Fully Learnable Neural Reward Machines

Hazem Dewidar[1,*,†], Elena Umili[1,†]

[1]La Sapienza University of Rome

Abstract

Non-Markovian Reinforcement Learning (RL) tasks present significant challenges, as agents must reason over entire trajectories of state-action pairs to make optimal decisions. A common strategy to address this is through symbolic formalisms, such as Linear Temporal Logic (LTL) or automata, which provide a structured way to express temporally extended objectives. However, these approaches often rely on restrictive assumptions—such as the availability of a predefined Symbol Grounding (SG) function mapping raw observations to high-level symbolic representations, or prior knowledge of the temporal task. In this work, we propose a fully learnable version of Neural Reward Machines (NRM), which can learn both the SG function and the automaton end-to-end, removing any reliance on prior knowledge. Our approach is therefore as easily applicable as classic deep RL (DRL) approaches, while being far more explainable, because of the finite and compact nature of automata. Furthermore we show that by integrating Fully Learnable Reward Machines (FLNRM) with DRL outperforms previous approaches based on Recurrent Neural Networks (RNNs).

Keywords

Automata Learning, Neurosymbolic learning, Deep Reinforcement Learning

1. Introduction

Reinforcement Learning (RL) has achieved remarkable success, yet its foundational assumption of the Markov property—that the present state contains all information needed for optimal decision-making—is often violated in real-world scenarios. Many tasks are inherently non-Markovian, requiring an agent to remember and reason over a history of events to succeed. The RL community has largely pursued two paths to address non-Markovian tasks. The most common is to equip agents with an RNN [1, 2], which learns a compressed representation of history from experience. While powerful, this approach creates a "black box," making the agent's internal reasoning opaque and difficult to verify or trust. An alternative path uses structured, symbolic formalisms like automata or temporal logic [3, 4] to explicitly model the task's temporal structure. These models offer outstanding interpretability and support formal verification. However, their practical application is severely limited by a critical bottleneck: they typically presume that both the task automaton and a symbol grounding (SG) function—which maps raw observations like pixels to abstract symbols like 'key' or 'door'—are provided beforehand [5]. This reliance on prior, hand-engineered knowledge prevents their use in novel environments where such information is unavailable. This paper bridges the gap between these two paradigms. We build on Neural Reward Machines (NRMs) [5], which use a NeSy architecture to encode automaton knowledge and learn the SG function in a semi-supervised fashion. Here, we show how NRMs can be made fully learnable, thus removing the assumption of a known automaton. We introduce Fully Learnable Neural Reward Machines (FLNRM), the first framework to learn both the task automaton and the symbol grounding function simultaneously and end-to-end, directly from raw sensory inputs and scalar rewards. Our core contributions can be summarized as follow: (i) we propose a novel extension of NRM: FLNRM, that eliminates the need for prior knowledge of the task's temporal logic or its symbolic representation, a major limitation of prior automata-based RL methods (ii) we show experimentally that the learned structure provides a powerful inductive bias, enabling FLNRM to outperform standard RNN-based

OVERLAY 2025 @ ECAI 2025 October 26th, Bologna (Italy)

*Corresponding author.

†These authors contributed equally.

hazem.dewidar@uniroma1.it (H. Dewidar); umili@diag.uniroma1.it (E. Umili)

https://gladia.di.uniroma1.it/authors/dewidar/ (H. Dewidar); https://sites.google.com/view/elenaumili/home (E. Umili)

0009-0003-2377-2871 (H. Dewidar); 0000-0002-5639-6038 (E. Umili)

baselines, especially in tasks with complex logical constraints.Our method therefore retains the general applicability of standard Deep RL approaches, while improving performance and interpretability, taking the best from both automata-based and deep learning-based RL.

2. Related Works

Temporal logic formalisms are widely used in Reinforcement Learning (RL) to specify non-Markovian tasks [6], allowing agents to reason about temporally extended goals and constraints. Much of the existing literature assumes that: (1) the temporal specification is given, and (2) the boolean propositions used in the specification are observable in the environment—either perfectly [7, 8, 9, 10, 11, 12] or with some noise [13, 14, 15]. Many prior approaches relax only assumption (1), by integrating automata learning within RL agents [9, 10, 12]; or only assumption (2), using neurosymbolic (NeSy) frameworks [5] or multi-task RL techniques [16]; yet they still rely on one of the two.

Notably, recent work [17] learns both automata and latent event triggers from data without requiring predefined labeling functions or prior temporal knowledge. However, its use of Inductive Logic Programming (ILP) limits applicability only to discrete and finite symbolic domains, excluding environments providing raw observations, such as images or sensor data. In our approach, we learn the automaton describing the RL task structure directly from raw experience, without any prior knowledge or assumptions about the type of observations—which may be high-dimensional and continuous.

3. Background

Notation In this work, we consider *sequential* data of various types, including both symbolic and subsymbolic representations. Symbolic sequences are also called *traces*. Each element in a trace is a symbol σ drawn from a finite alphabet Σ. We denote sequences using bold notation. For example, $\boldsymbol{\sigma} = (\sigma^{(1)}, \sigma^{(2)}, \ldots, \sigma^{(T)})$ represents a trace of length T. Each symbolic variable in the sequence can be grounded either categorically or probabilistically. In the case of categorical grounding, each element of the trace is assigned a symbol from Σ, denoted simply as $\sigma^{(i)}$. In the case of probabilistic grounding, each symbolic variable is associated with a probability distribution over Σ, represented as a vector $\tilde{\sigma}^{(i)} \in \Delta(\Sigma)$, where $\Delta(\Sigma)$ denotes the probability simplex defined as

$$\Delta(\Sigma) = \left\{ \tilde{\sigma} \in \mathbb{R}^{|\Sigma|} \;\middle|\; \tilde{\sigma}_j \geq 0,\ \sum_{j=1}^{|\Sigma|} \tilde{\sigma}_j = 1 \right\}.$$

Accordingly, we distinguish between categorically grounded sequences $\boldsymbol{\sigma}$, and probabilistically grounded sequences $\tilde{\boldsymbol{\sigma}}$ using the tilde notation. Finally, note that we use superscripts to indicate time steps in the sequence and subscripts to denote vector components. For instance, $\tilde{\sigma}_j^{(i)}$ denotes the j-th component of the probabilistic grounding of σ at time step i.

Non-Markovian Reward Decision Processes In Reinforcement Learning (RL) [18] the agent-environment interaction is generally modeled as a Markov Decision Process (MDP). An MDP is a tuple (S, A, t, r, γ), where S is the set of environment *states*, A is the set of agent's *actions*, $t : S \times A \times S \to [0, 1]$ is the *transition function*, $r : S \times A \to \mathbb{R}$ is the *reward function*, and $\gamma \in [0, 1]$ is the *discount factor* expressing the preference for immediate over future reward. In this classical setting, transitions and rewards are assumed to be Markovian – i.e., they are functions of the current state only. Although this formulation is general enough to model most decision problems, it has been observed that many natural tasks are non-Markovian [6]. A decision process can be non-markovian because markovianity does not hold on the reward function $r : (S \times A)^* \to \mathbb{R}$, or the transition function $t : (S \times A)^* \times S \to [0, 1]$, or both. In this work we focus on Non-Markovian *Reward* Decision Processes (NMRDP) [19].

Reward Machines Rather than developing new RL algorithms to tackle NMRDP, the research has focused mainly on how to construct Markovian state representations of NMRDP. An approach of this kind are the so called Reward Machines (RMs). RMs are an automata-based representation of non-Markovian reward functions [20]. Given a finite set of propositions P representing abstract properties or events observable in the environment, a Reward Machine RM is a tuple $(P, Q, R, q_0, \delta, \lambda, L)$, where P is the automaton alphabet, Q is the set of automaton states, R is a finite set of continuous reward values, q_0 is the initial state, $\delta : Q \times P \to Q$ is the automaton transition function, $\lambda : Q \to R$ is the reward function, and $L : S \to P$ is the labeling (or symbol grounding) function, which recognizes symbols in the environment states. Let $\boldsymbol{s} = (s^{(1)}, s^{(2)}, ..., s^{(t)})$ be a sequence of states the agent has observed in the environment up to the current time instant t. This is transformed into a sequence of symbols $\boldsymbol{p} = (L(s^{(1)}), L(s^{(2)}), ..., L(s^{(t)}))$ by using the labeling function. This string of symbols is processed by the Moore Machine $(P, Q, R, q_0, \delta, \lambda)$ so to produce an history-dependent reward (output) value at time t, $r^{(t)}$, and an automaton state at time t, $q^{(t)}$. The reward value can be used to guide the agent toward the satisfaction of the task expressed by the automaton, while the automaton state can be used to construct a Markovian state representation. In fact it was proven that the augmented state $(s^{(t)}, q^{(t)})$ is a Markovian state representation for the task expressed by the RM [8].

Neural Reward Machines Neural Reward Machines (NRMs) are a probabilistic relaxation of standard Reward Machines, where the Moore machine is represented in matrix form, and input symbols, states, and rewards are *probabilistically grounded*. Given a Moore machine $(P, Q, R, q_0, \delta, \lambda)$ representing the task's reward structure—which is assumed to be known—we denote the transition and output (reward) functions in matrix form as $\mathcal{T} \in \mathbb{R}^{|P|\times|Q|\times|Q|}$ and $\mathcal{R} \in \mathbb{R}^{|Q|\times|R|}$, respectively. NRMs assume that the labeling function L is unknown and must be approximated by a neural network sg, which takes an environment state $s \in S$ as input and outputs a probability distribution over symbols $\tilde{p} \in \Delta(P)$, having trainable parameters θ_{sg}. The full model is formulated as follows:

$$\tilde{p}^{(t)} = sg(s^{(t)}; \theta_{sg}) \qquad \tilde{q}^{(t)} = \sum_{j=1}^{j=|P|} \tilde{p}_j^{(t)} (\tilde{q}^{(t-1)} \cdot \mathcal{T}_j) \qquad \tilde{r}^{(t)} = \tilde{q}^{(t)} \cdot \mathcal{R} \tag{1}$$

The model is fully continuous and differentiable, allowing its parameters θ_{sg} to be learned through gradient-based optimization on input-output target sequences. In particular, [5] train the model on episodes $(\boldsymbol{s}, \boldsymbol{r})$ collected from interactions with the environment.

4. Method

Fully Learnable Reward Machines In this paper, we extend NRMs to be *fully learnable*, and refer to our model as Fully Learnable Neural Reward Machines (FLNRM). We assume that no prior knowledge is provided to the model and it must learn an approximation of both the labeling function and the Moore Machine from experience. Since the task's Moore machine specification is unknown, the number of required states and symbols is also unknown. We initialize the number of symbols to $|\hat{P}|$ and the number of states to $|\hat{Q}|$. In contrast, the number of distinct reward values can be inferred through interaction with the environment, so we assume $|\hat{R}| = |R|$. As a result, $|\hat{P}|$ and $|\hat{Q}|$ are the only two hyperparameters of our model. The FLNRM model is shown in Figure 1, and it is formulated as follows

$$\begin{aligned} &\mathcal{T} = \text{softmax}(\theta_{\mathcal{T}}/\tau) && \mathcal{R} = \text{softmax}(\theta_{\mathcal{R}}/\tau) \\ &\tilde{p}^{(t)} = \text{softmax}(sg(s^{(t)}; \theta_{sg})/\tau) && \tilde{q}^{(t)} = \textstyle\sum_{j=1}^{|\hat{P}|} \tilde{p}_j^{(t)} (\tilde{q}^{(t-1)} \cdot \mathcal{T}_j) \qquad \tilde{r}^{(t)} = \tilde{q}^{(t)} \cdot \mathcal{R} \end{aligned} \tag{2}$$

Our model has three learnable sets of parameters: θ_{sg}, $\theta_{\mathcal{T}}$, and $\theta_{\mathcal{R}}$. Specifically, $\theta_{\mathcal{T}} \in \mathbb{R}^{|\hat{P}|\times|\hat{Q}|\times|\hat{Q}|}$ and $\theta_{\mathcal{R}} \in \mathbb{R}^{|\hat{Q}|\times|\hat{R}|}$ are matrices with the same dimensions as $\mathcal{T}$ and $\mathcal{R}$, respectively. The matrices $\mathcal{T}$ and $\mathcal{R}$ are obtained by applying a softmax activation to the corresponding parameters. This activation ensures that $\mathcal{T}$ and $\mathcal{R}$ define valid probability distributions over the next state and output[1]. A temperature

[1]Unless otherwise specified, the activation operates over the last dimension of each tensor. In this case, softmax ensures that each row of the matrix sums to one.

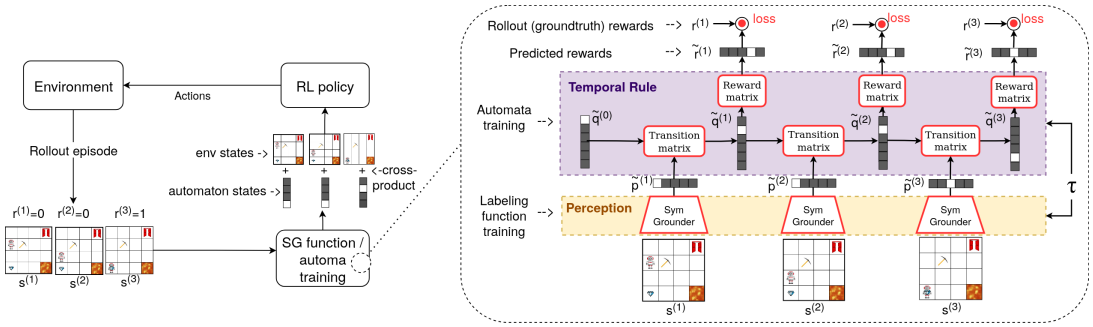

Figure 1: Integration of Fully Learnable Reward Machines within an RL schedule. The FLNRM is trained with episodes $(\boldsymbol{s}, \boldsymbol{r})$. At RL training time it produces the automaton states, that are used to augment the environment states so to mitigate non-markovianity. Models in red are trained through reward supervision.

parameter τ, with $0 < \tau \leq 1$, controls the sharpness of the softmax. When $\tau = 1$, the activation behaves normally; as τ approaches zero, the softmax approximates an argmax, and the model behaves increasingly like a deterministic finite state machine rather than a probabilistic one. Deterministic behavior emerges when all rows in the transition and reward matrices become one-hot vectors. We apply the same temperature-controlled activation to the symbol grounder network, so to smothly force the grounder to select only *one* symbol with maximum probability at each time-step.

Integrating FLNRM with deepRL In this section, we describe how FLNRM is integrated with policy learning through RL in non-Markovian domains. As in standard RL, we consider an agent interacting with an unknown environment. At each time step t, the agent takes an action $a^{(t)}$, observes the current state $s^{(t)}$, and receives a reward $r^{(t)}$. The agent's objective is to learn a policy $\pi : S \to A$ that maximizes the cumulative discounted reward: $\sum_{t=0}^{\infty} \gamma^t r^{(t+1)}$. We assume the reward signal is non-Markovian and can be modeled by a Reward Machine—namely, as the composition of a symbol perception function and a Moore machine. As the agent explores the environment, we record each episode as a sequence of states $\boldsymbol{s}$ and corresponding rewards $\boldsymbol{r}$. At regular intervals, we use the collected experience to train the FLNRM parameters by minimizing the cross-entropy loss between the predicted reward sequence $\tilde{\boldsymbol{r}}$ and the observed ground-truth rewards $\boldsymbol{r}$. Once the FLNRM has been trained, we use it to construct a history-dependent state representation to mitigate non-Markovianity. Specifically, we augment each environment state $s^{(t)}$ with the probabilistically grounded machine state $\tilde{q}^{(t)}$, and learn the policy over the augmented state space $\pi : S \times \Delta(\hat{Q}) \to A$. A schema of this process is shown in Figure 1.

5. Experiments

We validate our framework by replicating the experimental setup presented in the NRM paper [5]. Our implementation code is available at Github . In particular, we focus on navigation environments, where multiple items are present, and the agent must navigate among them so to satisfy a specific formula in Linear Temporal Logic over finite traces (LTLf) [21]. Two environments are designed to illustrate varying levels of difficulty in symbol grounding: (i) **Map Environment** – where the state is represented by a 2D vector indicating the agent's current (x, y) location; (ii) **Image Environment** – where the state consists of a $64 \times 64 \times 3$ pixel image depicting the agent within the grid. For each of these two environments we tested two classes of temporal tasks, focusing on formula patterns commonly used in non-Markovian reinforcement learning [22, 23] and denoted as in [24]: (i) **first class** - includes tasks defined as conjunctions of `Visit` formulas (the agent must reach some items without a predefined order) and `Seq_Visit` formulas (the agent must reach the items in a certain sequence). (ii) **second class** - includes tasks defined as conjunctions of `Visit`, `Seq_Visit`, and `Glob_Avoid` formulas (the agent must always avoid certain items). The complete list of formulas is reported in the Appendix.

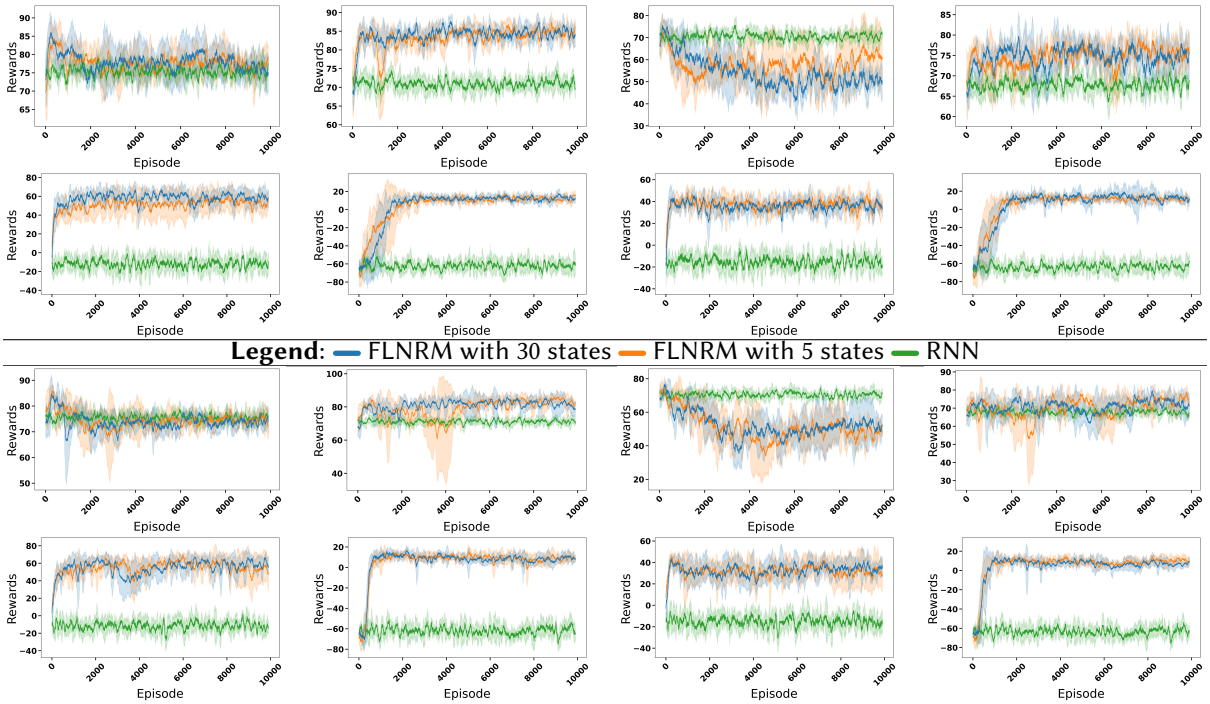

Figure 2: The first group of plots are about the image environment while the second bottom one is about the map environment. They show the training rewards obtained. First row: first class tasks. Second row: second class tasks.

Results We compare our method with RNN-based approaches using A2C [25] as RL algorithm, $|\hat{P}|$ equal to the groundtruth number of symbols $|P| = 5$, and $|\hat{Q}|$ equal to 5 and 30 states. Figures 2 show the training rewards obtained in both the image and map environments. For each task and method, we perform five runs with different random seeds. The results indicate that our method generally outperforms the baseline. Notably, the performance gap is most evident in the second class of tasks, which include the Global_Avoidance constraint. We attribute this to the strong and frequent feedback signals these clauses provide: violations trigger immediate and unambiguous negative rewards, which improve credit assignment and accelerate representation learning. All methods share the same hyperparameter settings for A2C, as well as for the neural networks used in the policy, value function, and feature extraction (the latter is only applied in the image environment), which are detailed in the appendix. The results shows that the number of states will not affect much the quality of the model (the rewards are almost the same). Also changing the observation function only brings minor variations in the results. Indeed, for the same LTLf task, the reward trend is similar in both environments, despite one being based on images and the other on vector observations. This demonstrates that our method effectively handle different types of raw observations without any issues.

6. Conclusions and Future Works

In this paper, we extend NRMs into Fully Learnable NRMs, which learn an automaton representation of the RL task directly from raw observations and exploit it in real time to accelerate RL performance. Through extensive experimentation, we show that our method generally surpasses the performance of Deep RL baselines based on RNNs. Our method thus retains the same broad applicability and improved performance compared to DRL approaches, while being grounded in symbolic, explainable, and logic-based methods—combining the best of both worlds. One current limitation of our experiments is the assumption of knowing the ground-truth number of symbols—an unrealistic constraint in many real-world scenarios. In future work, we aim to test the framework with imprecise estimates of the number of symbols, further widening its applicability.

Acknowledgments

The work of Hazem Dewidar was carried out when he was enrolled in the Italian National Doctorate on Artificial Intelligence run by Sapienza University of Rome. This work has been partially supported by PNRR MUR project PE0000013-FAIR.

Declaration on Generative AI

During the preparation of this work, the author(s) used chat-GPT in order to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] D. Ha, J. Schmidhuber, Recurrent world models facilitate policy evolution, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 31, Curran Associates, Inc., 2018.

[2] S. Kapturowski, G. Ostrovski, W. Dabney, J. Quan, R. Munos, Recurrent experience replay in distributed reinforcement learning, in: Proceedings of the 7th International Conference on Learning Representations (ICLR), 2019. URL: https://openreview.net/forum?id=r1lyTjAqYX.

[3] R. T. Icarte, T. Q. Klassen, R. A. Valenzano, S. A. McIlraith, Reward machines: Exploiting reward function structure in reinforcement learning, Journal of Artificial Intelligence Research 73 (2022) 173–208. doi:10.1613/JAIR.1.12440.

[4] G. De Giacomo, M. Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI '13), AAAI Press, 2013, pp. 854–860.

[5] E. Umili, F. Argenziano, R. Capobianco, Neural reward machines, 2024. URL: https://arxiv.org/abs/2408.08677. arXiv:2408.08677.

[6] M. L. Littman, U. Topcu, J. Fu, C. L. I. Jr., M. Wen, J. MacGlashan, Environment-independent task specifications via gltl, CoRR abs/1704.04341 (2017). URL: http://arxiv.org/abs/1704.04341.

[7] A. Camacho, R. T. Icarte, T. Q. Klassen, R. Valenzano, S. A. McIlraith, Ltl and beyond: Formal languages for reward function specification in reinforcement learning, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 6065–6073.

[8] G. D. Giacomo, L. Iocchi, M. Favorito, F. Patrizi, Foundations for restraining bolts: Reinforcement learning with ltlf/ldlf restraining specifications, in: Proceedings of the International Conference on Automated Planning and Scheduling, volume 29, 2021, pp. 128–136. URL: https://ojs.aaai.org/index.php/ICAPS/article/view/3549.

[9] M. Gaon, R. Brafman, Reinforcement learning with non-markovian rewards, Proceedings of the AAAI Conference on Artificial Intelligence 34 (2020) 3980–3987. URL: https://ojs.aaai.org/index.php/AAAI/article/view/5814. doi:10.1609/aaai.v34i04.5814.

[10] Z. Xu, B. Wu, A. Ojha, D. Neider, U. Topcu, Active finite reward automaton inference and reinforcement learning using queries and counterexamples, in: Machine Learning and Knowledge Extraction (CD-MAKE) 2021, 2021, pp. 115–135.

[11] A. Ronca, G. P. Licks, G. D. Giacomo, Markov abstractions for pac reinforcement learning in non-markov decision processes, in: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI 2022), Vienna, Austria, 2022, pp. 3408–3415. URL: https://doi.org/10.24963/ijcai.2022/473. doi:10.24963/ijcai.2022/473.

[12] D. Furelos-Blanco, M. Law, A. Jonsson, K. Broda, A. Russo, Induction and exploitation of subgoal automata for reinforcement learning, Journal of Artificial Intelligence Research 70 (2021) 1031–1116. URL: https://doi.org/10.1613/jair.1.12372. doi:10.1613/jair.1.12372.

[13] M. Cai, S. Xiao, B. Li, Z. Li, Z. Kan, Reinforcement learning based temporal logic control with maximum probabilistic satisfaction, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 806–812.
[14] C. K. Verginis, C. Koprulu, S. Chinchali, U. Topcu, Joint learning of reward machines and policies in environments with partially known semantics, CoRR abs/2204.11833 (2022). URL: https://doi.org/10.48550/arXiv.2204.11833. doi:10.48550/arXiv.2204.11833.
[15] A. C. Li, Z. Chen, P. Vaezipoor, T. Q. Klassen, R. T. Icarte, S. A. McIlraith, Noisy symbolic abstractions for deep rl: A case study with reward machines, CoRR abs/2211.10902 (2022). URL: https://doi.org/10.48550/arXiv.2211.10902. doi:10.48550/arXiv.2211.10902.
[16] Y. Kuo, B. Katz, A. Barbu, Encoding formulas as deep networks: Reinforcement learning for zero-shot execution of ltl formulas, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2020, 2020, pp. 5604–5610. URL: https://doi.org/10.1109/IROS45743.2020.9341325. doi:10.1109/IROS45743.2020.9341325.
[17] G. Hyde, E. S. Jr, Detecting hidden triggers: Mapping non-markov reward functions to markov, 2024. URL: https://arxiv.org/abs/2401.11325. arXiv:2401.11325.
[18] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed., The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.
[19] G. D. Giacomo, L. Iocchi, M. Favorito, F. Patrizi, Foundations for restraining bolts: Reinforcement learning with ltlf/ldlf restraining specifications, 2019.
[20] R. T. Icarte, T. Q. Klassen, R. A. Valenzano, S. A. McIlraith, Reward machines: Exploiting reward function structure in reinforcement learning, Journal of Artificial Intelligence Research 73 (2022) 173–208. doi:10.1613/JAIR.1.12440.
[21] G. D. Giacomo, M. Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI '13), AAAI Press, 2013, pp. 854–860.
[22] R. T. Icarte, T. Q. Klassen, R. A. Valenzano, S. A. McIlraith, Reward machines: Exploiting reward function structure in reinforcement learning, Journal of Artificial Intelligence Research 73 (2022) 173–208. doi:10.1613/JAIR.1.12440.
[23] P. Vaezipoor, A. C. Li, R. T. Icarte, S. A. McIlraith, Ltl2action: Generalizing LTL instructions for multi-task reinforcement learning, in: Proceedings of the 38th International Conference on Machine Learning (ICML), PMLR, Virtual Event, 2021, pp. 10497–10508.
[24] C. Menghi, C. Tsigkanos, P. Pelliccione, C. Ghezzi, T. Berger, Specification patterns for robotic missions, IEEE Transactions on Software Engineering 47 (2021) 2208–2224. URL: https://doi.org/10.1109/TSE.2019.2945329. doi:10.1109/TSE.2019.2945329.
[25] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, CoRR abs/1602.01783 (2016). URL: http://arxiv.org/abs/1602.01783. arXiv:1602.01783.

A. Experimental details

A.1. Task Formulas

We selected 8 formulas as RL tasks, 4 of class 1 and 4 of class 2, that are detailed in Table 1.

A.2. Hyperparameters Setting

The proposed model is designed to learn a variable number of latent states, denoted by $\hat{Q}$. In our experiments, we evaluated performance under two configurations: $\hat{Q} = 5$ and $\hat{Q} = 30$. The recurrent neural network (RNN) component was configured as an LSTM of two layers (`num_layers` = 2) and an output dimensionality of `rnn_outputs` = 5. The size of the hidden state in the RNN was set to `rnn_hidden_size` = 50. For the Advantage Actor-Critic (A2C) architecture, the hidden layer size

Task	Class	Pattern	Formula
1	1	Visit(a, b)	F(a) $\wedge$ F(b)
2	1	Visit(a, b, c)	F(a) $\wedge$ F(b) $\wedge$ F(c)
3	1	Sequenced_Visit(a, b)	F(a $\wedge$ F(b))
4	1	Sequenced_Visit(a, b) + Visit(c)	F(a $\wedge$ F(b)) $\wedge$ F(c)
5	2	Visit(a, b) + Global_Avoidance(c)	F(a) $\wedge$ F(b) $\wedge$ G($\neg$c)
6	2	Visit(a, b) + Global_Avoidance(c, d)	F(a) $\wedge$ F(b) $\wedge$ G($\neg$c) $\wedge$ G($\neg$d)
7	2	Sequenced_Visit(a, b) + Global_Avoidance(c)	F(a $\wedge$ F(b)) $\wedge$ G($\neg$c)
8	2	Sequenced_Visit(a, b) + Global_Avoidance(c, d)	F(a $\wedge$ F(b)) $\wedge$ G ($\neg$c) $\wedge$ G($\neg$d)

Table 1
Formulas selected as RL tasks

was fixed at `hidden_size` = 120. The learning rate of the optimizer is set to `lr=0.0004` while the temperature value used is 0.5.