Joint Learning of Reward Machines and Policies in Environments with Partially Known Semantics

Christos K. Verginis

Uppsala University, Uppsala, Sweden

Cevahir Koprulu, Sandeep Chinchali, Ufuk Topcu

University of Texas at Austin, Austin, Texas, USA

Abstract

We study the problem of reinforcement learning for a task encoded by a reward machine. The task is defined over a set of properties in the environment, called atomic propositions, and represented by boolean variables. One unrealistic assumption commonly used in the literature is that the truth values of these propositions are accurately known. In real situations, however, these truth values are uncertain since they come from sensors that suffer from imperfections. At the same time, reward machines can be difficult to model explicitly, especially when they encode complicated tasks. We develop an algorithm that infers the reward machine and learns how to accomplish the underlying task despite the uncertainties of the propositions' truth values. In order to address such uncertainties, the algorithm maintains a probabilistic estimate about the truth value of the atomic propositions; it updates this estimate according to new sensory measurements that arrive from the exploration of the environment. Additionally, the algorithm maintains a hypothesis reward machine, which acts as an estimate of the reward machine that encodes the task to be learned. As the agent explores the environment, the algorithm updates the hypothesis reward machine according to the obtained rewards and the estimate of the atomic

Email addresses: `christos.verginis@angstrom.uu.se` (Christos K. Verginis), `cevahir.koprulu@utexas.edu` (Cevahir Koprulu), `sandeepc@utexas.edu` (Sandeep Chinchali), `utopcu@utexas.edu` (Ufuk Topcu)

propositions' truth value. Finally, the algorithm uses a Q-learning procedure for the states of the hypothesis reward machine to determine the policy that accomplishes the task. We prove that the algorithm successfully infers the reward machine and asymptotically learns a policy that accomplishes the respective task.

Keywords: Reinforcement learning; Reward machines; Perception limitations; Uncertain

1. Introduction

Reinforcement learning (RL) studies the problem of learning an optimal behavior for autonomous agents with unknown dynamics in potentially unknown environments. The learning agent selects actions and navigates through the environment collecting rewards based on a reward function [1]. A variety of works incorporate high-level knowledge that can help the agent explore the environment more efficiently [2]. This high-level knowledge may be expressed as different levels of temporal or behavioral abstractions, or a hierarchy of abstractions [3, 4, 5, 6].

Recently, the authors in [7] proposed the concept of *reward machines* in order to provide high-level information to the agent in the form of rewards. Reward machines are discrete, finite-state structures that encode a possibly non-Markovian reward function, which consists of several temporally related subtasks, such as "bring coffee to the office without encountering obstacles". Reward machines allow the composition of such subtasks in flexible ways, including concatenations, loops, and conditional rules. In that way, the programmer exposes high-level structural relationships to the learning agent. Intuitively, as an agent acts in the environment, moving from one state to another, it also moves between states within a reward machine, as determined by high-level properties detected within the environment, called *atomic propositions.* After every transition, the reward machine outputs the reward the agent should obtain according to the encoded reward function. Furthermore, [7] develops a q-learning method that learns that optimal policy associated with the reward function that is encoded by the reward machine.

Despite their attractive performance, works using reward machines make two unrealistic assumptions. First, the underlying reward machine is assumed to be explicitly known by the learning agent. However, in many practical situations, the reward machines are too complex to encode, and the high-level structural relationships among the subtasks are implicit. Secondly, the location of the atomic propositions that drive the transitions of the reward machine are assumed to be a priori known and *accurately* detected within the environment. Nevertheless, several practical scenarios involve agents operating in a priori unknown environments, endowed with sensors that naturally suffer from imperfections, such as misclassifications or missed object detections from state-of-the-art computer vision models.

In this paper, we investigate the RL problem for an autonomous agent to deliver a reward-machine-encoded task over a set of atomic propositions of the environment. The reward machine and the location of the atomic propositions are a priori unknown to the agent, which renders the considered setup significantly challenging. Additionally, the agents perceives the atomic propositions through sensor units that are subject to uncertainties, which are modeled via an observation function.

Our contribution lies in the development of an algorithm that guarantees the learning of a policy that achieves the given task despite the aforementioned challenges. The algorithm holds a probabilistic belief over the distribution of the atomic propositions in the environment and deploys a hypothesis reward machine to account for the task uncertainty. The belief and hypothesis reward machine are iteratively updated using new sensory measurements and the obtained rewards of the agent's trajectories. We provide theoretical guarantees about the algorithm's convergence, and experimental results show its robustness with respect to inaccurate observation functions.

2. Related Work

Previous RL works on giving an agent higher-level structure and knowledge about the reward function focus mostly on abstractions and hierarchical RL; in such cases, a meta-controller decides which subtasks to perform, and a controller decides which actions to take within a subtask (e.g., [8, 3, 6]). Other works use temporal logic languages to express a specification, and then generate the corresponding reward functions [9, 10, 11, 12, 13, 14, 15, 16, 17, 18]. Inspired by [7], we use in this work the concept of reward machines, which encode more compactly and expressively high-level information on the reward function of the agent. Additionally, unlike the works in the related literature, we consider that the reward machine that encodes the task at hand is a priori *unknown*.

There exists a large variety of works dealing with perception uncertainty [19, 20, 21, 22, 23, 24, 25, 26, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37]. However, most of these works consider the *planning* problem, which consists of computing an optimal policy by employing the underlying agent model. Furthermore, the considered uncertainty usually arises from probabilistic dynamics, which is resolved either by active-perception procedures, belief-state propagation, or analysis with partially observable Markov decision processes (POMDPs). Besides, the works that consider high-level reward structure (like temporal logic [19, 22, 27, 29]), assume full availability of the respective reward models.

The recent works [38, 39, 40] consider tasks encoded by *a priori unknown* automata-based structures, which they infer from the system's trajectories. The authors in [40], however, do not consider any perception uncertainty, whereas [38] and [39] assume partial state observability by employing POMDP models. In contrast, in this work we consider uncertainty in the semantic representation of the underlying environment, i.e., the location of the environment properties that define the agent's task is a priori unknown.

3. Problem Formulation

We now introduce the agent model, the environment model, the observation model and the reward machine-encoded task specification that we use in the formal problem statement.

3.1. Agent Model

We model the interaction between the agent and the environment by a Markov decision process (MDP) [41, 42].

Definition 1. *A Markov decision process (MDP) is a tuple* $\mathcal{M} = (S, s_I, A, \mathcal{T}, R, \mathcal{AP}, L_G, \hat{L}, \gamma)$ *consisting of a finite state space* S*, an agent's initial state* $s_I \in S$*, a finite set of actions* A*, and a probabilistic transition function* $\mathcal{T} : S \times A \times S \to [0, 1]$*. A reward function* $R : S^* \times A \times S \to \mathbb{R}$ *and a discount factor* $\gamma \in [0, 1)$ *together specify payoffs to the agent;* $\mathcal{AP}$ *is a finite set of binary-valued atomic propositions, taking values in* $\{\mathsf{True}, \mathsf{False}\}$*,* $L_G : S \to 2^{\mathcal{AP}}$ *is the ground-truth labeling function, and* $\hat{L} : S \to 2^{\mathcal{AP}}$ *is a labeling function estimate. We define the size of* $\mathcal{M}$*, denoted as* $|\mathcal{M}|$*, to be* $|S|$ *(i.e., the cardinality of the set* S*).*

The atomic propositions $\mathcal{AP}$ is a set of task-relevant labels in the environment; we denote by $s \models p$ if an atomic proposition $p \in \mathcal{AP}$ holds true at state $s \in S$. The ground-truth labeling function L_G provides the location where the atomic propositions hold true, i.e., $L_G(s) = P \subseteq \mathcal{AP}$ is equivalent to $s \models p$, for all $p \in P$. We consider that L_G is *unknown* to the agent, which perceives the environment, and hence the location of $\mathcal{AP}$, through sensor units such as cameras or range sensors. However, such sensors suffer from imperfections (e.g., noise). Therefore, the agent maintains a time-varying probabilistic belief $\hat{\mathcal{L}}_i : S \times 2^{\mathcal{AP}} \to [0, 1]$ about the truth values of the atomic propositions, for some time step i. More specifically, for a state $s \in S$ and a subset of atomic propositions $P \subseteq \mathcal{AP}$, $\hat{\mathcal{L}}_i(s, P)$ assigns the probability of the event that P holds true at s, i.e., $\hat{\mathcal{L}}_j(s, P) = \mathsf{Pr}(\bigcap_{p \in P} s \models p)$. Notice that at each time step i and for every state $s \in S$, it holds that $\sum_{P \subset \mathcal{AP}} \hat{\mathcal{L}}_i(s, P) = 1$. The prior belief of the agent might be an uninformative prior distribution. We assume that the truth values of the propositions are mutually independent in each state, i.e.,

$$Pr\Big(\bigcap_{p\in P} s \models P\Big) = \prod_{p\in P} Pr(s \models p), \quad \forall s \in S, P \subset \mathcal{AP}$$

and also between every pair of different triples,

$$Pr\big(s \models P \wedge s' \models P'\big) = Pr(s \models P)Pr(s' \models P'), \forall s, s' \in S, P, P' \subset \mathcal{AP}$$

This independence assumption facilitates the update of the labeling function over time. Nevertheless, so long as the joint distribution model is known, the updates can be computed.

The agent's belief $\hat{\mathcal{L}}_i$, for some time index $i \geq 0$, determines the agent's inference of the environment configuration, and hence the estimate labeling function $\hat{L}$. More specifically, we define $\hat{L} : (S \times 2^{\mathcal{AP}} \to [0,1]) \times S \to 2^{\mathcal{AP}}$, with

$$\hat{L}(\hat{\mathcal{L}}_i, s) = \{p \in \mathcal{AP} : \hat{\mathcal{L}}_i(s,p) \geq 0.5\},$$

i.e, the most probable outcomes.

Example. *Let the 10-state office workspace shown in Fig. 1. The set of atomic propositions is $\mathcal{AP} = \{\text{c,o,X}\}$, corresponding to "coffee", "office", and "obstacle", respectively. The ground-truth labeling function, shown at the top of Fig. 1, is defined as $L_G(s_{12}) = \{\text{c}\}$, $L_G(s_{22}) = L_G(s_{23}) = \{\text{X}\}$, $L_G(s_{24}) = \{\text{o}\}$, and $\emptyset$ in the rest of the states. Imperfect sensory measurements might lead to an estimate labeling function $\hat{L}$ as shown at the bottom of Fig. 1, i.e., $\hat{L}(s_{12}) = \{\text{c,X}\}$, $\hat{L}(s_{22}) = \{\text{c}\}$, $\hat{L}(s_{14}) = \hat{L}(s_{24}) = \{\text{o}\}$, $\hat{L}(s_{23}) = \{\text{o}, X\}$, and $\emptyset$ in the rest of the states.*

A policy is a function mapping states in S to a probability distribution over actions in A. At state $s \in S$, an agent using policy π picks an action a with probability $\pi(s,a)$, and the new state s' is chosen with probability $\mathcal{T}(s,a,s')$. A policy π and the initial states s_I together determine a stochastic process and we write $S_0A_0S_1$...for the random trajectory of states and actions.

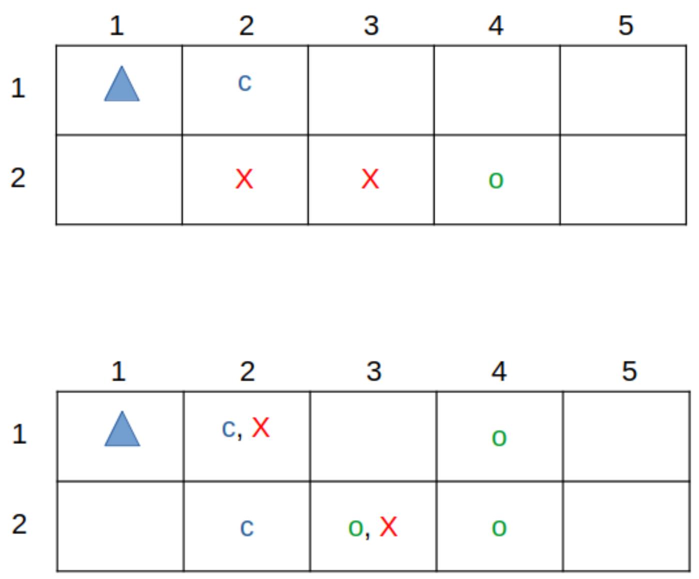


Figure 1: Top: A workspace with 10 states s_{ij} illustrating the ground-truth labeling function $\mathcal{L} : S \rightarrow 2^{\{\text{c,o,X}\}}$, were {c,o,X} stand for {"coffee", "office", "obstacle"}, respectively; Bottom: Perception uncertainty leads to the estimate $\hat{L}(\mathcal{L}, s) : S \rightarrow 2^{\{\text{c,o,X}\}}$, according to which the truth value of several properties is incorrect (e.g. X in s_{12} or the absence of X in s_{22}).

A trajectory is a realization of the agent's stochastic process $S_0 A_0 S_1$: a sequence of states and actions $s_0 a_0 s_1 \dots s_k a_k s_{k+1}$ with $s_0 = s_I$. Its corresponding label sequence is $\ell_0 \ell_1 \dots \ell_k$ with $\ell_j = L_G(s_j)$ for all $j \leq k$. Similarly, the reward sequence is $r_0 r_1 \dots r_k$, where $r_j = R(s_0 \dots s_j a_j s_{j+1})$ for all $j \leq k$. A trajectory $s_0 a_0 s_1 \dots s_k a_k s_{k+1}$ achieves a reward $\sum_{j=0}^{k} \gamma^j R(s_0 \dots s_j a_j s_{j+1})$. Note that the definition of the reward function assumes that the reward is a function of the whole trajectory; this allows the reward function to be non-Markovian, if it distinguishes among histories that can be represented by regular languages over $\mathcal{AP}$ [7]. Given a trajectory $s_0 a_0 s_1 \dots s_{k+1}$ and a fixed belief $\hat{\mathcal{L}}_i$, we naturally define then the *observed* label sequence by $\hat{\ell}_0 \hat{\ell}_1 \dots \hat{\ell}_k$, with $\hat{\ell}_j = \hat{L}(\hat{\mathcal{L}}_i, s_j)$ for all $j \leq k$ and some fixed $i \geq 0$.

Example (Continued)**.** *Let the 10-state office workspace shown in Fig. 1, and a trajectory $s_{11} s_{12} s_{13} s_{14} s_{24}$. According to the ground-truth labeling function (see top of Fig. 1), such a trajectory would would produce the label and reward sequence $(\emptyset, 0)(c, 0)(\emptyset, 0)(\emptyset, 0)(o, 1)$. However, the observed label sequence, based on the estimate $\hat{L}(\hat{\mathcal{L}}, \cdot)$ (see bottom of Fig. 1), is* $(\emptyset, 0)$ ((c,X), 0) $(\emptyset, 0)$(o, 0)(o, 1).

3.2. Reward Machines

Encoding a (non-Markovian) reward in a type of finite state-machine is achieved by reward machines [7], formally defined in the next definition.

Definition 2. *A reward machine is the tuple $\mathcal{A} = (V, v_I, 2^{\mathcal{AP}}, \mathcal{R}, \delta, \sigma)$ consisting of a finite set of states V, an initial state $v_I \in V$, an input alphabet $\mathcal{R}$, a (deterministic) transition function $\delta : V \times 2^{\mathcal{AP}} \to V$, and an output function $\sigma : V \times 2^{\mathcal{AP}} \to \mathcal{R}$. We define the size of $\mathcal{A}$, denoted by $|\mathcal{A}|$, to be $|V|$ (i.e., the cardinality of the set V).*

The run of a reward machine $\mathcal{A}$ on a sequence of labels $\ell_0 \dots \ell_k \in (2^{\mathcal{AP}})^*$ is a sequence $v_0(\ell_0, r_0) v_1(\ell_1, v_1) \dots\ v_k\ (\ell_k, v_k)$ of states and label-reward pairs such that $v_0 = v_I$ and for all $j \in \{0, \dots, k\}$, we have $\delta(v_j, \ell_j) = v_{j+1}$ and $\sigma(v_j, \ell_j) = r_j$. We write $\mathcal{A}(\ell_0 \dots \ell_k) = r_0 \dots r_k$ to connect the input label sequence to the sequence of rewards produced by the reward machine $\mathcal{A}$. We say that a reward machine $\mathcal{A}$ encodes the reward function R of an MDP *on the ground truth* if, for every trajectory $s_0a_0 \dots s_ka_ks_{k+1}$ and the corresponding label sequence $\ell_0 \dots \ell_k$, the reward sequence equals $\mathcal{A}(\ell_0 \dots \ell_k)$. Moreover, given a fixed $i \geq 0$, we say that a reward machine $\mathcal{A}$ *encodes the reward function R on $\hat{\mathcal{L}}_i$* if for every trajectory $s_0a_0 \dots s_ka_ks_{k+1}$ and the corresponding *observed* label sequence $\hat{\ell}_0 \dots \hat{\ell}_k$ the reward sequence equals $\mathcal{A}(\hat{\ell}_0 \dots \hat{\ell}_k)$. Note, however, that there might not exist a reward machine that encodes the reward function on $\hat{\mathcal{L}}_i$, as the next example shows.

Example (Continued)**.** *Let the 10-state office workspace shown in Fig. 1 and the trajectory $s_{11}s_{12}s_{13}s_{14}s_{24}$. As stated before, the trajectory produces (via the ground-truth labeling function) the label and reward sequence* $(\emptyset, 0)(c, 0)(\emptyset, 0)(\emptyset, 0)$ $(o, 1)$*. Such a sequence produces the run of the reward machine*

$$v_0(\emptyset, 0)v_0(c, 0)v_1(\emptyset, 0)v_1(\emptyset, 0)v_1(o, 1)v_3.$$

The observed label sequence, based on the estimate $\hat{L}(\hat{\mathcal{L}}, \cdot)$, is $(\emptyset, 0)$ $((\mathrm{c{,}X}), 0)$ $(\emptyset, 0)(\mathrm{o}, 0)(\mathrm{o}, 1)$*, which produces the reward-machine run*

$$v_0(\emptyset, 0)v_0((\mathrm{c{,}X}), 0)v_2(\emptyset, 0)v_2(\mathrm{o}, 0)v_2(\mathrm{o}, 0)v_2,$$

*which clearly does not comply with the observed reward sequence. Hence, we conclude that the reward machine does not encode the reward function on $\hat{\mathcal{L}}$. In fact, note that we cannot construct a reward machine that encodes the reward function on $\hat{\mathcal{L}}$. The actual reward function of bringing coffee from s_{12} to the office in s_{24} cannot be expressed via the set $\mathcal{AP}$ with this specific $\mathcal{L}$, because of the ambiguity in the states satisfying "*o*" and "*c*"; a single accepting state trajectory does not correspond to a unique observed label sequence.*

3.3. Observation Model

The probabilistic labeling function is updated based on an observation model $\mathcal{O}$. More specifically, at each time step, the agent's perception module processes a set of sensory measurements regarding the atomic propositions $\mathcal{AP}$. In the

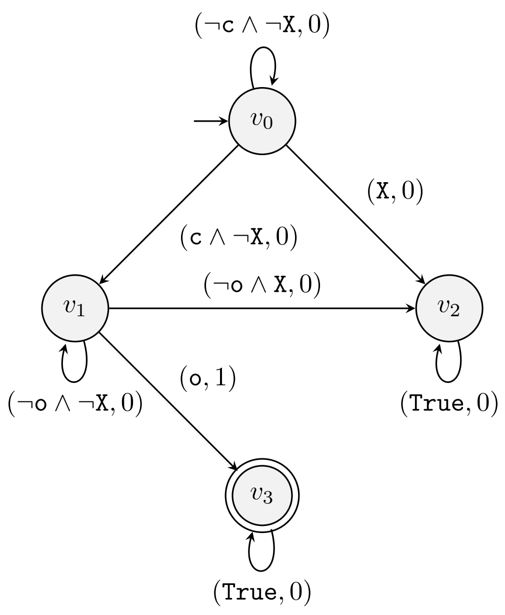

$(\neg \texttt{c} \wedge \neg \texttt{X}, 0)$
v_0
$(\texttt{X}, 0)$
$(\texttt{c} \wedge \neg \texttt{X}, 0)$
$(\neg \texttt{o} \wedge \texttt{X}, 0)$
v_1
v_2
$(\texttt{o}, 1)$
$(\neg \texttt{o} \wedge \neg \texttt{X}, 0)$
$(\texttt{True}, 0)$
v_3
$(\texttt{True}, 0)$

Figure 2: Reward machine for the task of Fig. 1.

Bayesian framework, the observation model is used for the update of the agent's belief, as specified in the following definition.

Definition 3. *Let* $\mathcal{Z}(s_1, s_2, p) \in \{\mathsf{True}, \mathsf{False}\}$ *denote the perception output of the agent, when at state* s_1 *for the atomic proposition* $p \in \mathcal{AP}$ *at* s_2*. The joint observation model of the agent is* $\mathcal{O} : S \times S \times \mathcal{AP} \times \{\mathsf{True}, \mathsf{False}\} \to [0, 1]$*, where* $\mathcal{O}(s_1, s_2, p, b)$ *represents the probability that* $\mathcal{Z}(s_1, s_2, p) = \mathsf{True}$ *if the truth value of* p *is according to* b*.*

In particular, an accurate observation model is the one for which the output probability of $\mathcal{O}(s_1, s_2, p, b)$ is one for $b = \mathsf{True}$ and zero for $b = \mathsf{False}$. In the Bayesian framework, the observation model is used for the update of the agent's belief. Nevertheless, in the absence of such observation model, one can perform the update in a frequentist way.

3.4. Problem Statement

We consider an MDP-modeled autonomous agent, whose task is encoded by a reward machine, initially unknown. Furthermore, the semantic representation of the environment is initially unknown to the agent, which, however, gathers sensory observations to revise its belief. The formal definition of the problem statement is as follows.

Problem 1. *Let an MDP agent model $\mathcal{M} = (S, s_I, A, \mathcal{T}, R, \mathcal{AP}, L_G, \hat{L}, \gamma)$ with unknown ground-truth labeling function L_G and transition function $\mathcal{T}$, an observation model $\mathcal{O}$, and an unknown reward machine $\mathcal{A}$ encoding an assigned task. Develop an algorithm that learns a policy π that maximizes $Z(s_I) = \mathbb{E}_\pi[\sum_{i=0}^{\infty} \gamma^i R(S_0, A_0 \dots S_{i+1})]$.*

4. Main Results

This section gives the main results of this paper, which is a joint perception and inference of policies and reward machines. We first give an overview of the JIRP algorithm of [40], which handles the joint inference of reward machines and policies, with perfect knowledge of the environment semantics.

The JIRP algorithm [40] (see Algorithm 1) aims at learning the optimal policy for maximizing $Z(s_I)$ by maintaining hypothesis reward machines. Its main component is the QRM_episode() (shown in Algorithm 2), originally proposed in [7] for accurately known reward machines. QRM_episode() maintains a set Q of q-functions, denoted as q^v for each state v of the reward machine. The current state v of the reward machine guides the exploration by determining which q-function is used to choose the next action (line 3 of Alg. 2). However, in each single exploration step, the q-functions corresponding to all reward machine states are updated (lines 7 and 11 of Alg. 2). Note that the returned rewards in ρ are *observed* (line 6 of Alg. 2), since the reward machine encoding the actual reward function is not known. Instead, JIRP operates with a hypothesis reward machine, which is updated using the traces of QRM_episode(). In particular, the episodes of QRM are used to collect traces and update q-functions. As long

Algorithm 1 JIRP Algorithm

```
1: $\mathcal{H} \leftarrow \mathsf{Initialize}(V)$
2: $Q = \{q^v | v \in V\} \leftarrow \mathsf{Initialize}()$
3: $X \leftarrow \emptyset$
4: for episode $n = 1, 2, \ldots$ do
5:     $(\lambda, \rho, Q) \leftarrow$ QRM_episode$(\mathcal{H}, Q, \mathcal{L})$
6:     if $\mathcal{H}(\lambda) \neq \rho$ then
7:         $\mathsf{Add}(X, (\lambda, \rho))$
8:         $\mathcal{H} \leftarrow \mathsf{Infer}(X)$
9:         $Q \leftarrow \mathsf{Initialize}()$
10:    end if
11: end for
```

as the traces are consistent with the current hypothesis reward machine, QRM explores more of the environment using the reward machine to guide the search. However, if a trace (λ, ρ) is detected that is inconsistent with the hypothesis reward machine (i.e., $\mathcal{H}(\lambda) \neq \rho$, line 6 of Alg. 1), JIRP stores it in a set X (line 7 of Alg. 1) - the trace (λ, ρ) is called a counterexample and the set X a sample. Once the sample is updated, the algorithm re-learns a new hypothesis reward machine (line 8 of Alg. 1) and proceeds.

4.1. Proposed Algorithm

In this paper, we extend the JIRP algorithm to take into account the uncertainty in the labeling function. More specifically, our algorithm tries to infer the reward machine $\mathcal{A}$ that encodes the reward function on the most probable outcomes, $\hat{L}(\hat{\mathcal{L}}_j, \cdot)$, i.e., $\mathcal{A}(\hat{\ell}_0, \ldots, \hat{\ell}_k) = r_0 \ldots r_k$ for some $j \geq 0$. At the same time, the agent's belief is updated at every time step based on the observation model of Definition 3. Similarly to [19], if the agent's knowledge about the environment has changed significantly, the agent then updates its belief $\hat{\mathcal{L}}$ and aims to infer a new reward machine.

Algorithm 3 illustrates the aforementioned procedure. The algorithm uses a

Algorithm 2 QRM_episode($\mathcal{H}$, Q, $\mathcal{L}$)

Input: A reward machine $\mathcal{H} = (V, v_I, 2^{\mathcal{P}}, \mathbb{R}, \delta, \sigma)$, a set of q-functions $Q = \{q^v | v \in V\}$, a labeling function $\mathcal{L}$.

```
1:  $s \leftarrow$ InitialState(); $v \leftarrow v_I$, $\lambda \leftarrow []$; $\rho \leftarrow []$
2:  for $0 \leq t < eplength$ do
3:      $a \leftarrow$ GetEpsilonGreedyAction($q^v, s$)
4:      $s' \leftarrow$ ExecuteAction($s, a$)
5:      $v' \leftarrow \delta(v, \mathcal{L}(s, a, s'))$
6:      $r \leftarrow$ ObserveReward()
7:      $q^v(s, a) \leftarrow$ Update($r$)
8:      for $\hat{v} \in V \backslash v$ do
9:          $\hat{v}' \leftarrow \delta(\hat{v}, \mathcal{L}(s, a, s'))$
10:         $\hat{r} \leftarrow \sigma(\hat{v}, \mathcal{L}(s, a, s'))$
11:         $q^{\hat{v}}(s, a) \leftarrow$ Update($\hat{r}$)
12:     end for
13:     Append($\lambda, \mathcal{L}(s, a, s')$); Append($\rho, r$)
14:     $s \leftarrow s'$; $v \leftarrow v'$
15: end for
16: return $(\lambda, \rho, Q)$
```

hypothesis reward machine $\mathcal{H}$ (line 1 of Alg. 3) to guide the learning process. Similarly to [40], it runs multiple QRM episodes to update the Q functions, and uses the collected traces to update the counterexamples in the set X and $\mathcal{H}$ (lines 7-12 of Alg. 3). Nevertheless, in our case, $\mathcal{H}$ aims to approximate the reward machine $\mathcal{A}$ that encodes the reward function on the most probable outcomes, $\hat{L}(\hat{\mathcal{L}}_h, \cdot)$, since that is the available information to the agent; $\hat{\mathcal{L}}_h$ is the "current" belief that the agent uses in its policy learning and reward-machine inference. This is illustrated via a modified version of the QRM episode algorithm (QRM_episode_mod invoked in line 7 of Alg. 3), shown in Algorithm 4. In particular, QRM_episode_mod uses the probabilistic belief $\hat{\mathcal{L}}_h$ in order to navigate in the hypothesis reward machine $\mathcal{H}$ (lines 5, 9 of Alg. 4). Moreover, $\hat{\mathcal{L}}_h$ is used to update the counterexample set X with the inconsistent traces (line 15 of Alg. 4 and lines 8-12 of Alg. 3). Hence, Alg. 3 aims to infer the reward machine that encodes the reward function on the belief $\hat{\mathcal{L}}_h$. Of course, as illustrated in Example 3.1, there might not exist a reward machine that encodes the reward function on $\hat{\mathcal{L}}_h$. In that case, the algorithm cannot infer the "correct" reward machine. For this reason, the modified QRM-episode algorithm updates the current probabilistic belief $\hat{\mathcal{L}}_j$ (line 14) in a Bayesian manner and based on the observation model of Def. 3, which is illustrated in Section 4.2. The environment probabilistic belief $\hat{\mathcal{L}}_h$ used in the policy- and reward-machine-learning is updated with the current belief $\hat{\mathcal{L}}_j$ only if the two are significantly different (lines 13-18 of Algorithm 3). More specifically, if $\hat{\mathcal{L}}_j$ is significantly changed with respect to the estimate $\hat{\mathcal{L}}_h$, the algorithm updates $\hat{\mathcal{L}}_h$ accordingly, re-initializes the Q functions, the hypothesis reward machine $\mathcal{H}_h$, and the counterexample set X. The difference among $\hat{\mathcal{L}}_h$ and $\hat{\mathcal{L}}_j$ is evaluated using a divergence test, illustrated in Section 4.3.

4.2. Information Processing

Consider the agent being at state $s_j \in S$ at some time index j. The agent will receive new perception outputs according to the observation model $\mathcal{O}(s_j, \dots)$ for all states and atomic propositions.

The agent employs the observations to update its learned model of the environment in a Bayesian approach. For ease of notation, let $\hat{\mathcal{L}}_j = \hat{\mathcal{L}}_j(s,p)$ and $\mathcal{O}(b) = \mathcal{O}(s_j, s, p, b)$. Given the prior belief of the agent $\hat{\mathcal{L}}_{j-1}$ and the received observations $\mathcal{Z}$, the posterior belief follows

$$Pr\big(s \models p | \mathcal{Z}(s_j, s, p) = \mathsf{True}\big) = \frac{\hat{\mathcal{L}}_{j-1}\mathcal{O}(\mathsf{True})}{\hat{\mathcal{L}}_{j-1}\mathcal{O}(\mathsf{True}) + (1-\hat{\mathcal{L}}_{j-1})\mathcal{O}(\mathsf{False})} \tag{1a}$$

$$Pr\big(s \models p | \mathcal{Z}(s_j, s, p) = \mathsf{False}\big) = \frac{\hat{\mathcal{L}}_{j-1}(1-\mathcal{O}(\mathsf{True}))}{\hat{\mathcal{L}}_{j-1}(1-\mathcal{O}(\mathsf{True})) + (1-\hat{\mathcal{L}}_{j-1})(1-\mathcal{O}(\mathsf{False}))} \tag{1b}$$

for all $s \in S$ and $p \in \mathcal{AP}$. Depending on the truth value observed for p, $\hat{\mathcal{L}}_j$

Algorithm 3 Joint Perception and Learning Algorithm

```
1: $\mathcal{H} \leftarrow \mathsf{Initialize}(V)$
2: $Q = \{q^v | v \in V\} \leftarrow \mathsf{Initialize}(V)$
3: $X \leftarrow \emptyset$
4: $j \leftarrow 0$
5: $\hat{\mathcal{L}}_h \leftarrow \hat{\mathcal{L}}_0$;
6: for episode $n = 1, 2, \dots$ do
7:     $(\lambda, \rho, Q, \hat{\mathcal{L}}_j) \leftarrow$ QRM_episode_mod$(\mathcal{H}, Q, \hat{\mathcal{L}}_h, \hat{\mathcal{L}}_j)$
8:     if $\mathcal{H}(\lambda) \neq \rho$ then
9:         add $(\lambda, \rho)$ to $X$
10:        $\mathcal{H} \leftarrow \mathsf{Infer}(X)$
11:        $Q \leftarrow \mathsf{Initialize}()$
12:    end if
13:    if SignifChange$(\hat{\mathcal{L}}_h, \hat{\mathcal{L}}_j)$ then
14:        $\mathcal{H} \leftarrow \mathsf{Initialize}$
15:        $Q \leftarrow \mathsf{Initialize}()$
16:        $X \leftarrow \emptyset$
17:        $\hat{\mathcal{L}}_h \leftarrow \hat{\mathcal{L}}_j$
18:    end if
19: end for
```

will be updated according to one of the aforementioned expressions. Besides, for any $P \subseteq \mathcal{AP}$, $\hat{\mathcal{L}}_j(s, P) = \prod_{p \in P} \hat{\mathcal{L}}_j(s, p)$.

4.3. Divergence Test on the Belief

If the agent's knowledge current knowledge about the environment configuration, encoded by the current belief $\hat{\mathcal{L}}_j$, has not changed significantly from its previous knowledge (encoded by $\hat{\mathcal{L}}_h$), then the agent will continue executing the algorithm using the current $\hat{\mathcal{L}}_h$. Nevertheless, if its knowledge has significantly changed, the agent will update its belief and aim to infer a new reward machine (lines 13-18 of Alg. 3). We use the Jensen-Shannon divergence to quantify the

Algorithm 4 QRM_episode_mod($\mathcal{H}$, Q, $\hat{\mathcal{L}}_h, \hat{\mathcal{L}}_j$)

```
Input: A reward machine $\mathcal{H} = (V, v_I, 2^{\mathcal{P}}, \mathbb{R}, \delta, \sigma)$, a set of $q$-functions $Q = \{q^v | v \in V\}$, two probabilistic beliefs $\hat{\mathcal{L}}_h, \hat{\mathcal{L}}_j$.
1:  $s \leftarrow$ InitialState(); $v \leftarrow v_I$, $\lambda \leftarrow []$; $\rho \leftarrow []$
2:  for $1 \leq t < eplength$ do
3:      $a \leftarrow$ GetEpsilonGreedyAction($q^v, s$)
4:      $s' \leftarrow$ ExecuteAction($s, a$)
5:      $v' \leftarrow \delta(v, \hat{L}(\hat{\mathcal{L}}_h, s'))$
6:      $r \leftarrow$ ObserveReward()
7:      $q^v(s, a) \leftarrow$ Update($r$)
8:      for $\hat{v} \in V \backslash v$ do
9:          $\hat{v}' \leftarrow \delta(\hat{v}, \hat{L}(\hat{\mathcal{L}}_h, s')$
10:         $\hat{r} \leftarrow \sigma(\hat{v}, \hat{L}(\hat{\mathcal{L}}_h, s'))$
11:         $q^{\hat{v}}(s, a) \leftarrow$ Update($\hat{r}$)
12:     end for
13:     $j \leftarrow (n-1) \cdot eplength + t$
14:     $\hat{\mathcal{L}}_j \leftarrow$ BayesUpdate($\hat{\mathcal{L}}_{j-1}, s'$)
15:     Append($\lambda, \hat{L}(\hat{\mathcal{L}}_h, s')$); Append($\rho, r$)
16:     $s \leftarrow s'$; $v \leftarrow v'$
17: end for
18: return $(\lambda, \rho, Q, \hat{\mathcal{L}}_j)$
```

change in the belief distribution between two consecutive time steps. The cumulative Jensen-Shannon divergence over the states and the propositions can be expressed as

$$\begin{aligned}
D_{\mathcal{JSD}}(\hat{\mathcal{L}}_h||\hat{\mathcal{L}}_j) = \frac{1}{2}D_{\mathcal{KL}}(\hat{\mathcal{L}}_h||\hat{\mathcal{L}}_m) + \frac{1}{2}D_{\mathcal{KL}}(\hat{\mathcal{L}}_j||\hat{\mathcal{L}}_m) = \\
\frac{1}{2}\sum_{(s,a,s')}\sum_{p\in\mathcal{AP}} \hat{\mathcal{L}}_h(s,a,s',p)\log\frac{\hat{\mathcal{L}}_h(s,a,s',p)}{\hat{\mathcal{L}}_m(s,a,s',p)} \\
+ (1-\hat{\mathcal{L}}_h(s,a,s',p))\log\frac{1-\hat{\mathcal{L}}_c(s,a,s',p)}{1-\hat{\mathcal{L}}_m(s,a,s',p)} \\
+ \frac{1}{2}\sum_{(s,a,s')}\sum_{p\in\mathcal{AP}} \hat{\mathcal{L}}_j(s,a,s',p)\log\frac{\hat{\mathcal{L}}_j(s,a,s',p)}{\hat{\mathcal{L}}_m(s,a,s',p)} \\
+ (1-\hat{\mathcal{L}}_j(s,a,s',p))\log\frac{1-\hat{\mathcal{L}}_j(s,a,s',p)}{1-\hat{\mathcal{L}}_m(s,a,s',p)}
\end{aligned}$$

where $\hat{\mathcal{L}}_m = \frac{1}{2}(\hat{\mathcal{L}}_h+\hat{\mathcal{L}}_j)$ is the average distribution. One of the input parameters to the algorithm is a threshold γ_d on the above divergence. If γ_d is not exceeded, the agent uses its previous estimate $\hat{L}(\hat{\mathcal{L}}_h,\cdot)$ to guide the learning and reward machine inference. Otherwise, it updates its estimate and reset the procedure.

4.4. Inference of Reward Machines

The goal of reward-machine inference is to find a reward machine $\mathcal{H}$ this is consistent with all the counterexamples in the sample set X, i.e., such that $\mathcal{H}(\lambda) = \rho$ for all $(\lambda, \rho) \in X$, for the current estimate $\hat{\mathcal{L}}_h$. Unfortunately, such a task is computationally hard in the sense that the corresponding decision problem "given a sample X and a natural number $k > 0$, does a consistent Mealy machine with at most k states exist?" is NP-complete [40, 43]. In this paper we follow the approach of [40], which is learning minimal consistent reward machines with the help of highly-optimized SAT solvers [44, 45, 46]. The underlying idea is to generate a sequence of formulas ϕ_k^X in propositional logic for increasing values of $k \in \mathbb{N}$ (starting with $k = 1$) that satisfy the following two properties:

- ϕ_k^X is satisfiable if and only if there exists a reward machine with k states that is consistent with X;
- a satisfying assignment of the variables in ϕ_k^X contains sufficient information to derive such a reward machine.

By increasing k by one and stopping once ϕ_k^X becomes satisfiable (or by using a binary search), an algorithm that learns a minimal reward machine that is consistent with the given sample is obtained.

As stressed in Example 3.1, however, a reward machine that encodes R on $\hat{\mathcal{L}}$ might *not* exist. In that case, ...find a tweak.

4.5. Convergence in the limit

We establish in this section the correctness of Algorithm 3. We show that, provided the improvement of the labeling function estimate $\hat{\mathcal{L}}_j$, the proposed algorithm eventually converges almost surely to a q-function that defines an optimal policy. We first define the *attainable trajectories*:

Definition 4. *Let $\mathcal{M} = (S, s_I, A, \mathcal{T}, R, \mathcal{AP}, \hat{L}, \gamma)$ be an MDP and $m \in \mathbb{N}$ a natural number. We call a trajectory $\zeta = s_0 a_0 s_1 \ldots s_k a_k s_{k+1} \in (S \times A)^* \times S$ m-attainable if (i) $k \leq m$ and (ii) $\mathcal{T}(s_i, a_i, s_{i+1}) > 0$ for each $i \in \{0, \ldots, k\}$. Moreover, we say that a trajectory ζ is attainable if there exists an $m \in \mathbb{N}$ such that ζ is m-attainable.*

Since the action choice (line 3 of Algorithm 4) follows an ϵ-greedy policy, we can show that the proposed algorithm almost surely explores every attainable trajectory in the limit (i.e., with probability 1 when the number of episodes goes to infinity).

Lemma 1. *Let $m \in \mathbb{N}$ be a natural number. Then, Algorithm 3, with eplength $\geq m$, almost surely explores every m-attainable trajectory at least once in the limit.*

Proof. The proof is identical the the one in [40, Lemma 1] and is omitted. $\square$

Similar to Definition 4, we call an observed label sequence $\hat{\lambda} = \hat{\ell}_0, \ldots, \hat{\ell}_k$ (m-)attainable on $\hat{\mathcal{L}}_h$ if there exists an (m-)attainable trajectory $s_0 a_0 s_1 \ldots s_k a_k s_{k+1}$ such that $\hat{\ell}_j = \hat{L}(\hat{\mathcal{L}}_h, s)$ for each $j \in \{0, \ldots, k\}$.

Consider now Algorithm 3 and assume that the condition of $\mathsf{SignifChange}(\hat{\mathcal{L}}_h, \hat{\mathcal{L}}_j)$ (line 13) is not satisfied after a certain number of episodes, i.e., $\hat{\mathcal{L}}_h$ is fixed. Then, Lemma 1 implies that Algorithm 3 almost surely explores every m-attainable label sequences on $\hat{\mathcal{L}}_h$ in the limit, formalized in the following corollary.

Corollary 1. *Assume that there exists a $n_r > 0$ such that* $\mathsf{SignifChange}(\hat{\mathcal{L}}_h, \hat{\mathcal{L}}_j)$ *in line 13 of Algorithm 3 is false for all episodes $n > n_r$. Then Algorithm 3, with eplength $\geq m$ and ϵ-random action policy, explores almost surely every m-attainable label sequence on $\hat{\mathcal{L}}_h$ in the limit.*

Therefore, if Algorithm 3 explores sufficiently many m-attainable label sequences on some distribution $\hat{\mathcal{L}}_h$ and for a large enough value of m, it is guaranteed to infer a reward machine that is "good enough" in the sense that it is equivalent to the reward machine encoding the reward function R on $\hat{\mathcal{L}}_h$ and on all attainable label sequences on $\hat{\mathcal{L}}_h$, assuming that such a reward machine exists (see Example 3.1). This is formalized in the next lemma.

Lemma 2. *Let $\mathcal{M}$ be a labeled MDP and assume that there exists a reward machine $\mathcal{A}$ that encodes the reward function R on $\hat{L}(\hat{\mathcal{L}}_h, \cdot)$ for some distribution $\hat{\mathcal{L}}_h$. Assume that there exists a $n_r > 0$ such that* $\mathsf{SignifChange}(\hat{\mathcal{L}}_h, \hat{\mathcal{L}}_j)$ *in line 13 of Alg. 3 is always false for all episodes $n > n_r$. Then Algorithm 3, with* $\mathsf{eplength} \geq 2^{|M|+1}(|\mathcal{A}| + 1) - 1$ *and current estimate $\hat{\mathcal{L}}_h$, almost surely learns a reward machine in the limit that is equivalent to $\mathcal{A}$ on all attainable label sequences on $\hat{\mathcal{L}}_h$.*

Proof. The proof is identical to the one in [40, Lemma 2] and is omitted. □

Following Lemma 2, Algorithm 3 will eventually learn the reward machine encoding the reward function on $\hat{\mathcal{L}}_h$, when $\hat{\mathcal{L}}_h$ is fixed. Intuitively, Lemma 2 suggests that, eventually, the algorithm is able to learn a reward machine the encodes the reward function on a *fixed* estimate $\hat{L}(\hat{\mathcal{L}}_h, \cdot)$, i.e., without the updates from the new observations. Nevertheless, the potential perception updates (line 14 of Algorithm 4) might prevent the algorithm to learn a reward machine, since $\hat{\mathcal{L}}_h$ might be changing after a finite number of episodes. However, if the observation model is accurate and the agent explores sufficiently many label sequences, intuition suggests that $\hat{\mathcal{L}}_j$ will be constantly improving, leading to less frequent updates with respect to the divergence test (line 13 of Algorithm 3). Therefore, there exists a finite number of episodes, after which $\hat{\mathcal{L}}_h$ will be fixed to some distribution $\hat{\mathcal{L}}_f$ and $L(\hat{\mathcal{L}}_f, \cdot)$ will be close to the ground-truth function L_G. By applying then Lemma 2, we conclude that Algorithm 3 will learn the reward machine that encodes the reward function on $\hat{\mathcal{L}}_f$, and consequently converge to the q-function that defines an optimal policy. This is formalized in the next theorem.

Theorem 1. *Let $\mathcal{M}$ be a labeled MDP and $\mathcal{A}$ be a reward machine that encodes the reward function R on the ground truth. Further, assume that there exists a constant $n_f > 0$ such that $\hat{L}(\hat{\mathcal{L}}_h, \cdot) = L_G(\cdot)$, for all episodes $n > n_f$ of Algorithm 3*[1]*. Then, Algorithm 3, with eplength $\geq 2^{|\mathcal{M}|+1}(|\mathcal{A}|+1)-1$ and ϵ-random action policy, converges almost surely to an optimal policy in the limit.*

Proof. Note first that the ϵ-greedy action policy and *eplength* $\geq |\mathcal{M}|$ imply that every action-pair of $\mathcal{M}$ will be visited infinitely often. Moreover, $\mathsf{SignifChange}(\hat{\mathcal{L}}_h, \hat{\mathcal{L}}_j)$ will be always false for all episodes $n > n_f$, since $\hat{\mathcal{L}}_h$ will be an accurate enough estimate. By definition, the reward machine $\mathcal{A}$ that encodes the reward function on the ground truth encodes the reward function on $\hat{\mathcal{L}}_h$ as well. Therefore, Lemma 2 guarantees that Algorithm 3 eventually learns a reward machine $\mathcal{H}$ that is equivalent to $\mathcal{A}$ on all attainable label sequences.

According to Observation 1 of [7], given the MDP $\mathcal{M} = (S, s_I, A, \mathcal{T}, R, \mathcal{AP}, \hat{L}, \gamma)$ and the reward machine $\mathcal{A} = (V, v_I, 2^{\mathcal{AP}}, \mathcal{R}, \delta, \sigma)$ one can construct an MDP $\mathcal{M}_{\mathcal{H}} = (S \times V, (s_I, v_I), A, \mathcal{T}', R', \mathcal{AP}, \hat{L}, \gamma)$ with

$$\mathcal{T}'((s,v), a, (s',v')) = \begin{cases} \mathcal{T}(s,a,s'), & \text{if } v' = \delta(v, L(s')) \\ 0 & \text{otherwise} \end{cases}$$

$$R'((s,v), a, (s',v')) = \sigma(v, L(s'))$$

and a Markovian reward function such that every attainable label sequence of $\mathcal{M}_{\mathcal{H}}$ receives the same reward as in $\mathcal{M}$. Thus, an optimal policy for $\mathcal{M}_{\mathcal{H}}$ will be also optimal for $\mathcal{M}$. Due to the ϵ-greedy action policy, the episode length being *eplength* $\geq |\mathcal{M}|$, and the fact that the updates are done in parallel for

[1]Note that this implies that $\{p \in \mathcal{AP} : \hat{L}_h(s,p) \geq 0.5\} = L_G(s)$ for all $s \in S$.

all states of the reward machine $\mathcal{H}$, every state-action pair of the MDP $\mathcal{M}_H$ will be visited infinitely often. Therefore, convergence of q-learning for $\mathcal{M}_{\mathcal{H}}$ is guaranteed [47]. Finally, since an optimal policy for $\mathcal{M}_{\mathcal{H}}$ is optimal for $\mathcal{M}$, Algorithm 3 converges to an optimal policy too.

$\square$

Theorem 1 requires that the estimate $\hat{L}(\hat{\mathcal{L}}_h, \cdot)$ improves over time and eventually reaches the ground-truth labeling function $L_G(\cdot)$, i.e., $\{p \in \mathcal{AP} : \hat{L}_h(s,p) \geq 0.5\} = L_G(s)$ for all $s \in S$. Such a property is tightly connected to the observation model $\mathcal{O}(\cdot)$ (see Definition 3) and its effect on the updates (1). It can be verified that, when there is no uncertainty in the observation model, i.e., it is sufficiently accurate or inaccurate (the values $\mathcal{O}(s_1, s_2, p, b)$ are close to one and zero depending on the value of b), the updates (1) converge to $L_G(\cdot)$. Intuitively, if the agent is aware that its observation model is either accurate or inaccurate, it adjusts its updates accordingly and based on the respective perception outputs. We note, nevertheless, that such observations models constitute only *sufficient* conditions for the correctness of Theorem 1. In the experimental results of the next section, we show the efficiency of the proposed algorithm even in cases where there is uncertainty in the observation model.

5. Experimental Results

We test the proposed algorithm in floor plan environments based on the indoor layouts collected in HouseExpo dataset [48]. HouseExpo consists of indoor layouts, built from 35,126 houses with a total of 252,550 rooms. There are 25 room categories, such as bedroom, garage, office, boiler room, etc. The dataset provides bounding boxes for the rooms with their corresponding categories, along with 2D images of layouts. We pre-processed these layouts to generate grid world-like floor plans, where every grid is labeled with the type of the room it is in (see Fig. 3). We use these labels as atomic propositions to define tasks encoded as reward machines. In particular, we are use the set of atomic propositions $\mathcal{AP} = \{\texttt{kitchen}, \texttt{bathroom}, \texttt{bedroom}, \texttt{office}, \texttt{indoor}\}$,

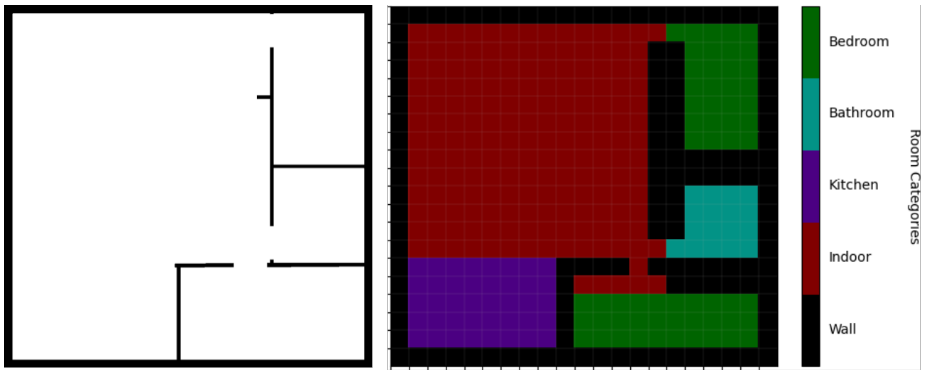


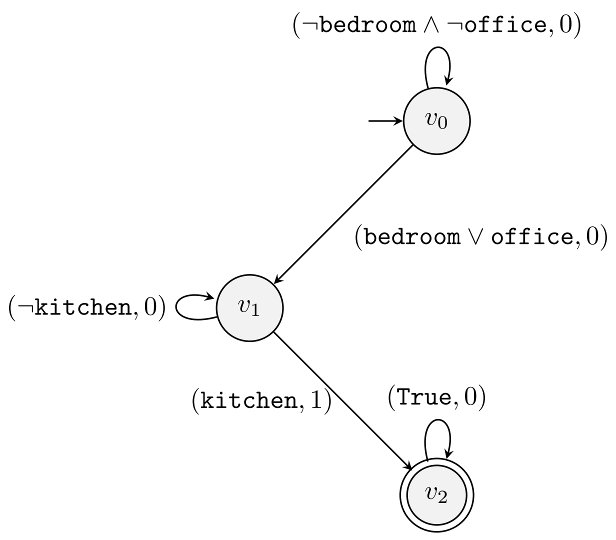
Figure 3: Left: Sample indoor layout from HouseExpo dataset; Right: Pre-processed sample layout

where `indoor` corresponds to all the categories not covered by $\mathcal{AP}$. We specify the following task to the agent: $\phi_1 =$ "Go to `bedroom` or `office`, and then to the `kitchen`", which is encoded by the reward machine shown in Fig. 4.

An episode terminates when the accepting state is reached or the maximum number of steps allowed in one episode is exceeded. The maximum number of steps per episode is 1000, whereas the maximum number of training steps is 500,000

We compare the following experimental settings regarding the observation model and the belief updates:

1. Time-varying Bayesian Observation Model (TvBNN): We use a number of training HouseExpo layouts to train a Bayesian neural network, which the algorithm use as an observation model in test layouts for the experiments. The algorithm samples observation probabilities of atomic propositions from the neural network at the beginning of every training episode.
2. Fixed Bayesian Observation Model (FiBNN): We use a number of training HouseExpo layouts to train a Bayesian neural network, which the algorithm use as an observation model in test layouts for the experiments. The algorithm samples observation probabilities of atomic propositions only once, at the beginning of training.
3. Random Observation Model: The algorithm uses an observation model



$(\neg\texttt{bedroom} \wedge \neg\texttt{office}, 0)$
v_0
$(\texttt{bedroom} \vee \texttt{office}, 0)$
$(\neg\texttt{kitchen}, 0)$
v_1
$(\texttt{kitchen}, 1)$
$(\texttt{True}, 0)$
v_2

Figure 4: Ground truth reward machine of the task ϕ_1.

that is uniformly sampled from $\mathbb{U}[0.1, 0.9]$ ...?.

4. No Belief Updates: The algorithm uses a belief function that is randomly initialized, without updating it.

Our implementation utilizes the RC2 SAT solver [49] from the PySAT library [50]. We compare four different experimental settings:[2]

6. Conclusion and Discussion

...

References

[1] D. P. Bertsekas, J. N. Tsitsiklis, Neuro-Dynamic Programming, 1st Edition, Athena Scientific, 1996.

[2]All experiments were conducted on a Lenovo laptop with 2.70-GHz Intel i7 CPU and 8-GB RAM

[2] M. E. Taylor, P. Stone, Cross-domain transfer for reinforcement learning, Proceedings of the 24th international conference on Machine learning (2007) 879–886.

[3] T. D. Kulkarni, K. Narasimhan, A. Saeedi, J. Tenenbaum, Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation, Advances in neural information processing systems 29 (2016) 3675–3683.

[4] O. Nachum, S. Gu, H. Lee, S. Levine, Data-efficient hierarchical reinforcement learning, arXiv preprint arXiv:1805.08296.

[5] D. Abel, D. Arumugam, L. Lehnert, M. Littman, State abstractions for lifelong reinforcement learning, International Conference on Machine Learning (2018) 10–19.

[6] J. Andreas, D. Klein, S. Levine, Modular multitask reinforcement learning with policy sketches, International Conference on Machine Learning (2017) 166–175.

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

[8] G. Konidaris, On the necessity of abstraction, Current opinion in behavioral sciences 29 (2019) 1–7.

[9] M. Wen, R. Ehlers, U. Topcu, Correct-by-synthesis reinforcement learning with temporal logic constraints, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2015) 4983–4990.

[10] X. Li, C.-I. Vasile, C. Belta, Reinforcement learning with temporal logic rewards, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2017) 3834–3839.

[11] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, S. A. Seshia, A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications, IEEE Conference on Decision and Control (2014) 1091–1096.

[12] D. Aksaray, A. Jones, Z. Kong, M. Schwager, C. Belta, Q-learning for robust satisfaction of signal temporal logic specifications, IEEE 55th Conference on Decision and Control (CDC) (2016) 6565–6570.

[13] M. Hasanbeig, A. Abate, D. Kroening, Logically-constrained reinforcement learning, arXiv preprint arXiv:1801.08099.

[14] R. Toro Icarte, T. Q. Klassen, R. Valenzano, S. A. McIlraith, Teaching multiple tasks to an rl agent using ltl, Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (2018) 452–461.

[15] Y. Wang, A. K. Bozkurt, M. Pajic, Reinforcement learning with temporal logic constraints for partially-observable markov decision processes, arXiv preprint arXiv:2104.01612.

[16] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, M. Pajic, Control synthesis from linear temporal logic specifications using model-free reinforcement learning, IEEE International Conference on Robotics and Automation (ICRA) (2020) 10349–10355.

[17] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, D. Wojtczak, Omega-regular objectives in model-free reinforcement learning, International Conference on Tools and Algorithms for the Construction and Analysis of Systems (2019) 395–412.

[18] S. Carr, N. Jansen, U. Topcu, Verifiable rnn-based policies for pomdps under temporal logic constraints, arXiv preprint arXiv:2002.05615.

[19] M. Ghasemi, E. Bulgur, U. Topcu, Task-oriented active perception and planning in environments with partially known semantics, International Conference on Machine Learning (2020) 3484–3493.

[20] H. Bai, D. Hsu, W. S. Lee, Integrated perception and planning in the continuous space: A pomdp approach, The International Journal of Robotics Research 33 (9) (2014) 1288–1302.

[21] R. Benenson, S. Petti, T. Fraichard, M. Parent, Integrating perception and planning for autonomous navigation of urban vehicles, 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (2006) 98–104.

[22] R. R. da Silva, V. Kurtz, H. Lin, Active perception and control from temporal logic specifications, IEEE Control Systems Letters 3 (4) (2019) 1068–1073.

[23] R. Platt, L. Kaelbling, T. Lozano-Perez, R. Tedrake, Efficient planning in non-gaussian belief spaces and its application to robot grasping, Robotics Research (2017) 253–269.

[24] A.-A. Agha-Mohammadi, S. Chakravorty, N. M. Amato, Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements, The International Journal of Robotics Research 33 (2) (2014) 268–304.

[25] X. C. D. Ding, S. L. Smith, C. Belta, D. Rus, Ltl control in uncertain environments with probabilistic satisfaction guarantees, IFAC Proceedings Volumes 44 (1) (2011) 3515–3520.

[26] J. Fu, N. Atanasov, U. Topcu, G. J. Pappas, Optimal temporal logic planning in probabilistic semantic maps, IEEE International Conference on Robotics and Automation (ICRA) (2016) 3690–3697.

[27] M. Guo, K. H. Johansson, D. V. Dimarogonas, Revising motion planning under linear temporal logic specifications in partially known workspaces,

2013 IEEE International Conference on Robotics and Automation (2013) 5025–5032.

[28] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, M. Y. Vardi, Iterative temporal planning in uncertain environments with partial satisfaction guarantees, IEEE Transactions on Robotics 32 (3) (2016) 583–599.

[29] S. C. Livingston, R. M. Murray, J. W. Burdick, Backtracking temporal logic synthesis for uncertain environments, IEEE International Conference on Robotics and Automation (2012) 5163–5170.

[30] F. J. Montana, J. Liu, T. J. Dodd, Sampling-based reactive motion planning with temporal logic constraints and imperfect state information, Critical Systems: Formal Methods and Automated Verification (2017) 134–149.

[31] X. Ding, S. L. Smith, C. Belta, D. Rus, Optimal control of markov decision processes with linear temporal logic constraints, IEEE Transactions on Automatic Control 59 (5) (2014) 1244–1257.

[32] A. M. Ayala, S. B. Andersson, C. Belta, Temporal logic motion planning in unknown environments, IEEE/RSJ International Conference on Intelligent Robots and Systems (2013) 5279–5284.

[33] A. Camacho, O. Chen, S. Sanner, S. A. McIlraith, Non-markovian rewards expressed in ltl: guiding search via reward shaping, Tenth Annual Symposium on Combinatorial Search.

[34] K. Chatterjee, M. Chmelik, R. Gupta, A. Kanodia, Qualitative analysis of pomdps with temporal logic specifications for robotics applications, IEEE International Conference on Robotics and Automation (ICRA) (2015) 325–330.

[35] R. Sharan, J. Burdick, Finite state control of pomdps with ltl specifications, American Control Conference (2014) 501–508.

[36] M. Ahmadi, A. Singletary, J. W. Burdick, A. D. Ames, Barrier functions for multiagent-pomdps with dtl specifications, 59th IEEE Conference on Decision and Control (CDC) (2020) 1380–1385.

[37] B. Ramasubramanian, A. Clark, L. Bushnell, R. Poovendran, Secure control under partial observability with temporal logic constraints, American Control Conference (ACC) (2019) 1181–1188.

[38] X. Zhang, B. Wu, H. Lin, Learning based supervisor synthesis of pomdp for pctl specifications, 54th IEEE Conference on Decision and Control (CDC) (2015) 7470–7475.

[39] R. Toro Icarte, E. Waldie, T. Klassen, R. Valenzano, M. Castro, S. McIlraith, Learning reward machines for partially observable reinforcement learning, Advances in Neural Information Processing Systems 32 (2019) 15523–15534.

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

[41] R. Bellman, A markovian decision process, Journal of mathematics and mechanics (1957) 679–684.

[42] M. L. Puterman, Markov decision processes: discrete stochastic dynamic programming, John Wiley & Sons, 2014.

[43] E. M. Gold, Complexity of automaton identification from given data, Information and control 37 (3) (1978) 302–320.

[44] D. Neider, N. Jansen, Regular model checking using solver technologies and automata learning, NASA Formal Methods Symposium (2013) 16–31.

[45] M. J. Heule, S. Verwer, Exact dfa identification using sat solvers, International Colloquium on Grammatical Inference (2010) 66–79.

[46] D. Neider, Applications of automata learning in verification and synthesis, Ph.D. thesis, Hochschulbibliothek der Rheinisch-Westfälischen Technischen Hochschule Aachen (2014).

[47] C. J. Watkins, P. Dayan, Q-learning, Machine learning 8 (3-4) (1992) 279–292.

[48] L. Tingguang, H. Danny, L. Chenming, Z. Delong, W. Chaoqun, M. Q.-H. Meng, Houseexpo: A large-scale 2d indoor layout dataset for learning-based algorithms on mobile robots, arXiv preprint arXiv:1903.09845.

[49] A. Morgado, C. Dodaro, J. Marques-Silva, Core-guided maxsat with soft cardinality constraints (2014) 564–573.

[50] A. Ignatiev, A. Morgado, J. Marques-Silva, PySAT: A Python toolkit for prototyping with SAT oracles (2018) 428–437`doi:10.1007/978-3-319-94144-8_26`.
URL `https://doi.org/10.1007/978-3-319-94144-8_26`