ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240676

Multi-Agent Reinforcement Learning for Alternating-Time Logic

Ernst Moritz Hahn[a,1], **Mateo Perez**[b,1], **Sven Schewe**[c,1], **Fabio Somenzi**[b,1], **Ashutosh Trivedi**[b,*,1] **and Dominik Wojtczak**[c,1]

[a]University of Twente, NL
[b]University of Colorado Boulder, USA
[c]University of Liverpool, UK
ORCID (Ernst Moritz Hahn): https://orcid.org/0000-0002-9348-7684, ORCID (Mateo Perez): https://orcid.org/0000-0003-4220-3212, ORCID (Sven Schewe): https://orcid.org/0000-0002-9093-9518, ORCID (Fabio Somenzi): https://orcid.org/0000-0002-2085-2003, ORCID (Ashutosh Trivedi): https://orcid.org/0000-0001-9346-0126, ORCID (Dominik Wojtczak): https://orcid.org/0000-0001-5560-0546

Abstract. Alternating-time temporal logic (ATL) extends branching time logic by enabling quantification over paths that result from the strategic choices made by multiple agents in various coalitions within the system. While classical temporal logics express properties of "closed" systems, ATL can express properties of "open" systems resulting from interactions among several agents. Reinforcement learning (RL) is a sampling-based approach to decision-making where learning agents, guided by a scalar reward function, discover optimal policies through repeated interactions with the environment. The challenge of translating high-level objectives into scalar rewards for RL has garnered increased interest, particularly following the success of model-free RL algorithms. This paper presents an approach for deploying model-free RL to verify multi-agent systems against ATL specifications. The key contribution of this paper is a verification procedure for model-free RL of quantitative and non-nested classic ATL properties, based on Q-learning, demonstrated on a natural subclass of non-nested ATL formulas.

1 Introduction

Reinforcement learning (RL) [30] is a sampling-based approach to optimal decision-making in which a learning agent converges to an optimal strategy, driven by feedback from its environment in the form of a reward signal. When multiple adversarial agents interact in an environment, the resulting dynamics can be represented by Markov games [22]. Due to the widespread appeal of RL, recent research has emphasised improving the programmability of RL by expressing learning objectives through formal logic [26, 15, 12, 4, 17] or reward machines [17, 16]. In this work, we study *alternating-time temporal logic* (ATL) [2]—an expressive formalism for characterising the strategic capabilities of various coalitions of agents—and introduce an RL approach based on Q-learning, which is tailored for ATL-based requirements.

Temporal Logic and Reinforcement Learning. The classic setup of RL requires a reward-based interface between the environment and the learning agents. While reward-based specification is straightforward in settings with a natural measure of progress, such as video games [27], board games [28], or robot motion planning, it is tedious and error-prone in tasks with a complex nesting of subtasks. Moreover, designing an RL-based solution with rewards might lead to misaligned objectives between the designer and the learning agents, resulting in reward hacking [10, 29, 34] and specification gaming [24, 3]. In response, logic-based specification formalisms such as Linear Temporal Logic (LTL) [26, 15, 12, 4, 17], Computation-Tree Logic (CTL) [33], as well as automata-based characterisations [12, 17, 16, 14] have recently been proposed to express learning requirements in RL. In this setting, a key research question is to design translations from formal requirements to scalar rewards that are faithful (in that maximising reward means maximising probability of achieving the objective) and effective (i.e., RL quickly converges to optimal strategies) [13].

Linear-time logics, like LTL, view a system as one that generates a set of infinite executions. They express properties that hold true over single executions of the system using temporal modalities such as $\bigcirc\varphi$ (meaning the property φ holds in the next step) and $\varphi \,\mathsf{U}\, \varphi'$ (indicating that the property φ remains true until property φ' becomes true). In contrast, branching-time logics, like CTL, interpret the system evolution as a branching tree of possibilities. They incorporate both existential and universal path quantification in addition to the temporal modalities. The branching-time logic CTL* extends both LTL and CTL to provide a more permissive nesting of path quantifications and temporal modalities.

Alternating-Time Temporal Logic. Both LTL and CTL can express properties of *closed* systems, where the system's evolution depends solely on the actions of a single agent. However, when addressing properties of *open* systems, in which multiple agents interact within a shared environment, there is often a need to discuss the capability of a coalition of agents to meet a specific overall specification. Alternating-time temporal logic (ATL) [2] adopts the structure of CTL, but it substitutes path quantifications with quantifications over strategic choices made by particular coalitions of agents. The ATL formula $\langle\!\langle C \rangle\!\rangle \psi$ indicates that the agents in coalition C

* Corresponding Author. Email: ashutosh.trivedi@colorado.edu.
[1] Equal contribution.

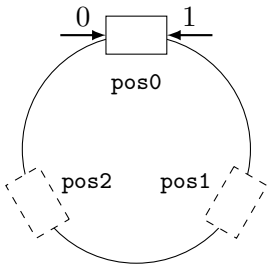

Figure 1. Two robots, 0 and 1, push the carriage around the track. If both push, then the carriage does not move.

can collaborate to ensure ψ is true. Viewed as a game, the agents in the coalition C would always make their decisions first, and the remaining agents are in a better position to take this information into account when making their decisions. Regarding the dual operator $[\![C]\!]\psi$, there have been two versions in the literature: one where the agents in the coalition C cannot ensure that ψ is false, and one where the agents in the coalition C can make sure that ψ holds, provided they move second. The two versions are equally expressive, and we choose the former version in this paper.

We study a generalisation of ATL (probabilistic ATL or PATL) with probabilistic quantification suitable for stochastic environments expressed as probabilistic concurrent games.

Example 1 (Robots and Carriage [5])**.** *Consider two robots pushing a carriage in a circular configuration, as depicted in Figure 1. At each step, both robots can choose to either* push *or* wait*. If both robots decide to push or both decide to wait, the carriage remains stationary. However, if only robot* 0 *pushes, the carriage moves clockwise. Conversely, it moves counter-clockwise if only robot* 1 *pushes. The ability of robot* 0 *to ensure that the carriage eventually arrives at position* `pos1` *can be expressed in ATL as*

$$\langle\!\langle 0 \rangle\!\rangle \Diamond \texttt{pos1}. \tag{1}$$

Similarly, the property that robot 0 *can ensure that position* `pos1` *will never be reached can be expressed as*

$$\langle\!\langle 0 \rangle\!\rangle \Box \neg \texttt{pos1}. \tag{2}$$

Note that property (1) is not valid for the system. Regardless of robot 0*'s strategy (push or wait), robot* 1 *possesses a counter-strategy (push or wait, respectively) that ensures the carriage remains unmoved. On the other hand, the second property is valid for the system when robot* 0 *chooses to* wait *in* `pos0` *and* push *in* `pos2`*.*

Even in a basic two-agent scenario, as shown above, the application of ATL can uncover nuanced details about the system. When the system is known[2], the ATL verification problem is Δ_3^P-complete [20]. When the system is known, but players have imperfect information the same problem becomes undecidable [2, 11], but becomes decidable when players only have finite memory [31]. No algorithm has been developed so far for verifying unknown systems against ATL specifications in the perfect information setting. In this paper, we develop a Q-learning based framework for this problem.

Contributions. We enable reinforcement learning for probabilistic ATL. To achieve this, we first introduce an ATL reachability normal form, which enables a basic ATL formula to be transformed into a turn-based reachability game with reward maximisation objective. This transformation guarantees that maximising reward results in optimal strategies for the ATL objective, for which the threshold in the formula can then be checked. We then introduce two simplifications on the size of the decision space to be learned over. The first is performed by considering transition functions that are decomposed with *outcomes*. The second is by finding antichains over subsets of these outcomes. Finally, we show the effectiveness of our transformation on a few case studies.

To motivate our approach in considering large action spaces with few outcomes, consider the following example.

Example 2. *In second round of the 2023 Turkish presidential elections, each voter had three primary choices: 1) supporting the incumbent, 2) backing the challenger, or 3) neither—whether by abstaining or submitting a blank/invalid vote. With approximately* 64 *million voters participating, the overall action space was immense:* $3^{64,000,000}$*. Yet, the election had only two possible outcomes.*

We show that it is only necessary to consider outcomes, not the entire action space, when learning. Moreover, it may not be necessary to consider all outcomes: this is because outcomes are played out in a short turn-taking game between two coalitions of agents, one of whom tries to maximise the chance of the path property to be satisfied, while the other tries to minimise it. This can be envisioned as the first coalition to move effectively offering a set of outcomes, while the opposing coalition picks from them. We argue that it is enough for the coalition who moves first to offer minimal sets, so that the offering constitutes an antichain[3]; consequently, only those outcomes that appear in any set of this antichain are needed.

Related work. Alternating-Time Temporal Logic (ATL) was first introduced in [2]. Due to its popularity, various extensions of this logic such as ATL*[2] and probabilistic ATL (PATL) [8] were proposed. Strategy Logic (SL) [7, 23] added explicit reasoning about the strategies of all agents, but even the inclusion of strategy context to ATL leads to non-elementary model checking problems [19], though restricting the way that strategy contexts are introduced and revoked in Basic Strategy Interaction Logic (BSIL) brings it down to PSPACE [32]. JMocha [1] was the first model-checking tool to have support for ATL. The PRISM-games tool [18] allows for model-checking against specifications given as rPATL which is probabilistic ATL extended with rewards [9]. MCMAS-SLK [6] tool supports specification language given as Strategy Logic, but due to the undecidability of the model-checking problem of multi-agent systems under perfect recall and incomplete information [2], the tool adopts imperfect recall semantics instead.

Model-free reinforcement learning approach has so far been adapted to handle objectives given as Linear Temporal Logic (LTL) [26, 15, 12, 4, 17], Computation-Tree Logic (CTL) [33], as well as automata-based characterisations [12, 17, 16, 14].

There has been some work (e.g. [21]) in enabling logic-based learning objectives in continuous state/action spaces. These works assume Lipschitz-continuity of the dynamics and provide convergence guarantees based on discretising the system. Since we provide a direct reduction from specifications to reward machines, extension to continuous state space is orthogonal to our approach.

[2] and given as a concurrent game structure (cf. Section 2); there is an alternative standard model, alternating transition systems, for which the complexity is Δ_2^P-complete [20]. Note that if the number of agents is fixed, this problem becomes PTIME-complete[2].

[3] An antichain is any subset of a partially ordered set such that any two distinct elements in the subset are incomparable.

2 Preliminaries

We write $\mathcal{D}(S)$ for the set of discrete distributions over S. A *Markov decision process* (MDP) $\mathcal{M}$ is a tuple $(S, s_0, A, O, T_a, T_o, AP, L)$, where S is a finite set of states, $s_0 \in S$ is the initial state, A is a finite set of *actions*, O is a finite set of *outcomes*, $T_a\colon S{\times}A{\to}O$ is the *outcome function*, $T_o\colon O{\to}\mathcal{D}(S)$ is the probabilistic transition function, AP is the set of *atomic propositions*, and $L\colon S \to 2^{AP}$ is the *labelling function*. We sometimes use the (combined) transition function $T = T_o \circ T_a$. For a state $s \in S$, we let $A(s)$ denote the set of actions that can be selected in s and let $O(s) = \{T_a(s,a) \mid a \in A(s)\}$ denote their outcomes.

An MDP is a Markov chain if $O(s)$ is singleton for all $s \in S$. For states $s, s' \in S$ and $a \in A(s)$, $T(s,a)(s')$ equals $\Pr(s'|s,a)$. A *run* of $\mathcal{M}$ is an ω-word $\langle s_0, a_1, s_1, \ldots\rangle \in S \times (A \times S)^\omega$ such that $\Pr(s_{i+1}|s_i, a_{i+1}){>}0$ for all $i \geq 0$. A finite run is a finite such sequence. For a *run* $r = \langle s_0, a_1, s_1, \ldots\rangle$ we define the corresponding labelled run as $L(r) = \langle L(s_0), L(s_1), \ldots\rangle \in (2^{AP})^\omega$. We write $Runs^{\mathcal{M}}(FRuns^{\mathcal{M}})$ for the set of runs (finite runs) of the MDP $\mathcal{M}$ and $Runs^{\mathcal{M}}(s)(FRuns^{\mathcal{M}}(s))$ for the set of runs (finite runs) of the MDP $\mathcal{M}$ starting from the state s.

A *strategy* is a function $\sigma\colon S \to A$ such that $\sigma(s) \in A(s)$. Let $Runs^{\mathcal{M}}_\sigma(s)$ denote the subset of runs $Runs^{\mathcal{M}}(s)$ that correspond to strategy σ with initial state s. Let $\Sigma_{\mathcal{M}}$ be the set of all strategies. An MDP $\mathcal{M}$ under a strategy σ results in a Markov chain $\mathcal{M}_\sigma$. The behaviour of an MDP $\mathcal{M}$ under a strategy σ and starting state $s \in S$ is defined on a probability space

$$(Runs^{\mathcal{M}}_\sigma(s), \mathcal{F}_{Runs^{\mathcal{M}}_\sigma(s)}, \Pr^{\mathcal{M}}_\sigma(s))$$

over the set of infinite runs of σ with starting state s. Given a random variable $f\colon Runs^{\mathcal{M}} \to \mathbb{R}$, we denote by $\mathbb{E}^{\mathcal{M}}_\sigma(s)\,\{f\}$ the expectation of f over the runs of $\mathcal{M}$ originating at s that follow σ.

For MDPs, it is unusual to include the *outcome* as a separate entity—although this is not a restriction: one can always use the trivial identity outcome function, and treat T_o as the standard transition function. What is common is to consider a situation where two or more actions in the same state lead to the same outcome ($T_a(s,a) = T_a(s',a')$); for MDPs, this is normally modelled as identifying these actions, which is reflected in the set of available actions being different for different states. Our reason to make the outcomes primary citizens is that they are useful for the extension to Probabilistic Concurrent Game Structures (PCGSs), where different coalitions effectively play out the outcome, so that identifying actions is not as powerful as identifying outcomes.

A *Probabilistic Concurrent Game Structure* (PCGS) $\mathcal{P}$ is an extended MDP $(S, s_0, A, O, T_a, T_o, AP, L; D, \{A_d \mid d \in D\})$, where the tuple $(S, s_0, A, O, T_a, T_o, AP, L)$ is an MDP, D a finite set of **D**ecision makers or agents, where each agent d has their own set of actions A_d, and where the set of joint actions is made up of the individual actions taken by the agents: $A = \bigtimes_{d\in D} A_d$.

A (non-probabilistic) Concurrent Game Structure (CGS) is a PGCS where T_o maps to probability distributions with singleton support, so that it can be viewed as a mapping $O \to S$. In other words, in a CGS the outcome determines the successor state.

3 Problem Definition

Alternating time logic [2] (ATL) is a specification mechanism to reason about the collaborative power of agents—the decision makers D in our probabilistic CGS—to enforce temporal properties. The underlying temporal logic is like CTL; indeed, in a non-probabilistic setting CTL would be the corner case where $D = \{d\}$ is unary, and 'for all paths' relates to 'the empty set of agents can enforce' while 'there is a path' relates to 'the single agent d can enforce'.

3.1 ATL and Reachability Normal Form

Our grammar presents ATL in what we refer to as *reachability normal form*, in that it only includes path properties that are essentially reachability properties. In the original definition of ATL, $[\![C]\!]\psi$ did not exist, but there were more temporal operators, including operators like weak until that are essentially safety properties.

We chose a reachability normal form because reinforcement learning algorithms can readily be used against reachability objectives. While reachability normal form is a novel contribution, related ideas have been explored elsewhere. In particular, the negative normal form of ATL formulas is frequently used, where negation is pushed to the leaves (literals). This is done by using that the dual temporal operators are (there) always available, and the new $[\![C]\!]$ operator is dual to $\langle\!\langle C\rangle\!\rangle$. This allows for pushing negations to the leaves, e.g. using $\neg\langle\!\langle C\rangle\!\rangle\Box\varphi = [\![C]\!]\Diamond\neg\varphi$.

ATL needs two of the following: (1) negation (not only at leaves), (2) dual strategy quantifiers, (3) dual temporal operators. We use (1 & 2), as this is closest related to maximising probabilities in RL, the original definition of ATL used (1 & 3), and negative normal form uses (2 & 3). The latter is good when translating it to automata (which we do not need). We have simply created the normal form that is useful for RL, which is not difficult, but useful and valuable.

Syntax. ATL formulas over atomic propositions AP and agent set D can be expressed using the following grammar:

$$\begin{aligned} \varphi &::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \langle\!\langle C\rangle\!\rangle\psi \mid [\![C]\!]\psi \\ \psi &::= \bigcirc\varphi \mid \varphi\,\mathsf{U}\,\varphi. \end{aligned}$$

Here $p \in AP$ is an atomic proposition and $C \subseteq D$ a coalition of agents in D. One can define the usual syntactic sugar over the aforementioned operators, such as the "eventually" modality $\Diamond$, which is defined as $\Diamond\varphi = \mathsf{true}\,\mathsf{U}\,\varphi$.

Semantics. The meaning of $\langle\!\langle C\rangle\!\rangle\psi$ is that the agents in C can jointly enforce the path property ψ if they move first in every round (not seeing what the agents in $D\backslash C$ have done), while $[\![D\backslash C]\!]\psi$ means that the agents in C can jointly enforce ψ if they move second, i.e., after seeing what the agents in $D \setminus C$ have done.[4]

Formally, the semantics of ATL on a given CGS are defined inductively as follows:

1. For an atomic proposition $p \in AP$ and a state s of PCGS, we have that $s \models p$ iff $p \in L(s)$.
2. Boolean connectives are treated as usual.
3. For a path formula $\psi = \bigcirc\varphi$ (read: next φ), a path $r = \langle s_0, a_1, s_1, \ldots\rangle$ models $\bigcirc\varphi$ (i.e., $r \models \psi$) if, and only if, $s_1 \models \varphi$.
4. For a path formula $\varphi\,\mathsf{U}\,\varphi'$, a path $s_0, s_1, s_2, \ldots$ models $\varphi\,\mathsf{U}\,\varphi'$ if, and only if, there exists an $i \geq 0$ with $s_i \models \varphi'$ and $s_j \models \varphi$ for all $j < i$.
5. For a coalition $C \subseteq D$, a strategy $\sigma\colon FRuns \to \bigtimes_{d\in C} A_d$, and a strategy $\tau\colon FRuns \times \bigtimes_{d\in C} A_d \to \bigtimes_{d\in D\setminus C} A_d$, we denote the path starting in $s \in S$ and created by C choosing their actions according to σ and $D \setminus C$ choosing theirs in according to τ by $r(s, C, \sigma, \tau)$. We then have

[4] Note this is equivalent to that the agents in the coalition $D\backslash C$ cannot ensure that ψ is false when they move first in every round.

- $s \models \langle\!\langle C \rangle\!\rangle \psi$ if, and only if, there exists $\sigma\colon FRuns \to \bigtimes_{d\in C} A_d$ such that, for all $\tau\colon FRuns \times \bigtimes_{d\in C} A_d \to \bigtimes_{d\in D\setminus C} A_d$, we have that $r(s, C, \sigma, \tau) \models \psi$; and
- $s \models [\![D\setminus C]\!]\psi$ if, and only if, for all $\sigma\colon FRuns \to \bigtimes_{d\in C} A_d$, there exists $\tau\colon FRuns \times \bigtimes_{d\in C} A_d \to \bigtimes_{d\in D\setminus C} A_d$ such that we have $r(s, C, \sigma, \tau) \models \psi$.

3.2 Probabilistic Alternating-Time Logic (PATL)

PATL extends ATL by allowing probabilistic bounds on coalition quantifications, leading to the following grammar:

$$\begin{aligned} \varphi &::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \langle\!\langle C \rangle\!\rangle_{\bowtie q}\psi \mid [\![C]\!]_{\bowtie q}\psi \\ \psi &::= \bigcirc\varphi \mid \varphi \,\mathsf{U}\, \varphi. \end{aligned}$$

where $q \in [0, 1]$ and $\bowtie \in \{>, \geq\}$. Note that this retains the existence of dual operators, so that a translation of a specification given in a different variant of PATL to our reachability normal form remains straightforward.

A PCGS $\mathcal{P}$ under a pair of strategies σ and τ for coalitions C (who move first) and $D\setminus C$ (who move second) results in a Markov chain $\mathcal{P}_{\sigma,\tau}$. If σ and τ are finite memory strategies, then $\mathcal{P}_{\sigma,\tau}$ is a finite-state Markov chain. The behaviour of a PCGS $\mathcal{P}$ under strategies σ and τ from starting state $s \in S$ is defined on a probability space $(Runs^{\mathcal{P}}_{\sigma,\tau}(s), \mathcal{F}_{Runs^{P}p_{\sigma,\tau}(s)}, \Pr^{\mathcal{P}}_{\sigma,\tau}(s))$ over the set of infinite runs of σ with starting state s.

The semantics of PATL follows those of ATL with the only difference that we view a path formula ψ as a random variable $\psi\colon Runs^{\mathcal{P}} \to \{0, 1\}$, where we denote by $\mathbb{E}^{\mathcal{P}}_{\sigma,\tau}(s)\,\{\psi\}$ the expectation of ψ over the runs of $\mathcal{P}$ originating at s that follow σ and τ. We then have

- $s \models \langle\!\langle C \rangle\!\rangle_{\bowtie q}\psi$ if, and only if, there exists $\sigma\colon FRuns \to \bigtimes_{d\in C} A_d$ such that for all $\tau\colon FRuns \times \bigtimes_{d\in C} A_d \to \bigtimes_{d\in D\setminus C} A_d$ we have

$$\mathbb{E}^{\mathcal{P}}_{\sigma,\tau}(s)\,\{\psi\} \bowtie q$$

- $s \models [\![D\setminus C]\!]\psi$ if, and only if, for all $\sigma\colon FRuns \to \bigtimes_{d\in D\setminus C} A_d$ there exists $\tau\colon FRuns \times \bigtimes_{d\in D\setminus C} A_d \to \bigtimes_{d\in C} A_d$ such that

$$\mathbb{E}^{\mathcal{P}}_{\sigma,\tau}(s)\,\{\psi\} \bowtie q.$$

for $\bowtie \in \{>, \geq\}$ and $q \in [0, 1]$.

Classic ATL Objectives. We define a (classic) ATL formula as *basic* if it contains only one strategy quantifier and this quantifier is at the root of the formula, such as in $\langle\!\langle C \rangle\!\rangle p \,\mathsf{U}\, p'$, where $p, p' \in AP$ are atomic propositions. In this example, the goal of the coalition C is to maximise the likelihood of the path formula $p \,\mathsf{U}\, p'$ to hold. This probability then relates to quantitative ATL in that it can simply be compared with the threshold value. For example, if the maximial probability with which the coalition C can achieve $p \,\mathsf{U}\, p'$ is 0.5, then $\langle\!\langle C \rangle\!\rangle_{>0.2} p \,\mathsf{U}\, p'$ and $\langle\!\langle C \rangle\!\rangle_{\geq 0.5} p \,\mathsf{U}\, p'$ hold, while $\langle\!\langle C \rangle\!\rangle_{>0.5} p \,\mathsf{U}\, p'$ and $\langle\!\langle C \rangle\!\rangle_{\geq 0.7} p \,\mathsf{U}\, p'$ do not. This framework enables us to simplify the solution for basic PATL formulas to solving a reachability game.

4 Reward Translation for PATL

In an environment based on a probabilistic concurrent game structure (PCGS) with a probabilistic ATL specification, our aim is to leverage model-free reinforcement learning algorithms to assess the system against a PATL objective. The central challenge here is translation. From a given PATL specification, we want to derive a reward function where the expected maximal reward mirrors the expected probability of meeting the objective. In this section, we first demonstrate how to devise rewards for classic ATL objectives when viewed as a goal to maximise the chances that the temporal path formula holds. Following that, we outline how to broaden this approach to more general specifications.

4.1 Classic ATL Objectives

Observe that the problem of finding a strategy against a qualitative ATL (QATL) objective, where one set of agents tries to maximise the chance that a path formula holds while the remaining agents try to minimise it, reduces to a turn-based reachability game, where:

1. one coalition of agents decides their respective individual actions;
2. then, the remaining agents, comprising the other coalition, decide their respective individual actions;
3. then, the outcome of these actions is determined;
4. and finally, the random move determined by this outcome is made in the probabilistic concurrent game structure.

We have three types of objectives: next-step ($\bigcirc\varphi$ or one step reachability), boolean objectives (that stem from next-step objectives), and constrained reachability (*until* path formulas). Designing rewards so that the expected maximal reward equals the expected probability to meet the objective is simple for all three types of objectives, where the expected payoff reflects the chances of meeting the objective:

1. For a boolean objective φ, the chances of meeting it—and thus the payoff—from a state s is 1, if φ is satisfied by the labels in the current state s ($L(s) \models \varphi$), and 0 otherwise.
2. For a path formula $\bigcirc\varphi$, and for an outcome o, the chance of meeting the objective $\bigcirc\varphi$ is the likelihood that the selected successor s satisfies φ (i.e. $\sum_{s'\in S, s'\models\varphi} T_o(o)(s')$).
   For a state s, the expected likelihood of meeting the objective defined by the path formula $\bigcirc\varphi$ is then defined by the outcome of a two-step game, where one of the players aims for the outcome that the minimises the expected likelihood to meet the objective, while the other player aims for the outcome that maximises it (see below for details).
3. For a path formula $\varphi \,\mathsf{U}\, \varphi'$, the likelihood to meet it, and thus the payoff, from a state s is:
   (a) 1 if φ' is satisfied in s ($L(s) \models \varphi'$).
   (b) Otherwise, it is 0 if φ is not satisfied ($L(s) \not\models \varphi$) in s.
   (c) If neither is the case—i.e., if $L(s) \models \varphi$ and $L(s) \not\models \varphi'$—then the chance of meeting the objective of the path quantifier is determined by the outcome o reached. For a given outcome o, the chance of meeting the objective $\varphi \,\mathsf{U}\, \varphi'$ is the likelihood that a path starting at the selected successor s' satisfies $\varphi \,\mathsf{U}\, \varphi'$ (i.e. $\sum_{s'\in S} T_o(o)(s') \cdot \mathbb{E}^{\mathcal{P}}(s')$).[5]

Under this description, we arrive at a turn-based (reachability) game with payoffs of 0 or 1. We can reduce this into a total reward

[5] Strictly speaking, this does not quite define the payoff, as it would not say what the payoff is on paths that contain only states that satisfy φ, but no state that satisfies φ'. As these paths do not satisfy the reachability objective, they are assigned a payoff of 0.

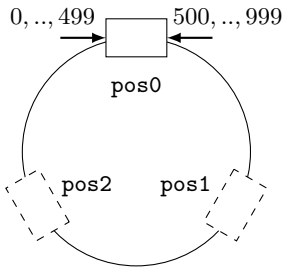

Figure 2. 1,000 robots push a carriage around a track: 500 from one side and the rest from the other. If the same number of robots push from each side, the carriage remains stationary; otherwise, it moves in the direction of the larger pushing team.

problem by giving a reward of 1 when the payoff is 1 and 0 otherwise. Since we consider finite-state games, selecting a large enough discount factor will result in discounted reward optimal strategies being optimal for total reward [25]. With this reduction, any RL algorithm for discounted reward Markov games can then be used. For example, we utilise minimax-Q learning [22] in our experiments.

4.2 Simplification via Outcomes and Antichains

The reduction so far considers controlling a PCGS by reasoning directly on its action space. Next, we examine how reasoning about outcomes, and specifically, antichains of subsets of outcomes, provides a benefit in reducing the decision space. These are optimisations that work best when the relevant outcomes are small compared to the action space.

We begin by describing the correspondence between actions and outcomes. In the evolution of a PCGS, the selection of an action by the first coalition restricts the possible outcomes to a particular subset of outcomes. This subset is then offered to the second coalition, where their choice of action results in a selection of the outcome from this subset. Thus, reasoning about the first coalition selecting restricted subsets of outcomes and the second coalition selecting an outcome from this subset is equivalent to reasoning on the actions.

From this correspondence between actions and outcomes, one can identify a new sets of actions that correspond to possible subsets of outcomes for the first coalition and outcomes for the second coalition. As illustrated in the following generalisation of Example 1, distributed selection of joint actions in PCGSs can result in vast action spaces with few distinct outcomes. This provides our first potential simplification of the decision space.

Example 3 (1,000 Robots and Carriage)**.** *Consider the following modification of Example 1. Instead of just two robots, we have now 500 robots pushing the carriage from the left and another 500 from the right in a circular configuration, as depicted in Figure 2. At each step, any of the robots can choose to either* push *or* wait*. If the number of robots pushing from each side is the same then the carriage remains stationary, otherwise it moves in the direction where more robots pushing.*

As each robot has two possible options and there are 1,000 of them, the size of the overall action space is huge: $2^{1,000}$*. However, in each round there are only three possible outcomes: the carriage moves left, right, or stays in place.*

Our second simplification comes from noting that not all subsets of outcomes need to be considered by the first coalition. Specifically, the first coalition can always offer fewer outcomes to its adversary, if possible, without diminishing optimality. This follows from the principle that the second coalition minimising/maximising over a subset will result in a value which is greater/less than or equal to the value over the superset, i.e., giving fewer options potentially moves the value of the game in favour of the first coalition.

In Example 3, assume that the coalition to choose first consists of the 500 robots on the left of the carriage. They have three choices in the abstracted form of offering sets of outcomes: they can offer

1. the two outcomes to move left or to stay;
2. the two outcomes to move right or stay; or
3. all three outcomes, to more left, to move right, or to stay.

The first option is achieved by none of the robots pushing, the second by all of them pushing, and the third by any number between 1 and 499 of these robots pushing.

We argue that the agents only need to consider those options that offer the minimial amount of choice, in this case options (1) and (2), but can discard option (3). This is because the two coalitions are playing out a 0-sum game, and offering more choice to a rational opponent in a 0-sum game cannot improve the outcome. Finding the set of subsets of outcomes that offers the minimal amount of choice corresponds to finding an antichain as we describe next.

We now formalise our two simplifications. For $s \in S$, an antichain $\mathcal{O} \subseteq 2^{O(s)}$ of outcomes is a non-empty set of non-empty subsets of $O(s)$, such that no two sets in $\mathcal{O}$ contain each other ($\forall O_1, O_2 \in \mathcal{O}.\ O_1 \subseteq O_2 \Rightarrow O_1 = O_2$). The antichain for a coalition $C \subseteq D$ that chooses first, denoted $\mathcal{O}(s, C)$, is an antichain such that

- for all $O' \in \mathcal{O}(s, C)$, there is $a \in \bigtimes_{d \in C} A_d$ such that $O' = \bigcup_{a' \in \bigtimes_{d \in D \setminus C} A_d} T_a(s, (a, a'))$, and
- for all $a \in \bigtimes_{d \in C} A_d$ with $O' = \bigcup_{a' \in \bigtimes_{d \in D \setminus C} A_d} T_a(s, (a, a'))$ there is a set $O'' \subseteq O'$ in $\mathcal{O}(s, C)$ (i.e., $O' \supseteq O'' \in \mathcal{O}(s, C)$).

We would now like to show that replacing the actions of the first coalition with $\mathcal{O}(s, C)$ and the actions of the second coalition with O preserves correctness (optimality). To show this, we simply need to show that the computation of the value functions used in RL remains the same with this change. We show this in the following theorem.

Theorem 1. *For any value function* $\mathsf{val}\colon O \to \mathbb{R}$ *on the outcomes, we have that*

$$\min_{a \in \bigtimes_{d \in C} A_d} \max_{a' \in \bigtimes_{d \in D \setminus C} A_d} \mathsf{val}(T_a(s, (a, a'))) = \min_{O' \in \mathcal{O}(s,C)} \max_{o \in O'} \mathsf{val}(o) \quad \textit{and}$$

$$\max_{a \in \bigtimes_{d \in C} A_d} \min_{a' \in \bigtimes_{d \in D \setminus C} A_d} \mathsf{val}(T_a(s, (a, a'))) = \max_{O' \in \mathcal{O}(s,C)} \min_{o \in O'} \mathsf{val}(o) \quad \textit{hold.}$$

Proof. For the proof, we will only show the first case. The second case follows symmetrically. Our proof proceeds in two steps. First, we show the correspondence between actions and subsets of outcomes. Second, we show that restricting to antichains does not affect the resulting value.

Let $f(s, a, C)$ denote the subset of outcomes that the coalition C induces when they select action $a \in \bigtimes_{d \in C} A_d$ in state s, defined as:

$$f(s, a, C) = \bigcup_{a' \in \bigtimes_{d \in D \setminus C} A_d} T_a(s, (a, a'))\ .$$

Now, let $\mathcal{O}_{\text{all}}(s,C) = \{f(s,a,C) | a \in \times_{d\in C} A_d\}$ be the set of all subsets of outcomes that the first coalition can induce. By expanding the definition of $f(s,a,C)$, we get that $\{\mathsf{val}(o) | o \in f(s,a,C)\} = \{\mathsf{val}(T_a(s,(a,a'))) | a' \in \times_{d\in D\setminus C} A_d\}$. In other words, the set of values over actions of the maximising coalition is the same as the set of values over the outcomes subset. Thus, their maximum is the same, and we get that

$$\min_{a\in\times_{d\in C} A_d} \max_{a'\in\times_{d\in D\setminus C} A_d} \mathsf{val}(T_a(s,(a,a'))) = \min_{O'\in\mathcal{O}_{\text{all}}(s,C)} \max_{o\in O'} \mathsf{val}(o) \ .$$

Finally, we would like to show that

$$\min_{O'\in\mathcal{O}_{\text{all}}(s,C)} \max_{o\in O'} \mathsf{val}(o) = \min_{O'\in\mathcal{O}(s,C)} \max_{o\in O'} \mathsf{val}(o) \ .$$

To show that this is the case, we need to show that, for every $O' \in \mathcal{O}_{\text{all}}(s,(a,a')) \setminus \mathcal{O}(s,(a,a'))$, there exists an $O'' \in \mathcal{O}(s,(a,a'))$ that was an equally good or better choice for the minimisation, i.e. $\max_{o\in O'} \mathsf{val}(o) \geq \max_{o\in O''} \mathsf{val}(o)$. By the definition of $\mathcal{O}(s,(a,a'))$, if $O' \in \mathcal{O}_{\text{all}}(s,(a,a')) \setminus \mathcal{O}(s,(a,a'))$ then there exists a set $O'' \subset O'$ such that $O'' \in \mathcal{O}(s,(a,a'))$. Since $O'' \subset O'$, we get that $\max_{o\in O'} \mathsf{val}(o) \geq \max_{o\in O''} \mathsf{val}(o)$. □

Minimax-Q Learning. To further illustrate our approach, we will describe how to instantiate minimax-Q learning with our outcome centric approach. Minimax-Q learning [22] is a variant of Q-learning designed for Markov games. After observing a transition from state s to s' under action a with reward r, the Q-values are updated by the following update rule

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \operatorname*{opt}_{a'\in A(s')} Q(s',a')) \ ,$$

where opt is max (min) if s' is controlled by max (min) player, $\gamma \in [0,1[$ is the discount factor, and $\alpha \in]0,1[$ is the learning rate. Our QATL formulas take one of two forms:

- a *maximin* formula for $\langle\!\langle C\rangle\!\rangle$ QATL formulas, where the first coalition to choose, C, plays as max player
- a *minimax* formula for $[\![D\setminus C]\!]$ QATL formulas, where the first coalition to choose, C, plays as min player.

The form of the formula determines which coalition is assigned max player and which is assigned min player. The first coalition selects an action $O \in \mathcal{O}(s,C)$ and updates the corresponding Q-value as

$$Q(s,O) \leftarrow (1-\alpha)Q(s,O) + \alpha\gamma \operatorname*{opt}_{o'\in O} Q((s,O),o') \ .$$

The second coalition then selects an action $o \in O$ and updates the corresponding Q-value as

$$Q((s,O),o) \leftarrow (1-\alpha)Q((s,O),o) + \alpha\gamma \operatorname*{opt}_{O'\in\mathcal{O}(s',C)} Q(s',O')$$

if the reachability game has not terminated, where s' is the observed next state, and

$$Q((s,O),o) \leftarrow 1$$

if the reachability game terminates. This final update for when the reachability game terminates simply sets the Q-value to the payoff directly instead of waiting for the standard update rule to converge.

4.3 Extension to full PATL

There are two steps involved in getting from QATL to PATL. A first step is to consider basic PATL. For the consideration of basic PATL, we can simply solve the QATL problem as described in the previous section and determine whether or not the probability is above the threshold. While this is in principle straightforward, we face the same problem all probabilistic branching time logics with probability threshold face for learning: when the real probability we try to find for a QATL formula is some probability $q \in]0,1[$, then determining whether the related basic PATL formula $\langle\!\langle C\rangle\!\rangle_{\geq q}$ or $\langle\!\langle C\rangle\!\rangle_{>q}$ resp. $[\![C]\!]_{\geq q}$ or $[\![C]\!]_{>q}$ hold will not be possible even in the limit. Once the basic PATL is covered, the extension to full PATL is straightforward: we can simply build the formula tree up from the leaves to the root, successively replacing the subformulas that start with a path quantifier (those starting with $\langle\!\langle C\rangle\!\rangle_{\geq q}$, $\langle\!\langle C\rangle\!\rangle_{>q}$, $[\![C]\!]_{\geq q}$, or $[\![C]\!]_{>q}$) by extending the labelling function, treating the evaluated subformulas as atomic propositions.

We make note of two points that change in this general setting. First, the set of antichains of outcomes depend on the coalition C. If we need to evaluate different coalitions that move first, we will have different antichains, and thus different sets of relevant outcomes to expand. Secondly, the direct evaluations—cases (1), (3a), and (3b) from Section 4.1—can rely on the truth of subformulas that start with a path quantifier. We have followed the strategy to evaluate them first, though it would also be possible to allow for their valuation to change over time, so long as they eventually stabilise.

We also note that, for full PATL, there is the additional step of comparing the predicted value to a threshold. When the objective exactly meets the threshold, there cannot be any convergence guarantees, and it stands to be expected that the truth value fluctuates, e.g. where a coalition C can enforce $\Box p$ when moving first, $\langle\!\langle C\rangle\!\rangle_{\geq 0.5}\Box p$ should be true and $\langle\!\langle C\rangle\!\rangle_{>0.5}\Box p$ should be false, while the observed probability will fluctuate around 0.5, so that the truth value of both formulas are likely to fluctuate. This does not pose a problem for simple PATL formulas as this is well enough expressed by the truth value converging towards 0.5, but for nested PATL formulas, changing truth values from subformulas have a knock-on effect on the satisfaction valuation of the path formula.

5 Experimental Results

We use minimax-Q learning as our reinforcement learning procedure to learn to satisfy QATL formulas. For our experiments, we use the discount factor $\gamma = 0.999$, a learning rate of $\alpha = 0.1$, and an ε-greedy strategy with $\varepsilon = 0.1$. We reset the episode when its length exceeds 100 steps. To encourage exploration, we initialise the Q-table values to 0.01 instead of zero. Our learning results are in Table 1. We report the example name, whether the QATL property is satisfied or unsatisfied, the number of states reachable in the resulting reachability game, the number of training episodes, and the training walltime averaged over three runs.

Robots and Carriage. The first case study we consider is the robots and carriage example introduced in Section 1 (see Figure 1). We consider the two properties $\langle\!\langle 0\rangle\!\rangle_{\geq 0.5}\Diamond\texttt{pos1}$ and $\langle\!\langle 0\rangle\!\rangle_{\geq 0.5}\Box\neg\texttt{pos1}$, referred to as `robotsF` and `robotsG`, respectively. After training for 1k episodes, the agent correctly learns the strategies discussed in Section 1. We also consider a variant of this example where with probability 0.01, the robots slip when trying to push, causing the robot to instead behave as if they waited (referred to as `robotsF-slip` and `robotsG-slip` for the two corresponding

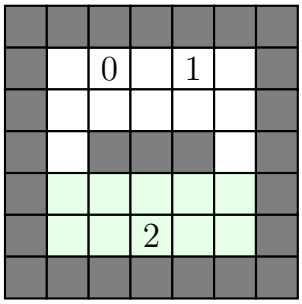

Figure 3. There are three agents denoted 0, 1, and 2. The goal region is denoted in green and the walls are denoted in grey. We consider the objectives $\langle\langle 0 \rangle\rangle_{\geq 0.5} \neg c \, \mathsf{U} (g \wedge \neg c)$ and $\langle\langle 0, 1 \rangle\rangle_{\geq 0.5} \neg c \, \mathsf{U} (g \wedge \neg c)$ where c denotes a collision and g denotes either 0 or 1 have reached the goal region.

Table 1. Learning results. Times are in seconds.

Example	result	states	episodes	time
`robotsF`	unsat.	6	1k	0.05
`robotsG`	sat.	6	1k	0.05
`robotsF−slip`	sat.	6	1k	0.07
`robotsG−slip`	unsat.	6	1k	0.05
`guardsAlone`	unsat.	649528	50k	5.65
`guards`	sat.	649528	75k	6.45

properties). Under this configuration, the satisfiability is the opposite of what is was before. This is because either robot can now force every state to be visited by waiting for the other robot to slip. Learning takes less than 0.1 seconds of wall-time for these examples.

Gridworld Guards. Next, we consider the example shown in Figure 3. In this example, there are three agents: 0, 1, and 2. Each agent can move in any of the cardinal directions, but will not move if it tries to move into a wall (shown in grey). If any two agents occupy the same cell, we say that they have collided (c is true). If agents 0 or 1 reach the green goal region, then we say that the goal has been reached (g is true). We consider the property that the goal is reached and that no collision has occurred at any timestep before and including the timestep the goal is reached. This is denoted by the path formula $\neg c \, \mathsf{U} (g \wedge \neg c)$. First, we consider if agent 0 can enforce this alone, i.e. $\langle\langle 0 \rangle\rangle_{\geq 0.5} \neg c \, \mathsf{U} (g \wedge \neg c)$ (referred to as `guardsAlone`). We train for 50k training episodes. The learner finds that agents 1 and 2 can collaborate to prevent agent 0 from reaching the green region without collision by having agent 2 go to the left gap and agent 1 to the right gap. Next, we consider if agents 0 and 1 can enforce this, i.e. $\langle\langle 0, 1 \rangle\rangle_{\geq 0.5} \neg c \, \mathsf{U} (g \wedge \neg c)$ (referred to as `guards`). The learned strategy enforces this, as agent 2 cannot stop both agents 0 and 1 on its own. All training instances of this case study took less than 10 seconds of wall-time.

6 Conclusion

This paper has developed a reinforcement learning (RL) based approach for probabilistic alternating-time logic. The *reachability normal form* for PATL, introduced in this paper, allows us to transform basic PATL formulas into a turn-based (reachability) game with a reward maximisation objective, enabling the use of RL. We discussed how this approach can be extended to arbitrary PATL formula.

The paper also introduced two methods for reducing the decision space needed for learning. The first method is to formulate the transition function in terms of *outcomes*. We showed that there is a direct correspondence between actions and outcomes, letting us utilise outcomes directly for learning. As there may be fewer outcomes than actions, this provides a potential savings in the size of the decision space. The second method for reducing the decision space is to remove subsets of outcomes that are not minimal. In other words, we show that antichains of subsets of outcomes are all that needs to be considered for optimality. Finally, we demonstrated the effectiveness of our approach on a few case studies.

Acknowledgements

This work was supported in part by the EPSRC through grants EP/X017796/1, EP/X021513/1, EP/X03688X/1, and EP/X042596/1; the NSF through grant CCF-2009022 and the NSF CAREER award CCF-2146563; and the EU's Horizon 2020 research and innovation programme under grant agreement No 864075 (CAESAR) and under the Marie Skłodowska-Curie grant agreement No 101008233 (MISSION).

References

[1] R. Alur, L. De Alfaro, R. Grosu, T. A. Henzinger, M. Kang, C. M. Kirsch, R. Majumdar, F. Mang, and B.-Y. Wang. jmocha: A model checking tool that exploits design structure. In *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*, pages 835–836. IEEE, 2001.

[2] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM (JACM)*, 49(5):672–713, 2002.

[3] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

[4] A. K. Bozkurt, Y. Wang, and M. Pajic. Model-free learning of safe yet effective controllers. In *2021 60th IEEE Conference on Decision and Control (CDC), Austin, TX, USA, December 14-17, 2021*, pages 6560–6565. IEEE, 2021. doi: 10.1109/CDC45484.2021.9683634. URL https://doi.org/10.1109/CDC45484.2021.9683634.

[5] N. Bulling and W. Jamroga. Comparing variants of strategic ability: how uncertainty and memory influence general properties of games. *Autonomous agents and multi-agent systems*, 28:474–518, 2014.

[6] P. Čermák, A. Lomuscio, F. Mogavero, and A. Murano. Practical verification of multi-agent systems against SLK specifications. *Information and Computation*, 261:588–614, 2018.

[7] K. Chatterjee, T. A. Henzinger, and N. Piterman. Strategy logic. *Information and Computation*, 208(6):677–693, 2010.

[8] T. Chen and J. Lu. Probabilistic alternating-time temporal logic and model checking algorithm. In *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, volume 2, pages 35–39. IEEE, 2007.

[9] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. Automatic verification of competitive stochastic systems. *Formal Methods in System Design*, 43:61–92, 2013.

[10] J. Clark and D. Amodei. Faulty Reward Functions in the Wild. https://openai.com/blog/faulty-reward-functions/, 2016. Accessed on: 01/18/2023.

[11] C. Dima and F. L. Tiplea. Model-checking atl under imperfect information and perfect recall semantics is undecidable. *arXiv preprint arXiv:1102.4225*, 2011.

[12] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak. Omega-regular objectives in model-free reinforcement learning. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 395–412, 2019. LNCS 11427.

[13] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak. Faithful and effective reward schemes for model-free reinforcement learning of omega-regular objectives. In D. V. Hung and O. Sokolsky, editors, *Automated Technology for Verification and Analysis*, pages 108–124, Cham, 2020. Springer International Publishing.

[14] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak. Omega-regular reward machines. In *ECAI 2023 - 26th European Conference on Artificial Intelligence*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 972–979. IOS Press, 2023.

[15] M. Hiromoto and T. Ushio. Learning an optimal control policy for a Markov decision process under linear temporal logic specifications. In

Symposium Series on Computational Intelligence, pages 548–555, Dec. 2015.

[16] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pages 2107–2116, 2018.

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

[18] M. Kwiatkowska, G. Norman, D. Parker, and G. Santos. PRISM-games 3.0: Stochastic game verification with concurrency, equilibria and time. In *Computer Aided Verification: 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part II 32*, pages 475–487. Springer, 2020.

[19] F. Laroussinie and N. Markey. Augmenting ATL with strategy contexts. *Inf. Comput.*, 245:98–123, 2015. doi: 10.1016/j.ic.2014.12.020. URL https://doi.org/10.1016/j.ic.2014.12.020.

[20] F. Laroussinie, N. Markey, and G. Oreiby. On the expressiveness and complexity of ATL. *Log. Methods Comput. Sci.*, 4(2), 2008. doi: 10.2168/LMCS-4(2:7)2008. URL https://doi.org/10.2168/LMCS-4(2:7)2008.

[21] A. Lavaei, F. Somenzi, S. Soudjani, A. Trivedi, and M. Zamani. Formal controller synthesis for continuous-space mdps via model-free reinforcement learning. In *11th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS 2020, Sydney, Australia, April 21-25, 2020*, pages 98–107. IEEE, 2020. doi: 10.1109/ICCPS48487.2020.00017. URL https://doi.org/10.1109/ICCPS48487.2020.00017.

[22] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[23] F. Mogavero, A. Murano, G. Perelli, and M. Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Transactions on Computational Logic (TOCL)*, 15(4):1–47, 2014.

[24] A. Pan, K. Bhatia, and J. Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. *arXiv preprint arXiv:2201.03544*, 2022.

[25] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994. ISBN 978-0-47161977-2. doi: 10.1002/9780470316887. URL https://doi.org/10.1002/9780470316887.

[26] D. Sadigh, E. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia. A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications. In *Conference on Decision and Control (CDC)*, pages 1091–1096, Dec. 2014.

[27] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[28] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[29] J. Skalse, N. H. Howe, D. Krasheninnikov, and D. Krueger. Defining and characterizing reward hacking. *arXiv preprint arXiv:2209.13085*, 2022.

[30] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, second edition, 2018.

[31] S. Vester. Alternating-time temporal logic with finite-memory strategies. *EPTCS 119*, page 194, 2013.

[32] F. Wang, S. Schewe, and C. Huang. An extension of ATL with strategy interaction. *ACM Trans. Program. Lang. Syst.*, 37(3):9:1–9:41, 2015. doi: 10.1145/2734117. URL https://doi.org/10.1145/2734117.

[33] Y. Wang, N. Roohi, M. West, M. Viswanathan, and G. E. Dullerud. Statistically model checking PCTL specifications on Markov decision processes via reinforcement learning. In *59th IEEE Conference on Decision and Control, CDC 2020, Jeju Island, South Korea, December 14-18, 2020*, pages 1392–1397. IEEE, 2020. doi: 10.1109/CDC42340.2020.9303982. URL https://doi.org/10.1109/CDC42340.2020.9303982.

[34] Y. Yuan, Z. L. Yu, Z. Gu, X. Deng, and Y. Li. A novel multi-step reinforcement learning method for solving reward hacking. *Applied Intelligence*, 49:2874–2888, 2019.