Natural Strategic Ability in Stochastic Multi-Agent Systems

Raphaël Berthon[1], **Joost-Pieter Katoen**[1], **Munyque Mittelmann**[2], **Aniello Murano**[2]
[1] RWTH Aachen University, Germany
[2] University of Naples Federico II, Italy
{berthon,katoen}@cs.rwth-aachen.de, {munyque.mittelmann, aniello.murano}@unina.it

Abstract

Strategies synthesized using formal methods can be complex and often require infinite memory, which does not correspond to the expected behavior when trying to model Multi-Agent Systems (MAS). To capture such behaviors, natural strategies are a recently proposed framework striking a balance between the ability of agents to strategize with memory and the model-checking complexity, but until now it has been restricted to fully deterministic settings. For the first time, we consider the probabilistic temporal logics PATL and PATL* under natural strategies (NatPATL and NatPATL*, resp.). As main result we show that, in stochastic MAS, NatPATL model-checking is **NP**-complete when the active coalition is restricted to deterministic strategies. We also give a **2NEXPTIME** complexity result for NatPATL* with the same restriction. In the unrestricted case, we give an **EXPSPACE** complexity for NatPATL and **3EXPSPACE** complexity for NatPATL*.

Introduction

In the last decade, much attention has been devoted to the verification of *Multi-Agent Systems* (MAS). One of the most important early developments was the Alternating-time Temporal Logics ATL and ATL* (Alur, Henzinger, and Kupferman 2002). Since its initial proposal, ATL has been extended in various directions, considering, for instance, strategy contexts (Laroussinie and Markey 2015) or adding imperfect information and epistemic operators (Jamroga and Bulling 2011). Strategy Logic (SL) (Chatterjee, Henzinger, and Piterman 2010; Mogavero et al. 2014) extends ATL to treat strategies as first-order variables. The probabilistic logics PATL, PATL* (Chen and Lu 2007), Stochastic Game Logic (Baier et al. 2012), and PSL (Aminof et al. 2019) enhances ATL, ATL*, ATL with strategy contexts, and SL, resp., to the probabilistic setting. Those logics allow us to express that a coalition can enforce that the probability of satisfying their goal meets a specified constraint.

The importance of the aforementioned logics lies in the *uncertainty* often faced by MAS, due to the occurrence of randomization, such as natural events and the behavior of their components (i.e., the agents). While those aspects cannot be known with certainty, they can be measured based on experiments or past observations. Examples include, among others, the affluence of users interacting with the system, unknown preference of its agents modeled with probabilistic distributions, and errors of its sensorial components. All the aforementioned logics also have downsides, either complexity-wise or memory-wise. PSL is undecidable, and is still **3EXPSPACE** when restricted to memoryless strategies. PATL model checking is in **NP** $\cap$ **co-NP** but requires infinite-memory strategies. Stochastic game logic is **PSPACE** with memoryless deterministic strategies, and **EXPSPACE** with memoryless probabilistic strategies. These last two results are of interest, but the memoryless assumption is quite restrictive.

Natural strategies, first defined in (Jamroga, Malvone, and Murano 2019a), are lists of condition-action pairs with a bounded memory representation. This definition contrasts with combinatorial strategies (i.e., functions from histories to actions), considered typically in the semantics of logics for MAS, including ATL and ATL*. The motivation for natural strategies, as argued in (Jamroga, Malvone, and Murano 2019a), is that combinatorial strategies are not realistic in the context of human behavior, because of the difficulty to execute and design complex plans. In particular, systems that are difficult to use are often ignored by the users, even if they respect design specifications such as security constraints. Artificial agents with limited memory or computational power cannot use combinatorial strategies either. On the other end of the spectrum, memoryless strategies that depend only on the current state cannot provide adequate solutions to many planning problems.

Natural strategies encompass both bounded memory and specifications of agents with “simple” strategies, by allowing agents to use some past observations without requiring infinite memory. They aim at capturing the intuitive approach a human would use when describing strategies. As a result, these strategies are easier to explain using natural language. They also intrinsically feature imperfect information, since they reason about the sequence of propositional variables observed in previous states, instead of the states themselves. Although the systems with whom these agents interact may be stochastic, the study of natural strategies has been until now restricted to fully deterministic settings. *For the first time, we consider* PATL *and* PATL* *under natural strategies and investigate their model checking problem for stochastic MAS.* Remarkably, the logics we consider can

	Det.∼Strategies	Prob.∼Strategies
$\mathsf{NatPATL_r}$	$\mathbf{NP}$-complete	$\mathbf{EXPSPACE}$
$\mathsf{NatPATL^*_r}$	$\mathbf{2NEXPTIME}$	$\mathbf{3EXPSPACE}$
$\mathsf{NatPATL_R}$	$\mathbf{NP}$-complete	$\mathbf{EXPSPACE}$
$\mathsf{NatPATL^*_R}$	$\mathbf{2NEXPTIME}$	$\mathbf{3EXPSPACE}$

Table 1: Summary of model checking complexity problems for NatPATL and NatPATL* with stochastic MAS.

also be seen as an extension of POMDPS to a setting with multiple agents with bounded memory strategies (Chatterjee, Chmelik, and Davies 2016).

Contribution. In this paper, we propose variants of the probabilistic logics PATL and PATL* with natural strategies (denoted NatPATL and NatPATL*, resp.) and study their complexity for model checking. We present complexity results for deterministic and, for the first time, *probabilistic natural strategies*. With respect to the agents' memory, we investigate both the memoryless and bounded recall settings [1]. Table 1 summarizes the results of this paper. The main advantage of the logics proposed is that they enable to express and verify the strategic abilities of stochastic MAS in which agents have limited memory and/or computational power, with a reasonably good model checking complexity. In particular, the model checking of $\mathsf{NatPATL_R}$ is $\mathbf{NP}$-complete for deterministic natural strategies, and in $\mathbf{EXPSPACE}$ for probabilistic natural strategies.

Related Work

Several works consider the verification of stochastic MAS with specifications given in probabilistic logics. In particular, Huang and Luo (2013) study an ATL-like logic for stochastic MAS when agents play deterministic strategies and have probabilistic knowledge. The model checking problem has been studied for Probabilistic Alternating-Time μ-Calculus (Song et al. 2019). Huang, Su, and Zhang (2012) consider the logic Probabilistic ATL* (PATL*) under incomplete information and synchronous perfect recall. PATL was also considered under imperfect information and memoryless strategies (Belardinelli et al. 2023), and with accumulated costs/rewards (Chen et al. 2013).

Also in the context of MAS, probabilistic logics were used for the verification of unbounded parameterized systems (Lomuscio and Pirovano 2020), resource-bounded systems (Nguyen and Rakib 2019), and under assumptions over opponents' strategies (Bulling and Jamroga 2009).

Our work is also related to the research on representation of strategies with limited memory. This includes the representation of finite-memory strategies by input/output automata (Vester 2013), decision trees (Brázdil et al. 2015), ATL with bounded memory (Ågotnes and Walther 2009), as well as the use of bounded memory as an approximation of perfect recall (Belardinelli, Lomuscio, and Malvone 2018). More recently, Deuser and Naumov (2020) represented strategies as Mealy machines and studied how bounded recall affects the agents' abilities to execute plans.

[1]As usual, we denote no recall with r and recall with R.

Natural strategies have first been studied in (Jamroga, Malvone, and Murano 2019a) on multiple deterministic settings: finding winning strategies in concurrent games with LTL specifications, deciding if a set of strategies defines a Nash equilibrium, and model checking ATL. This last use of natural strategies has later been extended to ATL with imperfect information (Jamroga, Malvone, and Murano 2019b) and SL (Belardinelli et al. 2022).

The study of partially observable MDPs (POMDPs) also considers a variety of strategy representations, as discussed in (Vlassis, Littman, and Barber 2012). When allowing infinite-memory strategies, finding an almost-sure winning strategy with a Büchi or reachability objective requires exponential time on POMDPs, while finding strategies for almost-sure parity objectives (Baier, Bertrand, and Größer 2008; Chatterjee, Doyen, and Henzinger 2010) and for maximizing a reachability objective (Madani, Hanks, and Condon 2003) is undecidable. However, when resticting the memory of the strategies to some fixed bound (Pajarinen and Peltonen 2011; Junges et al. 2018), the complexity of threshold reachability becomes $\mathbf{ETR}$-complete (the existential theory of the reals) with probabilistic strategies and $\mathbf{NP}$-complete with deterministic strategies (Junges 2020). The complexity of almost-sure reachability with bounded memory probabilistic strategies is also $\mathbf{NP}$-complete (Chatterjee, Chmelik, and Davies 2016).

Preliminaries

In this paper, we fix finite non-empty sets of agents Ag, actions Ac, and atomic propositions AP. We write $\boldsymbol{c}$ for a tuple of actions $(c_a)_{a\in \mathrm{Ag}}$, one for each agent, and such tuples are called *action profiles*. Given an action profile $\boldsymbol{c}$ and $C \subseteq \mathrm{Ag}$, we let c_C be the components of agents in C, and $\boldsymbol{c}_{-C}$ is $(c_b)_{b\notin C}$. Similarly, we let $\mathrm{Ag}_{-C} = \mathrm{Ag} \setminus C$.

Distributions. Let X be a finite non-empty set. A *(probability) distribution* over X is a function $\mathsf{d} : X \to [0,1]$ such that $\sum_{x\in X} \mathsf{d}(x) = 1$. Let $\mathrm{Dist}(X)$ be the set of distributions over X. We write $x \in \mathsf{d}$ for $\mathsf{d}(x) > 0$. If $\mathsf{d}(x) = 1$ for some element $x \in X$, then d is a *point (a.k.a. Dirac) distribution*. If, for $i \in I$, d_i is a distribution over X_i, then, writing $X = \prod_{i\in I} X_i$, the *product distribution* of the d_i is the distribution $\mathsf{d} : X \to [0,1]$ defined by $\mathsf{d}(x) = \prod_{i\in I} \mathsf{d}_i(x_i)$.

Markov Chains. A *Markov chain* M is a tuple (St, p) where St is a countable non-empty set of states and $p \in \mathrm{Dist}(\mathrm{St}\times\mathrm{St})$ is a distribution. For $s, t \in \mathrm{St}$, the values $p(s,t)$ are called *transition probabilities* of M. A *path* is an infinite sequence of states.

Concurrent Game Structures. A *stochastic concurrent game structure* (or simply *CGS*) $\mathcal{G}$ is a tuple $(\mathrm{St}, \mathrm{L}, \delta, \ell)$ where (i) St is a finite non-empty set of *states*; (ii) $\mathrm{L} : \mathrm{St}\times\mathrm{Ag} \to 2^{\mathrm{Ac}} \setminus \{\emptyset\}$ is a *legality function* defining the available actions for each agent in each state, we write $\mathrm{L}(\boldsymbol{s})$ for the tuple $(\mathrm{L}(s,a))_{a\in\mathrm{Ag}}$; (iii) for each state $s \in \mathrm{St}$ and each move $\boldsymbol{c} \in \mathrm{L}(\boldsymbol{s})$, the *stochastic transition function* δ gives the (conditional) probability $\delta(s,\boldsymbol{c})(s')$ of a transition from state s for all $s' \in \mathrm{St}$ if each player $a \in \mathrm{Ag}$ plays the action $\boldsymbol{c}_a$, and remark that $\delta(s,\boldsymbol{c}) \in \mathrm{Dist}(\mathrm{St})$; (iv) $\ell : \mathrm{St} \to 2^{\mathrm{AP}}$ is a *labelling function*.

For each state $s \in \mathrm{St}$ and joint action $\boldsymbol{c} \in \prod_{a\in\mathrm{Ag}} \mathrm{L}(s,a)$, we assume that there is a state $s' \in \mathrm{St}$ such that $\delta(s,\boldsymbol{c})(s')$ is non-zero, that is, every state has a successive state from a legal move, formally $\boldsymbol{c} \in \mathrm{L}(s,a)$.

Example 1 (Secure voting [2]). Assume a voting system with two types of agents: voters and coercers, represented by the disjoint sets $V \subset \mathrm{Ag}$ and $C \subset \mathrm{Ag}$, resp. We consider a finite set of receipts, and signatures. The actions of the voters are $scanBallot$, $enterVote$, $cnlVote$, $conf$, $checkSig_s$, $checkrec_r$, $shred_r$, and $noop$, which represent that the agent is scanning the ballot, entering their vote, canceling it, *conf*irming it, checking its signature s, checking the receipt r, shredding the receipt r, and doing nothing, resp. On its turn, the coercer can perform the actions $coerce_v$, $request_v$, $punish_v$, and $noop$, representing that she is coercing the voter v, requesting v to vote, punishing v, and doing nothing, resp.

The CGS has propositions denoting the state of the voting system. Specifically, they describe whether the voter v was coerced ($coerced_v$), punished ($punished_v$), requested to vote ($requested_v$), has a ballot available ($hasBallot_v$), scanned the ballot ($scanned_v$), entered the vote which has the signature s ($entVote_{v,s}$), and has already voted (vot_v). For a signature s, the proposition $sigOk_s$ denotes whether the signature s was checked and corresponds to the one in the system, while the proposition $sigFail_s$ denotes that it was checked but did not correspond. For a receipt r, the propositions $rec_{v,r}$ and $shreded_r$ denotes whether r associated with voter v and whether r was destroyed (and it's no longer visible), resp.

Actions performed by the agents may fail and may not change the state of the system as intended by them. For instance, the coercer may not succeed (attempting) to coerce a voter with the action $coerce_v$ (and thus, $coerced_v$ may not be true in the next state). Similarly, a voter's request to shred her receipt may fail, and the information on the receipt be still visible. The probability of an action failing is described by the CGS stochastic transition function.

Plays. A *play* or path in a CGS $\mathcal{G}$ is an infinite sequence $\pi = s_0 s_1 \cdots$ of states such that there exists a sequence $\boldsymbol{c}_0\boldsymbol{c}_1\cdots$ of joint-actions such that $\boldsymbol{c}_i \in \mathrm{L}(s_i)$ and $s_{i+1} \in \delta(s_i,\boldsymbol{c}_i)$ (i.e., $\delta(s_i,\boldsymbol{c}_i)(s_{i+1}) > 0$) for every $i \geq 0$. We write π_i for s_i, $\pi_{\geq i}$ for the suffix of π starting at position i. Finite paths are called *histories*, and the set of all histories is denoted Hist. We write last(h) for the last state of a history h and $len(h)$ for the size of h.

Behavioral Natural Strategies

In this section, we define behavioral[3] natural strategies over CGS, based on the definition in (Jamroga, Malvone, and Murano 2019a), and use them to provide the semantics of NatATL*. Natural strategies are conditional plans, represented through an ordered list of condition-action rules. The intuition is that the first rule whose condition holds in the history of the game is selected, and the corresponding action is executed. The conditions are regular expressions over *Boolean formulas* over AP, denoted $Bool(\mathrm{AP})$ and given by the following BNF grammar:

$$\varphi ::= p \mid \varphi \vee \varphi \mid \neg\varphi \qquad \text{where } p \in \mathrm{AP}.$$

Given a state $s \in \mathrm{St}$ and a formula $\varphi \in Bool(\mathrm{AP})$, we inductively define the satisfaction value of φ in s, denoted $s \models \varphi$, as follows:

$$\begin{array}{ll} s \models p & \text{iff } p \in \ell(s) \\ s \models \varphi_1 \vee \varphi_2 & \text{iff } s \models \varphi_1 \text{ or } s \models \varphi_2 \\ s \models \neg\varphi & \text{iff not } s \models \varphi \end{array}$$

Let $Reg(Bool(\mathrm{AP}))$ be the set of regular expressions over the conditions $Bool(\mathrm{AP})$, defined with the constructors $\cdot, \cup, *$ representing concatenation, nondeterministic choice, and Kleene iteration, respectively. Given a regular expression r and the language $\mathcal{L}(r)$ of finite words generated by r, a history h is *consistent* with r iff there exists a word $b \in \mathcal{L}(r)$ such that $|h| = |b|$ and $h[i] \models b[i]$, for all $0 \leq i \leq |h|$. Intuitively, a history h is consistent with a regular expression r if the i-th epistemic condition in r holds in the i-th state of h (for any position i in h).

A *behavioral natural strategy σ with recall* for an agent $a \in \mathrm{Ag}$ is a sequence of pairs $(r, \mathrm{Dist}(\mathrm{Ac}))$, where $r \in Reg(Bool(\mathrm{AP}))$ is a regular expression representing recall, and $\mathsf{d}(\mathrm{Ac})$ is a distribution over the actions with $\mathsf{d}(c) \neq 0$ if c is available for a in last(h) (i.e., for $c \in \mathrm{L}(a, \mathrm{last}(h))$), for all histories h consistent with r. The last pair in the sequence is required to be $(\top^*, \mathsf{d}(\mathrm{Ac}))$, with $\mathsf{d}(c) = 1$ for some $c \in \mathrm{L}(s,a)$ and every $s \in \mathrm{St}$. A *behavioral memoryless natural strategy* is a behavioral natural strategy without recall: each condition is a Boolean formula (i.e., all regular expressions have length 1). A strategy σ is deterministic if for all pairs (r, d), we have $|\{c \in \mathrm{Ac} \mid \mathsf{d}(c) \neq 0\}| = 1$. For readability of the examples, given a pair (r, d), we write (r, c) if $\mathsf{d}(c) = 1$ for some action $c \in \mathrm{Ac}$.

Example 2 (Secure voting, continued). Recall the voting system introduced in Example 1. The following is a deterministic memoryless natural strategy for the voter v:

1. $(hasBallot_v \wedge \neg scanned_v, scanBallot)$
2. $(\neg vot_v \wedge scanned_v, enterVote)$
3. $(\neg vot_v \wedge entVote_{v,s} \wedge \neg(sigOk_s \vee sigFail_s), checkSig_s)$, for each signature s
4. $(\neg vot_v \wedge entVote_{v,s} \wedge sigFail_s, cnlVote)$, for each s
5. $(\neg vot_v \wedge entVote_{v,s} \wedge sigOk_s, conf)$, for each s
6. $(vot_v \wedge rec_{v,r} \wedge \neg shreded_r, shred_r)$, for each receipt r
7. $(\top, noop)$

This strategy specifies that the agent first scans the ballot in case there is one, and it was not scanned (Pair 1). Otherwise, if the agent has not voted yet and has scanned, she enters her vote (Pair 2). If the agent did not vote, entered the vote and did not check the signature, she checks it (3).

[2]Our running example on secure voting is adapted from the case study from (Jamroga, Kurpiewski, and Malvone 2020, 2022).

[3]Behavioral strategies define the probability of taking an *action* in a state. This is different from mixed strategies, which define the probability of taking a *strategy* in a game. The relation of behavioral and mixed strategies is discussed in (Kaneko and Kline 1995).

When the signature is checked, the agent chooses to cancel or *conf*irm the vote, depending on whether the verification has failed or succeeded (Pairs 4 and 5). If the agent has voted and there is an unshredded visible receipt, the agent requests it to be shredded (Pair 6). Finally, if none of the previous conditions apply, the agent does not do any action (Pair 7).

A behavioral natural strategy with recall for a coercer is:

1. $\big(\top^* \cdot \bigwedge_{v \in V} \neg coerced_v, \mathsf{d}_V\big)$, where d_V is a probability distribution over the actions for coercing the voters, with $\sum_{v \in V} \mathsf{d}_V(coerce_v) = 1$
2. $(\top^* \cdot coerced_v \wedge \neg requested_v, request_v)$, for $v \in V$
3. $(\top^* \cdot \neg requested_v{}^* \cdot (requested_v \wedge \neg vot_v)^* \wedge \neg punished_v, punish_v)$, for each $v \in V$
4. $(\top^* \cdot \neg coerced_v \wedge \neg coerced_{v'}, \mathsf{d}_{v,v'})$, where $\mathsf{d}_{v,v'}$ is a probability distribution over the actions for coercing the voters v and v' , with $\mathsf{d}_{v,v'}(coerce_v) + \mathsf{d}_{v,v'}(coerce_{v'}) = 1$, for each pair (v, v') of distinct voters in V
5. $(\top^*, noop)$

This behavioral strategy considers first the situation in which no voter was already coerced, and the agent chooses the action to coerce one of them randomly (Pair 1). Next condition-action pair in the strategy says that if a voter was coerced, but her vote was not requested, the agent requests her vote (Pair 2). If the voter was requested (at least once in the history), but (continually) did not vote and was not punished, the agent punishes her (Pair 3). Next pair says that if there are two distinct non-coerced voters, the agent randomly chooses one to coerce (Pair 4). If none of those conditions apply, no operation is performed (Pair 5).

Throughout this paper, let $\rho \in \{r, R\}$ denote whether we consider memoryless or recall strategies respectively. Let Str^ρ_a be the set of behavioral natural strategies for agent a and $Str^\rho = \cup_{a \in \mathrm{Ag}} Str^\rho_a$.

Let $size(\sigma_a)$ denote the number of guarded actions in σ_a, $cond_i(\sigma_a)$ be the i-th guarded condition on σ_a, $cond_i(\sigma_a)[j]$ be the j-th Boolean formula of the guarded condition σ_a, and $act_i(\sigma_a)$ be the corresponding probability distribution on actions. Finally, $match(h, \sigma_a)$ is the smallest index $i \le size(\sigma_a)$ such that for all $0 \le j \le |last(h)|$, $h[j] \models cond_i(\sigma_a)[j]$[4] and $act_i(\sigma_a) \in \mathrm{L}(a, last(h))$. In other words, $match(h, \sigma_a)$ matches the state $last(h)$ with the first condition in σ_a that holds in h, and action available in $last(h)$.

Given a natural strategy σ_a for agent a and a history h, we denote by $\sigma_a(h)$ the probability distribution of actions assigned by σ_a in the last state of h, i.e., $\sigma_a(h) = act_{match(h,\sigma_a)(\sigma_a)}$.

Complexity of Natural Strategies. The complexity $c(\sigma)$ of strategy σ is the total size of its representation and is denoted as follows: $c(\sigma) := \sum_{(r,\mathsf{d}) \in \sigma} |r|$, where $|r|$ is the number of symbols in r, except parentheses.

[4]Note that, as in (Jamroga, Malvone, and Murano 2019a), we consider the case in which the condition has the same length of the history. There is also the case in which the condition is shorter than the history. This is due to the usage of the Kleene iteration operator. In the latter case, we need to check a finite number of times the same Boolean formula in different states of the history.

Relation to Game Theory. From a game-theoretic point of view, natural strategies can be encoded as finite-memory strategies using finite state machines (e.g., finite-state transducers) that only read the propositional variables holding in a state (akin to imperfect information). Natural strategies and finite-state machines are fundamentally different in their encoding, in particular, the finite state machine may be exponential in the size of the natural strategy it is associated with, as proved in Theorem 12 of (Jamroga, Malvone, and Murano 2019a).

PATL and PATL* with Natural Strategies

Next, we introduce the Probabilistic Alternating-Time Temporal Logics PATL* and PATL with Natural Strategies (denoted, NatPATL* and NatPATL, resp).

Definition 1. The syntax of NatPATL* is defined by the grammar:

$$\varphi ::= p \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\varphi \mid \varphi\mathbf{U}\varphi \mid \langle\!\langle C \rangle\!\rangle^{\bowtie d}_k \varphi$$

where $p \in \mathrm{AP}$, $k \in \mathbb{N}$, $C \subseteq \mathrm{Ag}$, d is a rational constant in $[0, 1]$, and $\bowtie \in \{\le, <, >, \ge\}$.

The intuitive reading of the operators is as follows: "next" $\mathbf{X}$ and "until" $\mathbf{U}$ are the standard temporal operators. $\langle\!\langle C \rangle\!\rangle^{\bowtie d}_k \varphi$ asserts that there exists a strategy with complexity at most k for the coalition C to collaboratively enforce φ with a probability in relation $\bowtie$ with constant d. We make use of the usual syntactic sugar $\mathbf{F}\varphi := \top\mathbf{U}\varphi$ and $\mathbf{G}\varphi := \neg\mathbf{F}\neg\varphi$ for temporal operators.

A NatPATL* formula of the form $\langle\!\langle C \rangle\!\rangle^{\bowtie d}\varphi$ is also called state formula. An important syntactic restriction of NatPATL*, namely NatPATL, is defined as follows.

Definition 2 (NatPATL syntax)**.** The syntax of NatPATL is defined by the grammar

$$\varphi ::= p \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle\!\langle C \rangle\!\rangle^{\bowtie d}_k(\mathbf{X}\varphi) \mid \langle\!\langle C \rangle\!\rangle^{\bowtie d}_k(\varphi\mathbf{U}\varphi)$$

where p, k, C, d, and $\bowtie$ are as above.

Before presenting the semantics, we show how to define the probability space on outcomes.

Probability Space on Outcomes. An *outcome* of a strategy profile $\boldsymbol{\sigma} = (\sigma_a)_{a \in \mathrm{Ag}}$ and a state s is a play π that starts in state s and is extended by $\boldsymbol{\sigma}$, formally $\pi_0 = s$, and for every $k \ge len(h)$ there exists $\boldsymbol{c}_k \in (\sigma_a(\pi_{\le k}))_{a \in \mathrm{Ag}}$ such that $\pi_{k+1} \in \delta(\pi_k, \boldsymbol{c}_k)$. The set of outcomes of a strategy profile $\boldsymbol{\sigma}$ and state h is denoted $Out(\boldsymbol{\sigma}, s)$. A given CGS $\mathcal{G}$, strategy profile $\boldsymbol{\sigma}$, and state s induce an infinite-state Markov chain $M_{\boldsymbol{\sigma},s}$ whose states are the histories in $Out(\boldsymbol{\sigma}, s)$. and whose transition probabilities are defined as $p(h, hs') = \sum_{\boldsymbol{c} \in \mathrm{Ac}^{\mathrm{Ag}}} \boldsymbol{\sigma}(h)(\boldsymbol{c}) \times \delta(\mathrm{last}(h), \boldsymbol{c})(s')$. The Markov chain $M_{\boldsymbol{\sigma},s}$ induces a canonical probability space on its set of infinite paths (Kemeny, Snell, and Knapp 1976), which can be identified with the set of plays in $Out(\boldsymbol{\sigma}, s)$ and the corresponding measure is denoted $out(\boldsymbol{\sigma}, s)$. [5]

Given a coalition strategy $\boldsymbol{\sigma_C} \in \prod_{a \in C} Str^\rho_a$, the set of possible outcomes of $\boldsymbol{\sigma_C}$ from a state $s \in \mathrm{St}$ to be

[5]This is a classic construction, see for instance (Clarke et al. 2018; Berthon et al. 2020).

the set $out_C(\boldsymbol{\sigma_C}, s) = \{out((\boldsymbol{\sigma_C}, \boldsymbol{\sigma_{-C}}), s) : \boldsymbol{\sigma_{-C}} \in \prod_{a \in \mathrm{Ag}_{-C}} \mathit{Str}^{\rho}_a\}$ of probability measures that the players in C enforce when they follow the strategy $\boldsymbol{\sigma_C}$, namely, for each $a \in \mathrm{Ag}$, player a follows strategy σ_a in $\boldsymbol{\sigma_C}$. We use $\mu_s^{\boldsymbol{\sigma_C}}$ to range over the measures in $out_C(\boldsymbol{\sigma_C}, s)$ as follows:

Definition 3 (NatPATL and NatPATL* semantics)**.** Given a setting $\rho \in \{r, R\}$, NatPATL and NatPATL* formulas are interpreted in a stochastic CGS $\mathcal{G}$ and a path π,

$$\begin{array}{ll}
\mathcal{G}, \pi \models_\rho p & \text{iff } p \in \ell(\pi_0) \\
\mathcal{G}, \pi \models_\rho \neg\varphi & \text{iff } \mathcal{G}, \pi \not\models_\rho \varphi \\
\mathcal{G}, \pi \models_\rho \varphi_1 \vee \varphi_2 & \text{iff } \mathcal{G}, \pi \models_\rho \varphi_1 \text{ or } \mathcal{G}, \pi \models_\rho \varphi_2 \\
\mathcal{G}, \pi \models_\rho \langle\langle C \rangle\rangle_k^{\bowtie d}\varphi & \text{iff } \exists \boldsymbol{\sigma_C} \in \prod_{a \in C} \{\alpha \in \mathit{Str}^{\rho}_a : c(\alpha) \le k\} \\
\multicolumn{2}{l}{\text{s.t. } \forall \mu_{\pi_0}^{\boldsymbol{\sigma_C}} \in out_C(\boldsymbol{\sigma_C}, \pi_0), \mu_{\pi_0}^{\boldsymbol{\sigma_C}}(\{\pi' : \mathcal{G}, \pi' \models_\rho \varphi\}) \bowtie d} \\
\mathcal{G}, \pi \models_\rho \mathbf{X}\varphi & \text{iff } \mathcal{G}, \pi_{\ge 1} \models_\rho \varphi \\
\mathcal{G}, \pi \models_\rho \psi_1 \mathbf{U} \psi_2 & \text{iff } \exists k \ge 0 \text{ s.t. } \mathcal{G}, \pi_{\ge k} \models_\rho \psi_2 \text{ and} \\
& \quad \forall j \in [0, k).\ \mathcal{G}, \pi_{\ge j} \models_\rho \psi_1
\end{array}$$

Motivating Examples

In this section, we present problems that motivate reasoning in stochastic MAS, and we illustrate how NatPATL*-formulas can be used to express properties on those systems.

Let us start with an example of door access control with a random robot. This example illustrates a setting in which it suffices to have deterministic strategies in stochastic CGSs.

Example 3 (Access control). We consider the example illustrated in Figure 1. We are given a set Ag of agents, a set of square tiles, where a non-controlled robot moves randomly either one tile right, left, up, or down at every time step. Between every tile, there is either a wall, a door controlled by some agent with actions *open* and *close*, or nothing. The robot can cross an empty space, cannot cross a wall, and can only cross a door if the agent controlling has taken action *open*. Given a set of targets represented by atomic propositions $T = \{t_i \in \mathrm{AP},\ i \in \{1, n\}\}$ labelling some tiles, and related coalitions $\{C_i \subseteq \mathrm{Ag}, i \in \{1, n\}\}$, we use NatPATL* to state that some coalition $C \subseteq \mathrm{Ag}$ has a strategy with memory $k \in \mathbb{N}$ reaching all targets infinitely often with probability 0.7, formally:

$$\langle\langle C \rangle\rangle_k^{\ge 0.7}\mathbf{G} \bigwedge_{t_j \in T, t_j \ne t_i} \mathbf{F}\, t_j \tag{1}$$

In the example of Figure 1, where $n = 2$, the coalition controls two doors adjacent to the initial state. Even though the structure is probabilistic, memoryless strategies are sufficient. Opening the left door gives the robot a chance to move left, which brings it closer to target t_0, but in this center-leftmost square, the agents not in the coalition may open the door leading to the bottom-left square, where they can then trap the robot: the robot only has probability $\frac{1}{2}$ to successfully reach t_0, and otherwise may be trapped forever. The other option available to the coalition in the initial state is to close the left door, and open all other doors. The robot will take longer, but has probability 1 to eventually reach target t_1. Thus, we can reach t_1 with probability 1, but t_0 with only probability $\frac{1}{2}$, and property 1 does not hold.

Going back to Example 1, we now illustrate how NatPATL* and NatPATL can be used for the formal security analysis of voting systems.

Example 4 (Secure voting, continued). A requirement usually considered for electronic voting systems is *voter-verifiability*, which captures the ability of the voter to verify her vote (Jamroga, Kurpiewski, and Malvone 2022). In our example, this is represented by the propositions $sigOk_s$ and $sigFail_s$. The NatPATL formula

$$\langle\langle v \rangle\rangle_k^{\ge 0.9}\mathbf{F}(sigOk_s \vee sigFail_s)$$

says that the voter v has a strategy of size at most k so that, at some point and with probability at least 0.9, she obtains either the positive or the negative outcome of verifying the signature s.

Another requirement is *receipt-freeness*, which expresses that the voters can not gain a receipt to prove that they voted in a certain way. In our example, the propositions $receipt_{v,r}$ and $shreded_r$ represent that a receipt r is associated with the voter v and that the information on it was destroyed. The NatPATL formula:

$$\neg\langle\langle v \rangle\rangle_k^{\ge 0.5}\mathbf{F} \bigvee_{\text{receipt } r} (receipt_{v,r} \wedge \neg shreded_r)$$

says that there is no strategy of complexity at most k to ensure with probability at least 0.5 that, eventually, there will be an unshredded receipt for her.

Model Checking Complexity

In this section, we look at the complexity of model checking for different versions of NatPATL.

Definition 4. Given a setting $\rho \in \{r, R\}$, a CGS $\mathcal{G}$, state $s \in \mathrm{St}$, and formula φ in NatPATLρ (NatPATL*ρ, resp.), the model checking problem for NatPATLρ (NatPATL*ρ, resp.) consists in deciding, whether $\mathcal{G}, s \models_\rho \varphi$.

Theorem 1. Model checking NatPATL$_\mathrm{r}$ (respectively NatPATL$_\mathrm{R}$) with deterministic natural strategies for the coalition is in $\mathbf{NP}$.

The main idea is that since we know the memory bound of every strategy of the coalition, we can guess these strategies, and polynomially verify their correctness since we only have to check MDPs with reachability or invariance objectives.

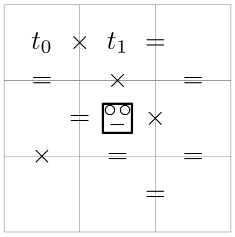

Figure 1: A robot in a maze, where $=$ and $\times$ denote a door respectively controlled by the coalition or the agents not in the coalition. Full lines represent walls. t_0, t_1 are two targets.

Theorem 2. Model checking $\mathsf{NatPATL}_{\mathrm{r}}$ (respectively $\mathsf{NatPATL}_{\mathrm{R}}$) with deterministic natural strategies for the coalition is $\mathbf{NP}$-hard.

Proof. We start by showing that $\mathsf{NatPATL}_{\mathrm{r}}$ with deterministic natural strategies for the coalition extends POMDPs with memoryless deterministic strategies and almost-sure reachability objective. Indeed, a POMDP represented as a CGS $\mathcal{G} = (\mathrm{St}, \mathrm{L}, \delta, \ell)$ (two states are indistinguishable if they are labelled by the same propositional variables), a single agent A, and a set of target states distinguished by some propositional variable t that holds only in these states, there exists a strategy almost surely reaching t from an initial state $s \in \mathrm{St}$ if and only if the $\mathsf{NatPATL}_{\mathrm{r}}$ formula $\langle\langle A\rangle\rangle^{=1}_{|\mathrm{St}|}(\mathbf{F}t)$ holds on $\mathcal{G}$. Indeed, memoryless strategies cannot have a complexity higher than $|\mathrm{St}|$, the number of states in the MDP, and so available strategies coincide. In Proposition 2 of (Chatterjee, Chmelik, and Davies 2016), it is shown that finding strategies for POMDPs with memoryless *randomized* strategies and almost-sure reachability objective is $\mathbf{NP}$-hard. It uses a reduction from Lemma 1 of (Chatterjee, Kößler, and Schmid 2013), that only uses deterministic strategies. As such, finding strategies for POMDPs with memoryless *deterministic* strategies and almost-sure reachability objective is $\mathbf{NP}$-hard, so it is the model checking $\mathsf{NatPATL}_{\mathrm{r}}$ with behavioral natural deterministic strategies for the coalition. □

Theorem 3. Model checking $\mathsf{NatPATL}^*_{\mathrm{r}}$ (respectively $\mathsf{NatPATL}^*_{\mathrm{R}}$) with deterministic natural strategies for the coalition is in $\mathbf{2NEXPTIME}$.

Sketch of proof. The idea is similar to Theorem 1, we guess strategies of bounded size for every coalition, but we now have to check LTL formulas on MDPs, which is $\mathbf{2EXPTIME}$. We also show $\mathbf{2EXPTIME}$-hardness with a reduction from LTL on MDPs. This complexity is not surprising, since classical ATL* has already doubly exponential complexity (Alur, Henzinger, and Kupferman 2002). □

Theorem 4. Model checking $\mathsf{NatPATL}^*_{\mathrm{r}}$ (respectively $\mathsf{NatPATL}^*_{\mathrm{R}}$) with deterministic natural strategies for the coalition is $\mathbf{2EXPTIME}$-hard.

Proof. We use a reduction from LTL model checking on MDPs, which is $\mathbf{2EXPTIME}$-complete. Given an LTL formula φ, a threshold $d \in [0,1]$ and a CGS $\mathcal{G}$ with only one agent Ag, we say φ holds with at least probability d on $\mathcal{G}$ if and only if the $\mathsf{NatPATL}^*_{\mathrm{r}}$ formula $\langle\langle\varnothing\rangle\rangle^{\geq 1-d}_{k}(\neg\varphi)$ holds on MDP $\mathcal{G}$: this formula states that for any strategy of the agent (without any complexity bound, since the coalition is empty), formula $\neg\varphi$ holds with probability at least $1-d$: this only happens if there is no strategy ensuring φ with at least probability d. □

When considering probabilistic strategies, we follow the same technique as (Aminof et al. 2019) to reduce the problem to model checking real arithmetic. Since LTL is subsumed by $\mathsf{NatPATL}^*_{\mathrm{R}}$, we also have a doubly-exponential blowup. On the other hand, with $\mathsf{NatPATL}_{\mathrm{R}}$, we roughly follow an idea introduced for stochastic game logic (Baier et al. 2012) to avoid this blowup. As in the proof of Theorem 1, it is sufficient to consider reachability and invariance problems, both of which are polynomial. The same holds for $\mathsf{NatPATL}^*_{\mathrm{r}}$. Next, we consider the model checking of behavioral strategies. We remark that the $\mathbf{2EXPTIME}$-hardness from Theorem 4 also applies to $\mathsf{NatPATL}_{\mathrm{r}}$ and $\mathsf{NatPATL}^*_{\mathrm{r}}$ with behavioral natural strategies. We now give model checking algorithms and their complexity.

Theorem 5. Model checking $\mathsf{NatPATL}^*_{\mathrm{r}}$ (respectively $\mathsf{NatPATL}^*_{\mathrm{R}}$) with behavioral natural strategies for the coalition is in $\mathbf{3EXPSPACE}$.

Sketch of proof. Probabilistic Strategy Logic (PSL) with an additional behavioral natural strategies operator $\exists^{nat}_{k}$ captures $\mathsf{NatPATL}^*_{\mathrm{R}}$, and we show its model checking is in $\mathbf{3EXPSPACE}$. When translating a $\mathsf{NatPATL}^*_{\mathrm{R}}$ formula into a PSL formula in a bottom-up manner, assuming formula φ can already be translated into some $PSL(\varphi)$ without any complexity blowup, the $\langle\langle C\rangle\rangle^{\bowtie d}_{k}\varphi$ subformulas, can be translated as $\exists^{nat}_{k}\sigma\forall\mu\mathbb{P}_{C\to\sigma,\ \mathrm{Ag}\setminus C\to\mu}(PSL(\varphi)) \bowtie d$: a coalition satisfies φ iff there exists a natural strategy for the coalition such that for all strategies of the other agents, the PSL translation of φ holds. To model check the operator $\exists^{nat}_{k}$, we modify the proof of Theorem 1 of (Aminof et al. 2019) showing that model checking PSL with memoryless strategies is in $\mathbf{3EXPSPACE}$. This proof translates PSL into real arithmetic, and a variable $r_{x,s,a}$ represents the probability for strategy x to take action a in state s. We can extend this notation to behavioral natural strategies: for a strategy with complexity k, we replace variables $r_{x,s,a}$ by $r_{x,s,a,q}$ where q is the current state of the automata representing the regular expressions of a behavioral natural strategy:

$$\bigvee_{\text{strategies } \sigma,\ compl(\sigma)\leq k}\ \bigwedge_{(r,a)\in\sigma} \mathcal{A}_{(r,a)}$$

is an automaton with current state $q[r]$. We state that two probabilities are equal if they are accepted by the same regular expressions:

$$\bigwedge_{(r,a)\in\sigma} acc(q[r], \mathcal{A}_{(r,a)}) \wedge acc(q'[r], \mathcal{A}_{(r,a)})$$
$$\Rightarrow r_{x,s,a,q} = r_{x,s,a,q'}$$

Both are exponential in the largest k in the formula, since there are exponentially many possible automata of size less or equal to k, and we need to describe at most k of them in every conjunction. Nothing else is changed in the proof of (Aminof et al. 2019), and thus we have a $\mathbf{3EXPSPACE}$ complexity in the size of the $\mathsf{NatPATL}^*_{\mathrm{R}}$ formula, exponential in the size of the system and $\mathbf{2EXPSPACE}$ in the largest complexity k used in the formula. □

Theorem 6. Model checking $\mathsf{NatPATL}_{\mathrm{r}}$ (respectively $\mathsf{NatPATL}_{\mathrm{R}}$) with behavioral natural strategies for the coalition is in $\mathbf{EXPSPACE}$.

Sketch of proof. The proof is slightly more complicated than the previous one. We again adapt the proof of Theorem 1 of (Aminof et al. 2019). In the proof of Theorem 5, we translate our fragment of PSL with natural strategies to real arithmetic. The only exponential blowup comes when translating the coalition operator into $\exists_k^{nat}\sigma\forall\mu\mathbb{P}_{C\to\sigma,\ \mathrm{Ag}\setminus C\to\mu}(PSL(\varphi)) \bowtie d$ where φ is assumed to be an LTL formula whose atoms are either propositional variables, or variables representing other formulas starting with $\mathbb{P}$. This translation constructs a deterministic Rabin automaton whose size is exponential in the size of the CGS, double exponential in the size of ψ, and uses a number of quantifiers double exponential in the size of ψ. Since we consider NatPATL$_\mathrm{R}$, this LTL formula may only be either $\mathbf{X}\varphi$ or $\varphi\mathbf{U}\varphi'$, where φ and φ' have been inductively represented as Boolean formulas. Proposition 5.1 and Theorem 5.2 of (Alur, Henzinger, and Kupferman 2002) show that such formulas can be polynomially translated to either reachability or invariance games, which can be done using an automaton and a number of variables both polynomial in the size of the CGS and ψ. Since model checking real arithmetic is exponential in the number of quantifiers of the formula (Ben-Or, Kozen, and Reif 1986; Fitchas, Galligo, and Morgenstern 1987), we obtain that model checking NatPATL$_\mathrm{R}$ with behavioral natural strategies for the coalition is in $\mathbf{EXPSPACE}$. □

Expressivity

We now compare the expressive power of NatPATL* to that of PATL*. We first recall the notions of distinguishing and expressive powers.

Definition 5 (Distinguishing power and expressive power (Wang and Dechesne 2009))**.** Consider two logical systems $\mathcal{L}_1$ and $\mathcal{L}_2$, with their semantics (denoted $\models_{\mathcal{L}_1}$ and $\models_{\mathcal{L}_2}$, resp.) defined over the same class of models $\mathcal{M}$. We say that $\mathcal{L}_1$ is *at least as distinguishing* as $\mathcal{L}_2$ (written: $\mathcal{L}_2 \preceq_d \mathcal{L}_1$) iff for every pair of models $M, M' \in \mathcal{M}$, if there exists a formula $\varphi_2 \in L_2$ such that $M \models_{\mathcal{L}_2} \varphi_2$ and $M' \not\models_{\mathcal{L}_2} \varphi_2$, then there is also $\varphi_1 \in L_1$ with $M \models_{\mathcal{L}_1} \varphi_1$ and $M' \not\models_{\mathcal{L}_1} \varphi_1$.

Moreover, $\mathcal{L}_1$ is *at least as expressive* as $\mathcal{L}_2$ (written: $\mathcal{L}_2 \preceq_e \mathcal{L}_1$) iff for every $\varphi_2 \in L_2$ there exists $\varphi_1 \in L_1$ such that, for every $M \in \mathcal{M}$, we have $M \models_{\mathcal{L}_2} \varphi_2$ iff $M \models_{\mathcal{L}_1} \varphi_1$.

NatPATL* and PATL* are based on different notions of strategic ability. As for the deterministic setting with ATL, each behavioral natural strategy can be translated to a behavioral combinatorial one (i.e., mappings from sequences of states to actions), but not vice versa. Consequently, PATL* can express that a given coalition has a combinatorial strategy to achieve their goal, which is not expressible in NatPATL*. On the other hand, NatATL* allows expressing that a winning natural strategy with bounded complexity does not exist, which cannot be captured in PATL*. Now we show that NatPATL* allows expressing properties that cannot be captured in PATL*, and vice versa.

Theorem 7. For both memoryless and recall semantics:

- NatPATL (resp. NatPATL*) and PATL (resp, PATL*) have incomparable *distinguishing* power over CGS.
- NatPATL (resp. NatPATL*) and PATL (resp, PATL*) have incomparable *expressive* power over CGS.

Sketch of proof. The proof is obtained by an adjustment of the proofs of Prop. 8 and 9 in (Belardinelli et al. 2022), comparing natural strategies with combinatorial strategies. □

Conclusion

In this work, we have defined multiple variations of PATL with natural strategies, and studied their model-checking complexity. We have illustrated with multiple examples the relevance of the probabilistic setting, which can represent uncertainty in a very precise way, and the interest in natural strategies, that are both efficient and much closer to what a real-world agent is expected to manipulate.

In terms of model checking, the $\mathbf{NP}$-completeness of NatPATL with deterministic strategies is promising, and shows we can capture POMDPs with bounded memory without any significant loss. While the $\mathbf{2NEXPTIME}$ complexity for NatPATL* with deterministic strategies is high, we have shown a close lower bound, namely $\mathbf{2EXPTIME}$-hardness. With probabilistic strategies, the $\mathbf{EXPSPACE}$ membership of NatPATL is quite similar to the result of (Baier et al. 2012), and the $\mathbf{3EXPSPACE}$ membership of NatPATL* is also similar to (Aminof et al. 2019). Since this exponential space blowup comes from the use of real arithmetic to encode probabilities, any improvement would likely come from the introduction of a totally new technique. Similarly, the doubly-exponential blowup between PATL and PATL* comes from the $\mathbf{2EXPTIME}$-completeness of LTL model checking on MDPs. We also keep the $\mathbf{2EXPTIME}$-hardness from the deterministic case. To our knowledge, similar works (Baier et al. 2012; Aminof et al. 2019), do also not give different lower bounds between deterministic and probabilistic strategies. A possible approach would be to use a construction from POMDPs, more precisely either (Junges et al. 2018), showing that synthesis on POMDPs with reachability objectives and bounded memory is $\mathbf{NP}$-complete for deterministic strategies and $\mathbf{ETR}$-complete for probabilistic finite-memory strategies or (Oliehoek 2012), showing that finding a policy maximizing a reward on a decentralized POMDPs with full memory is $\mathbf{NEXPTIME}$-complete). Our results on expressivity mean that there are properties of stochastic MAS with natural strategies that cannot be equivalently translated to properties based on combinatorial strategies, and vice versa.

The proof of Theorem 5 shows that we could extend natural strategies to PSL, but it would be difficult to get a better result than our $\mathbf{3EXPSPACE}$ complexity. Considering qualitative PATL* or PSL (i.e. only thresholds > 0 and $= 1$) may yield a better complexity. For the quantitative setting, i.e., thresholds such as $> \frac{1}{2}$, techniques from the field of probabilistic model checking can be applied, e.g., graph analysis, bisimilation minimization, symbolic techniques, and partial-order reduction (Katoen 2016). Another direction would be to consider *epistemic operators*. Indeed, many applications involving agents with a reasonable way to strategize also have to take into account the knowledge and beliefs of these agents. As such, we would have to find a good epistemic framework such that natural strategies keep the desired balance between expressivity and complexity.

Acknowledgements

This research has been supported by the EU H2020 Marie Sklodowska-Curie project with grant agreement No 101105549, PRIN project RIPER (No. 20203FFYLK), the PNRR MUR project PE0000013-FAIR, the InDAM 2023 project "Strategic Reasoning in Mechanism Design", and the DFG Project POMPOM (KA 1462/6-1).

References

Ågotnes, T.; and Walther, D. 2009. A Logic of Strategic Ability Under Bounded Memory. *Journal of Logic, Language and Information*, 18(1): 55–77.

Alur, R.; Henzinger, T.; and Kupferman, O. 2002. Alternating-time temporal logic. *J. ACM*, 49(5): 672–713.

Aminof, B.; Kwiatkowska, M.; Maubert, B.; Murano, A.; and Rubin, S. 2019. Probabilistic Strategy Logic. In *Proc. of IJCAI 2019*, 32–38. ijcai.org.

Baier, C.; Bertrand, N.; and Größer, M. 2008. On Decision Problems for Probabilistic Büchi Automata. In *FOSSACS*.

Baier, C.; Brázdil, T.; Größer, M.; and Kucera, A. 2012. Stochastic game logic. *Acta Informatica*, 49(4): 203–224.

Belardinelli, F.; Jamroga, W.; Malvone, V.; Mittelmann, M.; Murano, A.; and Perrussel, L. 2022. Reasoning about Human-Friendly Strategies in Repeated Keyword Auctions. In *AAMAS-22*.

Belardinelli, F.; Jamroga, W.; Mittelmann, M.; and Murano, A. 2023. Strategic Abilities of Forgetful Agents in Stochastic Environments. In *Proc. of KR-23*.

Belardinelli, F.; Lomuscio, A.; and Malvone, V. 2018. Approximating Perfect Recall When Model Checking Strategic Abilities. In *Proc. of KR 2018*.

Ben-Or, M.; Kozen, D.; and Reif, J. H. 1986. The Complexity of Elementary Algebra and Geometry. *J. Comput. Syst. Sci.*, 32(2): 251–264.

Berthon, R.; Fijalkow, N.; Filiot, E.; Guha, S.; Maubert, B.; Murano, A.; Pinault, L.; Pinchinat, S.; Rubin, S.; and Serre, O. 2020. Alternating Tree Automata with Qualitative Semantics. *ACM Trans. Comput. Logic*, 22(1): 1–24.

Brázdil, T.; Chatterjee, K.; Chmelik, M.; Fellner, A.; and Kretínský, J. 2015. Counterexample Explanation by Learning Small Strategies in Markov Decision Processes. In *CAV 2015*, LNCS 9206, 158–177. Springer.

Bulling, N.; and Jamroga, W. 2009. What Agents Can Probably Enforce. *Fundam. Informaticae*, 93(1-3): 81–96.

Chatterjee, K.; Chmelik, M.; and Davies, J. 2016. A Symbolic SAT-Based Algorithm for Almost-Sure Reachability with Small Strategies in POMDPs. In *AAAI*, 3225–3232. AAAI Press.

Chatterjee, K.; Doyen, L.; and Henzinger, T. A. 2010. Qualitative Analysis of Partially-Observable Markov Decision Processes. In *Proc. of MFCS*.

Chatterjee, K.; Henzinger, T. A.; and Piterman, N. 2010. Strategy Logic. *Inf. Comput.*, 208(6): 677–693.

Chatterjee, K.; Kößler, A.; and Schmid, U. 2013. Automated analysis of real-time scheduling using graph games. In *HSCC*, 163–172. ACM.

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

Chen, T.; and Lu, J. 2007. Probabilistic alternating-time temporal logic and model checking algorithm. In *Proc. of FSKD*, 35–39.

Clarke, E.; Grumberg, O.; Kroening, D.; Peled, D.; and Veith, H. 2018. *Model checking*. MIT press.

Deuser, K.; and Naumov, P. 2020. On composition of bounded-recall plans. *Artificial Intelligence*, 289: 103399.

Fitchas, N.; Galligo, A.; and Morgenstern, J. 1987. Algorithmes rapides en sequentiel et en parallele pour l'élimination des quantificateurs en Géométrie élementaire. *Seminaire sur les structures algébriques ordonnées*.

Huang, X.; and Luo, C. 2013. A logic of probabilistic knowledge and strategy. In *Proc. of AAMAS 2013*, 845–852.

Huang, X.; Su, K.; and Zhang, C. 2012. Probabilistic Alternating-Time Temporal Logic of Incomplete Information and Synchronous Perfect Recall. In *Proc. of AAAI 2012*.

Jamroga, W.; and Bulling, N. 2011. Comparing variants of strategic ability. In *Proc. of IJCAI 2011*, 252–257. IJCAI/AAAI.

Jamroga, W.; Kurpiewski, D.; and Malvone, V. 2020. Natural strategic abilities in voting protocols. In *Int. Workshop on Socio-Technical Aspects in Security and Trust*, 45–62.

Jamroga, W.; Kurpiewski, D.; and Malvone, V. 2022. How to measure usable security: Natural strategies in voting protocols. *Journal of Computer Security*, 30(3): 381–409.

Jamroga, W.; Malvone, V.; and Murano, A. 2019a. Natural strategic ability. *Artificial Intelligence*, 277: 103170.

Jamroga, W.; Malvone, V.; and Murano, A. 2019b. Natural Strategic Ability under Imperfect Information. In *Proc. AAMAS 2019*.

Junges, S. 2020. *Parameter synthesis in Markov models*. Ph.D. thesis, RWTH Aachen University, Germany.

Junges, S.; Jansen, N.; Wimmer, R.; Quatmann, T.; Winterer, L.; Katoen, J.; and Becker, B. 2018. Finite-State Controllers of POMDPs using Parameter Synthesis. In *Proc. of UAI 2018*.

Kaneko, M.; and Kline, J. J. 1995. Behavior strategies, mixed strategies and perfect recall. *International Journal of Game Theory*, 24: 127–145.

Katoen, J.-P. 2016. The probabilistic model checking landscape. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, 31–45.

Kemeny, J. G.; Snell, J. L.; and Knapp, A. W. 1976. Stochastic Processes. In *Denumerable Markov Chains*, 40–57. Springer.

Laroussinie, F.; and Markey, N. 2015. Augmenting ATL with strategy contexts. *Inf. Comput.*, 245: 98–123.

Lomuscio, A.; and Pirovano, E. 2020. Parameterised verification of strategic properties in probabilistic multi-agent systems. In *Proc. of AAMAS 2020*, 762–770.

Madani, O.; Hanks, S.; and Condon, A. 2003. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artif. Intell.*, 147(1-2): 5–34.

Mogavero, F.; Murano, A.; Perelli, G.; and Vardi, M. Y. 2014. Reasoning About Strategies: On the Model-Checking Problem. *ACM Trans. Comput. Log.*, 15(4).

Nguyen, H. N.; and Rakib, A. 2019. A Probabilistic Logic for Resource-Bounded Multi-Agent Systems. In *Proc. of IJCAI 2019*, 521–527.

Oliehoek, F. A. 2012. Decentralized POMDPs. In *Reinforcement Learning: State-of-the-Art*, 471–503. Springer.

Pajarinen, J.; and Peltonen, J. 2011. Periodic Finite State Controllers for Efficient POMDP and DEC-POMDP Planning. In *Proc. of NIPS 2011.*, 2636–2644.

Song, F.; Zhang, Y.; Chen, T.; Tang, Y.; and Xu, Z. 2019. Probabilistic alternating-time μ-calculus. In *Proc. of AAAI*, 6179–6186.

Vester, S. 2013. Alternating-time temporal logic with finite-memory strategies. In *GandALF 2013*, 194–207.

Vlassis, N.; Littman, M. L.; and Barber, D. 2012. On the Computational Complexity of Stochastic Controller Optimization in POMDPs. *ACM Trans. Comput. Theory*, 4(4): 12:1–12:8.

Wang, Y.; and Dechesne, F. 2009. On expressive power and class invariance. *arXiv preprint arXiv:0905.4332*.