Joint Learning of Temporal Logic Constraints and Policy in RL with Human Feedbacks

Duo Xu, Faramarz Fekri
Department of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA

Abstract: Deep reinforcement learning (RL) has achieved significant success in artificial domains and in some real-world applications. However, substantial challenges remain such as learning efficiently under safety constraints. Satisfying safety constraints is a hard requirement in many high-impact application domains. At a suitable level of abstraction, these constraints have rich temporal and logical structures, and can be expressed using formal languages like temporal logics. However, in previous papers, these constraints are assumed to be known, which may not be true in many practical scenarios. In this paper, we study safe RL under *unknown* temporal logical constraint and propose a joint learning framework for safety constraint and policies with human feedbacks. The proposed framework interleaves between two loops of learning safety constraint and logically-constrained RL. Specifically, in the outer loop, a new algorithm based on temporal logic neural network (TLNN) is proposed to learn the automaton of constraint formula with traces labeled by human feedbacks. In order to satisfy the safety constraint zero-shot, in the inner loop, we propose to use a pre-trained generalizable shield and a logically combined Q function for action selection. We evaluate the proposed framework over various environments and provide in-depth empirical analysis on performances of both automaton learning and safety guarantee, empirically verifying the advantages of our methods over previous ones.

1 Introduction

Deep reinforcement learning (RL) has demonstrated excellent success in a variety of domains, including games [36, 50], robotic control [32], and recommendation systems [42], etc. In typical RL settings like these, the fundamental principle of RL - in which the agent aims to maximize long-term reward by trial and error - drives the agent to explore the entire state space and experiment with all possible actions in unknown environments. The application of RL in real-world domains, however, is hindered by challenges. Despite the significant efforts on making RL agents safe, one of the key challenges remains how to impose “safety constraints that should never [...] be violated” [16].

Safe RL is the process of learning optimal policies for reward maximization while ensuring a reasonable performance of safety at the same time [6]. In most of safe RL papers [15, 17, 11, 18], safety is understood as visiting ”good states” and avoiding ”bad states”. However, in many real-world applications, complex missions typically involve constraints on not only the current state but also the system’s trajectory. For example, suppose that a robot must first visit region A and then region B. Regions A and B may not be categorized as safe or unsafe, but not visiting A and B in the right order may imply a mission failure. Temporal logics (TL) [5] provide a powerful way of describing such complex spatial and temporal specifications. Some studies in the literature address RL problems under TL constraints. For example, Linear Temporal Logic (LTL) constraints are considered in a model-free learning framework and maximum possible LTL constraint satisfaction is achieved in [21]. A reactive system called shield is proposed in [2] where the chosen action is corrected if it causes the violation of an LTL specification.

However, in all of previous papers on safe RL with temporal logic constraints [2, 21, 1, 14, 28], these constraints are assumed to be known to the RL agent as specifications from human user, which may not be true in many real-world applications. Manually defining safety constraints for all possible scenarios is challenging, especially for robots operating in open-world environments [25]. Besides, implicit constraint is an important category of safety constraints [35], whose expression or definition is unknown to the agent due to complexity of system, such as network latency in data-center [12]. Although there are some previous papers studying unknown safety constraints in control theory [12, 52, 51], none of them considered temporal logic cases.

In this paper, we propose a novel framework for learning and satisfying temporal logic constraints, where the constraints are inferred from traces with human feedbacks. A human is introduced for labeling the safety of agent's trace in every episode. The proposed framework interleaves between two loops, i.e., learning the automaton representation of safety constraint (outer loop) and satisfying the learned constraint in RL (inner loop), as shown in Figure 1. Since the target of our work is to make the agent to follow any unseen constraint formula, it is necessary to use multi-task environments [3, 47].

In the outer loop, we propose a new approach for learning the deterministic finite automaton (DFA) of the temporal logic constraint ϕ from traces labeled by a human observer. Due to the randomness in agent's exploration, the collected traces of agent's behavior may contain a lot of redundant propositions which do not make any progress toward the satisfaction of ϕ. Hence, in previous methods [38, 54], the prefix-tree acceptor directly built from positive traces can have many redundant states to be merged, where the greedy selection of states may produce sub-optimal solutions. Therefore, we propose to directly learn the minimum tree acceptor (MTA) as the automaton representation of the target DFA. Since we assume the target DFA $\mathcal{A}_\phi$ is acyclic (ignoring self-loops) [27], the DFA $\mathcal{A}_\phi$ is equivalent as an MTA composed by the set of *skeleton paths* over $\mathcal{A}_\phi$ [13]. Hence, given positive and negative traces, we propose to build a model based on temporal logic neural network (TLNN) to directly learn the MTA of $\mathcal{A}_\phi$, which can avoid expensive computations (state-merging) in previous methods [38, 31, 9].

In the inner loop, we also propose a novel RL approach for following an unseen safety constraint ϕ in a zero-shot manner. The agent needs to not only learn the unknown constraint ϕ via human interactions for several episodes, but also train the policy (q-function) to satisfy ϕ without reducing accumulated rewards, during which the agent may violate safety constraints for many times. So, quick generalization to any unseen constraint formula is key to minimize the number of safety violations. Therefore, we propose to first pre-train a shield (Q function) in an offline environment which can predict the reachability of any temporal logic formulas, so that the agent can know how to follow an unseen constraint ϕ in a zero-shot manner. Then, in the online environment, inspired by [37, 44], we propose to use disconjunction operator to combine Q functions of accumulated rewards and constraint reachability together, and select actions over this combined Q function by ϵ-strategy, so that no further learning is needed for safety satisfaction even when the constraint formula ϕ changes. These two techniques can make the shield and agent to achieve zero-shot generalization to unseen constraint formulas.

In empirical experiments on image-based environments, we show that the proposed framework has many advantages over representative previous methods.

2 Related Work

Our work is strongly related with the body of work on temporal logic safety constraints in RL. In [2, 26] the shield is constructed by the automaton of the given and known safety constraint and removes unsafe actions for the agent in action selection during RL. [55] encode legality of actions explicitly into rules to avoid unsafe action selections. [46] specifically target learning normative behaviors in a particular normative framework, whereas we focus on high-level, intentional safety constraints. The setting of safe RL under constraints that may impact performance negatively was empirically identified in [29]. Recently, [1] investigates the probabilistic guarantee of satisfaction of temporal logic constraint in stochastic MDP, and [28] formulates the constrained POMDP and leverage off-the-shelf unconstrained POMDP solvers to satisfy the temporal logic constraints. However, in these works, the temporal logic safety constraints are assumed to be known to the agent, and none of them considered the efficient transferring from one constraint to another.

Another line of related works aims to utilize temporal logic instructions or symbolic knowledge to improve the learning of RL agent. In [20], the reward shaping based on STRIPS-based symbolic operators is used to improve the learning efficiency of RL agent. [7, 8] uses temporal logics to formulate the reward function, where the agent can learn to execute tasks with rich logic structure. [24] informs an RL agent with high-level symbolic instructions using the options framework in which low-level learned policies can be reused. The instructions are of a directive nature which is arguably less generic than the constraints used here. A recent work [22] propose to use an intrinsic reward to encourage exploration and task segmentation, whose definition is based on state propositions.

Recently [14] proposes to use reward shaping to encourage the agent to satisfy safety constraints without reducing accumulated environmental rewards. However, this work is based on the availability of the abstract or symbolic transition model, which may not be true in real-world applications. Even though an abstract transition model can be pre-trained in a simulated environment, in order to satisfy the safety constraints, symbolic planning is needed to compute the intrinsic reward used in the reward shaping, which needs extra computation and cannot achieve zero-shot transferring.

3 Preliminaries

3.1 Temporal Logic and Deterministic Finite Automaton

We define a sequence of elements from some alphabet as a word and a linear temporal property as a set of finite or infinite words over the alphabet $\Sigma : 2^{\mathcal{P}}$, where $\mathcal{P}$ is the set of propositional symbols. Safety constraint of a system can be expressed in a formal language that extends propositional logic with temporal operators, and linear temporal logic (LTL) is such a temporal logic language widely used in practice [5]. LTL formulas are evaluated over sequences of observations (i.e., truth assignments to the propositional symbols in $\mathcal{P}$).

An LTL specification ϕ can be converted into an automaton as its representation [48]. A deterministic finite automaton (DFA) $\mathcal{A}_\phi = \langle \mathcal{Q}, q_0, \Sigma, \delta, \mathcal{F} \rangle$ corresponding to ϕ consists of a set of states $\mathcal{Q}$, an initial state q_0, an alphabet Σ, a transition function $\delta : \mathcal{Q} \times \Sigma \rightarrow \mathcal{Q}$ and a set of accepting states $\mathcal{F}$. A run is a finite or infinite sequence of states $q = q_0, q_1, \ldots \in \mathcal{Q}^\infty$ induced by a trace $\sigma = \sigma_0, \sigma_1, \ldots \in \Sigma^\infty$, where $\forall i \in \mathcal{N}, q_{i+1} = \delta(q_i, \sigma_i)$. A trace of some system satisfies the automaton $\mathcal{A}_\phi$ iff the corresponding run q only visits the accepting states after some finite time step, i.e., $\exists t < \infty$, s.t., $\forall t' > t, q_{t'} \in \mathcal{F}$, which is termed as *accepting trace*.

3.2 Logic Neural Network

In this paper, we leverage the logic neural network [53, 39] to learn the automaton representation of safety specification. In logic neural networks, the conjunction and disjunction functions are defined as $F_c(x, \omega) := 1 - \omega(1 - x)$ and $F_d(x, \omega) := x \cdot \omega$, respectively. Specifically, given the vector of atom values $\boldsymbol{x}$, choosing the logic activation function as $\mathbb{P}(v) = \frac{-1}{-1+\log(v)}$ [53], the i-the conjunction and disjunction operation can be expressed as

$$\begin{aligned}
\mathrm{Conj}_i(\boldsymbol{x}, W_i) \quad &:= \quad \mathbb{P}(\prod_{j=1}^{n}(F_c(x_j, W_{i,j}) + \epsilon)), \\
\mathrm{Disj}_i(\boldsymbol{x}, W_i) \quad &:= \quad 1 - \mathbb{P}(\prod_{j=1}^{n}(1 - F_d(x_j, W_{i,j}) + \epsilon))
\end{aligned} \tag{1}$$

where W_i is the trainable weight vector for the i-th node in the logic layer, and ϵ is a small constant, e.g., 10^{-10}, for numerical stability.

3.3 Reinforcement Learning

RL provides a framework for learning to select actions in an environment in order to maximize the collected rewards over time [43]. RL deals with problems formalized as Markov decision problems (MDP). We here define a MDP as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma, S_0 \rangle$, where $\mathcal{S}$ is a finite set of environment states, $\mathcal{A}$ is a finite set of agent actions, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a probabilistic transition function, $R : \mathcal{S} \times \mathcal{A} \rightarrow [R_{\text{min}}, R_{\text{max}}]$ is a reward function with $R_{\text{min}}, R_{\text{max}} \in \mathbb{R}$, $\gamma \in [0, 1)$ is a discount factor, $S_0 : s_0 \sim S_0$ is a distribution of initial states. In each time step t, the agent observes the environment state s_t and selects some action a_t to apply, according to some policy function $\pi \in \Pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, and then collects reward $r_t = R(s_t, a_t)$.

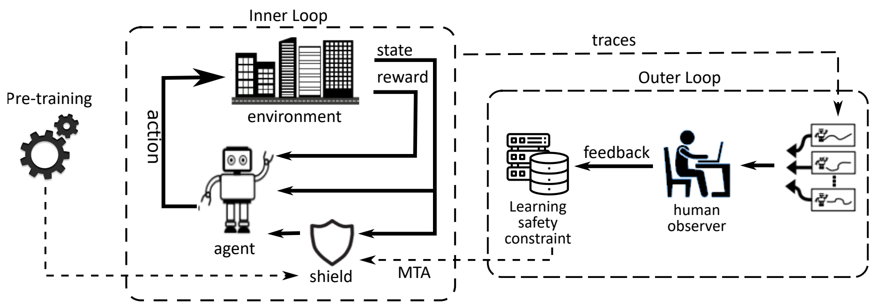


Figure 1: The diagram of the proposed framework. The inner loop is the safe RL with a shield for satisfying the safety constraint. The outer loop is for collecting human feedbacks and learning automaton representation of safety automaton.

For some policy π, the values V and Q for any state s and state-action pair (s, a) at time t can be defined as below,

$$V_\pi(s) = \mathbb{E}_\pi\left[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau | s_t = s\right], \quad Q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau | s_t = s, a_t = a\right] \tag{2}$$

where $\mathbb{E}_\pi$ is the expectation of accumulated rewards following some policy π. A policy is the optimal policy π^* if it produces the highest accumulated rewards: $\forall s \in \mathcal{S}, \forall \pi \in \Pi, \forall a \in \mathcal{A} : Q_{\pi^*}(s,a) > Q_\pi(s,a)$. Searching π^* can be addressed by parameterizing the policy and finding optimal parameters θ^* that maximize the accumulated rewards by a learning algorithm. Specifically, parameters θ can be weights of neural networks optimized by gradient descent.

A widely-used parameterized approach of searching π^* in the space of neural networks is known as deep Q-Networks (DQN) [36]. DQN uses deep neural networks with weights θ to approximate $Q_\pi(s, a|\theta)$, and at each step t, the agent selects actions uniform randomly with some probability $\epsilon \in [0, 1)$ or greedily over $Q_\pi(s, a)$ with probability $1-\epsilon$. The generated experience tuple (s_t, a_t, r_t, s_{t+1}) is stored to a buffer $\mathcal{B}$. The weights θ are updated by using advanced optimizer, such as Adam, iteratively. At each iteration, we update the weights θ of neural networks by minimizing the loss function as below

$$\mathcal{L}(\theta; \theta^-) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}\left(r + \gamma \max_{a'} Q(s', a'|\theta^-) - Q(s, a|\theta)\right)^2 \tag{3}$$

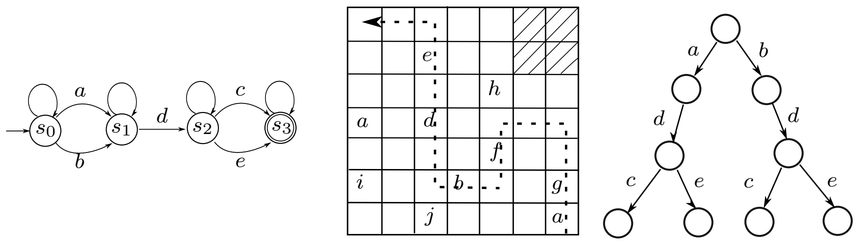
where θ^- are target weights of neural networks which are updated periodically for improving numerical stability of the learning process.

4 Methodology

In this section, we introduce the proposed framework for learning and satisfying temporal logic (symbolic) constraints in RL with human feedbacks. As far as we know, we are the first to tackle safe RL with *unknown* temporal logic constraints. Our proposed framework jointly learns the safety constraint and the policy (q-function) of RL agent which follows the safety constraint. It consists of outer and inner loops, as shown in Figure 1. In the outer loop, a human is introduced to give labels, positive (safety satisfied) or negative, over the traces agent's behavior, and the safety constraint is learned from human feedbacks. The inner loop of our framework contains the RL setup and considers a shield encoding the learned constraint, which is to make the agent safe while keeping accumulated rewards maximized.

We have innovations in both outer and inner loops. That is because the straightforward combination of automaton learning algorithm [38] and safe RL with symbolic shield [2, 21, 1] does not work well, which have the following problems to be solved:

- Due to the random exploration of RL agent, many visited propositions in traces are redundant and do not make any progress toward constraint satisfaction. The popular previous methods [38, 10] directly build prefix tree from positive traces and generate a lot of redundant states to be merged, where the greedy state selection can degrade the learning performance;



(a) Target DFA (b) Trace (c) MTA

Figure 2: Example of learning safety constraint ϕ in Mine-Craft game. (a): Target DFA is the real automaton of safety constraint (propositions on self-loops are omitted) $\mathcal{A}_\phi$. (b): the dashed line shows a trajectory of agent in Letter Game, along which the sequence of letters is a trace. (c): MTA of target automaton. $\mathcal{A}_\phi$.

- Since the agent does not know the safety constraint a priori, the number of safety violations may be large during the RL process. It is important for the agent to quickly generalize to the learned safety constraint in a zero-shot manner;
- The objective of satisfying safety constraint may not be associated with that of maximizing the environmental reward. How to achieve both objectives in a zero-shot manner remains unsolved.

In this work, we propose specific techniques to tackle problems above, which will be introduced one by one in the following sections.

4.1 Outer Loop: Learning Safety Automaton with Human Feedbacks

The outer loop is a feedback loop, where a human observer gives feedbacks on the safety of agent's traces, and the automaton representation of safety constraint ϕ is learned from labeled traces. In the following paragraphs, we define a trace as the sequence of propositions visited by the agent in one episode, as an example shown in Figure 2(b).

Learning deterministic finite automaton (DFA) from labeled traces is a classical problem in formal method. Many previous papers first learn LTL formula from traces [49, 34, 33] and convert it into DFA, which could lead to state-explosion problem [5]. Another line of works uses SAT-based methods to learn the DFA [9, 54], which do not have polynomial time complexity and cannot be deployed into RL applications. Here we focus on polynomial-time algorithms for learning the DFA $\mathcal{A}_\phi$ of the constraint formula ϕ. Popular polynomial-time algorithms for learning automaton, such as Regular Positive Negative Inference (RPNI) algorithm [38], always have two phases: first construct a prefix tree acceptor (PTA) from positive traces, and second try to merge states of PTA by comparing positive and negative traces until no more states can be merged. The target is to learn a minimum DFA which can perfectly classify positive and negative traces.

However, since the exploration is random, in observed traces, a lot of propositions are redundant and only cycle in self-loops at the target DFA which do not make any progress toward satisfaction of ϕ. For example, in Mine-Craft environment, assume that the target DFA of safety constraint ϕ is shown in Figure 2(a). And the trajectory of agent as shown Figure 2(b) contains a trace $agfbde$. We can see that in this trace, only propositions a, d, e trigger transitions to different automaton states, whereas other propositions only cycle in self-loops, which are redundant and do not move toward the satisfaction of ϕ. However, previous automaton learning methods always first build a PTA directly from positive traces [13, 38]. Hence, there will be a large number of redundant states in the PTA to be merged, which has high time complexity and storage overhead, especially when the traces are long.

Therefore, we propose to learn *minimal tree acceptor* (MTA) of the target DFA, which is defined as the minimal tree-like automaton which can accurately classify positive and negative traces according to the target DFA. First, we define the *skeleton path* as an accepting trace on the target DFA which does not go through any self-loops. For example, there are four skeleton paths on $\mathcal{A}_\phi$ in Figure 2(a), i.e., $\{adc, ade, bdc, bde\}$. We assume that the target FDA is acyclic (ignoring self-loops), which is a natural assumption for RL and robotics since only temporal logic formulas with finite satisfaction are considered here [27]. Then, we can have the MTA by aggregating all of skeleton paths of target

DFA together as a tree [13]. E.g., the MTA of $\mathcal{A}_\phi$ is shown in Figure 2(c). We can see that the MTA is concise and keeps the same temporal logic relationships as the target DFA. In the following, we propose temporal logic neural network (TLNN) to learn MTA, extending logic neural network introduced in Section 3.2 to the temporal domain.

4.1.1 Temporal Logic Neural Network

TLNN $\mathcal{T}$ is essentially a tree-like logic network, where the logic operations are realized by differentiable operators in (1). This $\mathcal{T}$ has L layers equal to the depth of target DFA. Every node in $\mathcal{T}$ has $|\Sigma|$ children each of which corresponds to one proposition in Σ, so that the l-th layer of $\mathcal{T}$ has $|\Sigma|^l$ nodes, $l = 0, 1, \dots, L-1$, and every node is uniquely indexed by its prefix s which concatenates propositions running from the root to this node, so that $|s| = l$. Hence, TLNN $\mathcal{T}$ contains every possible skeleton path of target DFA, and $\mathcal{T}$ contains MTA as its sub-tree. We train this $\mathcal{T}$ as a classifier of input positive and negative traces and then extract the MTA.

Denote the max length of input trace as T. Inspired by [9], we define the value function of every node of $\mathcal{T}$ as $V(e, t, s) \in [0, 1]$, where e is the input trace, $0 \leq t < T$ is the time index, and s is the node prefix. We first represent the input trace by its binary encoding, i.e., $e[\sigma, t] = \{0, 1\}$ denotes the existence of σ at time step t in e. In the first ($l = 1$) layer, where the prefix s has only one proposition, for $\forall \sigma \in \Sigma$, we have $V(e, 0, \sigma) = e[\sigma, 0]$, and $V(e, t, \sigma) = V(e, t-1, \sigma) \vee e[\sigma, t]$ for $1 \leq t < T$. In the l-th ($l > 1$) layer, for every prefix s, we have

$$V(e,t,s) = \begin{cases} e[s[-1], 0], & \text{if } t = 0 \\ V(e, t-1, s) \vee (V(e, t-1, s[:-1]) \wedge e[s[-1], t]), & \text{otherwise} \end{cases} \tag{4}$$

where $s[-1]$ denotes the tail proposition of s and $s[:-1]$ is the prefix with its tail removed. After computing as (4) from the first to bottom ($l = L-1$) layer, $\forall s$, the value $V(e, T-1, s)$ in every layer denotes the likelihood of the input trace e containing prefix s as its sub-sequence. Then, the predicted label of e is defined as the output of a fully-connected layer which integrates $V(e, T-1, s)$ of all the prefix s at $l > 1$ layers. With human labels as ground-truth, the loss function to train $\mathcal{T}$ can be formulated as the cross-entropy between predicted labels and ground-truth. If we realize the conjunction and disjunction operations (4) in every layer by differentiable operators in (1), $\mathcal{T}$ becomes differentiable and can be trained by gradient-descent based optimizers, such as ADAM. After training for sufficient iterations until the prediction accuracy becomes satisfied, we can extract the learned MTA by thresholding the weights in $\mathcal{T}$, i.e., $\omega = \mathbf{1}(\omega > 0.5)$.

4.2 Inner Loop: Satisfying Safety Constraint with a Pre-trained Shield

In the inner loop, given the learned MTA of safety constraint ϕ, the RL agent trains its policy to maximize the accumulated environmental reward while satisfying safety constraint at the same time. Here a shield is built to encourage the agent to follow ϕ. Since ϕ is unknown to the agent a priori, the agent may violate ϕ for many times when learning to satisfy the constraint ϕ. So, the key of reducing violations is to quickly adapt and transfer to unseen constraint ϕ in a *zero-shot* manner. In order to achieve this goal, we propose two techniques: 1) we first pre-train a shield in an *offline* environment, which can achieve zero-shot generalization to any unseen temporal logic formulas; 2) with this pre-trained shield as a Q function of reachability of constraint satisfaction, in the *online* environment, we propose to use disjunction operation to combine Q functions of accumulated reward and the shield, where the action is selected by this combined Q function.

4.2.1 Pre-training the Shield in Offline Environment

Inspired by the domain randomization technique which can significantly improve the generalization of RL agent [45, 47], we pre-train the shield in an offline environment where both the layout and task are randomly generated upon reset. In this environment, the task is to satisfy a given temporal logic formula φ, and the reward is the satisfaction of φ, where only formula whose corresponding DFA is acyclic (ignoring self-loops) is considered. Specifically, upon reset, the positions of objects and φ are randomly generated. The shield we pre-train is essentially a Q function predicting the reachability of the satisfaction of φ from the input state and action pair. The state input to the shield is the concatenation of representations of environment observation and MTA of the task φ. The closer the current state is to the satisfaction of φ, the closer the predicted Q value is to 1.

Since the strong expressivity of GNN can make the agent leverage the compositional syntax and semantics of the temporal logic tasks [47, 30], we use graphical neural network (GNN) [19, 40, 41] to learn the task representation in this shield. Specifically, given input automaton, we first decompose the MTA of φ into a finite set of skeleton paths, i.e., $\mathcal{K} = \{\tau_a^0, \tau_a^1, \ldots\}$, and represent every τ_a^i by GNN. Obviously, the reachability of φ is equal to the maximum of the reachability of every τ_a^i in $\mathcal{K}$, since if any τ_a^i in $\mathcal{K}$ is satisfied, the whole task φ becomes satisfied. Therefore, the Q network of the shield is essentially trained to predict the reachability of any sequence of propositions represented by a GNN.

Different from previous works [47, 30] which use on-policy RL algorithms, in order to improve the learning efficiency, we use off-policy deep Q-learning algorithm [36] to pre-train the generalizable shield. In addition, we also incorporate hindsight experience replay (HER) [4] to accelerate this training process. Specifically, HER is extended to temporal logic domain, modifying any unsuccessful trajectory whose associated temporal logic task was not finished. Since a non-zero reward is only given at the end of the trajectory, unsuccessful trajectory does not have any reward information and is not useful for training. Hence, in any unsuccessful trajectory with task φ, the sequence of propositions visited by the agent is used to modify the original task φ, replacing a randomly selected skeleton path τ_a^i of φ, so that this modified task φ' becomes finished and the trajectory associated with φ' becomes successful. This GNN-based shield can be trained to generalize to a large number of temporal logic tasks. More training and architecture details are in Appendix.

4.2.2 Following the Constraint in Online Environment

We apply the pre-trained shield to make the agent satisfy the safety constraint ϕ in online environment. In online environment, the reward is another function which may not be aligned with the satisfaction of ϕ. As long as the outer loop updates the formula of ϕ, the MTA of ϕ can be obtained and encoded into the shield via GNN. The pre-trained shield is essentially a Q function which can predict the reachability of satisfiability of formula ϕ. Therefore, according to the discussion in [37], we can use the disconjunction of Q functions of accumulated reward and satisfaction of ϕ to select actions, so that the objectives of maximizing the reward and satisfying ϕ can be both achieved zero-shot without further learning. Specifically, in state s_t, we can select action as

$$a_t^* = \arg\max_{a \in \mathcal{A}} \max\{Q_R(s_t, a), Q_s(s_t, a; \phi)\} \tag{5}$$

where $Q_R(s, a)$ and $Q_s(s, a; \phi)$ are the Q functions of reward and shield with formula ϕ encoded, respectively.

5 Experiment

In this section, we will provide comprehensive empirical evaluation of the proposed framework in environments with unknown safety constraint. Specifically, we evaluate the performance of the proposed framework on three aspects, including automaton learning, generalization of pre-trained shield, and overall performance of reward maximization and safety satisfaction, which are designed to answer the following questions:

- Does the proposed automaton learning method outperform previous methods?
- Can the pre-trained shield be generalized to other temporal logic constraints in a zero-shot manner?
- Can the agent in the proposed framework quickly learn to satisfy the safety constraint while maximizing accumulated rewards at the same time?

which correspond to three problems described in the beginning of Section 4. We use two environments in this work:

Letter Game: The first environment is similar as Mine-craft in [3, 47]. Out of the 49 squares, 16 are associated with 8 unique propositions/letters, and some letter may appear twice in the grid, so that there are multiple ways to reach some letter. Besides letters, the shadowed area is the reward region, where the agent can obtain a positive reward when reaching it first time. An example layout is shown in Figure 2(b).

Maze Game: This environment is a grid-world game whose observation is divided into four rooms by walls. There are 6 objects,denoted with different letters, randomly located in these rooms. An example of layout is shown in Figure 3. The agent is randomly placed in one of these rooms. Each room is connected with every neighboring room by a corridor, and two corridors of four rooms are blocked by locked doors. The agent can open locked doors by acquiring a key to that particular door and using it on the lock. These keys are placed in a position that is reachable for the agent. The reward here is to acquire some letter which is unknown to the agent a priori and not associated with the safety constraint.

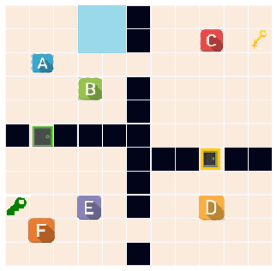


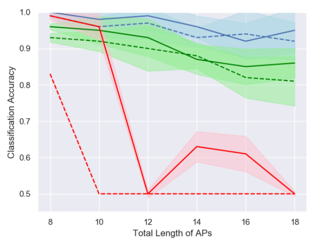
Figure 3: Maze Game

In these environments, at each step the agent can move along the cardinal directions (up, down, left and right). The agent observes the full grid (and letters) from an egocentric point of view together with the MTA of the temporal logic task/constraint. The observation of the grid is processed as an image where the positions of objects (letters, keys and doors) are not revealed to the agent. In the following sections, we will first present the performance of proposed automaton learning method. Then the pre-trained shield is empirically evaluated on two grid-world games. Finally, the overall learning performance of the proposed framework is presented, combining inner and outer loops together.

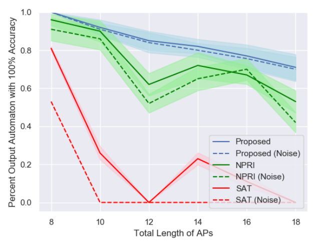
5.1 Automaton Learning

We first evaluate the proposed automaton learning method in a standalone mode, independent of other components in the proposed framework. The positive and negative traces of propositions are randomly generated. With the same data, in order to further test its ability to handle noise, 2% of labels are swapped. We use SAT-based method [9] and NPRI [38] as baselines with implementations at [1]. The SAT-based method iteratively increases the maximum allowable formula size and reruns the SAT solver until a formula is found. In the proposed method, we also gradually increase the maximum depth of MTA until a satisfying acceptor is found. This process guarantees the output formula is optimally compact.

Data Generation Assume that the proposition set $\mathcal{P} = \{a, b, c, d, e, f\}$. First, we generate a finite set $\mathcal{K}$ of skeleton paths τ_a and use them to build the MTA as the representation of target DFA. Assume $2 \le |\mathcal{K}| \le 16$. Every element of τ_a is uniformly selected from $\mathcal{P}$. The length of τ_a is randomly selected between 2 and 5. Second, when the MTA is constructed, we generate 1024 training traces, where elements in every trace are uniformly sampled from $\mathcal{P}$ and each trace is labeled according to the satisfaction of the MTA. The max length of traces is 10.



(a) Accuracy



(b) Recovery

Figure 4: Comparison of the proposed method, NPRI and SAT-based methods on testing traces with and without noise. 0.5 accuracy denotes the time-out of SAT solver.

Results The first performance metric for comparison is the accuracy, defined as the percentage of correctly classified traces. Here both SAT and NPRI methods are set to learn automaton with finite satisfaction over which we can easily obtain the MTA by extracting skeleton paths, since we only consider automaton having finite accepting traces. The second performance metric is recovery, referring to whether the MTA of target DFA is perfectly learned. Figure 4 shows the performance of

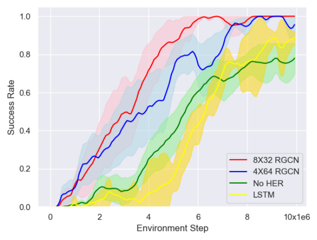
[1]https://github.com/adiojha629/JIRP_LRM

our method and baselines on the test traces after training on the original and noisy datasets, where the x-axis is the total length of skeleton paths in target DFA, reflecting the complexity of target DFA.

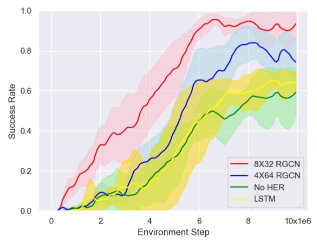
In both original and noisy settings, our method consistently performs better than baselines over all evaluations on both accuracy and recovery. Since state selection of state-merging in NPRI is greedy, it has poor performance in some cases with complex target DFA. We observe that the standard SAT method [9] goes time out in many cases with large target DFA. Additionally, the SAT method performs badly in the noisy setting, but our method is robust to noisy labels due to the large number of trainable weights.

5.2 Pre-training of the Shield

In the proposed framework, we first pre-train a shield as a DQN agent in an offline environment. By leveraging the power of GNN, it is pre-trained to be generalized to any unseen temporal logic formula, so that it can be directly used in online environment without any further learning (zero-shot). In offline environment, the reward is to satisfy the task described by a temporal logic formula, and the shield is trained by off-policy DQN algorithm [36] augmented by the HER technique [4] extended to logic formula.



(a) Letter



(b) Maze

Figure 5: Performance of pre-training shield, compared with baselines for ablation study.

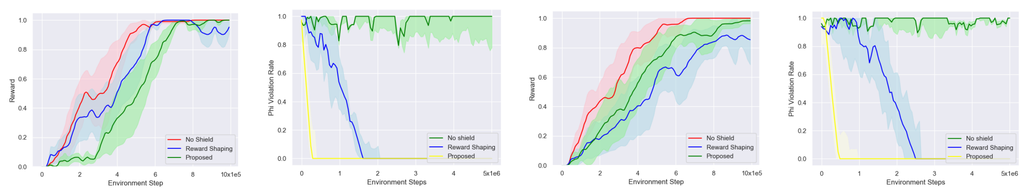
Setup Curriculum learning is used to pre-train the shield. The shield is first trained to satisfy task formulas with single proposition, which is the level 1. If the average testing accuracy is above 0.8, the training proceeds to the next level, increasing the task formula by one proposition. The highest level is 5, which is the maximum depth of the target DFA considered in this paper. We adopt three baselines here, which is actually ablation study to the proposed model of the shield. The first baseline uses a different structure of GNN, the second one does not use HER, and the third one simply uses an LSTM [23] to process the input formula without using GNN.

Results The experiment results are shown in Figure 5. We can observe that with the proposed training method and model, the shield can achieve 100% success rate of task solving in letter game, and can go above 90% success rate in maze game. Due to environment complexity, the convergence rate in maze game is significantly slower than that in letter game. Based on Baseline-2, we can see that the HER technique can significantly accelerate the training of shield in both letter and maze games. The learning performance of Baseline-3 shows that GNN is a better choice to learn the representation of task formula than LSTM.

5.3 Performance of the Proposed Framework

In the online environment, with the shield pre-trained offline, we evaluate the proposed framework, integrating inner and outer loops as a whole as shown in Figure 1. The human feedback is binary and does not have any other information about state or reward in the target DFA, which is different from previous work on inferring task automaton [54].

Baseline Baseline-1 is the reward shaping method [14]. In this baseline, the environment reward is augmented by a positive reward ρ based on the progression toward satisfaction of the safety constraint formula. Since we only focus on model-free RL in this work, the abstract transition model is not available and the product symbolic model [2, 14] cannot be built here. So, the auxiliary reward ρ is computed as the number of steps to any accepting state only in target DFA $\mathcal{A}_\phi$. The agent model of



(a) Reward in Letter (b) Safety in Letter (c) Reward in Maze (d) Safety in Maze

Figure 6: Performance of the proposed full framework, compared with baselines.

this baseline does not have GNN part to process constraint formulas ϕ, so it cannot be generalized to other unseen formulas. Baseline-2 is the naive shielding, meaning that the shield simply filter out actions from which the reachability of ϕ (Q value) is below a threshold (0.1), which is similar as the original shield in previous papers [2, 1].

Result In the online environment, we assume that there is a real constraint formula ϕ fixed in human mind. The learning performance in both letter and maze games is shown in Figure 6. For reward learning, since the agent is constrained to follow ϕ, the performances of the proposed method and Baseline-1 are worse than Baseline-2 which has less constraint on the agent's behavior. For safety guarantee, the proposed framework performs best. Due to reward shaping, the agent in Baseline-1 needs many iterations to learn and incorporate the added rewards into its Q function, so that it has more safety violations, compared with our method with zero-shot safety satisfaction. Even though Baseline-2 also uses same pre-trained shield as the proposed method, the naive shielding there does not work, since its interference on agent's action is too weak to follow the constraint formula ϕ. Its comparison with our method verifies that the disjunction operator (5) is a simple and effective method on constraint satisfaction. In previous paper [2, 1], the environment is small and does not have domain randomization, so that the naive shielding is enough for safety guarantee.

References

[1] Derya Aksaray, Yasin Yazıcıoğlu, and Ahmet Semi Asarkaya. Probabilistically guaranteed satisfaction of temporal logic constraints during reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6531–6537. IEEE, 2021.

[2] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[3] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pages 166–175. PMLR, 2017.

[4] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[5] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[6] Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5, 2021.

[7] Alberto Camacho, Oscar Chen, Scott Sanner, and Sheila A McIlraith. Non-markovian rewards expressed in ltl: guiding search via reward shaping. In *Tenth Annual Symposium on Combinatorial Search*, 2017.

[8] 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*, 2019.

[9] Alberto Camacho and Sheila A McIlraith. Learning interpretable models expressed in linear temporal logic. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 621–630, 2019.

[10] Rafael C Carrasco and Jose Oncina. Learning stochastic regular grammars by means of a state merging method. In *International Colloquium on Grammatical Inference*, pages 139–152. Springer, 1994.

[11] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

[12] Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

[13] Colin De la Higuera. *Grammatical inference: learning automata and grammars*. Cambridge University Press, 2010.

[14] Floris den Hengst, Vincent François-Lavet, Mark Hoogendoorn, and Frank van Harmelen. Planning for potential: efficient safe reinforcement learning. *Machine Learning*, pages 1–20, 2022.

[15] Kurt Driessens and Sašo Džeroski. Integrating guidance into relational reinforcement learning. *Machine Learning*, 57(3):271–304, 2004.

[16] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

[17] Yonathan Efroni, Shie Mannor, and Matteo Pirotta. Exploration-exploitation in constrained mdps. *arXiv preprint arXiv:2003.02189*, 2020.

[18] Jaime F Fisac, Anayo K Akametalu, Melanie N Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2018.

[19] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks*, volume 2, pages 729–734, 2005.

[20] Marek Grzes and Daniel Kudenko. Plan-based reward shaping for reinforcement learning. In *2008 4th International IEEE Conference Intelligent Systems*, volume 2, pages 10–22. IEEE, 2008.

[21] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Cautious reinforcement learning with logical constraints. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 483–491, 2020.

[22] 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, pages 7647–7656, 2021.

[23] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[24] León Illanes, Xi Yan, Rodrigo Toro Icarte, and Sheila A McIlraith. Symbolic plans as high-level instructions for reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 540–550, 2020.

[25] Jeevana Priya Inala, Yecheng Jason Ma, Osbert Bastani, Xin Zhang, and Armando Solar-Lezama. Safe human-interactive control via shielding. *arXiv preprint arXiv:2110.05440*, 2021.

[26] Nils Jansen, Bettina Könighofer, Sebastian Junges, Alex Serban, and Roderick Bloem. Safe reinforcement learning using probabilistic shields. In *31st International Conference on Concurrency Theory (CONCUR 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[27] Kishor Jothimurugan, Rajeev Alur, and Osbert Bastani. A composable specification language for reinforcement learning tasks. *Advances in Neural Information Processing Systems*, 32, 2019.

[28] Krishna C Kalagarla, Dhruva Kartik, Dongming Shen, Rahul Jain, Ashutosh Nayyar, and Pierluigi Nuzzo. Optimal control of partially observable markov decision processes with finite linear temporal logic constraints. *arXiv preprint arXiv:2203.09038*, 2022.

[29] Bettina Könighofer, Florian Lorber, Nils Jansen, and Roderick Bloem. Shield synthesis for reinforcement learning. In *International Symposium on Leveraging Applications of Formal Methods*, pages 290–306. Springer, 2020.

[30] Yen-Ling Kuo, Boris Katz, and Andrei Barbu. Encoding formulas as deep networks: Reinforcement learning for zero-shot execution of ltl formulas. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5604–5610. IEEE, 2020.

[31] Bernard Lambeau, Christophe Damas, and Pierre Dupont. State-merging dfa induction algorithms with mandatory merge constraints. In *International Colloquium on Grammatical Inference*, pages 139–153. Springer, 2008.

[32] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[33] Xiao Li, Zachary Serlin, Guang Yang, and Calin Belta. A formal methods approach to interpretable reinforcement learning for robotic planning. *Science Robotics*, 4(37):eaay6276, 2019.

[34] Xiao Li, Cristian-Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3834–3839. IEEE, 2017.

[35] Yongshuai Liu, Avishai Halev, and Xin Liu. Policy learning with constraints in model-free reinforcement learning: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 2021.

[36] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[37] Geraud Nangue Tasse, Steven James, and Benjamin Rosman. A boolean task algebra for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:9497–9507, 2020.

[38] José Oncina and Pedro Garcia. Inferring regular languages in polynomial updated time. In *Pattern recognition and image analysis: selected papers from the IVth Spanish Symposium*, pages 49–61. World Scientific, 1992.

[39] Ali Payani and Faramarz Fekri. Learning algorithms via neural logic networks. *arXiv preprint arXiv:1904.01554*, 2019.

[40] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[41] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[42] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. An mdp-based recommender system. *Journal of Machine Learning Research*, 6(9), 2005.

[43] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[44] Geraud Nangue Tasse, Steven James, and Benjamin Rosman. Generalisation in lifelong reinforcement learning through logical composition. In *Deep RL Workshop NeurIPS 2021*, 2021.

[45] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[46] Stevan Tomic, Federico Pecora, and Alessandro Saffiotti. Learning normative behaviors through abstraction. In *24th European Conference on Artificial Intelligence (ECAI 2020), Santiago de Compostela, Spain, August 29-September 8, 2020*, volume 325, pages 1547–1554. IOS Press, 2020.

[47] 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.

[48] Moshe Y Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for concurrency*, pages 238–266. Springer, 1996.

[49] Marcell Vazquez-Chanlatte, Susmit Jha, Ashish Tiwari, Mark K Ho, and Sanjit Seshia. Learning task specifications from demonstrations. *Advances in neural information processing systems*, 31, 2018.

[50] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[51] Akifumi Wachi and Yanan Sui. Safe reinforcement learning in constrained markov decision processes. In *International Conference on Machine Learning*, pages 9797–9806. PMLR, 2020.

[52] Akifumi Wachi, Yanan Sui, Yisong Yue, and Masahiro Ono. Safe exploration and optimization of constrained mdps using gaussian processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[53] Zhuo Wang, Wei Zhang, Ning Liu, and Jianyong Wang. Scalable rule-based representation learning for interpretable classification. *Advances in Neural Information Processing Systems*, 34, 2021.

[54] Zhe Xu, Ivan Gavran, Yousef Ahmad, Rupak Majumdar, Daniel Neider, Ufuk Topcu, and Bo Wu. Joint inference of reward machines and policies for reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 590–598, 2020.

[55] Haodi Zhang, Zihang Gao, Yi Zhou, Hao Zhang, Kaishun Wu, and Fangzhen Lin. Faster and safer training by embedding high-level knowledge into deep reinforcement learning. *arXiv preprint arXiv:1910.09986*, 2019.