Provably Correct Automata Embeddings for Optimal Automata-Conditioned Reinforcement Learning

Beyazit Yalcinkaya BEYAZIT@BERKELEY.EDU
University of California, Berkeley

Niklas Lauffer NLAUFFER@BERKELEY.EDU
University of California, Berkeley

Marcell Vazquez-Chanlatte MARCELL.CHANLATTE@NISSAN-USA.COM
Nissan Advanced Technology Center Silicon Valley

Sanjit A. Seshia SSESHIA@BERKELEY.EDU
University of California, Berkeley

Editors: G. Pappas, P. Ravikumar, S. A. Seshia

Abstract

Automata-conditioned reinforcement learning (RL) has given promising results for learning multi-task policies capable of performing temporally extended objectives given at runtime, done by pretraining and freezing automata embeddings prior to training the downstream policy. However, no theoretical guarantees were given. This work provides a theoretical framework for the automata-conditioned RL problem and shows that it is probably approximately correct learnable. We then present a technique for learning provably correct automata embeddings, guaranteeing optimal multi-task policy learning. Our experimental evaluation confirms these theoretical results.[1]

Keywords: reinforcement learning, representation learning, formal specifications.

1. Introduction

Goal-Conditioned Reinforcement Learning (GCRL) is a framework for learning policies capable of performing multiple tasks given at runtime. The recent success of foundation models has popularized both natural language (Rocamonde et al. (2023); Brohan et al. (2022); Black et al. (2024)) and demonstrations (Ren et al. (2025); Sontakke et al. (2023)) as ergonomic means of task specification. Yet, the inherent ambiguity of these instruction modalities remains a challenge for correctness guarantees.

Formal specifications have been proposed for specifying tasks to goal-conditioned policies. While their well-defined semantics make them appealing, approaches that rely on hierarchical planning, i.e., planning over the induced automaton of a formal specification and instructing a goal-conditioned policy to execute the plan (Jothimurugan et al. (2021); Qiu et al. (2023)), are inherently suboptimal due to the myopia of their goal-conditioned policies. Conditioning Reinforcement Learning (RL) policies on Linear Temporal Logic (LTL) specifications was proposed by Vaezipoor et al. (2021), using a Graph Neural Network (GNN) to encode abstract syntax trees of LTL formulas. However, generalization is an inherent limitation due to their use of LTL (Yalcinkaya et al. (2024)).

In our previous work (Yalcinkaya et al. (2023, 2024)), we proposed using Deterministic Finite Automaton (DFA) as a means of task specification and conditioning the policy on pretrained DFA *embeddings*. To do so, we first identified a large class of DFAs that capture most of the finite temporal tasks studied in the literature. We then pretrained a Graph Attention Network (GATv2) (Brody et al.

1. For more information about the project including the code and extensions visit: https://rad-embeddings.github.io/.

(2021)) to map these DFAs to latent vector representations. Our empirical evaluation demonstrated that conditioning on DFA embeddings enables optimal multi-task policy learning for a large class of DFAs. However, no theoretical analysis or guarantees were given for this technique.

In this work, we present a theoretical framework for the DFA-conditioned RL problem and show that it is Probably Approximately Correct (PAC)-learnable. As we showed in Yalcinkaya et al. (2024), learning to encode the DFAs while simultaneously learning a policy is challenging due to the sparse reward specified by DFA acceptance. To this end, in the same work, we demonstrated that pretraining and freezing DFA embeddings and then passing these embeddings to a downstream policy greatly improve learning efficiency. Therefore, in order for our PAC learnability guarantee to be useful in practice, one must show that such guarantees hold w.r.t. such pretrained and frozen DFA embeddings. To address this, we present a novel method for learning provably correct DFA embeddings, guaranteeing optimal DFA-conditioned RL. We first observe that *bisimilar* DFAs represent the same task and then use *bisimulation metrics*, a relaxation of the notion of a bisimulation relation, to embed unique tasks to unique latent representations. Our experimental evaluation shows that the correctness of the learned DFA embeddings improves downstream policy learning.

Contributions. We provide a theoretical framework for DFA-conditioned RL in Section 3 and prove that it is PAC-learnable in Theorem 1. We present a technique for learning provably correct DFA embeddings in Section 4. Lastly, an empirical evaluation of this approach is given in Section 5.

Related Work. PAC-learnability of RL objectives given by formal specifications has been studied before. Yang et al. (2021) proved that the optimal policy for any LTL formula is PAC-learnable if and only if the formula can be checked within a finite horizon. A similar result by Alur et al. (2022) shows that without additional information on the transition probabilities, such as the minimum nonzero transition probability, LTL is not PAC-learnable. Later, a positive result for discounted LTL was given by Alur et al. (2023). Our PAC-learnability result can be considered a multi-task generalization of these previous results, where the policy must satisfy a class of specifications, not a single objective. See Yalcinkaya et al. (2024) for a more detailed literature review on using formal specification in RL.

Ferns et al. (2004) showed that bisimulation metrics can be computed as unique fixed points of a contraction map. A special case of this result for deterministic dynamics and on-policy samples was proved by Castro (2020). Later, Zhang et al. (2020) used bisimulation metrics to learn invariant observation embeddings for RL, where they proved these metrics can be computed while learning a policy. We will use these results in Section 4 to learn provably correct DFA embeddings. To our knowledge, no prior work considered using bisimulation metrics for learning task representations.

2. Background

Notation. Given a set X, we write $\mathcal{X}$ to denote a distribution over it, i.e., $\mathcal{X} \in \Delta(X)$, where $\Delta(X) \subset X \to [0, 1]$ represents the set of all distributions over X. We use $\mathbb{I}\{\}$ for the event indicator function, where $\mathbb{I}\{p\} = 1$ if and only if p is true, and $\mathbb{I}\{p\} = 0$ otherwise.

2.1. Markov Decision Processes

We model the environment with a Markov Decision Process (MDP), formally defined as follows.

Definition 1 (Markov Decision Process) *A Markov Decision Process (MDP) is defined as the tuple* $\mathcal{M} = \langle S, A, P, R, \iota, \gamma \rangle$*, where* S *is the state space,* A *is the action space,* $P : S \times A \to \Delta(S)$ *is*

the transition probability function, $R : S \times A \to \mathbb{R}$ is the reward function, $\iota \in \Delta(S)$ is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. An MDP $\mathcal{M}$ is called deterministic if it has a deterministic transition function $T : S \times A \to S$ instead of a probabilistic transition function P.

2.2. Deterministic Finite Automata

We use Deterministic Finite Automata (DFAs) with three-valued semantics to represent tasks.

Definition 2 (Deterministic Finite Automaton) *A Deterministic Finite Automaton (DFA) is defined as the tuple $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, where Q is the finite set of states, Σ is the finite alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, where $\delta(q, a) = q'$ denotes a transition to a state $q' \in Q$ from a state $q \in Q$ by observing a symbol $a \in \Sigma$, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of final states. The three-valued semantics of a DFA is defined by a partition of its final states $F = F^\top \cup F^\bot$ and its extended (lifted) transition function $\delta^* : Q \times \Sigma^* \to Q$, where*

$$\delta^*(q, \varepsilon) \triangleq q,$$
$$\delta^*(q, aw) \triangleq \delta^*(\delta(q, a), w).$$

*If $\delta^*(q_0, w) \in F^\top$, then we say that $\mathcal{A}$ **accepts** w. Similarly, $\delta^*(q_0, w) \in F^\bot$, then we say that $\mathcal{A}$ **rejects** w. $\mathcal{A}$ is called a **plan DFA** if its final states are sink states, i.e., $\forall q \in F, \forall a \in \Sigma, \delta(q, a) = q$.*

DFAs can be minimized to a canonical form (up to a state isomorphism) using the algorithm of Hopcroft (1971), denoted by $\text{minimize}(\mathcal{A})$. All minimized plan DFAs with nonempty accepted or rejected languages have single accepting or rejecting states, denoted by $\top$ or $\bot$, respectively, as their final states are all sink states. We denote the single-state accepting DFA by $\mathcal{A}_\top$ and the rejecting one by $\mathcal{A}_\bot$. Note that the approach presented in this paper is agnostic to the syntax of DFAs and can be trivially extended to other syntactic forms, e.g., *compositional DFAs* from Yalcinkaya et al. (2024).

Assumption 1 *In what follows, unless stated otherwise, all DFAs are plan DFAs.*

We use DFAs to represent temporal tasks, which can be understood as *plans*. However, one can define multiple DFAs for the same task, i.e., DFAs without any additional assumptions are not canonical task representations. Therefore, we will need a notion of similarity between DFAs so that we can compare the tasks represented by them–we define *bisimulation relation over DFAs* next.

Definition 3 (Bisimulation Relation over DFAs) *Given two DFAs $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ and $\mathcal{A}' = \langle Q', \Sigma, \delta', q_0', F' \rangle$ over the same alphabet Σ. A relation $B \subseteq Q \times Q'$ is called a **bisimulation relation** between $\mathcal{A}$ and $\mathcal{A}'$ if the following conditions hold:*

1. *$(q_0, q_0') \in B$.*
2. *For all $(q, q') \in B$, $q \in F^\top \iff q' \in F'^\top$ and $q \in F^\bot \iff q' \in F'^\bot$.*
3. *For all $(q, q') \in B$ and $a \in \Sigma$, $(\delta(q, a), \delta'(q', a)) \in B$.*

*We say that $\mathcal{A}$ and $\mathcal{A}'$ are **bisimilar**, denoted by $\mathcal{A} \sim \mathcal{A}'$, if there exists such a bisimulation relation.*

A bisimulation relation over DFAs is an equivalence relation on the DFA states preserving both the transition structure and the acceptance condition–meaning if two states are related under this relation, then for every input symbol, their successor states are also related, and they either both accept or both reject. Bisimilar DFAs are behaviorally indistinguishable–they represent the same task.

3. DFA-Conditioned Reinforcement Learning

We will define the DFA-conditioned RL problem over a distribution of DFAs, similar to the GCRL problem given in Appendix A. However, we need some extra structure over these DFAs since, in our case, goals are not simply sets of states but are DFAs encoding temporal tasks given to the policy.

Definition 4 (DFA Space) *A set of DFAs $D_{\Sigma,n}$ over some shared alphabet Σ and with at most n states is called a **DFA space** if $\mathcal{A}_\top, \mathcal{A}_\bot \in D_{\Sigma,n}$ and taking any path in a DFA from $D_{\Sigma,n}$ and minimizing the resulting DFA gives a DFA in $D_{\Sigma,n}$, i.e.,*

$$\forall \mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle \in D_{\Sigma,n}, \forall w \in \Sigma^*, \quad \text{minimize}(\mathcal{A}' = \langle Q, \Sigma, \delta, \delta^*(q_0, w), F\rangle) \in D_{\Sigma,n}.$$

A DFA space $D_{\Sigma,n}$ induces an MDP defined by the tuple $\mathcal{M}_{D_{\Sigma,n}} = \langle D_{\Sigma,n}, \Sigma, T_{D_{\Sigma,n}}, R_{D_{\Sigma,n}}\rangle$, where

- *$D_{\Sigma,n}$, the set of DFAs, is the set of states,*
- *Σ, the shared alphabet, is the set of actions,*
- *$T_{D_{\Sigma,n}} : D_{\Sigma,n} \times \Sigma \to D_{\Sigma,n}$ is the transition function defined by*

$$T_{D_{\Sigma,n}}(\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle, a) = \text{minimize}(\mathcal{A}' = \langle Q, \Sigma, \delta, \delta(q_0, a), F\rangle), \text{ and}$$

- *$R_{D_{\Sigma,n}} : D_{\Sigma,n} \times \Sigma \to \{-1, 0, 1\}$ is the reward function defined by*

$$R_{D_{\Sigma,n}}(\mathcal{A}_t, a) = \begin{cases} 1 & \text{if } T_{D_{\Sigma,n}}(\mathcal{A}_t, a) = \mathcal{A}_\top \\ -1 & \text{if } T_{D_{\Sigma,n}}(\mathcal{A}_t, a) = \mathcal{A}_\bot \\ 0 & \text{otherwise.} \end{cases}$$

A DFA space is a set of DFAs closed under random walks, i.e., taking any random path in a DFA from this set and pruning the unreachable states results in a DFA in the set. In other words, one cannot get a DFA outside this set by taking a random walk with minimization, hence the name *space*.

Assumption 2 *$D_{\Sigma,n}$ denotes a DFA space, and $\mathcal{D}_{\Sigma,n} \in \Delta(D_{\Sigma,n})$ is a distribution over it.*

We now have all the theoretical machinery needed to formally state the DFA-conditioned RL problem. We first define the environment model and then continue with the statement of the problem.

Definition 5 (DFA-Conditioned MDP) *Let $\mathcal{M} = \langle S, A, P, R, \iota, \gamma\rangle$ be an MDP, $D_{\Sigma,n}$ be a DFA space, and $L : S \to \Sigma$ be a labeling function. A DFA-conditioned MDP is the cascade composition of $\mathcal{M}$ and $\mathcal{M}_{D_{\Sigma,n}}$, using L to map states to alphabet symbols, defined by*

$$\mathcal{M} \mid_L \mathcal{M}_{D_{\Sigma,n}} = \langle S \times D_{\Sigma,n}, A, P_{\mathcal{M}\mid_L \mathcal{M}_{D_{\Sigma,n}}}, R_{\mathcal{M}\mid_L \mathcal{M}_{D_{\Sigma,n}}}, \iota_{\mathcal{M}\mid_L \mathcal{M}_{D_{\Sigma,n}}}, \gamma\rangle$$

where:

- *$S \times D_{\Sigma,n}$ is the (product) state space,*
- *A is the action space (of $\mathcal{M}$),*

- $P_{\mathcal{M}|_L\mathcal{M}_{D_{\Sigma,n}}} : S \times D_{\Sigma,n} \times A \to \Delta(S \times D_{\Sigma,n})$ *is the transition probability function defined by*

$$P_{\mathcal{M}|_L\mathcal{M}_{D_{\Sigma,n}}}(s, \mathcal{A}, a, s', \mathcal{A}') = P(s, a, s')\mathbb{I}\left\{\mathcal{A}' = T_{D_{\Sigma,n}}\left(\mathcal{A}, L(s')\right)\right\},$$

- $R_{\mathcal{M}|_L\mathcal{M}_{D_{\Sigma,n}}} : S \times D_{\Sigma,n} \times A \to \{-1, 0, 1\}$ *is the reward function defined by*

$$R_{\mathcal{M}|_L\mathcal{M}_{D_{\Sigma,n}}}(s, \mathcal{A}, a) = \begin{cases} 1 & \text{if } T_{D_{\Sigma,n}}(\mathcal{A}, L(s')) = \mathcal{A}_\top \\ -1 & \text{if } T_{D_{\Sigma,n}}(\mathcal{A}, L(s')) = \mathcal{A}_\bot \\ 0 & \text{otherwise,} \end{cases}$$

  where $s' \sim P(s, a)$ *is the next MDP state.*

- $\iota_{\mathcal{M}|_L\mathcal{M}_{D_{\Sigma,n}}} \in \Delta(S \times D_{\Sigma,n})$ *is the initial state distribution defined by*

$$\iota_{\mathcal{M}|_L\mathcal{M}_{D_{\Sigma,n}}}(s, \mathcal{A}) = \iota(s)\mathcal{D}_{\Sigma,n}(\mathcal{A}),$$

  and

- $\gamma \in [0, 1)$ *is the discount factor (of* $\mathcal{M}$*).*

A DFA-conditioned MDP essentially couples an MDP with a DFA space, where the policy interacts with the MDP while simultaneously navigating the DFA space to reach the accepting DFA. Next, we formalize this notion and finally state the DFA-conditioned RL problem.

Definition 6 (DFA-Conditioned Reinforcement Learning Problem) *Given an DFA-conditioned MDP* $\mathcal{M} \mid_L \mathcal{M}_{D_{\Sigma,n}}$ *as defined in Definition 5, a* ***DFA-conditioned policy*** *is a mapping*

$$\pi : S \times D_{\Sigma,n} \to \Delta(A),$$

that assigns to each pair $(s, \mathcal{A})$ *a probability distribution over the action space* A*. The* ***DFA-conditioned RL problem*** *is to find a policy* π *maximizing expected cumulative discounted reward:*

$$J_{\mathcal{M}|_L\mathcal{M}_{D_{\Sigma,n}}}(\pi) = \mathbb{E}_{(s_0,\mathcal{A}_0)\sim\iota_{\mathcal{M}|_L\mathcal{M}_{D_{\Sigma,n}}}}\left[\sum_{t=0}^{\mathcal{A}_t=\mathcal{A}_\top\vee\mathcal{A}_t=\mathcal{A}_\bot} \gamma^t R_{\mathcal{M}|_L\mathcal{M}_{D_{\Sigma,n}}}(s_t, \mathcal{A}_t, a_t)\right],$$

where trace $\{(s_t, \mathcal{A}_t, a_t)\}_{t\geq 0}$ *is generated by:*

$$a_t \sim \pi(s_t, \mathcal{A}_t), \quad s_{t+1} \sim P(s_t, a_t), \quad \mathcal{A}_{t+1} = T_{D_{\Sigma,n}}\left(\mathcal{A}_t,\, L(s_{t+1})\right),$$

until the accepting or rejecting DFA is reached, i.e., $\mathcal{A}_t = \mathcal{A}_\top \vee \mathcal{A}_t = \mathcal{A}_\bot$*. The objective is to solve*

$$\pi^* = \arg\max_\pi J_{\mathcal{M}|_L\mathcal{M}_{D_{\Sigma,n}}}(\pi),$$

i.e., to learn a policy that maximizes the probability of satisfying a given temporal specification from $D_{\Sigma,n}$ *(by driving its DFA representation to* $\mathcal{A}_\top$*) while operating in the underlying MDP* $\mathcal{M}$*.*

Notice the difference between the GCRL problem formulation given in Appendix A and the DFA-conditioned one given above. Specifically, in GCRL, goals are static, i.e., they are given at the beginning, and the policy conditions on the same goal until it is accomplished. In the DFA-conditioned RL setting, through the labeling function, a given DFA task also evolves (based on the transition dynamics of its DFA space) as the policy interacts with the underlying MDP. Therefore, the policy is essentially learning to navigate two MDPs: the underlying MDP and the DFA space.

3.1. PAC-Learnability of the DFA-Conditioned Reinforcement Learning Problem

We show that the DFA-conditioned RL problem is Probably Approximately Correct (PAC)-learnable. To do so, we will use the PAC-MDP framework introduced by Strehl et al. (2006). An RL algorithm is called PAC-MDP (PAC in MDPs) if it finds a near-optimal policy with high probability in any MDP after a number of interactions that is polynomial in the problem's key parameters, stated next.

Definition 7 (Probably Approximately Correct Learnability in MDPs) *A learning algorithm $\mathbb{A}$ is said to be **Probably Approximately Correct in MDPs (PAC-MDP)** if for any MDP $\mathcal{M} = (S, A, T, R, \iota, \gamma)$, $\epsilon > 0$, and $p \in (0, 1)$, there exists a polynomial function*

$$N = f\left(|S|, |A|, \frac{1}{\epsilon}, \frac{1}{p}, \frac{1}{1-\gamma}\right)$$

s.t., with probability at least $1 - p$, the total number of time steps during which the policy π (current policy being trained) executed by $\mathbb{A}$ is more than ϵ-suboptimal is at most N, i.e., we have

$$|\{t \geq 0 \ : \ V^{\pi}(s_t) < V^{*}(s_t) - \epsilon\}| \leq N$$

with probability at least $1-p$, where V^{π} denotes the current value function and V^{*} is the optimal one.

We want to show that if an algorithm is PAC-MDP, then it is also PAC in any DFA-conditioned MDP. Observe that, in Definition 5, we take the cascade composition of the underlying MDP and the MDP induced by the DFA space which is finite, giving us the following PAC-learnability result.

Theorem 1 *If a learning algorithm $\mathbb{A}$ is PAC-MDP as defined in Definition 7, then for any DFA-conditioned MDP $\mathcal{M} \mid_L \mathcal{M}_{D_{\Sigma,n}}$, $\epsilon > 0$, and $p \in (0, 1)$, there exists a polynomial function*

$$N' = f\left(|S| \cdot |D_{\Sigma,n}|, |A|, \frac{1}{\epsilon}, \frac{1}{p}, \frac{1}{1-\gamma}\right)$$

s.t. the total number of ϵ-suboptimal steps taken by $\mathbb{A}$ is at most N' with probability at least $1 - p$.

The proof is in the appendix. Theorem 1 proves that the DFA-conditioned RL problem is PAC-learnable, assuming the underlying MDP is solvable. However, in practice, one cannot input a DFA to a policy directly. Instead, one uses an encoder (possibly pretrained) mapping DFAs to embeddings. In such cases, the optimality of the learned DFA-conditioned policy depends on the encoder.

4. Learning Provably Correct Automata Embeddings

In the previous section, we introduced the idealized, theoretical formulation of the DFA-conditioned RL problem and proved that it is PAC-learnable. However, a policy implemented by a feed-forward neural network, as is usually the case, cannot condition on a DFA directly, but rather an encoding of the DFA is required, such as a vector representation. Then, the question is whether the policy conditioning on DFA encodings is optimal w.r.t. the theoretical formulation of the problem. This is the problem we tackle in the following, but before doing so, we formally state the problem.

Problem 1 *Given a DFA space $D_{\Sigma,n}$, learn an encoder $\phi : D_{\Sigma,n} \to \mathcal{Z}$ s.t. for any MDP $\mathcal{M}$ and labeling function L, solving $\mathcal{M} \mid_L \mathcal{M}_{D_{\Sigma,n}}$ with a policy $\pi_{\mathcal{Z}} : S \times \mathcal{Z} \to \Delta(A)$ conditioning on DFA embeddings is equivalent to solving it with a DFA-conditioned policy $\pi : S \times D_{\Sigma,n} \to \Delta(A)$, i.e.,*

$$\forall \mathcal{M}, \forall L, \quad \arg\max_{\pi} J_{\mathcal{M}|_L \mathcal{M}_{D_{\Sigma,n}}}(\pi) = \arg\max_{\pi_{\mathcal{Z}} \circ \phi} J_{\mathcal{M}|_L \mathcal{M}_{D_{\Sigma,n}}}(\pi_{\mathcal{Z}} \circ \phi),$$

where $\pi_{\mathcal{Z}} \circ \phi(s, \mathcal{A}) = \pi_{\mathcal{Z}}(s, \phi(\mathcal{A}))$.

Assumption 3 *$\phi : D_{\Sigma,n} \to \mathcal{Z}$ has enough capacity to represent DFAs in its domain. That is, the learnable encoder ϕ has a parametrization that can map distinct DFAs in $D_{\Sigma,n}$ to unique embeddings.*

Intuitively, we want a policy conditioning on the latent representations of DFAs (rather than DFAs themselves) to be equivalent to the theoretical formulation given in Definition 6, i.e., one finds the optimal solution whenever the other does so. Observe that even under Assumption 3, one does not get such a guarantee directly since the claim is not only an expressivity argument but also involves proving that the training procedure of the encoder provides such representations. In the following, we present a training technique for such encoders and prove that it solves Problem 1. Our method involves learning a *bisimulation metric* over the induced MDP of a given DFA space. Therefore, we first define bisimulation metrics and then show how they can be computed in deterministic MDPs.

4.1. Bisimulation Relations and Metrics over MDP states

A bisimulation metric can be viewed as a relaxation of the notion of a bisimulation relation over MDP states. We start by defining the latter and then continue with the former.

Definition 8 (Bisimulation Relation over MDP states) *Let $\mathcal{M} = \langle S, A, P, R, \iota, \gamma \rangle$ be an MDP. A relation $B \subseteq S \times S$ is called a **bisimulation relation** if for every pair $(s, t) \in B$ and for every action $a \in A$, the following conditions hold:*

1. *$R(s, a) = R(t, a)$.*
2. *For all B-closed set $X \subseteq S$ (i.e., if $x \in X$ and $(x, y) \in B$ then $y \in X$),*

$$\sum_{x \in X} P(s, a, x) = \sum_{x \in X} P(t, a, x).$$

*We say $s, t \in S$ are **bisimilar**, denoted by $s \sim_{\mathcal{M}} t$, if there is a bisimulation relation B s.t. $(s, t) \in B$.*

Intuitively, two states are bisimilar if they are behaviorally indistinguishable–taking any action in either state yields the same immediate reward and leads to similar probabilistic outcomes, so an agent cannot tell them apart when making decisions. Next, we show how this relates to Definition 3.

Lemma 1 *Given a DFA space $D_{\Sigma,n}$, two DFAs $\mathcal{A}, \mathcal{A}' \in D_{\Sigma,n}$ are bisimilar if and only if they are bisimilar states in the induced deterministic MDP $\mathcal{M}_{D_{\Sigma,n}}$, i.e.,*

$$\forall \mathcal{A}, \mathcal{A}' \in \mathcal{D}_{\Sigma,n}, \quad \mathcal{A} \sim \mathcal{A}' \iff \mathcal{A} \sim_{\mathcal{M}_{D_{\Sigma,n}}} \mathcal{A}'.$$

The proof is given in the appendix. Essentially, Lemma 1 shows that Definition 3 and Definition 8 are equivalent for DFAs in a DFA space, which will later help us with Problem 1. We will continue with the formal definition of a bisimulation metric. But before doing so, informally, a function $d : X \times X \to \mathbb{R}_{\geq 0}$ is called a *pseudometric* on a set X if it satisfies non-negativity, symmetry, and the triangle inequality but may allow $d(x, y) = 0$ for $x \neq y$, see Appendix B for a formal definition.

Definition 9 (Bisimulation Metric) *Let $\mathcal{M} = \langle S, A, P, R, \iota, \gamma \rangle$ be an MDP. A pseudometric d is called a **bisimulation metric** if the equivalence relation induced by d is a bisimulation relation, i.e.,*

$$\sim_{\mathcal{M}} = \{(s, t) \in S \times S \mid_L d(s, t) = 0\}.$$

Recall that a given DFA space $D_{\Sigma,n}$ induces a deterministic MDP $\mathcal{M}_{D_{\Sigma,n}}$, where each state of this MDP is a DFA. We want our learned encoder to uniquely distinguish between different behaviors, but we do not care whether we can distinguish between different representations of the same task. Therefore, we can use a bisimulation metric to measure *how bisimilar* two DFAs are and utilize this idea to learn a provably correct embedding space by ensuring that if two DFAs are not bisimilar, then they must have different embeddings. To this end, we first present the following result stating that bisimulation metrics over deterministic MDPs can be computed as fixed-point solutions.

Theorem 2 *Let $\mathcal{M} = \langle S, A, T, R, \iota, \gamma \rangle$ be a deterministic MDP. Define operators:*

$$d^k(s, t) \leftarrow \left| R(s, \pi^k(s, t)) - R(t, \pi^k(s, t)) \right| + \gamma\, d^{k-1}\left(T(s, \pi^k(s, t)),\, T(t, \pi^k(s, t)) \right),$$
$$\pi^k(s, t) \leftarrow \arg\max_{a \in A} \left\{ |R(s, a) - R(t, a)| + \gamma\, d^{k-1}\left(T(s, a),\, T(t, a) \right) \right\}.$$

Then, there exists unique fixed points d^* and π^*, and d^* is a bisimulation metric.

The proof is given in the appendix. Theorem 2 is essentially an adaptation of the results previously given in this domain to our setting. Specifically, Ferns et al. (2004) first proved that a bisimulation metric can be computed as a unique fixed point of a contraction map. Castro (2020) later showed special cases of this result for deterministic MDPs and for on-policy variants where actions are given by a policy. Zhang et al. (2020) then presented a result showing that one can learn a bisimulation metric jointly while learning a control policy predicting actions for a downstream task. We combine these results to show that a bisimulation metric can be computed while simultaneously learning a policy maximizing it. Given π^*, a bisimulation metric d^* can be computed up to an α accuracy by iteratively applying the operator $\left\lceil \frac{\ln \alpha}{\ln \gamma} \right\rceil$ times, with an overall complexity of $O\left(|A||S|^4 \log |S| \frac{\ln \alpha}{\ln \gamma} \right)$.

4.2. Learning Automata Embeddings by Computing Bisimulation Metrics

Given a DFA space $D_{\Sigma,n}$, to learn an encoder $\phi : D_{\Sigma,n} \to \mathcal{Z}$ that solves Problem 1, we train it to learn latent representations s.t. their normalized ℓ_2-norms form a bisimulation metric. To do so, we use the operators from Theorem 2 and define our pseudometric as follows:

$$d(\mathcal{A}, \mathcal{A}') \triangleq \|\hat{\phi}(\mathcal{A}) - \hat{\phi}(\mathcal{A}')\|_2, \tag{1}$$

where $\hat{\phi}(\mathcal{A}) = \frac{\phi(\mathcal{A})}{\|\phi(\mathcal{A})\|_2}$ denotes vector normalization. Since it is hard to compute the $\arg\max$ in Theorem 2, we simultaneously learn a policy $\pi : \mathcal{Z} \times \mathcal{Z} \to \Delta(\Sigma)$ approximating it in the latent space. We generate episodes starting from $\mathcal{A}_0, \mathcal{A}'_0 \sim \mathcal{D}_{\Sigma,n}$ and evolving as follows:

$$a_t \sim (\pi \circ \phi)(\mathcal{A}_t, \mathcal{A}'_t), \quad \mathcal{A}_{t+1} = T_{D_{\Sigma,n}}(\mathcal{A}_t, a_t), \quad \mathcal{A}'_{t+1} = T_{D_{\Sigma,n}}(\mathcal{A}'_t, a_t),$$

where $(\pi \circ \phi)(\mathcal{A}_t, \mathcal{A}'_t) = \pi(\phi(\mathcal{A}_t), \phi(\mathcal{A}'_t))$. We use Proximal Policy Optimization (PPO) by Schulman et al. (2017) to jointly learn π and ϕ with the following objective:

$$J_{D_{\Sigma,n}}(\pi \circ \phi) = J_{\text{clip}}(\pi \circ \phi) + J_{\text{val}}(\phi), \tag{2}$$

where $J_{\text{clip}}(\pi \circ \phi)$ is the clipped surrogate objective computed using Equation (1) as its value function, i.e., $V_t = d(\mathcal{A}_t, \mathcal{A}'_t)$. The details of $J_{\text{clip}}(\pi \circ \phi)$ and PPO are not relevant to us; however, note that while it is not guaranteed, it usually finds the optimal solution, see Schulman et al. (2017) for details. The second term in the objective given in Equation (2), the value objective, is defined as follows:

$$\begin{aligned} J_{\text{val}}(\phi) &= -\left(V_t - \left(\left|R_{D_{\Sigma,n}}(\mathcal{A}_t, a_t) - R_{D_{\Sigma,n}}(\mathcal{A}'_t, a_t)\right| + \gamma \bar{V}_{t+1}\right)\right)^2 \\ &= -\left(d(\mathcal{A}_t, \mathcal{A}'_t) - \left(\left|R_{D_{\Sigma,n}}(\mathcal{A}_t, a_t) - R_{D_{\Sigma,n}}(\mathcal{A}'_t, a_t)\right| + \gamma \bar{d}(\mathcal{A}_{t+1}, \mathcal{A}'_{t+1})\right)\right)^2, \end{aligned}$$

where $\bar{V}_{t+1}$ and $\bar{d}(\mathcal{A}_{t+1}, \mathcal{A}'_{t+1})$ denotes calls with stop gradients, i.e., no gradient flow to ϕ. $J_{\text{val}}(\phi)$ implements the objective for the pseudometric given in Theorem 2, penalizing for diverging from the one-step lookahead target. The combined objective of ϕ is then to learn latent representations that form a bisimulation metric under normalized ℓ_2-norm while also providing representations for π. Next, we show that an encoder maximizing Equation (2) maps two DFAs to the same embedding if and only if they are bisimilar, therefore proving that such encoders can distinguish distinct tasks.

Lemma 2 *Let $D_{\Sigma,n}$ be a DFA space, ϕ^* be an encoder, and π^* be a policy s.t.*

$$\pi^* \circ \phi^* = \arg\max_{\pi \circ \phi} J_{D_{\Sigma,n}}(\pi \circ \phi),$$

where $J_{D_{\Sigma,n}}(\pi \circ \phi)$ is given by Equation (2). Then, ϕ^* satisfies:

$$\forall \mathcal{A}, \mathcal{A}' \in D_{\Sigma,n}, \quad \mathcal{A} \sim \mathcal{A}' \iff \phi^*(\mathcal{A}) = \phi^*(\mathcal{A}').$$

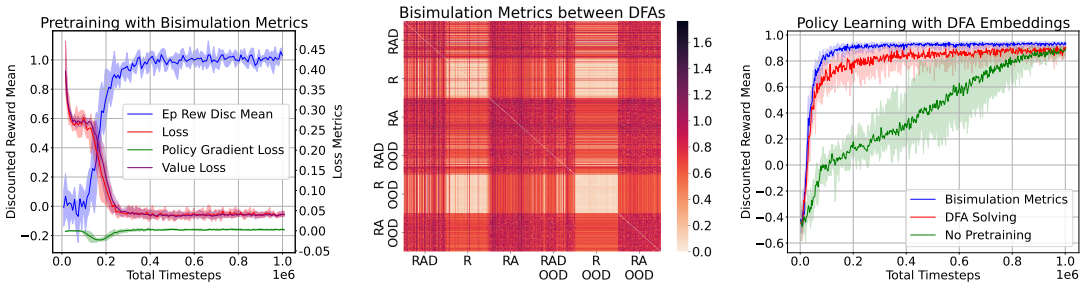
The proof is given in the appendix. Observe that if our trained encoder can distinguish between DFAs that are not bisimilar, then it solves Problem 1, as bisimilar DFAs are different representations for the same task–no need to distinguish them. Next, we formally state this result, solving Problem 1.

Theorem 3 *Let $D_{\Sigma,n}$ be a DFA space, ϕ be an encoder, and π^* be a policy s.t.*

$$\pi^* \circ \phi^* = \arg\max_{\pi \circ \phi} J_{D_{\Sigma,n}}(\pi \circ \phi),$$

where $J_{D_{\Sigma,n}}(\pi \circ \phi)$ is given by Equation (2). Then, ϕ^* solves Problem 1.

The proof is given in the appendix. Intuitively, since our encoder can distinguish bisimilar DFAs and bisimilar DFAs represent the same task, one can equivalently reformulate the DFA-conditioned RL problem given in Definition 6, which is defined over $D_{\Sigma,n}$, as one over $\mathcal{Z}$, solving Problem 1.



(*a*) Learning curves (*b*) Normalized ℓ_2 distances (*c*) DFA-condition policies

Figure 1: Left: learning curves for the method in Section 4. Center: heatmap of Equation (1) between various DFAs. Right: learning DFA-conditioned policies for RA tasks with number of states sampled uniformly from $[3, 6]$, comparing DFA embeddings from Section 4 and Yalcinkaya et al. (2024).

5. Experiments

We implemented the technique given in Section 4 using a GATv2 model as our DFA encoder and Reach-Avoid Derived (RAD) DFAs with at most 10 states, which are plan DFAs, both presented in Yalcinkaya et al. (2024). One difference in our GATv2 model is that given a DFA with n states, we do n message-passing steps, instead of doing it for a fixed number as in Yalcinkaya et al. (2024). To break the symmetry, caused by taking the absolute value of the reward difference, we trained the policy using the reward difference without the absolute value. Figure 1(*a*) shows that our training technique finds the fixed point, where the objectives from Section 4 are given as losses. We then tested the accuracy as well as the generalization capabilities of these DFA embeddings. To do so, we generated RAD, Reach (R), and Reah-Avoid (RA) DFAs. During training the number of states of a RAD DFA was sampled from a truncated geometric distribution (with 10 as the upper bound) whereas during testing we sampled it using a bounded uniform distribution. We also generated out-of-distribution (OOD) DFAs with the number of states sampled uniformly between 11 and 20. We computed bisimulation metrics, i.e., the normalized ℓ_2-norms, between the embeddings of these DFAs. Figure 1(*b*) gives these results in the form of a heatmap, demonstrating the correctness of the learned DFA embeddings–0 on the diagonal. We further checked whether any of these sampled DFAs (both in-distribution and OOD ones) are mapped to the same embedding (up to a 10^{-8} accuracy) or not, and we confirm that the encoder has a 100% success rate in these samples. Figure 1(*c*) compares our new pretraining technique from Section 4 with our previous pretraining procedure based on solving DFAs (Yalcinkaya et al. (2024)), which does not guarantee correctness, showing that the correctness of DFA embeddings improves downstream policy learning. All results are over 5 seeds.

6. Conclusion

In this work, we established a theoretical framework for DFA-conditioned RL and showed its PAC-learnability. We then introduced a method for learning provably correct automata embeddings, ensuring optimal multi-task policy learning. Our approach builds on the promising results of DFA-conditioned RL, leveraging pretrained and frozen DFA embeddings to enable the learning of policies for temporally extended objectives specified at runtime. Our experimental evaluation confirms the theoretical guarantees of our method, demonstrating DFA-embeddings enable optimal multi-task RL.

Acknowledgments

This work is partially supported by the DARPA contract FA8750-23-C-0080 (ANSR), by Nissan and Toyota under the iCyPhy Center, and by C3DTI. Niklas Lauffer is supported by an NSF fellowship.

References

Rajeev Alur, Suguman Bansal, Osbert Bastani, and Kishor Jothimurugan. A framework for transforming specifications in reinforcement learning. In *Principles of Systems Design: Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday*, pages 604–624. Springer, 2022.

Rajeev Alur, Osbert Bastani, Kishor Jothimurugan, Mateo Perez, Fabio Somenzi, and Ashutosh Trivedi. Policy synthesis and reinforcement learning for discounted ltl. In *International Conference on Computer Aided Verification*, pages 415–435. Springer, 2023.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10069–10076, 2020.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *UAI*, volume 4, pages 162–169, 2004.

John Hopcroft. An n log n algorithm for minimizing states in a finite automaton. In *Theory of machines and computations*, pages 189–196. Elsevier, 1971.

Kishor Jothimurugan, Suguman Bansal, Osbert Bastani, and Rajeev Alur. Compositional reinforcement learning from logical specifications. *Advances in Neural Information Processing Systems*, 34:10026–10039, 2021.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*, 2022.

Wenjie Qiu, Wensen Mao, and He Zhu. Instructing goal-conditioned reinforcement learning agents with temporal logic objectives. *Advances in Neural Information Processing Systems*, 36:39147–39175, 2023.

Juntao Ren, Priya Sundaresan, Dorsa Sadigh, Sanjiban Choudhury, and Jeannette Bohg. Motion tracks: A unified representation for human-robot transfer in few-shot imitation learning. *arXiv preprint arXiv:2501.06994*, 2025.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. *arXiv preprint arXiv:2310.12921*, 2023.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sumedh Sontakke, Jesse Zhang, Séb Arnold, Karl Pertsch, Erdem Bıyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. Roboclip: One demonstration is enough to learn robot policies. *Advances in Neural Information Processing Systems*, 36:55681–55693, 2023.

Alexander L Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L Littman. Pac model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 881–888, 2006.

Pashootan Vaezipoor, Andrew C Li, Rodrigo A Toro Icarte, and Sheila A Mcilraith. Ltl2action: Generalizing ltl instructions for multi-task rl. In *International Conference on Machine Learning*, pages 10497–10508. PMLR, 2021.

Beyazit Yalcinkaya, Niklas Lauffer, Marcell Vazquez-Chanlatte, and Sanjit Seshia. Automata conditioned reinforcement learning with experience replay. In *NeurIPS 2023 Workshop on Goal-Conditioned Reinforcement Learning*, 2023.

Beyazit Yalcinkaya, Niklas Lauffer, Marcell Vazquez-Chanlatte, and Sanjit A Seshia. Compositional automata embeddings for goal-conditioned reinforcement learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Cambridge Yang, Michael Littman, and Michael Carbin. On the (in) tractability of reinforcement learning for ltl objectives. *arXiv preprint arXiv:2111.12679*, 2021.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

Appendix A. Goal-Conditioned Reinforcement Learning

Here, for reference, we present the standard Goal-Conditioned Reinforcement Learning (GCRL) problem. We start by defining the environment model for conditioning on *goals*, usually given as sets of states or continuous regions, as done by Schaul et al. (2015); Liu et al. (2022).

Definition 10 (Goal-Conditioned MDP) *A goal-conditioned MDP extends the standard MDP by incorporating a goal space given by a goal distribution $\mathcal{G} \in \Delta(2^S)$, where $\mathcal{G}$ is a distribution over sets of states, and therefore a goal is a set of states. It is defined as the tuple*

$$\mathcal{M}_{\mathcal{G}} = \langle S, A, P, R_{\mathcal{G}}, \iota_{\mathcal{G}}, \gamma \rangle,$$

where:

- $R_{\mathcal{G}} : S \times A \times G \to \mathbb{R}$ *is the goal-conditioned reward function, and*
- $\iota_{\mathcal{G}} : S \times G \to [0, 1]$ *is the initial state-goal distribution defined by* $\iota_{\mathcal{G}}(s, g) = \iota(s)\mathcal{G}(g)$.

Given a goal-conditioned MDP, the standard GCRL problem is to find a policy that achieves a given goal, which was first introduced by Schaul et al. (2015).

Definition 11 (Goal-Conditioned Reinforcement Learning) *Given a goal-conditioned MDP $\mathcal{M}_{\mathcal{G}}$, the **Goal-Conditioned Reinforcement Learning** (GCRL) problem is to find a policy*

$$\pi : S \times G \to \Delta(A),$$

which maps a state-goal pair (s, g) to a distribution over actions, that maximizes the expected cumulative discounted reward:

$$J_{\mathcal{M}_{\mathcal{G}}}(\pi) = \mathbb{E}_{s_0, g \sim \iota_{\mathcal{G}}} \left[\sum_{t=0}^{s_t \in g} \gamma^t R_{\mathcal{G}}(s_t, a_t, g) \right],$$

where trace $\{(s_t, a_t)\}_{t \geq 0}$ is generated by $a_t \sim \pi(s_t)$ and $s_{t+1} \sim P(s_t, a_t)$ until $s_t \in g$ is reached. The objective is to solve

$$\pi^* = \arg\max_{\pi} J_{\mathcal{M}_{\mathcal{G}}}(\pi).$$

The standard GCRL formulation doesn't inherently allow for specifying temporally extended tasks since the goals are defined as sets of states. In theory, one can extend the state definition to a product state and specify temporal tasks within that product state; however, such approaches limit the scalability of the GCRL framework. On the other hand, our DFA-conditioned RL formulation given in Definition 6 allows for specifying temporal tasks and enables optimal multi-task policy learning.

Appendix B. Pseudometrics and Metrics

Definition 12 (Pseudometric and Metric) *Let X be a nonempty set. A function $d : X \times X \to [0, \infty)$ is called a **pseudometric** on X if for all $x, y, z \in X$ the following conditions hold:*

1. ***Non-negativity:*** $d(x, y) \geq 0$.

2. ***Identity on the diagonal:*** $d(x, x) = 0$.
3. ***Symmetry:*** $d(x, y) = d(y, x)$.
4. ***Triangle Inequality:*** $d(x, z) \leq d(x, y) + d(y, z)$.

If $d(x, y) = 0$ *implies* $x = y$*, then* d *is a* ***metric****.*

Essentially, a metric is a function measuring the distance between any two points in a space, satisfying non-negativity, symmetry, the triangle inequality, and it equals zero if and only if the two points are identical. A pseudometric, on the other hand, allows distinct points to have a distance of zero, meaning it might not fully distinguish between different points in the space.

Appendix C. Proofs of Theorems and Lemmas

Theorem 1 *If a learning algorithm* $\mathbb{A}$ *is PAC-MDP as defined in Definition 7, then for any DFA-conditioned MDP* $\mathcal{M} \mid_L \mathcal{M}_{D_{\Sigma,n}}$*,* $\epsilon > 0$*, and* $p \in (0, 1)$*, there exists a polynomial function*

$$N' = f\left(|S| \cdot |D_{\Sigma,n}|, |A|, \frac{1}{\epsilon}, \frac{1}{p}, \frac{1}{1-\gamma}\right)$$

s.t. the total number of ϵ*-suboptimal steps taken by* $\mathbb{A}$ *is at most* N' *with probability at least* $1 - p$*.*

Proof A DFA-conditioned MDP $\mathcal{M} \mid_L \mathcal{M}_{D_{\Sigma,n}}$ is defined over the state space $S \times D_{\Sigma,n}$, where $D_{\Sigma,n}$ is a DFA space. Since $D_{\Sigma,n}$ is finite (as all DFAs in $D_{\Sigma,n}$ has a finite alphabet Σ and at most n states), the product state space has size $|S| \cdot |D_{\Sigma,n}|$ and is an MDP. Thus, any PAC-MDP algorithm that works for MDPs with state space size $|S|$ will also work on the product MDP with state space size $|S| \cdot |D_{\Sigma,n}|$, with sample complexity increasing by at most a factor polynomial in $|D_{\Sigma,n}|$. ■

Lemma 1 *Given a DFA space* $D_{\Sigma,n}$*, two DFAs* $\mathcal{A}, \mathcal{A}' \in D_{\Sigma,n}$ *are bisimilar if and only if they are bisimilar states in the induced deterministic MDP* $\mathcal{M}_{D_{\Sigma,n}}$*, i.e.,*

$$\forall \mathcal{A}, \mathcal{A}' \in \mathcal{D}_{\Sigma,n}, \quad \mathcal{A} \sim \mathcal{A}' \iff \mathcal{A} \sim_{\mathcal{M}_{D_{\Sigma,n}}} \mathcal{A}'.$$

Proof If $\mathcal{A} \sim \mathcal{A}'$, then they must agree on acceptance, by Definition 3. We have $R_{D_{\Sigma,n}}(\mathcal{A}) = 1$ if and only if $\mathcal{A} = \mathcal{A}_\top$. Since $\mathcal{A}$ and $\mathcal{A}'$ are bisimilar, $\mathcal{A} = \mathcal{A}_\top \iff \mathcal{A}' = \mathcal{A}_\top$. The same reasoning for the -1 reward case gives $R_{D_{\Sigma,n}}(\mathcal{A}) = R_{D_{\Sigma,n}}(\mathcal{A}')$, i.e., $\mathcal{A}$ and $\mathcal{A}'$ satisfy reward equivalence in $\mathcal{M}_{D_{\Sigma,n}}$. For any $a \in \Sigma$, $T_{D_{\Sigma,n}}(\mathcal{A}, a)$ results in a DFA bisimilar to $T_{D_{\Sigma,n}}(\mathcal{A}', a)$ due to Definition 3. By induction on the structure of $\mathcal{A}$ and $\mathcal{A}'$, their transitions preserve bisimilarity, satisfying the transition equivalence. Therefore, we have $\mathcal{A} \sim \mathcal{A}' \implies \mathcal{A} \sim_{\mathcal{M}_{D_{\Sigma,n}}} \mathcal{A}'$.

If $\mathcal{A} \sim_{\mathcal{M}_{D_{\Sigma,n}}} \mathcal{A}'$, then $R_{D_{\Sigma,n}}(\mathcal{A}) = R_{D_{\Sigma,n}}(\mathcal{A}')$. Thus, $\mathcal{A}$ and $\mathcal{A}'$ must agree on acceptance by Definition 5. For every $a \in \Sigma$, $T_{D_{\Sigma,n}}(\mathcal{A}, a) \sim_{\mathcal{M}_{D_{\Sigma,n}}} T_{D_{\Sigma,n}}(\mathcal{A}', a)$ by Definition 5. By induction on the DFA transition structure (which is finite), $T_{D_{\Sigma,n}}(\mathcal{A}, a)$ and $T_{D_{\Sigma,n}}(\mathcal{A}', a)$ are bisimilar. As transitions under all $a \in \Sigma$ preserve bisimilarity, the initial states q_0 and q_0' must be related under the bisimulation relation. Thus, $\mathcal{A}$ and $\mathcal{A}'$ are bisimilar, i.e., $\mathcal{A} \sim \mathcal{A}' \impliedby \mathcal{A} \sim_{\mathcal{M}_{D_{\Sigma,n}}} \mathcal{A}'$. ■

Theorem 2 *Let $\mathcal{M} = \langle S, A, T, R, \iota, \gamma \rangle$ be a deterministic MDP. Define operators:*

$$d^k(s,t) \leftarrow \left| R(s, \pi^k(s,t)) - R(t, \pi^k(s,t)) \right| + \gamma\, d^{k-1}\left(T(s, \pi^k(s,t)),\, T(t, \pi^k(s,t)) \right),$$

$$\pi^k(s,t) \leftarrow \arg\max_{a \in A} \left\{ |R(s,a) - R(t,a)| + \gamma\, d^{k-1}\left(T(s,a),\, T(t,a)\right) \right\}.$$

Then, there exists unique fixed points d^* and π^*, and d^* is a bisimulation metric.

Proof Let $\mathcal{M} = \langle S, A, P, R, \iota, \gamma \rangle$ be an MDP. Define

$$d^k(s,t) \leftarrow \arg\max_{a \in A} \left\{ |R(s,a) - R(t,a)| + \gamma \mathcal{W}_1(d)\left(P(s,a), P(t,a)\right) \right\},$$

where $\mathcal{W}_1$ is the 1-Wasserstein metric. Ferns et al. (2004) showed that this operator has a unique fixed point d^*, and d^* is a bisimulation metric. Later, Castro (2020) proved that if $\mathcal{M}$ is deterministic, with transition function T, then the 1-Wasserstein metric above implies as follows:

$$\mathcal{W}_1(d)\left(P(s,a), P(s,a)\right) = d\left(T(s,a), T(s,a)\right).$$

We write the distance and the policy update separately since we want to learn the $\arg\max$–hard to compute directly. However, then we need to prove that the distance metric still has a unique fixed point when updated with actions from a policy being simultaneously learned. A useful result due to Zhang et al. (2020) (which we present by combining with the result of Castro (2020) and our notation) shows that given a continuously improving policy π, the following operator:

$$d^k(s,t) \leftarrow |R(s, \pi(s,t)) - R(t, \pi(s,t))| + \gamma d\left(T(s, \pi(s,t)), T(s, \pi(s,t))\right)$$

has a unique fixed point d^*, and d^* is a π^*-bisimulation metric. In our case, π^* is the $\arg\max$ policy and therefore the unique fixed point d^* is a bisimulation metric.

■

Lemma 2 *Let $D_{\Sigma,n}$ be a DFA space, ϕ^* be an encoder, and π^* be a policy s.t.*

$$\pi^* \circ \phi^* = \arg\max_{\pi \circ \phi} J_{D_{\Sigma,n}}(\pi \circ \phi),$$

where $J_{D_{\Sigma,n}}(\pi \circ \phi)$ is given by Equation (2). Then, ϕ^* satisfies:

$$\forall \mathcal{A}, \mathcal{A}' \in D_{\Sigma,n}, \quad \mathcal{A} \sim \mathcal{A}' \iff \phi^*(\mathcal{A}) = \phi^*(\mathcal{A}').$$

Proof By Theorem 2, $d^*(\mathcal{A}, \mathcal{A}') = \|\hat{\phi}^*(\mathcal{A}) - \hat{\phi}^*(\mathcal{A}')\|_2$ is a bisimulation metric. Therefore, $d^*(\mathcal{A}, \mathcal{A}') = 0 \implies \mathcal{A} \sim_{\mathcal{M}_{D_{\Sigma,n}}} \mathcal{A}'$ by Definition 9 and thus, by Lemma 1, we have $\mathcal{A} \sim \mathcal{A}'$. As $d^*(\mathcal{A}, \mathcal{A}') = 0$ implies $\phi^*(\mathcal{A}) = \phi^*(\mathcal{A}')$, we have $\mathcal{A} \sim \mathcal{A}' \iff \phi^*(\mathcal{A}) = \phi^*(\mathcal{A}')$. The forward direction is true since if $\mathcal{A} \sim \mathcal{A}'$, then $d^*(\mathcal{A}, \mathcal{A}') = 0$ by Definition 9; thus, $\phi^*(\mathcal{A}) = \phi^*(\mathcal{A}')$. ■

Theorem 3 *Let $D_{\Sigma,n}$ be a DFA space, ϕ be an encoder, and π^* be a policy s.t.*

$$\pi^* \circ \phi^* = \arg\max_{\pi \circ \phi} J_{D_{\Sigma,n}}(\pi \circ \phi),$$

where $J_{D_{\Sigma,n}}(\pi \circ \phi)$ is given by Equation (2). Then, ϕ^* solves Problem 1.

Proof The optimal encoder ϕ^* maps two DFAs to the same latent representation if and only if they are bisimilar by Lemma 2. So, every unique task in $D_{\Sigma,n}$ is represented in $\mathcal{Z}$. Therefore, the problem given in Definition 6 can be equivalently reformulated over $\mathcal{Z}$, which solves Problem 1. ■