Active Grammatical Inference for Non-Markovian Planning

Noah Topper[1], **George Atia**[1,2], **Ashutosh Trivedi**[3], **Alvaro Velasquez**[4]

[1]Department of Computer Science, University of Central Florida
[2]Department of Electrical and Computer Engineering, University of Central Florida
[3]Department of Computer Science, University of Colorado Boulder
[4]Information Directorate, Air Force Research Laboratory
noah.topper@knights.ucf.edu, george.atia@ucf.edu, ashutosh.trivedi@colorado.edu, alvaro.velasquez.1@us.af.mil

Abstract

Planning in finite stochastic environments is canonically posed as a Markov decision process where the transition and reward structures are explicitly known. Reinforcement learning (RL) lifts the explicitness assumption by working with sampling models instead. Further, with the advent of reward machines, we can relax the Markovian assumption on the reward. Angluin's active grammatical inference algorithm L^* has found novel application in explicating reward machines for non-Markovian RL. We propose maintaining the assumption of explicit transition dynamics, but with an implicit non-Markovian reward signal, which must be inferred from experiments. We call this setting non-Markovian planning, as opposed to non-Markovian RL. The proposed approach leverages L^* to explicate an automaton structure for the underlying planning objective. We exploit the environment model to learn an automaton faster and integrate it with value iteration to accelerate the planning. We compare against recent non-Markovian RL solutions which leverage grammatical inference, and establish complexity results that illustrate the difference in runtime between grammatical inference in planning and RL settings.

Introduction

While there has been tremendous success in the RL and planning communities in recent years, much of it has been underpinned by strict Markovian assumptions. This means the current state of an agent must be sufficient to determine the dynamics and optimal behavior going forward. This assumption holds true for the impressive AlphaGo (Silver et al. 2016), AlphaGo Zero (Silver et al. 2017), and AlphaZero (Silver et al. 2018) planning agents, but it is often the case that the underlying goal of an agent is non-Markovian. In such settings, it is standard to assume a characterization of the underlying objective using temporal logic or an equivalent automaton representation thereof. Given this, conventional planning solutions such as value iteration can be adopted to solve the problem (Thiébaux et al. 2006).

An interesting problem arises when we assume that the foregoing logical or automaton characterization is not given, but must be learned via experimentation. In recent years, this has been explored in the context of RL, where access to the model is not available and the agent must learn through black-box exploration of the environment. In this setting, quintessential algorithms from the field of grammatical inference have been applied with some success (Gaon and Brafman 2020; Xu et al. 2020, 2021). However, non-Markovian planning with known transition dynamics remains largely unexplored (with the exception of the pre-print in (Rens and Raskin 2020)). In this paper, we provide an algorithm that integrates active grammatical inference in the form of the L^* algorithm (Angluin 1987) to learn a deterministic finite automaton (DFA) representation of the goal in the underlying decision process. This DFA is then used to compute an augmented decision process where conventional value iteration can be adopted. As a second contribution, we rigorously prove that the fundamental problem of answering membership queries for grammatical inference can be computed efficiently in the planning context, but is **NP-complete** in popular RL regimes where the computed policy is Markovian and deterministic.

Related Work

Non-Markovian planning has been studied extensively (Thiébaux et al. 2006) (Brafman and De Giacomo 2019) (Bacchus, Boutilier, and Grove 1997) (Thiébaux, Kabanza, and Slaney 2002) (Gretton 2006) (Velasquez et al. 2021). However, it is typically assumed that the goal is given in some automaton representation. We are concerned with non-Markovian planning when this goal is not known, but its automaton representation must be learned through interactions with the given environment model. Once learned, it can be leveraged using existing techniques.

Reward machines were introduced in (Icarte et al. 2018) as a finite-state machine representation of a non-Markovian reward signal. Since then, the RL community has seen the application of grammatical inference techniques to represent such underlying objectives. Grammatical inference is a branch of machine learning typically concerned with learning an automaton representation of some grammar. Active inference allows for the system under learning to be queried, whereas its passive counterpart leverages an existing static repository of samples. The work in (Xu et al. 2020) uses passive grammatical inference by storing traces of behavior that arise in the standard Q-learning methodology. This repository can then be used to synthesize a reward machine using techniques like satisfiability solving. A similar approach is

explored in deep RL by leveraging perceptual abstractions over the visually complex state space in order to derive simple atomic propositions over which the learned automaton is defined (Hasanbeig et al. 2021). In (Xu et al. 2021), active grammatical inference is used in the form of L^* to learn an automaton by proposing an additional policy to answer membership queries (a simpler version of this approach can be seen in (Gaon and Brafman 2020)). These queries answer whether a given trace is in the language of the automaton to be learned and are integral to the functioning of L^*.

It is worth noting that some classes of infinite languages can also be captured by finite automata. This is the case with the omega-regular languages and their representation as deterministic rabin automata. In this context, non-Markovian planning using value iteration has been explored by shaping the reward signal of the decision process so that the agent is incentivized to reach the accepting components of the augmented decision process (Hasanbeig, Abate, and Kroening 2018).

To the best of our knowledge, the only other work on learning automata for non-Markovian planning is the preprint in (Rens and Raskin 2020), where membership queries for L^* are answered in the planning context. Our paper differs in that we answer both membership and equivalence queries by leveraging the known model, establish complexity results, and compare against competing approaches.

Preliminaries

We assume a non-Markovian reward decision process (NMRDP) as the model of the agent-environment dynamics, and our learning algorithm has full access to this model aside from the reward function R.

Definition 1 (Non-Markovian Reward Decision Process)**.** *An NMRDP is a tuple $M = (X, x_0, A, P, \mathsf{AP}, L, R)$, where X is a finite set of states, $x_0 \in X$ is a distinguished initial state, A is a finite set of actions, $P : X \times A \times X \to [0, 1]$ is a probabilistic transition function, AP is a finite set of atomic propositions, $L : X \to 2^{\mathsf{AP}} \cup \{\varepsilon\}$ is a labeling function, and $R : X^* \to \{0, 1\}$ is a non-Markovian reward function.*

The labeling function $L : X \to 2^{\mathsf{AP}} \cup \{\varepsilon\}$ maps states of the NMRDP to the alphabet $\Sigma = 2^{\mathsf{AP}}$ that defines the language of the underlying objective. This alphabet denotes semantically meaningful events observed in given states. We add the empty string ε to label semantically meaningless states, which can therefore be ignored as part of the grammatical inference procedure. Given a sequence of states $x_1, x_2, \ldots, x_k$, the corresponding trace is given by $L(x_1)L(x_2)\ldots L(x_k)$.

For simplicity, we assume a binary reward signal. Learning a reward machine is then equivalent to learning a DFA, allowing us to apply L^* directly. A trace is in the language iff it yields a reward of 1. The proposed approach can be generalized to any finitary reward signal, as any reward machine has an equivalent DFA representation (Xu et al. 2021).

Definition 2 (Deterministic Finite Automaton)**.** *A DFA D is a tuple $D = (Q, q_0, \Sigma, \delta, F)$, where Q is a finite set of states, $q_0 \in Q$ is a distinguished initial state, Σ is a finite input alphabet, $\delta : Q \times \Sigma \to Q$ is a transition function, and F is a set of accepting states.*

The quintessential algorithm to learn DFAs is L^* (Angluin 1987). This algorithm assumes that a teacher is available to answer membership and equivalence queries. Given a trace $w \in \Sigma^*$, membership queries ask whether w is in the language of the DFA to be learned. L^* incrementally builds a table (S, E) with rows S and columns E, where every entry in the table is answered using membership queries. Once the table is consistent and closed, defined below, a hypothesis DFA can be generated from the table. At this point, the teacher must answer an equivalence query to determine whether the hypothesis DFA captures the true language to be learned. If not, a counterexample is provided and integrated into (S, E). Membership queries are then answered to make (S, E) consistent and closed once again. The L^* algorithm repeats these steps until the correct DFA is learned.

In order to define consistency and closure, it is useful to define the notion of E-Equivalence.

Definition 3 (E-Equivalence)**.** *Given $E \subseteq \Sigma^*$, traces $w, w' \in \Sigma^*$ are E-equivalent with respect to language $\mathcal{L}$, denoted $w \equiv_E w'$, if $we \in \mathcal{L} \iff w'e \in \mathcal{L}$ for all $e \in E$.*

Definition 4 (Consistency)**.** *(S, E) is consistent if for all $s, s' \in S$, if $s \equiv_E s'$, then $s\ell \equiv_E s'\ell$ for all $\ell \in \Sigma$.*

Definition 5 (Closure)**.** *(S, E) is closed if for all $s \in S$ and $\ell \in \Sigma$, there exists some $s' \in S$ such that $s\ell \equiv_E s'$.*

In the context of RL and planning, the concept of a teacher and the problems of answering membership and equivalence queries take on a different form when compared to classical grammatical inference. In our setting, the teacher is the given NMRDP model M and the language to be learned is the DFA representation of the non-Markovian reward signal R. As such, answering membership and equivalence queries requires exploring the state space of M to find sequences of states that yield a reward. The traces corresponding to such sequences are used to learn the underlying DFA.

As we show, the seemingly innocuous problem of answering membership queries is **NP-complete** in settings where positional policies are required. A positional policy is one that is Markovian (i.e. only depends on the current state) and deterministic (Oura, Sakakibara, and Ushio 2020). Such are the settings of reinforcement learning using Q-learning and deep Q-networks. Due to this heretofore unknown complexity, a simple heuristic is employed in (Gaon and Brafman 2020) to make answering membership queries tractable. In particular, the query is answered negatively if some threshold of attempts to answer it is exceeded. The work in (Xu et al. 2021) handles this membership query problem in a different manner by posing it as a reinforcement learning problem. More specifically, their proposed approach defines two Q-functions from which policies are derived to solve the original reinforcement learning problem as well as the problem of generating a trajectory of states whose induced trace answers the underlying membership query. The latter is accomplished by rewarding the agent as it observes elements of the membership query in the correct order.

Complexity

The preceding discussion on $L^\star$ illustrates the importance of membership queries. We now formulate a decision problem related to membership queries and analyze its complexity in the context of positional policies.

Definition 6 (Positional Membership Query (PMQ) Problem)**.** *Given an NMRDP $M = (X, x_0, A, P, \mathsf{AP}, L, R)$ and a trace $w \in (2^{\mathsf{AP}})^+$, is there a sequence of states $x_0, x_1, \ldots, x_k$ in X achievable by a positional policy such that $L(x_0)L(x_1)\ldots L(x_k) = w$?*

We now recall the definitions of a problem related to graph homeomorphism, that we use to establish the complexity of the PMQ problem.

Definition 7 (Subgraph Homeomorphism (SH) Problem)**.** *Given directed graphs $G = (V, E)$ and $P = (V', E')$ and a one-to-one mapping $m_v : V' \to V$, the pattern graph P is said to be homeomorphic to some subgraph of G if there exists a mapping $m_e : E' \to V^+$ such that $m_e(v'_i, v'_j)$ maps $(v'_i, v'_j) \in E'$ to a simple path in G with starting vertex $m_v(v'_i)$ and destination vertex $m_v(v'_j)$ such that the simple paths given by the mapping m_e are pairwise node-disjoint. The restricted subgraph homeomorphism (RSH) problem asks the same question, but restricts P to be a line graph consisting of three vertices and two edges.*

By reduction from the RSH problem, which is known to be **NP-complete** (Fortune, Hopcroft, and Wyllie 1980), we establish that even a restricted version of the MQ problem is **NP-complete**. Intuitively, this RSH problem entails finding two node-disjoint simple paths visiting the nodes in the given pattern graph.

Theorem 1. *The PMQ problem is **NP-complete**.*

Proof. The PMQ problem is clearly in **NP** since we can verify whether $L(x_0)L(x_i), \ldots L(x_k) = w$ holds, for a given sequence of states in polynomial time. We show that the PMQ problem is **NP-hard** by reducing the RSH problem to the PMQ problem. Consider an arbitrary graph $G = (V, E)$, a pattern graph $P = (V', E')$ with $V' = \{v'_1, v'_2, v'_3\}$ and $E' = \{(v'_1, v'_2), (v'_2, v'_3)\}$, and a vertex mapping $m_v : V' \to V$. We construct a corresponding MDP as follows. For each $v \in V$, we have a state $x \in X$ and for each $(v_i, v_j) \in E$, we have a deterministic transition function $P(x_j, x_i, a_{ij}) = 1, a_{ij} \in A(x_i)$. The set of atomic propositions is given by $\mathsf{AP} = \{\ell_1, \ell_2, \ell_3\}$ and the labeling function maps states in the MDP according to the given node mapping $m_v : V' \to V$. That is, if $m_v(v'_i) = v_j$, then $L(x_j) = \{\ell_i\}$. For all vertices $v_j \notin m_v(v'_1) \cup m_v(v'_2) \cup m(v'_3)$, we have $L(x_j) = \varepsilon$. The reward signal can be defined arbitrarily and the starting state is chosen to be $x_0 \in L^{-1}(\ell_1)$, where $L^{-1} : 2^{\mathsf{AP}} \to 2^X$ denotes the state(s) where a given label holds. In this reduction, note that the inverse labeling function L^{-1} returns a single state. We can now show that a sequence $x_0, x_i, \ldots, x_k$ achievable by some positional policy in M such that $L(x_0, x_i, \ldots, x_k) = \ell_1\ell_2\ell_3$ exists if and only if G contains a subgraph homeomorphic to P.

($\implies$) Abusing notation, let $\vec{x}$ denote the sequence of states achievable by some positional policy with $L(\vec{x}) = \ell_1\ell_2\ell_3$. To show that the path $\vec{x}$ is a subgraph homeomorphic to P, we must show that it consists of simple paths from x_0 to $L^{-1}(\ell_2)$ and from $L^{-1}(\ell_2)$ to $L^{-1}(\ell_3)$ such that these simple paths are node-disjoint. This follows directly from the deterministic transition dynamics of M. Indeed, the out-degree of every state in $\vec{x}$ must be 1. Two cases arise. First, if the in-degree of every state in $\vec{x}$ is at most 1, then $\vec{x}$ is a simple path and is therefore node-disjoint. However, if some state in $\vec{x}$ has in-degree strictly greater than 1, then there must be a cycle. In this case, a simple path can be trivially obtained by removing all states from $\vec{x}$ observed after $L^{-1}(\ell_3) \in \vec{x}$.

($\impliedby$) Assume G contains a subgraph homeomorphic to P and let $m_e : E' \to V^+$ denote the mapping from edges of the pattern graph P to node-disjoint simple paths in G. Without loss of generality, let $v_1 = m_v(v'_1)$, $v_2 = m_v(v'_2)$, $v_3 = m_v(v'_3)$ denote the node mappings of the pattern graph. The sequence of states $\vec{x}$ is easily obtained from $m_e(v'_1, v'_2) = (m_v(v'_1), v_{i_1}, v_{i_2}, \ldots, m_v(v'_2))$ and $m_e(v'_2, v'_3) = (m_v(v'_2), v_{j_1}, v_{j_2}, \ldots, m_v(v'_3))$ as $\vec{x} = (x_1, x_{i_1}, x_{i_2}, \ldots, x_2, x_{j_1}, x_{j_2}, \ldots, x_3)$. This is a simple path and is thus achievable by some positional policy. □

Methodology

Since we are learning a DFA, we may implement the classic L^* algorithm directly. Let $\Sigma = 2^{\mathsf{AP}}$ denote the alphabet of the DFA. We need only specify how to perform membership and equivalence queries in the given NMRDP M. To answer such queries, we need one additional assumption. For every trace $w \in \Sigma^*$, we must be able to query if the trace receives a reward of 0 or 1 in M. Thus, each trace $w = \ell_1 \ldots \ell_n$ must be observable in M via some sequence of states $x_0, x_1, \ldots, x_k$ so that $L(x_0)L(x_1)\ldots L(x_k) = w$. Recall that states with label ε are ignored in the computation of the trace, making this a reasonable assumption.

Unlike the positional setting where we have proven that answering membership queries is **NP-complete**, answering these queries can be done straightforwardly in the non-Markovian planning context by using an iterative Breadth-First Search (BFS) as follows. For $w \in \Sigma^*$, we must search for a trajectory in M with corresponding trace w and observe the resulting reward. Suppose $w = \ell_1 \ldots \ell_n$. Starting from x_0, we remove all states with labels not equal to ℓ_1 or ε. We then BFS to all states $x^{\ell_1} \subset X$ in M labeled ℓ_1 and reintroduce all states that were removed prior. We remove all states not labeled ℓ_2 or ε and perform BFS from the set of states x^{l_1} to the states $x^{l_2} \subset X$ labeled ℓ_2, and so on. We then observe the reward from an arbitrary sequence of states derived from the preceding iterative BFS whose induced trace is w, where a reward of 1 denotes that the corresponding trace is in the language of the DFA we are trying to learn from M. It is worth noting that many different state sequences may correspond to the same trace w. We thus assume that all such state sequences yield the same reward.

Once enough membership queries have been solved so that (S, E) is closed and consistent, a hypothesis DFA H is derived per standard methods. Given H, we must issue an equivalence query to determine if H captures the language of the non-Markovian reward signal R. Methods like JIRP

Algorithm 1: Membership Query

Function MEMBERSHIPQUERY(M, w)**:**
  Suppose $w = \ell_1 \cdots \ell_n$
  $x \leftarrow x_0$
  $trajectory \leftarrow ()$
  for $1 \leq i \leq n$ **do**
    // BFS returns trajectory from x to state labeled ℓ_i
    $trajectory \leftarrow trajectory \cdot \text{BFS}(M, x, \ell_i)$
    $x \leftarrow$ last state in $trajectory$
  return $R(trajectory)$

Algorithm 2: Equivalence Query

Data: Ω
Function EQUIVALENCEQUERY(M, H)**:**
  for $w \in \Sigma^*$ *with* $|w| \leq \Omega$ **do**
    $reward \leftarrow$ MEMBERSHIPQUERY(M, w)
    $y \leftarrow \delta^*(y_0, w)$
    if $reward = 1$ *and* $y \notin F$ *or* $reward = 0$ *and* $y \in F$ **then**
      return w
  return None

(Xu et al. 2020) and AFRAI (Xu et al. 2021) accomplish this by implementing Q-learning over the product $M \times H$ until a counterexample is observed or some time limit is exceeded. However, in the planning context we can check equivalence to a limited extent using our access to the model. We sample all traces $w \in \Sigma^*$ up to some length Ω. In our experiments, we set $\Omega = 5$. We can then check that H agrees with the model about the reward of each w. The trace w is a counterexample if it reaches an accepting state of H but yields a reward of 0 in M, or if it does *not* reach an accepting state of H but *does* yield a reward of 1 in M. Once there are no more counterexamples of this kind, we run value iteration on $M \times H$ and begin to execute the policy. We can also explore during execution to continue searching for longer counterexamples. If we ever find one, we pass back to a learning phase to rebuild the automaton and rerun value iteration.

Experiments

We evaluate our algorithm on three different gridworld tasks. Tasks 1 and 2 are in the 9×12 office gridworld introduced in (Icarte et al. 2018). Task 1 consists of delivering the coffee to the office without stepping on any decorations (See Figure 1 in (Icarte et al. 2018)). Task 2 entails repeatedly patroling the office (See Figure 2 in (Icarte et al. 2018)). Task 3 is the Minecraft-like gridworld introduced in (Andreas, Klein, and Levine 2017), where the objective is to build a spear by gathering wood, string, and stone in any order, then reaching a workbench (See Figure 3 in (Andreas, Klein, and Levine 2017)). We evaluate the time and sample efficiency of our non-Markovian planning method compared to the JIRP and AFRAI methods presented in (Xu et al. 2020) and (Xu et al. 2021), respectively. A sample in our method includes the queries to the model during learning and the environment samples observed during execution. We ran the proposed approach for 20 runs and averaged the observed rewards. The episode lengths and training steps used were 150, 200, 100 and 10,000, 20,000, and 10,000, respectively, for Tasks 1, 2, and 3. See Figure 1 for results. Each method is run for the same number of training steps. Since these differ in execution speed, some graph curves end sooner than others.

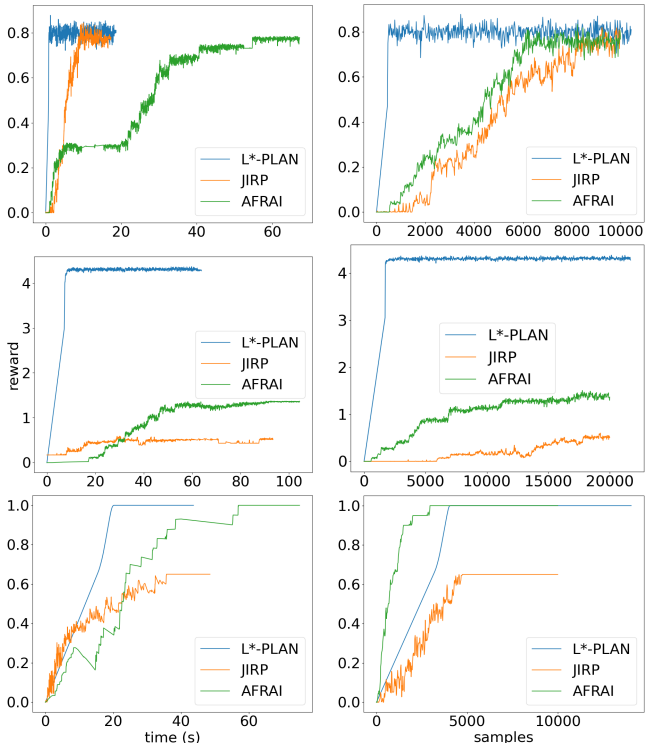

Figure 1: Results for Tasks 1 (*top*), 2 (*middle*), 3 (*bottom*).

In each task, our method always learned the correct automaton exactly. JIRP also learned the correct automaton for each run of Task 1, but never did for Tasks 2 and 3. AFRAI always learned the correct automaton in Task 1, learned the correct automaton in Task 2 for 85% of runs, and learned the correct automaton in Task 3 only 5% of runs. AFRAI appears competitive in sample efficiency in the last graph. This is due to how we count samples. In our equivalence query, we sample many short sequences from our model. AFRAI uses the same L^*-based algorithm we do, and so otherwise queries a similar number of traces, each of which takes longer. Our extra equivalence queries account for our ability to learn an exact automaton when AFRAI does not.

Conclusion

For decades, the planning community has studied how to dispense with the Markovian assumption by instead assuming some automaton representation of the objective is given. In this paper, we addressed the problem where this automaton is not known and must be learned from interactions with the environment. While this idea has been explored in the context of RL, we showed how in the planning context, an environment model can be exploited to speed up learning. We also established complexity results for the adoption of active grammatical inference under positional policies.

Acknowledgments

This research was supported in part by the Air Force Research Laboratory through the Information Directorate's Information Institute® Contract Number FA8750-20-3-1003 and FA8750-20-3-1004, the Air Force Office of Scientific Research through Award 20RICOR012, and the National Science Foundation through CAREER Award CCF-1552497 and Award CCF-2106339.

References

Andreas, J.; Klein, D.; and Levine, S. 2017. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, 166–175. PMLR.

Angluin, D. 1987. Learning regular sets from queries and counterexamples. *Information and computation*, 75(2): 87–106.

Bacchus, F.; Boutilier, C.; and Grove, A. 1997. Structured solution methods for non-Markovian decision processes. In *AAAI/IAAI*, 112–117. Citeseer.

Brafman, R. I.; and De Giacomo, G. 2019. Planning for LTLf/LDLf Goals in Non-Markovian Fully Observable Nondeterministic Domains. In *IJCAI*, 1602–1608.

Fortune, S.; Hopcroft, J.; and Wyllie, J. 1980. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2): 111–121.

Gaon, M.; and Brafman, R. 2020. Reinforcement Learning with Non-Markovian Rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 3980–3987.

Gretton, C. 2006. Properties of Planning with Non-Markovian Rewards. In *Journal of Artificial Intelligence Research*. Citeseer.

Hasanbeig, M.; Abate, A.; and Kroening, D. 2018. Logically-constrained reinforcement learning. *arXiv preprint arXiv:1801.08099*.

Hasanbeig, M.; Jeppu, N. Y.; Abate, A.; Melham, T.; and Kroening, D. 2021. DeepSynth: Automata Synthesis for Automatic Task Segmentation in Deep Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 7647–7656.

Icarte, R. T.; Klassen, T.; Valenzano, R.; and McIlraith, S. 2018. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, 2107–2116.

Oura, R.; Sakakibara, A.; and Ushio, T. 2020. Reinforcement Learning of Control Policy for Linear Temporal Logic Specifications Using Limit-Deterministic Generalized Büchi Automata. *IEEE Control Systems Letters*, 4(3): 761–766.

Rens, G.; and Raskin, J.-F. 2020. Learning non-markovian reward models in mdps. *arXiv preprint arXiv:2001.09293*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587): 484.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of Go without human knowledge. *Nature*, 550(7676): 354.

Thiébaux, S.; Gretton, C.; Slaney, J.; Price, D.; and Kabanza, F. 2006. Decision-theoretic planning with non-Markovian rewards. *Journal of Artificial Intelligence Research*, 25: 17–74.

Thiébaux, S.; Kabanza, F.; and Slaney, J. 2002. Anytime state-based solution methods for decision processes with non-Markovian rewards. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, 501–510.

Velasquez, A.; Bissey, B.; Barak, L.; Beckus, A.; Alkhouri, I.; Melcer, D.; and Atia, G. 2021. Dynamic Automaton-Guided Reward Shaping for Monte Carlo Tree Search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 12015–12023.

Xu, Z.; Gavran, I.; Ahmad, Y.; Majumdar, R.; Neider, D.; Topcu, U.; and Wu, B. 2020. Joint inference of reward machines and policies for reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, 590–598.

Xu, Z.; Wu, B.; Ojha, A.; Neider, D.; and Topcu, U. 2021. Active Finite Reward Automaton Inference and Reinforcement Learning Using Queries and Counterexamples. 12844: 115–135.

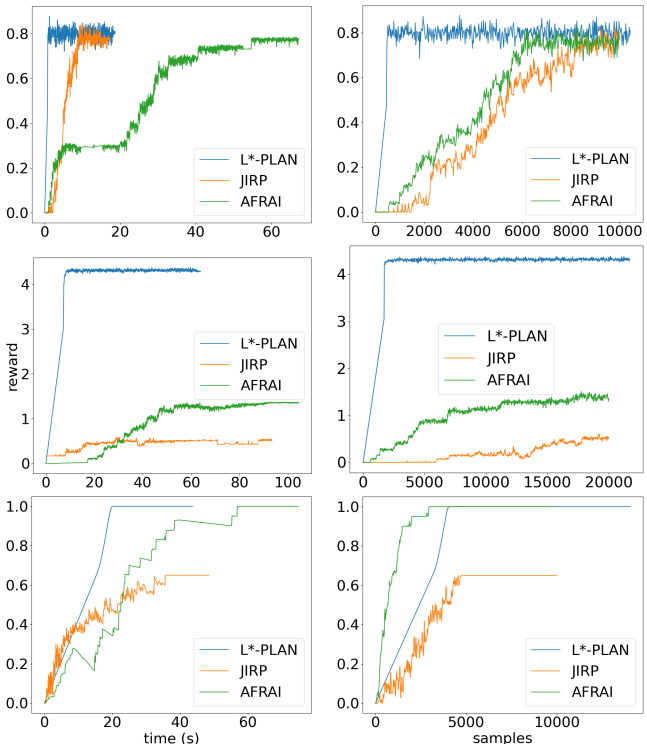

Figure 2: Results for Tasks 1 (*top*), 2 (*middle*), 3 (*bottom*).