MARL with Reward Machines

Natasha Alechina[1,2] and Brian Logan[2,3]

[1]Open University NL, [2]Utrecht University, [3]University of Aberdeen

natasha.alechina@ou.nl, b.s.logan@uu.nl

Outline of this lecture

Multi-Agent reinforcement learning

Recap: multi-agent reinforcement learning

- in (cooperative) multi-agent reinforcement learning (MARL), agents learn to perform a joint task
- objective is maximising the expected future reward of the team
- MARL is more challenging than single-agent RL:
  - the agents must coordinate, as the correctness of the actions of each agent may depend on the actions of other agents in the team
  - the agents are learning and updating their policies simultaneously
  - from the point of view of any individual agent, the learning problem is non-stationary; i.e., the optimal policy for each agent is constantly changing

Recap: Markov Games

Definition (Cooperative Markov Game)

A cooperative Markov Game with n agents is a tuple $\mathcal{G} = \langle S_1, \ldots, S_n, A_1, \ldots, A_n, p, R, \gamma \rangle$ where:

- S_i is a finite set of states of agent i;
- A_i is a finite set of actions of agent i;
- $Pr : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \Delta(\mathcal{S})$ is a joint state transition probability distribution and $\Delta(\mathcal{S})$ is the set of all probability distributions over $\mathcal{S}$;
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}$ is a joint reward function that assigns the reward shared by all agents; and
- $\gamma \in (0, 1]$ is the discount factor.

The cooperative multi-agent reinforcement learning problem is to learn an optimal group policy $\pi^* : \mathcal{S} \to \Delta(\mathcal{A})$ that maximises the expected discounted future reward

MARL with reward machines

Multi-Agent Reward Machines

- in Lecture 2, we saw how Reward Machines can be used to encode the structure of single-agent learning problems
- recently, Neary et al. (2021) proposed Multi-Agent Reward Machines (MARMs) as a means of specifying rewards for joint tasks in MARL
- a MARM encodes the structure of a joint task
- the MARM is decomposed into individual reward machines, one for each member of the team, by "projecting" the MARM onto the set of observable events of each agent
- combined behaviour of the individual reward machines is bisimulation equivalent to that of the team reward machine

Decentralised learning

- this allows decentralised learning
- each agent can be trained using its individual RM to perform its part of the team task independently, while still ensuring that the team task will be achieved by the joint action of the agents
- avoids the problem of non-stationarity
- in Neary et al. (2021) an algorithm called “Decentralized Q-Learning with Projected Reward Machines” (DQPRM) was shown to be more sample efficient than independent Q-learners (IQL) and hierarchical independent learners (h-IL)

Synthesising MARMs

- in Neary et al., the Multi-Agent Reward Machine is assumed to be given as an input
- how a MARM can be computed from joint task specification expressed in a property specification language is not considered

- **key idea:** synthesise individual RMs from a declarative specification of a multi-agent task
- multi-agent tasks are specified in Alternating Time Temporal Logic with imperfect information and imperfect recall, ATL_{ir}

Alternating-time Temporal Logic with Imperfect Information

Coalition modalities

- ATL_{ir} is a branching-time logic of strategic ability
- it has coalition modalities $\langle\langle A \rangle\rangle \varphi$ which express that, by acting together, the coalition of agents A can enforce a temporal property φ, no matter what the other agents in the system do
  - $\langle\langle agents \rangle\rangle F \varphi$ "*agents* can achieve the goal φ"
  - $\langle\langle agents \rangle\rangle G \varphi$ "*agents* can enforce the safety property φ"

Branching time

- in branching time we reason about all possible computations (runs) of a system
- in ATL_{ir} we have branching future because we consider all possible executions of the coalition's strategy
- each branch is a different course of events that results from different responses by other agents

Syntax of ATL_{ir}

Definition (Syntax of ATL_{ir})

Formulas of ATL_{ir} are defined by the following syntax

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \langle\langle A \rangle\rangle X\varphi \mid \langle\langle A \rangle\rangle G\varphi \mid \langle\langle A \rangle\rangle \varphi_1 \, U \, \varphi_2$$

where p is a proposition and $A \subseteq Agt$.

- note that $\langle\langle A \rangle\rangle F\varphi$ is definable as $\langle\langle A \rangle\rangle \top \, U \, \varphi$

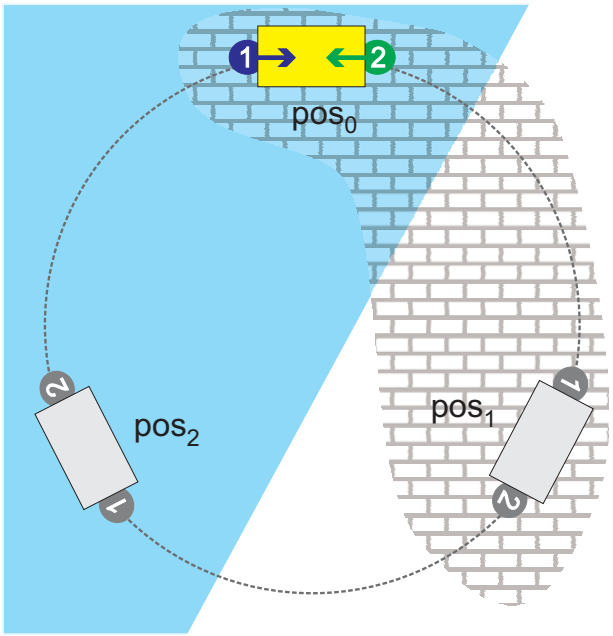
Semantics of ATL_{ir}

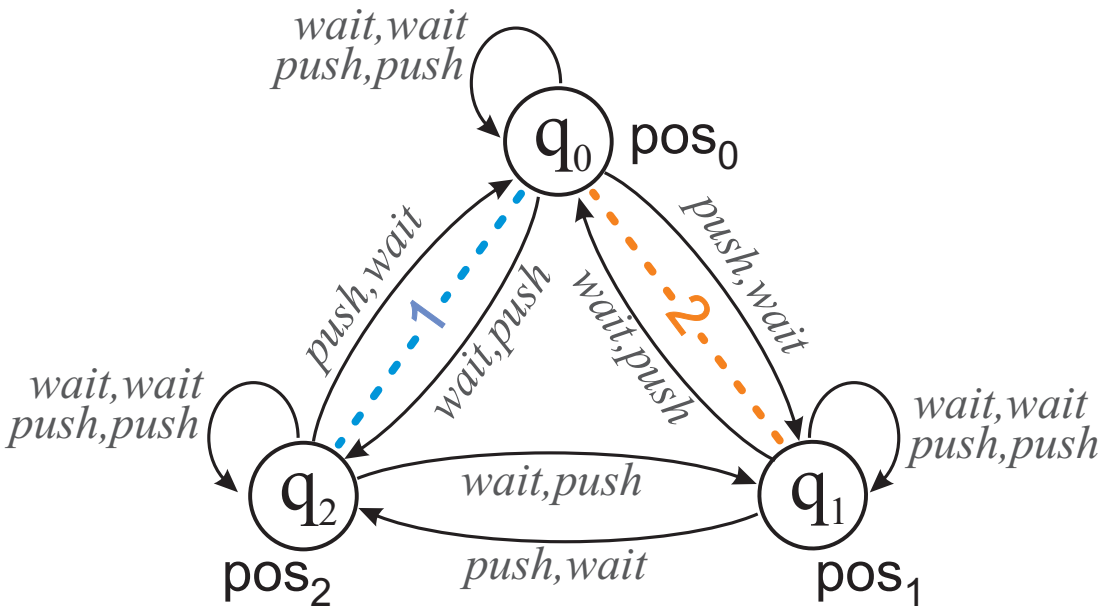
Definition (Epistemic Concurrent Game Structure (ECGS))

A **epistemic concurrent game structure** is a tuple $M = \langle Agt, St, \{\sim_i | \ i \in Agt\}, \mathcal{V}, Act, d, o \rangle$, where:

- Agt: a finite set of all agents
- St: a set of states
- $\{\sim_i | \ i \in Agt\}$: epistemic indistinguishability relations between states
- $\mathcal{V}$: a valuation of propositions
- Act: a finite set of (atomic) actions
- $d : Agt \times St \to 2^{Act}$ the actions available to an agent in a state
- o: a deterministic transition function that assigns outcome states $q' = o(q, \alpha_1, \ldots, \alpha_k)$ to states and tuples of actions

Example ECGS: Robots and Carriage





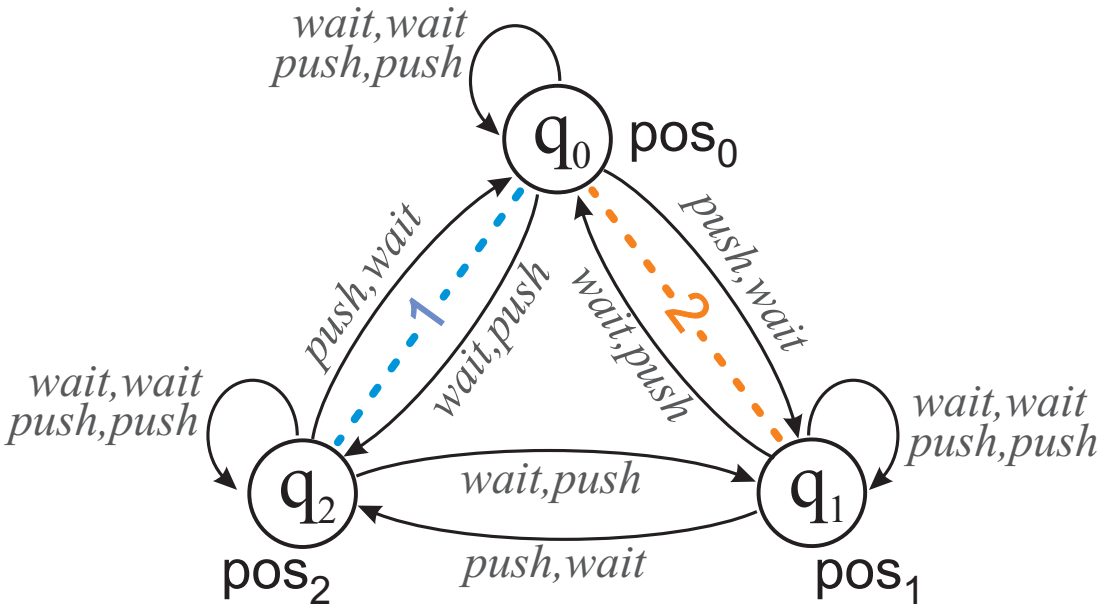
Strategies

- a **strategy** for an agent a is a function $s_a : St \rightarrow Act$
- a **collective strategy**, s_A for a coalition A is simply a tuple of individual strategies $\langle s_{a_1}, s_{a_2}, \ldots, s_{a_n} \rangle$ for $a_i \in A$

Definition (Outcome of a strategy)

The function $out(q, s_A)$ returns the set of all paths that may result from agents A executing strategy s_A from state q onward.

Why some strategies are useless



agent 1 has a strategy to always avoid pos_1:

$q_0 \mapsto$ *wait*

$q_2 \mapsto$ *push*

$q_1 \mapsto$ *wait*

but how can it execute it?

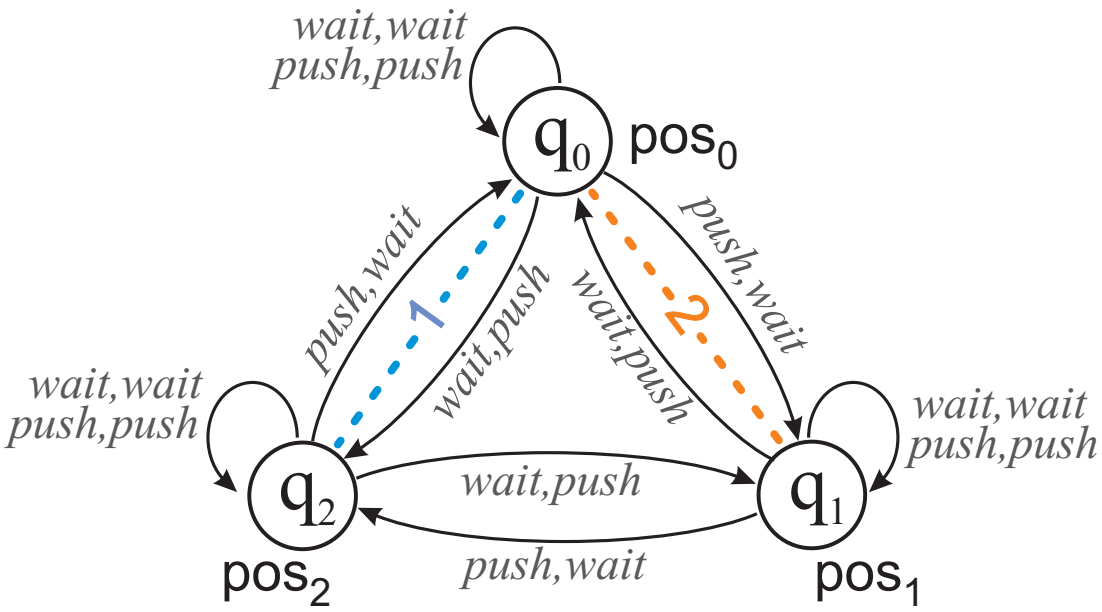
Uniform Strategies

Definition (Uniform strategy)

Strategy s_a is **uniform** iff it specifies the same choices for indistinguishable states: if $q \sim_a q'$ then $s_a(q) = s_a(q')$

A collective strategy is uniform iff it consists only of uniform individual strategies

Why uniform strategies are not enough



in q_0, agent 1 has a uniform strategy to avoid pos_1 in the next state:

$q_0 \mapsto$ *wait*

$q_2 \mapsto$ *wait*

$q_1 \mapsto$ *wait*

and in q_2 as well (*push*)

but which one should it use?

Uniform strategies are not enough

- having a uniform strategy is not enough to enforce a temporal property φ
- a uniform strategy which specifies the same actions in indistinguishable states may achieve φ from some states but not from others

- we need a strategy that will enforce φ from all indistinguishable states

Strongly uniform strategies

- single agent case: we take into account the paths starting from indistinguishable states (i.e., $\bigcup_{q' \sim_a q} out(q', s_A)$)
- what does it mean that two states q and q' are indistinguishable for a coalition?
- usually means that for at least one agent in the coalition the two states are indistinguishable, denoted as $\sim_A^E = \cup_{i \in A} \sim_i$

Intuitive meaning of cooperation modalities

- $\langle\langle a \rangle\rangle \gamma$: agent a has a uniform strategy to enforce γ from all the states it considers possible
- $\langle\langle A \rangle\rangle \gamma$: agents A have a collective uniform strategy to enforce γ from all the states at least one of them considers possible

Truth definition for ATL_{ir}

$M, q \models p$ iff p is in $\mathcal{V}(q)$;
$M, q \models \neg\varphi$ iff $M, q \not\models \varphi$;
$M, q \models \varphi_1 \wedge \varphi_2$ iff $M, q \models \varphi_1$ and $M, q \models \varphi_2$;

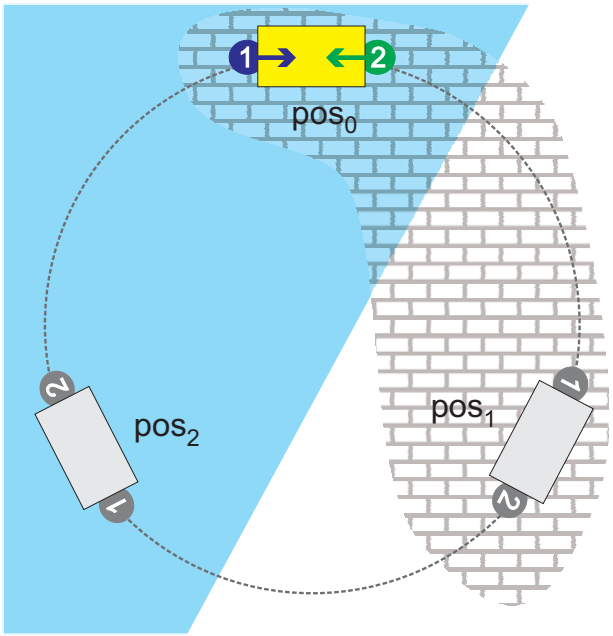
Truth definition for ATL_{ir} continued

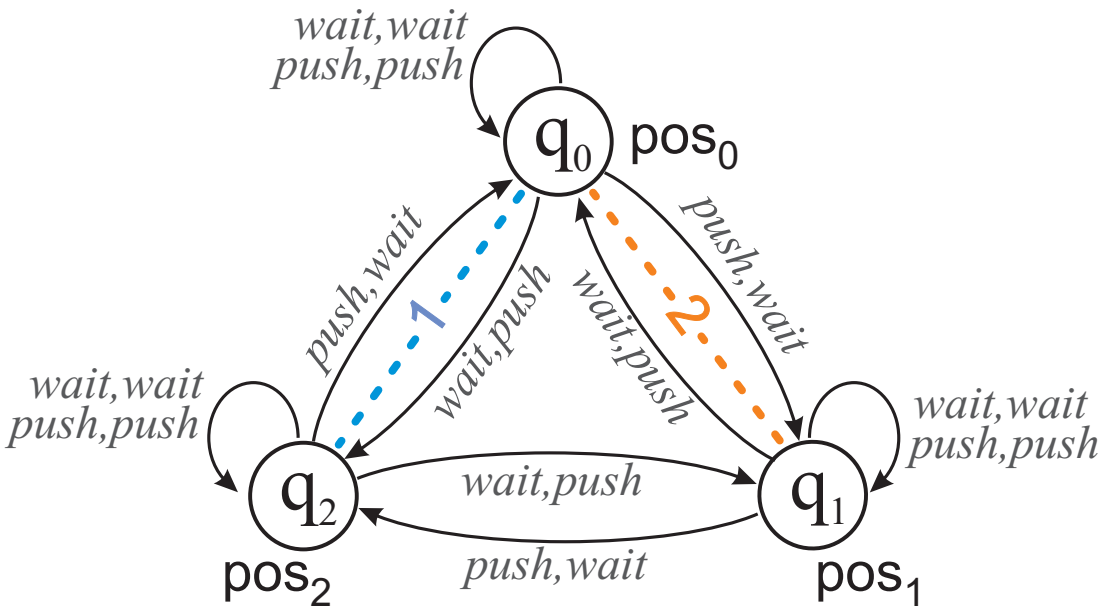
$M, q \models \langle\langle A \rangle\rangle X \varphi$ iff there is a collective uniform strategy s_A such that, for every path $\lambda \in \bigcup_{q' \sim_A^E q} out(q', s_A)$, we have $M, \lambda[1] \models \varphi$

$M, q \models \langle\langle A \rangle\rangle G \varphi$ iff there is a collective uniform strategy s_A such that, for every path $\lambda \in \bigcup_{q' \sim_A^E q} out(q', s_A)$, we have $M, \lambda[i] \models \varphi$ for all $i \geq 0$

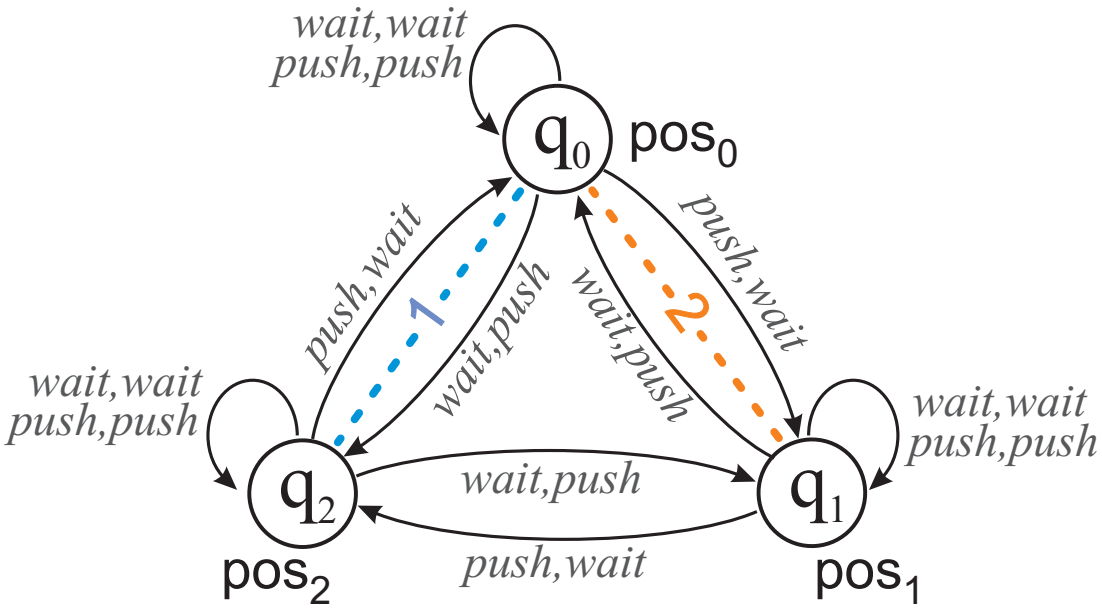
$M, q \models \langle\langle A \rangle\rangle \varphi_1 \, U \, \varphi_2$ iff there is a collective uniform strategy s_A such that, for every path $\lambda \in \bigcup_{q' \sim_A^E q} out(q', s_A)$, we have $M, \lambda[i] \models \varphi_2$ for some $i \geq 0$, and $M, \lambda[j] \models \varphi_1$ forall $0 \leq j < i$;

Example: Robots and Carriage





Example: Robots and Carriage



$$\text{pos}_0 \rightarrow \neg\langle\langle 1 \rangle\rangle G \neg \text{pos}_1$$

$$\text{pos}_0 \rightarrow \neg\langle\langle 1, 2 \rangle\rangle G \neg \text{pos}_1$$

$$\text{pos}_0 \rightarrow \langle\langle 1, 2 \rangle\rangle F \text{pos}_1$$

ATL_{ir} model checking

Verification

- in formal verification, we are given a property that should hold of a system, and a model of the system
- aim is to ensure that the system satisfies the property
- there are various ways of doing this—we will focus on:
  - **model checking:** given a model and a formula, does the model satisfy the formula
  - **synthesis:** automatically synthesise (provably) correct behaviours from a logical task specification
- when model checking ATL_{ir} the model is an ECGS
- if $\langle\langle A \rangle\rangle \varphi$ is true, we can also compute a witness, i.e., a strategy for the coalition to achieve $\langle\langle A \rangle\rangle \varphi$

Model-checking ATL_{ir}

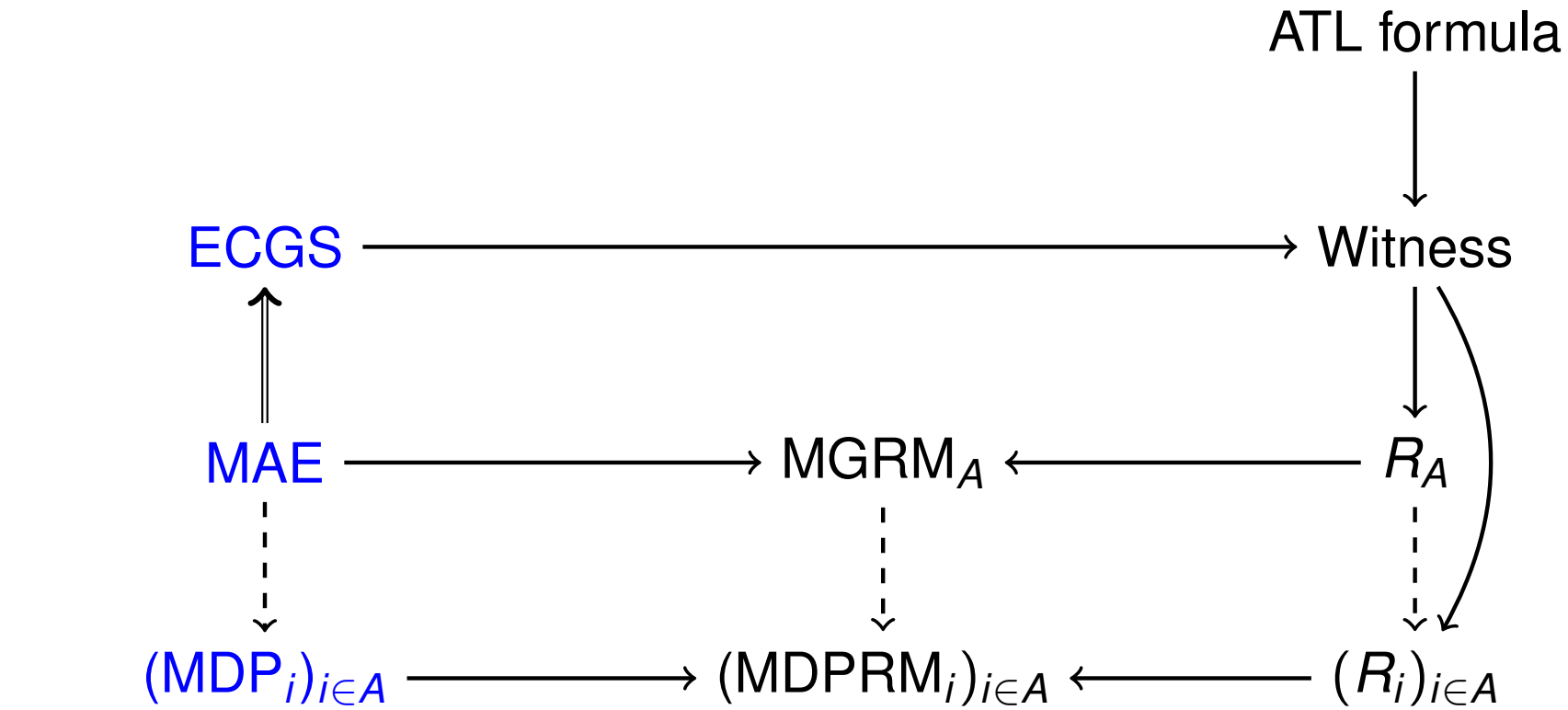
- the model-checking problem for ATL_{ir} is quite complex, because of the strong uniformity requirement.
- ATL_{ir} model checking is Δ_2^P (also denoted P^{NP}) which means solvable with polynomially many calls to an NP oracle
- intuitively, the oracle is used to "guess" uniform strategies; the rest of the computation is polynomial
- algorithms are in the Appendix

Synthesising multi-agent reward machines

Outline of the approach

- given
  - a Multi-Agent Environment (MAE) in which the agents learn
  - an ECGS which is an abstraction of the underlying MAE
  - a specification of the joint task as a formula $\langle\langle A \rangle\rangle \varphi$ in ATL_{ir}
- synthesise a strategy s_A for $\langle\langle A \rangle\rangle \varphi$
- decompose the strategy s_A into individual RMs for each agent
- train each agent independently in an MDP corresponding to the part of the MAE the agent can observe

Abstraction levels



$\Longrightarrow$ abstraction $\longrightarrow$ generates $\dashrightarrow$ projection

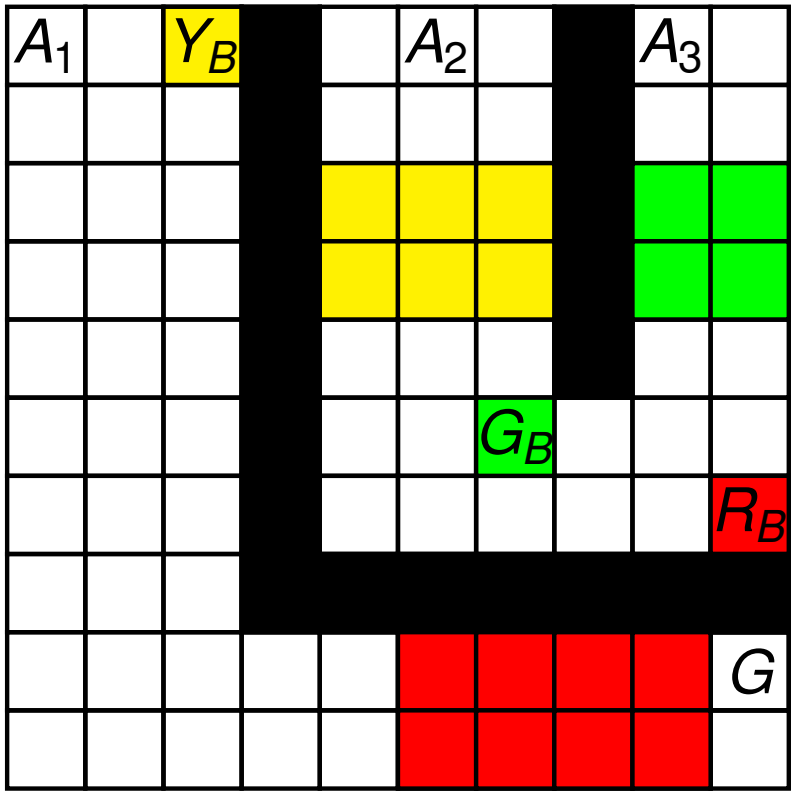
Multi-agent environment

Definition (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;
- A_i is the finite set of actions of agent i;
- $Pr : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \Delta(\mathcal{S})$ is the joint state transition probability distribution and $\Delta(\mathcal{S})$ is the set of all probability distributions over $\mathcal{S}$;
- $(Prop_i)_{i \in Agt}$ is the set of propositional symbols "observable" by agent i;
- $Val : \bigcup_{i \in Agt} Prop_i \to 2^{\mathcal{S}}$ is a valuation function mapping each propositional symbol to the set of joint states in which it is true.

Example: CooperativeButtons



CooperativeButtons domain from Neary et al. (2021)

Example: CooperativeButtons

- 10x10 grid world containing walls, three buttons, a goal location and three agents A_1, A_2 and A_3
- black walls are fixed; coloured walls can be lowered by pressing the appropriate coloured button.
- the task consists in allowing agent A_1 to reach the goal location
- agents can move left, up, right, down or stay in the same cell
- movement actions are nondeterministic: an agent may end up in a cell adjacent to the one it was trying to reach with probability 0.2
- $Prop = \{Y_B, G_B, A_2^{R_B}, A_2^{\neg R_B}, A_3^{R_B}, A_3^{\neg R_B}, R_B, Goal\}$

Labelling

Definition (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 a function $L : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \wp(\overline{Prop})$ mapping transitions $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s'})$ in the MAE to the set of associated literals that are brought about by the transition

Relationship between MAE and ECGS

- the MAE corresponds to the agents' joint environment
- the ECGS corresponds to a high-level abstraction of the agents' environment
- states in the ECGS correspond to abstract states in the MAE, e.g., that a phase of the joint task has been completed
- actions in the ECGS correspond to the high-level actions agents can perform in the abstract environment, e.g., moving to a particular location
- labelling function maps state-action-state triples of the MAE to literals of the propositional symbols over which the states of the ECGS are defined

Recap: Reward machine

Definition (Simple Reward Machine, Toro Icarte et al. (2022))

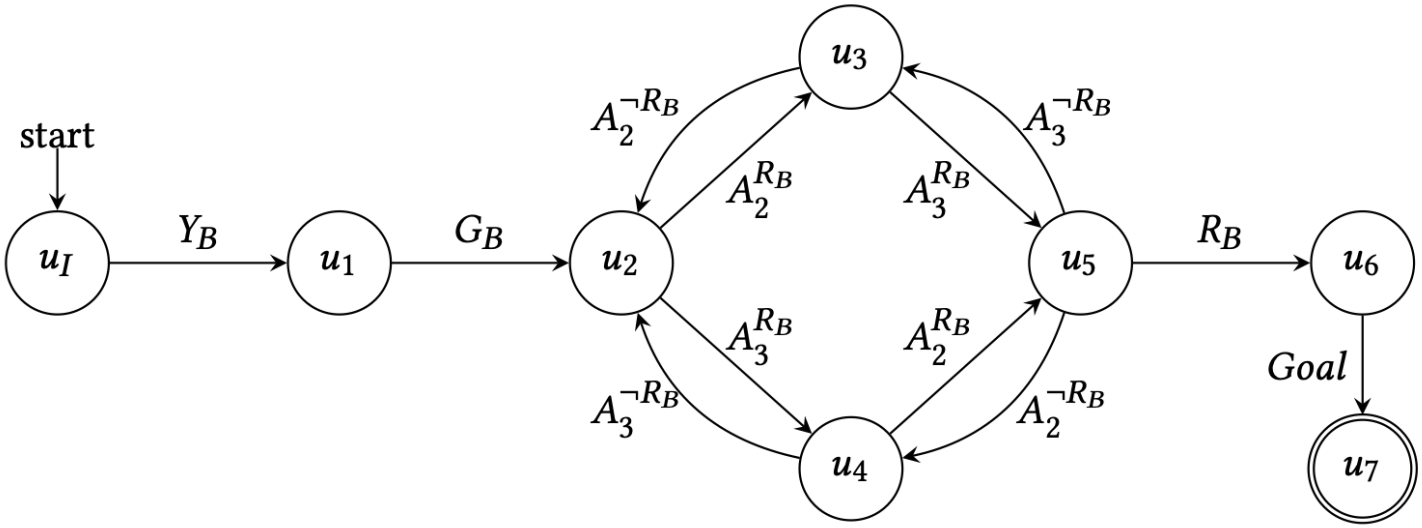
A reward machine is a tuple $R = (U, u_I, F, \Sigma, \delta_R, r_R)$ where:

- U is a finite non-empty set of states;
- $u_I \in U$ is the initial state;
- F is a set of terminal states ($U \cap F = \emptyset$);
- Σ is a finite set of environment events;
- $\delta_R : U \times \Sigma \to U \cup F$ is a transition function that, for every state $u \in U$ and environment event $\sigma \in \Sigma$, gives the state resulting from observing event σ in state u; and
- $r_R : U \times \Sigma \to \mathbb{R} \cup \{-\infty\}$ is a reward function that for every state $u \in U$ and event $\sigma \in \Sigma$ gives the reward resulting from observing event σ in state u.

Multi-agent and individual reward machines

- multi-agent reward machine represents joint (coalition) task
- individual reward machine represents the part of a coalition task performed by an agent
- note that the two kinds of reward machines are identical from a formal point of view
- transition relation δ_R does not specify *who* brings about an environment event: the coalition collectively, or an individual agent in the coalition

Example: MARM for CooperativeButtons



From Neary et al. (2021)

Markov Game with a Reward Machine

Definition (Markov Game with a Reward Machine)

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

- $Agt, S_j, A_j, Pr, (Prop_i)_{i \in Agt}, Val$ are as for MAEs;
- $L : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \wp(\overline{Prop})$ is the labelling function of the MAE;
- $\gamma \in [0, 1]$ is a discount factor;
- U, u^I, Σ, F, t, r are as for RMs, with $\Sigma = \wp(\overline{Prop})$; and
- 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 $r(u, L(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s'}))$.

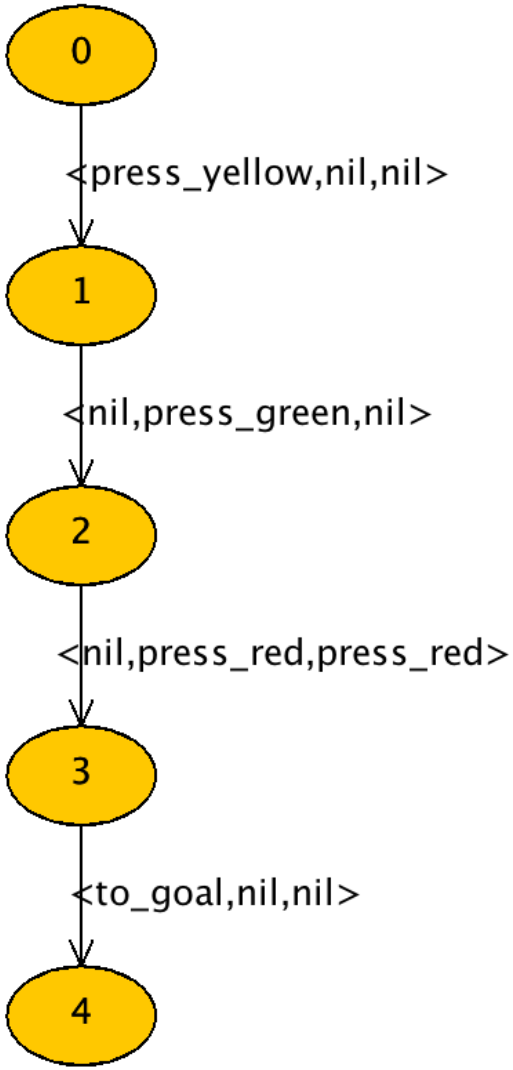
Putting it all together

- given a specification of the joint task as a formula $\langle\langle A \rangle\rangle \varphi$ in ATL_{ir} and an ECGS which abstracts the MAE
- we use model checking synthesise a witness (i.e., a strategy s_A for the coalition) which achieves $\langle\langle A \rangle\rangle \varphi$
- we decompose the strategy s_A into individual RMs for each agent
- train each agent independently in an MDP corresponding to the part of the MAE the agent can observe
- agents are evaluated in a MGRM corresponding to the MAE

Witness for a strategy

- if a formula of the form $\varphi = \langle\langle A \rangle\rangle \psi$ is true in a ECGS that is an abstraction of MAE a state q, there is a strategy F_A such that all paths generated by this strategy satisfy ψ
- a witness $W(q, F_A, \varphi)$ is a finite tree rooted in q that is generated by executing F_A
  - for φ of the form $\langle\langle A \rangle\rangle X \psi$, the tree is cut off at the first "step", meaning that only states that can be reached from the initial state in one transition are considered
  - for φ of the form $\langle\langle A \rangle\rangle \psi_1 \, U \, \psi_2$, the tree is cut off at the states satisfying ψ_2
  - for φ of the form $\langle\langle A \rangle\rangle G \psi$, the tree is cut off at the first repeating state encountered on the branch (intuitively, it represents cyclic paths satisfying ψ)

Witness for $\langle\langle Agt \rangle\rangle F goal$



From witnesses to MARMs

From $W(q, F_{Agt}, \varphi)$ to $R = (U, u_I, F, \Sigma, \delta_R, r_R)$:

- U is the set of nodes in W plus u_{err}
- u_I is q
- Σ is the set of events in $\overline{\mathcal{PV}}$, plus, if φ is of the form $\langle\langle Agt \rangle\rangle \psi_1 \, U \, \psi_2$ or $\langle\langle Agt \rangle\rangle G \, \psi_1$, $\neg\psi_1$ (a token corresponding to violating an invariant)
- $\delta_R(u, e) = u'$ iff either there are q, q' in $W(q, F_{Agt}, \varphi)$ such that there is a transition from q to q' labelled σ and e is the postcondition of σ; or, $\delta_R(u, e) = u_{err}$ if $e = \neg\psi_1$
  - $r_R(u, e) = 1$ if e is the postcondition of the action in the witness and $r_R(u, e) = u_f$ (accepting or leaf state of the witness) for $\varphi = \langle\langle Agt \rangle\rangle X \psi$ or $\langle\langle Agt \rangle\rangle \psi_1 \, U \, \psi_2$
  - $r_R(u, e) = -1$ if $u \neq u_{err}$, $\varphi = \langle\langle Agt \rangle\rangle G \, \psi_1$ or $\langle\langle Agt \rangle\rangle \psi_1 \, U \, \psi_2$ and e is a violation of ψ_1
  - in all other cases, $r_R(u, e) = 0$

From witnesses to individual RMs

From $W(q, F_{Agt}, \varphi)$ to $R^i = (U^i, u_I^i, F^i, \Sigma^i, \delta_R^i, r_R^i)$:

- U^i is the set of equivalence classes of nodes in W wrt $\sim_i$ plus u_{err}
- u_I is $[q]_{/\sim_i}$
- Σ^i is the set of events observable by i plus invariant violations
- everything else is as before but projected on events observable to i (and for i-equivalence classes of nodes in the witness)

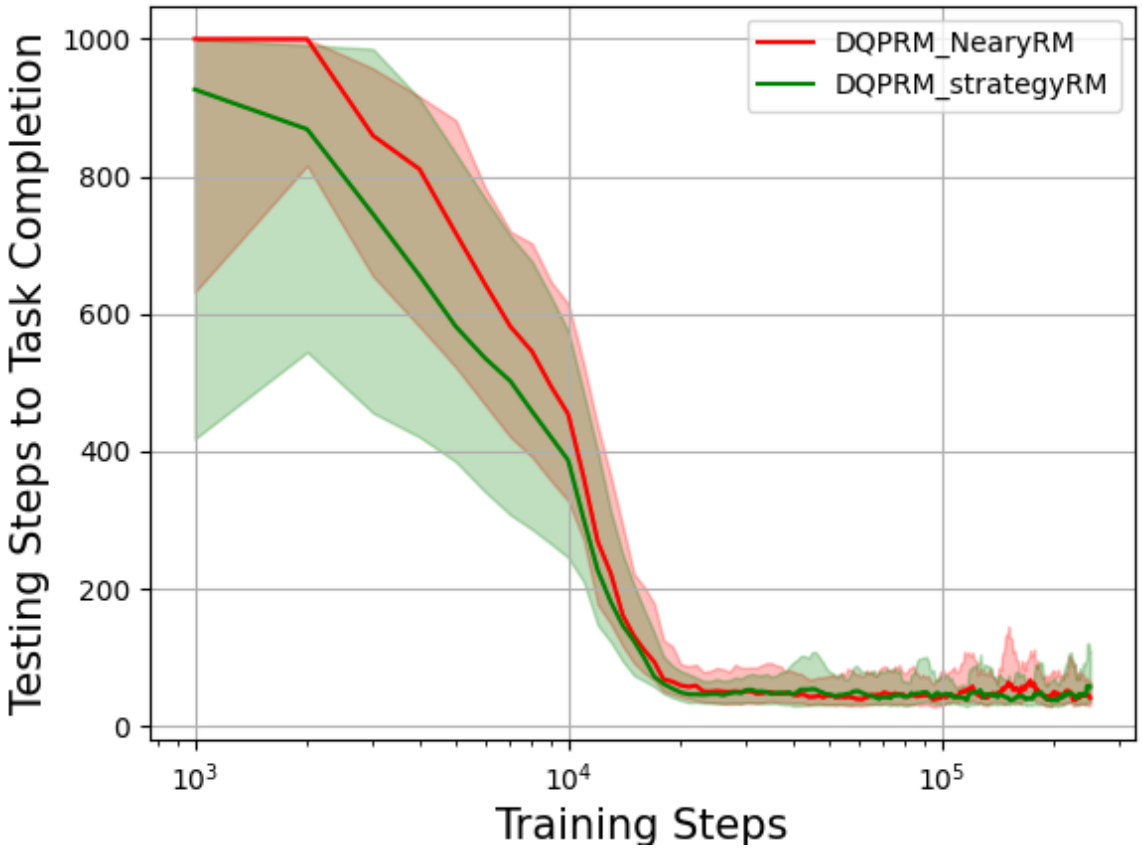
Note that some events observable by i cannot be generated by i; those will need to be 'injected' during i's decentralised training

Correctness

- for every event sequence compatible with F_A, the reward given by the joint reward machine is at every step equal to the reward generated by the product of reward machines of individual agents
- the probability of failing to complete the task specified by φ is the same for training with the joint reward machine and separate training with individual reward machines
- proofs are in Varricchione et al. (2026)

Sample efficiency

Example: CooperativeButtons

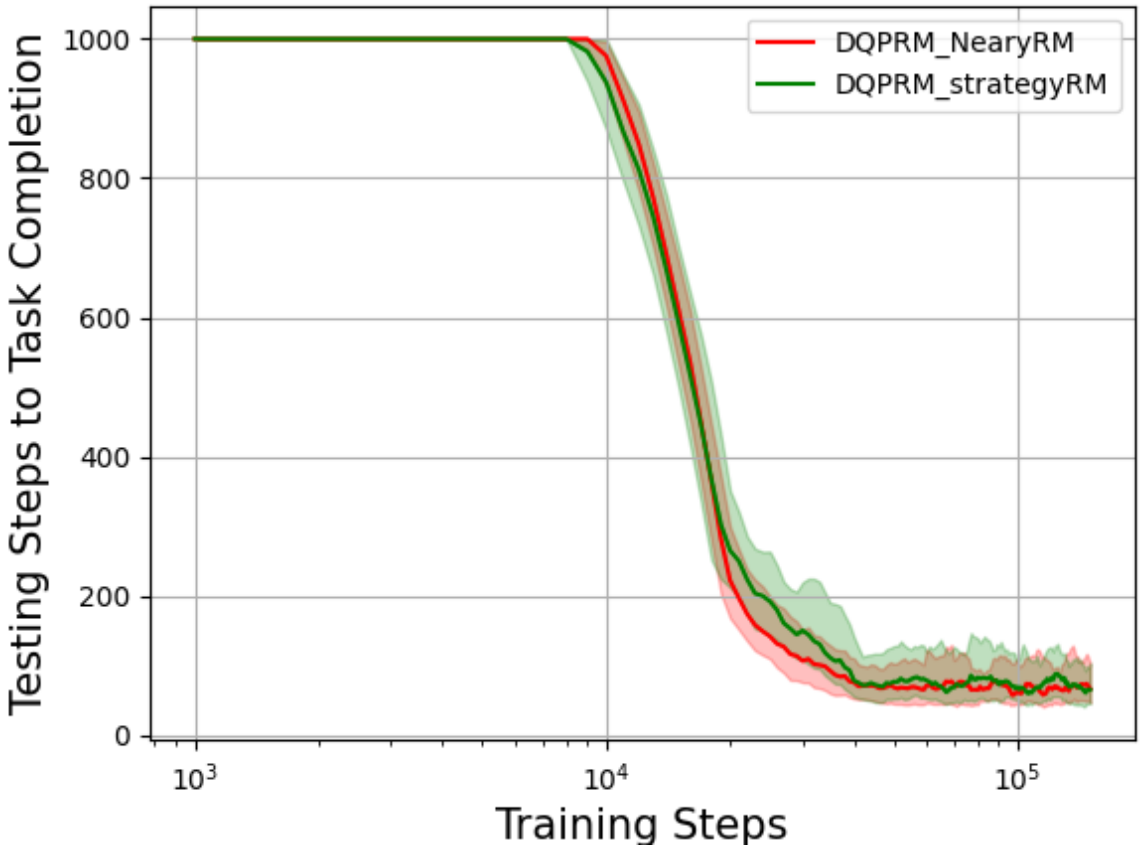


Number of training steps required by the DQPRM algorithm for the CooperativeButtons task (Neary et al. 2021)

Example: Rendezvous

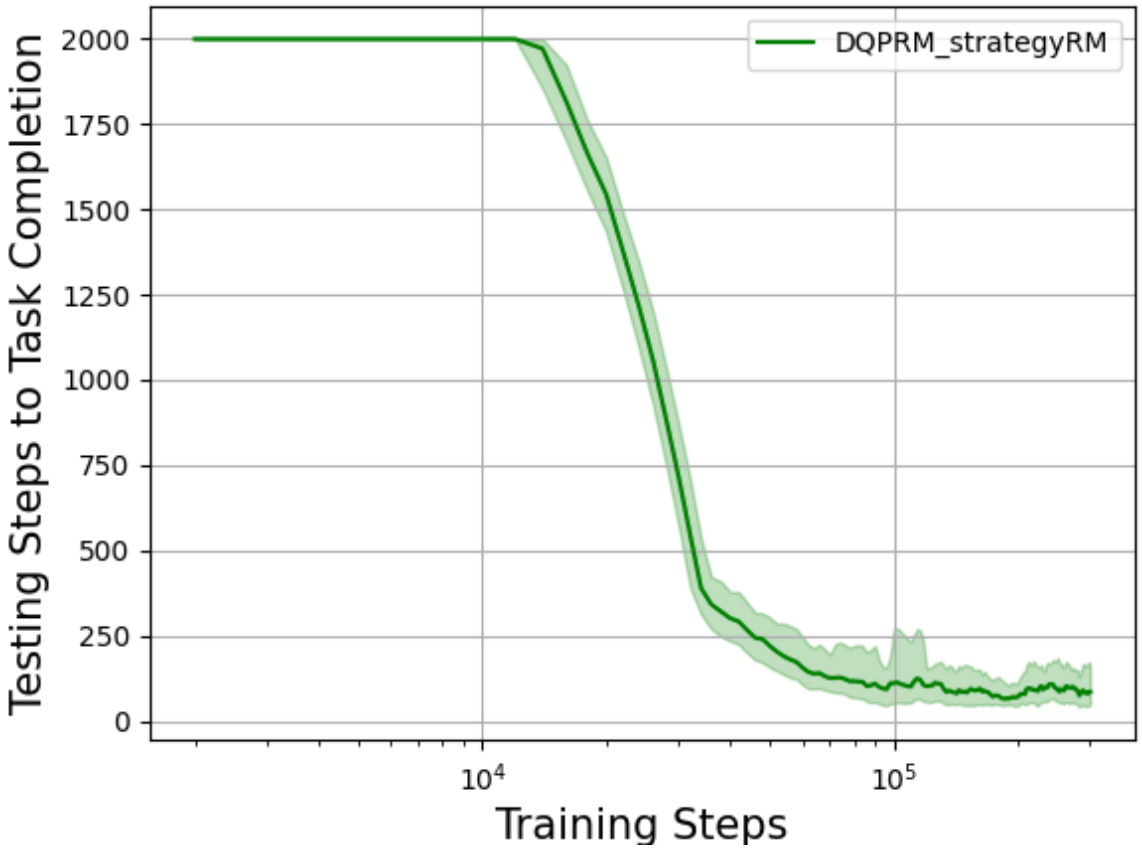
- Neary et al. also defined a “Rendezvous” task
- the Rendezvous environment is a 10x10 grid world which contains 10 agents
- task of the agents is to first rendezvous at a common location and then for each agent to reach their own goal
- to evaluate scalability of the approach we increased the number of agents to 25

Example: Rendezvous 10 agents



Number of training steps required by the DQPRM algorithm for the Rendezvous task (Neary et al. 2021)

Example: Rendezvous 25 agents



Number of training steps required by the DQPRM algorithm for the Rendezvous task (Neary et al. 2021)

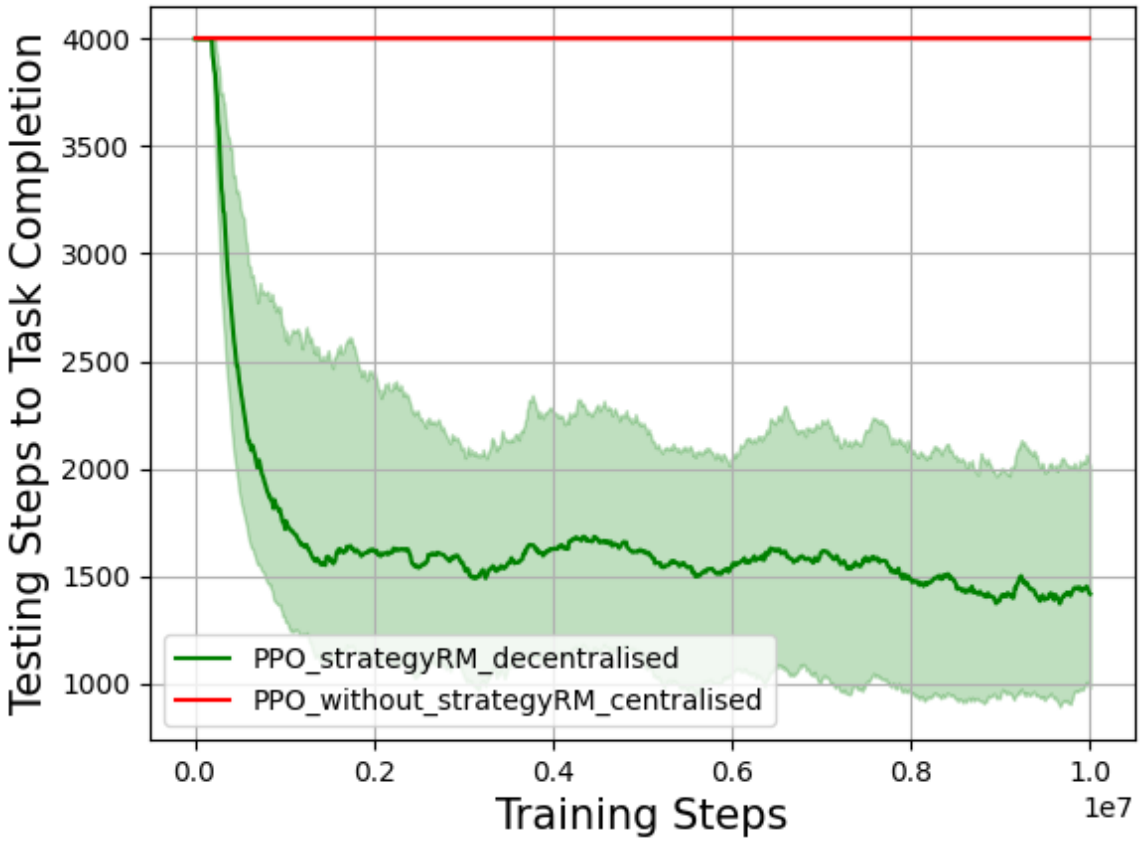
Example: WaterWorld

- WaterWorld is a continuous environment consisting of a two dimensional box in which agents (called “pursuers”) have to coordinate to capture (touch) “evaders”
- evaders move in one direction at a constant speed and change directions when they collide with an edge of the box
- pursuers can change their speed and direction
- evaluation used two pursuers A_1, A_2 and three evaders F_1, F_2, F_3
- agents were as evaluated on 2 tasks: simple and complex requiring different levels of coordination between the pursuers’ tasks to achieve the joint task

Example: WaterWorld algorithms

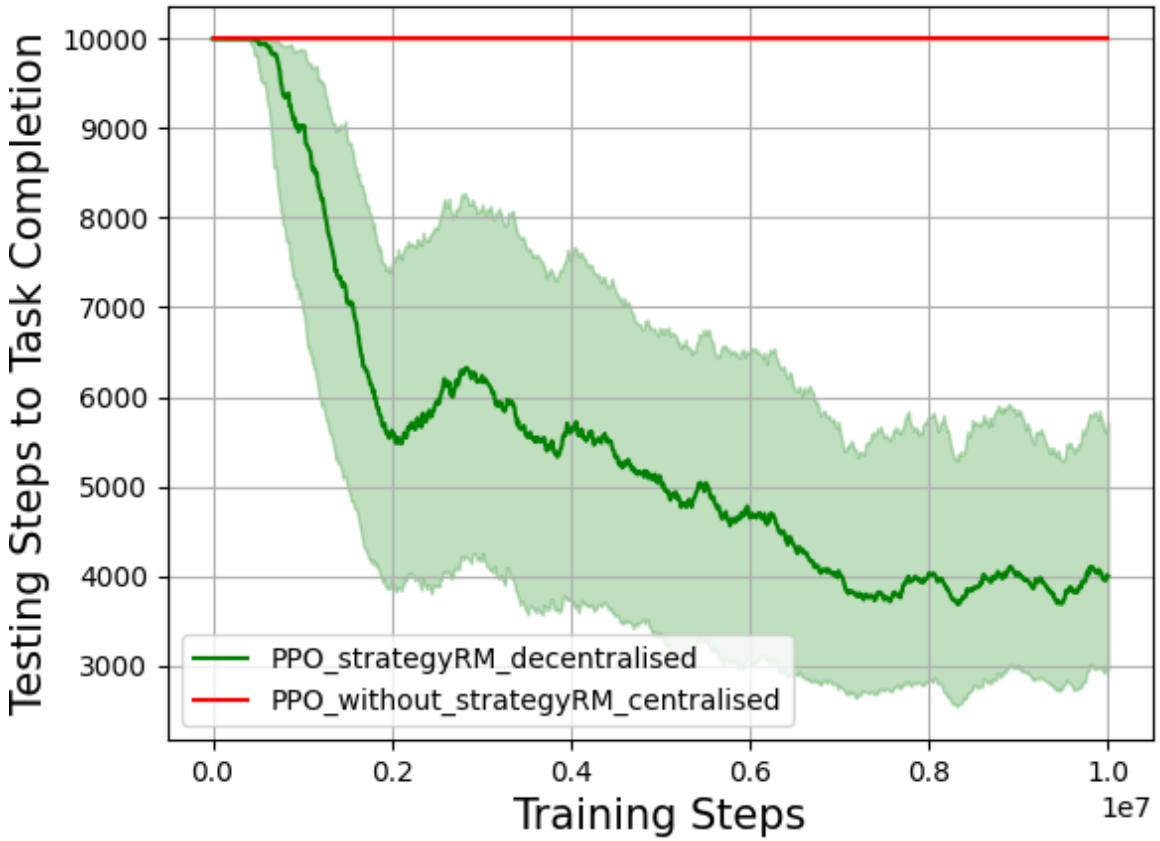
- PPO_strategyRM_decentralised: pursuer agents were trained using a decentralised approach using PPO with individual RMs
- PP0_without_strategyRM_centralised: “baseline” in which purser agents were trained using a centralised approach without a reward machine but where the rewards obtained by the agents were the same as those given by the multi-agent RM

Example: WaterWorld simple task



Pursuer A_1 must capture evader F_1, then pursuer A_2 must capture evader F_2, and finally both pursuers A_1 and A_2 must capture evader F_3 at the same time

Example: WaterWorld complex task



Pursuers A_1 and A_2 must respectively capture evaders F_1 and F_3 then they must respectively capture evaders F_3 and F_1 and finally they both must capture evader F_2 at the same time

References

- P. Y. Schobbens. Alternating-time logic with imperfect recall. Electronic Notes in Theoretical Computer Science, 85(2):82–93, 2004.
- W. Jamroga and J. Dix. Model checking ATLir is indeed Δ_2^P -complete. EUMAS 2006
- Neary et al. Reward machines for cooperative multi- agent reinforcement learning. AAMAS 2021 934–942
- Varricchione et al. Synthesising Reward Machines for Cooperative Multi-Agent Reinforcement Learning. JAIR vol. 85, 2026.

Appendix

How to model check ATL_{ir}?

- The model checking problem is: given a model and a formula, does the model satisfy the formula? In strategic logics, we also want to get the witness strategy for a coalition if $\langle\langle A \rangle\rangle \varphi$ is true.
- basic idea: given a model M, generate all possible memoryless uniform strategies for the coalition A
- each strategy induces a new model M_i (where A just have a single joint action in each state, assigned by the strategy)
- check whether the property $\langle\langle A \rangle\rangle \varphi$ holds in at least one such model

Deterministic algorithm for checking $M, q \models_{ir} \langle\langle A \rangle\rangle \varphi$

In the initial model M

- $d : Agt \times St \to 2^{Act}$ defines actions available to an agent in a state
- e.g., $d(a_1, q_1) = \{\alpha^1_1, \ldots, \alpha^1_{m_1}\}$, $d(a_2, q_1) = \{\alpha^2_1, \ldots, \alpha^2_{m_2}\}$, etc.

We generate new models $M_1, \ldots, M_N$, where $N = O(|Act|^{|St|\,k})$

- $|Act|$ is the number of actions available
- $|St|$ is the number of states in M, and
- k is the number of agents in A

Induced models

- in each M_i, each agent in A is restricted to a single action
- e.g., $d_1(a_1, q_1) = \{\alpha_1^1\}$, $d_1(a_2, q_1) = \{\alpha_1^2\}$, $\ldots$ $d_1(a_k, q_1) = \{\alpha_1^k\}$
- the d_i are such that, if two states q, q' are indistinguishable for an agent a_j, $q \sim_j q'$, then $d_i(a_j, q) = d_i(a_j, q')$, i.e., the agents' strategies are uniform
- the models $M_1, \ldots, M_N$ therefore encode all possible uniform strategies for the coalition A

Model checking $M, q \models_{ir} \langle\langle A \rangle\rangle \varphi$

- for each induced model, M_i, we run the standard ATL model checking algorithm MCHECK(M_i, φ) to label the states $[\langle\langle A \rangle\rangle \varphi]_{M_i}$ in which the uniform strategy encoded by M_i makes $\langle\langle A \rangle\rangle \varphi$ true
- we check whether $q \in [\langle\langle A \rangle\rangle \varphi]_{M_i}$
- for strong uniformity, we check if $\forall q' \in St$ such that $q \sim_A^E q'$ it holds that $q' \in [\langle\langle A \rangle\rangle \varphi]_{M_i}$
- i.e., that the uniform strategy encoded by M_i enforces $\langle\langle A \rangle\rangle \varphi$ in all states indistinguishable from q
- if such M_i is found, $M, q \models_{ir} \langle\langle A \rangle\rangle \varphi$ holds

Non-deterministic algorithm for checking $M, q \models_{ir} \langle\langle A \rangle\rangle \varphi$

- key idea: when dealing with subformulas of the form $\langle\langle A \rangle\rangle \varphi$, *guess* a uniform strategy s_A for A
- remove from M all actions by A not conforming to s_A
- check whether in the resulting model, all states $q' \sim_A^E q$ satisfy $\langle\langle A \rangle\rangle \varphi$

Non-deterministic algorithm complexity

Theorem (Schobbens 2004; Jamroga & Dix 2006)

Model checking ATL_{ir} is Δ_2^P-complete in the number of transitions in the model and the length of the formula.

- the complexity class Δ_2^P (also denoted P^{NP}) means solvable with polynomially many calls to an NP oracle (for guessing strategies; the rest of the computation is polynomial)
- ATL_{ir} model checking is Δ_2^P-complete, so there is no polynomial algorithm (unless $P = \Delta_2^P$ which is even more unlikely than P = NP).