Synthesising Reward Machines for Cooperative Multi-Agent Reinforcement Learning

Giovanni Varricchione[1(✉)], Natasha Alechina[1], Mehdi Dastani[1], and Brian Logan[1,2]

[1] Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands
{g.varricchione,n.a.alechina,m.m.dastani,b.s.logan}@uu.nl
[2] Department of Computing Science, University of Aberdeen, Aberdeen, UK

Abstract. 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. However, current work assumes the multi-agent reward machine to be given. In this paper, we show how reward machines for team tasks can be synthesised automatically from an Alternating-Time Temporal Logic specification of the desired team behaviour and a high-level abstraction of the agents' environment. We present results suggesting that our automated approach has comparable, if not better, sample efficiency than reward machines generated by hand for multi-agent tasks.

1 Introduction

Reward machines (RMs) [4,18,19] have recently been proposed as a way of specifying rewards for reinforcement learning (RL) agents. RMs are Mealy machines used to specify tasks and rewards based on 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 in [4] has been shown to out-perform state-of-the-art RL algorithms, especially in tasks requiring specific temporally extended behaviours.

Recently, in [12], RMs were proposed as a means of specifying rewards for team tasks in multi-agent reinforcement learning. In cooperative multi-agent reinforcement learning (MARL) [13] 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 [2]. In addition, the agents are learning and updating their policies simultaneously. From the point of view of

V. Malvone and A. Murano (Eds.): EUMAS 2023, LNAI 14282, pp. 328–344, 2023.
https://doi.org/10.1007/978-3-031-43264-4_21

each individual agent, the learning problem is non-stationary; i.e., the optimal policy for each agent is constantly changing [7].

In [12] these problems are addressed 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 coalition 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 action of the agents. This avoids the problem of non-stationarity, and in [12] an algorithm based on this approach called "Decentralized Q-Learning with Projected Reward Machines" (DQPRM) is shown to be more sample efficient than independent q-learners (IQL) [16] and hierarchical independent learners (h-IL) [17].

However, in [12], the multi-agent reward machine is generated "by hand". Although reward machines are usually specified by hand, some works, such as [5,9], have shown how these can be synthesised *automatically*. Inspired by this line of research, in this paper we show how individual reward machines can be synthesised automatically from a high-level description of the agents' environment and an Alternating-time Temporal Logic (ATL) specification of the desired team behaviour. As in [12], we provide formal guarantees ensuring that the behaviour learned from our automatically synthesised individual RMs is guaranteed to result in coordinated behaviour on the team task. Moreover, as tasks are specified in ATL, we can easily incorporate additional constraints on team goals, e.g., invariant properties, which were not dealt with in [12].

The structure of this paper is as follows. In Sect. 2, we give preliminaries for our work; these include defining reward machines and the syntax and semantics of (imperfect information) ATL. In Sect. 3, we present our approach and show how to synthesise team and individual RMs. Moreover, we provide theoretical results in line to those of [12]. Section 4 provides an empirical evaluation of our work. As the main focus of our approach is to automatize the construction of RMs in multi-agent reinforcement learning, we will show how agents trained with our automatically synthesised RMs have comparable, if not better, performance than those of [12]. Then, in Sect. 5 we present related works, and in Sect. 6 we conclude and indicate possible future directions.

2 Preliminaries

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

2.1 Multi-agent Environments

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 $E = \langle Agt, S_1, \ldots, S_n, A_1, \ldots, A_n, Pr, (Prop_i)_{i \in Agt}, Val \rangle$ where:*

- *Agt is a non-empty finite set of n agents;*
- *S_i is the finite set of states of agent i. We denote the set of joint states, i.e., the cartesian product of all the sets of states, with $\mathcal{S} = S_1 \times \cdots \times S_n$;*
- *A_i is the finite 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 $\mathcal{A} = A_1 \times \cdots \times A_n$;*
- *$Pr : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \Delta(\mathcal{S})$ 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 $\boldsymbol{a} \in \mathcal{A}$;*
- *$(Prop_i)_{i \in Agt}$ is the set of propositional symbols "observable" by agent i, $Prop := \bigcup_{i \in Agt} Prop_i$ is the entire set of observable propositions;*
- *$Val : Prop \to 2^{\mathcal{S}}$ 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;*

A joint policy $\boldsymbol{\pi} : \mathcal{S} \to \Delta(\mathcal{A})$ maps any state in a MAE E to a probability distribution over the set of joint actions.

When agents are trained individually, we will consider their induced Markov Decision Process (MDP), i.e., $M_i = \langle S_i, A_i, Pr, Prop_i, Val \rangle$.

In the context of a MAE, we might be interested in some specific propositions that can aid us when specifying some task we want the agents to accomplish. In such case, we give a "labelling", used to describe how the evolution of the MAE affects the truth of these propositions. 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 2 (Labelling). *Given a MAE $E = \langle Agt, S_1, \ldots, S_n, A_1, \ldots, A_n, Pr, (Prop_i)_{i \in Agt}, Val \rangle$, 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 associated literals that are brought about by it. To be precise, 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

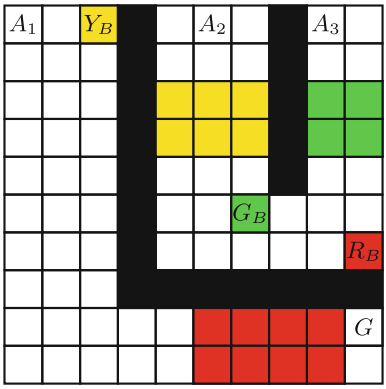


Fig. 1. CooperativeButtons domain from [12].

$p \in Prop$, 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. We now give an example of a MAE. The CooperativeButtons environment [12] is a 10×10 gridworld containing walls (some of which can be lowered) and buttons (Fig. 1). There are three agents, i_1, i_2 and i_3. 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. Walls stop agents from moving to the desired cell, but the coloured ones can be lowered by pressing the corresponding button. The red button requires two agents to press it together in order to lower the red wall. The goal of the agents is to cooperate to allow agent i_1 to reach the *Goal* location. The task can be achieved as follows: first i_1 presses the yellow button, then i_2 the green button, then i_2 and i_3 the red button, and finally i_1 reaches the *Goal* location.

The set of propositional symbols is $Prop = \{Y_B, G_B, A_2^{R_B}, A_3^{R_B}, R_B, Goal\}$. 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. If i_2 and i_3 are on the button, then propositional symbols $A_2^{R_B}$ and $A_3^{R_B}$ are respectively true. As soon as both $A_2^{R_B}$ and $A_3^{R_B}$ are true, R_B becomes true as well. Agent i_1's set of propositions is $Prop_{i_1} = \{Y_B, R_B, Goal\}$, agent i_2's is $Prop_{i_2} = \{Y_B, G_B, A_2^{R_B}, R_B\}$, and agent i_3's is $Prop_{i_3} = \{G_B, A_3^{R_B}, R_B\}$. Note that there are no propositions corresponding to exact location of agents in the environment, which would be relevant for the low-level MARL environment.

A joint state corresponds to the pair of coordinates $\langle x_j, y_j \rangle$ of each agent $i_j \in \{i_1, i_2, i_3\}$ 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.

We now explain how the set of propositional symbols is crucial in order to correctly define the dynamics of MAEs. Suppose that agent i_2 is in cell $\langle 5, 1\rangle$, i.e., in front of the yellow wall. If the agent were to perform (successfully) the `down` action, then its resulting state would depend on whether Y_B is true or false: in the former case, i_2 reaches cell $\langle 5, 2\rangle$, in the latter it will hit the wall and remain on cell $\langle 5, 1\rangle$.

Finally, 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 i_1 correctly 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 i_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 $\{Y_B^+\}$.

2.2 Reward Machines

A *reward machine* (RM) (referred to as a *simple reward machine* in [18]) is a Mealy machine over an alphabet Σ. Intuitively, an RM takes abstract descriptions of an event in the environment as input, and outputs a reward.

Definition 3. *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.*

In our case, the set of events Σ will correspond to sets of literals over the finite set of propositional symbols *Prop*, as given in the definition of MAEs.

2.3 Multi-Agent RL with RMs

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 the multi-agent analogous of a Markov Decision Process with Reward Machine (MDPRM), as defined in [18].

Definition 4. *A (cooperative) Markov Game with a Reward Machine is a tuple $G = \langle Agt, S_1, \ldots, S_n, A_1, \ldots, A_n, Pr, (Prop_i)_{i\in Agt}, Val, L, \gamma, U, u^I, \Sigma, t, r\rangle$ where:*

- *$Agt, S_j, A_j, Pr, (Prop_i)_{i\in Agt}, Val$ are as in Definition 1;*
- *$L : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \wp(\overline{Prop})$ is the labelling function, defined as in Definition 2;*

- $\gamma \in [0,1]$ *is a discount factor;*
- U, u^I, Σ, t, r *are as in Definition 3, with* $\Sigma = \wp(\overline{Prop})$*;*
- *If in states* $\boldsymbol{s} \in \mathcal{S}$*,* $u \in U$*, the agents perform an action* $\boldsymbol{a}$ *to move from* $\boldsymbol{s}$ *to* $\boldsymbol{s'}$*, then* $u' = t(u, L(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s'}))$ *and the agents receive a reward* $r(u, L(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s'}))$*.*

The alphabet Σ is a *labelling* L of triples from $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ by consistent sets of literals over $\overline{Prop}$. 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 $\boldsymbol{s}$. As each agent $i \in Agt$ has a set of observable variables $Prop_i \subseteq Prop$, we define the set of observable events of agent i as $\Sigma_i := \wp(\overline{Prop_i}) \cap \Sigma$. Notice that Σ_i is defined as the powerset of the literals obtained by considering the propositions observable by i. Similarly, for a coalition $A \subseteq Agt$, we define $\Sigma_A := \bigcup_{i \in Agt} \Sigma_i$. We assume that $\Sigma_{Agt} = \Sigma$, i.e. the grand coalition is able to observe all events. For a given event $e \in \Sigma$ and a subset of events $\Sigma' \subseteq \Sigma$, we denote the restriction of e onto Σ' by $e \upharpoonright \Sigma'$, where $e \upharpoonright \Sigma' \subseteq e$ is the maximal subset[2] (with respect to inclusion) of e that is also in Σ'. This will be used to define the 'part of' the event e that is observable by a given subset of agents.

The (cooperative) multi-agent reinforcement learning problem [3,15] is to learn an optimal group policy $\boldsymbol{\pi}^* : \mathcal{S} \to \Delta(\mathcal{A})$ that maximises the expected discounted future reward from any joint state.

2.4 Alternating-Time Temporal Logic

Alternating-time Temporal Logic (ATL) [1] is a standard formalism for specifying the high-level behaviour of agents in multi-agent systems. In this section, we define the syntax and semantics of ATL with imperfect information. We need imperfect information because we cannot 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 (finite) set of propositional symbols. Formulas of ATL are defined by the following syntax:

$$\phi, \psi ::= p \mid \neg\phi \mid \phi \vee \psi \mid \langle\langle A \rangle\rangle \bigcirc \phi \mid \langle\langle A \rangle\rangle \Box \phi \mid \langle\langle A \rangle\rangle \phi \,\mathcal{U}\, \psi$$

where $p \in Prop$ is a proposition and $A \subseteq Agt$. Here, $\langle\langle A \rangle\rangle \bigcirc \phi$ means that coalition A has a strategy to ensure that the next state satisfies ϕ, $\langle\langle A \rangle\rangle \Box \phi$ that A has a strategy to ensure that ϕ is always true, and $\langle\langle A \rangle\rangle \phi \,\mathcal{U}\, \psi$ that A has a strategy to ensure that ϕ holds until it eventually enforces ψ.

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*".

[2] We assume that Σ' is a subset of events obtained by taking the powerset of a subset of propositional symbols $Prop' \subseteq Prop$. This is to ensure that $e \upharpoonright \Sigma'$ is always well-defined as the unique maximal subevent of e in Σ'.

Definition 5. *An epistemic concurrent game structure (ECGS) is a tuple* $M = \langle Agt, Q, Prop, v, (\sim_i | \ i \in Agt), Act, d, \delta \rangle$ *where:*

- Agt *is a non-empty finite set of* n *agents;*
- Q *is a non-empty finite set of states;*
- $Prop$ *is a finite set of propositional symbols;*
- $v : Prop \to \wp(Q)$ *is a valuation which associates each proposition in* $Prop$ *with a subset of states where it is 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;*
- $d : Q \times Agt \to \wp(Act) \backslash \{\emptyset\}$ *is a 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* $D(q) = d(q, i_1) \times \ldots \times d(q, i_n)$*;*
- $\delta : (q, \sigma) \mapsto Q$ *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* $\delta(q, \sigma) = q'$*.*

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, for any natural $j \in \mathbb{N}$, the notation $\lambda[j]$ to denote the j-th state q_j in the computation λ. 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 $\sigma_i \in d(q, i)$ for all $i \in A$. 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 $out(q, \sigma_A) = \{q' \in Q \mid \exists \sigma' \in D(q) : \sigma_A = \sigma'_A \wedge q' = \delta(q, \sigma')\}$.

In our case, we specifically consider ECGSs in which each action $\mathfrak{a} \in Act$ has a set of (consistent) postconditions $post(\mathfrak{a}) \subseteq \overline{Prop}$ associated to. For any coalition $A \subseteq Agt$, we define $post(\sigma_A) := \bigcup_{i \in A} post(\sigma_i)$. The transition function δ is defined accordingly: $\delta(q, \sigma)$ leads to the state q' in which the propositional symbols of positive and negative literals from $post(\sigma)$ are, respectively, true and false, and $q' \in v(p) \iff q \in v(p)$ for all propositional symbols p without a literal in $post(\sigma)$. For joint actions σ such that $post(\sigma)$ is not consistent, δ is undefined.

Example 2. As an example, we provide an ECGS that abstracts the CooperativeButtons MAE. Obviously, $Agt = \{i_1, i_2, i_3\}$. We take $Q = 2^{Prop}$, where $Prop$ is the original set of propositional symbols from the CooperativeButtons domain, as given in Example 1, which also acts as the set of propositional symbols in the ECGS. v is the naturally induced valuation, i.e., $v(p) = \{q \in Q \mid p \in q\}$. For the equivalence (indistinguishability) relationships $\sim_i$ of agent i, we take the one naturally induced by the set of "observable" propositional symbols $Prop_i$ of agent i as described in Example 1. In other words, for any agent $i \in Agt$, two states q, q' are such that $q \sim_i q'$ if and only if $q \in v(p) \iff q' \in v(p)$ for all $p \in Prop_i$. The set of actions

is $Act = \{\texttt{press\_yellow}, \texttt{press\_green}, \texttt{press\_red}, \texttt{to\_goal}, \texttt{nil}\}$, where `nil` is the "null" action that can be executed by any agent in any state and leads to no consequence. The action `press_yellow` can be performed only by agent i_1, whenever Y_B is false. `press_green` can be performed only by agent i_2, whenever G_B and Y_B are, respectively, false and true. `press_red` can be performed by agents $_i2$ and i_3: for the former, whenever Y_B is true, and, for the latter, whenever G_B is true. Finally, `to_goal` can be performed only by agent i_1, whenever R_B is true. As for the transition function δ, all valid joint actions have the "intuitive" set of postconditions, e.g., if agent i_1 performs the `press_yellow` action and i_2 and i_3 the `nil` action, then the ECGS moves from state q to state q', where the only difference is that $q \notin v(Y_B)$ and $q' \in v(Y_B)$. The only action that requires "coordination" is `press_red`, in the sense that δ is defined so that any joint action σ moves the ECGS to a state where R_B is true if and only if $\sigma_{i_2} = \sigma_{i_3} = \texttt{press\_red}$.

We would like to stress how the ECGS differs from the MAE in this example: as one can notice, the ECGS does not contain any information about the precise position of the agents in the environment, unlike the MAE. Moreover, the set of actions are completely different: the MAE's actions describe how the agents "*phisically*" move in the environment, whereas the ECGS's describe how the agents can press buttons or reach the goal. Due to this, having just a "strategy" to achieve the task in the ECGS does not suffice for the agents to be able to also achieve the task in the MAE: they need to learn how to move in the latter environment in order to do so. However, as we will later see, having a "*high-level strategy*" can aid them in learning how to act in the MAE.

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 iff, 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. Some strategies are unrealistic in that they require agents to select different actions in two states that they cannot distinguish. For this reason, the strategies are usually restricted to being uniform:

Definition 6 (Uniform strategy). *A strategy for agent i, 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 $i \in A$.*

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., all $q' \in [q]_A$, where $[q]_A$ is the equivalence class of q for $\sim_A := \bigcap_{i \in A} \sim_i$.

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:

- $M, q \models p$ iff $q \in v(p)$;
- $M, q \models \neg\phi$ iff $M, q \not\models \phi$;

- $M, q \models \phi \vee \psi$ iff $M, q \models \phi$ or $M, q \models \psi$;
- $M, q \models \langle\langle A \rangle\rangle \bigcirc \phi$ iff there exists a uniform strategy F_A such that for all $q' \sim_A q$, for all $\lambda \in out(q', F_A)$: $M, \lambda[1] \models \phi$;
- $M, q \models \langle\langle A \rangle\rangle \Box \phi$ iff there exists a uniform strategy F_A such that for all $q' \sim_A q$, for all $\lambda \in out(q', F_A)$ and $j \geq 0$: $M, \lambda[j] \models \phi$;
- $M, q \models \langle\langle A \rangle\rangle \phi \mathcal{U} \psi$ iff 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 $k \in \{0, \ldots, j-1\}$.

Finally, we define a *witness* for a coalitional modality formula (see e.g., [11]). If a formula of the form $\langle\langle A \rangle\rangle \varphi$ is true in a state q, there is a strategy F_A such that all paths generated by this strategy satisfy φ. A *witness* $W(q, F_A)$ for the truth of $\langle\langle A \rangle\rangle \varphi$ in q is a finite tree rooted in q that is generated by executing F_A. For φ of the form $\bigcirc \phi$, the tree is cut off at the first "step", meaning that only states satisfying ϕ and that can be reached from the initial state in one transition are considered. For φ of the form $\phi \mathcal{U} \psi$, the tree is cut off at the states satisfying ψ. For φ of the form $\Box \phi$, the tree is cut off at the first repeating state encountered on the branch (intuitively, it represents cyclic paths satisfying ϕ).

We specify tasks for for agent teams using ATL formulas. Hence, in our approach, the ECGS will be the "high-level environment" which abstracts the low-level MAE in which the agents act. For example, the task from the Buttons domain can be specified as $\langle\langle Agt \rangle\rangle \top \mathcal{U}\, Goal$: this formula is true if the grand coalition Agt has a strategy to reach the goal. The ATL formula plays a role similar to that of a planning goal in the synthesis of single-agent reward machines in [9]. However, the use of ATL means we can specify more flexible properties: for example, that Agt can bring about ψ while maintaining ϕ, or that Agt can maintain some property forever ϕ, etc. We can also talk about abilities of $A \subset Agt$ allowing for, e.g., the presence of opponent coalitions. Nevertheless, in this work we focus on the case in which $A = Agt$, leaving the treatment of a non-fully-cooperative setting to future research. As we always assume that $A = Agt$, we will write just A to refer to the grand coalition.

3 Synthesising MGRMs

In this section, we show how, given an ECGS M, an initial state q of such ECGS, and an ATL formula $\langle\langle A \rangle\rangle \varphi$ specifying a team task, we can synthesise a MGRM from a witness $W(q, F_A)$, where F_A is a strategy for coalition A to enforce φ (if there exists any).

In Fig. 2, 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 and individual agents' environments (represented by Markov Decision Processes), the ECGS and the ATL formula are all given in input, whereas the rest is computed in our approach. We would like to stress the fact that the dynamics of the low-level environments are hidden to the agents, which means that it is not possible for them to compute a policy to perform the task by only having a witness of a high-level strategy for it.

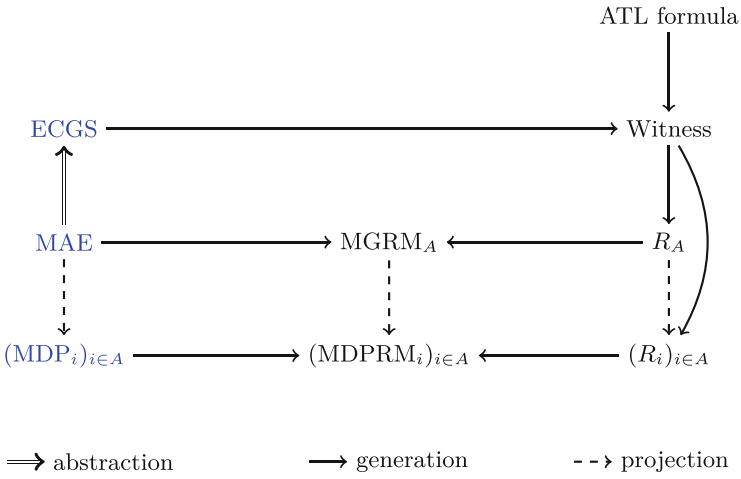


Fig. 2. High-level overview and relationship between the objects used in our approach. Objects in blue can be reused in other tasks.

3.1 Synthesising Reward Machines

Fix an ECGS $M = \langle Agt, Q, Prop, v, (\sim_i |\ i \in Agt), Act, d, \delta \rangle$ with some initial state $q \in Q$ and an ATL formula $\langle\langle A \rangle\rangle \varphi$. We can use an ATL model checker to synthesise a uniform strategy F_A to achieve the task encoded by the ATL formula $\langle\langle A \rangle\rangle \varphi$. For example, Fig. 3 shows a uniform strategy synthesised by the MCMAS model checker [11] for the CooperativeButtons task. From a uniform strategy F_A to achieve $\langle\langle A \rangle\rangle \varphi$ and some initial ECGS state q, we can generate a witness $W(q, F_A)$ for F_A in time polynomial in M and $\langle\langle A \rangle\rangle \varphi$. Then, from the witness, we can synthesise both the coalitional 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 the (high-level) actions of M and are not directly executable in the MAE 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 E should be performed to accomplish each step in the RM.

Intuitively, for a sub-coalition $B \subseteq A$, states of R_B are equivalence classes of nodes in the witness, plus an extra "error" state. Edge labels (ECGS actions) in the witness are replaced with events corresponding to postconditions of those actions. The reward machines for $\langle\langle A \rangle\rangle \phi \mathcal{U} \psi$ and $\langle\langle A \rangle\rangle \Box \phi$ transit to the error state on events corresponding to a violation of ϕ (we assume that $\neg\phi$ is always observable by A). The error state has a self-loop and no transitions to other states of the reward machine. Finally, the state corresponding to the second last state of the witness for $\langle\langle A \rangle\rangle \phi \mathcal{U} \psi$ 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).

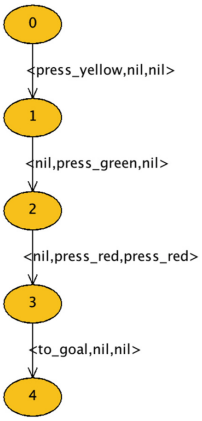


Fig. 3. Strategy for the CooperativeButtons domain. To obtain the corresponding coalition RM, it suffices to modify the action labels with their corresponding postconditions and give 1 as a reward on the transition from state 3 to 4 and 0 on all other transitions.

Reward Machine Synthesis. Given the ECGS $M = \langle Agt, Q, Prop, v, (\sim_i |\ I \in Agt), Act\ d, \delta\rangle$ 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};
- $u_B^I = [q]_B$;
- Σ_B is defined as usual. If $\langle\langle A\rangle\rangle\varphi$ is of the form $\langle\langle A\rangle\rangle\Box\phi$ or $\langle\langle A\rangle\rangle\phi\,\mathcal{U}\,\psi$, then we also add the event $\neg\phi$ to Σ_B;
- $t_B(u, e) = u'$ iff 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 $q_1 \xrightarrow{\sigma} q_2$, $\sigma_A = F_A(q_1)$ and such that $e = post(\sigma) \upharpoonright \Sigma_B$. If $\langle\langle A\rangle\rangle\varphi = \Box\phi$ or $\langle\langle A\rangle\rangle\varphi = \langle\langle A\rangle\rangle\phi\,\mathcal{U}\,\psi$ and $\neg\phi \in e$, then $u' = u_{err}$ unless $e = post(\sigma) \upharpoonright \Sigma_B$ 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. $r_B(u, e) = 1$ iff $\langle\langle A\rangle\rangle\varphi = \langle\langle A\rangle\rangle\bigcirc\phi$, $u = [q]$, and $e = post(F_A(q)) \upharpoonright \Sigma_B$, or $\langle\langle A\rangle\rangle\varphi = \langle\langle A\rangle\rangle\phi\,\mathcal{U}\,\psi$ and $e = post(\sigma) \upharpoonright \Sigma_B$ for an action σ leading to a final state q^f in $W(q, F_A)$.
  2. $r_B(u, e) = -1$ iff $\langle\langle A\rangle\rangle\varphi = \langle\langle A\rangle\rangle\Box\phi$, $u \neq u_{err}$ and $\neg\phi \in e$;
  3. For all other (u, e), $r_B(u, e) = 0$.

3.2 Correctness of the Approach

We now show that the RMs generated by our approach are "correct" decompositions in the sense of [12]. In [12] it is shown that the reward the individual

RMs grant to the agents is always equal to the reward the coalition RM would have granted them. Moreover, it is shown how the probabilities that each agent achieves its subtask bound the probability that the whole coalition achieves the team task. In this Subsection we replicate these results for our approach.

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

Theorem 1. *Fix a strategy F_A for the coalition of agents $A = \{i_1, \ldots, i_n\}$ and an initial state q. Let $R_A = \langle U_A, u_A^I, \Sigma_A, t_A, r_A \rangle$ be the coalition RM and $R_\otimes = \langle U_\otimes, u_\otimes^I, \Sigma_\otimes, t_\otimes, r_\otimes \rangle$ be the product RM $R_{i_1} \otimes \ldots \otimes R_{i_n}$. Given a compatible event sequence $\xi \in \Xi_{F_A,q}$, then for any step $j \in |\xi|$ we have that $r_A(u_A^j, \xi[j]) = r_\otimes(g(u^j), \xi[j])$, where $u^j = (u_{i_1}^j, \ldots, u_{i_n}^j)$ is the j-th state reached by the product RM following the event sequence ξ, and $g((u_{i_1}, \ldots, u_{i_n})) := \bigcap_{i \in A} u_i$ for any $(u_{i_1}, \ldots, u_{i_n}) \in U_{i_1} \times \cdots \times U_{i_n}$, with U_{i_j} being the set of states of R_{i_j}.*

Proof Sketch. The claim is proven by showing that g is a homomorphism, with respect to the transition and reward functions, from the set of states of the product RM (the parallelization of all the individual RMs) to that of the coalition RM. This can be done via an induction on the length of the input sequence.

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

Theorem 2. *If $\langle\langle A \rangle\rangle \varphi$ is of the form $\langle\langle A \rangle\rangle \bigcirc \phi$ 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 $\langle\langle A \rangle\rangle \Box \phi$, 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 Sketch. Observe that $V_A^{\pi}(\boldsymbol{s})$ and $V_{i_j}^{\pi}(\boldsymbol{s})$ are, respectively, the probabilities that coalition A and agent i_j complete, if $\langle\langle A\rangle\rangle\varphi$ is of the form $\langle\langle A\rangle\rangle \bigcirc \phi$ or $\langle\langle A\rangle\rangle \phi \mathcal{U} \psi$, or fail, if $\langle\langle A\rangle\rangle\varphi$ is of the form $\langle\langle A\rangle\rangle \Box\phi$, their (sub)task. Since A completes the task if and only if all its agents complete their task, and fails the task if and only if some of its agents fail theirs, the claim follows by applying the Fréchet inequalities for logical conjunctions and disjunctions.

Theorem 2 bounds the probability that a coalition completes or fails (depending on the formula) a task, assuming all agents follow the policy specified by $\boldsymbol{\pi}$. For formulas of the form $\langle\langle A\rangle\rangle \bigcirc \phi$ 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 team task: for all agents $V_{i_j}^{\pi}(\boldsymbol{s}) = 1$, then $\max\{0, V_{i_1}^{\pi}(\boldsymbol{s}) + \ldots + V_{i_n}^{\pi}(\boldsymbol{s}) - (n-1)\} = 1$, and so $V_A^{\pi}(\boldsymbol{s}) = 1$. Similarly, for formulas of the form $\langle\langle A\rangle\rangle \Box\phi$, if agent i_j violates ϕ, then the coalition will violate ϕ: $-V_{i_j}^{\pi}(\boldsymbol{s}) = 1$, then $\max\{-V_{i_1}^{\pi}(\boldsymbol{s}), \ldots, -V_{i_n}^{\pi}(\boldsymbol{s})\} = 1$, and so $V_A^{\pi}(\boldsymbol{s}) = -1$. When $\gamma = 1$, optimality of individual policies implies optimality of the joint policy. Note this does not imply the same holds when $\gamma < 1$, thus we leave this case to future research.

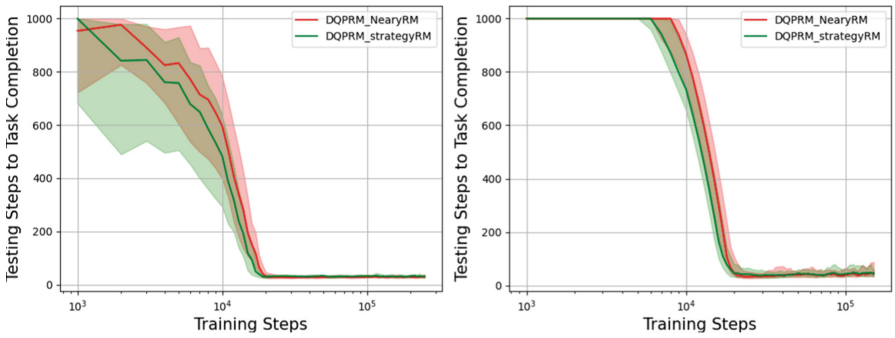
4 Evaluation

In this section we present an evaluation of the automatically synthesised reward machines. Specifically, we show how strategies for team goals generated by the model checker MCMAS [11] can be used to produce an RM for each agent in a team of agents, and present results from the two benchmarks from [12], CooperativeButtons and 10-Agent Rendezvous[3]. The Rendezvous environment is a 10×10 gridworld in which the agents first have to rendezvous at a common location, and then each agent has to reach its individual goal location.

We compare the performance of DQPRM when using our automatically synthesised RMs and the RMs from [12] in both the CooperativeButtons and Rendezvous environments. We used the same experimental setup as in [12] for these two tasks. In both the CooperativeButtons and Rendezvous environments, if an agent, during its individual training, observes an event that can also be observed by another agent, it is provided with a signal that simulates successful synchronisation with probability 0.3. This is needed to "simulate" the behaviour of other agents during the individual training. For action selection, agents use softmax exploration with a constant temperature of $\tau = 0.2$. The discount factor is $\gamma = 0.9$ and the learning rate $\alpha = 0.8$. For both tasks each experiment consists of 10 episodes: for CooperativeButtons each episode consists of 250000 training steps, while for Rendezvous 150000 training steps. For both tasks a test is run every 1000 training steps to evaluate the agents, with every test running for at most 1000 steps (after which the test is ended and the task is considered failed). Performance is measured as the number of steps necessary to complete the task. For both plots, lines represent median performance, whereas the shaded areas the 25^{th} and 75^{th} percentiles.

[3] Code is available at github.com/giovannivarr/SynthesisingRMsMARL.

The results are shown in Fig. 4. As can be seen, agents trained with our automatically generated RMs converge faster than those trained with the hand-crafted ones from [12]. We believe this might be due to the fact that in both tasks the generated RMs have less states than the hand-crafted ones. Though we do not have any formal results about this, we also think this is a side effect of synthesising RMs against defining them by hand, as in the latter case one could include information that might turn out to be superfluous to complete the final task. It might also be the case that one does not include enough information, hence obtaining an RM that is not informative enough to the agent to achieve the task. Regardless, the experimental evaluation suggests that, at least for these scenarios, automated synthesis generates RMs that successfully encode the task.



(a) Three-agent buttons task.

(b) Ten-agent rendezvous task.

Fig. 4. CooperativeButtons and Rendezvous [12]. The x-axis shows the number of elapsed training steps (in a logarithmic scale). The y-axis the number of steps required for the learned policies to complete the task – note that the agents have a maximum limit of 1000 steps to complete the task, after which the test is considered "failed".

5 Related Works

There is a large literature on the problem of non-stationarity in MARL [7,8,14, 22]. Some approaches address the problem by training each agent individually. For example, in IQL [16], each agent learns a policy by treating other agents as part of the environment. Others, e.g., [6,21], adopt a hierarchical approach, where a task is decomposed so that agents learn how to cooperate only at the highest level of the hierarchy. This seems to be more efficient than learning how to cooperate 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 RM ensures coordination.

An approach employing high-level planning for formally specified single-agent RL tasks was proposed in [10]. First, low-level policies for a set of subtasks are trained, and then high-level planning is used to identify the sequence of subtasks

which maximizes the probability of achieving the task, as described by a formal specification, given the current policies. Our work differs from [10] as we consider a multi-agent setting and use a different specification language.

Reward machines were introduced in [18] as a way of improving the sample efficiency of reinforcement learning by providing an RL agent with a high-level abstract description of its task and environment. There have been several proposals for the automated generation of RMs. For example, in [19,20] an RM is learned by an agent through experience in the environment. Closer to our work is [4], where an RM for a single agent is generated using LTL and other logics that are equivalent to regular languages, and [9] where a single-agent RM is generated from a sequential or a partial order plan. However, to the best of our knowledge, our approach is the first to synthesise individual RMs in a multi-agent setting.

6 Conclusions and Future Work

We have given a procedure to synthesise team reward machines for a cooperative MARL task from a given ATL specification. As in [12], the team RM is then decomposed in individual RMs, one per agent in the team, that are used to train such agents individually. We have provided theoretical bounds on the probability of the team completing the task after its agents are trained individually, similarly to what was done in [12]. Empirically, we have shown that the performance we obtain by using our synthesised RMs is broadly similar to that obtained by using hand-crafted ones from [12].

One direction for future work would be to investigate whether the use of multiple or "partially-ordered" strategies improves performance in the multi-agent setting. The RMs we construct are based on witnesses. Essentially, they correspond to sequential plans, each of them representing a single strategy. However, in [9] it was shown that, for single-agent RL, using partial order plans to construct RMs improves performance. In our approach, this would translate to having a witness that, instead of representing a single strategy, shows all possible strategies to achieve the task. In truth, this can already be done in the current version of MCMAS. While this approach can be easily implemented in a single-agent setting, it is not as trivial in a multi-agent one due to various reasons, e.g., it would require the agents to communicate to decide which plan to follow.

Another future direction would be to investigate non-cooperative RL scenarios. In these cases, ATL could be easily employed to produce a strategy for the coalition of agents we are interested in. MCMAS, the model checker we used in this work, is able to generate witnesses for such settings. To the best of our knowledge, this would also be a novelty in the reward machines literature, as RMs have never been employed in a non-cooperative multi-agent setting.

Finally, one could consider to enrich the specification language to ATL*. This would enable even more flexible specification of tasks and generation of strategies for several temporal formulas simultaneously.

References

1. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM **49**(5), 672–713 (2002)
2. Boutilier, C.: Planning, learning and coordination in multiagent decision processes. In: Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge, pp. 195–210. Morgan Kaufmann Publishers Inc. (1996)
3. Buşoniu, L., Babuška, R., De Schutter, B.: Multi-agent reinforcement learning: an overview. In: Srinivasan, D., Jain, L.C. (eds.) Innovations in Multi-Agent Systems and Applications - 1. Studies in Computational Intelligence, vol. 310, pp. 183–221. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14435-6_7
4. Camacho, A., Toro Icarte, R., Klassen, T.Q., Valenzano, R., McIlraith, S.A.: LTL and beyond: formal languages for reward function specification in reinforcement learning. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI-19, pp. 6065–6073. IJAI (2019)
5. Camacho, A., Toro Icarte, R., Klassen, T.Q., Valenzano, R., McIlraith, S.A.: LTL and beyond:formal languages for reward function specification in reinforcement learning. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, pp. 6065–6073. International Joint Conferences on Artificial Intelligence Organization (2019). https://doi.org/10.24963/ijcai.2019/840
6. Ghavamzadeh, M., Mahadevan, S., Makar, R.: Hierarchical multi-agent reinforcement learning. Auton. Agent. Multi-Agent Syst. **13**(2), 197–229 (2006)
7. Hernandez-Leal, P., Kaisers, M., Baarslag, T., de Cote, E.M.: A survey of learning in multiagent environments: dealing with non-stationarity (2019)
8. Hernandez-Leal, P., Kartal, B., Taylor, M.E.: A survey and critique of multiagent deep reinforcement learning. Auton. Agent. Multi-Agent Syst. **33**(6), 750–797 (2019)
9. Illanes, L., Yan, X., Toro Icarte, R., McIlraith, S.A.: Symbolic plans as high-level instructions for reinforcement learning. In: Beck, J.C., Buffet, O., Hoffmann, J., Karpas, E., Sohrabi, S. (eds.) Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020), pp. 540–550. AAAI Press (2020). www.ojs.aaai.org/index.php/ICAPS/article/view/6750
10. Jothimurugan, K., Bansal, S., Bastani, O., Alur, R.: Compositional reinforcement learning from logical specifications. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems, vol. 34, pp. 10026–10039. Curran Associates, Inc. (2021). www.proceedings.neurips.cc/paper_files/paper/2021/file/531db99cb00833bcd414459069dc7387-Paper.pdf
11. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: an open-source model checker for the verification of multi-agent systems. Int. J. Softw. Tools Technol. Transfer **19**(1), 9–30 (2017)
12. Neary, C., Xu, Z., Wu, B., Topcu, U.: Reward machines for cooperative multi-agent reinforcement learning. In: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2021), pp. 934–942. ACM (2021)
13. Panait, L., Luke, S.: Cooperative multi-agent learning: the state of the art. Auton. Agent. Multi-Agent Syst. **11**(3), 387–434 (2005)
14. Silva, F.L.D., Taylor, M.E., Costa, A.H.R.: Autonomously reusing knowledge in multiagent reinforcement learning. In: Proceedings of the 27th International Joint

Conference on Artificial Intelligence, IJCAI-18, pp. 5487–5493. International Joint Conferences on Artificial Intelligence Organization (2018)
15. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT press, Cambridge (2018)
16. Tan, M.: Multi-agent reinforcement learning: independent vs. cooperative agents. In: In Proceeedings of the 10th International Conference on Machine Learning, pp. 330–337 (1993)
17. Tang, H., et al.: Hierarchical deep multiagent reinforcement learning with temporal abstraction (2019)
18. Toro Icarte, R., Klassen, T.Q., Valenzano, R., McIlraith, S.A.: Reward machines: exploiting reward function structure in reinforcement learning. J. Artif. Intell. Res. **73**, 173–208 (2022)
19. Toro Icarte, R., Waldie, E., Klassen, T.Q., Valenzano, R.A., Castro, M.P., McIlraith, S.A.: Learning reward machines for partially observable reinforcement learning. In: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, pp. 15497–15508 (2019)
20. Xu, Z., et al.: Joint inference of reward machines and policies for reinforcement learning. In: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 30, pp. 590–598 (2020)
21. Yang, J., Borovikov, I., Zha, H.: Hierarchical cooperative multi-agent reinforcement learning with skill discovery. In: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, pp. 1566–1574. International Foundation for Autonomous Agents and Multiagent Systems (2020)
22. Zhang, K., Yang, Z., Başar, T.: Multi-agent reinforcement learning: a selective overview of theories and algorithms. In: Vamvoudakis, K.G., Wan, Y., Lewis, F.L., Cansever, D. (eds.) Handbook of Reinforcement Learning and Control. SSDC, vol. 325, pp. 321–384. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-60990-0_12