Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning
Reasoning, Learning & Decision-Making Track

Learning Robust Reward Machines from Noisy Labels

Roko Parać[1] , **Lorenzo Nodari**[2] , **Leo Ardon**[1] , **Daniel Furelos-Blanco**[1] ,
Federico Cerutti[2,3] , **Alessandra Russo**[1]
[1]Imperial College London
[2]University of Brescia
[3]Cardiff University
roko.parac18@ic.ac.uk, lorenzo.nodari@unibs.it, leo.ardon19@ic.ac.uk, d.furelos-blanco18@ic.ac.uk,
federico.cerutti@unibs.it, a.russo@ic.ac.uk

Abstract

This paper presents PROB-IRM, an approach that learns robust reward machines (RMs) for reinforcement learning (RL) agents from noisy execution traces. The key aspect of RM-driven RL is the exploitation of a finite-state machine that decomposes the agent's task into different subtasks. PROB-IRM uses a state-of-the-art inductive logic programming framework robust to noisy examples to learn RMs from noisy traces using the Bayesian posterior degree of beliefs, thus ensuring robustness against inconsistencies. Pivotal for the results is the interleaving between RM learning and policy learning: a new RM is learned whenever the RL agent generates a trace that is believed not to be accepted by the current RM. To speed up the training of the RL agent, PROB-IRM employs a probabilistic formulation of reward shaping that uses the posterior Bayesian beliefs derived from the traces. Our experimental analysis shows that PROB-IRM can learn (potentially imperfect) RMs from noisy traces and exploit them to train an RL agent to solve its tasks successfully. Despite the complexity of learning the RM from noisy traces, agents trained with PROB-IRM perform comparably to agents provided with handcrafted RMs.

1 Introduction

Reinforcement learning (RL; Sutton and Barto 2018) is a machine learning paradigm where agents learn to solve a task by interacting with an environment to maximize their cumulative reward. Significant advancements have demonstrated its potential to perform tasks as well as, if not better than, humans. A famous example is AlphaGo (Silver et al. 2016), the first AI system to beat a Go world champion. In addition, RL plays a key role in the training of Large Language Models, such as ChatGPT (OpenAI 2023).

Many open challenges still need to be addressed to make RL more widely applicable to real-world problems. Among these are the ability to generalise and transfer across tasks, operate robustly in the presence of partial observability and noise, and make the learned policies interpretable (Dulac-Arnold et al. 2021).

Reward machines (RMs; Toro Icarte et al. 2018) are a recent mechanism for addressing some of these challenges. RMs are finite-state machines representing non-Markovian reward functions in terms of high-level propositional events. The RM structures the agent's task into sequences of intermediate abstract states that act as an external memory for the agent. This makes the reward Markovian, thus enabling the application of standard RL algorithms in non-Markovian reward settings. The RM structure facilitates task decomposition, making policy learning more efficient when rewards are sparse. Recent work has extended the applicability of RMs by learning them instead of handcrafting them (Toro Icarte et al. 2019; Xu et al. 2020; Furelos-Blanco et al. 2021; Hasanbeig et al. 2021), and hierarchically composing them, for reusability (Furelos-Blanco et al. 2023). However, RM learning approaches assume a *perfect labelling function*, a construct that enables agents to accurately observe the high-level events occurring in the environment. This assumption is *unrealistic*; for instance, robot sensors, which resemble the role of the labelling function, are seldom perfect, due to environmental conditions, limitations in technology, and inherent inaccuracies.

We propose PROB-IRM (**Prob**abilisitic **I**nduction of **R**eward **M**achines), a method for learning and exploiting RMs from noisy propositions perceived by an RL agent through a *noisy labelling function*. PROB-IRM uses ILASP (Law, Russo, and Broda 2015), a state-of-the-art inductive logic programming system capable of learning from noisy examples. The learned RM is exploited by an RL algorithm that leverages the RM structure using a novel probabilistic reward-shaping mechanism based on an RM state belief. The PROB-IRM algorithm interleaves the RL and RM learning processes, enabling the agent to immediately exploit the newly learned (possibly sub-optimal) RMs. A new RM is learned during the interleaving process when the currently used one does not recognise noisy traces.

We evaluate PROB-IRM in several existing grid-world problems with sparse non-Markovian rewards. We show PROB-IRM learns RMs from noisy traces that are exploited by an RL agent. Our results demonstrate that under a wide array of noise configurations, PROB-IRM performs similarly to approaches where RMs are handcrafted.

The paper is organised as follows. Section 2 introduces the background of our work. Section 3 describes PROB-IRM, including the problem formalization and the learning and exploitation of RMs from noisy traces. Section 4 presents our experimental results, Section 5 discusses related work, and Section 6 concludes the paper with suggestions for future directions.

2 Background

In this section, we introduce the basic notions and terminology. Given a finite set $\mathcal{X}$, $\Delta(\mathcal{X})$ is the probability simplex over $\mathcal{X}$, $\mathcal{X}^*$ denotes (possibly empty) sequences of elements from $\mathcal{X}$, $\mathcal{X}^+$ denotes non-empty sequences of elements from $\mathcal{X}$, and $2^{\mathcal{X}}$ is the power set of $\mathcal{X}$.

2.1 Reinforcement Learning

We formalize RL tasks as *labelled Markov decision processes* (MDPs; Fu and Topcu 2014; Furelos-Blanco et al. 2023). A MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, p, r, \tau, \gamma, \mathcal{P}, \mathcal{L} \rangle$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is a probability transition function, $r : (\mathcal{S} \times \mathcal{A})^+ \times \mathcal{S} \to \mathbb{R}$ is a reward function, $\tau : (\mathcal{S} \times \mathcal{A}) \times \mathcal{S}^* \to \{\bot, \top\} \times \{\bot, \top\}$ is a termination function, $\gamma \in [0, 1)$ is a discount factor, $\mathcal{P}$ is a finite set of *propositions* representing high-level events, and $\mathcal{L} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to 2^{\mathcal{P}}$ is a *(perfect) labelling function* mapping state-action-state triplets into sets of propositions. We refer to these sets as *labels*. The transition function p is Markovian, whereas the reward function r and the termination function τ are not (i.e., they are history-dependent).

Given a state-action *history* $h_t = \langle s_0, a_0, \ldots, s_t \rangle \in (\mathcal{S} \times \mathcal{A})^* \times \mathcal{S}$, a *trace* $\lambda_t = \langle \mathcal{L}(\emptyset, \emptyset, s_0), \ldots, \mathcal{L}(s_{t-1}, a_{t-1}, s_t) \rangle \in (2^{\mathcal{P}})^+$ assigns a label to all triplets in h_t. The goal is to find a *policy* $\pi : (2^{\mathcal{P}})^+ \times \mathcal{S} \to \Delta(\mathcal{A})$ that maps traces-states to a probability distribution over actions that maximizes the expected cumulative discounted reward (or *return*) $R_t = \mathbb{E}_\pi[\sum_{k=t}^{n} \gamma^{k-t} r(h_t)]$, where n is the episode's last step. Traces must be faithful representations of history to find such a policy, i.e., reward and termination functions could depend on traces instead of history.

The agent-environment interaction is as follows. At time t, the (label) trace is $\lambda_t \in (2^{\mathcal{P}})^+$ and the agent observes a tuple $\langle s_t, s_t^T, s_t^G \rangle$, where $s_t \in \mathcal{S}$ is the state, $s_t^T \in \{\bot, \top\}$ indicates whether the history is terminal, and $s_t^G \in \{\bot, \top\}$ indicates whether the history accomplishes the task's goal. Both s_t^T and s_t^G are determined by the termination function τ. The agent also observes a label $L_t = \mathcal{L}(s_{t-1}, a_{t-1}, s_t)$. If the history is non-terminal, the agent runs an action $a_t \in \mathcal{A}$, and the environment transitions to state $s_{t+1} \sim p(\cdot \mid s_t, a_t)$. The agent then observes a new tuple $\langle s_{t+1}, s_{t+1}^T, s_{t+1}^G \rangle$ and label L_{t+1}, extends the trace as $\lambda_{t+1} = \lambda_t \oplus L_{t+1}$, and receives reward r_{t+1}. A trace λ_t is a *goal trace* if $\langle s_t^T, s_t^G \rangle = \langle \top, \top \rangle$, a *dead-end trace* if $\langle s_t^T, s_t^G \rangle = \langle \top, \bot \rangle$, and an *incomplete trace* if $s_t^T = \bot$.

Example 1. *The* OFFICEWORLD *(Toro Icarte et al. 2018), illustrated in Figure 1 (left), is a* 12×9 *grid labeled with some special locations. At each step, the agent* 🧍 *observes its current position in the grid and moves in one of the four cardinal directions; that is,* $\mathcal{S} = \{0, \ldots, 11\} \times \{0, \ldots, 8\}$ *and* $\mathcal{A} = \{\text{up}, \text{down}, \text{left}, \text{right}\}$. *The agent always moves in the intended direction and stays put if it moves towards a wall. The set of propositions* $\mathcal{P} = \{☕, ✉, o, A, B, C, D, *\}$ *is constituted of the special locations. The labelling function maps a* $\langle s, a, s' \rangle \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ *triplet to the set of propositions observed in* s'*, e.g.* $\mathcal{L}((4, 6), \text{left}, (3, 6)) = \{☕\}$. *In this paper, we consider different tasks that consist of visiting a sequence of special locations while avoiding the decorations* $(*)$*:*

- COFFEE*: go to the coffee machine (☕) then go to the office (o).*
- COFFEEMAIL*: go to the coffee machine (☕) and the mail location (✉), in any order, then go to the office (o).*
- VISITABCD*: go to locations A, B, C and D in order.*

A reward of 1 is obtained for completing the task; otherwise, the reward is 0. Rewards are non-Markovian since the current state (i.e., position on the grid) alone cannot determine the reward. The history $h = \langle \langle 4, 6 \rangle, \text{left}, \langle 3, 6 \rangle, \text{right}, \langle 4, 6 \rangle, \text{down}, \langle 4, 5 \rangle, \text{down}, \langle 4, 4 \rangle \rangle$ *yields a reward of 1 and is mapped to the goal trace* $\lambda = \langle \{\}, \{☕\}, \{\}, \{\}, \{o\} \rangle$.

Learning policies over histories or traces is impractical since they can grow arbitrarily. In this paper, we employ *reward machines* to compactly encode traces, enabling efficient policy learning.

2.2 Reward Machines

A *reward machine* (RM; Toro Icarte et al. 2018; 2022) is a finite-state machine representation of a reward function. Formally, an RM is a tuple $M = \langle U, \mathcal{P}, \delta_u, \delta_r, u_0, u_A, u_R \rangle$, where U is a set of states, $\mathcal{P}$ is a set of propositions constituting the RM's alphabet, $\delta_u : U \times 2^{\mathcal{P}} \to U$ is a state-transition function, $\delta_r : U \times U \to \mathbb{R}$ is a reward-transition function, $u_0 \in U$ is the initial state, $u_A \in U$ is the accepting state, and $u_R \in U$ is the rejecting state.

Example 2. *Figure 1 (right) illustrates the RM for the* OFFICEWORLD*'s* COFFEE*. The edges are labelled by propositional logic formulas over* $\mathcal{P} = \{☕, ✉, o, A, B, C, D, *\}$ *and rewards for transitioning between states. To verify whether a formula is satisfied by a label* $L \in 2^{\mathcal{P}}$*,* L *is used as a truth assignment: propositions contained in the label are true, and false otherwise. For example,* $\{☕\} \models ☕ \wedge \neg o \wedge \neg *$.

Reward machines are revealed to the agent during agent-environment interactions. Starting from the RM's initial state, the agent moves in the RM according to the state-transition function and obtains rewards through the reward-transition function. Given an RM M and a trace $\lambda = \langle L_0, \ldots, L_n \rangle$, a *traversal* $M(\lambda) = \langle v_0, v_1, \ldots, v_{n+1} \rangle$ is a unique sequence of RM states such that (i) $v_0 = u_0$, and (ii) $\delta_u(v_i, L_i) = v_{i+1}$ for $i = 0, \ldots, n$. Traversals for goal and dead-end traces should terminate in the accepting and rejecting states, respectively; in contrast, traversals for incomplete traces should terminate somewhere different from the accepting and rejecting states.

Example 3. *Given the RM in Figure 1 (right) and the goal trace* $\lambda = \langle \{\}, \{☕\}, \{\}, \{\}, \{o\} \rangle$*, the traversal is* $\langle u_0, u_0, u_1, u_1, u_1, u_A \rangle$*. As expected, the traversal ends with the accepting state.*

Reward machines constitute compact trace representations: each RM state encodes a different completion degree of the task. Consequently, rewards become Markovian when defined over $\mathcal{S} \times U$. In line with this observation, Toro Icarte et al. (2022) propose an algorithm that learns an

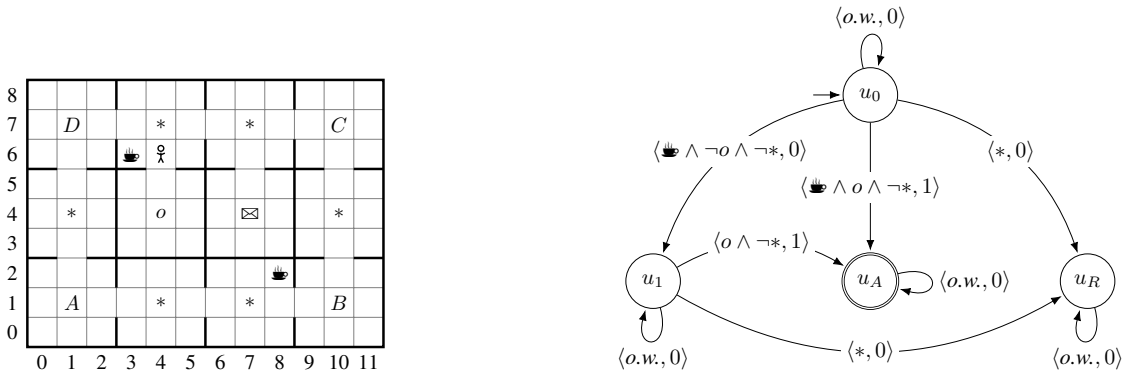

Figure 1: An OFFICEWORLD instance (left) and a reward machine for the COFFEE task (right), where *o.w.* stands for *otherwise*.

action-value function (or Q-function)—an estimation of the expected return following the execution of an action from a given state—over $\mathcal{S} \times U$. Given a transition from state s to s' with an action a and its label $L = \mathcal{L}(s, a, s')$, the Q-function $q : \mathcal{S} \times U \times \mathcal{A} \to \mathbb{R}$ is updated as follows:

$$q(s, u, a) \xleftarrow{\alpha} \delta_r(u, u') + \gamma \max_{a' \in \mathcal{A}} q(s', u', a'), \qquad (1)$$

where $u' = \delta_u(u, L)$, and $x \xleftarrow{\alpha} y$ is shorthand for $x \leftarrow x + \alpha(y - x)$. In the tabular case, where estimates are stored for each state-action pair, the algorithm converges to an optimal policy in the limit (Toro Icarte et al. 2022, Theorem 4.1).

2.3 Learning from Noisy Examples

Previous work shows that RMs can be learned from traces during the RL agent's training (Toro Icarte et al. 2019; Xu et al. 2020; Furelos-Blanco et al. 2021; Hasanbeig et al. 2021). We adopt a methodology similar to that by Furelos-Blanco et al. (2020; 2021), who used a state-of-the-art inductive logic programming system to induce RMs represented as answer set programs (ASP; Gelfond and Kahl 2014) that represent the RMs. In what follows, we describe the fundamentals of the learning system we use. We refer the reader to the work by Law (2018) for details.

Learning from Answer Sets (LAS; Law, Russo, and Broda 2020) is a paradigm for learning answer set programs. A LAS task is a tuple $\langle B, S_M, E \rangle$ where B is an ASP program called background knowledge, S_M is a set of rules known as the hypothesis space, and E is a set of examples. The hypothesis space is specified through a set of mode declarations M, describing which predicates can appear in the head or body of a rule.

ILASP (Law, Russo, and Broda 2015) is a state-of-the-art system for solving LAS tasks robust to *noisy examples* (Law, Russo, and Broda 2018). Noisy examples in ILASP are referred to as *weighted context-dependent partial interpretations* (WCDPIs), each a tuple $\langle e_{id}, e_{pen}, \langle e^{inc}, e^{exc} \rangle, e_{ctx} \rangle$, where e_{id} is an identifier, e_{pen} is a penalty denoting the level of certainty of an example which can be either a positive integer or infinite, $\langle e^{inc}, e^{exc} \rangle$ is a pair of atom sets known as *partial interpretation*, and e_{ctx} is an ASP program known as *context* that expresses example-specific information. A hypothesis $H \subseteq S_M$ covers (or *accepts*) a WCDPI if there exists an answer set of $B \cup e_{ctx} \cup H$ that contains every atom in e^{inc} and no atom in e^{exc}.[1] Informally, the *cost* of a hypothesis H is the sum of the hypothesis length plus the penalties of the examples not accepted by H. Examples with an infinite penalty are not noisy and must be covered by the induced hypothesis. The *goal* of a LAS task with noisy examples is to find a hypothesis that minimises the cost over a given hypothesis space for a given set of WCDPIs. This is formally defined as follows.

Definition 1. *A LAS task T is a tuple of the form $\langle B, S_M, E \rangle$, where B is an ASP program, S_M is a hypothesis space and E is a set of WCDPIs. Given a hypothesis $H \subseteq S_M$:*

1. *$uncov(H, T)$ is the set of all examples $e \in E$ such that H does not accept e.*
2. *the penalty of H, denoted as $pen(H, T)$, is the sum $\sum_{e \in uncov(H,T)} e_{pen}$.*
3. *the length of H, denoted as $length(H)$, is the sum $\sum_{r \in H} length(r)$, where $length(r) \in \mathbb{N}_{>0}$.*
4. *the score of H, denoted as $\mathcal{S}(H, T)$, is the sum $length(H) + pen(H, T)$.*
5. *H is an inductive solution of T iff $\mathcal{S}(H, T)$ is finite.*
6. *H is an* optimal inductive solution *of T iff $\mathcal{S}(H, T)$ is finite and $\nexists H' \subseteq S_M$ such that $\mathcal{S}(H, T) > \mathcal{S}(H', T)$.*

The length of a hypothesis H is often defined as the number of literals that appear in H, thus considering $length(r) = |r|$. However, this aspect can be customised to the specific task in hand, particularly when the task does not require the shortest hypotheses to be learned.

3 Methodology

In this section, we present PROB-IRM, an approach for interleaving the learning of RMs from noisy traces with the learning of policies to solve a given task. We consider episodic labelled MDP tasks where the reward is 1 if the agent observes a goal trace and 0 otherwise. The noisy traces

[1] We later show a WCDPI construction for our method (see Example 4).

are caused by a *noisy labelling function*, which plays the role of an imperfect set of sensors. To cater for this uncertainty, RM traversals must account for the agent's *belief* of being at a certain RM state.

3.1 Noisy Traces

We assume the agent has a binary sensor for each proposition $l \in \mathcal{P}$ with a given sensitivity and specificity of detection. We denote with $l = \top$ (resp. $l = \bot$) the *actual* occurrence (resp. absence) of l in the environment. We denote with $\tilde{l} = \top$ (resp. $\tilde{l} = \bot$) the detection (resp. non-detection) of l with the agent's sensors. The *sensitivity* of the sensor is the probability of the sensor detecting a proposition given that it occurred; formally, $P(\tilde{l} = \top \mid l = \top)$. The *specificity* of the sensor is the probability of not detecting a proposition given that it did not occur; formally, $P(\tilde{l} = \bot \mid l = \bot)$.

As the sensor's prediction may differ from the actual occurrence of a proposition, the agent's belief of the (non)occurrence of a proposition is a posterior probability conditional to its sensor's value. We define the *posterior* probability of a proposition l using Bayes' rule:

$$P(l = \top \mid \tilde{l} = y) = \frac{P(\tilde{l} = y \mid l = \top)P(l = \top)}{\sum_{x \in \{\top, \bot\}} P(\tilde{l} = y \mid l = x)P(l = x)}$$

where:

- $y \in \{\top, \bot\}$ is the sensor detection outcome.
- $P(\tilde{l} = \top \mid l = \bot) = 1 - P(\tilde{l} = \bot \mid l = \bot)$, and represents the probability of an unexpected detection. Similarly, $P(\tilde{l} = \bot \mid l = \top) = 1 - P(\tilde{l} = \top \mid l = \top)$.
- $P(l = \top)$ is the prior probability of l occurring, and $P(l = \bot) = 1 - P(l = \top)$. We refer the reader to Section 4 for further details on defining the prior.

The labelling function must report the sensors' degree of uncertainty on the detected propositions. We introduce the notion of *noisy labelling function*, which defines (for a given transition) the probability of every proposition conditioned to detecting their respective sensor readings. First, we define the probability of a proposition at a given transition. Given a transition triplet (s_t, a_t, s_{t+1}), we denote with $\tilde{\mathcal{P}}_{t+1} \in 2^{\mathcal{P}}$ all sensors' detections (i.e. all detected propositions) at that transition. We assume that the probability of a proposition l at a given transition is conditional only to its related sensor detection at that transition ($\tilde{l}_{t+1}$). The value $\tilde{l}_{t+1}$ is $\top$ if the proposition $l \in \tilde{\mathcal{P}}_{t+1}$ (detected); otherwise $\tilde{l}_{t+1}$ is $\bot$. So we define $P(l = \top \mid s_t, a_t, s_{t+1}) = P(l = \top \mid \tilde{\mathcal{P}}_{t+1}) = P(l = \top \mid \tilde{l}_{t+1})$.

The noisy labelling function defines the probability of each proposition at a given transition.

Definition 2 (Noisy labelling function)**.** *Let $\mathcal{S}$ be the set of states, $\mathcal{A}$ the set of actions, and $\mathcal{P}$ be the set of propositions. Given a transition $(s_t, a_t, s_{t+1}) \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, the noisy labelling function $\widetilde{\mathcal{L}}$ maps the transition to the set of all possible propositions with their respective probabilities at that transition. Formally, $\widetilde{\mathcal{L}}(s_t, a_t, s_{t+1}) = \{(l, P(l = \top \mid s_t, a_t, s_{t+1})) \mid l \in \mathcal{P}\}$.*

We can then define the probability of a set of propositions at a given transition in terms of the noisy labelling function. Propositions are assumed to be conditionally independent. We use $[\widetilde{\mathcal{L}}(s_t, a_t, s_{t+1})]_l$ to denote the probability of a proposition l at the transition (s_t, a_t, s_{t+1}) given by our noisy labelling function. So, given a label $L \in 2^{\mathcal{P}}$, we have:

$$P(L|s_t, a_t, s_{t+1}) = \prod_{l \in \mathcal{P}} p_l, \tag{2}$$

where

$$p_l = \begin{cases} 1 - [\widetilde{\mathcal{L}}(s_t, a_t, s_{t+1})]_l & \text{if } l \notin L; \\ [\widetilde{\mathcal{L}}(s_t, a_t, s_{t+1})]_l & \text{if } l \in L. \end{cases}$$

Definition 3 (Noisy trace)**.** *Given a state-action history $h_t = \langle s_0, a_0, s_1, a_1, ..., s_t \rangle$, the noisy trace $\tilde{\lambda}_t$ is given by:*

$$\tilde{\lambda}_t = \langle \widetilde{\mathcal{L}}(s_0, a_0, s_1), \widetilde{\mathcal{L}}(s_1, a_1, s_2), ..., \widetilde{\mathcal{L}}(s_{t-1}, a_{t-1}, s_t) \rangle.$$

Our RL task can be formalised as a *noisy labelled MDP*; that is, a labelled MDP (see Section 2.1) but with a noisy labelling function. Despite the labelling function now being noisy, we *assume* the termination function τ remains deterministic; hence, the termination of a noisy trace is determined with certainty throughout the agent-environment interaction.

3.2 Learning RMs from Noisy Traces

We learn candidate RMs using ILASP. Recall that a LAS task is a tuple $\langle B, S_M, E \rangle$, where B is the background knowledge, S_M is the hypothesis space, and E is a set of WCDPI examples (see Section 2.3). The background knowledge B and the hypothesis space S_M are similar to the ones proposed by Furelos-Blanco et al. (2021). The former is a set of ASP rules that describes the general behaviour of any RM (i.e., how an RM is traversed). The latter contains all possible rules that can constitute the state-transition function of the RM.

We now focus on representing the WCDPI examples in E from a given set of noisy traces.

Generating Examples from Noisy Traces. One of the key aspects of learning RMs from noisy traces is how to map these traces into WCDPIs. A direct approach involves aggregating the probabilities generated by the noisy labelling function, for a given noisy trace, into a single trace-level probability. This aggregated probability can then be used as the weight for the WCDPI generated from that trace. Approaches along this line have been proposed in the literature for other domains (Cunnington et al. 2023). They define the aggregation function as a t-norm over the collection of probabilistic predictions. This value is used as the *penalty* for the examples, while the predictions are converted into their most likely outcome and stored within the WCDPI *context*. Such a solution proved too restrictive since such WCDPIs could only represent the most likely trace. Instead, we adopt a *sampling-based method*: the probabilities in a noisy trace are used to define proposition-specific Bernoulli distributions, which are then sampled to determine the propositions that would be part of the context of the associated WCDPI.

Let $\tilde{\lambda}$ be a noisy trace. We denote with $\tilde{\lambda}_{i,l} = [\widetilde{\mathcal{L}}(s_{i-1}, a_{i-1}, s_i)]_l$ the probability of proposition l occurring at step i of $\tilde{\lambda}$. We generate a *sample trace* λ' from $\tilde{\lambda}$ by determining the occurrence of each proposition $l \in \mathcal{P}$ at each step i using the corresponding Bernoulli distribution; that is, $\lambda'_{i,l} \sim Ber(\tilde{\lambda}_{i,l})$. For each sampled trace λ', an *ASP trace* $\lambda^{ASP} = \{\texttt{prop}(l, i) \mid \lambda'_{i,l} = 1\}$ is built by mapping each occurring proposition into a $\texttt{prop}(l, i)$ fact indicating that proposition l occurs at time i. As proposed by Furelos-Blanco et al. (2021), we *compress* sampled traces by removing consecutive occurrences of the same sampled proposition set.

The WCDPI example generated from $\tilde{\lambda}$ is given by $\langle e_{id}, e_{pen}, \langle e^{inc}, e^{exc}\rangle, e_{ctx}\rangle$, where e_{id} is a unique example identifier, and the penalty is $e_{pen} = 1$. The partial interpretation $\langle e^{inc}, e^{exc}\rangle$ is $\langle\{\texttt{accept}\}, \{\texttt{reject}\}\rangle$ for goal traces, $\langle\{\texttt{reject}\}, \{\texttt{accept}\}\rangle$ for dead-end traces, and $\langle\emptyset, \{\texttt{accept}, \texttt{reject}\}\rangle$ for incomplete traces. The atom $\texttt{accept}$ (resp. $\texttt{reject}$) indicates that the accepting (resp. rejecting) state of the RM is reached; therefore, for instance, the partial interpretation for goal traces indicates that the accepting state must be reached, whereas the rejecting state must not. Finally, the context e_{ctx} is given by the ASP representation λ^{ASP} of the sample trace λ'.

Example 4 (Generation of WCDPI from a noisy trace)**.** *Let us consider the proposition set* $\mathcal{P} = \{☕, o\}$ *and the noisy goal trace* $\tilde{\lambda} = \langle\{☕ : 0.01, o : 0.01\}, \{☕ : 0.9, o : 0.01\}, \{☕ : 0.9, o : 0.01\}, \{☕ : 0.01, o : 0.01\}, \{☕ : 0.01, o : 0.9\}\rangle$*. A WCDPI generated from* $\tilde{\lambda}$ *will be* $\langle id, 1, \langle\{\texttt{accept}\}, \{\texttt{reject}\}\rangle, e_{ctx}\rangle$*, where* e_{ctx} *is constructed as follows:*

1. *Produce a sample trace, e.g.* $\lambda' = \langle\{☕ : 0, o : 0\}, \{☕ : 1, o : 0\}, \{☕ : 1, o : 0\}, \{☕ : 0, o : 0\}, \{☕ : 0, o : 1\}\rangle$.
2. *Compress the trace:* $\lambda' = \langle\{☕ : 0, o : 0\}, \{☕ : 1, o : 0\}, \{☕ : 0, o : 0\}, \{☕ : 0, o : 1\}\rangle$.
3. *Construct the ASP representation* $\lambda^{ASP} = \{\texttt{prop}(☕, 1), \texttt{prop}(o, 3)\}$ *of the compressed sample trace* λ'.

Once all the WCDPI for all the noisy traces are sampled, we make the following adjustments to their penalties. First, we reweigh the penalties to balance the classes (goal, dead-end, incomplete). Second, since the sampling process may produce x identical WCDPIs $\langle e_{id_i}, e_{pen}, \langle e^{inc}, e^{exc}\rangle, e_{ctx}\rangle$, we replace them (without loss of generality) with one WCDPI of the form $\langle e_{id_{new}}, x \cdot e_{pen}, \langle e^{inc}, e^{exc}\rangle, e_{ctx}\rangle$.

To ensure the RM is well-formed and to make the RM learning more efficient, we enforce the determinism and the symmetry-breaking constraints proposed by Furelos-Blanco et al. (2021).

3.3 Exploitation of Reward Machines

In this section, we describe how an RM is exploited to learn policies.

Reward Machine State Belief. Because of the noisy labelling function, the current RM state cannot be known: we can only determine the *RM state belief* (Li et al. 2022; 2024).

Definition 4 (RM state belief $\tilde{u}_t$)**.** *The RM state belief* $\tilde{u}_t \in \Delta(U)$ *is a categorical probability distribution expressing the RL agent's belief of being in an RM state* u *at timestep* t*. Formally,*

$$\tilde{u}_0(u) = \begin{cases} 1 & \text{if } u = u_0; \\ 0 & \text{otherwise}, \end{cases}$$

$$\tilde{u}_{t+1}(u) = \sum_{\substack{u_t \in U, \\ L_t \in 2^{\mathcal{P}}}} P(L_t | s_t, a_t, s_{t+1}) \tilde{u}_t(u_t) \mathbb{1}[\delta_u(u_t, L_t) = u],$$

where δ_u *is the RM state-transition function, and* $\mathbb{1}$ *is the indicator function.*

Probabilistic Reward Shaping. Reward shaping aims to provide additional rewards to guide the agent towards completing a task. Previous works use the *potential-based reward shaping* (Ng, Harada, and Russell 1999), which generates intermediate rewards from the difference in values of a *potential function* $\Phi(s)$ over consecutive MDP states. Under this formulation, reward shaping does not shrink the set of optimal policies.

In the context of RM-based RL, Camacho et al. (2019) and Furelos-Blanco et al. (2021) define the potential function $\Phi : U \to \mathbb{R}$ in terms of RM states. Formally,

$$r_s(u, u') = \gamma\Phi(u') - \Phi(u),$$

where γ is the MDP's discount factor.

Given that the agent has access to the belief vector $\tilde{u}_t \in \Delta(U)$, we propose the *potential function on RM state beliefs* $\tilde{\Phi} : \Delta(U) \to \mathbb{R}$, which is defined as the sum of every plausible RM state's potential weighted by its belief. Formally,

$$\tilde{\Phi}(\tilde{u}_t) = \sum_{u \in U} \tilde{u}_t(u)\Phi(u),$$

where $\Phi : U \to \mathbb{R}$ is a potential function on RM states. The resulting reward-shaping function can thus be expressed as:

$$\begin{aligned} r_s(\tilde{u}_t, \tilde{u}_{t+1}) &= \gamma\tilde{\Phi}(\tilde{u}_{t+1}) - \tilde{\Phi}(\tilde{u}_t) \\ &= \sum_{u \in U} \gamma\tilde{u}_{t+1}(u)\Phi(u) - \tilde{u}_t(u)\Phi(u) \\ &= \sum_{u \in U} (\gamma\tilde{u}_{t+1}(u) - \tilde{u}_t(u))\Phi(u). \end{aligned}$$

Eck et al. (2013) introduced a similar formulation in the context of state beliefs in partially observable MDPs.

Akin to Furelos-Blanco et al. (2021), we define the potential function on RM states following the intuition that the agent should be rewarded for getting closer to u_A. Formally,

$$\Phi(u) = |U| - d_{\min}(u, u_A),$$

where $d_{\min}(u, u_A)$ is the minimum distance between u and u_A. If u_A is unreachable from u, then $d_{\min}(u, u_A) = \infty$.

Example 5 (Reward shaping in the OFFICEWORLD's COFFEE task)**.** *From the reward machine in Figure 1, we obtain the following potential function* Φ*:*

$$\Phi(u_0) = 4 - 1 = 3, \qquad \Phi(u_1) = 4 - 1 = 3,$$
$$\Phi(u_A) = 4 - 0 = 4, \qquad \Phi(u_R) = 4 - \infty = -\infty.$$

Given $\gamma = 0.9$ and the RM state beliefs $\tilde{u}_t = [1, 0, 0, 0]^\top$ and $\tilde{u}_{t+1} = [0, 0.5, 0.5, 0]^\top$, the shaped reward is:

$$\begin{aligned} r_s(\tilde{u}_t, \tilde{u}_{t+1}) &= \sum_{u \in U} (\gamma \tilde{u}_{t+1}(u) - \tilde{u}_t(u))\Phi(u) \\ &= (0.9 \cdot 0 - 1) \cdot 3 + (0.9 \cdot 0.5 - 0) \cdot 3 \\ &+ (0.9 \cdot 0.5 - 0) \cdot 4 + (0.9 \cdot 0 - 0) \cdot -\infty \\ &= 0.15. \end{aligned}$$

3.4 Interleaved Learning Algorithm

We now describe PROB-IRM, our method for interleaving the learning of RMs from noisy traces with RL. The pseudocode is shown in Algorithm 1.

Algorithm 1 PROB-IRM algorithm

```
1: M ← INITRM({u_0, u_A, u_R})
2: E ← {}                                   ▷ Set of noisy examples
3: step_cnt ← 0                             ▷ Steps since last RM learning
4: ce_sum ← 0                               ▷ Cross entropy sum
5: INITQFUNCTION(M)
6: for ep ∈ {1, ..., num_episodes}
7:     s, ũ_p ← ENVINITIALSTATE()
8:     λ̃ ← ⟨⟩ ; t ← 0 ; done ← ⊥
9:     while done = ⊥
10:        a ← GETACTION(s, ũ_p)
11:        s', done ← ENVSTEP(s, a)
12:        ũ_q ← GETRMBELIEF(L̃(s, a, s'), ũ_p)
13:        UPDATETRACE(L̃(s, a, s'), λ̃)
14:        UPDATEQFUNCTION(s, a, s', δ_r(ũ_p, ũ_q), ũ_q)
15:        s ← s' ; ũ_p ← ũ_q ; t ← t + 1
16:        if ISTERMINAL(M, ũ_p) or t > max_ep_len
17:            done ← ⊤
18:    E_new ← GENERATEEXAMPLES(λ̃)
19:    E_new_inc ← GENERATEINCEXAMPLES(λ̃)
20:    E ← E ∪ E_new ∪ E_new_inc
21:    step_cnt ← step_cnt + 1
22:    ce ← RECOGNIZEBELIEF(M, λ̃, ũ_p)
23:    ce_sum ← ce + ce_sum
24:    if SHOULDRELEARN(ep, ce_sum, step_cnt)
25:        M ← RELEARNREWARDMACHINE(E)
26:        step_cnt ← 0
27:        ce_sum ← 0
28:        INITQFUNCTION(M)
29: function RECOGNIZEBELIEF(M, λ̃, ũ_{t+1})
30:    expected_belief ← TRACEOUTCOME(λ̃)
31:    return CROSSENTROPY(ũ_{t+1}, expected_belief)
32: function SHOULDRELEARN(ep, ce_sum, step_cnt)
33:    if step_cnt < warmup_steps
34:        return ⊥
35:    return ce_sum/step_cnt < β
```

Lines 1–5 initialise the candidate RM M, the set of noisy ILASP examples E, variables tracking RM relearning, and the Q-function. The algorithm is then executed for a fixed number of episodes. For each episode step, the agent executes an action in the environment (line 11), gets the new RM state belief $\tilde{u}$ (line 12), and updates the noisy trace (line 13) and the Q-function (line 14). The episode terminates if (i) the environment signals that the task has been completed (line 11); (ii) the most likely state of the RM is the accepting/rejecting state; or (iii) after a fixed number of steps (lines 16–17). After the episode terminates, the ILASP examples are updated (line 20). This process differs from the original algorithm (Furelos-Blanco et al. 2021, see Section 3.2). We generate incomplete examples (line 19) from the newly seen traces similarly to Ardon, Furelos-Blanco, and Russo (2023).

The belief that the RM is correct is updated using the observed trace (lines 22–23). If the RM should be relearned (line 24), then ILASP is called (line 25), variables tracking the relearning condition are reset (lines 26–27), and the Q-function is reinitialised (line 28).

The function RECOGNIZEBELIEF (lines 29–31) computes how well the current trace conforms to the candidate RM. We assume that the ground-truth outcome of an episode (goal, dead-end, incomplete) is known with certainty. This outcome is a one-hot vector obtained through the TRACEOUTCOME function (line 30). On the other hand, given that we know the belief of being in an accepting ($[\tilde{u}_{t+1}]_{acc}$) and rejecting state ($[\tilde{u}_{t+1}]_{rej}$), we can compute the predicted outcome: ($[\tilde{u}_{t+1}]_{acc}$, $[\tilde{u}_{t+1}]_{rej}$, 1 - $[\tilde{u}_{t+1}]_{acc}$ - $[\tilde{u}_{t+1}]_{rej}$). The RECOGNIZEBELIEF returns the categorical cross-entropy between the predicted and ground truth outcome.

The function SHOULDRELEARN (lines 32–35) determines if the RM should be relearned. We use the average cross-entropy between the expected state belief for the observed trace and the current RM state as decision criteria to trigger the relearning of the RM: if the cross-entropy falls under the hyperparameter β, a new RM is learned. Intuitively, this metric can be seen as a way to evaluate how well the RM captures the traces that have been seen. Also, the RM should not be relearned too often to prevent relearning with similar examples and allow the new RM to influence the generation of new examples. Hence, the RM should not be relearned if a number of warm-up steps have not passed since the last relearning (lines 33–34).

4 Experimental Results

We evaluate PROB-IRM using the OFFICEWORLD (see Section 2.1). We aim to answer three research questions:

RQ1: Does PROB-IRM successfully allow agents to be trained to complete their tasks?

RQ2: In terms of agent performance, how do the RMs learned with PROB-IRM compare with hand-crafted ones?

RQ3: How sensitive is PROB-IRM to different noisy settings?

After a brief overview of our experimental setup, we start by comparing the performance of PROB-IRM agents with a baseline composed of agents provided with handcrafted RMs that perfectly expose the structure of each task. Then, we focus our analysis on the impact of significantly higher

noise to highlight the robustness of PROB-IRM in such scenarios. Finally, we present the results of two ablation studies that evaluate (i) the effectiveness of our reward-shaping scheme, and (ii) the performance difference between our belief-based approach and thresholding for handling noisy labels.

The source code is available at https://github.com/rparac/Prob-IRM.

4.1 Experimental Setup

Environment Configurations. We focus our analysis on the OFFICEWORLD domain presented in Section 2.1. On the one hand, the COFFEE, COFFEEMAIL, and VISITABCD tasks enable assessing PROB-IRM's ability to learn good-quality RMs and policies in progressively harder scenarios. On the other hand, thanks to its adoption in other existing RM-based work (Toro Icarte et al. 2018; Furelos-Blanco et al. 2021; Dohmen et al. 2022), this choice allows our results to be easily compared with similar research.

The complexity of solving any of the three tasks strongly depends on the OFFICEWORLD layout. To account for this, we conducted each experiment over 3 sets of 10 random maps for COFFEE and COFFEEMAIL, and 3 sets of 50 random maps for VISITABCD.

Sensor Configurations. We experiment with multiple sensor configurations, each with a specific choice of:

- *noise targets*: the set of sensors subject to noise. Either only the one associated with the first event needed to solve the task (*noise-first*),[2] or all of them (*noise-all*);
- *noise level*: the sensor detection specificity and sensitivity. To reduce the number of possible configurations, we only consider scenarios where both parameters are set to the same value, and we will thus refer to them jointly as *sensor confidence*.

While sensor confidence is the parameter we have direct control over in our experiments, its value is not very informative to a human reader, as its impact on the accuracy of an agent's sensor strongly depends on the prior probabilities of each label being detected. Therefore, we pick the values for this parameter in such a way as to determine specific values for the posterior probability of detecting any noisy label correctly. In particular, we experiment with three posterior values, each representing increasing noise levels: 0.9, 0.8 and 0.5. We also consider the absence of noise (the posterior equal to 1) as a baseline.

Policy Learning. The agent policies are learned via the RL algorithm for RMs outlined in Section 2.2. The Q-functions are stored as tables. To index a reasonably sized Q-function using an RM belief vector, we resort to *binning*: beliefs are truncated to a fixed number of decimals, effectively resulting in close values being considered identical.

In terms of exploration, we rely on an ϵ-greedy strategy. For all the experiments, we start from $\epsilon = 1$ and decay its value to $\epsilon = 0.1$ over 2000 agent steps. We then keep the parameter fixed for the rest of the training process.

We employ our proposed formulation of *probabilistic reward shaping* in all experiments (both with fixed and learnt RMs) using $\gamma = 0.99$.

Performance Metrics. We record the *undiscounted return* collected by each agent at the end of every episode. Then, we aggregate their values over the set of all agents trained in the context of a single experiment to compute their associated mean and standard deviation, and the various metrics needed to answer our research questions. The average results are represented through learning curves, each smoothed using a moving average with a window size of 100. The shaded areas reflect the standard deviation of the values computed using an identical sliding window.

4.2 Baseline Results

We compare the performance of PROB-IRM against a baseline composed of agents provided with handcrafted RMs. We focus on the *noise-first* scenario, as it represents an easier learning setting for agents.

The learning curves of both approaches are shown in Figure 2. The baseline agents (upper row) consistently converge to a high level of proficiency. As expected, the more noise, the more episodes are required to converge.

We make two observations in the case of PROB-IRM (lower row). First, most learning curves display one or more sudden changes: when a new RM is learnt, the agent's policy is discarded and a new one starts being trained, thus leading to a temporary decrease in performance. The frequency of RM relearning increases with the level of noise. Second, we observe that PROB-IRM often reaches the baseline performance in a comparable number of episodes, thus providing a positive answer to research questions **RQ1** and **RQ2**. We highlight three exceptions: the first is found in the noisiest configuration for COFFEEMAIL, where many agents triggered RM relearning near the end of their training, thus ending with subpar performance. The second and third exceptions are represented by the two most noisy configurations for VISITABCD, which is the hardest task.

4.3 Multiple Noise Sources

We here assess PROB-IRM's ability to deal with multiple independent sources of noise; hence, we consider the *noise-all* setting for the COFFEE task to answer **RQ3**. In this setting, the number of noise sources is effectively tripled as compared to our baseline experiments: in addition to the noisy ☕ sensor, the $*$ and *o* sensors are also noisy.

We initially considered the set of posterior values used in the previous experiments; however, our experiments do not terminate within a 24-hour timeout period using a posterior of 0.5. This is unsurprising, as the learning task associated with this configuration is extremely complex. After experimenting with different posteriors ranging from 0.5 to 0.8, we find 0.75 to be a soft limit for the reliable applicability of PROB-IRM. Although more noise can be handled correctly, the training time increases rapidly.

[2] In COFFEEMAIL, there is not a single event that satisfies this criterion; thus, we apply the noise on both the ☕ and ✉ sensors.

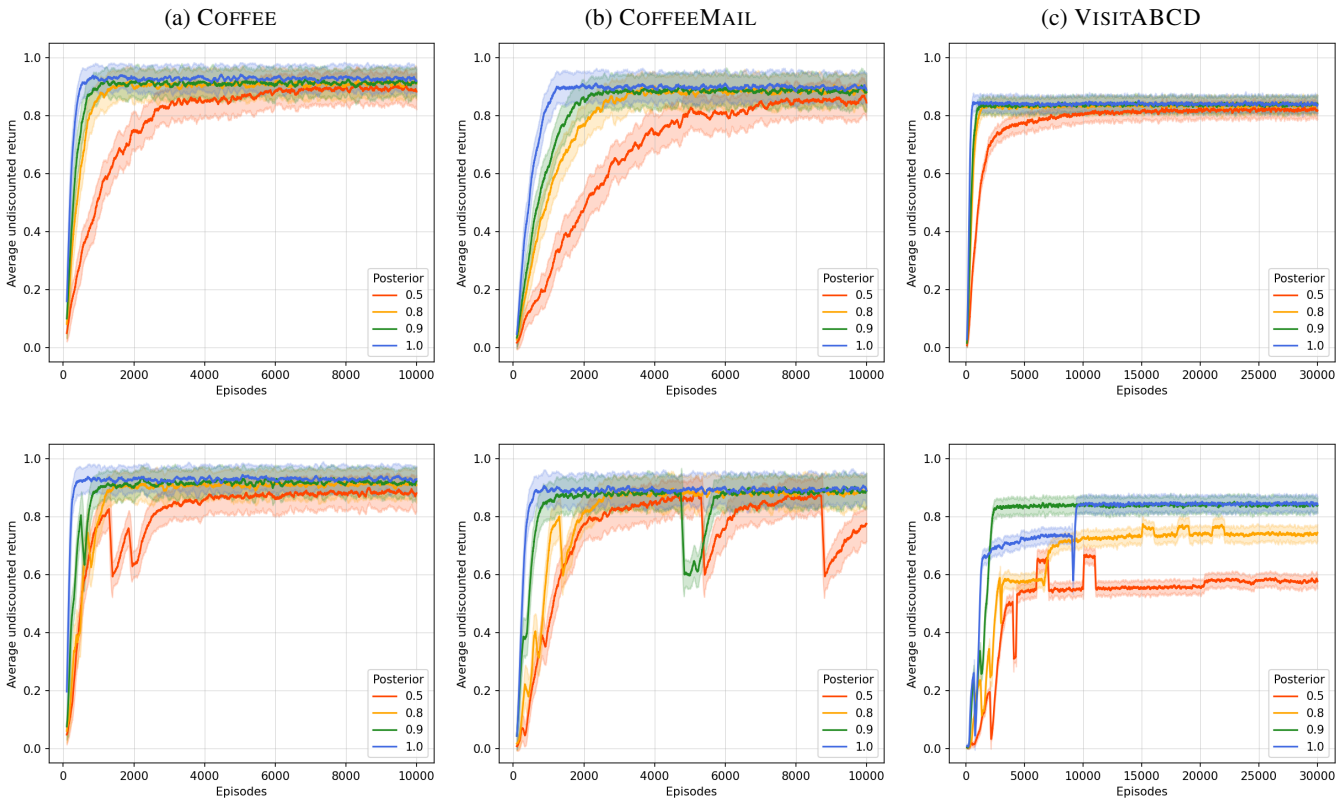

Figure 2: Learning curves for the three OFFICEWORLD tasks, where the *upper row* corresponds to the baseline agents provided with hand-crafted RMs, and the *lower row* corresponds to the agents trained with PROB-IRM.

Figure 3 presents the results. We observe that PROB-IRM agents consistently achieve a high performance despite the notable increase in noise. When comparing these results with the *noise-first* scenario considered in the previous section, we notice a slight decrease in the average undiscounted return and a larger number of episodes required to converge. The magnitude of these effects appears to be directly proportional to the amount of noise on each sensor, as corroborated by the learning curve for a posterior of 0.75.

4.4 Ablation Studies

Reward Shaping. To confirm the effectiveness of our reward shaping method, we replicate the setup in Section 4.3 but train the agents *without* reward shaping. Figure 4 presents the resulting learning curves. When comparing them with those in Figure 3, the positive impact of reward shaping is immediately apparent: for each level of noise, its use leads to a substantial decrease in the number of training episodes required to reach high performance, in addition to an actual increase in the agents' proficiency at the end of training. Moreover, its effectiveness grows higher as the amount of noise in the environment increases, thus proving it to be a very useful tool for dealing with such scenarios.

The reason behind its effectiveness lies in how reward shaping influences both policy training and RM learning. On the one hand, the intermediate rewards it provides help the agents improve their Q-value estimates faster, especially when rewards are sparse, thus leading to better policies sooner. On the other hand, they allow for incorrect RMs to be identified and discarded earlier in the learning process. When a candidate RM does not correctly reflect the task's structure, reward shaping nevertheless encourages the agents to act consequently. This, in turn, induces the agents to follow trajectories that are more likely to provide inconsistent examples for the RM learner.

Post-Hoc Analysis: Belief Updating *vs* Thresholding. In the last set of experiments, we aim to provide an empirical justification for our choice of a belief-updating paradigm, presented in Section 3.3. A valid alternative is represented by *thresholding*, which treats every proposition whose associated belief exceeds a fixed value as true. The main advantage of this strategy is not needing to maintain a belief over the RM states; instead, the RM state is updated by following the transitions triggered by the propositions deemed to be true after thresholding. Unfortunately, this upside is overshadowed by the fact that, under our sensor model, thresholding either works perfectly or does not work at all.

To support this claim, we train two sets of PROB-IRM agents to solve the COFFEE task under the *noisy-all* setting with a posterior of 0.8. The first set operates with a thresh-

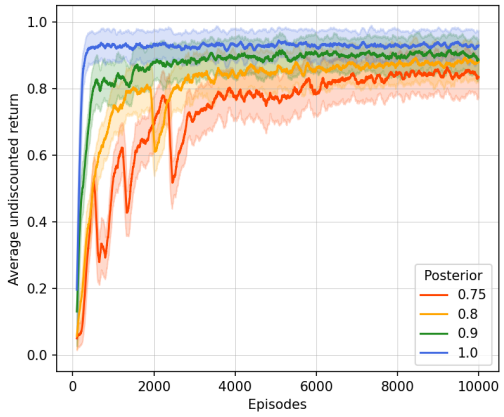

Figure 3: Learning curves for PROB-IRM agents trained to solve the COFFEE task in the *noise-all* scenario.

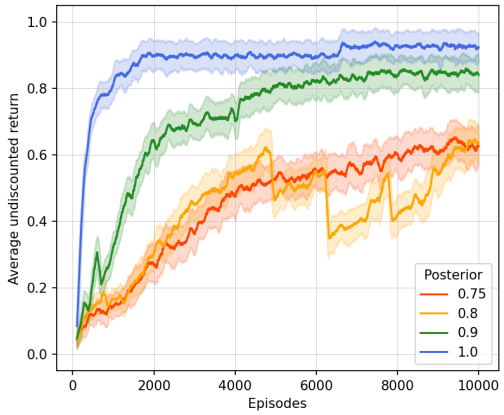

Figure 4: Learning curves for PROB-IRM agents trained to solve the COFFEE task in the *noise-all* scenario *without* reward shaping.

old of 0.7 (lower than the noise posterior), whereas the second uses a threshold of 0.9. Figure 5 presents the resulting learning curves. We observe that thresholding achieves good results when the threshold is lower than the posterior since it recognizes the truth value of every proposition correctly; however, when the threshold is too high, the agents *incorrectly* identify *every* proposition, thus making learning impossible. This highlights the issue of thresholding under our sensor model: to be successful, thresholding requires either precise knowledge of the noise posterior an agent is subject to, or a careful tuning of the threshold parameter. Both requirements can be restrictive in many practical use cases.

Unlike thresholding, the belief-updating approach we use is applicable regardless of the availability of any information, does not introduce an additional hyperparameter that might require extensive tuning, and performs very well in the same experimental setting we discuss in this section, as shown in Figure 3.

5 Related Work

Since the introduction of reward machines by Toro Icarte et al. (2018), there have been several approaches for learning them from traces in non-noisy settings. These approaches include discrete optimization (Toro Icarte et al. 2019; 2023), inductive logic programming (ILP; Furelos-Blanco et al. 2021; Ardon, Furelos-Blanco, and Russo 2023; Furelos-Blanco et al. 2023), program synthesis (Hasanbeig et al. 2021), or SAT solving (Xu et al. 2020; Corazza, Gavran, and Neider 2022). Our method employs ILASP (Law, Russo, and Broda 2015), a system also used in the aforementioned ILP works, to learn the RMs. Given its inherent robustness to noisy examples, ILASP-based approaches were the best positioned to be extended.

Existing RM learning methods focus on either accurately predicting the next label from the previous one (Toro Icarte et al. 2019; Hasanbeig et al. 2021; Toro Icarte et al. 2023), or learning a minimal RM (i.e., with the fewest number of states) that makes the reward signal Markovian (Xu et al. 2020; Furelos-Blanco et al. 2021; Hasanbeig et al. 2021; Corazza, Gavran, and Neider 2022; Ardon, Furelos-Blanco, and Russo 2023; Furelos-Blanco et al. 2023). Our method falls into the latter category. We hypothesize that the appearance of erroneous labels in the noisy setting makes RMs challenging to learn for methods in the former category.

The learning of RMs from noisy labels has only been previously considered by Verginis et al. (2024). Their work focuses on optimizing the noisy labelling function to model the perfect labelling function after thresholding, enabling the use of existing algorithms for RM learning. In contrast, our approach directly integrates noise into the RM learning procedure. Besides, PROB-IRM is orthogonal to the choice of the sensor model, so we could integrate a similar mechanism to achieve better results; however, their mechanism is less general since it assumes that in state s, the perfect labelling function $\mathcal{L}(s)$ always returns the same label L, making it unusable if the environment configuration (e.g., layout) changes between episodes.

Considering RMs with uncertainty more broadly, Corazza, Gavran, and Neider (2022) learn RMs that output stochastic rewards. Dohmen et al. (2022) propose an algorithm for learning probabilistic RMs, which are stochastic in both the state-transition and the reward-transition functions. The exploitation of RMs with a noisy labelling function was previously studied by Li et al. (2022; 2024); indeed, we employ their *independent belief updating* approach in the RM exploitation and discuss the *thresholding* approach in the evaluation.

6 Conclusions and Future Work

In this paper, we have introduced PROB-IRM, a method for learning and exploiting RMs from noisy traces perceived by a RL agent. We have shown that the method performs comparably to an algorithm where the RM is given in advance. Being able to deal with noisy labelling functions enables more realistic applications of RM-learning.

In future work, we plan to assess PROB-IRM's performance in continuous domains such as WATER-

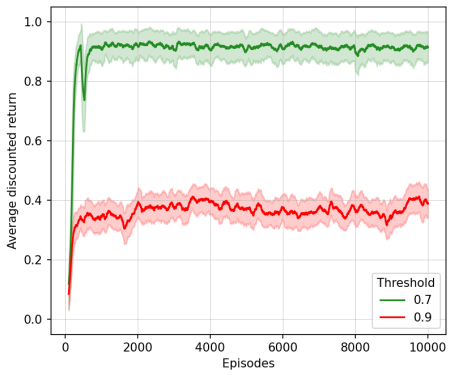

Figure 5: Learning curves for PROB-IRM agents employing thresholding to solve the COFFEE task under a *noise-all* setting with a posterior of 0.8.

WORLD (Karpathy 2015) and with autonomous embodied agents. Previous studies have demonstrated that incorporating an adversary into the RL training process enhances policy robustness (Pinto et al. 2017); likewise, we aim to investigate whether RM learning from noisy examples is resilient against attacks on RMs (Nodari 2023). Finally, we strive to improve the system's scalability to learn RM with more than seven states. Learning hierarchies of RMs (Lauffer et al. 2022; Furelos-Blanco et al. 2023) is a promising possibility since they enable learning smaller yet equivalent RMs.

Acknowledgements

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-222-0243. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorised to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

This work was also supported by the UK EPSRC project EP/X040518/1. Roko Parać is supported by the UKRI Centre for Doctoral Training in Safe and Trusted AI (EPSRC Project EP/S023356/1). We thank Julian de Gortari Briseno for his help with the noisy sensor model.

References

Ardon, L.; Furelos-Blanco, D.; and Russo, A. 2023. Learning Reward Machines in Cooperative Multi-Agent Tasks. In *Proceedings of the Neuro-Symbolic AI for Agent and Multi-Agent Systems (NeSyMAS) Workshop at the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Camacho, A.; Toro Icarte, R.; Klassen, T. Q.; Valenzano, R. A.; and McIlraith, S. A. 2019. LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 6065–6073.

Corazza, J.; Gavran, I.; and Neider, D. 2022. Reinforcement Learning with Stochastic Reward Machines. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI)*, 6429–6436.

Cunnington, D.; Law, M.; Lobo, J.; and Russo, A. 2023. FFNSL: Feed-Forward Neural-Symbolic Learner. *Machine Learning* 112(2):515–569.

Dohmen, T.; Topper, N.; Atia, G. K.; Beckus, A.; Trivedi, A.; and Velasquez, A. 2022. Inferring Probabilistic Reward Machines from Non-Markovian Reward Signals for Reinforcement Learning. In *Proceedings of the 32nd International Conference on Automated Planning and Scheduling (ICAPS)*, 574–582.

Dulac-Arnold, G.; Levine, N.; Mankowitz, D. J.; Li, J.; Paduraru, C.; Gowal, S.; and Hester, T. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning* 110(9):2419–2468.

Eck, A.; Soh, L.; Devlin, S.; and Kudenko, D. 2013. Potential-Based Reward Shaping for POMDPs. In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 1123–1124.

Fu, J., and Topcu, U. 2014. Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints. In *Proceedings of the 10th Robotics: Science and Systems Conference (RSS)*.

Furelos-Blanco, D.; Law, M.; Russo, A.; Broda, K.; and Jonsson, A. 2020. Induction of Subgoal Automata for Reinforcement Learning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, 3890–3897.

Furelos-Blanco, D.; Law, M.; Jonsson, A.; Broda, K.; and Russo, A. 2021. Induction and Exploitation of Subgoal Automata for Reinforcement Learning. *Journal of Artificial Intelligence Research* 70:1031–1116.

Furelos-Blanco, D.; Law, M.; Jonsson, A.; Broda, K.; and Russo, A. 2023. Hierarchies of Reward Machines. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 10494–10541.

Gelfond, M., and Kahl, Y. 2014. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. Cambridge University Press.

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 35th AAAI Conference on Artificial Intelligence (AAAI)*, 7647–7656.

Karpathy, A. 2015. REINFORCEjs: WaterWorld demo.

Lauffer, N.; Yalcinkaya, B.; Vazquez-Chanlatte, M.; Shah, A.; and Seshia, S. A. 2022. Learning Deterministic Finite Automata Decompositions from Examples and Demonstrations. In *Proceedings of the 22nd Conference on Formal Methods in Computer-Aided Design (FMCAD)*, 1–6.

Law, M.; Russo, A.; and Broda, K. 2015. The ILASP System for Learning Answer Set Programs.

Law, M.; Russo, A.; and Broda, K. 2018. Inductive Learning of Answer Set Programs from Noisy Examples. *Advances in Cognitive Systems* (7):57–76.

Law, M.; Russo, A.; and Broda, K. 2020. The ILASP system for Inductive Learning of Answer Set Programs. *arXiv preprint arXiv:2005.00904*.

Law, M. 2018. *Inductive Learning of Answer Set Programs*. Ph.D. Dissertation, Imperial College London, UK.

Li, A. C.; Chen, Z.; Vaezipoor, P.; Klassen, T. Q.; Toro Icarte, R.; and McIlraith, S. A. 2022. Noisy Symbolic Abstractions for Deep RL: A Case Study with Reward Machines. In *Proceedings of the Deep Reinforcement Learning Workshop at the 36th Conference on Advances in Neural Information Processing Systems (NeurIPS)*.

Li, A. C.; Chen, Z.; Klassen, T. Q.; Vaezipoor, P.; Toro Icarte, R.; and McIlraith, S. A. 2024. Reward Machines for Deep RL in Noisy and Uncertain Environments. *arXiv preprint arXiv:2406.00120*.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*, 278–287.

Nodari, L. 2023. Adversarial attacks to reward machine-based reinforcement learning. Master's thesis, Università degli Studi di Brescia. Available at https://arxiv.org/abs/2311.09014.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Pinto, L.; Davidson, J.; Sukthankar, R.; and Gupta, A. 2017. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2817–2826.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T. P.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. MIT Press.

Toro Icarte, R.; Klassen, T. Q.; Valenzano, R. A.; and McIlraith, S. A. 2018. Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2112–2121.

Toro Icarte, R.; Waldie, E.; Klassen, T. Q.; Valenzano, R. A.; Castro, M. P.; and McIlraith, S. A. 2019. Learning Reward Machines for Partially Observable Reinforcement Learning. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, 15497–15508.

Toro Icarte, R.; Klassen, T. Q.; Valenzano, R. A.; and McIlraith, S. A. 2022. Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning. *Journal of Artificial Intelligence Research* 73:173–208.

Toro Icarte, R.; Klassen, T. Q.; Valenzano, R.; Castro, M. P.; Waldie, E.; and McIlraith, S. A. 2023. Learning reward machines: A study in partially observable reinforcement learning. *Artificial Intelligence* 323:103989.

Verginis, C. K.; Koprulu, C.; Chinchali, S.; and Topcu, U. 2024. Joint learning of reward machines and policies in environments with partially known semantics. *Artificial Intelligence* 104146.

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 30th International Conference on Automated Planning and Scheduling (ICAPS)*, 590–598.