Synthesising Reward Machines for Cooperative Multi-Agent Reinforcement Learning

GIOVANNI VARRICCHIONE*, Utrecht University, The Netherlands
NATASHA ALECHINA, Open University, The Netherlands Utrecht University, The Netherlands
MEHDI DASTANI, Utrecht University, The Netherlands
BRIAN LOGAN, University of Aberdeen, United Kingdom Utrecht University, The Netherlands

Reward machines have recently been proposed as a means of encoding team tasks in cooperative multi-agent reinforcement learning. The resulting *multi-agent reward machine* is then decomposed into individual reward machines, one for each member of the team, allowing agents to learn in a decentralised manner while still achieving the team task. In this paper, we show how multi-agent reward machines for team tasks can be synthesised automatically from an abstraction of the environment in which the agents act and a high-level specification of the desired team behaviour expressed in a fragment of Alternating-time Temporal Logic. We present results from a number of benchmarks which suggest that our automated approach performs as well or better than reward machines in the literature.

JAIR Associate Editor: Sasha Rubin

JAIR Reference Format:
Giovanni Varricchione, Natasha Alechina, Mehdi Dastani, and Brian Logan. 2026. Synthesising Reward Machines for Cooperative Multi-Agent Reinforcement Learning. *Journal of Artificial Intelligence Research* 85, Article 38 (April 2026), 30 pages. DOI: 10.1613/jair.1.20151

1 Introduction

Reward machines (RMs) (Camacho et al. 2019; Toro Icarte et al. 2022, 2018, 2019) have been proposed as a way of specifying non-Markovian rewards for reinforcement learning (RL) agents. RMs are Mealy machines which represent tasks and rewards in terms of a high-level abstraction of the agent's environment. Providing an explicit encoding of the structure of the task has been shown to increase sample efficiency in reinforcement learning. For example, the RM-based algorithm proposed by Camacho et al. (2019) has been shown to out-perform state-of-the-art RL algorithms, especially in tasks involving temporally extended behaviours.

More recently, RMs have been proposed as a means of specifying rewards for team tasks in cooperative multi-agent reinforcement learning (Neary et al. 2021). In cooperative multi-agent reinforcement learning (MARL) (Panait and Luke 2005) the aim is to train a group of agents to perform a team task with the objective of maximising the expected future reward of the team. MARL is more challenging than single-agent RL. As the correctness of the actions of each agent may depend on the actions of other agents in the team, the agents must coordinate their actions (Boutilier 1996). In addition, the agents are learning and updating their policies simultaneously. From the

*Corresponding Author.

Authors' Contact Information: Giovanni Varricchione, ORCID: 0000-0002-5466-9012, g.varricchione@uu.nl, Utrecht University, Utrecht, The Netherlands; Natasha Alechina, ORCID: 0000-0003-3306-9891, natasha.alechina@ou.nl, Open University, Herleen, The Netherlands and Utrecht University, Utrecht, The Netherlands; Mehdi Dastani, ORCID: 0000-0002-4641-4087, m.m.dastani@uu.nl, Utrecht University, Utrecht, The Netherlands; Brian Logan, ORCID: 0000-0003-0648-7107, brian.logan@abdn.ac.uk, University of Aberdeen, Aberdeen, United Kingdom and Utrecht University, Utrecht, The Netherlands.

DOI: 10.1613/jair.1.20151

point of view of each individual agent, the learning problem is non-stationary; i.e., the optimal policy for each agent is constantly changing (Hernandez-Leal, Kaisers, et al. 2019).

Neary et al. (2021) address these problems by specifying a multi-agent reward machine which encodes the abstract structure of the team task. The multi-agent reward machine is then decomposed into individual reward machines, one for each member of the team. The decomposition is carried out by projecting the multi-agent RM onto the set of observable events of each agent in the team. If the decomposition is done in such a way that the combined behaviour of the individual reward machines is "bisimulation equivalent" to that of the team reward machine, each agent can be trained using its individual reward machine to perform its part of the team task in a decentralised manner while still ensuring that the team task will be achieved by the joint actions of the agents. This avoids the problem of non-stationarity, and Neary et al. (2021) propose an algorithm based on this approach called "Decentralised Q-Learning with Projected Reward Machines" (DQPRM) which is shown to be more sample efficient than independent Q-learners (IQL) (Tan 1993) and hierarchical independent learners (h-IL) (Tang et al. 2019). However, Neary et al. (2021) assume that the multi-agent reward machine is given as input.

Specifying automata-based rewards can be challenging, and there have been a number of proposals which aim to to *synthesise* RMs from a high-level specification of the agent's task and environment. For example, Illanes et al. (2019, 2020) and Varricchione et al. (2024) have used AI planning techniques to generate RMs for single agent learning tasks from a high-level goal the agent has to achieve and a high-level description of the agent's environment specified in terms of fluents and STRIPS-like actions. In this approach, the high-level specification of the agent's task and environment are typically expressed in terms of high-level states, actions and fluents which abstract away the details of the agent's environment, i.e., high-level actions are typically "coarse-grained" and deterministic ("move from X to Y"), while the actions available to the agent are, e.g., steering, accelerating, braking, etc., which may be nondeterministic.

In this paper, we show how to synthesise multi-agent and individual reward machines for fully cooperative multi-agent reinforcement learning. Specifically, we synthesise the reward machines from a high-level abstract description of the environment in which the agents act given as an epistemic concurrent game structure (ECGS), and a specification of the team task represented in a fragment of ATL, which we name "*Flat Cooperative ATL*" (*FC-ATL*), designed to express fully cooperative temporal multi-agent learning tasks. Using ATL model checking, we generate a witness for a coalition strategy that achieves the task in the high-level ECGS abstraction and then synthesise a multi-agent reward machine. The multi-agent reward machine is then decomposed into multiple single-agent rewards machines, one for each agent in the coalition, which are then used to train the agents individually in the environment in which they act. As in the approach by Neary et al. (2021), the individual reward machines constitute a high-level abstract specification of each agent's part of the team task which cannot be directly executed in the agent's environment. However they guide the agents in learning which low-level actions should be performed to accomplish each high-level action in their part of the team task. We state constraints on the definition of abstract states and actions of the ECGS necessary for agents to be able to learn to realise a high-level action as sequences of MAE actions. Moreover, the use of Flat Cooperative ATL allows us to incorporate additional constraints on team goals, e.g., invariant properties, which were not considered by Neary et al. (2021).

Synthesising a reward machine from a high-level abstraction of the agent's task and environment makes agents "taskable" (Illanes et al. 2020), as the user can define new agent tasks as goal conditions at the abstract level and reward machines are automatically computed for the task. With the exception of C. Zhu et al. (2024), synthesis of RMs has been confined to the single-agent setting. The key contribution of our approach is to extend RM synthesis to multi-agent settings and decompose the resulting multi-agent reward machine into individual reward machines allowing decentralised learning as in Neary et al. (2021). To evaluate our approach, we present results from a number of benchmarks from (Neary et al. 2021) and Toro Icarte et al. (2022) which suggest that RMs synthesised using our automated approach perform as well or better in terms of sample efficiency than those in the literature. In addition, we present results from a cooperative version of the continuous domain

WaterWorld introduced by Gupta et al. (2017), showing how our approach scales to more complex environments with continuous action- and state-spaces.

This paper is a significant extension of (Varricchione et al. 2023). Specifically, we provide formal definitions of the the notions "*safe abstractions*" and "*learnable tasks*" in the context of multi-agent systems which allow us to state precisely when it is possible to apply our approach given a multi-agent system and an abstraction of it. Full proofs are given of all theorems. We have also extended the evaluation to include experiments in the domains OfficeWorld and WaterWorld. The OfficeWorld domain illustrates how our approach can be used to express tasks involving (safety) invariants, and compares our automatically synthesised RMs with additional RMs from the literature. The WaterWorld domain shows how our decentralised learning approach can be applied in multi-agent scenarios with continuous action- and state-spaces. In addition, we have extended the evaluation in the Rendezvous domain, in order to evaluate how scalability (in terms of the number of agents) of our approach. Finally, we have extended the discussion of the related work.

The remainder of the paper is structured as follows. In Section 2 we introduce the necessary formal preliminaries that form the basis of our approach. In Section 3 we show how, given a multi-agent environment and an FC-ATL formula specifying a team task, we can synthesise a reward machine for each agent in the team from a witness for the FC-ATL formula. In Section 4 we present an evaluation of our automatically synthesised reward machines in four domains: CooperativeButtons and 10-Agent Rendezvous from Neary et al. (2021), OfficeWorld from Toro Icarte et al. (2022), and WaterWorld from Gupta et al. (2017). Section 5 contains an overview of related work. Finally, in Section 6 we conclude and outline directions for future work.

2 Preliminaries

In this section, we briefly introduce multi-agent reinforcement learning with reward machines and Alternating-time Temporal Logic which form the basis of our approach.

2.1 Multi-Agent RL with RMs

We begin by defining a *multi-agent environment* (MAE) that specifies the low-level environment in which the agents act.

Definition 1 (Multi-Agent Environment)**.** A multi-agent environment with n agents is a tuple $\langle Agt, (S_i)_{i\in Agt}, (A_i)_{i\in Agt}, Pr, (Prop_i)_{i\in Agt}, Val\rangle$ where:

- Agt is a non-empty finite set of n agents;
- S_i is the set of local states of agent i. We denote the set of joint states, i.e., the cartesian product of all the sets of states, with $\mathcal{S} = S_1 \times \ldots \times S_n$;
- A_i is the set of actions of agent i. We denote the set of joint actions, i.e., the cartesian product of all the sets of actions, with $\mathcal{A} = A_1 \times \ldots \times A_n$;
- $Pr : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \Delta(\mathcal{S})$ is the joint state transition probability distribution and $\Delta(\mathcal{S})$ is the set of all probability distributions over $\mathcal{S}$; $Pr(\boldsymbol{s'}|\boldsymbol{s}, \boldsymbol{a})$ denotes the probability of transitioning from a joint state $\boldsymbol{s} \in \mathcal{S}$ to a joint state $\boldsymbol{s'} \in \mathcal{S}$ by performing a joint action $\boldsymbol{a} \in \mathcal{A}$;
- $Prop_i$ is the finite set of propositional symbols "observable" by agent i, $Prop := \bigcup_{i\in Agt} Prop_i$ is the entire set of observable propositions. Note that the sets of propositional symbols observable by two agents i and j are not necessarily disjoint, i.e., it is possible that $Prop_i \cap Prop_j \neq \emptyset$ for $i, j \in Agt$;
- $Val : Prop \to 2^{\mathcal{S}}$ is a valuation function mapping each propositional symbol to the set of joint states in which it is true. For each agent i, we can also obtain its individual valuation function Val_i by taking the restriction of Val onto $Prop_i$ and S_i;

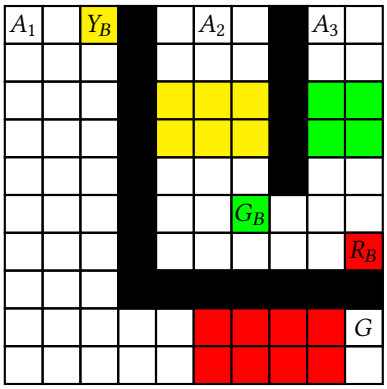


Fig. 1. CooperativeButtons domain of Neary et al. (2021).

A multi-agent environment specifies the dynamics of the low-level environment in which agents learn: the states and actions that are possible, and the probability of transitioning from a state $\boldsymbol{s}$ to a state $\boldsymbol{s'}$ on a (joint) action $\boldsymbol{a}$. A MAE is similar to a Markov Game (Boutilier 1996; Littman 1994), i.e., a Markov Decision Process where the state and action spaces are factored to account for the fact that there are multiple agents acting in the environment. Unlike Markov Games, MAEs lack a reward function, as in our setting this will be specified by a reward machine.

A joint policy $\boldsymbol{\pi} : \mathcal{S} \to \Delta(\mathcal{A})$ maps any state in a MAE $\boldsymbol{E}$ to a probability distribution over the set of joint actions. We write $\boldsymbol{\pi}(\boldsymbol{s}, \boldsymbol{a})$ to denote the probability of performing the joint action $\boldsymbol{a}$ in the joint state $\boldsymbol{s}$ according to $\boldsymbol{\pi}$.

Example 1. As an example of a MAE, consider the CooperativeButtons domain of Neary et al. (2021) shown in Figure 1. The CooperativeButtons environment is a 10x10 grid world containing walls, three buttons, a goal location and three agents A_1, A_2 and A_3. Black walls are fixed, while the yellow, green and red walls can be lowered to allow an agent to pass by pressing the appropriate coloured buttons, Y_B, G_B and R_B. The task consists in allowing agent A_1 to reach the goal location G.

Agents can move left, up, right, down or stay in the same cell. Movement actions are nondeterministic: if an agent moves in any direction, it may end up in a cell adjacent to the one it was trying to reach with probability 0.2.

The set of propositional symbols is $Prop = \{Y_B, G_B, A_2^{R_B}, A_2^{\neg R_B}, A_3^{R_B}, A_3^{\neg R_B}, R_B, Goal\}$. Propositional symbols Y_B, G_B and R_B are true if, respectively, the yellow, green and red button has been pressed, while $Goal$ is true if an agent is on the goal location G. For the red button to be pressed, we require both agents A_2 and A_3 to press it together: if they are on the button, then propositional symbols $A_2^{R_B}$ and $A_3^{R_B}$ are true, if they leave the button then propositional symbols $A_2^{\neg R_B}$ and $A_3^{\neg R_B}$ are true. Agent A_1's set of propositions is $Prop_{A_1} = \{Y_B, R_B, Goal\}$, agent A_2's is $Prop_{A_2} = \{Y_B, G_B, A_2^{R_B}, A_2^{\neg R_B}, R_B\}$ and agent A_3's is $Prop_{A_3} = \{G_B, A_3^{R_B}, A_3^{\neg R_B}, R_B\}$.

A joint state corresponds to the pair of coordinates $\langle x_j, y_j \rangle$ of each agent $A_j \in \{A_1, A_2, A_3\}$ and the set of propositions true in it.[1] Individual states for agent i contain only its coordinates and the set of propositions, from $Prop_i$, true in it.

[1]For convenience, we will omit the set of propositional symbols true in joint states, and just give them as triples of coordinates. Whenever the set of propositional symbols is needed, we will explicitly state it beforehand.

Definition 2 (Single-Agent Environment)**.** Given a MAE $\boldsymbol{E}$, the *single-agent environment* (SAE) for agent $i \in Agt$ is the tuple $E_i = \langle S_i, A_i, Pr_i, Prop_i, Val \rangle$ where:

$$Pr_i(s, a, s') := \frac{\sum_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') \in \mathcal{S}\times\mathcal{A}\times\mathcal{S} : s \in \boldsymbol{s}, a \in \boldsymbol{a}, s' \in \boldsymbol{s}'} Pr(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')}{\sum_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') \in \mathcal{S}\times\mathcal{A}\times\mathcal{S} : s \in \boldsymbol{s}, a \in \boldsymbol{a}} Pr(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')}$$

The function Pr_i is a probability measure over pairs $s \in S_i, a \in A_i$, i.e., $Pr_i(s, a, s') \geq 0$ for any $s' \in S_i$, and $\sum_{s' \in S_i} Pr_i(s, a, s') = 1$. Single-agent environments are used to define the "part" of a MAE in which an agents learns its part of a team task independently of other agents.

In order to define tasks, we use a "labelling" which specifies how the evolution of the MAE/SAE affects the truth values of its propositional symbols. As in previous work on RMs, labellings are functions defined over the transitions of a MAE/SAE, rather than their set of states. In this way, after any transition, the RMs can be updated via the label associated with the transition. Given a set of propositional symbols $Prop$, we denote with $\overline{Prop}$ the set of literals we derive from it. For a given propositional symbol $p \in Prop$, we denote with, respectively, p^+ and p^- its positive and negative literal.

Definition 3 (Labelling)**.** Given a MAE[2] $\boldsymbol{E} = \langle Agt, (S_i)_{i\in Agt}, (A_i)_{i\in Agt}, Pr, (Prop_i)_{i\in Agt}, Val \rangle$, its labelling is the, naturally induced, function $L : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \wp(\overline{Prop})$ mapping transitions in the MAE to the set of literals that hold after the transition. More precisely, given joint states $\boldsymbol{s}, \boldsymbol{s}' \in \mathcal{S}$ and joint action $\boldsymbol{a} \in \mathcal{A}$, we have that $L(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') := \{p^+ \mid \boldsymbol{s} \notin Val(p) \wedge \boldsymbol{s}' \in Val(p)\} \cup \{p^- \mid \boldsymbol{s} \in Val(p) \wedge \boldsymbol{s}' \notin Val(p)\}$. As each agent $i \in Agt$ has its own set of observable propositional symbols $Prop_i$, we can define its individual labelling $L_i : S_i \times A_i \times S_i \to \wp(\overline{Prop_i})$ by analogously taking $L_i(s_i, a_i, s_i) := \{p^+ \mid s_i \notin Val_i(p) \wedge s_i' \in Val_i(p)\} \cup \{p^- \mid s_i \in Val_i(p) \wedge s_i' \notin Val_i(p)\}$.

In this paper we focus on *postcondition labelling*, where $L(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$ is the set of literals made true in $\boldsymbol{s}'$ by executing $\boldsymbol{a}$ in $\boldsymbol{s}$. If, for a given atomic proposition $p \in Prop$, we have that neither of its literals appear in $L(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$, then it means that its truth value has not changed from $\boldsymbol{s}$ to $\boldsymbol{s}'$.

Example 1 (continued)**.** Consider the previous example MAE CooperativeButtons with propositional symbols $Prop = \{Y_B, G_B, A_2^{R_B}, A_2^{\neg R_B}, A_3^{R_B}, A_3^{\neg R_B}, R_B, Goal\}$. As an example of a label for a transition, suppose the initial joint state $\boldsymbol{s}$ is $\langle\langle 1, 0\rangle, \langle 5, 0\rangle, \langle 8, 0\rangle\rangle$ (with no propositional symbol being true) and that the joint action is $\langle \texttt{right}, \texttt{stay}, \texttt{stay} \rangle$. In this case, if agent A_1 successfully moves to the right, the next joint state will be $\langle\langle 2, 0\rangle, \langle 5, 0\rangle, \langle 8, 0\rangle\rangle$, meaning that agent A_1 has correctly pressed the yellow button. Therefore, the transition $\langle\langle 1, 0\rangle, \langle 5, 0\rangle, \langle 8, 0\rangle\rangle, \langle \texttt{right}, \texttt{stay}, \texttt{stay} \rangle, \langle\langle 2, 0\rangle, \langle 5, 0\rangle, \langle 8, 0\rangle\rangle$ will be labelled with $\{Y_B^+\}$.

A *reward machine* (RM), termed a *simple reward machine* in (Toro Icarte et al. 2022), is a Mealy machine over an alphabet Σ. Intuitively, an RM takes abstract descriptions of an event in a multi-agent environment as input, and outputs a reward.

Definition 4 (Reward Machine)**.** A reward machine is a tuple $R = \langle U, u^I, \Sigma, t, r \rangle$ where:

- U is a finite non-empty set of states;
- u^I is the initial state;
- Σ is a finite set of environment events;
- $t : U \times \Sigma \to U$ is a transition function that, for every state $u \in U$ and event $e \in \Sigma$, gives the state resulting from observing event e in state u; and
- $r : U \times \Sigma \to \mathbb{R}$ is a reward function that for every state $u \in U$ and event $e \in \Sigma$ gives the reward resulting from observing event e in state u.

[2]The labelling of a SAE is obtained in an analogous manner.

In our case, an RM can be seen as a specification of a particular task with respect to a given underlying multi-agent environment $\boldsymbol{E}$. The labelling L connects the MAE E to the RM, by providing the RM with events (i.e., the set of literals) that occur as a result of transitions in the MAE. We therefore assume that the alphabet Σ of events corresponds to sets of literals built over the finite set of propositions $Prop$ of E, i.e., $\Sigma = \wp(\overline{Prop})$.

Example 1 (continued)**.** Consider again the CooperativeButtons domain with the labelling L introduced previously. Figure 2 shows the reward machine given by Neary et al. (2021) that makes use of the labelling L to describe the task of reaching the goal location G.

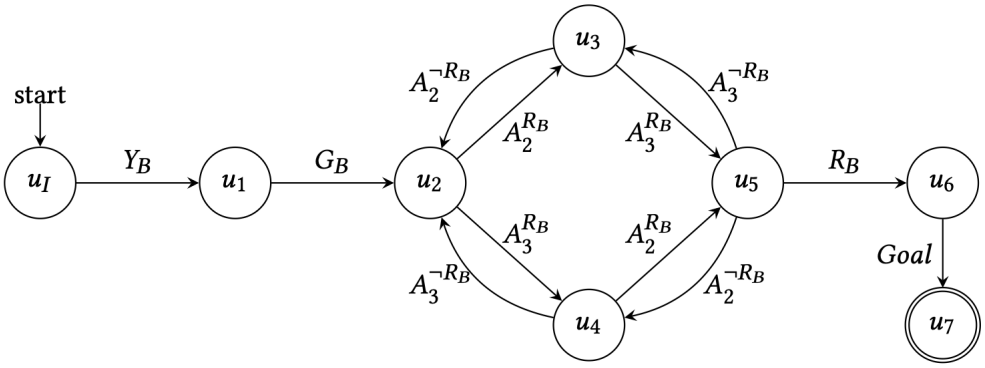


Fig. 2. Reward machine by Neary et al. (2021) that describes the task of reaching the goal location in the CooperativeButtons domain.

As agents have their own set of observable propositional symbols, we define the set of observable events of agent i as $\Sigma_i := \wp(\overline{Prop_i})$. Similarly, for a coalition $A \subseteq Agt$, we define $\Sigma_A := \wp(\bigcup_{i \in A} \overline{Prop_i})$. For a given event $e \in \Sigma$ and an agent i with its respective set of observable literals $\overline{Prop_i}$, we denote the restriction of e onto Σ_i by $e \restriction \Sigma_i$, where $e \restriction \Sigma_i = e \cap \overline{Prop_i}$. This will be used to define the "*part of*" the event e that is observable by a given subset of agents.

In what follows, we will refer to reward machines for coalitions as "*coalition reward machines*", and to reward machines for individual agents as "*individual reward machines*". Note that the two kinds of reward machines are identical from a formal point of view. However, intuitively, a coalition RM represents a coalition task, while each individual RM represents the contribution of a particular agent to the coalition task.

To formally define the *multi-agent reinforcement learning problem with reward machines*, we introduce the notion of a Markov game with a reward machine (MGRM). An MGRM is essentially a product of a multi-agent environment and a reward machine. It is similar to a Markov decision process with a reward machine (MDPRM) as defined by Toro Icarte et al. (2022), but generalised to the multi-agent setting, where the reward function is induced by a reward machine and the game is Markovian with respect to a set of states in the product.

Definition 5 (Markov Game with a Reward Machine)**.** A Markov game with a reward machine with n agents is a tuple $\boldsymbol{G} = \langle Agt, (S_i)_{i \in Agt}, (A_i)_{i \in Agt}, Pr, (Prop_i)_{i \in Agt}, Val, L, \gamma, U, u^I, \Sigma, t, r \rangle$ where:

- $Agt, S_i, A_i, Pr, (Prop_i)_{i \in Agt}, Val$ are as in Definition 1 (for all $1 \leq i \leq n$);
- $L : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \wp(\overline{Prop})$ is the labelling function of the underlying MAE as in Definition 3;
- $\gamma \in [0, 1]$ is a discount factor;
- U, u^I, Σ, t, r are as in Definition 4, with $\Sigma = \wp(\overline{Prop})$; and
- if in states $\boldsymbol{s} \in \mathcal{S}$, $u \in U$, the agents perform an action $\boldsymbol{a}$ and the MAE state changes from $\boldsymbol{s}$ to $\boldsymbol{s'}$, then $u' = t(u, L(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s'}))$ and the agents receive reward $r(u, L(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s'}))$.

As we will use a decentralised approach for training of each agent $i \in Agt$, we will also employ Markov decision processes with reward machines.

Definition 6 (Markov Decision Process with a Reward Machine)**.** A Markov decision process with a reward machine is a tuple $G = \langle S, A, Pr, Prop, Val, L, \gamma, U, u^I, \Sigma, t, r \rangle$, where:

- $S, A, Pr, Prop, Val$ are as in Definition 2;
- $L : S \times A \times S \to \wp(\overline{Prop})$ is the labelling function of the underlying SAE as in Definition 3;
- $\gamma \in [0, 1]$ is a discount factor;
- U, u^I, Σ, t, r are as in Definition 4, with $\Sigma = \wp(\overline{Prop})$; and
- if in states $s \in S$, $u \in U$, the agent performs an action a and the SAE state changes from s to s', then $u' = t(u, L(s, a, s'))$ and the agent receives reward $r(u, L(s, a, s'))$.

We define joint policies $\boldsymbol{\pi}$ as maps from pairs containing a MAE joint state $\boldsymbol{s}$ and a RM state u to probability distributions over the joint actions of the MAE, i.e., $\boldsymbol{\pi} : \mathcal{S} \times U \to \Delta(\mathcal{A})$. Analogously, an individual policy π is a map from pairs containing a SAE state s and a RM state u to probability distributions over the actions of the SAE, i.e., $\pi : S \times U \to \Delta(A)$.

Definition 7 (MGRM (MDPRM) Learning Problem)**.** For a given MGRM (MDPRM) $\boldsymbol{G}$ (G), the "*MGRM (MDPRM) learning problem*" consists in learning an optimal joint (individual) policy $\boldsymbol{\pi^\star} : \mathcal{S} \times U \to \Delta(\mathcal{A})$ ($\pi : S \times U \to \Delta(A)$) that maximises the expected discounted future reward from any joint state of the MAE (SAE) and RM state.

2.2 Alternating-Time Temporal Logic

Alternating-time Temporal Logic (ATL) (Alur et al. 2002) is a standard formalism for specifying the high-level behaviour of agents in multi-agent systems. The ATL formula plays a role similar to that of a planning goal in the synthesis of single-agent reward machines by Illanes et al. (2019) and Varricchione et al. (2024). However, the use of ATL means we can easily specify more flexible properties; for example, that the agents can bring about ψ while maintaining ϕ, or can maintain some property forever, etc. Here we define the syntax and semantics of ATL with imperfect information. We need imperfect information because we do not assume that the agents can observe all the effects of each other's actions, and it is important for decomposability that each agent bases its choice of actions only on what it can observe.

Let $Agt = \{i_1, \ldots, i_n\}$ be a set of n agents and $Prop$ denote a set of propositions. Formulas of ATL are defined by the following syntax

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle A \rangle\rangle \bigcirc \varphi \mid \langle\langle A \rangle\rangle \Box \varphi \mid \langle\langle A \rangle\rangle \varphi \, \boldsymbol{\mathcal{U}} \, \varphi$$

where $p \in Prop$ is a proposition and $A \subseteq Agt$. Here, $\langle\langle A \rangle\rangle \bigcirc \varphi$ means that a coalition A can ensure that the next state satisfies φ, $\langle\langle A \rangle\rangle \Box \varphi$ means that A has a strategy to make sure that φ is always true, and $\langle\langle A \rangle\rangle \varphi \, \boldsymbol{\mathcal{U}} \, \psi$ means that A has a strategy to eventually enforce ψ while maintaining the truth of φ before doing so.

The models of ATL are concurrent game structures. Imperfect information is modelled by indistinguishability relations between states, one for each agent. The resulting concurrent game structures are called "*epistemic concurrent game structures*" (ECGS).

Definition 8 (Epistemic Concurrent Game Structure)**.** An epistemic concurrent game structure is a tuple $M = \langle Agt, Q, (Prop_i)_{i\in Agt}, v, (\sim_i)_{i\in Agt}, Act, d, \delta\rangle$ where:

- Agt is a non-empty finite set of n agents;
- Q is a non-empty finite set of states;
- $Prop_i$ is the finite set of propositional symbols "observable" by agent i, $Prop := \bigcup_{i\in Agt} Prop_i$ is the entire set of observable propositions;
- $v : Prop \to \wp(Q)$ is a valuation which associates each proposition in $Prop$ with a subset of states where it is true. We assume that two distinct states cannot have the same valuation, i.e., they cannot make the same subset of propositional symbols true;
- $\sim_i \subseteq Q \times Q$ for each $i \in Agt$ is an equivalence relation. For each state $q \in Q$, we denote with $[q]_i$ the equivalence class of q for $\sim_i$;
- Act is a non-empty finite set of actions;
- $d : Q \times Agt \to \wp(Act) \setminus \{\emptyset\}$ is a deterministic function which assigns to each $q \in Q$ a non-empty set of actions available to each agent $i \in Agt$, with the constraint that $q_1 \sim_i q_2$ implies that $d(q_1, i) = d(q_2, i)$. We denote joint actions by all agents in Agt available at q by $D(q) = d(q, i_1) \times \cdots \times d(q, i_n)$;
- $\delta : (q, \sigma) \mapsto Q$ is a function that gives for every $q \in Q$ and joint action $\sigma \in D(q)$ the state resulting from executing σ in q. We write $q \xrightarrow{\sigma} q'$ to abbreviate $\delta(q, \sigma) = q'$.

Example 1 (continued)**.** Let us now formalize the CooperativeButtons as an ECGS. The set of agents is $Agt = \{A_1, A_2, A_3\}$. The set of states is $Q = \{q_\emptyset, q_{\{Y_B\}}, q_{\{Y_B,G_B\}}, q_{\{Y_B,G_B,R_B\}}, q_{\{Y_B,G_B,R_B,Goal\}}\}$. For the observable propositions, we have that $Prop_{A_1} = \{Y_B, R_B, Goal\}$, $Prop_{A_2} = \{Y_B, G_B, R_B\}$, and $Prop_{A_3} = \{G_B, R_B\}$. A state q_X is in $v(p)$ for some proposition $p \in Prop$ if and only if $p \in X$. For each agent, $\sim_i$ is the relationship where, for any pair of states $q, q' \in Q$, $q \sim_i q'$ if, for any $p \in Prop_i$, $q \in V(p)$ if and only if $q' \in V(p)$. The set of actions is $A = \{$`press_yellow`, `press_green`, `press_red`, `to_goal`, `nil`$\}$. `nil` $\in d(q, i)$ for any state $q \in Q$ and agent $i \in Agt$. For the other actions: `press_yellow` $\in d(q_\emptyset, A_1)$, `to_goal` $\in d(q_{\{Y_B,G_B,R_B\}}, A_1)$, `press_green` $\in d(q_{\{Y_B\}}, A_2)$, `press_red` $\in d(q_{\{Y_B,G_B\}}, A_2)$, and `press_red` $\in d(q_{\{Y_B,G_B\}}, A_3)$. Finally, for the transition function: $\delta(q_\emptyset, \sigma) = q_{\{Y_B\}}$ if the action of agent A_1 in σ is `press_yellow`, $\delta(q_{\{Y_B\}}, \sigma) = q_{\{Y_B,G_B\}}$ if the action of agent A_2 in σ is `press_green`, $\delta(q_{\{Y_B,G_B\}}, \sigma) = q_{\{Y_B,G_B,R_B\}}$ if the actions of agents A_2 and A_3 in σ are `press_red`, and $\delta(q_{\{Y_B,G_B,R_B\}}, \sigma) = q_{\{Y_B,G_B,R_B,Goal\}}$ if the action of agent A_1 in σ is `to_goal`.

Given an ECGS M, we denote the set of all infinite sequences of states (computations) by Q^ω. For a computation $\lambda = q_0 q_1 \ldots \in Q^\omega$, we use the notation $\lambda[j] = q_j$. We use variables α, β for individual actions and σ for joint actions.

Given an ECGS M and a state $q \in Q$, a *joint action by a coalition* $A \subseteq Agt$ is a tuple $\sigma_A = (\sigma_i)_{i\in A}$ such that $\sigma_i \in d(q, i)$. The set of all joint actions for A at state q is denoted by $D_A(q)$. Given a joint action by the grand coalition $\sigma \in D(q)$, σ_A denotes the joint action executed by A: $\sigma_A = (\sigma_i)_{i\in A}$. The set of all possible outcomes of a joint action $\sigma_A \in D_A(q)$ at state q is denoted by $out(q, \sigma_A) = \{q' \in Q \mid \exists \sigma' \in D(q) : \sigma_A = \sigma'_A \wedge q \xrightarrow{\sigma'} q'\}$. Given an ECGS M, a *strategy for a coalition* $A \subseteq Agt$ is a mapping $F_A : Q \to Act^{|A|}$ such that, for every $q \in Q$, $F_A(q) \in D_A(q)$. A computation $\lambda \in Q^\omega$ is consistent with a strategy F_A if and only if, for all $j \geq 0$, $\lambda[j+1] \in out(\lambda[j], F_A(\lambda[j]))$. We denote by $out(q, F_A)$ the set of all consistent computations λ of F_A that start from q. Given a coalition strategy F_A for some coalition A, for any agent $i \in A$, we denote by F_i the restriction of strategy F_A to agent i.

As we consider postcondition labelling, we also "label" actions $\sigma_i \in Act$ (for any agent $i \in Agt$) with two subsets of propositional symbols $P^+_{\sigma_i}$ and $P^-_{\sigma_i}$ which are, respectively, the sets of propositional symbols true and false after σ_i is executed. Obviously, for any action σ_i, we must have that $P^+_{\sigma_i} \cap P^-_{\sigma_i} = \emptyset$. For a joint action $\sigma = (\sigma_i)_{i\in Agt}$, $P^+_\sigma = \bigcup_{i\in Agt} P^+_{\sigma_i}$ and $P^-_\sigma = \bigcup_{i\in Agt} P^-_{\sigma_i}$. We assume that the ECGS contains only joint actions such

that for any distinct pair of agents $i, i' \in Agt$, $P^+_{\sigma_i} \cap P^-_{\sigma_{i'}} = \emptyset$. Hence, transitions are defined so that $q \xrightarrow{\sigma} q'$ with: (i) $q' \in v(p)$ for any $p \in P^+_\sigma$, (ii) $q' \notin v(p)$ for any $p \in P^-_\sigma$, and (iii) for all $p \in Prop$ such that $p \notin P^+_\sigma \cup P^-_\sigma$ we have that $q' \in v(p)$ if and only if $q \in v(p)$.

For a given agent action σ_i, $\overline{post(\sigma_i)} := \{p \mid p \in P^+(\sigma_i)\} \cup \{\neg p \mid p \in P^-(\sigma_i)\}$. Analogously, we define $\overline{post(\sigma)}$ for any joint action σ.

Some strategies are unrealistic in that they require agents to select different actions in two states that they cannot distinguish. For this reason, we will only consider so-called "uniform" strategies:

Definition 9 (Uniform strategy)**.** A strategy for agent $i \in Agt$, F_i, is *uniform* if and only if it specifies the same choices for indistinguishable situations: if $q \sim_i q'$ then $F_i(q) = F_i(q')$. A strategy for a coalition A is uniform if and only if it is uniform for each $i \in A$.

Example 1 (continued)**.** We now give a uniform strategy F for the coalition of agents in the CooperativeButtons task:

- $F_{A_1}(q_\emptyset) =$ `press_yellow`, $F_{A_1}(q_{\{Y_B,G_B,R_B\}}) =$ `to_goal`, and $F_{A_1}(q) =$ `nil` for any other $q \in Q$;
- $F_{A_2}(q_{\{Y_B\}}) =$ `press_green`, $F_{A_2}(q_{\{Y_B,G_B\}}) =$ `press_red`, and $F_{A_2}(q) =$ `nil` for any other $q \in Q$;
- $F_{A_3}(q_{\{Y_B,G_B\}}) =$ `press_red`, and $F_{A_2}(q) =$ `nil` for any other $q \in Q$.

The strategy F is clearly uniform, as it satisfies the condition outlined in Definition 9 for each agent.

Strong uniformity requires, in addition, that in order for a formula of the form $\langle\langle A\rangle\rangle\varphi$ to be true in a state q, the same uniform strategy by A should ensure φ from all the states indistinguishable from q by A, i.e., for all states $q' \in [q]_A$, where $[q]_A$ is the equivalence class of q for $\sim_A := \bigcap_{i \in A} \sim_i$, we have that the uniform strategy that enforces φ from q also does from all $q' \in [q]_A$.

Given an ECGS M, a state q of M, the truth of an ATL formula φ with respect to M and q is defined inductively on the structure of φ as follows:

- $M, q \models p$ if and only if $q \in v(p)$;
- $M, q \models \top$ always;
- $M, q \models \neg\phi$ if and only if $M, q \not\models \phi$;
- $M, q \models \phi \vee \psi$ if and only if $M, q \models \phi$ or $M, q \models \psi$;
- $M, q \models \langle\langle A\rangle\rangle \bigcirc \psi$ if and only if there exists a uniform strategy F_A such that for all $q' \sim_A q$, for all $\lambda \in out(q', F_A)$: $M, \lambda[1] \models \psi$;
- $M, q \models \langle\langle A\rangle\rangle \Box\phi$ if and only if there exists a uniform strategy F_A such that for all $q' \sim_A q$, for all $\lambda \in out(q', F_A)$ and $j \geq 0$: $M, \lambda[j] \models \phi$;
- $M, q \models \langle\langle A\rangle\rangle \phi \,\mathcal{U}\, \psi$ if and only if there exists a uniform strategy F_A such that for all $q' \sim_A q$, for all $\lambda \in out(q', F_A)$, $\exists j \geq 0$: $M, \lambda[j] \models \psi$ and $M, \lambda[k] \models \phi$ for all $k \in \{0, \ldots, j-1\}$.

We specify tasks for coalitions in a fragment of ATL, which we call "Flat Cooperative ATL" (FC-ATL. As we focus on fully cooperative multi-agent reinforcement learning, where the set of agents Agt cooperate to reach a common goal (and actions by the grand coalition are deterministic), FC-ATL contains only formulas of the form $\langle\langle A\rangle\rangle\varphi$ such that: (i) the coalition of agents A is always the grand coalition Agt, and (ii) φ contains no other coalitional modalities (and hence, temporal modalities). As $\langle\langle A\rangle\rangle\varphi$ represents the (single) team task the agents Agt will be trained to achieve, it suffices to consider the ATL fragment with only one coalitional modality and no disjunction or conjunction of coalitional formulas. (A conjunction or disjunction of coalitional formulas represents two different learning tasks.) Note that, even with a single coalitional modality we can still encode reachability and safety (including invariants) tasks.

The syntax of FC-ATL is given by:

$$\varphi_0 ::= p \mid \top \mid \neg\varphi_0 \mid \varphi_0 \vee \varphi_0$$
$$\varphi_1 ::= \langle\langle Agt \rangle\rangle \bigcirc \varphi_0 \mid \langle\langle Agt \rangle\rangle \Box \varphi_0 \mid \langle\langle Agt \rangle\rangle \varphi_0 \, \mathcal{U} \, \varphi_0$$

where only formulas generated by φ_1 are used to encode tasks for the grand coalition of agents. Given the syntax of FC-ATL, in the remainder of the paper we use A to refer to the grand coalition Agt.

For example, the task from the CooperativeButtons domain in Example 1 can be specified as:

$$\langle\langle Agt \rangle\rangle \top \, \mathcal{U} \, Goal$$

This formula is true if the grand coalition Agt has a uniform strategy to reach a state that satisfies $Goal$.

We note that we could have used a different formalism to specify tasks; for example, a fragment of Linear-time Temporal Logic (LTL) or of Computation Tree Logic (CTL) (Clarke and Emerson 1981). We chose ATL over LTL and CTL because its ECGS-based semantics allow to express in a natural way the individual abilities of agents and what they can observe. Moreover, by model checking an FC-ATL formula, we can obtain the specific sequences of high-level ECGS actions that each agent should perform in order to achieve the given task. However, approaches based on synthesis for (fragments of) CTL and LTL could also have been used.[3]

We now define the *witness* of a (uniform) strategy for an ATL coalitional modality formula, given an ECGS and an initial state in it. Witnesses are used to define both coalition reward machines and individual reward machines, as we will see in the following section.

Definition 10 (Witness)**.** Given an ECGS $M = \langle Agt, Q, (Prop_i)_{i \in Agt}, v, (\sim_i)_{i \in Agt}, Act, d, \delta \rangle$, a state $q \in Q$ and a coalition formula $\langle\langle A \rangle\rangle \varphi$ such that $M, q \models \langle\langle A \rangle\rangle \varphi$ using a uniform strategy F_A for A, a *witness* $W(q, F_A)$ for F_A to achieve φ starting from q is defined as follows, depending on the form of φ:

- $\langle\langle A \rangle\rangle \varphi = \langle\langle A \rangle\rangle \bigcirc \psi$: the witness is a tree of height 1 where the root is q, all outgoing edges are labelled with the action $F_A(q)$, and the leaves are states $q' \in out(q, F_A(q))$. Note that, in the case where $A = Agt$, there is a single leaf node $q \xrightarrow{F_A(q)} q'$.
- $\langle\langle A \rangle\rangle \varphi = \langle\langle A \rangle\rangle \phi \, \mathcal{U} \, \psi$: the witness is a tree where the root is q and each branch corresponds to the set of prefixes of λ in $out(q, F_A)$, each prefix ends in the first state where ψ is true, and edges out of each q' are labelled by $F_A(q')$. In the case where $A = Agt$, the witness tree has a single branch $q, F_A(q), q_1, F_A(q_1), \ldots, F_A(q_n), q_{n+1}$ of length $n + 1$ corresponding to the prefix of the single element λ of $out(q, F_A)$, where $\lambda[n] = q_{n+1}$ is the first state on λ that satisfies ψ.
- $\langle\langle A \rangle\rangle \varphi = \langle\langle A \rangle\rangle \Box \phi$: the witness is a graph where the initial vertex is q and containing each computation $\lambda \in out(q, F_A)$, ending in the first occurrence of a repeated state q_n. Edges out of each state q' are labelled by $F_A(q')$. In the graph, for such repeated states q_n, there is a transition labelled by $F_A(q_n)$ to states in $out(q_n, F_A(q_n))$ that occur earlier in the graph (since q_n itself occurs earlier in the graph, its successors by $F_A(q_n)$ are there as well). In the case were $A = Agt$, the witness graph contains the ECGS states $q, q_1, \ldots, q_n$ from the unique computation $q, q_1, \ldots, q_n, (q'_1, \ldots, q'_m, q_n)^*$ (where $q_n, q'_1, \ldots, q'_m, q_n$ is the first cycle encountered on the computation).

In Figure 3 we show the witness obtained from the uniform strategy for the CooperativeButtons task.

For formulas of the form $\langle\langle A \rangle\rangle \phi \, \mathcal{U} \, \psi$ or $\langle\langle A \rangle\rangle \bigcirc \psi$, we will refer to the states in the witness in which ψ becomes true as the "final states", and use the notation q^f to refer to them. We note that a witness is just a representation of a (uniform) strategy as a graph: it therefore does not affect how the strategy itself is defined or obtained.

[3]We discuss the availability of tools which can perform ATL/LTL/CTL synthesis in Section 4.1.

Fig. 3. Witness for the CooperativeButtons task.

2.3 Safe Abstractions and Learnable Tasks

We conclude this section by defining "*safe* abstractions" and "*learnable* tasks" with respect to a given MAE (SAE). Intuitively, safe abstractions are abstractions that correctly represent how, via high-level events, an MAE (SAE) evolves; whereas learnable tasks are tasks expressed in a safe abstraction for which there exists a policy that achieves the task in the MAE (SAE).

In general, there are many possible abstractions of a given multi-agent environment $\boldsymbol{E}$ and its corresponding SAEs. For agents to be able to learn how to perform a task using a reward machine synthesised using an ECGS M, M must satisfy a number of properties. Fix a multi-agent environment $\boldsymbol{E}$. We say that an ECGS state $q \in Q$ "*abstracts*" an MAE joint state $\boldsymbol{s} \in \mathcal{S}$ if and only if the sets of propositional symbols $\{p \mid q \in v(p)\}$ and $\{p \mid \boldsymbol{s} \in Val(p)\}$ are equal, i.e., q and $\boldsymbol{s}$ satisfy the same propositional symbols.[4] Note that this does not imply a bijection between the two: in distinct joint states of an MAE the same set of propositional symbols can be true, which is not the case in an ECGS.

Definition 11 (Safe abstraction)**.** An ECGS M is a "*safe abstraction*" of a MAE $\boldsymbol{E}$ if, for any transition $q \xrightarrow{\sigma} q'$ in M, there is a joint policy $\boldsymbol{\pi}_{q \xrightarrow{\sigma} q'}$ that "implements" the transition in $\boldsymbol{E}$, i.e., if agents follow joint policy $\boldsymbol{\pi}$ from any joint state $\boldsymbol{s}$ in $\boldsymbol{E}$ that is abstracted by q, then $\boldsymbol{E}$ will eventually transition to some joint state $\boldsymbol{s}'$ that is abstracted by q'.

The notion of a safe abstraction is similar to standard techniques for producing existential abstractions of multi-agent systems, such as the one given by Cohen et al. (2009). It is also related to the notion of safe state abstraction in reinforcement learning introduced, for example, by Kokel et al. (2021); however, we do not consider rewards in our safe abstractions. In practice, safe abstractions ensure that for any high-level transition there is a low-level policy that allows the agents to "perform" the high-level transition in the low-level MAE. This means that the policy is able to enforce the same "changes" in the propositional symbols (i.e., the same "events") that are caused by the high-level transition. This is crucial, as otherwise it might be the case that a high-level strategy for some task (computed from the ECGS) might not be executable in the low-level MAE if the strategy includes a high-level transition for which there is no policy implementing the transition in the low-level MAE. Given this, we can define when a task is "learnable" by a coalition of agents A.

Definition 12 (Learnable tasks)**.** Fix a MAE $\boldsymbol{E}$ and an ECGS M that is a safe abstraction of $\boldsymbol{E}$. A task, expressed by an FC-ATL formula $\langle\langle A \rangle\rangle \varphi$, is "*learnable*" by coalition A if and only if there is some ECGS state $q \in Q$ from which there is a uniform strategy for A, i.e., $M, q \models \langle\langle A \rangle\rangle \varphi$.

In the rest of the paper, we will implicitly assume that the ECGS we are given is a safe abstraction of the input MAE, and that the task expressed by the input FC-ATL formula is learnable with respect to such ECGS. This assumption ensures that, if there exists a strategy to achieve the task in the input ECGS, then there also exists a joint policy to achieve it in the MAE. The existence of a useful (small enough to admit model-checking, and detailed enough to give rise to a non-trivial reward machine) safe abstraction is not guaranteed. However, generating a useful safe abstraction is not a more demanding task than generating an abstract planning domain

[4] If the sets of propositional symbols of the MGRM and the ECGS are not the same, then $\boldsymbol{s}$ and q must satisfy the same propositional symbols up to the function that connects the two sets.

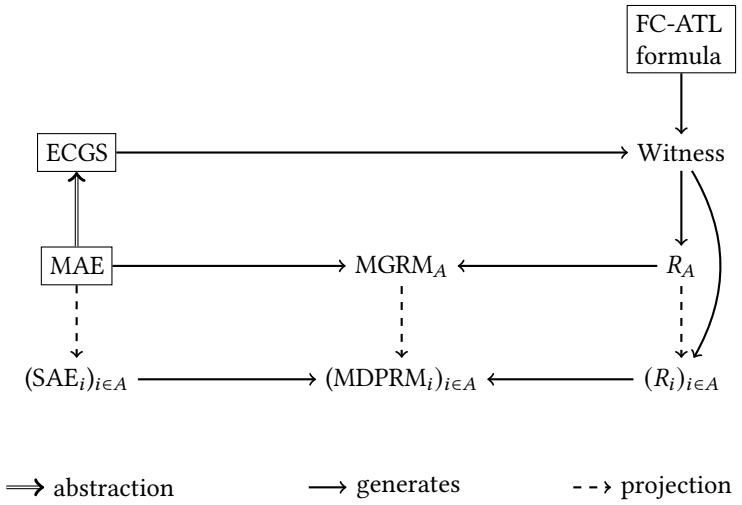


Fig. 4. Overview and relationship between the objects used in our approach. Objects in a box are given as input by the user, who we assume has access to an ECGS which is a safe abstraction of the MAE.

corresponding to an MDP as in (Kokel et al. 2021) or a set of high-level options implementable by RL policies as in (Sutton et al. 1999), given the similarities of these approaches with our notion of safe abstractions.

3 Synthesising MGRMs

In this section, we show how, given a multi-agent environment $\boldsymbol{E}$, a labelling L for it, and an FC-ATL formula $\langle\langle Agt\rangle\rangle\varphi$ specifying the team task, we can synthesise a MGRM for any set of agents $B \subseteq Agt$ (MDPRM for singletons of agents) from a witness for $\langle\langle Agt\rangle\rangle\varphi$. While for generality B may be any subset of Agt, we focus on the cases where $B = Agt$ and where B is a singleton, i.e., on generating the coalition MGRM and the MDPRM for each agent. The aim of this construction is to obtain the reward machine for the set of agents B. If B is the grand coalition, the MGRM can obtained by just "appending" the reward machine to the (labelled) multi-agent environment that is given in input. If B is a singleton $\{i\}$, its MDPRM is obtained by deriving the labelled single-agent environment E_i of agent i from the multi-agent environment and then appending the reward machine to it. In Figure 4, we provide a high-level overview of the objects that are used in our approach and how they are related to each other. The MAE, the ECGS and the FC-ATL formula are all given as input, whereas the rest is computed in our approach. We would like to stress the dynamics of the low-level environments (MAE, SAE, MGRM, and MDPRM) are hidden from the agents, which means that it is not possible for them to compute a policy to perform the task given only a witness of a high-level strategy (which is obtained in the ECGS) for it. Finally, we observe that so long as the FC-ATL formula is true given the ECGS and the initial ECGS state, we can generate a witness and thus the reward machines. In case the FC-ATL formula is false, then it is not possible to carry out our synthesis procedure.

3.1 Synthesising Reward Machines

Fix a multi-agent environment $\boldsymbol{E} = \langle Agt, S_1, \ldots, S_n, A_1, \ldots, A_n, Pr\rangle$. Given an ECGS $M = \langle Agt, Q, (Prop_i)_{i\in Agt}, v, (\sim_i)_{i\in Agt}, Act, d, \delta\rangle$ which is safe for $\boldsymbol{E}$, we can use an ATL model checker to synthesise a witness for a uniform

strategy that achieves the task encoded by the FC-ATL formula $\langle\langle A\rangle\rangle\varphi$. From the witness, we can synthesise both the coalition RM R_A and each of the individual RMs R_i, for each agent $i \in A$.

Notice that the witness and the reward machine derived from it are defined in terms of high-level actions from the ECGS M and are not directly executable in the MAE $\boldsymbol{E}$ (or in the individual agents' environments). However, the synthesised RM can be used to guide an agent in learning which low-level actions in $\boldsymbol{E}$ should be performed to accomplish each step in the RM.

The construction of a reward machine from a witness $W(q, F_A)$ is given in Construction 1. First we explain the construction informally. The set of states of the coalition RM is the same set of the witness, whereas the set of states of agent i's individual RM is the set of equivalence classes, with respect to the indistinguishability relation $\sim_i$, of the states in the witness. In both cases, we include an extra "error" state. For the edge labels, we simply replace the high-level actions that label such edges in the original witness with their corresponding postconditions. For the coalition RM, transition between states are the same as in the witness. Instead, for an individual agent's RM, transitions between states, i.e., equivalence classes, are defined depending on the transitions between states in such equivalence classes. Given two equivalence classes, if there is a state in the former that transitions to a state in the latter given some postcondition, then there will be a transition between the two classes labelled with the same postcondition. The reward machines for $\langle\langle A\rangle\rangle\phi\,\boldsymbol{\mathcal{U}}\,\psi$ and $\langle\langle A\rangle\rangle\Box\phi$ transit to the error state on events corresponding to a violation of the invariant property ϕ (we assume that $\neg\phi$ is always observable by any agent in A). The error state has a self-loop and no transitions to other states of the reward machine. As a minor complication, the state corresponding to the second last state of the witness for $\langle\langle A\rangle\rangle\phi\,\boldsymbol{\mathcal{U}}\,\psi$ transits to the error state only on events that *both* violate ϕ *and* are not postconditions of the last action in the witness (do not achieve ψ).

Notice that, although the construction can be applied to any subcoalition $B \subseteq A$, we are only interested in generating the RMs for the entire coalition A and for each individual agent $i \in A$.

Construction 1 (Construction of a reward machine)**.** Given the ECGS $M = \langle Agt, Q, (Prop_i)_{i\in Agt}, v, (\sim_i)_{i\in Agt}, Act, d, \delta\rangle$ and a witness $W(q, F_A)$, we construct a reward machine $R_B = \langle U_B, u_B^I, \Sigma_B, t_B, r_B\rangle$ for $B \subseteq A$ as follows:

- $U_B = Q(W(q, F_A))/{\sim_B} \cup \{u_{err}\}$ is the set of equivalence classes of states in $W(q, F_A)$ with respect to the indistinguishability relation of B, plus u_{err};
- $u_B^I = [q]_B$;
- $\Sigma_B = \wp(\bigcup_{i\in B} \overline{Prop_i})$. If $\langle\langle A\rangle\rangle\varphi$ is of the form $\langle\langle A\rangle\rangle\Box\phi$ or $\langle\langle A\rangle\rangle\phi\,\boldsymbol{\mathcal{U}}\,\psi$, then we also add all the events e that entail $\neg\phi$ to Σ_B;
- $t_B(u, e) = u'$ if and only if there are q_1, q_2 in the set of nodes of $W(q, F_A)$ such that $u = [q_1]_B$ and $u' = [q_2]_B$ with q_1 connected to q_2 through an edge labelled with $F_A(q_1)$, and there is a joint action $\sigma \in D(q_1)$ with $q_1 \xrightarrow{\sigma} q_2$, $\sigma_A = F_A(q_1)$ such that $e = \overline{post(\sigma)} \upharpoonright \Sigma_B$. If $\langle\langle A\rangle\rangle\varphi = \Box\phi$ or $\langle\langle A\rangle\rangle\varphi = \langle\langle A\rangle\rangle\phi\,\boldsymbol{\mathcal{U}}\,\psi$ and $e \models \neg\phi$ then $u' = u_{err}$ unless $e = \overline{post(\sigma)} \upharpoonright \Sigma_B$ for an action σ leading to a final state q^f in $W(q, F_A)$. In the latter case the RM transitions to $u' = [q^f]$ in the witness $W(q, F_A)$ (as ϕ needs to hold only strictly before ψ holds);
- For the definition of r_B, there are three different cases:
  (1) $r_B(u, e) = 1$ if and only if $\langle\langle A\rangle\rangle\varphi = \langle\langle A\rangle\rangle\bigcirc\psi$, $u = [q]$, and $e = \overline{post(F_A(q))} \upharpoonright \Sigma_B$, or $\langle\langle A\rangle\rangle\varphi = \langle\langle A\rangle\rangle\phi\,\boldsymbol{\mathcal{U}}\,\psi$ and $e = post(\sigma) \upharpoonright \Sigma_B$ for an action σ leading to a final state q^f in $W(q, F_A)$.
  (2) $r_B(u, e) = -1$ if and only if $\langle\langle A\rangle\rangle\varphi = \langle\langle A\rangle\rangle\Box\phi$, $u \neq u_{err}$ and e entails $\neg\phi$;
  (3) For all other[5] (u, e), $r_B(u, e) = 0$.

[5]Note that, in case the task is specified by a formula of the form $\langle\langle A\rangle\rangle\phi\,\boldsymbol{\mathcal{U}}\,\psi$ and the formula ϕ is violated, the agents in B are given a reward of 0, while they receive a reward of -1 when the task is encoded by a formula of the form $\langle\langle A\rangle\rangle\Box\phi$. While counterintuitive, this is crucial in order to prove Theorem 2 below. Note that, even though the agents do not receive a negative reward for violating the invariant ϕ, it is still the case that their optimal (joint) policy must achieve the task: the only way for them to receive a non-zero reward is to satisfy ψ while maintaining the invariant ϕ true.

We now show that the RMs synthesised by our approach are "correct" decompositions in the sense of Neary et al. (2021), i.e., that under certain sequences of events, both the individual and coalition RMs will give the same reward. First, we define the product of RMs. Before giving it, if $\langle\langle A\rangle\rangle\varphi$ is either of the form $\langle\langle A\rangle\rangle\bigcirc\psi$ or $\langle\langle A\rangle\rangle\phi\,\mathcal{U}\,\psi$, then for each individual agent's i RM $R_i = \langle U_i, u_i^I, \Sigma_i, t_i, r_i\rangle$ we also define the set of "*achievement*" states U_i^f, i.e., the set containing all and only the states that are reached when the agent is given a reward of 1: $U_i^f := \{u \in U_i \mid \exists u' \in U_i \exists e \in \Sigma_i : t_i(u', e) = u \wedge r_i(u', e) = 1\}$. Notice that since formulas of the form $\langle\langle A\rangle\rangle\bigcirc\psi$ and $\langle\langle A\rangle\rangle\phi\,\mathcal{U}\,\psi$ are satisfied when ψ is true, any transition that leads to a state in R_i representing states in the ECGS where ϕ and ψ are respectively true, will be such that the reward given for that transition will be 1, as the agent will be able to recognize that the goal has been achieved. For convenience, we will assume that all agents' RMs loop in achievement states.

Definition 13 (Product of RMs)**.** Fix RMs for agents $i_1, \ldots, i_n$, where the RM for agent i_k is $R_k = \langle U_k, u_k^I, \Sigma_k, t_k, r_k\rangle$, based on a common witness $W(q, F_A)$ with the event sets Σ_k obtained from the set of observables $Prop_{i_k}$. We define the product RM $R_\otimes = \langle U_\otimes, u_\otimes^I, \Sigma_\otimes, t_\otimes, r_\otimes\rangle$ of the RMs, denoted by $R_1 \otimes \ldots \otimes R_n$ as follows:

- $U_\otimes := U_1 \times \ldots \times U_n$;
- $u_\otimes^I := (u_1^I, \ldots, u_n^I)$;
- $\Sigma_\otimes$ is the event set obtained from the set of observables $Prop_{i_1} \cup \ldots \cup Prop_{i_n}$;
- $t_\otimes((u_1, \ldots, u_n), e) := (t_1(u_1, e \upharpoonright \Sigma_1), \ldots, t_n(u_n, e \upharpoonright \Sigma_n)$. Notice that if $\langle\langle A\rangle\rangle\varphi = \langle\langle A\rangle\rangle\Box\phi$ or $\langle\langle A\rangle\rangle\varphi = \langle\langle A\rangle\rangle\phi\,\mathcal{U}\,\psi$ and e entails $\neg\phi$ then $t_\otimes((u_1, \ldots, u_n), e) = (u_{err}, \ldots, u_{err})$ (respective error states in R_j);
- If $\langle\langle A\rangle\rangle\varphi$ is of the form $\langle\langle A\rangle\rangle\bigcirc\psi$ or $\langle\langle A\rangle\rangle\phi\,\mathcal{U}\,\psi$: let $U_1^f, \ldots, U_n^f$ be the set of achievement states of the agents' RMs. Then, we set $r_\otimes((u_1, \ldots, u_n), e) := 1$ if there is at least one $u_i \notin U_i^f$ and, for all $u_i \notin U_i^f$, we have that $t_i(u_i, e \upharpoonright \Sigma_i) \in U_i^f$. Intuitively, we give a reward of 1 as soon as all the agents enter one of their achievement states;
- $r_\otimes((u_1, \ldots, u_n), e) := -1$ if $\langle\langle A\rangle\rangle\varphi = \langle\langle A\rangle\rangle\Box\psi$ and e entails $\neg\psi$;
- $r_\otimes((u_1, \ldots, u_n), e) := 0$ otherwise.

Let A be a coalition of agents, F_A a strategy for the coalition and q the initial ECGS state. We define the set of *compatible event sequences* $\Xi_{F_A,q}$ for strategy F_A and initial state q as the set of event sequences that is observed by coalition A while following strategy F_A, i.e. $\Xi_{F_A,q} := \{\xi \mid \exists\lambda \in out(q, F_A) \forall i \in \mathbb{N} \exists\sigma \in D(\lambda[i]) : \sigma_A = F_A(\lambda[i]) \wedge \xi[i] = post(\sigma) \upharpoonright \Sigma_A\}$.

Finally, before formalizing and proving our 'correctness' claim, we state and prove a lemma about the states of the coalition RM and of the product RM. Fix a coalition of agents $A = \{i_1, \ldots, i_n\}$, its coalition RM $R_A = \langle U_A, u_A^I, \Sigma_A, t_A, r_A\rangle$ and a set of observables Σ_A for the coalition. For each agent $i \in A$, let $R_i = \langle U_i, u_i^I, \Sigma_i, t_i, r_i\rangle$ be its individual RM, with $\Sigma_i \subseteq \Sigma_A$ be the set of observables of such agent.

Lemma 1. *There is a (partial) homomorphism g, with respect to the transition functions of the RMs, from states of the product RM $R_\otimes$ to those of the coalition RM R_A.*

Proof. For the coalition RM, states are equivalence classes of $\sim_A$, whereas in the product RM they are tuples of equivalence classes, containing precisely one class for each agent $i \in A$. We show that any state $u \in U_A$ can be obtained by some other state $u' \in U_\otimes$. This mapping is induced by observing that an equivalence class of $\sim_A$ is the intersection of a tuple containing one equivalence class per $\sim_i$. To prove this, we show that for an arbitrary state $q \in Q$, we have that $[q]_A = \bigcap_{i\in A}[q]_i$. First, we show that $[q]_A \subseteq \bigcap_{i\in A}[q]_i$: let q' be an arbitrary $q' \in [q]_A$. Since $q' \in [q]_A$ we have that q' agrees with q on the propositions observable by A. Notice that the set of observable propositions $Prop_A$ is exactly the union of the set of observable propositions $\bigcup_{i\in A} Prop_i$ Thus, let $i \in A$ be arbitrary: it is easy to see that $q' \in [q]_i$, because as q' agrees on the propositional symbols in $Prop_A$ with q, it must also agree on the propositional symbols in $Prop_i$ with q, since $Prop_a \subseteq Prop_A$.Thus, $q' \in \bigcap_{i\in A}[q]_i$. For the

other inclusion, the proof is similar: suppose that $q' \in \bigcap_{i \in A} [q]_a$. Then, it must agree with q on the propositional symbols in $\bigcup_{i \in A} Prop_i$. Since $Prop_A = \bigcup_{i \in A} Prop_i$, it follows that $q' \in [q]_A$. We denote by $g : U_\otimes \rightharpoonup U_A$ the partial function such that $g((u_{i_1}, \ldots, u_{i_n})) := \bigcap_{i \in A} u_i$ for any $(u_{i_1}, \ldots, u_{i_n}) \in U_{i_1} \times \ldots \times U_{i_n}$, if such intersection is non-empty, and is otherwise undefined. Now, we treat states from the product RM that contain the error state u_{err}. In order to do this, it suffices to notice that if a tuple contains an error state u_{err}, then all states in the tuple must be u_{err} for g to be defined over it, as otherwise the intersection is empty since all agents are able to observe whether the invariant has been falsified.

We can now prove that g is a homomorphism between $U_\otimes$ and U_A with respect to the transition functions $t_\otimes$ and t_A. We need to show that that $g(t_\otimes(u^I_\otimes, \xi[j])) = t_A(g(u^I_\otimes), \xi[j])$ for any index j in an arbitrary event sequence ξ. We prove this claim by induction on the number of steps j. We first consider the base case, i.e., $j = 0$. Notice that at $j = 0$, $u^j = u^I_\otimes$, thus $g(u^0) = u^I_A$. After observing event $\xi[0]$, the product RM moves to the new state $u^1 = (u^1_{i_1}, \ldots, u^1_{i_n})$, where each u^1_i is the state to which the RM of agent i transitions from u^I_i after observing $\xi[0]$, while the coalition RM moves to the new state u^1_A. We want to show that $g(u^1) = u^1_A$. Since the RM of agent i moves according to the observable propositions in $Prop_i$, notice that the new RM state u^1_i is the equivalence class (with respect to $\sim_i$) of the states of the ECGS which: (i) agree with the states in the equivalence class represented by the RM state u^0_i on propositional symbols in $Prop_i$ for which no literal appears in $\xi[0]$, and (ii) have the truth values of propositional symbols in $Prop_i$ corresponding to the literals appearing in $\xi[0]$ (i.e., if $p^+ \in \xi[0]$, then p must be true in the states in u^1_i, and if $p^- \in \xi[0]$, then p must be false in such states). By taking the intersection of all the new states of the individual RMs, it is clear that the previous points apply to the whole set of observables $Prop_A$. Also, the intersection cannot be empty as two agents cannot observe contradictory literals. Now, notice that the coalition RM transitions in the same way as the RM of any agent i, but considering $Prop_A$ instead of $Prop_i$. Therefore, we have indeed that $g(u^1) = u^1_A$. For the inductive case $j = n + 1$, the argument is exactly the same, albeit we need to apply the inductive hypothesis to know that $g(u^n) = u^n_A$. □

Now, we can give the claim and prove it.

Theorem 1. *Fix a strategy F_A for the formula $\langle\langle A \rangle\rangle \varphi$ for the coalition of agents $A = \{i_1, \ldots, i_n\}$ and an initial ECGS state q. Let $R_A = \langle U_A, u^I_A, \Sigma_A, t_A, r_A \rangle$ be the coalition RM and $R_\otimes = \langle U_\otimes, u^I_\otimes, \Sigma_\otimes, t_\otimes, r_\otimes \rangle$ be the product RM $R_{i_1} \times \ldots \times R_{i_n}$. Given a compatible event sequence $\xi \in \Xi_{F_A,q}$, then for any step $j \in |\xi|$ we have that $r_\otimes(u^j, \xi[j]) = r_A(g(u^j), \xi[j])$, where $u^j = (u^j_{i_1}, \ldots, u^j_{i_n})$ is the j-th state reached by the product RM following the event sequence ξ and $g : U_\otimes \rightharpoonup U_A$ is defined as in the proof of Lemma 1.*

Proof. The proof essentially amounts to showing that g not only preserves transitions, but also output rewards. We consider two different cases, depending on the structure of the formula that encodes the task. If $\langle\langle A \rangle\rangle \varphi$ is of the form $\langle\langle A \rangle\rangle \bigcirc \psi$ or $\langle\langle A \rangle\rangle \phi \, \mathcal{U} \, \psi$, then $R_\otimes$ can either output 0 or 1 at any step j. In case it outputs 0, either some agent has not reached one of its achievement states in its RM or the invariant ϕ, for $\langle\langle A \rangle\rangle \phi \, \mathcal{U} \, \psi$ formulas, has been falsified (notice in this latter case how every agent is able to observe that $\neg\phi$ has become true). Thus, let u^j be the state of the product RM $R_\otimes$ at step j and let $\xi[j]$ be the event observed at such step. By Lemma 1, we know that u^j_A, the state of the coalition RM R_A at step j, is exactly $g(u^j)$. Thus, assume by way of contradiction that $r_A(u^j_A, \xi[j]) \neq 0$: this can only mean that $r_A(u^j_A, \xi[j]) = 1$, i.e., that the whole coalition is able to observe that ψ has become true. However, this contradicts our starting assumption, thus showing the claim. Instead, if $R_\otimes$ outputs 1, then this must be due to the fact that from state u^j and thanks to event $\xi[j]$ all agents in A will be able to observe that the following state will be a goal one, i.e., a state where ψ is true. Hence, their respective RMs will all be in one of the achievement states. From here, we can prove the claim with a similar argument to the one we gave for the other case.

Finally, if $\langle\langle A \rangle\rangle \varphi$ is of the form $\langle\langle A \rangle\rangle \Box \psi$, then we can show the claim by considering the same argument we gave before in case the reward was 0. Recall that for these formulas, the RMs give a reward of 0 so long as the

invariant is true, and -1 when it is falsified. Thus, it suffices to consider exactly that argument, as it already covers such two cases for formulas of the form $\langle\langle A\rangle\rangle \phi \, \mathcal{U} \, \psi$.

□

Theorem 1 thus establishes that, with respect to output rewards, the coalition RM and the RM obtained from the product of all the individual RMs are indistinguishable. Therefore, we immediately obtain a result analogous to Theorem 1 from Neary et al. (2021), without the need to check whether such RMs are bisimilar.

Note that in Theorem 1 we consider the case in which the RM initial state is fixed. To construct a reward machine that works from an arbitrary initial state, we construct a reward machine $W(q, F_A)$ for every $q \in Q$ and then connect them by adding an extra state u_0 that has a transition labelled by $e \in \Sigma_A$ where e describes each equivalence class with respect to $\sim_A$. Since the truth definition for $\langle\langle A\rangle\rangle \varphi$ we use corresponds to strong uniformity, there is a uniform strategy F_A that works from all states indistinguishable by $\sim_A$, and all different classes are distinguishable by some event $e \in \Sigma_A$. The proof of Theorem 1 carries over to the case when the initial state is not fixed.

To conclude, we state a theorem relating the future undiscounted rewards obtained by a coalition and the individual rewards obtained by the agents in the coalition. Consider a coalition $A = \{i_1, \ldots, i_n\}$, a witness $W(q, \langle\langle A\rangle\rangle \varphi)$ for some formula $\langle\langle A\rangle\rangle \varphi$ from state q of an ECGS, a joint state $\boldsymbol{s}$ of a MAE and an arbitrary joint policy $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$ for the same MAE. For the coalition RM R_A built from $W(q, \langle\langle A\rangle\rangle \varphi)$, we denote by $V_A^{\boldsymbol{\pi}}(\boldsymbol{s})$ the sum of expected future undiscounted rewards produced by R_A, given all agents follow their policy as specified by $\boldsymbol{\pi}$ from $\boldsymbol{s}$ of the MGRM and the initial state u_A^I of R_A. Analogously, for the individual RM R_{i_j} built from the same witness $W(q, \langle\langle A\rangle\rangle \varphi)$, we denote by $V_{i_k}^{\boldsymbol{\pi}}(\boldsymbol{s})$ the sum of expected future undiscounted rewards produced by R_{i_j} under the same assumptions. Moreover, recall that if $\langle\langle A\rangle\rangle \varphi$ is of the form $\langle\langle A\rangle\rangle \bigcirc \psi$ or $\langle\langle A\rangle\rangle \phi \, \mathcal{U} \, \psi$, then any RM synthesised from a witness for a strategy for the formula can give a reward of only 0 or 1. Similarly, any RM for a formula of the form $\langle\langle A\rangle\rangle \Box \phi$ can give a reward of only 0 or -1.

From the observations above and the definition of V, regardless of the policy $\boldsymbol{\pi}$, the environment state $\boldsymbol{s}$ and of whether we are considering a coalition or a single agent, it follows that if the formula is of the form $\langle\langle A\rangle\rangle \bigcirc \psi$ or $\langle\langle A\rangle\rangle \phi \, \mathcal{U} \, \psi$, then $V_B^{\boldsymbol{\pi}}(\boldsymbol{s})$ is exactly the probability that the (sub)coalition $B \subseteq A$ is able to complete the (sub)task encoded by their RM. On the other hand, if the formula is of the form $\langle\langle A\rangle\rangle \Box \phi$, then $-V_B^{\boldsymbol{\pi}}(\boldsymbol{s})$ is equal to the probability of failing to complete the (sub)task.

Theorem 2. *If $\langle\langle A\rangle\rangle \varphi$ is of the form $\langle\langle A\rangle\rangle \bigcirc \psi$ or $\langle\langle A\rangle\rangle \phi \, \mathcal{U} \, \psi$, then*

$$\max\{0, V_{i_1}^{\boldsymbol{\pi}}(\boldsymbol{s}) + \ldots + V_{i_n}^{\boldsymbol{\pi}}(\boldsymbol{s}) - (n-1)\} \leq V_A^{\boldsymbol{\pi}}(\boldsymbol{s}) \leq \min\{V_{i_1}^{\boldsymbol{\pi}}(\boldsymbol{s}), \ldots, V_{i_n}^{\boldsymbol{\pi}}(\boldsymbol{s})\}$$

If $\langle\langle A\rangle\rangle \varphi$ is of the form $\langle\langle A\rangle\rangle \Box \psi$, then

$$\max\{-V_{i_1}^{\boldsymbol{\pi}}(\boldsymbol{s}), \ldots, -V_{i_n}^{\boldsymbol{\pi}}(\boldsymbol{s})\} \leq -V_A^{\boldsymbol{\pi}}(\boldsymbol{s}) \leq \min\{1, -V_{i_1}^{\boldsymbol{\pi}}(\boldsymbol{s}) - \ldots - V_{i_n}^{\boldsymbol{\pi}}(\boldsymbol{s})\}.$$

Proof. If $\langle\langle A\rangle\rangle \varphi$ is of the form $\langle\langle A\rangle\rangle \bigcirc \psi$ or $\langle\langle A\rangle\rangle \phi \, \mathcal{U} \, \psi$, then notice all RMs will give a reward of 1 when the task they encode has been completed and 0 otherwise. Thus, we can see the expected sum of undiscounted future rewards $V_A^{\boldsymbol{\pi}}(\boldsymbol{s})$ as the probability that coalition A completes the task assuming every agent is following the joint policy $\boldsymbol{\pi}$. Notice that the same holds as well for any other agent $i_k \in A$. Moreover, by Theorem 1 we know that for any compatible event sequence $\xi \in \Xi_{F_A, q}$ at any timestep the rewards given by the individual RMs and the coalition one are equal. Thus, the probability that the whole coalition A completes the task described by R_A is equal to the probability that all agents complete the individual tasks described by their respective RMs, assuming that all agents follow the joint policy $\boldsymbol{\pi}$. Then, by applying directly the Frechét inequality for logical conjunctions we get the first part of the claim.

On the other hand, if $\langle\langle A\rangle\rangle\varphi$ is of the form $\langle\langle A\rangle\rangle\Box\psi$, it suffices to notice that the opposite of the expected sum of undiscounted future rewards (both for coalition A and for any agent $i_k \in A$) is the probability of failing the task. This is because in this case the RMs will output -1 when the task they describe is failed and 0 otherwise. Now, observe that as soon as one agent fails the task, then all other agents will fail the task as the invariant has been falsified. Since every agent can observe whether the invariant has been falsified, their RMs will move to the error state and give a reward of -1. By Theorem 1 then also the coalition RM will give a reward of -1 at such timestep. On the other hand, if the coalition RM gives a reward of -1, then all the individual RMs will also give a reward of -1, by the same reasoning. Therefore, the probability that coalition A fails the task is equivalent to the probability that at least a single agent in A fails the task. Thus, by applying the Frechét inequalities for logical disjunctions we get the second part of the claim. □

Intuitively, Theorem 2 bounds the probability that a coalition completes a task (or not, depending on the formula type), assuming all agents follow their policy specified by $\boldsymbol{\pi}$. For formulas of the form $\langle\langle A\rangle\rangle\bigcirc\psi$ or $\langle\langle A\rangle\rangle\phi\,\mathcal{U}\,\psi$, if all agents $i \in A$ are able to (eventually) complete their subtask, then coalition A is able to (eventually) complete the joint task: if all the individual $V^{\boldsymbol{\pi}}_{i_k}(\boldsymbol{s}) = 1$, then $\max\{0, V^{\boldsymbol{\pi}}_{i_1}(\boldsymbol{s}) + \ldots + V^{\boldsymbol{\pi}}_{i_n}(\boldsymbol{s}) - (n-1)\} = 1$ and so $V^{\boldsymbol{\pi}}_{A}(\boldsymbol{s}) = 1$. Similarly, for formulas of the form $\langle\langle A\rangle\rangle\Box\psi$. If no agent violates ψ, then the coalition will not violate ψ: if each individual $-V^{\boldsymbol{\pi}}_{i_k}(\boldsymbol{s}) = 0$, then $\min\{1, -V^{\boldsymbol{\pi}}_{i_1}(\boldsymbol{s}) - \ldots - V^{\boldsymbol{\pi}}_{i_n}(\boldsymbol{s}))\} = 0$ and so $V^{\boldsymbol{\pi}}_{A}(\boldsymbol{s}) = 0$.

Theorem 2 is analogous to Theorem 2 from Neary et al. (2021), but our claim is made with respect to FC-ATL formulae, instead of RMs. Both results establish the same idea, i.e., that it makes no difference to the probability of task completion whether agents are trained individually or together.

4 Evaluation

In this section we present an evaluation of automatically synthesised reward machines. We show how strategies for team tasks generated by the model checker MCMAS (Lomuscio, Qu, et al. 2017) can be used to produce an RM for each agent in a team of agents, and present results from four benchmarks: CooperativeButtons and Rendezvous from (Neary et al. 2021), OfficeWorld from (Toro Icarte et al. 2022), and WaterWorld from (Gupta et al. 2017).[6]

For all experiments, we present the results in plots where lines represent median performance, and the shaded areas show the 25^{th} and 75^{th} percentiles. For the experiments comparing against the RMs of Neary et al. (2021) and the ones in WaterWorld, performance is measured as the number of steps necessary to complete the task, whereas in the experiment where we compare against the RM of Toro Icarte et al. (2022) the performance measure is the average reward per step, as in their experiments.

4.1 Synthesising Reward Machines

Before presenting the experimental results we motivate the use MCMAS, the model checker we have used to obtain witnesses from FC-ATL formulas, and briefly discuss alternative tools which could have been used.

MCMAS[7] is a model checker which is able to produce witnesses for formulas in several logics, including ATLK and CTLK (Lomuscio, Qu, et al. 2017). This ability to model imperfect information scenarios and to generate uniform strategies is crucial for our approach. The generation of uniform strategies (and witnesses) by MCMAS is enabled using `-uniform` flag. However, it is important to note that the ATL semantics used by MCMAS with the `-uniform` flag differs from the standard semantics for ATL with imperfect information and memoryless strategies. When the `-uniform` flag is enabled, MCMAS first creates all possible *uniform protocols* for each agent

[6]Our code can be found at `https://github.com/giovannivarr/MARM-JAIR-code`; the code of Neary et al. (2021) at `github.com/cyrusneary/rm-cooperative-marl`, and the code of Toro Icarte et al. (2022) at `github.com/RodrigoToroIcarte/reward_machines`.

[7]We used MCMAS version `1.3.0` to generate the witnesses in the experiments below.

(Lomuscio and Raimondi 2006). A protocol is uniform if, in all states indistinguishable by an agent, the agent is restricted to a single action from the set of actions available in that state for the rest of the run. For each set of uniform protocols for each agent in the coalition, MCMAS generates a uniform strategy, and checks whether the *entire* ATL formula is satisfied under this strategy, i.e., the same uniform strategy is used for all modalities in the formula. If the formula is satisfied, MCMAS halts and outputs true; otherwise it moves to the next uniform strategy. If no uniform coalition strategy (i.e., none of the possible profiles containing a uniform protocol per each agent) satisfy the formula, MCMAS outputs false. This differs from the standard uniform semantics for ATL, where each nested coalitional modality may be associated with a different (uniform) strategy. Nevertheless, the following observation is, as we will see, almost immediate.

Observation 1. With MCMAS's `-uniform` flag enabled, the semantics of FC-ATL and the standard semantics of ATL with imperfect information and memoryless strategies are equivalent.

This follows from the fact that in FC-ATL formulas contain exactly one coalitional modality for the grand coalition. Thus FC-ATL formulas are true if and only if there is a single uniform strategy for the grand coalition which makes the entire formula true. Hence, using MCMAS with the `-uniform` flag enabled will correctly produce a witness for a uniform strategy for an FC-ATL formula encoding a coalition's task.

We conclude this section with a brief discussion of alternative tools. STV (Kurpiewski et al. 2024) and VITAMIN (Ferrando and Malvone 2025) are model checkers which support model checking of ATL formulas in a setting with imperfect information. However, VITAMIN does not currently support witness generation, and at the time when the work on this paper started (2023), the actively maintained STV version supported only singleton coalitions. MCK is a model checker which is able to check formulas in several temporal logics (including CTL and LTL) and which also provides support for imperfect information through the inclusion of knowledge modalities (Gammie and Meyden 2004). Moreover, MCK can produce counterexamples for LTL formulas of the form $\bigcirc\phi$ and $\Box\phi$ (constrained to formulas where ϕ can contain knowledge modalities but not temporal ones). By synthesising a counterexample for LTL formulas of the form $\bigcirc\neg\phi$ and $\Box\neg\phi$, MCK can be used to obtain witnesses for tasks encoded by ATL formulas of the form $\langle\langle A\rangle\rangle\bigcirc\psi$ (where $\psi \equiv \neg\psi$) and $\langle\langle A\rangle\rangle\top\,\mathcal{U}\,\phi$. However, it is not possible to obtain a witness for formulas of the form $\langle\langle A\rangle\rangle\Box\phi$ or $\langle\langle A\rangle\rangle\phi\,\mathcal{U}\,\psi$ using MCK, as these formulas are not equivalent to any formula for which MCK can produce a counterexample. As such, MCK can only generate witnesses for a fragment of FC-ATL. There are also several tools for CTL/CTL* synthesis. For CTL, these include (Klenze et al. 2016), (De Angelis et al. 2012), and CTLSAT, an implementation of the original CTL synthesis algorithm proposed in (Clarke and Emerson 1981);[8] and for CTL* synthesis (Bloem et al. 2017), (Khalimov and Bloem 2017), and (Khalimov 2018). Finally, both Strix (Luttenberger et al. 2020; Meyer et al. 2018) and Syft (S. Zhu et al. 2017) can synthesise strategies for LTL/LTL$_f$ specifications. However, all these synthesis tools lack direct support for incomplete information and the actions of individual agents without significant modification or extension. In (Tabajara and Vardi 2020) an approach to LTL synthesis under partial information is presented; however, to the best of our knowledge, no implementation is publicly available. In summary, MCMAS provides us with exactly what we need: a way to generate strategies for tasks encoded in FC-ATL and accounting for partial information. Nevertheless, we stress that our approach is agnostic to the tool used to model check FC-ATL.

4.2 CooperativeButtons & Rendezvous

We first compared the performance of DQPRM when using our automatically synthesised RMs and the RMs by Neary et al. (2021) in both the CooperativeButtons and Rendezvous environments. The CooperativeButtons environment was introduced in Section 2 (Figure 1). The Rendezvous environment is a 10x10 gridworld in which the task of the agents is to first rendezvous at a common location and then for each agent to reach their own goal.

[8] CTLSAT's implementation can be found at `https://github.com/nicolaprezza/CTLSAT`.

For the Rendezvous task we present results for teams of 10 and 25 agents. The 10-agent case allows comparison with the same experiment performed by Neary et al. (2021). The 25-agent case was performed to determine the scalability of our approach, from the strategy generation using MCMAS to the agents' training using DQPRM.

We used the same experimental setup of Neary et al. (2021) for these two tasks. For action selection, agents used softmax exploration with a constant temperature of $\tau = 0.2$. The discount factor was $\gamma = 0.9$ and the learning rate $\alpha = 0.8$. As in (Neary et al. 2021), for both tasks each experiment consisted of 10 episodes. For the CooperativeButtons task each episode consisted of 250,000 training steps, while for the Rendezvous task each episode consisted of 150,000 training steps. For both tasks a test was run every 1,000 training steps to evaluate the agents, with every test running for at most 1,000 steps (if the task was not achieved in 1,000 steps it was considered failed and the agent received no reward). It was necessary to slightly modify the original code used by Neary et al. (2021) to work with our automatically synthesised RMs. We note that this did not affect the results for the Neary et al. (2021) RM, as the code we introduced was used only when training agents with our RMs. Both the CooperativeButtons and Rendezvous experiments were run on a machine with an Apple M1 CPU and 16 GB RAM.

The results are shown in Figures 5, 6, and 7. As it can be seen, in the CooperativeButtons and 10-agents Rendezvous tasks, the performance of our automatically synthesised RMs is comparable to that of the RMs of Neary et al. (2021) in terms of sample efficiency, suggesting that, at least for these scenarios, automatically synthesized RMs allow the agents to learn the task. The results for the 25 agent Rendezvous task are shown in Figure 7. While the agents are still able to learn to achieve the task in a number of steps similar to the one required to achieve the task by 10 agents (as learning is decentralised), generation of the coalition RM can be a bottleneck, as witness generation using MCMAS required 23 hours. However, we note that approaches in the cooperative MARL literature usually do not consider larger numbers of agents. For example, a recent survey by Cui et al. (2022) notes that the number of agents in cooperative MARL typically does not exceed 10 (see, e.g., (Papoudakis et al. 2021)). An exception is the work of Ellis et al. (2023) that benchmarks cooperative MARL algorithms with coalitions of up to 23 agents in StarCraft domains.

4.3 OfficeWorld

In the OfficeWorld scenario, we compare our automatically synthesised RM to that of Toro Icarte et al. (2022) for the same task. The OfficeWorld environment is a 12x9 gridworld (see Figure 8), in which the objective is to collect coffee and mail and bring it to the office. The environment contains some decorations which the agent must not break while performing the task. Thus, the agent must learn how to navigate the world in order to collect the resources and then bring them to the office while avoiding decorations. We present results from this environment to: (i) show how our approach can also handle tasks where an invariant must be maintained, and (ii), how it can be used to synthesise RMs also in the single-agent case.

We used the same experimental setup as the one of Toro Icarte et al. (2022), i.e., the exploration rate $\epsilon = 0.1$, the discount factor $\gamma = 0.9$, and the learning rate $\alpha = 0.5$. We trained the agents using two algorithms introduced by Toro Icarte et al. (2022): the cross-product baseline (QL) and the algorithm exploiting counterfactual experiences (CRM). Agents were trained for 60 episodes, each lasting 100,000 training steps. In the OfficeWorld scenario, we were able to use the code and RM of Toro Icarte et al. (2022) without modification, by simply adding a .txt file encoding our automatically synthesised RM. Due to compatibility requirements with the code of Toro Icarte et al. (2022), the OfficeWorld experiment was run on a machine equipped with a 2.3 GHz quad-core Intel Core i5 CPU and an 8 GB RAM.

The results are shown in Figures 9 and 11. As it can be seen, the performance of our automatically synthesised RMs is again comparable to that of the RM used by Toro Icarte et al. (2022). Indeed, the agent trained with our automatically synthesised RM converges faster to the optimal reward per step compared to the agent trained

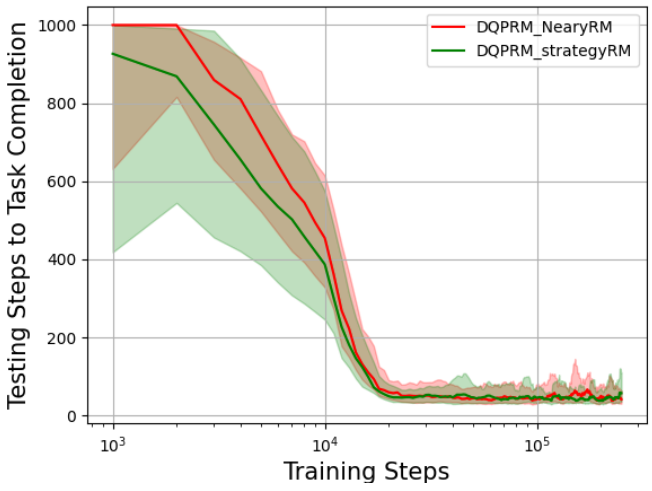


Fig. 5. Three-agents CooperativeButtons task (Neary et al. 2021). Timeout is set to 1,000 timesteps, at which point the task is considered failed.

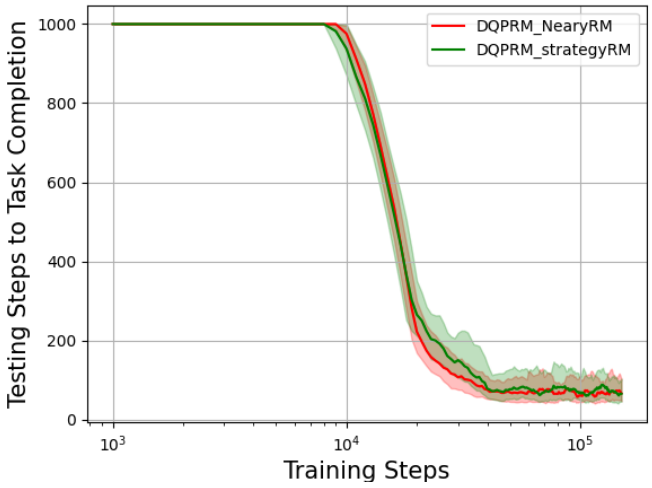


Fig. 6. Ten-agents Rendezvous task (Neary et al. 2021). Timeout is set to 1,000 timesteps, at which point the task is considered failed.

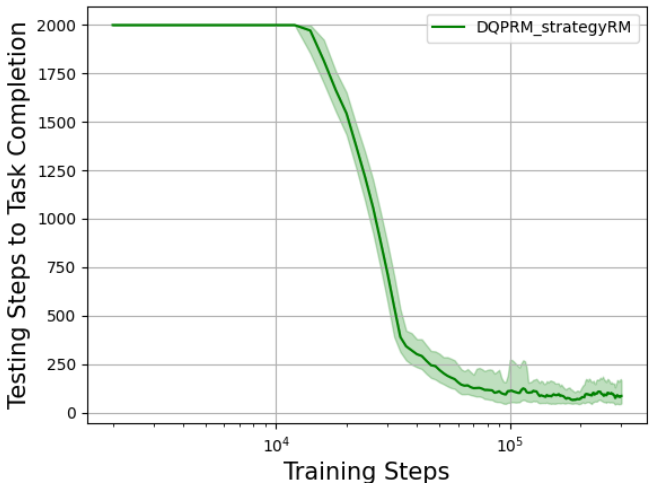


Fig. 7. Twenty-five-agents Rendezvous task (Neary et al. 2021) Timeout is set to 2,000 timesteps, at which the task is considered failed.

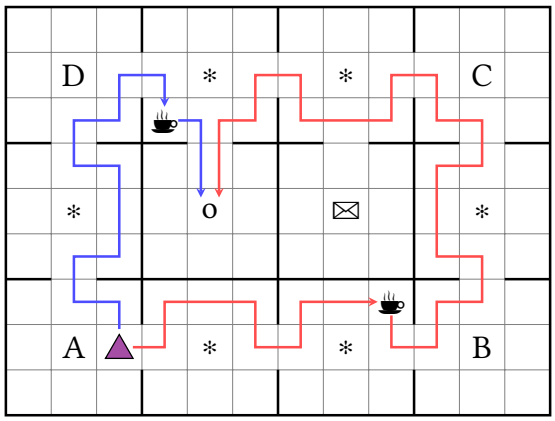


Fig. 8. OfficeWorld rendition from (Toro Icarte et al. 2022).

with the RM of Toro Icarte et al. We believe this is due to the synthesised RM being a "sub-RM" of the RM in (Toro Icarte et al. 2022) (see Figure 10). As the RM used by Toro Icarte et al. (2022) is more "permissive" (in the sense of (Varricchione et al. 2024)) than the synthesised RM, we speculate that the Toro Icarte et al. (2022) RM may result in policies achieving better rewards if it is more convenient for the agent to, e.g., first grab coffee and

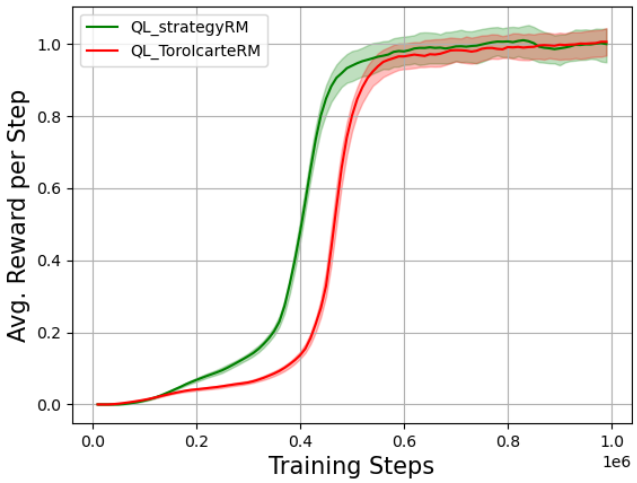
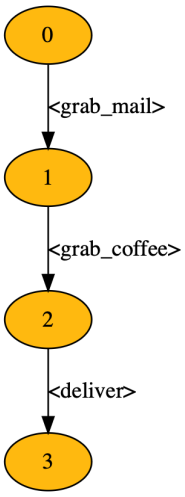
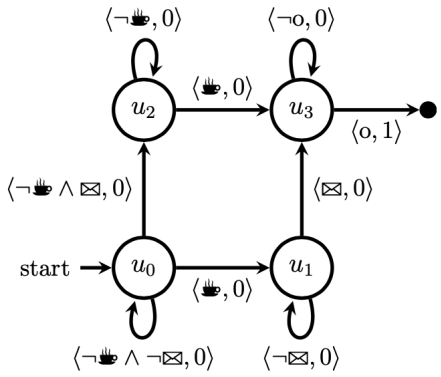


Fig. 9. Single-agent OfficeWorld task (Toro Icarte et al. 2022), agents trained with Q-learning.



(a) Witness for the OfficeWorld task; the RM is obtained by labelling the transitions with the corresponding post-conditions.



(b) RM of Toro Icarte et al. (2022) for the OfficeWorld task.

Fig. 10. Comparison between the witness used to automatically synthesise our RM and the RM used by Toro Icarte et al. (2022) for the OfficeWorld task.

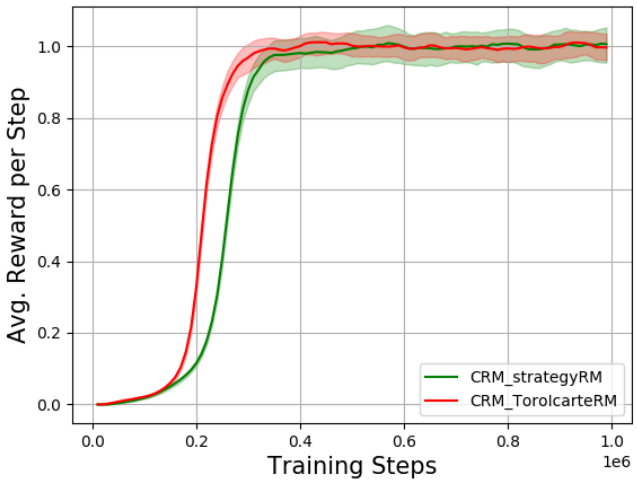


Fig. 11. Single-agent OfficeWorld task (Toro Icarte et al. 2022), agents trained with CRM.

then mail (note how in our RM the agent must first grab mail and then coffee). A more permissive RM can also be beneficial when using algorithms specifically tailored to exploit the structure of reward machines. For example, Figure 11 shows the performance of agents trained using the counterfactual RM (CRM) algorithm (Toro Icarte et al. 2022). As it can be seen, the RM of Toro Icarte et al. (2022) converges faster than the synthesised RM when using CRM. We believe this is because, at each step, CRM generates 3 experiences when using the synthesised RM and 4 experiences for the Toro Icarte et al. (2022) RM.

4.4 WaterWorld

Finally, we consider the WaterWorld environment, to show how our automaticall synthesised reward machines can also be used in a continuous domain. WaterWorld is a continuous domain where agents (called "*pursuers*") have to coordinate to capture evaders.[9] In our case, we used two pursuers A_1 and A_2, and three evaders F_1, F_2, and F_3. Pursuers and evaders are circles that move in a 2D continuous delimited space (see Figure 12 for a visualisation of the environment). Pursuers can change their speed and direction, whereas evaders cannot: when the environment is initialized, each evader is initialized with a direction vector and a fixed speed. Each pursuer has a set of evenly spread sensors that allows it to perceive the environment; each sensor reports the relative position (with respect to the pursuer) and the velocity (represented as a two-dimensional vector of real numbers) of the evaders it has perceived. In our experiments, the pursuers have 60 sensors and their range is large enough to perceive all evaders from any position of the space. A pursuer "*captures*" an evader whenever it touches the evader.

We consider two tasks of increasing difficulty requiring different levels of coordination in individual agent tasks and the joint task. In the first (simple) task agent A_1 had to capture agent F_1, then agent A_2 had to capture evader F_2, and finally both agents A_1 and A_2 had to capture evader F_3 at the same time. Figure 13 shows the witness from which the coalition RM for this task is obtained. In the second (complex) task agents A_1 and A_2 had

[9] WaterWorld was also used by Toro Icarte et al. (2022), although in a single-agent setting.

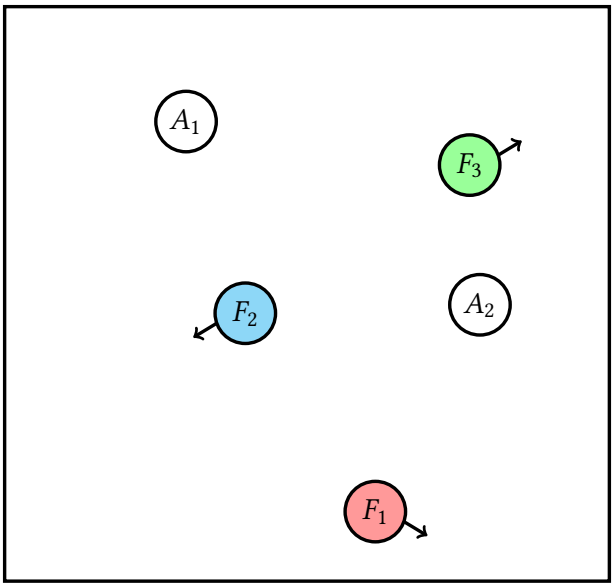


Fig. 12. WaterWorld rendition (positions of pursuers and evaders are set randomly at the start of each episode).

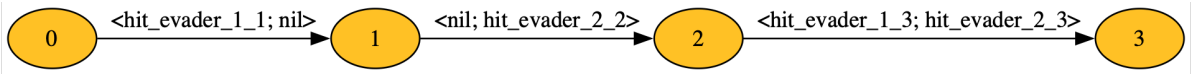


Fig. 13. Witness for WaterWorld's "*simple*" task. In the transition labels, the first number is the pursuer's and the second the evader's.

to respectively capture evaders F_1 and F_3, then they had to respectively capture evaders F_3 and F_1, and finally (as in the first task) they both had to capture evader F_2 at the same time.

We compared a pair of pursuer agents trained using our decentralised approach against a "baseline" pair which were trained using a centralised approach without a reward machine but where the rewards obtained by the agents were the same as those given by the coalition RM. We used the WaterWorld implementation from PettingZoo (Terry et al. 2021). The agents were trained using Proximal Policy Optimization (PPO) (Schulman et al. 2017) in both experiments. For the PPO implementation, we used the RLlib library (Liang et al. 2018) with the following parameters: the discount factor $\gamma = 0.99$, the generalized advantage estimation parameter $\lambda = 0.99$, the PPO clip parameter was set to 0.3 in the simple task and 0.1 in the complex task, the coefficient of the value function loss was set to 0.7, and the learning rate is initialized at $\alpha = 10^{-5}$ and linearly decayed to $\alpha = 10^{-9}$ in the first 3,000,000 training steps (after which it remained constant). These values were selected using hyperparameter tuning. RLlib counts a timestep in multi-agent environments every time an agent takes an action. Therefore, in order to not give an advantage to the decentralised approach (where there is only one pursuer agent in the environment which moves at each timestep) over the baseline (where each agent moves at consecutive timesteps), the length of training and of the episodes when training agents individually is halved compared to when they are trained with the centralised baseline. In the decentralised approach, the two agents were each trained for

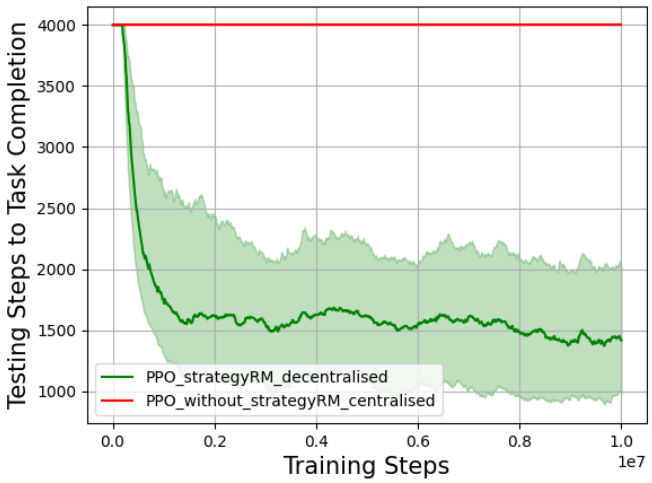


Fig. 14. Two-agents WaterWorld "*simple*" task. Timeout is set to 4,000 timesteps, at which point the task is considered failed.

10,000,000 timesteps; training was divided in episodes which lasted at most 2,000 timesteps in the simple task and 5,000 timesteps in the complex task. For the centralised baseline, training lasted 20,000,000 steps; training was divided in episodes which lasted at most 4,000 timesteps in the simple task and 10,000 timesteps in the complex task. After reaching these limits, in both tasks the environment was reset, the agents were granted no reward, and they had to restart the task. Every 10,000 training steps, we evaluate the policies learned (so far) by the agents by running 10 test episodes. Each of these 10 test episodes is associated to a distinct seed, which is used to initialise the episode during every test. Since the agents act together in the environment during the tests, the test episodes had the same limits on the number of timesteps of the training episodes for the centralised baseline. The WaterWorld experiment was run on a machine equipped with two 3.0 GHz 16-core AMD 7313 CPUs and a 1024 GB RAM.

The results are shown in Figures 14 and 15. As it can be seen, the agents trained using the automatically synthesised reward machines obtained better results than the centralised baseline. Indeed, the baseline agents are not able to achieve the task at any point during training. However, in comparison to the CooperativeButtons and Rendezvous domains, the results for the the agents trained with a synthesised RM show higher variance in terms of steps needed to achieve the task during the test episodes. In the simple task, there is a difference of approximately 500 steps between the median and the 25^{th} and 75^{th} percentiles, and for the complex task, the difference is approximately 1,000 steps. We hypothesise that this is due to the complexity and randomness of the environment, as WaterWorld is a continuous domain (both in the state and action spaces) unlike the other domains we have considered.

4.5 Summary of Results

The results indicate that, at least for the environments and tasks considered, our automatically synthesised RMs are comparable to RMs given in the literature in terms of sample efficiency. The 25 agent Rendezvous experiment also suggests that our approach is scalable to the numbers of the agents typically considered in the multi-agent

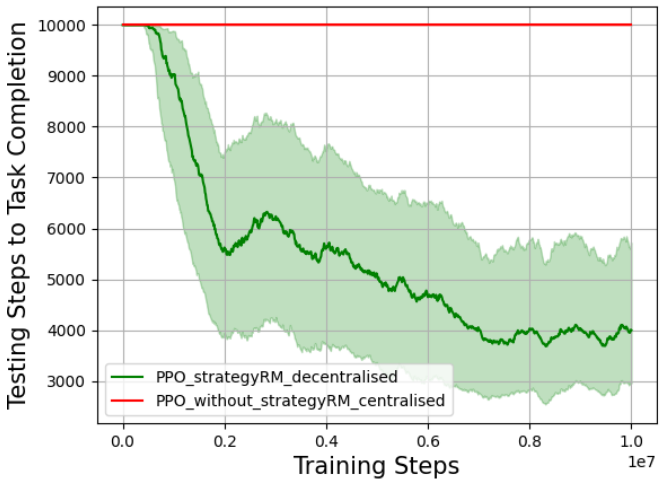


Fig. 15. Two-agents WaterWorld "*complex*" task. Timeout is set to 10,000 timesteps, at which point the task is considered failed.

reinforcement learning literature, and the WaterWorld experiment shows that our approach can be applied in continuous domains with very large state spaces, where baseline MARL approaches are unable to learn the task. However, the results for OfficeWorld suggest that witness generation techniques which are able to generate more "permissive" strategies (Varricchione et al. 2024) may be beneficial in exploiting RL algorithms tailored for reward machines.

5 Related Work

There is a large literature on the problem of non-stationarity in MARL (Hernandez-Leal, Kaisers, et al. 2019; Hernandez-Leal, Kartal, et al. 2019; Silva et al. 2018; Zhang et al. 2021). Some approaches address the problem by training each agent individually. For example, Tan (1993) introduced independent Q-learning (IQL), where each agent learns its own Q-table by treating other agents as part of the environment. Others, such as Ghavamzadeh et al. (2006) and Yang et al. (2020), adopt a hierarchical approach, where a task is decomposed so that agents learn how to cooperate only at the highest level of the hierarchy, which is much more efficient than learning how to cooperate using actions in the low-level environment. In a sense, we also employ a hierarchical approach, but in our case there is no need for the agents to *learn* how to cooperate at the high level because the policy learnt using their RMs ensures coordination. Orthogonally, some works have instead leveraged multi-agent RL to solve formally specified objectives, e.g., parity games (Hahn et al. 2020), or verification of non-nested quantitative ATL formulas (Hahn et al. 2024).

Reward machines were introduced by Toro Icarte et al. (2018) as a way to: (i) improve the sample efficiency of reinforcement learning by providing an RL agent with a high-level abstract description of its task and environment, and (ii), easily define rewards for non-Markovian tasks. The literature on reward machines is now quite large, and we focus on two strands related to our work: the automatic synthesis of reward machines, and the use of reward machines in multi-agent reinforcement learning.

There have been several proposals for the automated generation of RMs. For example, Toro Icarte et al. (2019) and Xu et al. (2020) propose to learn an RM via experience gathered by the agent in the environment. Instead, Camacho et al. (2019) propose an approach to synthesise RMs for single-agent RL from LTL specifications (considering only the safe and co-safe fragments of LTL) and other formal languages which are as expressive as regular languages. Since in our approach tasks are specified in FC-ATL, we are not able to encode tasks which can be represented as "nested" LTL formulas, e.g., a task where some goal φ needs to be achieved infinitely often ($\Box\Diamond\varphi$). Illanes et al. (2019) and Varricchione et al. (2024) propose approaches where a single-agent RM is derived from a sequential plan, or a partial-order plan, or the set of all partial-order plans to achieve the task. Finally, closest to our work is that of C. Zhu et al. (2024), in which reward machines are synthesised for multi-agent RL and then decomposed for individual training. However, instead of specifying tasks in FC-ATL, the authors use GR(1), a fragment of linear-time temporal logic. As FC-ATL and GR(1) are incomparable in terms of expressive power, our approach and that of C. Zhu et al. (2024) can be used to train coalitions of agents in achieving different sets of tasks.

The literature on using reward machines in multi-agent RL is also extensive, and we therefore focus work which is closest to ours. As noted above, the approach of C. Zhu et al. (2024) is the most similar to ours; however, it is specifically tailored to discrete environments, and, as the authors note in the paper, cannot be extended (in a trivial manner) to continuous domains. Hu et al. (2024) consider a scenario in which agents have access to the information of the other agents to which they are "close" in the environment, which makes it difficult to train agents individually as in our approach. Following a concept introduced by Furelos-Blanco et al. (2023), Zheng and Yu (2024) propose to use a hierarchy of reward machines in MARL, by decomposing the team task into tasks for subsets of agents in a hierarchical manner. While they synthesise reward machines, their formal language is based solely on propositional logic, meaning that it is not possible to specify complex temporal properties. Smith et al. (2023) propose several ways of decomposing team reward machines into individual ones, based on several criteria. However, while the decomposition is automatic, the original coalition reward machine is given. Finally, Ardon et al. (2023) take a different approach and try to learn the coalition reward machine from the experiences collected by the agents, extending a line of research from single-agent reinforcement learning (Toro Icarte et al. 2019; Xu et al. 2020) to the multi-agent case.

6 Conclusions and Future Work

We have presented a procedure to synthesise coalition reward machines for fully cooperative MARL tasks from specifications given in a fragment of ATL. By decomposing the coalition RM into individual RMs, we can train agents individually, so avoiding the issue on non-stationarity. Moreover, for each kind of ATL coalitional formula we consider, we have provided theoretical bounds on the probability that after agents have been trained individually, the coalition will complete the task specified by the ATL formula. In a comparison with previous approaches, we have shown empirically that the performance obtained by using our automatically synthesised RMs is comparable to that obtained by using the RMs in the literature. Moreover, our experiments in the WaterWorld domain suggest that our approach can also scale to more complex, continuous domains.

There are several directions for future work. One would be to investigate whether the use of multiple or "partially-ordered" strategies improves performance in the multi-agent setting. The RMs we synthesise are based on witnesses which essentially correspond to sequential plans, as they represent a single strategy. However, Illanes et al. (2019) showed that, for single agent RL, using partial order plans to construct RMs improves performance, and in previous work (Varricchione et al. 2024) we showed theoretically (and empirically in a Minecraft-inspired domain) how an RM obtained from the set of all partial-order plans to achieve a (single-agent) task results in policies that achieve higher rewards compared to RMs generated from a single plan. In the approach presented here, this would translate to having a witness that, instead of representing a single strategy, includes the possible

strategies to achieve the task. Note that a witness representing all strategies to complete the OfficeWorld task would result in an RM that is identical to the one of Toro Icarte et al. (2022), as can be seen in Figure 16. This may also lead to faster training when using counterfactual experiences using CRM as in the OfficeWorld experiment. While this approach can be easily implemented in a single-agent setting, it is not as trivial in a multi-agent one, as it would require a further level of coordination between the agents, which should be able to agree on which strategy to follow (for example, by being able to communicate with each other). Another direction for future work would be to consider non-fully cooperative RL scenarios. In these cases, an ATL specification could be used to generate a strategy for the coalition of agents learning the task.

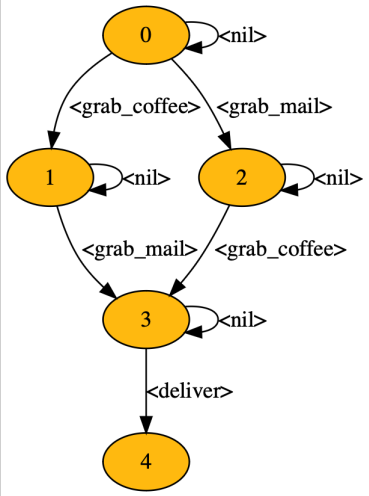


Fig. 16. Witness showing all strategies to achieve the OfficeWorld task.

References

R. Alur, T. A. Henzinger, and O. Kupferman. 2002. "Alternating-Time Temporal Logic." *Journal of the ACM*, 49, 5, 672–713.

L. Ardon, D. Furelos-Blanco, and A. Russo. 2023. "Learning Reward Machines in Cooperative Multi-agent Tasks." In: *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 43–59.

R. Bloem, S. Schewe, and A. Khalimov. 2017. "CTL* synthesis via LTL synthesis." *Electronic Proceedings in Theoretical Computer Science*, 260, 4–22.

C. Boutilier. 1996. "Planning, Learning and Coordination in Multiagent Decision Processes." In: *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, 195–210.

A. Camacho, R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith. 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.

E. M. Clarke and E. A. Emerson. 1981. "Design and synthesis of synchronization skeletons using branching time temporal logic." In: *Logics of Programs*, 52–71.

M. Cohen, M. Dam, A. Lomuscio, and F. Russo. 2009. "Abstraction in model checking multi-agent systems." In: *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 945–952.

K. Cui, A. Tahir, G. Ekinci, A. Elshamanhory, Y. Eich, M. Li, and H. Koeppl. 2022. "A Survey on Large-Population Systems and Scalable Multi-Agent Reinforcement Learning." *CoRR*, abs/2209.03859.

E. De Angelis, A. Pettorossi, and M. Proietti. 2012. "Synthesizing concurrent programs using answer set programming." *Fundamenta Informaticae*, 120, 205–229.

B. Ellis, J. Cook, S. Moalla, M. Samvelyan, M. Sun, A. Mahajan, J. Foerster, and S. Whiteson. 2023. "Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning." In: *Proceedings of the 37th Conference Advances in Neural Information Processing Systems (NeurIPS)*, 37567–37593.

A. Ferrando and V. Malvone. 2025. "VITAMIN: A compositional framework for model checking of multi-agent systems." In: *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART)*, 648–655.

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

P. Gammie and R. van der Meyden. 2004. "MCK: Model Checking the Logic of Knowledge." In: *Computer Aided Verification*, 479–483.

M. Ghavamzadeh, S. Mahadevan, and R. Makar. 2006. "Hierarchical multi-agent reinforcement learning." *Autonomous Agents and Multi-Agent Systems*, 13, 2, 197–229.

J. K. Gupta, M. Egorov, and M. Kochenderfer. 2017. "Cooperative multi-agent control using deep reinforcement learning." In: *Proceedsings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. AAMAS 2017 Workshops, Best Papers, Revised Selected Papers 16, 66–83.

E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak. 2020. "Model-free reinforcement learning for stochastic parity games." In: *Proceedings of the 31st International Conference on Concurrency Theory (CONCUR)*, 21:1–21:16.

E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak. 2024. "Multi-Agent Reinforcement Learning for Alternating-Time Logic." In: *Proceedings of the 27th European Conference on Artificial Intelligence (ECAI)*, 1680–1687.

P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote. 2019. "A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity." *CoRR*, abs/1707.09183.

P. Hernandez-Leal, B. Kartal, and M. E. Taylor. 2019. "A survey and critique of multiagent deep reinforcement learning." *Autonomous Agents and Multi-Agent Systems*, 33, 6, 750–797.

J. Hu, Z. Xu, W. Wang, G. Qu, Y. Pang, and Y. Liu. 2024. "Decentralized graph-based multi-agent reinforcement learning using reward machines." *Neurocomputing*, 564, 126974.

L. Illanes, X. Yan, R. Toro Icarte, and S. A. McIlraith. 2019. "Symbolic Planning and Model-Free Reinforcement Learning: Training Taskable Agents." In: *Proceedings of the 4th Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*, 191–195.

L. Illanes, X. Yan, R. Toro Icarte, and S. A. McIlraith. 2020. "Symbolic Plans as High-Level Instructions for Reinforcement Learning." In: *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS)*, 540–550.

A. Khalimov. 2018. "Reactive Synthesis: Branching Logics and Parameterized Systems." Ph.D. Dissertation. Graz University of Technology (Austria).

A. Khalimov and R. Bloem. 2017. "Bounded synthesis for streett, rabin, and CTL*." In: *International Conference on Computer Aided Verification (CAV)*. Springer, 333–352.

T. Klenze, S. Bayless, and A. J. Hu. 2016. "Fast, flexible, and minimal CTL synthesis via SMT." In: *International Conference on Computer Aided Verification (CAV)*. Springer, 136–156.

H. Kokel, A. Manoharan, S. Natarajan, B. Ravindran, and P. Tadepalli. 2021. "RePReL: Integrating Relational Planning and Reinforcement Learning for Effective Abstraction." In: *Proceedings of the 31st International Conference on Automated Planning and Scheduling (ICAPS)*, 533–541.

D. Kurpiewski, M. Kaminski, and W. Jamroga. 2024. "STV+FLY: On-the-Fly Model Checking of Strategic Ability in Multi-Agent Systems." In: *ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024)* (Frontiers in Artificial Intelligence and Applications). Ed. by U. Endriss, F. S. Melo, K. Bach, A. J. B. Diz, J. M. Alonso-Moral, S. Barro, and F. Heintz. Vol. 392. IOS Press, 4483–4486. doi:10.3233/FAIA241035.

E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica. 2018. "RLlib: Abstractions for Distributed Reinforcement Learning." In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 3053–3062.

M. L. Littman. 1994. "Markov Games as a Framework for Multi-Agent Reinforcement Learning." In: *Procedings of the 11th on International Conference on Machine Learning (ICML)*, 157–163.

A. Lomuscio, H. Qu, and F. Raimondi. 2017. "MCMAS: An Open-Source Model Checker for the Verification of Multi-Agent Systems." *International Journal on Software Tools for Technology Transfer*, 19, 1, 9–30.

A. Lomuscio and F. Raimondi. 2006. "Model checking knowledge, strategies, and games in multi-agent systems." In: *Proceedings of the 5th international joint conference on Autonomous agents and multiagent systems (AAMAS)*, 161–168.

M. Luttenberger, P. J. Meyer, and S. Sickert. 2020. "Practical synthesis of reactive systems from LTL specifications via parity games." *Acta Informatica*, 57, 3–36.

P. J. Meyer, S. Sickert, and M. Luttenberger. 2018. "Strix: Explicit Reactive Synthesis Strikes Back!" In: *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I*, 578–586.

C. Neary, Z. Xu, B. Wu, and U. Topcu. 2021. "Reward Machines for Cooperative Multi-Agent Reinforcement Learning." In: *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 934–942.

L. Panait and S. Luke. 2005. "Cooperative multi-agent learning: The state of the art." *Autonomous agents and multi-agent systems*, 11, 3, 387–434.

G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht. 2021. "Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks." In: *Proceedings of the 35th Conference on Advances in Neural Information Processing Systems (NeurIPS)*.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. 2017. "Proximal Policy Optimization Algorithms." *CoRR*, abs/1707.06347.

F. L. D. Silva, M. E. Taylor, and A. H. R. Costa. 2018. "Autonomously Reusing Knowledge in Multiagent Reinforcement Learning." In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 5487–5493.

S. Smith, C. Neary, and U. Topcu. 2023. "Automatic Decomposition of Reward Machines for Decentralized Multiagent Reinforcement Learning." In: *Proceedings of the 62nd IEEE Conference on Decision and Control (CDC)*, 5423–5430.

R. S. Sutton, D. Precup, and S. Singh. 1999. "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning." *Artificial intelligence*, 112, 181–211.

L. M. Tabajara and M. Y. Vardi. 2020. "LTLf Synthesis under Partial Observability: From Theory to Practice." *Electronic Proceedings in Theoretical Computer Science*, 326, 1–17.

M. Tan. 1993. "Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents." In: *Proceedings of the 10th International Conference on Machine Learning (ICML)*, 330–337.

H. Tang et al.. 2019. "Hierarchical Deep Multiagent Reinforcement Learning with Temporal Abstraction." *CoRR*, abs/1809.09332.

J. K. Terry et al.. 2021. "PettingZoo: Gym for Multi-Agent Reinforcement Learning." *CoRR*, abs/2009.14471.

R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith. 2022. "Reward machines: Exploiting reward function structure in reinforcement learning." *Journal of Artificial Intelligence Research*, 73, 173–208.

R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith. 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 2018)*, 2112–2121.

R. Toro Icarte, E. Waldie, T. Q. Klassen, R. A. Valenzano, M. P. Castro, and S. A. McIlraith. 2019. "Learning Reward Machines for Partially Observable Reinforcement Learning." In: *Proceedings of the 32nd Conference on Advances in Neural Information Processing Systems (NeurIPS)*, 15497–15508.

G. Varricchione, N. Alechina, M. Dastani, and B. Logan. 2024. "Maximally Permissive Reward Machines." In: *Proceedings of the 27th European Conference on Artificial Intelligence (ECAI)*, 1181–1188.

G. Varricchione, N. Alechina, M. Dastani, and B. Logan. 2023. "Synthesising Reward Machines for Cooperative Multi-Agent Reinforcement Learning." In: *Proceedings of the 20th European Conference on Multi-Agent Systems (EUMAS)*, 328–344.

Z. Xu, I. Gavran, Y. Ahmad, R. Majumdar, D. Neider, U. Topcu, and B. Wu. 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.

J. Yang, I. Borovikov, and H. Zha. 2020. "Hierarchical Cooperative Multi-Agent Reinforcement Learning with Skill Discovery." In: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 1566–1574.

K. Zhang, Z. Yang, and T. Başar. 2021. "Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms." In: *Handbook of Reinforcement Learning and Control*. Ed. by K. G. Vamvoudakis, Y. Wan, F. L. Lewis, and D. Cansever. Springer International Publishing. Chap. 12, 321–384.

X. Zheng and C. Yu. 2024. "Multi-Agent Reinforcement Learning with a Hierarchy of Reward Machines." *CoRR*, abs/2403.07005.

C. Zhu, J. Zhu, W. Si, X. Wang, and F. Wang. 2024. "Multi-agent reinforcement learning with synchronized and decomposed reward automaton synthesized from reactive temporal logic." *Knowledge-Based Systems*, 306, 112703.

S. Zhu, L. M. Tabajara, J. Li, G. Pu, and M. Y. Vardi. 2017. "Symbolic LTLf Synthesis." In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 1362–1369.

Received 18 August 2025; accepted 21 January 2026.