* ψ *over* X is defined with the following recursive grammar:

$$\psi := \top \mid \bot \mid \mu(X) \sim 0 \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi,$$

where $\mu(X)$ is a symbolic function corresponding to a hyperplane in $D^{|X|}$, $k \in D$, and $\sim \in \{<, \leq, >, \geq, =\}$. We denote by $\Psi(X)$ the set of all predicates over X.

We will abuse notation and let $\mu(v(X)) \in D$ be the realization of the function for a valuation v such that $\mu : D^{|X|} \to D$ is a scalar function. Then, given a valuation mapping $v : X \to D$, we define the semantics of ψ in terms of a satisfaction relation $v \models \psi$ as follows:

$$\begin{array}{lcl}
v \models \top & \iff & \top \\
v \models \bot & \iff & \bot \\
v \models \mu(X) \sim k & \iff & \mu(v(X)) \sim k \\
v \models \neg\psi & \iff & v \not\models \psi \\
v \models \psi_1 \wedge \psi_2 & \iff & (v \models \psi_1) \wedge (v \models \psi_2) \\
v \models \psi_1 \vee \psi_2 & \iff & (v \models \psi_1) \vee (v \models \psi_2)
\end{array}$$

Definition 2 (Value-Predicate Distance [21]): For a set of variables, X, let $d : D^{|X|} \times D^{|X|}$ be a distance function such that $(D^{|X|}, d)$ is a compact metric space. Given a predicate $\psi \in \Psi(X)$ and a valuation v, we define the *value-predicate distance* as the distance between v and the set of valuations that satisfy ψ:

$$\mathsf{vpd}(v, \psi) = \min_{v' \models \psi} d(v, v'). \tag{1}$$

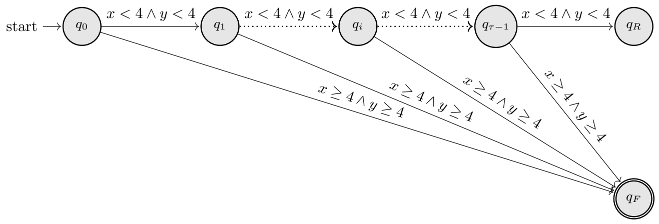


Fig. 1. Symbolic automaton for the task "reach the set represented by the constraints $x \geq 4$ and $y \geq 4$ within τ time steps". The dotted lines represent a continuation of states q_i for all $0 \leq 1 < \tau$. The state $q_0 = q_{\text{init}}$ is the initial state, q_F is the accepting final state, and the state q_R is the "reject" state, to which the system will transition if the agent fails to reach the goal state within the time bound.

Example 1: Let $X = \{x, y\}$ being a set of variables defined over $D \subset \mathbb{R}$, and let the Manhattan distance (1-norm) between two valuations:

$$d_{man}(v, v') = \sum_{i=1}^{n} |v(x_i) - v'(x_i)|. \tag{2}$$

If we have two valuation mappings v_1 and v_2 defined as: $v_1 : \{x \mapsto 3, y \mapsto 5\}$ and $v_2 : \{x \mapsto 2, y \mapsto 1\}$, then $d_{man}(v_1, v_2) = 1 + 4 = 5$.

Definition 3 (Symbolic Automaton [20]): A *symbolic automaton* is a tuple $\mathcal{A} = (X, Q, q_{\text{init}}, F, \Delta, G)$, where X is a finite set of variables, where each variable takes values in D; Q is a finite set of locations with initial location q_{init}; $F \subseteq Q$ is a set of *accepting* locations; $\Delta \subseteq Q \times Q$ is a nonempty set of transitions; and $G : Q \times Q \to \Psi(X)$ is the guard labeling the transition.

We say that a symbolic automaton $\mathcal{A}$ is *terminally accepting* if for every accepting state in F, all outgoing transitions are to some state in F. Such an automaton allows us to replace all accepting states by a single "sink" accepting state q_F, such that $\forall (q_F, q) \in \Delta$, $q = q_F$. In this paper, we restrict our attention to such terminally accepting symbolic automata.

A *run* of the symbolic automaton is defined as a sequence of states and valuations for variables in X: $q_0 \xrightarrow{v_1} q_1 \to \ldots \to q_{n-1} \xrightarrow{v_n} q_n$. Here, $q_0 = q_{\text{init}}$, and for all $i \in [0, n-1]$: $(q_i, q_{i+1}) \in \Delta$, and $v_{i+1} \models G(q_i, q_{i+1})$. A run is accepting if for some n, $q_n \in F$.

B. Reinforcement Learning

Definition 4 (Markov Decision Process (MDP) [3]): An MDP is a tuple $\mathcal{M} = (S, s_{\text{init}}, A, P, R)$, where S is a finite set of states with initial state s_{init}; A is a finite set of possible actions; $P : S \times A \times S \to [0, 1]$ is a (partial) probabilistic transition function, where $P(s, a, s') = \Pr(s' \mid s, a)$ defines the probability of arriving in state s' after taking action a from state s; and $R : S \times S \to \mathbb{R}$ is a reward function defined on $\mathcal{M}$, where $R(s, s')$ denotes the immediate reward received by transitioning from s to s'.

An *episode* $\xi = (s_0, \ldots, s_N)$ is a trace of length N in the MDP $\mathcal{M}$ such that $s_0 = s_{\text{init}}$ and for all $t \in [0, N-1]$, $P(s_t, a_t, s_{t+1}) > 0$ for some $a_t \in A$, and N is the maximum episode length.

Given a set Y, we let $\mathcal{D}(Y)$ denote the set of all probability distributions over Y.

Definition 5 (Policy of an MDP): A policy $\pi : S \to \mathcal{D}(A)$ is a function that maps a state $s \in S$ to a probability distribution over the set of actions $\mathcal{D}(A)$.

Fixing a policy π in $\mathcal{M}$ induces a probability space of episodic trajectories characterized by the distribution $\mathcal{M}^\pi$ where the probability of generating a trajectory ξ in $\mathcal{M}$ under the policy π (denoted $\xi \sim \mathcal{M}^\pi$) is as follows:

$$\Pr(\xi \sim \mathcal{M}^\pi) = \Pr\Big((s_0, \dots, s_N) \;\Big|\; s_0 = s_{\text{init}}\Big),$$

where for all t, the action a_t is sampled from the distribution $\pi(s_t)$, and $P(s_t, a_t, s_{t+1}) > 0$.[1]

Let R_t denote the immediate reward given to the agent at some time t when the MDP transitions from state s_{t-1} to s_t: $R_t = R(s = s_{t-1}, s' = s_t)$.

Definition 6 (Value function [3]): Under a policy $\pi : S \to \mathcal{D}(A)$, the *state-value function* $V^\pi : S \to \mathbb{R}$ of some state $s \in S$ at time t is the expected total reward[2] induced in $\mathcal{M}^\pi$ starting from state s_t:

$$V^\pi(s) = \mathbb{E}_\pi\left[\sum_{i=t}^{N} R_i \mid s_t = s\right]. \tag{3}$$

The *optimal* state-value function $V^*(s)$ is defined as

$$V^*(s) = \max_\pi V^\pi(s), \; \forall s \in S.$$

The goal in reinforcement learning is to find a policy π^* on $\mathcal{M}$ such that $V^{\pi^*}(s_{\text{init}}) = V^*(s_{\text{init}})$, or

$$\pi^* = \operatorname*{argmax}_\pi V^\pi(s_{\text{init}}) \tag{4}$$

III. Problem Statement

Definition 7 (Augmented Product MDP): Given an MDP $\mathcal{M} = (S, s_{\text{init}}, A, P)$ with $S \subset \mathbb{R}^n$ and a symbolic automaton $\mathcal{A} = (X, Q, q_{\text{init}}, F, \Delta, G)$ with a valuation function $v : S \times X \to D$ (where $D \subset \mathbb{R}$), we can construct a product MDP (with additional annotation of accepting states) $\mathcal{P} = \mathcal{M} \otimes \mathcal{A}$ as a tuple $(S_\otimes, s_{\text{init}\otimes}, A, P_\otimes, \mathsf{Acc})$, where:

- $S_\otimes = S \times Q$,
- $s_{\text{init}\otimes} = (s_{\text{init}}, q_{\text{init}})$,
- $P_\otimes : S_\otimes \times A \times S_\otimes \to [0, 1]$ is defined as:

$$P_\otimes((s, q), a, (s, q')) = \begin{cases} P(s, a, s') & \text{if } (q, q') \in \Delta, s \models G(q, q'), \\ 0 & \text{otherwise.} \end{cases}$$

- $\mathsf{Acc} = \{(s, q) \mid q \in F\}$.

An episode $\xi = ((s_0, q_0), \dots, (s_N, q_N))$ in $\mathcal{P}$ (with $(s_0, q_0) = (s_{\text{init}}, q_{\text{init}})$) is considered *accepting* if and only if $(s_N, q_N) \in \mathsf{Acc}$. We use $\xi \models \mathcal{A}$ to denote that the episode ξ is accepted by the specification automaton $\mathcal{A}$.

[1]We will omit the conditions $a_t \sim \pi(s_t)$ and $P(s_t, a_t, s_{t+1}) > 0$ where doing so is not ambiguous.

[2]Usually, the policy synthesis is performed for discounted, infinite runs [3]. In this paper, we only consider the *episodic reinforcement learning* setting [22], where the goal is to maximize expected total returns from trajectories with a finite time bound N and initial state s_{init}.

Given a policy $\pi : S_\otimes \to \mathcal{D}(A)$, we let $\Pr(\pi \models \mathcal{A})$ denote the probability that an episode sampled from $\mathcal{P}^\pi$ is accepted by $\mathcal{A}$ (also called the *probability of* π *being accepting*):

$$\Pr(\pi \models \mathcal{A}) = \Pr_{\xi \sim \mathcal{P}^\pi}\big[\xi \models \mathcal{A}\big] = \mathbb{E}_{\xi \sim \mathcal{P}^\pi}\big[\mathbf{1}(\xi \models \mathcal{A})\big], \tag{5}$$

where $\mathbf{1}(\cdot)$ is the indicator function such that

$$\mathbf{1}(f) = \begin{cases} 1, & \text{if } f \text{ evaluates to } \top \\ 0, & \text{otherwise.} \end{cases}$$

Problem 1: Given an MDP $\mathcal{M} = (S, s_{\text{init}}, A, P, R)$ and a terminally accepting specification automaton $\mathcal{A}$, let $\mathcal{P} = \mathcal{M} \otimes \mathcal{A}$. Synthesize a policy $\pi^* : S \times Q \to \mathcal{D}(A)$ that maximizes the probability of acceptance.

$$\pi^* = \operatorname*{argmax}_\pi \Pr(\pi \models \mathcal{A})$$

IV. Rewards For Symbolic Automata Goals

Given a product MDP, $\mathcal{P} = \mathcal{M} \otimes \mathcal{A}$, we define a reward function $R : S_\otimes \times S_\otimes \to \mathbb{R}$ as:

$$R((s, q), (s', q')) = \begin{cases} d_{\max}, & \text{if } (s', q') \in \mathsf{Acc} \\ & \text{and } (s, q) \notin \mathsf{Acc} \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

where $d_{\max}$ is a hyperparameter for our framework and is usually set to $d_{\max} = \max_{s,s'} d(s, s')$, for some distance function $d : S \times S \to \mathbb{R}$.

Given the reward function R, we claim that any policy π that maximizes the expected total rewards using R will also maximize the probability of satisfying the specification automaton $\mathcal{A}$. Formally:

Theorem 1: Let π_1 and π_2 be some policies on $\mathcal{P}$ such that $V^{\pi_1}((s_{\text{init}}, q_{\text{init}})) > V^{\pi_2}((s_{\text{init}}, q_{\text{init}}))$. Then, $\Pr(\pi_1 \models \mathcal{A}) > \Pr(\pi_2 \models \mathcal{A})$.

Proof: For some trajectory $\xi = ((s_0, q_0), \dots, (s_N, q_N))$ in $\mathcal{P}^\pi$, let the total return for the trajectory be

$$\mathrm{G}(\xi) = \sum_{t=0}^{N-1} R((s_t, q_t), (s_{t+1}, q_{t+1}))$$

Then, from, Eq. 6, we have

$$\mathrm{G}(\xi) = \begin{cases} d_{\max}, & \text{if } \xi \models \mathcal{A} \\ 0, & \text{otherwise.} \end{cases}$$

For notational convenience, we will use V^π_{init} denote the state-value function of policy π for the initial state $(s_{\text{init}}, q_{\text{init}})$ in $\mathcal{P}$. For some policy π, we know that

$$\begin{aligned} V^\pi_{\text{init}} &= \mathbb{E}_{\xi \sim \mathcal{P}^\pi} G(\xi) \\ &= \mathbb{E}_{\xi \sim \mathcal{P}^\pi}[d_{\max} \mid \xi \models \mathcal{A}] + \mathbb{E}_{\xi \sim \mathcal{P}^\pi}[0 \mid \xi \not\models \mathcal{A}] \\ &= d_{\max} \mathbb{E}_{\xi \models \mathcal{P}^\pi}[\mathbf{1}(\xi \models \mathcal{A})] \\ &= d_{\max} \Pr(\pi \models \mathcal{A}). \end{aligned}$$

Thus, if $V^{\pi_1}_{\text{init}} > V^{\pi_2}_{\text{init}}$, $\Pr(\pi_1 \models \mathcal{A}) > \Pr(\pi_2 \models \mathcal{A})$. ■

Corollary 1: Let $p^* = \max_\pi \Pr(\pi \models \mathcal{A})$ be the maximum probability of acceptance of a policy π in $\mathcal{P}$, and let $\pi^* = \operatorname{argmax}_\pi V^\pi((s_{\text{init}}, q_{\text{init}}))$ be an optimal policy with respect to the reward function $R : S_\otimes \times S_\otimes$. Then, $\Pr(\pi^* \models \mathcal{A}) = p^*$.

While the reward in Equation 6 provides theoretical guarantees for an optimal accepting policy, this reward is *sparse*, i.e., for large episode lengths or task horizons, the agent may not see any rewards from accepting runs in the early stages of its training. This can cause a significant slowdown in the training process, and can potentially make it unfeasible to use RL to synthesize such controllers in large or continuous state spaces. To mitigate this, we will present a *reward shaping* technique inspired by the results from [23], [24] that supplements $R((s,q),(s',q')$ with spatial information from the symbolic constraints.

Potential-based Reward Shaping. To speed up the training process, we need to make the reward function R defined in Equation 6 more *dense*, i.e., each transition in $\mathcal{P}$ needs to receive a reward such that the agent receives:

1) a positive reward only if it moves closer to the goal, and,
2) the shaped reward does not alter the set of optimal policies.

To this end, we define a potential-based reward shaping method using a *symbolic potential function*. This symbolic function is a heuristic that takes into account the shortest possible accepting trajectory from the current state in $\mathcal{P}$, *solely by looking at the symbolic constraints in $\mathcal{A}$.*

Definition 8 (Task Progress Level, η): Given a terminally accepting automaton $\mathcal{A} = (X, Q, q_0, F, \Delta, g)$, the *task progress level* is a mapping $\eta : Q \to \mathbb{N} \cup \{\infty\}$ such that $\eta(q)$ is the length of the shortest simple path from the state $q \in Q$ to the state $q_F \in F$. If there is no such path from q to q_F, then $\eta(q) = \infty$.

Let $[\![\psi(q,q')]\!] = \{s \mid s \in S \land s \models \psi(q,q')\}$ be the set of all $s \in S$ that satisfy the predicate $\psi(q,q')$, and let $d_H(\psi_1,\psi_2)$ be the Hausdorff distance between $[\![\psi_1]\!]$ and $[\![\psi_2]\!]$ using some distance measure d in S.

Definition 9 (Symbolic Subtask Progress, Φ_{sym}): For a state $q \in Q$, and a potential future state $q' \in Q$ such that $(q,q') \in \Delta$, $\Phi_{\mathsf{sym}} : \Delta \to \mathbb{R}_{\geq 0}$ provides an *underapproximation* of the distance to the final goal state $q_F \in F$ if the agent takes the transition (q,q'),

$$\Phi_{\mathsf{sym}}(q,q') = \begin{cases} 0 & \text{if } q' \in F \\ \min\limits_{\substack{q'' : (q',q'') \in \Delta \\ q' \neq q''}} d_H([\![\psi(q,q')]\!], [\![\psi(q',q'')]\!]) \\ \qquad\qquad + \quad \Phi_{\mathsf{sym}}(q',q'') & \text{otherwise.} \end{cases} \tag{7}$$

Definition 10 (Symbolic Potential Function, Φ): Thus, given an MDP $\mathcal{M} = (S, s_0, A, P)$ and a symbolic automaton task specification $\mathcal{A}$, we define the function $\Phi : S \times Q \to \mathbb{R}_{\geq 0}$ in the product MDP $\mathcal{P} = \mathcal{M} \otimes \mathcal{A}$ as:

$$\Phi(s,q) = \begin{cases} 0 & \text{if } (s,q) \in \mathsf{Acc} \\ \min\limits_{\substack{q' : (q,q') \in \Delta \\ \eta(q) \neq \eta(q') \lor \\ s \not\models \psi(q,q')}} \mathsf{vpd}(s, \psi(q,q')) \\ \qquad\qquad + \Phi_{\mathsf{sym}}(q,q') & \text{otherwise.} \end{cases} \tag{8}$$

Using the quantity $\eta(\cdot)$, we filter out any edges in the automaton that do not make progress towards the accepting goal, and thus, we compute the underapproximation of the minimum length path in the MDP that leads to Acc starting from some state (s,q). To gain some intuition behind how this potential function works, we refer to Figure 2.

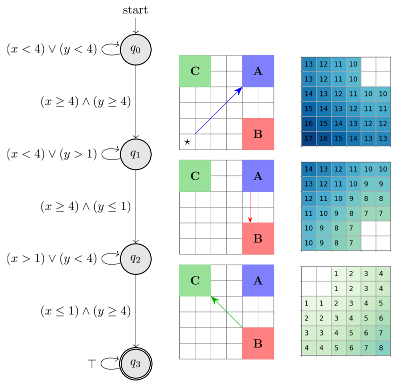


Fig. 2. **Left:** The symbolic automaton for a sequential specification for the agent, where the goal is to visit regions A, B, and C in order. **Middle:** The approximate "shortest path" for the agent to satisfy the specification from $\star$. **Right:** The evaluation of $\Phi(s,q)$ from Equation 8 such that the top image is for $\Phi(s,q_0)$, the middle for $\Phi(s,q_1)$, and the bottom for $\Phi(s,q_2)$, for any s.

From this definition, we can define the shaped reward function, $\hat{R} : S_\otimes \times S_\otimes \to \mathbb{R}$ as:

$$\hat{R}((s,q),(s',q')) = R((s,q),(s',q')) - (\Phi(s',q') - \Phi(s,q)), \tag{9}$$

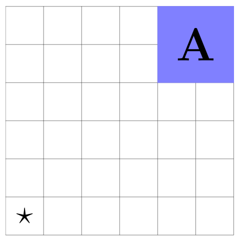
where $R((s,q),(s',q'))$ is as defined in Equation 6. Moreover, we can show that any policy that optimizes $\hat{R}(\cdot,\cdot)$ will remain optimal using Corollary 1 and the policy invariance theorem for potential functions. We formalize this in Theorem 2.

Theorem 2 (Policy Invariance under Shaping [23]): Let p^* be the maximum probability of acceptance. Let π be a policy that maximizes expected total rewards with respect to the sparse reward function R, and $\hat{\pi}$ be one that does so with the potential-based reward shaping $\hat{R}$. Then, $\Pr(\hat{\pi} \models \mathcal{A}) = \Pr(\pi \models \mathcal{A}) = p^*$.

V. Experiments

Environments. In the following case studies, we consider an agent moving through a discrete, grid environment, where the agent can use the actions $A = \{\uparrow, \nearrow, \rightarrow, \searrow, \downarrow, \swarrow, \leftarrow, \nwarrow, 0\}$ which allow also for diagonal movements and no movement (0). We model the probabilistic transition function P in the grid such that if the controller decides to move along a direction, it will move to the next state (if there is no wall) with probability $1 - p_{\text{slip}}$, or move along an adjacent direction with probability $0.5p_{\text{slip}}$ each. Here, p_{slip} is the probability of the agent "slipping", and is set to 0.1.

Baselines. To draw a contrast with other spatiotemporal reward-shaping strategies, we compare the performance of our proposed symbolic potential-based reward (Equation 9) against the sparse rewarding baseline (Equation 6), the automata-based reward presented in [12], and the bounded-horizon quantitative approach presented in [17].



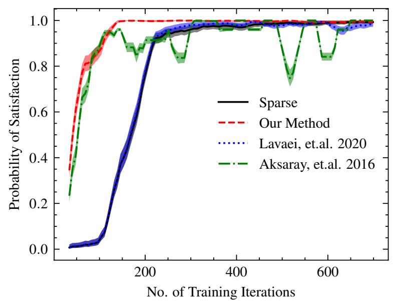
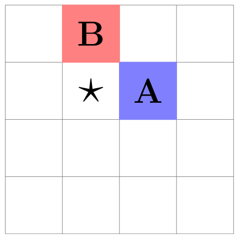


Fig. 3. **Left**: The map for a simple "reach" task, where the goal is to get the agent from $\star$ to the region labeled A within 14 time steps. **Right:** Plot of the probability of the learned policy generating an accepting trajectory vs. the training epoch.



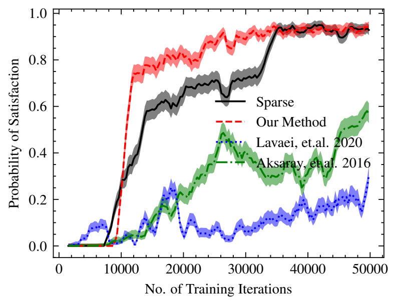
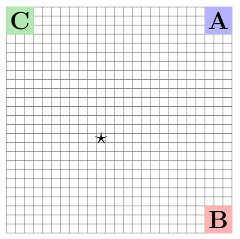


Fig. 4. **Left:** The map for a "recurrence" or patrolling task, where the agent has to visit A and B within 5 time steps of each other over a span of 15 time steps. **Right:** Plot of the probability of the learned policy generating an accepting trajectory vs. the training epoch.



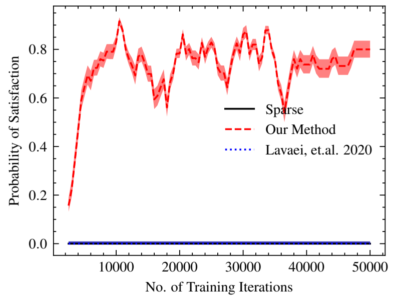


Fig. 5. **Left:** The map for a simple sequential task, where the agent has to visit regions A, B, and C in that order. **Right:** Plot of the probability of the learned policy generating an accepting trajectory vs. the training epoch.

Evaluation. For each rewarding strategy, we train a Q-learning agent [25] and at fixed intervals, we evaluate the learned policy 100 times, and record the number of satisfying runs. The evaluated metrics are then aggregated across 5 training runs with different random seeds. The probability of satisfaction is computed as a binomial distribution with 95% confidence interval.

We will now describe the different environments and tasks performed in these environments, and analyze the performance of our approach and the different baselines:

a) Reachability: Here, the task of the RL agent is to start at some initial location (s_0, q_0) and reach some goal set A (represented in the automaton as a predicate). Specifically, we are interested in synthesizing a controller in a 6×6 grid environment, where the agent starts at state $(0, 0)$ and needs to reach the states satisfying $x \geq 4 \wedge y \geq 4$ with a hard deadline of 15 time steps. The efficacy of our approach can be seen in Figure 3.

From Figure 3, we can see that our proposed method for reward shaping is faster at finding a policy with an acceptance probability equal to 1.0 than other methods. The purely quantitative approach in [17] is the next best solution, but suffers from poor stability in its results. Moreover, in this scenario, the sparse rewarding strategy is exactly as performant as the automaton potential-based reward shaping in [12]. This is due to the fact that while a transition hasn't been taken in the automaton, the potential function in [12] provides no extra information.

b) Recurrence: Based on an environment presented in [17], the goal of the agent in this task is to repeatedly visit two regions in the map as often as possible within a certain time limit. Here, the goal is to visit two regions in a 4×4 grid, labeled $x = 2 \wedge y = 2$ and $x = 1 \wedge y = 3$.

In Figure 4, we can see that the approach presented in [12] performs poorly. This is due to the fact that in such specifications, using $\eta(\cdot)$ to compute the potential may mislead the agent into taking choices that are local maximums in the rewards. This is similar to the issue present in the "Branching Paths" task presented later.

On the other hand, increasing the task horizon (as defined in [17]) by a few time steps causes the τ-MDP method to perform poorly due to a state-space explosion.

c) Sequential: This task requires an agent to visit regions in a strict sequence. In this example, the agent is placed in a 25×25 grid environment, with 3 labeled regions:

- $A = \{(x, y) \mid (x \geq 22) \wedge (y \geq 22)\}$
- $B = \{(x, y) \mid (x \geq 22) \wedge (y \leq 3)\}$
- $C = \{(x, y) \mid (x \leq 3) \wedge (y \geq 22)\}$

The goal of the agent is to learn a controller that visits region A, then the region B, and finally region C in sequence.

Remark 1: Since the environment and the task horizon are considerably large (even for artificially bounded task specifications), the state space for the τ-MDP construct presented in [17] explodes greatly. This made the experiments unfeasible to run with this method, and thus the results for this method are omitted from Figure 5.

In this task, we notice that the sparse reward baseline, along with the potential-based reward shaping presented in [12] do not learn any good information for 50000 training epochs. This is due to the highly sparse rewards provided by both the methods. Similar to the results in the "Bounded Reach" task presented earlier, the method in [12] does not provide any information to the agent until it enters the region corresponding to the next task. On the other hand, our proposed method learns to find satisfying traces relatively quickly due to the information from the potential function defined in Eq. (8).

d) Branching Paths: In this specification, the agent operates on a 16×16 grid with a few obstacles, as seen in Figure 7. The goal of the agent is to visit regions in either of the following orders: $A \to B \to D$ or $C \to D$. From the

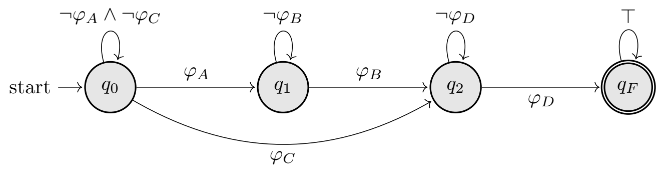


Fig. 6. This symbolic automaton represents the task "reach D by either going through A and then B, or by going through C". The state $q_0 = q_{\text{init}}$ is the initial state, $A = ((0 \leq x \leq 1) \wedge (14 \leq y \leq 15))$, $B = ((5 \leq x \leq 8) \wedge (14 \leq y \leq 15))$, and $C = ((x >= 14) \wedge (y <= 1))$. Note how the "obviously" shorter path in the automaton doesn't necessarily correspond to the spatially shorter path.

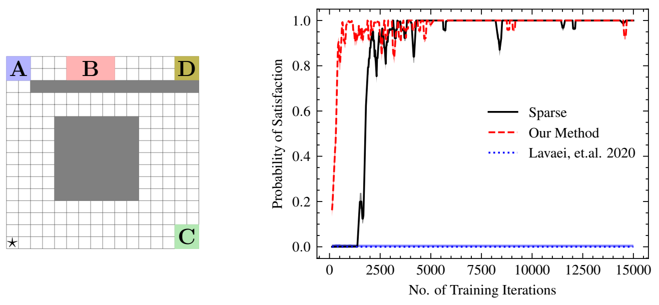


Fig. 7. **Left:** The map for a task with two possible paths: the agent can either take the path $A \to B \to D$ or $C \to D$. Walls and obstacles have been intentionally placed to make one path easier than the other, but the agent does not have any knowledge of these environment features. **Right:** Plot of the probability of the learned policy generating an accepting trajectory vs. the training epoch.

figure, we can see that one of the above orders is significantly longer than the other, but since the agent does not have any prior knowledge of the environment (except for the locations of these regions), it cannot rule out either branch. We can see from Figure 7 that our proposed method significantly outperforms the approach presented in [12]. (Note that due to size restrictions, we do not see that there is a small error band around the graph for [12] for when they do find some accepting trajectories.) Moreover, we can see that the sparse rewarding case also performs well as it weights all the accepting paths in the automaton equally, while the reward function in [12] weights an unfeasible path as being better in the automaton shown in Figure 6.

Remark 2: For the same reasons as in the "Sequential" task, we do not evaluate the performance of the τ-MDP approach presented in [17].

VI. Related Work and Conclusions

Related Work: Reward engineering in RL based on formal specifications is a well-established research topic [17], [19], [10], [26], [12]. Early work in [7] and [8] encoded rewards using deterministic Rabin automata generated from Linear Temporal Logic (LTL) specifications, while [9] and [10] do so using limit-deterministic Büchi automata. These types of automata define tasks over infinite horizon behavior. Recent work shows that it is impossible for an RL algorithm to provide PAC guarantees, i.e., an algorithm that guarantees with a high probability that it learns a near-optimal policy with only a polynomial number of interactions with the environment [27].

In contrast, [11] and [12] use a fragment of LTL to define finite horizon tasks, which are translated to deterministic finite automaton (DFA). In [12], the authors propose the use of DFAs to encode task specifications over discrete labeled inputs, and define a state-based potential function on the automaton. Similarly, the authors of [28], [26] propose the use of a custom specification language to generate a similar DFA, but rather than learning a single controller for the entire specification, they propose to learn a controller for each "subtask" encoded on an edge in the form of some guard. These multiple controllers are then scheduled using the automaton structure as a guide. In [13] and [14], the authors propose the use of reward machines — Mealy machines with reward functions as outputs — while their extension work in [29] proposes to simplify these constructions as Moore machines with numeric outputs.

In [17], the authors define an effective approach to learning robust controllers using Q-learning [25]. The history-based dependency of formula satisfaction is resolved by encoding n-step history in every state. The authors use bounded horizon robustness as a reward, which requires transforming MDP by enhancing it with n-step MDP history. The works in [30] and [19] use similar robustness-based approaches to defining reward functions over bounded horizons.

In recent years, the use of control barrier functions and control Lyapunov functions for motion planning using RL has become popular due to their inherent potential-based formulation [31], [32], [33], [34], [35]. In [33], the authors propose to use neural networks to learn estimates of Lyapunov function derivatives under uncertainty and update the controller accordingly. Similarly, [31] and [32] try to directly formulate the RL problem as one involving Lyapunov and barrier certificate constraints. The work in [35] proposes an approach similar to ours, where temporal logic specifications are translated to DFAs, a learned controller handles the task of reaching the accepting state, and the task of avoiding unsafe conditions is delegated to control barrier functions.

Conclusions and Future Work: This paper presents a novel approach to using symbolic automata as task specifications to encode complex tasks. Compared to other automata-based solutions, we show that the reward function obtained from this symbolic task specification can encode rich, quantitative information about the environment. We present theoretical guarantees for the correctness of the constructed reward and empirically compare the approach against related works.In future work, we hope to 1) extend these results in episodic reinforcement learning to infinite horizon tasks; 2) provide guarantees for this approach in continuous space and continuous time settings; 3) study how to construct robust plans for multi-agent systems with global and local tasks.

Acknowledgment

This work is part of the AI4CSM project that has received funding within the ECSEL JU in collaboration with the European Union's H2020 Framework Programme

(H2020/2014-2020) and National Authorities, under grant agreement No.101007326. This work was also supported by the National Science Foundation through the following grants: CAREER award (SHF-2048094), CNS-1932620, CNS-2039087, FMitF-1837131, CCF-SHF-1932620, a grant from the Airbus Institute for Engineering Research, and funding by Toyota R&D and Siemens Corporate Research through the USC Center for Autonomy and AI.

References

[1] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 834–846, Sep. 1983.

[2] D. Bertsekas and J. Tsitsiklis, "Neuro-Dynamic Programming: An Overview," in *Proceedings of 1995 34th IEEE Conference on Decision and Control*, vol. 1, Dec. 1995, pp. 560–564 vol.1.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2nd ed., ser. Adaptive Computation and Machine Learning Series. Cambridge, Massachusetts: MIT press, 2018.

[4] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete Problems in AI Safety," *arXiv:1606.06565 [cs]*, Jul. 2016.

[5] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-Logic-Based Reactive Mission and Motion Planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, Dec. 2009.

[6] M. Lahijanian, J. Wasniewski, S. B. Andersson, and C. Belta, "Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 3227–3232.

[7] J. Fu and U. Topcu, "Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints," in *Robotics: Science and Systems X*, vol. 10, Jul. 2014.

[8] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia, "A Learning Based Approach to Control Synthesis of Markov Decision Processes for Linear Temporal Logic Specifications," in *53rd IEEE Conference on Decision and Control*, Dec. 2014, pp. 1091–1096.

[9] M. Hasanbeig, A. Abate, and D. Kroening, "Logically-Constrained Reinforcement Learning," *arXiv:1801.08099 [cs]*, Jan. 2018.

[10] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak, "Omega-Regular Objectives in Model-Free Reinforcement Learning," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, T. Vojnar and L. Zhang, Eds. Cham: Springer International Publishing, 2019, pp. 395–412.

[11] X. Li, Y. Ma, and C. Belta, "Automata-Guided Hierarchical Reinforcement Learning for Skill Composition," *arXiv:1711.00129 [cs]*, May 2018.

[12] A. Lavaei, F. Somenzi, S. Soudjani, A. Trivedi, and M. Zamani, "Formal Controller Synthesis for Continuous-Space MDPs via Model-Free Reinforcement Learning," in *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. Sydney, Australia: IEEE, Apr. 2020, pp. 98–107.

[13] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith, "Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, Jul. 2018, pp. 2107–2116.

[14] A. Camacho, R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. Macao, China: International Joint Conferences on Artificial Intelligence Organization, Aug. 2019, pp. 6065–6073.

[15] G. E. Fainekos and G. J. Pappas, "Robustness of Temporal Logic Specifications for Continuous-Time Signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, Sep. 2009.

[16] A. Donzé and O. Maler, "Robust Satisfaction of Temporal Logic over Real-Valued Signals," in *Formal Modeling and Analysis of Timed Systems*, ser. Lecture Notes in Computer Science, K. Chatterjee and T. A. Henzinger, Eds. Berlin, Heidelberg: Springer, 2010, pp. 92–106.

[17] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-Learning for Robust Satisfaction of Signal Temporal Logic Specifications," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec. 2016, pp. 6565–6570.

[18] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement Learning with Temporal Logic Rewards," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 3834–3839.

[19] A. Balakrishnan and J. V. Deshmukh, "Structured reward shaping using signal temporal logic specifications," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Nov. 2019, pp. 3481–3486.

[20] L. D'Antoni and M. Veanes, "The Power of Symbolic Automata and Transducers," in *Computer Aided Verification*, R. Majumdar and V. Kunčak, Eds. Cham: Springer International Publishing, 2017, vol. 10426, pp. 47–67.

[21] S. Jakšić, E. Bartocci, R. Grosu, and D. Ničković, "An Algebraic Framework for Runtime Verification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2233–2243, Nov. 2018.

[22] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, ser. Optimization and Neural Computation Series. Belmont, Mass: Athena Scientific, 1996.

[23] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, ser. ICML '99, vol. 99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jun. 1999, pp. 278–287.

[24] M. Grześ, "Reward Shaping in Episodic Reinforcement Learning," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '17. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 2017, pp. 565–573.

[25] C. J. C. H. Watkins and P. Dayan, "Q-Learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992.

[26] K. Jothimurugan, S. Bansal, O. Bastani, and R. Alur, "Compositional Reinforcement Learning from Logical Specifications," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[27] C. Yang, M. L. Littman, and M. Carbin, "On the (In)Tractability of Reinforcement Learning for LTL Objectives," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. Vienna, Austria: International Joint Conferences on Artificial Intelligence Organization, Jul. 2022, pp. 3650–3658. [Online]. Available: https://www.ijcai.org/proceedings/2022/507

[28] K. Jothimurugan, R. Alur, and O. Bastani, "A Composable Specification Language for Reinforcement Learning Tasks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[29] R. T. Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning," *Journal of Artificial Intelligence Research*, vol. 73, pp. 173–208, Jan. 2022.

[30] X. Li, Y. Ma, and C. Belta, "A policy search method for temporal logic specified reinforcement learning tasks," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 240–245.

[31] J. Choi, F. Castañeda, C. Tomlin, and K. Sreenath, "Reinforcement Learning for Safety-Critical Control under Model Uncertainty, using Control Lyapunov Functions and Control Barrier Functions," in *Robotics: Science and Systems XVI*, vol. 16, Jul. 2020.

[32] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A Lyapunov-based Approach to Safe Reinforcement Learning," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018.

[33] A. J. Taylor, V. D. Dorobantu, H. M. Le, Y. Yue, and A. D. Ames, "Episodic Learning with Control Lyapunov Functions for Uncertain Robotic Systems," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 6878–6884.

[34] W. Liu, N. Mehdipour, and C. Belta, "Recurrent Neural Network Controllers for Signal Temporal Logic Specifications Subject to Safety Constraints," *IEEE Control Systems Letters*, vol. 6, pp. 91–96, 2022.

[35] X. Li, Z. Serlin, G. Yang, and C. Belta, "A Formal Methods Approach to Interpretable Reinforcement Learning for Robotic Planning," *Science Robotics*, vol. 4, no. 37, Dec. 2019.