FORM: Learning Expressive and Transferable First-Order Logic Reward Machines

Leo Ardon*
Imperial College London
London, United Kingdom
leo.ardon19@ic.ac.uk

Daniel Furelos-Blanco
Imperial College London
London, United Kingdom
d.furelos-blanco18@ic.ac.uk

Roko Parac
Imperial College London
London, United Kingdom
roko.parac18@ic.ac.uk

Alessandra Russo†
Imperial College London
London, United Kingdom
a.russo@ic.ac.uk

ABSTRACT

Reward machines (RMs) are an effective approach for addressing non-Markovian rewards in reinforcement learning (RL) through finite-state machines. Traditional RMs, which label edges with propositional logic formulae, inherit the limited expressivity of propositional logic. This limitation hinders the learnability and transferability of RMs since complex tasks will require numerous states and edges. To overcome these challenges, we propose First-Order Reward Machines (FORMs), which use first-order logic to label edges, resulting in more compact and transferable RMs. We introduce a novel method for *learning* FORMs and a multi-agent formulation for *exploiting* them and facilitate their transferability, where multiple agents collaboratively learn policies for a shared FORM. Our experimental results demonstrate the scalability of FORMs with respect to traditional RMs. Specifically, we show that FORMs can be effectively learnt for tasks where traditional RM learning approaches fail. We also show significant improvements in learning speed and task transferability thanks to the multi-agent learning framework and the abstraction provided by the first-order language.

CCS CONCEPTS

• **Theory of computation → Reinforcement learning**; • **Computing methodologies → Inductive logic learning**; *Logic programming and answer set programming*.

KEYWORDS

Reward Machine; Reinforcement Learning; First-Order Logic

ACM Reference Format:
Leo Ardon, Daniel Furelos-Blanco, Roko Parac, and Alessandra Russo. 2025. FORM: Learning Expressive and Transferable First-Order Logic Reward Machines. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 10 pages.

*Corresponding author.
†Sponsored in part by DEVCOM Army Research Lab under W911NF2220243 and EPSRC project EP/Y037421/1.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

1 INTRODUCTION

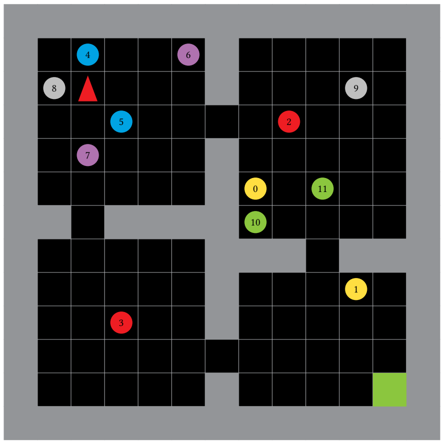
In recent years, reinforcement learning [RL; 33] has emerged as a powerful technique to train autonomous agents able to reach superhuman performance across a wide range of applications, including games [23, 29, 40, 41], autonomous driving [16, 26], finance [4, 30], and science [28, 31]. Despite these successes, RL struggles to handle tasks that require long-term planning and abstraction, particularly when rewards are non-Markovian, i.e. they depend on histories of states and actions. Reward machines [RMs; 37] offer a promising solution by leveraging finite-state machines to encode the temporal structure of tasks and allow agents to handle non-Markovian rewards. However, traditional RMs rely on propositional logic to label transitions, which limits their expressivity and scalability. Abstracting over object properties (e.g., *colour* or *type*) is unfeasible with propositional logic: it requires all combinations to be encoded in the RM, leading to a combinatorial explosion in the number of states and edges. As a result, RMs are difficult to learn and transfer across different tasks, especially as the complexity of the tasks increases.

In this paper, we make the following contributions. **(i)** We introduce First-Order Reward Machines (FORMs), a novel formulation of RMs that uses first-order logic to label transitions, enhancing expressivity and transferability. **(ii)** We propose a new RM learning method which is, to the best of our knowledge, the first to tackle the problem of *learning* RMs with first-order logic. **(iii)** We formalise the exploitation of an RM as a multi-agent problem, where multiple agents collaboratively learn the policies of the RM. Finally, **(iv)** we demonstrate empirically the benefit of our approach over traditional RM learning methods, in both learning and transferability.

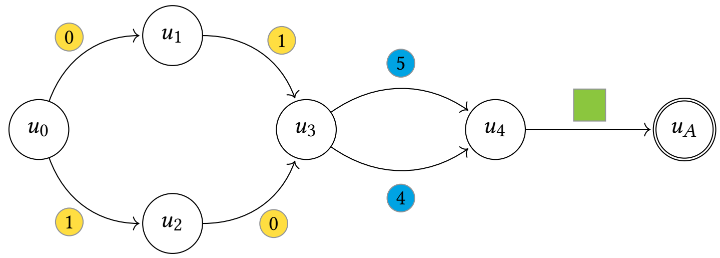
The paper is organised as follows. Section 2 introduces the background of our work. Section 3 formalises FORMs, and describes our methods for learning and exploiting them. We evaluate our methods' performance and the reusability of FORMs in Section 4. Section 5 discusses related work, and Section 6 concludes the paper.

2 BACKGROUND

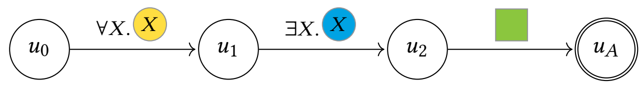
In this section, we cover three basic topics needed to understand our approach. We discuss reinforcement learning and a state-of-the-art algorithm which we use to learn policies. We present the notion of a reward machine which we build upon and generalise it to a First-Order Reward Machine (FORM). Finally, we discuss the Learning from Answer Sets framework, which we use to learn FORMs.



(a) Environment instance.



(b) Traditional RM encoding of the example task.



(c) FORM encoding of the example task.

Figure 1: An instance of the environment (a), and RMs for the task *"visit all followed by any before reaching "* (b, c). See Example 2.1 for details.

2.1 Reinforcement Learning

Reinforcement learning [RL; 33] is a machine learning paradigm where an agent learns to make sequential decisions by interacting with an environment, often modeled as a Markov decision process (MDP). The agent's objective is to learn a policy π, a mapping from states to actions, that maximizes the expected cumulative discounted reward (called return) $R = \sum_{t=0}^{T} \gamma^t r_t$, where $r_t \in \mathbb{R}$ is the reward received at time-step t, $\gamma \in [0, 1)$ is the discount factor, and T is the time horizon. The agent observes the current state s_t of the environment, takes an action $a_t \sim \pi(\cdot|s_t)$ according to its current policy π, transitions to a new state $s_{t+1} \sim p(\cdot|s_t, a_t)$ according to the environment transition function p, and receives a reward $r_t = r(s_t, a_t, s_{t+1})$ from the reward function r.

Proximal policy optimization [PPO; 27] is a state-of-the-art RL algorithm that, based on the policy gradient theorem [34], directly learns a policy $\pi(a_t|s_t)$ as an action distribution conditioned on a state s_t. PPO optimizes a surrogate objective function that balances exploration and exploitation; specifically, PPO uses a clipped probability ratio to constrain the policy update, ensuring that the new policy does not deviate too much from the old policy, preventing large and destabilizing changes. PPO has the advantage of being robust and effective on a wide range of tasks while requiring minimal hyperparameter tuning [43].

2.2 Learning from Answer Sets

Answer set programming [ASP; 15] is a declarative programming language for knowledge representation. A problem is expressed in ASP using logic rules, and the models (known as *answer sets*) of its representation are its solutions. Within the context of this paper, an ASP *program P* is a set of *normal* rules. Given any atoms $\mathsf{h}, \mathsf{b}_1, \ldots, \mathsf{b}_\mathsf{n}, \mathsf{c}_1, \ldots, \mathsf{c}_\mathsf{m}$, a normal rule is of the form `h :- b1, ..., bn, not c1, ..., not cm` where h is the *head*, $\mathsf{b}_1, \ldots, \mathsf{b}_\mathsf{n}$, not $\mathsf{c}_1, \ldots$, not c_m is (collectively) the *body* of the rule, and "not" represents negation as failure. An atom is *ground* if it is variable free. Informally, given a set of ground atoms (or *interpretation*) I, a ground normal rule is satisfied if the body is not satisfied by I or the head is satisfied by I. The reader is referred to [14] for further details on the semantics of ASP programs. ILASP [17] is a state-of-the-art inductive logic programming system for learning ASP programs from partial answer sets. A program P is said to accept an example e if and only if there exists an answer set A of $P \cup e$. An ILASP task [18] is a tuple $T = \langle B, S_M, E \rangle$, where B is the ASP background knowledge, S_M is the set of rules allowed in the hypotheses (called *hypothesis space*), and E is a set of examples . A *hypothesis* $H \subseteq S_M$ is a *solution* of T if and only if $B \cup H$ accepts all examples $e \in E$.

2.3 Reward Machines

Reward machines [RMs; 37, 38] are finite-state machine representations of reward functions. RMs encode the temporal structure of a task's rewards, enabling expressive and interpretable specifications; besides, they enable handling non-Markovian reward tasks by using the state of the RM as an external memory.

Reward machines are defined in terms of high-level events (or *observables*) $\mathcal{P}$, which aim to abstract the environment's state space S. More specifically, the transitions of an RM are labelled with sets of observables O (or *observations*). We denote by $\mathcal{O}$ the set of all possible observations, i.e. the powerset of $\mathcal{P}$, such that $|\mathcal{O}| \ll |S|$. The abstraction from environment states and actions to observations is performed by a *labelling function* $L : S \times A \times S \to \mathcal{O}$.

Definition 2.1 (Reward Machine). *An RM is a tuple $\mathcal{RM} = \langle \mathcal{U}, \mathcal{P}, u_0, u_A, u_R, \delta_u, \delta_r \rangle$, where $\mathcal{U}$ is a finite set of states; $\mathcal{P}$ is a set of observables whose powerset is denoted as $\mathcal{O}$; $u_0, u_A, u_R \in \mathcal{U}$ are the initial, accepting and rejecting state, respectively; $\delta_u : \mathcal{U} \times \mathcal{O} \to \mathcal{U}$ is a deterministic state-transition function that maps a RM state and an observation to the next RM state; and $\delta_r : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ is a deterministic reward-transition function that returns the reward given a pair of RM states determining a transition.*

We now introduce a running example used throughout the paper.

Example 2.1. *Let us consider an agent ( ) navigating the grid environment in Figure 1a. The grid consists of several checkpoints* *(e.g. ④), each characterised by a unique identifier and a colour, and a goal location ■ placed in the bottom right corner. These checkpoints and the goal constitute the set of observables $\mathcal{P}$. At each step, the agent observes the state of the environment. The labelling function L returns the set of observables in the cell where the agent has stepped on, e.g. {④} if the agent moves forward in Figure 1a.*

Several non-Markovian tasks can be formulated in this environment, each consisting of observing a sequence of observables, e.g. "visit all yellow checkpoints ● followed by any blue checkpoint ● before reaching ■". Figure 1b shows the propositional *RM for this task. To "visit all yellow checkpoints", the agent may start with ⓪ followed by ① or vice-versa. To "visit any blue checkpoints", the agent can visit either ④ or ⑤. After completing these sub-tasks, in the specified order, the agent can reach the goal ■.*

We *assume* the tasks are such that the reward is 1 if completed and 0 otherwise. Accordingly, in the case of RMs, we assume the reward is 1 only on transitions to the accepting state u_A. We thus omit the rewards from the RM figures for clarity.

Learning of Reward Machines. When the RM is unknown 'a priori', it can be learnt from sequences of observations (or *traces*) seen by the agent. Furelos-Blanco et al. [11] present an approach that learns RMs from traces using the `ILASP` inductive logic programming system introduced in Section 2.2. The RM is expressed in ASP using the following rules:

- Facts $\texttt{ed}(u, u', i)$, which encode the presence of an edge from u to u' with identifier i; and
- Rules with head $\bar{\phi}(u, u', i, \texttt{T})$, which encode the negation of the formulae to be satisfied for the RM transition i from u to u'.

Example 2.2. *Given the RM in Figure 1b, the ASP rules that encode the transition $u_3 \rightarrow u_4$ are:*

$$\begin{Bmatrix} \texttt{ed}(u_3, u_4, 0). \\ \texttt{ed}(u_3, u_4, 1). \\ \bar{\phi}(u_3, u_4, 0, \texttt{T})\texttt{:-not obs(④,T), step(T).} \\ \bar{\phi}(u_3, u_4, 1, \texttt{T})\texttt{:-not obs(⑤,T), step(T).} \end{Bmatrix},$$

where atoms $\texttt{obs}(o, \texttt{T})$ and $\texttt{step(T)}$ indicate respectively that observable o is observed at time $\texttt{T}$, and $\texttt{T}$ is a time-step. Note that (i) the disjunction is encoded with two edges whose indices are 0 and 1, and (ii) the transition rules encode $\bar{\phi}$ (i.e., the negation of ϕ), explaining the use of `not` [11].

The hypothesis space for the `ILASP` task learning the ASP rules that encode the reward machine, like the ones from Example 2.2, is defined as:

$$S_M = \begin{Bmatrix} \texttt{ed}(u, u', i). \\ \bar{\phi}(u, u', i, \texttt{T})\texttt{:-obs}(o,\texttt{T})\texttt{, step(T).} \\ \bar{\phi}(u, u', i, \texttt{T})\texttt{:-not obs}(o,\texttt{T})\texttt{, step(T).} \end{Bmatrix}, \quad (1)$$

where $u \in \mathcal{U} \setminus \{u_A, u_R\}$, $u' \in \mathcal{U} \setminus \{u\}$, $i \in [1, k]$, $o \in \mathcal{P}$, and k is a hyperparameter denoting the maximum number of disjuncts (i.e., edges) between each pair of states. Traces, which form the examples of an `ILASP` task, are encoded as a set of $\texttt{obs}(o, \texttt{T})$ facts; e.g., {`obs(■,0).obs(①,1).step(0).step(1)`}. To be considered as a solution, a hypothesis (i.e., the ASP encoding of an RM) must accept all traces.

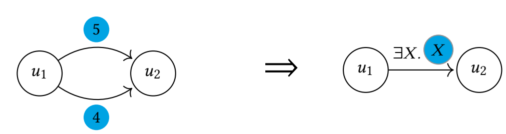
As it can be observed in Equation 1, the number of `ed` facts in the hypothesis space scales quadratically with the number of states in the RM ($|\mathcal{U}|$) and linearly with the maximum of disjuncts allowed in the transition rules (k): $(|\mathcal{U}| - 2) \cdot (|\mathcal{U}| - 1) \cdot k$. Additionally, the number of $\bar{\phi}$ rules depends on the number of observables ($|\mathcal{P}|$): $(|\mathcal{U}| - 2) \cdot (|\mathcal{U}| - 1) \cdot k \cdot 2|\mathcal{P}|$. As the number of states, disjuncts, or observables increase, learning the RM quickly becomes unfeasible. This underscores the need for new solutions to ensure that RM learning can scale effectively.

3 METHODOLOGY

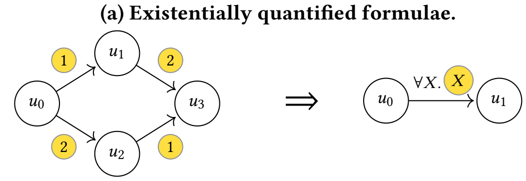
The limited expressivity of propositional logic formulae presents significant challenges to the broader adoption of RM-based approaches for RL. For instance, propositional logic cannot express tasks in terms of the properties of a specific or a group of entities, such as the colour of checkpoints in Example 2.1. This limitation substantially increases the number of sub-task combinations to be captured (if expressed at propositional level), resulting in larger RMs that are more difficult to learn and exploit. In addition, the restriction to a fixed set of observables hinders the re-use of RMs across different scenarios (e.g., on environments with different numbers of objects of the same colour). To address these limitations, we propose to label the edges of RMs using *first-order* formulae. The high expressivity of first-order logic allows for more compact and general RMs, making them easier to learn and transfer across different scenarios. We first illustrate the intuition behind our approach by building upon the RM from Example 2.1.

Example 3.1. *Figure 1c displays the First-Order RM (*`FORM`*) for the task "visit all yellow checkpoints ● followed by any blue checkpoint ● before reaching ■". Despite encoding the same task, the* `FORM` *is more compact than its propositional counterpart (Figure 1b).*

Existentially quantified formulae reduce the number of edges required to encode the disjunctions, impacting the parameter k in Equation 1 (Figure 2a). Universally quantified formulae reduce the number of states and, consequently, the number of edges required to encode a task (Figure 2b).[1]



(a) Existentially quantified formulae.



(b) Universally quantified formulae.

Figure 2: Equivalence between Propositional RMs and `FORM`s.

[1]An example with three yellow checkpoints is shown in [2, Appendix A].

We now formalise the first-order language over which the transitions of FORMs are defined (Section 3.1) and provide a formal definition of a first-order RM (Section 3.2). We then introduce our method for learning FORMs (Section 3.3) and exploiting them (Section 3.4).

3.1 Language

In this section, we introduce the first-order language $\mathcal{L}$ used to label transitions in FORMs. The *signature* of $\mathcal{L}$, denoted as $\Sigma = \langle C, \mathcal{K} \rangle$, is composed of a set C of *constants* and a set $\mathcal{K}$ of n-ary predicates. Predicates with 0-arity are referred to as *propositions.* A variable X of a predicate $\mathsf{P} \in \mathcal{K}$ is *bound* if it is in the scope of a quantifier $Q \in \{\forall, \exists\}$. A *ground atom* is a predicate whose arguments are constants. A *quantified atom* is a predicate whose variables are bound, e.g. $\exists X.\mathsf{P}(X)$. A *ground instance*, ψ_g, of a quantified atom ψ is a ground atom generated by replacing all variables in ψ with constants. Propositions, ground atoms and quantified atoms are *atomic formulae* of $\mathcal{L}$. The first-order language $\mathcal{L}$ of a FORM is defined as follows.

Definition 3.1. *Given a signature $\Sigma = \langle C, \mathcal{K} \rangle$, the language $\mathcal{L}$ is the set of* formulae *defined inductively as follows:*

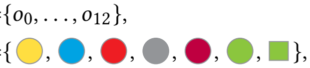
(1) *An atomic formula is a formula.*
(2) *If ψ is a formula, then so is $\neg\psi$.*
(3) *If ψ_1 and ψ_2 are formulae, then $\psi_1 \land \psi_2$, and $\psi_1 \lor \psi_2$ are formulae.*

We refer to formulae in a first-order language $\mathcal{L}$ as $\mathcal{L}$-formulae.

The set of all propositions and ground atoms in $\mathcal{L}$ is called the Herbrand Base of $\mathcal{L}$, denoted $\mathcal{HB}_{\mathcal{L}}$.

Example 3.2. *Consider the environment given in Figure 1a. The language $\mathcal{L}$ for this environment has the signature $\Sigma = \langle C, \mathcal{K} \rangle$ given by the following sets:*

$$C = \{o_0, \ldots, o_{12}\},$$

$\mathcal{K} =$ 

where the predicates are pictograms for unary predicates yellow*,* blue*,* red*,* gray*,* purple*,* green*, and the proposition* goal*, respectively. We depict a ground atom of $\mathcal{L}$ as a predicate pictogram with the subscript of its constant argument, e.g.* $\mathsf{blue}(o_4)$ *is depicted as* 4 *. Analogously, we depict quantified atoms with the quantifier and the predicate pictogram with the bound variable, e.g.* $\forall X.\mathsf{yellow}(X)$ *is depicted as* $\forall X; .$ X *. The proposition* ■ *and the negated formula* $\neg$ 4 *are examples of $\mathcal{L}$-formulae.*

3.2 First-Order Reward Machines (FORMs)

We formalise now the notion of a first-order RM (FORM), where transitions are labelled with formulae of a first-order language $\mathcal{L}$. FORMs enhance the expressivity of traditional RMs. The core entities that characterise a FORM are its observables, labelling function, and state-transition function. In what follows, we describe how these entities are defined and how the satisfiability of $\mathcal{L}$-formulae labelling the transitions of a FORM is checked.

Definition 3.2 (First-Order RM). *A First-Order RM over a first-order language $\mathcal{L}$ is a tuple* $\mathsf{FORM}_{\mathcal{L}} = \langle \mathcal{U}, \mathcal{HB}_{\mathcal{L}}, u_0, u_A, u_R, \delta_u, \delta_r \rangle$ *where $\mathcal{U}, u_0, u_A, u_R$ and δ_r are as defined in Definition 2.1, and*

- *$\mathcal{HB}_{\mathcal{L}}$ is the set of observables, and*
- *$\delta_u : \mathcal{U} \times O^* \to \mathcal{U} \times O^*$ is the state-transition function taking a state and a history of observations to a new state and an updated history.*

Observables and Labelling Function. Differently from traditional RMs, where *observables* are given as a set of propositions $\mathcal{P}$, the observables in a $\mathsf{FORM}_{\mathcal{L}}$ are defined as the Herbrand base of the first-order language $\mathcal{L}$. The *labelling function* L thus returns at each time-step, a subset of $\mathcal{HB}_{\mathcal{L}}$ as an observation.

State-Transition Function. The *state-transition function* δ_u generalises that of a traditional RM in two ways. Firstly, it supports the use of first-order formulae returned by a logical transition function $\phi : \mathcal{U} \times \mathcal{U} \to \mathcal{L}$. Secondly, it takes a *history of observations* instead of a single observation. This is important to semantically evaluate universally quantified atoms. To relax the requirement of having to see *all* the ground instances of the universally quantified atoms in the *same* observation, we use the following interpretation. Given a history, a universally quantified atom is TRUE when all its ground instances are contained in the history. We refer to this history as a *buffer* $B = [O_{t-k}, \ldots, O_t]$ which gathers observations from time-step $t - k$ to t, where $k \geq 0$ is the length of the buffer.

Algorithm 1 shows the pseudo-code that defines δ_u. The buffer B contains all the observations perceived since the current RM state was reached. The buffer is emptied when a transition to a new state is taken (1.4), or updated with the current observation otherwise (1.1). The $\mathcal{L}$-formulae are given by the logical transition function ϕ and are evaluated against the buffer (1.3).

Algorithm 1: State-Transition Function δ_u

Input: The current FORM state u, the latest observation O_t, the buffer B.
Output: The next FORM state u', the updated buffer .

```
1: B ← B ⊕ O_t
2: for all u' ∈ U do
3:    if B ⊨ φ(u, u') then
4:       B ← []
5:       return u', B
6: u' ← u
7: return u', B
```

In the following section, we define the notion of satisfiability of an $\mathcal{L}$-formula given a buffer B.

Satisfiability of $\mathcal{L}$-formulae. Given a buffer $B = [O_{t-k}, \ldots, O_t]$, observables that are included in B are assumed to be TRUE. Any other observable from $\mathcal{HB}_{\mathcal{L}}$ not included in B is assumed to be FALSE. We can now define the satisfiability of a $\mathcal{L}$-formula given a buffer $B = [O_{t-k}, \ldots, O_t]$.

Definition 3.3. *Let $B = [O_{t-k}, \ldots, O_t]$ be a buffer for some $k \geq 0$. Let ψ be a $\mathcal{L}$-formula. B satisfies ψ, written $B \models \psi$, is defined as follows:*

- *$B \models \psi$ if $\psi \in O_t$, where ψ is a proposition or a ground atom.*
- *$B \models \psi$ if $\psi_g \in O_t$ for* some *ground instance ψ_g of ψ, where ψ is an* ***existentially*** *quantified atom.*
- *$B \models \psi$ if $\psi_g \in \bigcup_{0 \leq i \leq k} O_{t-i}$ for* all *ground instance ψ_g of ψ, where ψ is a* ***universally*** *quantified atom.*

- $B \models \neg\psi$ *if* $B \not\models \psi$
- $B \models \psi_1 \wedge \psi_2$ *if* $B \models \psi_1$ *and* $B \models \psi_2$.

Note that a universally quantified formula is satisfied over a buffer when all ground instances of the formula, in the language $\mathcal{L}$ have been observed, i.e. are included in the buffer. The number of occurrences of the same observation throughout the buffer is not relevant. All other atomic formulae of $\mathcal{L}$ are satisfied over a buffer, if the most recent observation satisfies them.

Example 3.3. *Given the RM in Figure 1c, we exemplify the satisfiability of different* $\mathcal{L}$*-formulae* ψ *for different buffer examples:*

Proposition *Let* ψ = ■ *and* $B = [\{$■$\}]$*; then,* $B \models \psi$.

Existentially Quantified Atom *Let* $\psi = \exists X$ Ⓧ *and* $B = [\{$⓪$\}, \{$④$\}]$*; then,* $B \models \psi$.

Universally Quantified Atom *Let* $\psi = \forall X.$ Ⓧ, $B_1 = [\{$■$\}, \{$⓪$\}, \{$④$\}, \{$①$\}]$*, and* $B_2 = [\{$■$\}, \{$⓪$\}, \{$④$\}]$*; then,* $B_1 \models \psi$ *and* $B_2 \not\models \psi$.

Conjunctions *Let* $\psi = \forall X.$ Ⓧ $\wedge$ ■, $B_1 = [\{$■$\}, \{$⓪$\}, \{$④$\}, \emptyset, \{$①$\}]$*, and* $B_2 = [\{$■$\}, \{$⓪$\}, \{$④$\}, \emptyset, \{$①$\}, \emptyset, \{$■$\}]$*; then,* $B_1 \not\models \psi$ *since* $B_1 \models \forall X.$ Ⓧ *but* $B_1 \not\models$ ■*, and* $B_2 \models \psi$ *since* $B_2 \models \forall X.$ Ⓧ *and* $B_2 \models$ ■.

3.3 `FORM` Learning

We now focus on the problem of learning a `FORM` from the observation traces collected by the agent. Although observations are sets of ground atoms and propositions, the goal is to learn an abstract representation of the task's structure using lifted first-order formulae. To achieve this, we leverage the support of ASP for variables and learn ASP rules for the $\mathcal{L}$-formulae constituting the `FORM`.

The set of observables ($\mathcal{HB}_{\mathcal{L}}$) and the set of predicates ($\mathcal{K}$) are *not* known 'a priori'; instead, they are automatically derived from the observation traces and used to dynamically generate a new ILASP learning task. We model existentially and universally quantified atoms as ASP rules and expand the hypothesis space from Equation 1 to allow the edges of the RM to be labelled with quantified atoms or their negations. Formally,

$$S_M = \left\{ \begin{array}{l} \texttt{ed}(u, u', i). \\ \bar{\phi}(u, u', i, \texttt{T}) \texttt{:- obs}(o, \texttt{T}), \texttt{step(T)}. \\ \bar{\phi}(u, u', i, \texttt{T}) \texttt{:- not obs}(o, \texttt{T}), \texttt{step(T)}. \\ \bar{\phi}(u, u', i, \texttt{T}) \texttt{:- e\_pred}(p, \texttt{T}), \texttt{step(T)}. \\ \bar{\phi}(u, u', i, \texttt{T}) \texttt{:- not e\_pred}(p, \texttt{T}), \texttt{step(T)}. \\ \bar{\phi}(u, u', i, \texttt{T}) \texttt{:- a\_pred}(p, \texttt{T}), \texttt{step(T)}. \\ \bar{\phi}(u, u', i, \texttt{T}) \texttt{:- not a\_pred}(p, \texttt{T}), \texttt{step(T)}. \end{array} \right\},$$

where $u \in \mathcal{U} \setminus \{u_A, u_R\}$, $u' \in \mathcal{U} \setminus \{u\}$, $i \in [1, \kappa]$, $o \in \mathcal{HB}_{\mathcal{L}}$, and $p \in \mathcal{K}$.

Listing 1 presents the ASP encoding for an *existentially quantified atom*. The rule `e_pred(p,T)` is TRUE iff a ground atom of `p` is observed at time-step `T`. Listing 2 shows the ASP encoding of a *universally quantified atom*. The rule `p_holds(O,T)` is TRUE whenever the observable `O` is present in the *buffer* at time-step `T`. We use this rule to encode the `all_p_hold(T)`, evaluating to TRUE for all time-steps `T` for which all the ground atoms of `p` holds. Finally, the rule `a_pred(p,T)`, used in the hypothesis space, encodes a universally quantified atom and is TRUE for the first time-step `T` when all the ground atoms of `p` have been observed. These rules make use of the atoms:

- `obs(O, T)`: the observable `O` is seen at time-step `T`;
- `st(T, X)`: the RM was in state `X` at time-step `T`;
- `step(T)`: `T` is a valid time-step (i.e. its value is between 0 and the maximum trace length); and
- `state(X)`: `X` is an RM state.

Listing 1: ASP encoding for an existentially quantified sentence $\exists X.\texttt{p}(X)$

```
e_pred(p, T) :- obs(O, T), p(O).
```

The rule `e_pred(p,T)` *is* TRUE *when an observable seen at time-step* `T` *is a ground atom of* `p`.

Listing 2: ASP encoding for a universally quantified sentence $\forall X.\texttt{p}(X)$

```
# evaluate if the observation p(O) was observed at T
# while being in the current state of the RM.
p_holds(O, T) :-
    obs(O, T2), p(O),
    T >= T2, st(T, X), st(T2, X),
    step(T), step(T2), state(X).
# rules to evaluate whether all the ground atoms
# of p hold at T.
not_all_p_hold(T) :-
    not p_holds(O, T), p(O), step(T).
all_p_hold(T) :-
    not not_all_p_hold(T), step(T).
# rule used in the hypothesis space that will be true
# when all the ground atoms of p have been seen for
# the first time in the current state of the RM
a_pred(p, T) :- all_p_hold(T),
    not all_p_hold(T-1), step(T).
```

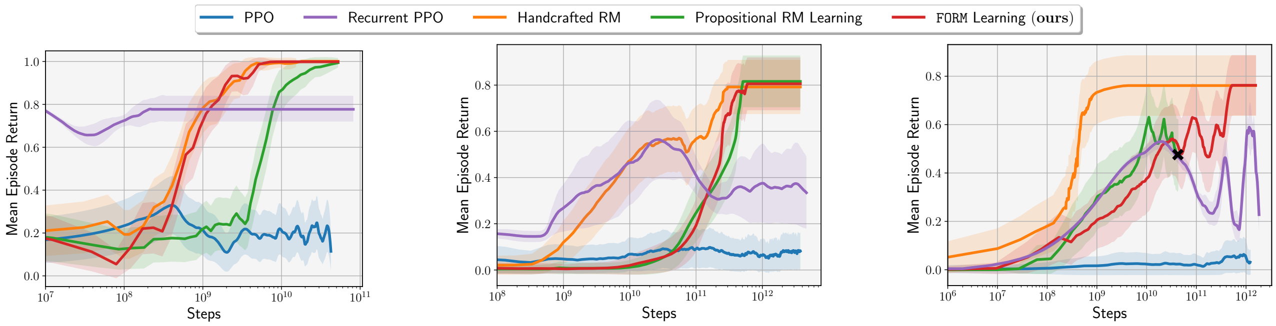
The rule `a_pred(p, T)` *is* TRUE *when all the ground atoms of* `p` *have all been observed while being in the same RM state.*

Determinism. The state-transition function δ_u, like in traditional RMs, must be deterministic; that is, two or more transitions cannot be simultaneously satisfied from a given state. To guarantee this property, the $\mathcal{L}$-formulae on such transitions must be *mutually exclusive*. Two transitions are mutually exclusive if an atomic formula appears positively on one transition and negatively on the other.

Mutual exclusivity is encoded through ASP rules and enforced at learning time so that the output `FORM`s are guaranteed to be deterministic. Since quantified atoms can represent multiple associated ground instances, we also define rules that translate quantified atoms into their corresponding sets of ground instances. We refer the reader to [2, Appendix E] for the encoding of mutual exclusivity and the mapping from quantified atoms into ground instances.

3.4 Policy Learning

The vast majority of the algorithms used to learn the policy associated with the RM are based on CRM [38], an algorithm that learns a *single* policy for the whole RM. This poses some concerns on the transferability of the learnt solution, and more specifically on the ability to reuse the policy associated with a given sub-task of the RM. To address this issue, we propose a novel formulation as a *multi-agent problem*, where a team of agents (one for each RM state) collaborate to complete the task. Our approach grounds the



(a) AllYellow (b) GreenButOne-NoLava (c) Blue-AllYellow-7

Figure 3: Average undiscounted return for the three tasks.

problem of RL with RM into the Markov game framework and, combined with FORM, enables the transfer of the learnt policies across scenarios (e.g., with different objects) to speed up the learning.

We formalise the problem as a collaborative *Markov game* of N agents defined with the tuple $\mathcal{G} = \langle \mathcal{RM}, N, \boldsymbol{S}, \boldsymbol{s_I}, \boldsymbol{A}, p, r, \gamma \rangle$, where $\mathcal{RM}$ is a FORM, $N = |\mathcal{U}_{\mathcal{RM}}|$ is the number of agents equal to the number of states in the FORM; $\boldsymbol{S} = S'_1 \times \cdots \times S'_N$ is the set of joint states; $\boldsymbol{s_I} \in \boldsymbol{S}$ is a joint initial state; $\boldsymbol{A} = A_1 \times \cdots \times A_N$ is the set of joint actions; $p : \boldsymbol{S} \times \boldsymbol{A} \to \boldsymbol{S}$ is a deterministic joint transition function; and $r : (\boldsymbol{S} \times \boldsymbol{A})^+ \times \boldsymbol{S} \to \mathbb{R}^N$ is the joint reward function. The objective of the game is to find a team policy $\boldsymbol{\pi} = \pi_1 \times \cdots \times \pi_N$ mapping joint states to joint actions such that the sum of the individual expected cumulative rewards is maximised.

In practice, each state of the FORM is assigned an RL agent responsible for learning a policy to accomplish the associated sub-task. Since the FORM state-transition function is deterministic, we can only be in a single state at any given time; therefore, only one RL agent acts at each time-step. The joint reward function is designed so that the reward received is uniformly shared among all the agents that contributed to reaching the current state (i.e., $\frac{r}{n'}$ where r is the global reward obtained and n' is the number of agents that have participated in getting r), implying that their assigned FORM state was visited. Consequently, the agents work *collaboratively* and *in turns* to traverse the FORM towards the accepting state.

Agent State Space. Since all agents act in the same environment, their state space S'_u is the environment's state space S. However, depending on the outgoing transitions from a given FORM state $u \in \mathcal{U}$, the state space S'_u of the associated agent may be extended. As described in Section 3.2, universally quantified formulae are evaluated over multiple time-steps using an observation buffer B, making the sub-task non-Markovian as the policy becomes dependent on a history (i.e., the buffer).

We address this issue by extending the state space S'_u of agent u with indicators encoding whether the ground atoms of interest have been perceived by the agent, hence making the sub-task Markovian. Let g_u denote the set of ground atoms of a universally quantified formula that must be satisfied to exit a FORM state u. The extended state s'_u is constructed by concatenating the state s from the environment with the indicators computed using the buffer B:

$$s'_u = s \oplus [\mathbb{1}_{\{p \in B\}} \; ; \forall p \in g_u].$$

Example 3.4. *Let us consider* $\phi_{u \to u'} = \forall X.$ (X) $\vee$ (2) *. The steps followed to extend the state space are:*

(1) *Identify the universally quantified predicates:* { ( ) }.
(2) *Get the ground atoms associated:* $g_u = [$ (4), (5) $]$.
(3) *Extend the state space:* $S'_u = S \times \{0, 1\}^{|g_u|}$.

At time-step $t = 5$*, assume* $B = [\{$ (0) $\}, \{$ (4) $\}]$*; then,* $s'_{u,5} = s_5 \oplus [1, 0]$.

The FORM states without outgoing transitions labelled with universally quantified formulae keep the state S of the environment.

4 EXPERIMENTS

We demonstrate the benefits of the proposed FORM learning and exploitation approach in the environment from Figure 1a. The tasks posed in this environment are challenging to traditional RM learning methods, especially as the number of objects increases. We first show the benefits of FORM learning with respect to standard RL algorithms and methods leveraging propositional RMs (with and without learning them). Second, we show that first-order logic eases the transfer of RMs and policies to more complex scenarios (e.g., with more objects). Additional experimental details are described in [2, Appendix C].[2]

4.1 FORM Learning

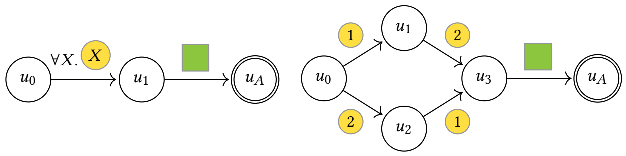
We evaluate the performance of our approach in its ability to learn and exploit FORMs. We consider three baselines: (i) PPO [27], to show how RMs help deal with non-Markovian tasks; (ii) Recurrent PPO using a recurrent network to keep track of the history (iii) Handcrafted FORM, which assumes the FORM is known 'a priori'; and (iv) a state-of-the-art propositional RM learning approach [11]. Both (iii) and (iv) employ our multi-agent based exploitation method.

In the following paragraphs, we compare our approach against the baselines in three, increasingly complex, tasks.

Task 1 – AllYellow. The task consists in visiting all yellow checkpoints ( ) before going to the ( ) location. Figure 1a illustrates the environment with two yellow checkpoints. This task is chosen to

[2] Code available at https://github.com/leoardon/form.

be relatively simple to ensure that traditional RM learning methods work so that we can compare against them.



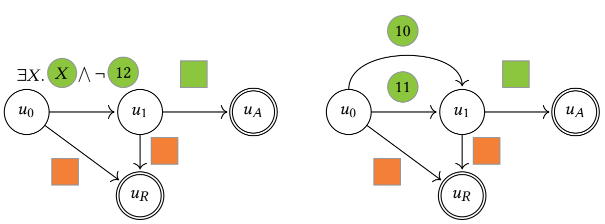
(a) FORM encoding. (b) Propositional RM encoding.

Figure 4: RMs for the AllYellow task.

Objective. We wish to assess the performance impact of learning FORMs as opposed to using a traditional RM learning method [11]. With fewer states, fewer sub-tasks need to be learnt, which should make the policy learning easier and therefore faster.

Results. Figure 4 presents the FORM and propositional RM encodings of the task and Figure 3a the learning curves of the different approaches. The traditional RL learning algorithm PPO (in *blue*) fails at solving this non-Markovian task since it does not learn history-dependent policies by default. Recurrent PPO (in *purple*) converges rapidly to a high return but fails to find the optimal policy. On the other hand, the RM-based solutions successfully learn an optimal policy to solve the task. The perfect FORM (Figure 4a) is learnt quickly (policy learning iteration #3) by our algorithm. Similarly, the perfect propositional RM (Figure 4b) is learnt at iteration #4; however, it requires learning more policies since the RM has more states. Our method (in *red*) converges significantly faster than the propositional RM learning method (in *green*), almost on par with the case where the RM is not learnt (in *orange*).

Task 2 – GreenButOne-NoLava. For this task, the environment is made more complex with the introduction of five randomly placed lava cells, ending the episode whenever the agent steps on any of them. The number of green checkpoints is increased from two to three. In this task, the agent must visit *any* green checkpoint different than 12 (which is also green), then reach the goal but without ever stepping on the lava.



(a) FORM encoding. (b) Propositional RM encoding.

Figure 5: RMs for the GreenButOne-NoLava task.

Objective. This experiment intends to show the expressivity of the learnable rules, including complex disjunctions between first-order formulae and ground atoms or propositions, and the ability to learn a FORM with rejecting states.

Results. The results are presented in Figure 3b, where we can see that the Handcrafted FORM baseline (in *orange*) outperforms the other two RM-based approaches converging faster. However, both the Propositional RM-based method (in *green*) and ours (in *red*) achieve a very similar level of performance. In both cases, they manage to correctly learn the RMs illustrated in Figure 5, showing that akin to traditional RM learning methods, our FORM-learning method can induce complex task structures involving both accepting and rejecting states. Both RMs share the same number of states, and thus the policies to train, which explains the similar performance.

Task 3 – Blue-AllYellow-7. The task consists of visiting *any* blue checkpoint, followed by visiting all yellow checkpoints, visiting the checkpoint 7, and reaching the location. The grid in Figure 1a contains two checkpoints for each colour. This task is much more complex than the previous ones since it combines existentially and universally quantified atoms, ground atoms, and propositions over a long sequence of sub-tasks.

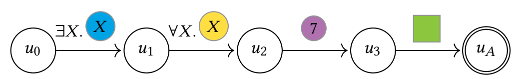


Figure 6: FORM for the Blue-AllYellow-7 task.

Objective. This experiment aims to demonstrate the scalability of our approach compared to existing RM learning techniques. The expressivity of the first-order language supported by FORMs enables a more compact encoding, addressing the scalability concerns of learning propositional RMs.

Results. The results are shown in Figure 3c. Our approach manages to learn the FORM encoding the structure of this task (Figure 6), while the propositional RM approach (in *green*) fails to find the correct RM (see Figure 9 in [2, Appendix A]) and times out due to the size of the RM to learn (two more states and twice as many edges as the FORM encoding). Moreover, our approach (in *red*) reaches the same performance level as that with a handcrafted FORM (in *orange*). Our approach therefore scales to more complex problems whereas existing methods cannot.

4.2 FORM Transfer

One of the advantages of FORMs lies in their ability to learn RMs that capture the structure of a task without necessarily being bound to the individual objects of the environments but rather to their properties (e.g., their colour in our case). It makes the RM transferable to new tasks, considerably improving learning efficiency. Our multi-agent formulation lets us also transfer the trained policies to the new task. Depending on the 'a priori' knowledge available, we may want to restrict the re-training to the ones that need it (see results below) or re-train all of them (see [2, Appendix F]). In both cases, the transferred policies are used as a warm start.

Setup. We reuse the FORM and policies learnt for the AllYellow task (Figure 3a) in two tasks. These tasks increase the number of yellow checkpoints to four and six, respectively. We evaluate the change in learning speed between learning from scratch *(w/o*

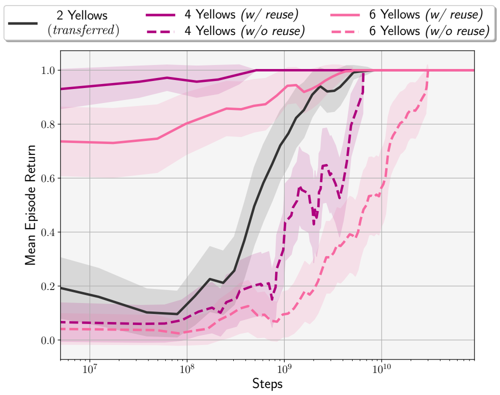


Figure 7: Transfer of the FORM learnt for the task from Figure 1c, to an environment with more yellow checkpoints *(w/ reuse)* compared against full FORM learning *(w/o reuse)*.

reuse) and reusing the FORM and policies *(w/ reuse)*. The policy associated with the RM state u_0 where the agent must "visit all " is retrained, whereas that for going to remains unchanged.

Objective. This experiment aims to show the transferability of FORMs and the ability to adapt quickly to new tasks. FORMs help abstract the environment settings, while the multi-agent formulation helps the transfer of the policies associated with the sub-tasks.

Results. Figure 7 shows *(in black)* the performance of the base solution being transferred (two yellow checkpoints) along with the performance of our approach learning the FORM and the policies for four and six yellow checkpoints *(coloured dotted lines)*. We then reuse the transferred FORM and policies to solve those tasks *(coloured solid lines)*. We see a significant speed-up in the learning of the optimal policies in both cases, highlighting the benefits of reusing existing solutions trained on simpler tasks. See [2, Appendix F] for more experiments.

5 RELATED WORK

Exploiting RMs. Recent work has also aimed at increasing the expressivity of RMs, e.g. by introducing counter variables that support richer grammars [7]. In parallel, other languages such as *linear temporal logic* have also been used to formulate task specifications [1, 9, 10, 36]. These approaches primarily focus on *exploiting* the *given* task structure. Unlike these methods, our approach tackles the challenge of *learning* the task structure itself.

The exploitation of RMs is often performed in the literature through QRM [37] or CRM [38]. Although both learn policies over the joint state space $S \times \mathcal{U}$ of the MDP and RM, the former associates an action-value function with each RM state, whereas the latter keeps a single action-value function. These algorithms employ a counterfactual mechanism that enables reusing the experiences for an RM state to update the value associated with other RM states; as such, the underlying RL algorithms are off-policy, e.g. DDQN [38], DDPG [32, 38], and SAC [32]. QRM and our approach both learn a policy for each RM state; however our approach uses an on-policy algorithm (PPO). Li et al. [20] also use PPO with RMs but learn a single policy conditioned on $S \times \mathcal{U}$. Besides, the Q-value function in QRM is updated by using the successor state's value function, creating a dependency between the policies of each state. We, on the other hand, take the approach of using the global reward returned by the RM as a signal for each agent, limiting the dependency between the policies, hence increasing transferability.

Our multi-agent exploitation method aligns with the *options* framework [35]. In our formulation, each agent operates an option determining the next action to accomplish a sub-task. In our case, an option terminates when the RM state changes (i.e., the option for the next RM state initiates). Unlike the options framework, which usually considers rewards during an option's activation, our approach distributes rewards to all agents that contributed to reaching the current RM state.

Learning RMs. Most RM research has focused on relaxing the assumption that the RM is known 'a priori', proposing various methods to learn it dynamically. These methods include discrete optimization [39], SAT solving [42], state-merging [13], inferring LTL formula from expert data [5], and inductive logic programming [11, 24], the latter of which forms the base of our approach. These techniques leverage traces collected through the agent's exploration or provided by an expert to *learn* the RM. All these methods have focused on learning propositional RMs, which are tailored to specific scenarios and thus lack reusability. Additionally, as the task complexity increases, the size of the RMs typically grows exponentially, making them impossible to learn. Finally, more loosely related to our work, other methods learn ASP rules as a proxy for the policy [6, 22]. However, unlike our work, the ASP encode the decision rule of the policy rather than the structure of the task.

To address the scalability issues inherent to propositional RMs, recent work has proposed to compose them hierarchically [12], learning small RMs that call each other. This approach requires knowing the level of each RM in the hierarchy and learns them bottom-up. Besides, existentially and universally quantified atoms cannot be hierarchically decomposed, making HRM suffer the same problem as standard RM learning approaches in these cases. Similarly, work on multi-agent RL decomposes a task across multiple agents, each learning their own RM [3]. While these approaches still focus on propositional RMs, they could be extended to incorporate our contributions.

6 CONCLUSION

While the literature has actively contributed to improving the exploitation and learning of RMs, existing methods primarily focus on propositional RMs, which are limited in their generalizability and scalability. In this paper, we introduced First-Order Reward Machines (FORMs), a novel extension of reward machines (RMs) that leverages first-order logic to enhance expressivity and transferability. We propose a learning algorithm that effectively *learns* FORMs from traces, and a multi-agent framework that facilitates efficient policy learning and transfer. Experimental results demonstrate that, unlike traditional RM methods, FORMs enhance scalability, enable faster policy and RM learning, and ease task transferability. Although our approach provides greater expressivity than propositional logic, extending our methodology to include languages like LTL could be an interesting avenue to be able to also learn the structure of temporally extended tasks.

REFERENCES

[1] Brandon Araki, Xiao Li, Kiran Vodrahalli, Jonathan A. DeCastro, Micah J. Fry, and Daniela Rus. 2021. The Logical Options Framework. In *Proceedings of the International Conference on Machine Learning (ICML)*. 307–317.

[2] Leo Ardon, Daniel Furelos-Blanco, Roko Parać, and Alessandra Russo. 2024. FORM: Learning Expressive and Transferable First-Order Logic Reward Machines. arXiv:2501.00364 [cs.AI] https://arxiv.org/abs/2501.00364

[3] Leo Ardon, Daniel Furelos-Blanco, and Alessandra Russo. 2023. Learning Reward Machines in Cooperative Multi-Agent Tasks. In *Autonomous Agents and Multiagent Systems. Best and Visionary Papers*. 43–59.

[4] Leo Ardon, Nelson Vadori, Thomas Spooner, Mengda Xu, Jared Vann, and Sumitra Ganesh. 2021. Towards a fully RL-based Market Simulator. In *Proceedings of the ACM International Conference on AI in Finance*. 1–9.

[5] Mattijs Baert, Sam Leroux, and Pieter Simoens. 2024. Learning Temporal Task Specifications From Demonstrations. In *Proceedings of the International Workshop on Explainable and Transparent AI and Multi-Agent Systems (EXTRAAMAS) at the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 81–98.

[6] Kexin Gu Baugh, Luke Dickens, and Alessandra Russo. 2025. Neural DNF-MT: A Neuro-symbolic Approach for Learning Interpretable and Editable Policies. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[7] Tristan Bester, Benjamin Rosman, Steven James, and Geraud Nangue Tasse. 2024. Counting Reward Automata: Sample Efficient Reinforcement Learning Through The Exploitation of Reward Function Structure. In *Proceedings of the Neuro-Symbolic Learning and Reasoning in the Era of Large Language Models Workshop (NucLeaR) at the AAAI Conference on Artificial Intelligence (AAAI)*.

[8] Alberto Camacho, Oscar Chen, Scott Sanner, and Sheila A. McIlraith. 2017. Non-Markovian Rewards Expressed in LTL: Guiding Search Via Reward Shaping. In *Proceedings of the International Symposium on Combinatorial Search (SOCS)*. 159–160.

[9] Alberto Camacho, Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. 2019. LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 6065–6073.

[10] Giuseppe De Giacomo, Marco Favorito, Luca Iocchi, Fabio Patrizi, and Alessandro Ronca. 2020. Temporal Logic Monitoring Rewards via Transducers. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*. 860–870.

[11] Daniel Furelos-Blanco, Mark Law, Anders Jonsson, Krysia Broda, and Alessandra Russo. 2021. Induction and Exploitation of Subgoal Automata for Reinforcement Learning. *Journal of Artificial Intelligence Research* 70 (2021), 1031–1116.

[12] Daniel Furelos-Blanco, Mark Law, Anders Jonsson, Krysia Broda, and Alessandra Russo. 2023. Hierarchies of Reward Machines. In *Proceedings of the International Conference on Machine Learning (ICML)*. 10494–10541.

[13] Maor Gaon and Ronen Brafman. 2020. Reinforcement Learning with Non-Markovian Rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 3980–3987.

[14] Michael Gelfond and Yulia Kahl. 2014. *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach.* Cambridge University Press.

[15] Michael Gelfond and Vladimir Lifschitz. 1988. The Stable Model Semantics for Logic Programming. In *Proceedings of the International Conference and Symposium on Logic Programming (ICLP/SLP)*. 1070–1080.

[16] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. 2019. Learning to Drive in a Day. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*. 8248–8254.

[17] Mark Law, Alessandra Russo, and Krysia Broda. 2015. The ILASP System for Learning Answer Set Programs. https://www.ilasp.com

[18] Mark Law, Alessandra Russo, and Krysia Broda. 2016. Iterative Learning of Answer Set Programs from Context Dependent Examples. *Theory and Practice of Logic Programming* 16, 5-6 (2016), 834–848.

[19] Mark Law, Alessandra Russo, and Krysia Broda. 2018. *The Meta-program Injection Feature in ILASP.* Technical Report. https://www.doc.ic.ac.uk/~ml1909/ILASP/inject.pdf

[20] Andrew Li, Zizhao Chen, Toryn Klassen, Pashootan Vaezipoor, Rodrigo Toro Icarte, and Sheila McIlraith. 2024. Reward Machines for Deep RL in Noisy and Uncertain Environments. In *Proceedings of the Advances in Neural Information Processing Systems Conference (NeurIPS)*. 110341–110368.

[21] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. 2018. RLlib: Abstractions for Distributed Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*. 3053–3062.

[22] Daniele Meli, Alberto Castellini, and Alessandro Farinelli. 2024. Learning Logic Specifications for Policy Guidance in POMDPs: an Inductive Logic Programming Approach. *Journal of Artificial Intelligence Research* 79 (2024), 725–776.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[24] Roko Parać, Lorenzo Nodari, Leo Ardon, Daniel Furelos-Blanco, Federico Cerutti, and Alessandra Russo. 2024. Learning Robust Reward Machines from Noisy Labels. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*. 909–919.

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of the Advances in Neural Information Processing Systems Conference (NeurIPS)*. 8024–8035.

[26] Ahmad El Sallab, Mohammed Abdou, Etienne Perot, and Senthil Kumar Yogamani. 2017. Deep Reinforcement Learning framework for Autonomous Driving. In *Autonomous Vehicles and Machines*. Society for Imaging Science and Technology, 70–76.

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347

[28] Andrew W. Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Zídek, Alexander W. R. Nelson, Alex Bridgland, Hugo Penedones, Stig Petersen, Karen Simonyan, Steve Crossan, Pushmeet Kohli, David T. Jones, David Silver, Koray Kavukcuoglu, and Demis Hassabis. 2020. Improved protein structure prediction using potentials from deep learning. *Nature* 577, 7792 (2020), 706–710.

[29] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[30] Thomas Spooner, John Fearnley, Rahul Savani, and Andreas Koukorinis. 2018. Market Making via Reinforcement Learning. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. 434–442.

[31] Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. 2020. A deep learning approach to antibiotic discovery. *Cell* 180, 4 (2020), 688–702.

[32] Haolin Sun and Yves Lespérance. 2023. Exploiting Reward Machines with Deep Reinforcement Learning in Continuous Action Domains. In *Proceedings of the European Conference on Multi-Agent Systems (EUMAS)*. 83–99.

[33] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction.* MIT Press.

[34] Richard S. Sutton, David A. McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Proceedings of the Advances in Neural Information Processing Systems Conference (NeurIPS)*. 1057–1063.

[35] Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112 (1999), 181–211.

[36] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. 2018. Teaching Multiple Tasks to an RL Agent using LTL. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*. 452–461.

[37] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. 2018. Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*. 2112–2121.

[38] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. 2022. Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning. *Journal of Artificial Intelligence Research* 73 (2022), 173–208.

[39] Rodrigo Toro Icarte, Ethan Waldie, Toryn Klassen, Rick Valenzano, Margarita Castro, and Sheila McIlraith. 2019. Learning Reward Machines for Partially Observable Reinforcement Learning. In *Proceedings of the Advances in Neural Information Processing Systems Conference (NeurIPS)*.

[40] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Çaglar Gülçehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.

[41] Peter R. Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J. Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, Leilani Gilpin, Piyush Khandelwal, Varun Raj Kompella, HaoChih Lin, Patrick MacAlpine, Declan Oller, Takuma Seno, Craig Sherstan, Michael D. Thomure, Houmehr Aghabozorgi, Leon Barrett, Rory Douglas, Dion Whitehead, Peter Dürr, Peter Stone, Michael Spranger, and Hiroaki Kitano. 2022. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature* 602, 7896 (2022), 223–228.

[42] Zhe Xu, Ivan Gavran, Yousef Ahmad, Rupak Majumdar, Daniel Neider, Ufuk Topcu, and Bo Wu. 2020. Joint Inference of Reward Machines and Policies for Reinforcement Learning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*. 590–598.

[43] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre M. Bayen, and Yi Wu. 2022. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Proceedings of the Advances in Neural Information Processing Conference (NeurIPS)*.