Automaton Distillation: Neuro-Symbolic Transfer Learning for Deep Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

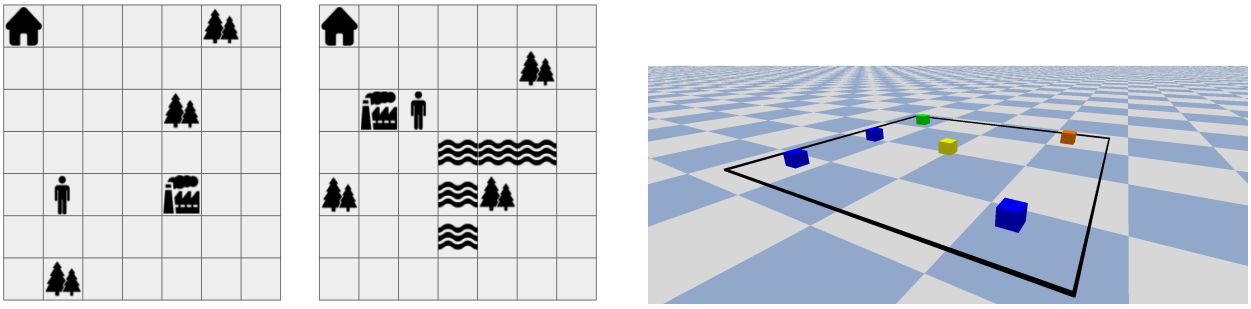
Reinforcement learning (RL) is a powerful tool for finding optimal policies in sequential decision processes. However, deep RL methods have two weaknesses: collecting the amount of agent experience required for practical RL problems is prohibitively expensive, and the learned policies exhibit poor generalization on tasks outside the training data distribution. To mitigate these issues, we introduce automaton distillation, a form of neuro-symbolic transfer learning in which Q-value estimates from a teacher are distilled into a low-dimensional representation in the form of an automaton. We then propose methods for generating Q-value estimates where symbolic information is extracted from a teacher's Deep Q-Network (DQN). The resulting Q-value estimates are used to bootstrap learning in the target discrete and continuous environment via a modified DQN and Twin-Delayed Deep Deterministic (TD3) loss function, respectively. We demonstrate that automaton distillation decreases the time required to find optimal policies for various decision tasks in new environments, even in a target environment different in structure from the source environment.

1 Introduction

Sequential decision tasks, in which an agent learns policies to maximize long-term rewards through trial and error, are often solved using reinforcement learning (RL). However, a critical limitation of conventional RL methods is their poor adaptability when faced with even minor changes in task objectives or environmental dynamics. This limitation is particularly problematic in real-world applications, such as robotics, where environments and tasks frequently evolve, requiring substantial retraining and resulting in high sample complexity and inefficient adaptation. This issue becomes even more pronounced for tasks characterized by sparse, non-Markovian rewards or tasks with long-term action dependencies, scenarios known to challenge standard RL approaches significantly.

Humans, in contrast, efficiently adapt to new situations by leveraging abstract concepts and generalizing knowledge from prior experiences to unfamiliar scenarios. Traditional deep learning methods, lacking explicit abstraction capabilities, struggle to replicate such human-like adaptability. Recently, neuro-symbolic computing has emerged as a promising integration of symbolic reasoning and neural approaches, enabling effective knowledge transfer, enhanced interpretability, and improved generalization across unseen tasks (Tran & Garcez, 2016; Verma et al., 2018; Anderson et al., 2020). For example, consider a mobile robot deployed in an office tasked with delivering personalized coffee orders. The robot must efficiently adapt knowledge from delivering coffee in a conference room to serving coffee in an office hall, a seemingly minor shift that typically demands significant retraining under traditional RL paradigms.

Non-Markovian Reward Decision Processes (NMRDPs) Bacchus et al. (1996); Littman et al. (2017) have successfully modeled tasks with sparse or non-Markovian rewards by augmenting state representations to encode temporal or historical context, making the problem effectively Markovian. Such augmented representations can naturally integrate symbolic approaches, particularly automata, to represent task objectives clearly and compactly (Camacho et al., 2017; Gaon & Brafman, 2020; De Giacomo et al., 2013). Despite these advances, enabling efficient transfer learning for NMRDPs remains challenging, especially when trans-



(a) (b) (c)

Figure 1: Example environment configurations for the Blind Craftsman teacher (a) and student (b) environments with additional obstacles introduced. (c) A continuous state and action student environment where the yellow, blue, green, and orange cubes represent the agent, wood, factory, and home, respectively, positioned at random continuous positions.

ferring knowledge across domains with differing state-action spaces or environmental dynamics. We focus on improving *few-shot* transfer capabilities, where an agent leverages prior experience from related tasks to quickly adapt to a new task with different dynamics and characteristics, minimizing the training time needed to reach near-optimal performance. A central challenge in this setting is how to effectively represent rewards in a way that both enables transfer and supports policy optimization. Symbolic representations, such as reward machines, have been used to capture task structure, where transitions in the reward machine encode high-level progress across subtasks toward completing the overall objective (Icarte et al., 2022; Camacho et al., 2018).

In this work, we propose a strategy that leverages symbolic representations of RL objectives to facilitate knowledge transfer from an expert agent trained in a related source domain (the "teacher") to another agent learning a target task (the "student"). While reward functions commonly express task objectives, many decision-making problems are more naturally described through high-level intermediate steps articulated in natural language. Such descriptions can be translated into formal languages like linear temporal logic (LTL) (Brunello et al., 2019), which can be converted into an equivalent automaton representation (Wolper et al., 1983). For tasks sharing common goals, this automaton provides a unified symbolic language through which states and actions in the source and target domains correspond to automaton nodes and transitions. Furthermore, assigning value estimates to automaton transitions converts this representation into a compact, abstract model, significantly facilitating the learning process.

We introduce two variants of transfer learning that, to the best of our knowledge, have not been previously explored in the literature. These variants leverage the *automaton representation of an objective* to convey information about the reward signal from the teacher to the student. The first, static transfer, generates estimates of the Q-value of automaton transitions by performing value iteration over the abstract Markov Decision Process (MDP) defined by the automaton. The second variant, dynamic transfer, distills knowledge from a teacher Deep Q-Network (DQN) (Mnih & et al., 2015) into the automaton by mapping teacher Q-values of state-action pairs in the experience replay buffer to their corresponding transition in the automaton.

We evaluate our proposed methods across multiple environments with varying dynamics (such as that presented in Figure 1), demonstrating significant performance improvements over standard RL and baseline methods. Notably, our experiments include challenging scenarios where the student environment contains novel features absent in the teacher's domain or when transferring knowledge from discrete to continuous state-action spaces. Results highlight our method's ability to achieve superior performance and faster convergence, particularly emphasizing the benefits of symbolic abstraction for rapid adaptation in complex, dynamic real-world tasks.

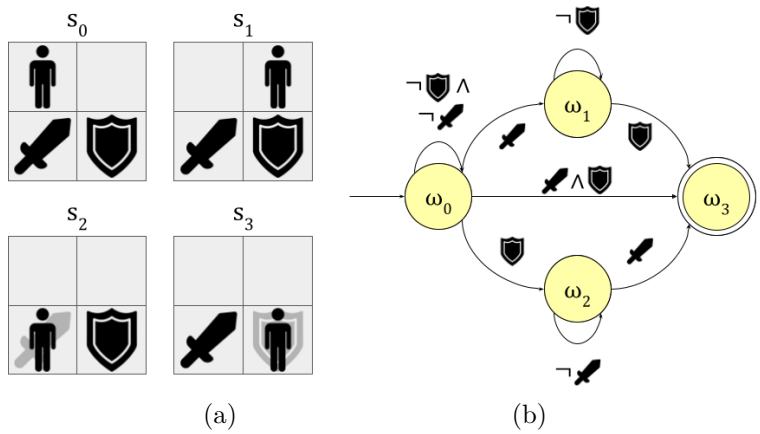


Figure 2: (a) A simple NMRDP. At each time step, the agent may move one square in any cardinal direction. A sequence of actions satisfies the objective if and only if the agent obtains both the sword and the shield. The objective is decomposed using the atomic propositions $AP = \{\text{sword}, \text{shield}\}$, with a labeling function L such that $L(s_0) = \{\}, L(s_1) = \{\}, L(s_2) = \{\text{sword}\}, L(s_3) = \{\text{shield}\}$. Rollouts which achieve the objective also satisfy the LTL$_f$ specification $\phi = \mathbf{F}(\text{sword}) \wedge \mathbf{F}(\text{shield})$. (b) An automaton defined over the alphabet $\Sigma = \{\{\}, \{\text{sword}\}, \{\text{shield}\}, \{\text{sword, shield}\}\}$. The automaton accepts the subset of strings in Σ^* that satisfy the LTL$_f$ formula.

2 Preliminaries

We model the teacher and student decision processes as a NMRDP defined below.

Definition 1 (Non-Markovian Reward Decision Process (NMRDP))**.** *An NMRDP is a decision process defined by the tuple $\mathcal{M} = \langle S, s_0, A, T, R \rangle$, where S is the set of valid states, $s_0 \in S$ is the initial state, A is the set of valid actions, $T : S \times A \times S \to [0, 1]$ is a transition function defining transition probabilities for each state-action pair to every state in S, and $R : (S \times A)^* \to \mathbb{R}$ defines the reward signal observed at each time step based on the sequence of previously visited states and actions.*

NMRDPs differ from MDPs in that the reward signal R may depend on the entire history of observations rather than only the current state. However, the reward signal is often a function of a set of abstract properties of the current state, which is of much smaller dimension than the original state space. Thus, it can be beneficial to represent the reward signal in terms of a simpler vocabulary defined over features extracted from the state. We assume the existence of a set of atomic propositions AP for each environment, which capture the dynamics of the reward function, as well as a labeling function, $L : S \to 2^{AP}$, that translates experiences into truth assignments for each proposition $p \in AP$. In cases where such a labeling function does not explicitly exist, it is possible to find one automatically (Hasanbeig et al., 2021). Using these atomic propositions, NMRDP objectives, assumed to be the same in the teacher and student environment, can be succinctly expressed as formal specifications. A particularly convenient representation is a Deterministic Finite-State Automaton (DFA), formally defined as follows:

Definition 2 (Deterministic Finite-State Automaton (DFA))**.** *A DFA is an automaton defined by the tuple $\mathcal{A} = \langle \Sigma, \Omega, \omega_0, F, \delta \rangle$, where Σ is the alphabet of the input language, Ω is the set of states with starting state ω_0, $F \subseteq \Omega$ is the set of accepting states, and $\delta : \Omega \times \Sigma \to \Omega$ defines a state transition function.*

The atomic propositions AP comprise a vocabulary of abstract properties of the state space which directly correspond to the reward structure. Using the labeling function L, states in an NMRDP can be mapped to an element in the alphabet $\Sigma = 2^{AP}$. Then, the set of rollouts which satisfy the objective constitute a regular language over Σ. The parameters $\Omega, \omega_0, F, \delta$ are chosen such that the set of strings accepted by the objective automaton $\mathcal{A}$ is equivalent to the aforementioned regular language; we illustrate this with a simple example in Figure 2. An additional consequence of developing such a vocabulary is that RL objectives can be expressed as a regular language and subsequently converted into a DFA.

Remark 1. *One benefit of the automaton representation is the ability to express non-Markovian reward signals in terms of the automaton state. During an episode, the state of the automaton can be computed in parallel with observations from the learning environment.*

Given the current automaton state ω and a new observation s', the new automaton state can be computed as $\omega' = \delta(\omega, L(s'))$. We assume that the atomic propositions capture the non-Markovian behavior of the reward signal, thus, the system dynamics are Markovian in the cross-product of the observation and automaton state spaces (De Giacomo et al., 2019). Formally, we represent the cross-product of an NMRDP and its corresponding objective automaton as an MDP.

Definition 3 (Cross-Product Markov Decision Process)**.** *The cross-product of an NMRDP* $\mathcal{M} = \langle S, s_0, A, T, R\rangle$ *and a DFA* $\mathcal{A} = \langle \Sigma, \Omega, \omega_0, F, \delta\rangle$ *which captures the non-Markovian behavior of the reward signal is a Markov Decision Process (MDP)* $\mathcal{M}_{prod} = \langle S \times \Omega, (s_0, \omega_0), A, T \times \delta, R'\rangle$ *where* $R' : \Omega \times \Sigma \to \mathbb{R}$ *is a Markovian reward signal (i.e., can be expressed as a function of only the current state and action).*

Note that transforming an NMRDP into a cross-product MDP permits the use of traditional RL algorithms, which rely upon the Markovian assumption, for decision tasks with non-Markovian reward signals.

Implicit in the previous discussion is that the objective is translated into an automaton prior to learning. The difficulty of explicitly constructing an automaton to represent a desired objective has motivated the development of automated methods for converting a reward specification into an automaton representation. Such methods observe a correspondence between formal logics (such as regular expressions, LTL, and its variants) and finite-state automata (Wolper et al., 1983). In particular, finite-trace Linear Temporal Logic (LTL_f) has been used to represent RL objectives (Camacho et al., 2019; Velasquez et al., 2021).

Definition 4 (Finite-Trace Linear Temporal Logic (LTL_f))**.** *A formula in* LTL_f *consists of a set of atomic propositions AP which are combined by the standard propositional operators and the following temporal operators: the next operator* $\boldsymbol{X}\phi$ *(*ϕ *will be true in the next time step), the eventually operator* $\boldsymbol{F}\phi$ *(*ϕ *will be true in some future time step), the always operator* $\boldsymbol{G}\phi$ *(*ϕ *will be true in all future time steps), the until operator* $\phi_1 \ \boldsymbol{U} \ \phi_2$ *(*ϕ_2 *will be true in some future time step, and until then* ϕ_1 *must be true), and the release operator* $\phi_1 \ \boldsymbol{R} \ \phi_2$ *(*ϕ_2 *must be true always or until* ϕ_1 *first becomes true).*

It has been shown that specifications in LTL_f can be transformed into an equivalent deterministic finite automaton (De Giacomo & Vardi, 2015), and tools for compiling automata are readily available (Zhu et al., 2017). Moreover, it is possible, in principle, to convert descriptions using a predefined subset of natural language into LTL (Brunello et al., 2019). Thus, it is feasible to translate a specification provided by a domain expert into an automaton representation using automated methods.

In this paper, we leverage this symbolic automaton representation specifically for knowledge transfer in RL. Given a source and a target environment with similar objectives but that differ in dynamics or state-action characteristics, our transfer learning approach uses DFAs to abstract and transfer task-relevant knowledge explicitly. Notably, we assume that the LTL_f specifications used to generate DFAs are provided externally, either derived from expert domain knowledge or automatically synthesized from natural language inputs Fuggitti & Chakraborti (2023); Brunello et al. (2019). Our primary transfer learning objectives are inspired by the metrics (Taylor & Stone, 2009):

- Time to Threshold (TT): Number of training steps required to reach an acceptable performance threshold in target environments, highlighting few-shot transfer efficiency.
- Transfer Ratio (TR): The ratio of the cumulative rewards obtained by an agent leveraging transferred knowledge to those obtained by an agent trained from scratch. It quantifies the performance gain from knowledge transfer.

3 Automaton Distillation

In this section, we introduce our knowledge transfer approach. First, we train a DQN expert in the teacher environment. Next, we distill the expert's Q-values into an automaton that encapsulates the task objective,

associating each transition's value with an estimated Q-value for corresponding state-action pairs in the teacher environment. Finally, this automaton guides the student during training, enabling effective transfer of expertise. We refer to this methodology as *"automaton distillation"* since it involves extracting insights from a teacher automaton to enhance learning in the student environment.

Dynamic Transfer. Our primary focus lies in a dynamic transfer learning algorithm that distills value estimates from an agent trained with Deep Q-learning into the objective automaton. This process uses a contractive mapping from the teacher NMRDP's state-action pairs to the abstract MDP defined by the automaton. Additionally, an analogous expansive mapping from the abstract MDP to the student NMRDP provides an initial Q-value estimate for state-action pairs in the target domain, assisting the student's learning process. The automaton mediates between the teacher and student domains, enabling experience sharing without requiring manual state-space mappings (Taylor & Stone, 2005) or unsupervised map learning (Ammar et al., 2015).

The *teacher* DQN is trained using only standard RL methods (Wang et al., 2016b; Van Hasselt et al., 2016; Schaul et al., 2015). However, to track both the current node in the automaton and the environment state, we store samples of the form $((s,\omega),a,r,(s',\omega'))$ in the experience replay ER buffer. We define $\eta_{\text{teacher}} : \Omega \times \Sigma \to \mathbb{N}$ as the number of times each automaton node ω and a set of atomic propositions $\sigma \in 2^{AP}$ appear in the experience replay ER of the teacher (note that ω and σ define a transition in the automaton objective as given by $\delta(\omega,\sigma)=\omega'$):

$$\eta_{\text{teacher}}(\omega,\sigma) = |\{((s,\omega),a,r,(s',\omega')) \in ER | L(s') = \sigma\}|. \tag{1}$$

Similarly, we define $Q^{\text{avg}}_{\text{teacher}} : \Omega \times \Sigma \to \mathbb{R}$ to be the average Q-value corresponding to the automaton transition given by ω and $\sigma \in 2^{AP}$, according to the teacher DQN:

$$Q^{\text{avg}}_{\text{teacher}}(\omega,\sigma) = \frac{\sum_{\{((s,\omega),a,r,(s',\omega')) \in ER | L(s')=\sigma\}} Q_{\text{teacher}}((s,\omega),a)}{\eta_{\text{teacher}}(\omega,\sigma)}. \tag{2}$$

where $Q_{\text{teacher}}((s,\omega),a)$ is the Q-value learned by the teacher DQN at augmented state (s,ω) and action a.

In DQN and actor-critic networks, target networks are critical for stabilizing training by providing fixed targets for temporal difference (TD) updates. These networks are typically updated at a slower rate or through a smoothing process to avoid instability caused by rapidly changing Q-values.

In this paper, we incorporate the teacher's knowledge into the student's learning by modifying the *student's target* Q-value:

$$Q'_{\text{student}}((s,\omega),a) = \beta(\omega, L(s'))\, Q^{\text{avg}}_{\text{teacher}}(\omega, L(s')) + \big(1 - \beta(\omega, L(s'))\big) Q_{\text{target}}. \tag{3}$$

Here, $\beta : \Omega \times \Sigma \to [0,1]$ controls the influence of the teacher's knowledge, $Q^{\text{avg}}_{\text{teacher}}(\omega, L(s'))$ is the teacher's average Q-value for transition $(\omega, L(s'))$, and Q_{target} is the standard target Q-value. This approach applies to both DQN and actor-critic algorithms like TD3 (Fujimoto et al., 2018), where Q_{target} in equation 3 is defined as follows. For DQN,

$$Q_{\text{target}} = r + \gamma \max_{a'} Q_{\text{student}}((s',\omega'),a';\theta^{\text{target}}), \tag{4}$$

and for TD3,

$$Q_{\text{target}} = r + \gamma \min_{i=1,2} Q_{\text{student}}\big((s',\omega'),\tilde{a};\theta^{\text{target}}_i\big). \tag{5}$$

Here, $r := R'(\omega,\sigma)$ is the reward received within the student environment during transition $\omega \xrightarrow{\sigma} \omega'$ in the product MDP, γ is the discount factor, s' is the next state, Q_{student} is the student's Q-value function, θ^{target} represents the parameters of the target network, a' and $\tilde{a} = \pi_{\phi'}(s') + \varepsilon$ are the action suggested by the target policy network at state s', where ε is the exploration noise.

The annealing function is defined as $\beta(\omega,\sigma) = \rho^{\eta_{\text{student}}(\omega,\sigma)}$, where $\rho = 0.999$ and $\eta_{\text{student}}(\omega,\sigma)$ is the number of times the transition (ω,σ) has been sampled during student training.

The loss function for updating the student's network is:

$$\text{Loss}(\theta) = \mathbb{E}_{((s,\omega),a,r,(s',\omega'))\sim P(ER)}[Q'_{\text{student}}((s,\omega),a) - Q((s,\omega),a;\theta)]^2, \tag{6}$$

where $Q((s,\omega),a;\theta)$ is the student's Q-value prediction, and P is a priority function favoring samples, from the student's ER, with higher prediction errors.

This method performs non-Markovian knowledge transfer, as the automaton compactly represents environment dynamics and encodes the non-Markovian reward signal. This versatility enables our method to facilitate knowledge transfer not only between discrete environments but also from discrete to continuous environments, as demonstrated in our experiments. Our approach enhances the student's learning by integrating the teacher's knowledge directly into the target Q-value. This allows the student to benefit from the teacher's expertise while adapting to its own environment, leading to faster convergence and improved performance in complex tasks.

The asymptotic behavior of automaton Q-learning depends on the annealing function β in the student Q-udpate in equation 3. When $\beta = 0$, automaton Q-learning reduces to vanilla Q-Learning. Next, we establish a convergence result for tabular automaton Q-learning.

Theorem 1 (Convergence of Automaton Q-Learning)**.** *Let the Q-values be updated in the product MDP $\mathcal{M}_{prod} = \langle S \times \Omega, (s_0, \omega_0), A, T \times \delta, R' \rangle$, where $R' : \Omega \times \Sigma \to \mathbb{R}$ is the reward function defined over automaton transitions. Define $\sigma_t = L(s_{t+1})$ and $r_t := R'(\omega_t, \sigma_t)$ as the reward observed during step t, corresponding to the transition $\omega_t \xrightarrow{\sigma_t} \omega_{t+1} = \delta(\omega_t, \sigma_t)$. Let the Q-values be updated as:*

$$\begin{aligned} Q_{t+1}((s_t,\omega_t),a_t) = (1-\alpha_t)Q_t((s_t,\omega_t),a_t) + \alpha_t\beta_t Q^{avg}_{teacher}(\omega_t,\sigma_t) \\ + \alpha_t(1-\beta_t)\left[r_t + \gamma V_t(s_{t+1},\omega_{t+1})\right], \end{aligned} \tag{7}$$

where $V_t(s,\omega) = \max_a Q_t((s,\omega),a)$. Then, $Q_t((s,\omega),a) \to Q^*((s,\omega),a)$ with probability 1 under the following conditions:

1. *The state space S, automaton state space Ω, and action space A are finite.*
2. *The learning rates $\alpha_t \in [0,1)$ satisfy $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$.*
3. *The distillation weights $\beta_t \geq 0$ satisfy $\lim_{t\to\infty} \beta_t = 0$ and $\sum_t \alpha_t(1-\beta_t) = \infty$.*
4. *The symbolic reward variance $\text{Var}(r_t)$ is uniformly bounded.*
5. *Either $\gamma = 1$ and all policies reach a cost-free terminal state, or $\gamma \in [0,1)$.*

Proof. We decompose the Q-values as $Q_t = q_t + h_t$ with updates:

$$q_{t+1}((s_t,\omega_t),a_t) = (1-\alpha_t)q_t((s_t,\omega_t),a_t) + \alpha_t(1-\beta_t)\left[r_t + \gamma V_t(s_{t+1},\omega_{t+1})\right], \tag{8}$$

$$h_{t+1}((s_t,\omega_t),a_t) = (1-\alpha_t)h_t((s_t,\omega_t),a_t) + \alpha_t\beta_t Q^{\text{avg}}_{\text{teacher}}(\omega_t,\sigma_t). \tag{9}$$

Convergence of q_t: This update corresponds to Q-learning in the product MDP using rewards $r_t = R'(\omega_t, \sigma_t)$. Since the reward is bounded, the learning rate $\alpha_t(1-\beta_t)$ satisfies Robbins–Monro conditions, and the state-action space is finite, it follows (e.g., Jaakkola et al., 1994) that:

$$q_t((s,\omega),a) \to Q^*((s,\omega),a) \quad \text{w.p. } 1.$$

Convergence of h_t: Let $c_t := Q^{\text{avg}}_{\text{teacher}}(\omega_t, \sigma_t)$. Since c_t is drawn from a fixed and bounded function over the finite domain $\Omega \times \Sigma$, there exists $C < \infty$ such that $|c_t| \leq C$ for all t. The update becomes:

$$h_{t+1} = (1-\alpha_t)h_t + \alpha_t\beta_t c_t.$$

This is a stochastic approximation with a vanishing forcing term $\beta_t c_t$, and under the given assumptions on α_t and β_t, it converges to zero almost surely.

Since $q_t \to Q^*$ and $h_t \to 0$ almost surely, we conclude:

$$Q_t = q_t + h_t \to Q^* \quad \text{w.p. } 1.$$

□

Static Transfer. Static value estimates can be effective when the abstract MDP defined by the automaton accurately captures the environment dynamics. Such estimates can be computed via tabular Q-learning over the abstract MDP:

$$Q(\omega, \sigma) \leftarrow Q(\omega, \sigma) + \alpha(R'(\omega, \sigma) + \gamma \max_{\sigma'} Q(\omega', \sigma') - Q(\omega, \sigma)) \,. \tag{10}$$

The resulting Q-values can be used in the place of $Q^{\text{avg}}_{\text{teacher}}$ in Equation 6. This method has the benefit of stabilizing training in the early stages without requiring a DQN oracle or any additional information beyond the reward structure.

4 Related Work

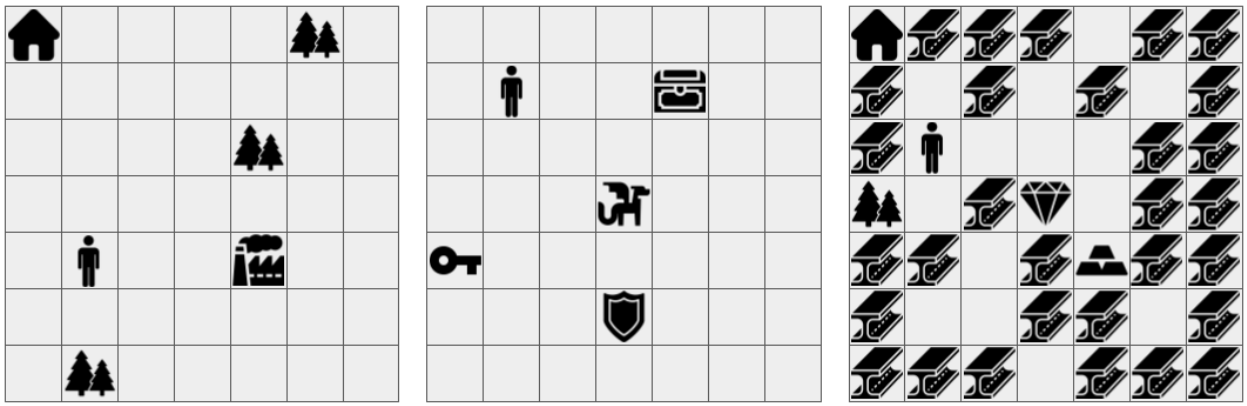
Deep RL has made remarkable progress in many practical problems, such as recommendation systems, robotics, and autonomous driving. However, despite these successes, insufficient data and poor generalization remain open problems. In most real-world problems, it is difficult to obtain training data, so RL agents often learn with simulated data. However, RL agents trained with simulated data usually have poor performance when transferred to unknown environment dynamics in real-world data. To address these two challenges, transfer learning techniques (Zhu et al., 2023) have been adopted to solve RL tasks.

Among transfer learning techniques, Domain Adaptation (DA) is the most well-studied in deep RL. Early attempts in DA constructed a map from states and actions in the source domain onto the target domain by hand (Taylor & Stone, 2005). Subsequent efforts aimed to learn a set of general latent environment representations that can be transferred across domains such as inter-task mapping and representation reuse (Ammar & Taylor, 2012), and learning disentangled representations (Higgins et al., 2017; Srinivas et al., 2020; Xing et al., 2021; Yi et al., 2023). Additionally, integrating meta-learning, often referred to as "learning to learn", with domain adaptation has enabled agents to rapidly adjust to novel tasks or environments using minimal data (Finn et al., 2017; Wang et al., 2016a). These methods rely on extra layer of exploitation to learn the domain before knowledge transfer.

In contrast to DA, methods like traditional policy distillation (Rusu et al., 2015), where knowledge is directly transferred from the teacher to the student (Jin et al., 2024; Yang et al., 2024), use a direct approach in transferring knowledge. While this approach emphasizes directly copying behaviors, some recent works focus on constructing an intermediate, abstract representation of the environment. For example, high-level symbolic domain descriptions have been used to build low-dimensional abstractions of the original state space (Kokel et al., 2022), which can be used to model the dynamics of the original system. Also, (Icarte et al., 2022) further constructs an automaton which realizes the abstract decision process and uses the automaton to convey information about the reward signal. These methods focused on single-task while in Qiu et al. (2023) task-oriented and goal-conditioned within a multitasking framework were learned.

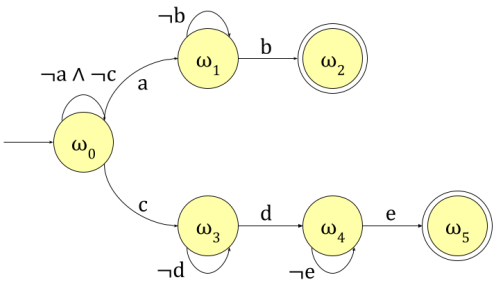
Static transfer learning methods based on reward machines were developed in (Camacho et al., 2018; Icarte et al., 2022). By treating the nodes and edges in the automaton as states and actions, respectively, the automaton can be transformed into a low-dimensional abstract MDP which can be solved using Q-learning or value iteration approaches. The solution to the abstract MDP can speed up learning in the original environment using a potential-based reward shaping function (Camacho et al., 2019), by introducing counterfactual experiences during training (Icarte et al., 2022; Voloshin et al., 2023) or by providing contextual guidance derived from reward structures learned across diverse training contexts. Azran et al. (2024).

However, static transfer methods perform poorly when the abstract MDP fails to capture the behavior of the underlying process. Consider applying the static transfer approach proposed in (Icarte et al., 2022) for the objective defined by the LTL_f formula $\phi = \mathbf{F}(\text{b} \vee \text{e}) \wedge (\neg\mathbf{F}(\text{a}) \vee \neg\mathbf{F}(\text{c})) \wedge (\text{a } \mathbf{R} \neg\text{b}) \wedge (\text{c } \mathbf{R} \neg\text{d}) \wedge (\text{d } \mathbf{R} \neg\text{e})$, whose automaton is given in Figure 4a.

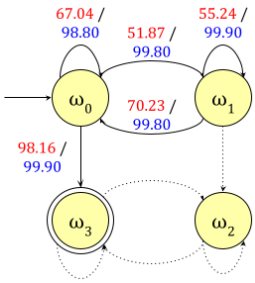


(a) (b) (c)

Figure 3: Example 7×7 environment configurations for the *Blind Craftsman* (a), *Dungeon Quest* (b), and *Diamond Mine* (c) environments.



(a) Simple automaton with two traces.



(b) Teacher Q-values produced by dynamic (red) and static (blue) automaton distillation on the *Blind Craftsman* environment.

Assume that the reward function grants a reward of 1 for transitions leading to either terminal state and a reward of 0 for all other transitions. (Icarte et al., 2022) perform value iteration over the abstract MDP with update

$$V(\omega) := \max_{\omega' = \delta(\omega, \sigma)} R(\omega, \sigma) + \gamma V(\omega'). \tag{11}$$

As can be seen in the automaton in Figure 4a, there are two traces which satisfy the objective: one of length 2 and one of length 3. Due to the discount factor $\gamma < 1$, value iteration will favor taking transition a, which has a shorter accepting path, over transition c in the starting state. However, it may be the case that observing b after observing a takes many steps in the original environment, and thus the longer trace $c \to d \to e$ takes less steps to reach an accepting state.

In contrast to static transfer, which uses prior knowledge of the reward function to model the behavior of the target process, dynamic transfer leverages experience acquired by interaction in a related domain to empirically estimate the target value function. Dynamic transfer has the advantage of implicitly factoring in knowledge of the teacher environment dynamics; in the previous example, if the shorter trace $a \to b$ takes more steps in the teacher environment than the longer trace $c \to d \to e$, this will be reflected in the discounted value estimates learned by the teacher.

5 Experimental Results

In this section, we evaluate our automaton distillation approach. For each time step in the student's training, we update our target policy with respect to Q', following the algorithm in 1.

Table 1: An evaluation of automaton distillation and other reward machine transfer approaches. We show that automaton distillation successfully transfers knowledge to the student environment. We show the mean and standard deviation of the transfer ratio averaged over 8 seeds. The italicized figures indicate negative TR. *BlindCraftManObs* refers to the environment blindcraftsman with obstacles

Metric	Transfer-Dynamics	Environment	CRM	CPREP	Static Distill	Dyn Distill
TT_{AUC}	Discrete–Discrete	BlindCraftMan	199835.98 ± 39626.54	237009.07 ± 30064.85	194037.82 ± 9396.00	**111515.92 ± 2818.89**
		BlindCraftManObs	434982.79 ± 122704.47	392687.42 ± 99958.54	186997.95 ± 23397.48	**38668.11 ± 9272.46**
		DungeonQuest	827653.48 ± 119397.18	438406.01 ± 44567.97	375656.36 ± 61651.82	**90682.37 ± 15536.00**
		GoldMine	1000000.0 ± 0.0	1000000.0 ± 0.0	**290313.72 ± 43304.58**	723871.45 ± 296460.19
	Discrete–Continuous	BlindCraftMan	335160.81 ± 42506.67	311190.14 ± 78927.61	175176.02 ± 66449.40	126996.97 ± 45392.95
		BlindCraftManObs	437139.52 ± 53840.96	299458.60 ± 63050.75	**154929.25 ± 53437.07**	216010.25 ± 51547.48
		DungeonQuest	213265.48 ± 41918.38	283279.45 ± 35904.54	149336.94 ± 71913.11	**196528.84 ± 54569.72**
		GoldMine	1000000.0 ± 0.0	1000000.0 ± 0.0	588773.29 ± 123336.65	**469081.81 ± 71926.05**
TR	Discrete–Discrete	BlindCraftMan	10.17 ± 0.017	10.93 ± 0.02	9.83 ± 0.03	**12.50 ± 0.02**
		BlindCraftManObs	*−1.06 ± 0.03*	0.24 ± 0.2	2.45 ± 0.02	**3.013 ± 0.03**
		DungeonQuest	*−3.54 ± 0.11*	19.87 ± 0.13	20.58 ± 0.10	**43.54 ± 0.39**
		GoldMine	*−3.86 ± 0.15*	*−2.76 ± 0.10*	**0.90 ± 0.16**	0.86 ± 0.16
	Discrete–Continuous	BlindCraftMan	*−0.05 ± 0.048*	0.29 ± 0.062	0.94 ± 0.022	**1.073 ± 0.05**
		BlindCraftManObs	0.43 ± 0.058	0.37 ± 0.02	**1.22 ± 0.030**	0.882 ± 0.067
		DungeonQuest	0.88 ± 0.028	1.07 ± 0.045	1.28 ± 0.05	**1.36 ± 0.056**
		GoldMine	*−0.45 ± 0.001*	*−0.45 ± 0.001*	0.009 ± 0.020	**0.20 ± 0.046**

5.1 Experimental Setup

We evaluate our algorithm on three environments, continuous and discrete environments, with long-horizon and sparse reward signals.

Blind Craftsman: This environment consists of woods, a factory, and a home; obstacles can be added for extra difficulty. The objective is satisfied when the agent has crafted three tools and arrived home. One wood is required to craft a tool. However, since the agent can only carry two pieces of wood at a time, the agent must alternate between collecting wood and crafting tools. The objective is defined over the atomic propositions $AP = \{\text{wood}, \text{factory}, \text{tools} \geq 3, \text{home}\}$ and given by the LTL_f formula $\phi = \mathbf{G}(\text{wood} \implies \mathbf{F}\ \text{factory}) \wedge \mathbf{F}(\text{tools} \geq 3\ \wedge\ \text{home})$ with a corresponding automaton of 4 nodes and 12 transitions.

Dungeon Quest: This environment consists of a key, a chest, a shield, and a dragon. The agent can acquire a key and a shield by interacting with a key or shield, respectively. Additionally, the agent can obtain a sword by interacting with a chest with a key in its inventory. Once the agent has the sword and the shield, it may interact with the dragon to defeat it and complete the objective. The objective is defined over the atomic propositions $AP = \{\text{key}, \text{shield}, \text{sword}, \text{dragon}\}$ and given by the LTL_f formula $\phi = \mathbf{F}(\text{dragon}) \wedge (\text{key}\ \mathbf{R}\ \neg\text{sword}) \wedge (\text{sword}\ \mathbf{R}\ \neg\text{dragon}) \wedge (\text{shield}\ \mathbf{R}\ \neg\text{dragon})$ with a corresponding automaton of 7 nodes and 17 transitions.

Gold Mine: This environment consists of a wood tile, a diamond tile, GoldMine tiles, and iron tiles. The agent may acquire wood, iron, or GoldMine by interacting with the respective objects. Once the agent has collected wood and 30 iron, it automatically crafts a pickaxe. The agent may then obtain diamond by interacting with the diamond while holding a pickaxe. Once the agent has acquired either 1 diamond or 10 GoldMine, it may return to home to complete the objective. To simplify the resulting automaton and limit unnecessary reward, once the agent has collected GoldMine, it cannot obtain the diamond, and vice versa. Intuitively, although collecting diamond requires less automaton transitions, collecting 30 iron is relatively time-consuming. Thus, this environment represents a scenario where the objective automaton misrepresents the difficulty of the task. The objective is defined over the atomic propositions $AP = \{\text{wood}, \text{diamond}, \text{GoldMine} = 1, \text{GoldMine} = 2, ..., \text{GoldMine} = 10, \text{home}\}$ and given by the LTL_f formula $\phi = \mathbf{F}(\text{home}) \wedge (\neg\mathbf{F}(\text{GoldMine} = 1) \vee \neg\mathbf{F}(\text{wood})) \wedge (\text{wood}\ \mathbf{R}\ \neg\text{diamond}) \wedge (\text{GoldMine} = 1\ \mathbf{R}\ \neg\text{GoldMine} = 2) \wedge ... \wedge (\text{GoldMine} = 9\ \mathbf{R}\ \neg\text{GoldMine} = 10) \wedge ((\text{diamond} \vee \text{GoldMine} = 10)\ \mathbf{R}\ \neg\text{home})$ with a corresponding automaton of 15 nodes and 29 transitions.

The source domains is randomly generated maps 7×7 as in Figure 3, and the target domains, either an independently generated 10×10 map or an environment with *width* and *height* within a continuous range of $[7\,\text{m}, 7\,\text{m}]$ as shown in fig. 1(c). For the grid-world, agents can obtain objects by being on the object tile. On the other hand, objects are collected in a continuous environment when their Euclidean distance to the object is $< 0.25\,\text{m}$. The agent receives a reward of $+1$ for collecting each item, $+100$ for expecting

the final task, and a -0.1 per time step, an additional -0.1 for going out of the boundary in the continuous environment.

Each agent is represented by a Dueling DQN (Wang et al., 2016b) or TD3 (Fujimoto et al., 2018).To learn the policy in the discrete environment, the network consists of a convolutional feature extractor and separate value and advantage heads. The feature extractor is a residual network with 3 residual blocks, each using a 3×3 convolutional kernel with 32 filters and Leaky ReLU activation. The resulting feature map is flattened and split into equal halves, which are fed separately to the value and advantage heads. Each head contains a single fully connected layer with 1 and # *actions* nodes, respectively. Q-values are reconstructed by re-centering advantages to have a mean equal to the output of the value head. The neural network takes as input a stack of 2D grids. Each layer in this stack represents a distinct entity type, which includes the agent, tile types, or inventory item types. Inventory items are represented through constant-valued input planes.

In the continuous environment, the network consists of a two-layer feedforward neural network of 800 and 600 hidden nodes, respectively, with rectified linear units (ReLU) between each layer for both the actor and critic and a final tanh unit following the output of the actor. The network receives a compact vector comprising the 2D position of the agent, velocity components, and current inventory counts. This vector is generated by the underlying Box2D physics engine that simulates realistic environmental dynamics.

Additionally, as in (Icarte et al., 2022), we incorporate the automaton state into the input, training on elements of the cross-product MDP state space $(s, \omega) \in S \times \Omega$. The objective automaton is generated using the Python FLLOAT synthesis tool based on the LTL$_f$ behavioral specification; automata for each environment are shown in Figure 5. During training episodes, the automaton state is continuously tracked and stored alongside each experience in the replay buffer. It is then represented as a one-hot vector and concatenated to the network input, enriching the network's representation of the current state.

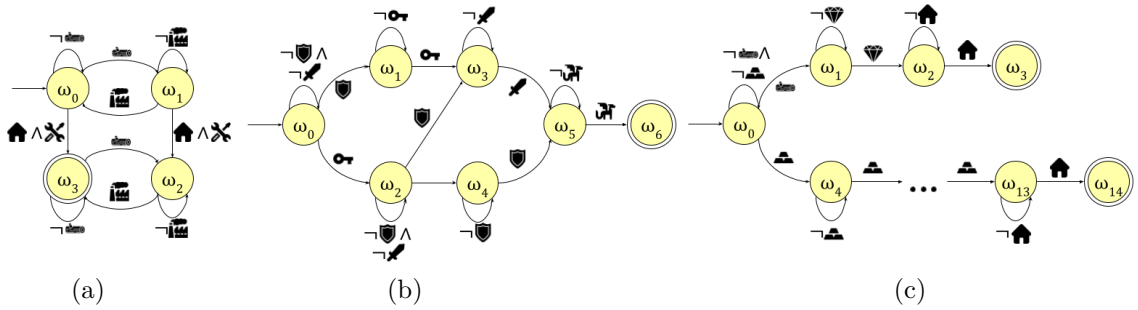


Figure 5: Objective automata for the *Blind Craftsman* (a), *Dungeon Quest* (b), and *Diamond Mine* environments.

5.2 Evaluation

During each training session, the optimal source policy is identified, and its knowledge is distilled into the automaton as per Equation 2. Following Algorithm 1, the target policy is evaluated in eight parallel experiments, each initialized with a distinct random seed. The average reward from these experiments, illustrated in Figure 6, is subsequently used to calculate the transfer metrics (TT and TR). Additionally, these metrics are summarized in Table 1, reporting the mean, standard deviation and 95% confidence intervals to quantify performance variability across the trials.

5.2.1 Evaluation Metrics

We define the training history under policy π as $h_\pi(t)$, representing the expected return at timestep t. Using this history, we compute the following transfer evaluation metrics:

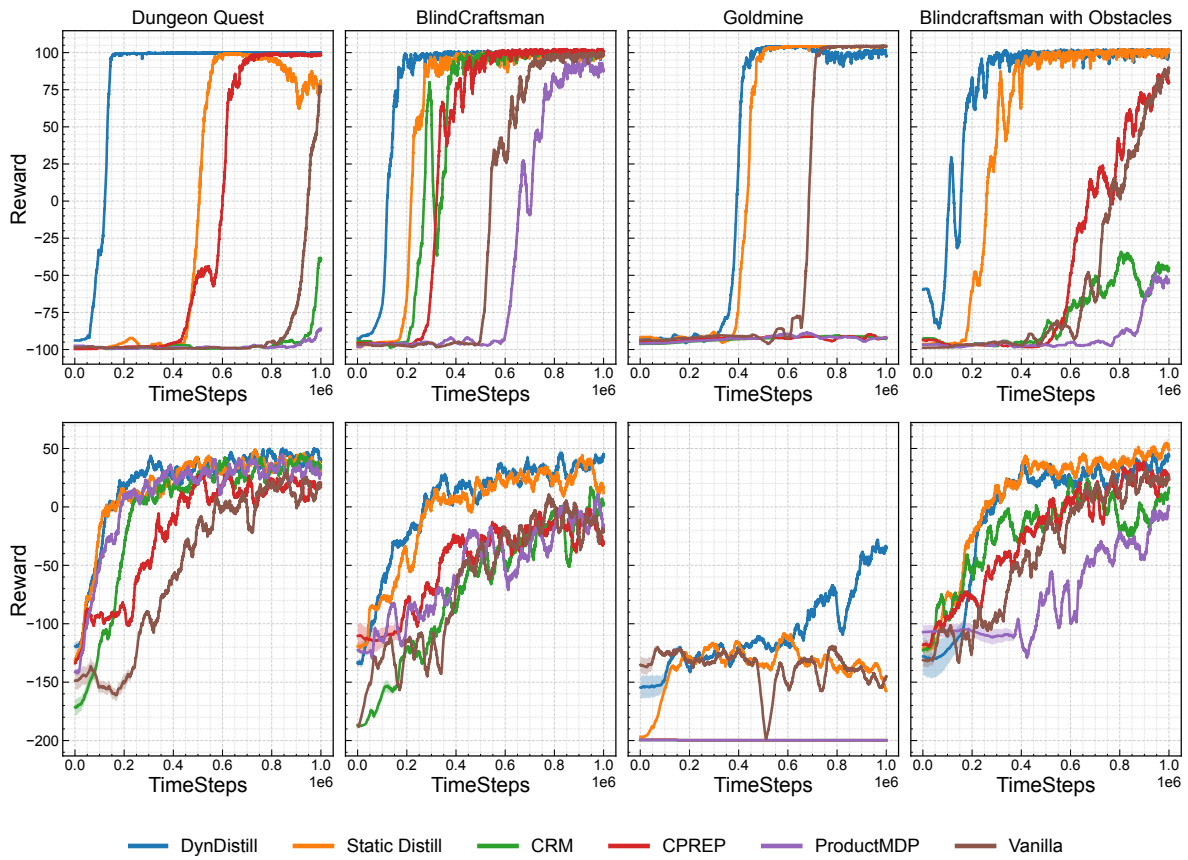


Figure 6: Reward per episode (y-axis) over time (x-axis) during training. The top row shows results from the discrete student environment, while the bottom row corresponds to the continuous student environment.

$$\begin{aligned} \text{Time to Threshold (TT)} &= \min\{t \mid h_\pi(t) \geq \kappa\} \\ \text{Transfer Ratio (TR)} &= \frac{AUC\big(h_{\pi_{\text{student}}}\big) - AUC\big(h_{\pi_{\text{target}}}\big)}{\big|AUC\big(h_{\pi_{\text{target}}}\big)\big|} \end{aligned} \tag{12}$$

Here, κ is a predefined performance threshold, and t is the timestep at which the expected return $h_\pi(t)$ first meets or exceeds κ. The area under the curve (AUC) is computed as the average of return values along the training curve. In addition to evaluating the Time-to-Threshold (TT) for a single threshold, we define TT_{AUC} as the AUC of the TT values over a set of predefined thresholds. This aggregate metric provides a comprehensive measure of how quickly the policy attains various performance levels.

We compare our method against several benchmarks, including counterfactual experiences for reward machines (CRM), a static transfer method proposed in (Icarte et al., 2022), and CPREP (Azran et al., 2024), a contextual transfer method, and RL over product MDP (i.e. including the automaton state in the DQN/TD3 input) and RL baselines.

As illustrated in Figure 6, our method achieves faster convergence compared to baseline methods. Regarding the TT_{AUC}, our method consistently outperforms alternatives except in specific cases mentioned explicitly. The TR metric demonstrates that our approach effectively avoids negative transfer, an issue observed in certain cases with competing methods.

These findings confirm that our approach enhances transfer performance even when environmental dynamics differ significantly. This improvement is particularly notable in the Gold Mine environment, where the assumption that short automaton traces correlate with shorter trajectories in the original environment is misleading. Dynamic automaton distillation utilizes an empirical estimate of trajectory length. Dynamic automaton distillation utilizes an empirical estimate of trajectory length over the teacher's decision process

and circumvents inaccuracies in the abstract MDP. Thus, dynamic automaton distillation is effective when the optimal policies in the teacher and student environments follow similar automaton traces.

Some behavioral specifications can lead to objective automata with cycles, as evidenced by the *Blind Craftsman* environment. Figure 4b shows the Q-values generated by automaton distillation. Unlike static distillation, dynamic distillation can distinguish between states which are (nearly) equivalent in the abstract MDP by incorporating episode length; states which are further from the goal receive discounted rewards, resulting in smaller Q-values.

Cycles in the automaton do not necessarily lead to infinite reward loops as it is often the case that in the original environment, the cycle may be taken only a finite number of times. While it is possible to construct an automaton without cycles by expanding the state space of the automaton to include the number of cycles taken, the maximum number of cycle traversals must be known *a priori* and incorporated into the objective specification, which may not be possible. Additionally, environments that share an objective may admit different numbers of cycle traversals; thus, cycles offer a compact representation that permits knowledge transfer between environments. However, cycles can aggravate the differences between the abstract MDP and the original decision process, resulting in negative knowledge transfer. In such cases, state-of-the-art transfer methods (Icarte et al., 2022) may actually *increase* training time relative to a naïve learning algorithm.

6 Conclusion

In this paper, we proposed automaton distillation, which leverages symbolic knowledge of the objective and reward structure in the form of formal language, to stabilize and expedite training of reinforcement learning agents.

Value estimates for transitions in the automaton are generated using static (i.e. *a priori*) methods such as value iteration over an abstraction of the target domain or dynamically estimated by mapping experiences collected in a related source domain to automaton transitions. The resulting value estimates are used as initial learning targets to bootstrap the student learning process resulting in faster learning even for a target environment different in structure from the source environment. We illustrate several failure cases of existing automaton-based transfer methods, which exclusively reason over *a priori* knowledge, and argue instead for the use of dynamic transfer. We demonstrate that both static and dynamic automaton distillation reduce training costs and outperform state-of-the-art knowledge transfer techniques.

7 AI Impact Statement

Neuro-symbolic transfer learning via automata combines the adaptability of neural networks with the interpretability of symbolic reasoning, creating a robust AI approach. This synergy enhances knowledge transfer across diverse tasks and structurally different domains, improving generalization and enabling AI systems to learn efficiently from limited data while reasoning about complex relationships and sequential tasks.

By facilitating the transfer of knowledge from discrete to continuous environments, our method helps mitigate the sim-to-real gap, enabling autonomy learned in simulation to be more effectively applied in real-world settings. This capability is crucial in robotics and other fields where simulated training must translate to practical applications. Utilizing automata for knowledge transfer fosters the development of intelligent, adaptable, and interpretable AI systems that can learn rapidly in complex real-world environments.

References

Haitham Bou Ammar and Matthew E Taylor. Reinforcement learning transfer via common subspaces. In *Adaptive and Learning Agents: International Workshop, ALA 2011, Held at AAMAS 2011, Taipei, Taiwan, May 2, 2011, Revised Selected Papers*, pp. 21–36. Springer, 2012.

Haitham Bou Ammar, Eric Eaton, José Marcio Luna, and Paul Ruvolo. Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

Greg Anderson, Abhinav Verma, Isil Dillig, and Swarat Chaudhuri. Neurosymbolic reinforcement learning with formally verified exploration. *Advances in neural information processing systems*, 33:6172–6183, 2020.

Guy Azran, Mohamad H Danesh, Stefano V Albrecht, and Sarah Keren. Contextual pre-planning on reward machine abstractions for enhanced transfer in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10953–10961, 2024.

Fahiem Bacchus, Craig Boutilier, and Adam Grove. Rewarding behaviors. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 1160–1167, 1996.

Andrea Brunello, Angelo Montanari, and Mark Reynolds. Synthesis of ltl formulas from natural language texts: State of the art and research directions. In *26th International Symposium on Temporal Representation and Reasoning (TIME 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

Alberto Camacho, Oscar Chen, Scott Sanner, and Sheila McIlraith. Non-markovian rewards expressed in ltl: guiding search via reward shaping. In *Proceedings of the International Symposium on Combinatorial Search*, volume 8, pp. 159–160, 2017.

Alberto Camacho, Oscar Chen, Scott Sanner, and Sheila A McIlraith. Non-markovian rewards expressed in ltl: Guiding search via reward shaping (extended version). In *GoalsRL, a workshop collocated with ICML/IJCAI/AAMAS*, 2018.

Alberto Camacho, Rodrigo Toro Icarte, Toryn Q Klassen, Richard Anthony Valenzano, and Sheila A McIlraith. Ltl and beyond: Formal languages for reward function specification in reinforcement learning. In *IJCAI*, volume 19, pp. 6065–6073, 2019.

Giuseppe De Giacomo and Moshe Vardi. Synthesis for ltl and ldl on finite traces. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Giuseppe De Giacomo, Moshe Y Vardi, et al. Linear temporal logic and linear dynamic logic on finite traces. In *Ijcai*, volume 13, pp. 854–860, 2013.

Giuseppe De Giacomo, Luca Iocchi, Marco Favorito, and Fabio Patrizi. Foundations for restraining bolts: Reinforcement learning with ltlf/ldlf restraining specifications. In *Proceedings of the international conference on automated planning and scheduling*, volume 29, pp. 128–136, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Francesco Fuggitti and Tathagata Chakraborti. Nl2ltl–a python package for converting natural language (nl) instructions to linear temporal logic (ltl) formulas. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 16428–16430, 2023.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Maor Gaon and Ronen Brafman. Reinforcement learning with non-markovian rewards. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 3980–3987, 2020.

Mohammadhosein Hasanbeig, Natasha Yogananda Jeppu, Alessandro Abate, Tom Melham, and Daniel Kroening. Deepsynth: Automata synthesis for automatic task segmentation in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7647–7656, 2021.

Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. DARLA: Improving zero-shot transfer in reinforcement learning. In *International Conference on Machine Learning*, pp. 1480–1490. PMLR, 2017.

Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022.

Can Jin, Tong Che, Hongwu Peng, Yiyuan Li, Dimitris Metaxas, and Marco Pavone. Learning from teaching regularization: Generalizable correlations should be easy to imitate. *Advances in Neural Information Processing Systems*, 37:966–994, 2024.

Harsha Kokel, Nikhilesh Prabhakar, Balaraman Ravindran, Erik Blasch, Prasad Tadepalli, and Sriraam Natarajan. Hybrid deep reprel: Integrating relational planning and reinforcement learning for information fusion. In *2022 25th International Conference on Information Fusion (FUSION)*, pp. 1–8. IEEE, 2022.

Michael L Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. Environment-independent task specifications via gltl. *arXiv preprint arXiv:1704.04341*, 2017.

Volodymyr Mnih and et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

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.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

Matthew E Taylor and Peter Stone. Behavior transfer for value-function-based reinforcement learning. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 53–59, 2005.

Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

Son N Tran and Artur S d'Avila Garcez. Deep logic networks: Inserting and extracting knowledge from deep belief networks. *IEEE transactions on neural networks and learning systems*, 29(2):246–258, 2016.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Alvaro Velasquez, Brett Bissey, Lior Barak, Andre Beckus, Ismail Alkhouri, Daniel Melcer, and George Atia. Dynamic automaton-guided reward shaping for monte carlo tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 12015–12023, 2021.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*, pp. 5045–5054. PMLR, 2018.

Cameron Voloshin, Abhinav Verma, and Yisong Yue. Eventual discounting temporal logic counterfactual experience replay. In *International Conference on Machine Learning*, pp. 35137–35150. PMLR, 2023.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016a.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016b.

Pierre Wolper, Moshe Y Vardi, and A Prasad Sistla. Reasoning about infinite computation paths. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pp. 185–194. IEEE, 1983.

Jinwei Xing, Takashi Nagata, Kexin Chen, Xinyun Zou, Emre Neftci, and Jeffrey L Krichmar. Domain adaptation in reinforcement learning via latent unified state representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10452–10459, 2021.

Tianpei Yang, Heng You, Jianye Hao, Yan Zheng, and Matthew E Taylor. A transfer approach using graph neural networks in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16352–16360, 2024.

Qi Yi, Rui Zhang, Shaohui Peng, Jiaming Guo, Yunkai Gao, Kaizhao Yuan, Ruizhi Chen, Siming Lan, Xing Hu, Zidong Du, et al. Online prototype alignment for few-shot policy transfer. In *International Conference on Machine Learning*, pp. 39968–39983. PMLR, 2023.

Shufang Zhu, Lucas M Tabajara, Jianwen Li, Geguang Pu, and Moshe Y Vardi. Symbolic ltlf synthesis. *arXiv preprint arXiv:1705.08426*, 2017.

Zhuangdi Zhu, Kaixiang Lin, Anil K Jain, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):13344–13362, 2023.

Appendix

A Implementation and Evaluation Details

A.1 Algorithm

In Algorithm 1, we provide a detailed algorithm for our proposed automaton Distillation method.

Algorithm 1: Automaton Distillation with DQN or TD3

Input: Teacher automaton Q-values Q_{teacher}, automaton state transition function, δ, labeling function L, number of training steps T, number of steps per target update, t_{update}, batch size M, actor update period d, learning rate α, annealing rate ρ, soft-update,τ, exploration constant ϵ, discount factor γ
Output: Student parameters θ
Initialize Networks:
if DQN **then**
  Critic: Q_θ; Target: $\theta_{target} \leftarrow \theta$
end
else TD3
  Critics : $Q_{\theta_1}, Q_{\theta_2}$; Actor: π_ϕ;
  Targets: $\theta_1^{target} \leftarrow \theta_1$, $\theta_2^{target} \leftarrow \theta_2$, $\phi' \leftarrow \phi$
end
Initialize replay buffer ER, automaton transition visit counts η
for $t \leftarrow 1$ **to** T **do**
  if DQN **then**
    Take ϵ-greedy action a;
  end
  else TD3
    Select action with exploration noise $a \leftarrow \pi_\phi(s) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma_{noise})$
  end
  Observe reward r and new state s';
  Compute new automaton state $\omega' = \delta(\omega, L(s'))$;
  Append augmented experience $((s, \omega), a, r, (s', \omega'))$ to the replay buffer ER with priority 1;
  Sample M transitions, $\{(s_i, \omega_i), a_i, r_i, (s_i', \omega_i')\}_{i=1}^{M}$, from replay with priority p_i;
  Compute annealing parameters for each transition $\beta_i \leftarrow \rho^{\eta(\omega_i, L(s_i'))}$;
  Update Q_{target} according to Eq. equation 4 or Eq. equation 5;
  Generate adjusted targets $Q_i' \leftarrow \beta_i Q_{\text{teacher}}(\omega_i, L(s_i')) + (1 - \beta_i) Q_{\text{target}}$;
  Update Q-networks $\theta \leftarrow \theta - \frac{\alpha}{M} \sum_i p_i \nabla_\theta (Q_i' - Q((s_i, \omega_i), a_i; \theta))^2$;
  if *TD3* and t *mod* $d = 0$ **then**
    $\phi \leftarrow \phi - \frac{\alpha}{M} \sum_i \nabla_\phi Q_1((s_i, \omega_i), \pi_\phi(s_i))$;
  end
  Update buffer priorities $p_i = (Q_i' - Q((s_i, \omega_i), a_i; \theta))^2$;
  for $i \leftarrow 1$ **to** M **do**
    Update visit count $\eta(\omega_i, L(s_i')) \leftarrow \eta(\omega_i, L(s_i')) + 1$;
  end
  if $t \mod t_{update} = 0$ **then**
    if DQN **then**
      $\theta_{target} \leftarrow \theta$;
    end
    else TD3
      $\theta_k^{target} \leftarrow \tau\theta_k + (1 - \tau)\theta_k^{target} \quad k = 1, 2$;
      $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$;
    end
  end
end