Probabilistic Alternating-time Temporal Logic with Stochastic Abilities

Sarra Zaghbib[1], Gabriel Ballot[1,2], Vadim Malvone[1] and Jean Leneutre[1]
[1]*Télécom Paris, Institut Polytechnique de Paris, Palaiseau, France*
{*sarra.zaghbib, vadim.malvone, jean.leneutre, gabriel.ballot*}*@telecom-paris.fr*

Keywords: multi-agent systems verification, strategic logics, cybersecurity

Abstract: Model checking provides a rigorous means to analyze complex systems by ensuring properties hold across all executions. While originally applied to closed systems, model checking now extends to multi-agent systems, such as distributed protocols, communication systems, robotics, and cybersecurity. A recurring challenge across these domains is reasoning about agents with hidden and uncertain profiles; for example adversarial traders in markets, coordinated users in social media, or attackers in cybersecurity. Addressing this requires logics capable of capturing both probabilistic profiling and reasonig. Alternating-time Temporal Logic (ATL) offers a foundation for reasoning about strategic abilities in multi-agent systems. Extensions such as ATL with Stochastic Abilities (ATL-SA) incorporate stochastic capacities, but existing frameworks remain limited: they can model uncertainty over profiles or allow reasoning about capacities from observed actions, yet not both simultaneously. In this work, we extend ATL-SA with a probabilistic capacity operator, enabling the specification and verification of properties that combine stochastic profiling, inference of hidden information, and strategic reasoning. This framework broadens the scope of formal verification as a general methodology for adversarial and uncertain environments, with applications spanning markets, social platforms, distributed computing, and cybersecurity.

1 INTRODUCTION

In many critical domains, static approaches to system monitoring or defense are easily identified and circumvented by sophisticated adversaries. More adaptive mechanisms address this limitation by evolving their responses based on observed behavior. However, the design of such systems hinges on formal reasoning frameworks capable of modeling strategic interactions between competing agents in multi-agent environments.

Model checking is a formal verification technique employed to automatically ascertain whether a system adheres to a specified set of criteria. This method systematically examines all potential system states to ensure that the system operates as anticipated according to predefined properties. Consequently, it serves as a robust tool for guaranteeing the correctness, safety, and reliability of complex systems. In the realm of Multi-Agent Systems (MAS), where multiple autonomous agents interact, model checking is foundational in verifying that the system performs as intended. It emphasizes the strategies, coordination, and decision-making of agents, considering their diverse objectives and knowledge.

Applying model checking techniques to MAS involves reasoning about both the local strategies of individual agents and the global behavior that emerges from their interactions. This is particularly relevant in adversarial settings, where understanding heterogeneous behaviors and ensuring reliable system responses are key to maintaining integrity.

In such contexts, agents can exhibit different profiles that reflect their capabilities, expertise, and behavioral tendencies. For a system to be robust, it must not only react to observed actions, but also identify and adapt to these underlying profiles. In our logical framework, such a profile is referred to as a capacity, formally defined as a set of feasible actions available to an agent. Reasoning about capacities allows observers to infer agent types from their behavior and tailor strategies accordingly.

Existing logics like Alternating-time Temporal Logic with Stochastic Abilities (ATL-SA) (Ballot et al., 2025b) provide foundational tools for this purpose. However, it has critical limitations in the context of adaptive systems. ATL-SA assigns agents a distribution *a priori* over possible profiles but cannot reason about the *posterior* probability that an agent has a certain profile based on observed behav-

ior. To address this gap, we introduce Probabilistic Alternating-Time Temporal Logic with Stochastic Abilities (PATL-SA) as an extension of ATL-SA that enriches the strategic layer with reasoning over posterior capacity probability, conditioned on observed behavior.

Contributions. The contributions of this paper are the following:

1. We introduce Probabilistic Alternating-Time Temporal Logic with Stochastic Abilities (PATL-SA), a novel logic that combines the probabilistic capacity modeling of ATL-SA with posterior capacity probability reasoning, enabling capacity inference with probability.
2. We define the formal semantics of PATL-SA and provide a model-checking algorithm. We prove that this algorithm is sound and complete and that its complexity remains consistent with that of ATL-SA.
3. We demonstrate the practical utility of PATL-SA through a cybersecurity case study involving adaptive honeypots, showing that it enables both precise attacker profiling and effective dynamic defense synthesis.

Outline. Section 2 reviews the related literature and Section 3 presents the general notation and background. Section 4 introduces PATL-SA's syntax and semantic. Section 5 studies its model checking problem. Section 6 shows PATL-SA's applicability in a cybersecurity setting. Finally, Section 7 concludes the paper.

2 RELATED WORK

In 2002, Alur et al. introduced Alternating-Time Temporal Logic (ATL) (Alur et al., 2002) to verify strategic properties of concurrent games. Since then, different extensions have been proposed to provide more expressivity to the logics. This includes epistemic properties (van der Hoek and Wooldridge, 2002; Schobbens, 2004; Jamroga and van der Hoek, 2004; Schnoor, 2010), probabilistic outcomes (Chen and Lu, 2007; Huang et al., 2012; Mittelmann et al., 2023; Belardinelli et al., 2024; Berthon et al., 2024; Ballot et al., 2025b), quantitative reasoning (Catta et al., 2024a; Ferrando et al., 2024; Nguyen and Rakib, 2019), real-time constraints (Brihaye et al., 2005; David et al., 2014), strategy specifications (Walther et al., 2007; Ågotnes et al., 2007; Mogavero et al., 2014), or action specifications (Ballot et al., 2024; Belardinelli et al., 2017; Belardinelli et al., 2020; Ågotnes, 2006; Herzig et al., 2013). Thanks to these extensions, ATL-related logics can specify more accurate properties of systems and broaden possible use cases.

We discuss, in more details, three closely related logics.

CapATL. Capacity Alternating-Time Temporal Logic (CapATL) (Ballot et al., 2024) extends ATL (Ballot et al., 2025b) by introducing agent capacities. Formally, CapATL is interpreted over Capacity Concurrent Game Structures (CapCGS), where each agent's chosen capacity restricts its actions, and agents can observe only states and their own actions, but not others' capacities or actions. A capacity knowledge operator $\left(K^a_{\text{cap}}(\varphi)\right)$ specifies what an agent a can infer about another agent's capacities over time. For example, a honeypot defender may infer whether an attacker has advanced exploitation skills based on the actions observed; such strategic inferences are naturally expressed via CapATL's memoryful strategies and knowledge operator. Although CapATL accounts for imperfect information, which leads to undecidability in the general case (Dima and Ţiplea, 2011), CapATL model checking remains decidable by limiting uncertainty to other agents' actions. However, CapATL lacks probabilistic modeling, which hinders its ability to capture partially stochastic or uncertain attacker behaviors, an important limitation if defenders need to weigh probabilities of different attacks or outcomes. CapATL was also extended to imperfect information settings (Ballot et al., 2025a), where states might be indistinguishable.

ATL-SA. The logic Alternating-time Temporal Logic with Stochastic Abilities (ATL-SA) (Ballot et al., 2025b) builds on ATL (Alur et al., 2002) by introducing probabilistic agent capacities (corresponding to their profiles). An agent's capacity is the set of available actions for that agent. In real-world applications, domain knowledge or prior statistical analysis may provide a probability distribution over possible agent profiles, influencing their available actions. Key features of ATL-SA include stochastic agent modeling (agents are assigned probabilistic profiles that determine their abilities), strategic verification with probability thresholds, and decision-making under uncertainty, modeling how agents can optimize their strategies when faced with probabilistic adversaries. ATL-SA formalizes these aspects using Concurrent Game Structures with Stochastic Abilities (CGS-SA),

where each agent's profile influences available actions probabilistically. In practical terms, domain expertise or statistical analysis might provide a capacity distribution (e.g., a 70% chance that an attacker is "advanced" and a 30% chance that he is a "beginner"), influencing which actions the attacker can use. The simple following formula expresses that the defender D can ensure, with at least 80% probability, that the system eventually reaches a secure state.

$$\langle D \rangle^{\geq 0.8} \mathsf{F}(secure)$$

In practice, a winning strategy might first gather information on the attacker's profile by guiding the adversary to perform actions specific to a certain capacity, then adapt its plan to guarantee reaching a secure state, exploiting the inferred attacker limitations. While ATL-SA elegantly blends stochastic modeling with strategic reasoning, it lacks the expressivity to reason about posterior capacity probabilities, such as specifying a strategy that aims at identifying a profile with high probability.

PATL. Probabilistic Alternating-Time Temporal Logic (PATL) was introduced by Chen and Lu (2007) as a probabilistic extension of ATL (Alur et al., 2002). PATL enriches strategic reasoning by allowing the specification of probability thresholds over temporal objectives, enabling statements about the likelihood that a coalition can enforce a given property against adversarial behavior. Formulas in PATL existentially quantify over strategies of a coalition while accounting for the probabilistic evolution of the system.

Semantically, once a coalition's strategy is fixed, the strategies of the opposing agents remain unrestricted. As a result, the outcome of a coalition strategy is characterized by a set of probability measures over execution paths, each corresponding to a possible response of the opponents. Satisfaction of a PATL formula is then defined by universally quantifying over these induced probability measures.

While PATL provides a powerful formalism for probabilistic strategic verification in stochastic game structures, it associates probabilities with state transitions rather than with agents' abilities or action availability. Consequently, it cannot capture uncertainty or belief evolution regarding agents' capabilities based on observed behavior, a limitation that motivates later extensions such as ATL-SA. Importantly, probabilities on state transitions, as studied in PATL, are orthogonal to probabilities on agents' capacities. While PATL-SA focuses on uncertainty over capacities, combining both dimensions by also considering probabilistic transitions constitutes a natural direction for future work.

3 NOTATIONS AND BACKGROUND

The set of non-negative integers is denoted by $\mathbb{N} = \{0, 1, \dots\}$. Let A be a finite set. We denote by $\mathcal{P}(A)$ the *power set* of A. A (discrete) *probability distribution* over A is a function $\mu : A \to [0,1]$,where $[0,1]$ denotes the unit real interval, such that $\sum_{a \in A} \mu(a) = 1$. The value $\mu(a)$ represents the probability assigned to the element $a \in A$. The set of all probability distributions over A is denoted by $\mathcal{D}(A)$. A probability distribution μ over A induces a probability operator $\mathbb{P} : \mathcal{P}(A) \to [0,1]$ such that, for $A' \subseteq A$, $\mathbb{P}(A') = \sum_{a \in A'} \mu(a)$. For $A_1, A_2 \subseteq A$ with $\mathbb{P}(A_2) \neq 0$, the *conditional probability* of A_1 given A_2 is $\mathbb{P}(A_1 \mid A_2) = \mathbb{P}(A_1 \cap A_2)/\mathbb{P}(A_2)$.

Concurrent Game Structures (CGS) (Alur et al., 2002) are formal models used to represent multi-agent systems in which agents make decisions simultaneously, and the outcome depends on the combination of these choices. In a CGS, each agent independently selects an action from a set of available options, and the system transitions to a new state based on the joint action of all agents. This framework captures the interaction dynamics in environments where multiple autonomous agents operate concurrently, often with distinct goals and incomplete knowledge.

Concurrent Game Structures with Stochastic Abilities (CGS-SA) (Ballot et al., 2025b) extend CGS by associating each agent with a probability distribution over capacities that is, over subsets of feasible actions. In this model, an agent's capacity (its set of admissible actions) is not fixed but probabilistically sampled from a known distribution at the beginning of the execution. This framework is particularly well-suited for scenarios where agents may exhibit different profiles such as skill levels, resources, or intent and where uncertainty plays a central role in decision-making. Such modeling is especially relevant in adversarial and dynamic environments like cybersecurity, where behavioral variability is expected (Iqbal and Pearson, 2017). By capturing both strategic reasoning and probabilistic uncertainty, CGS-SA provides a powerful foundation for verifying adaptive behaviors and synthesizing robust strategies under profile uncertainty.

Formally, a CGS-SA is a tuple

$$\mathcal{M} = \big(St, Ag, Ac, \Pi, \pi, d, o, \Delta_1, \dots, \Delta_{|Ag|}\big),$$

where:

- St is a finite set of states.
- Ag is a finite set of agents.
- Ac is a finite set of actions.

- Π is a finite (it will be easier) set of atomic propositions.
- $\pi : St \to \mathcal{P}(\Pi)$ is a labeling function mapping states to propositions.
- $d : Ag \times St \to \mathcal{P}(Ac)$ is the function that defines the set of possible actions that each agent can make based on the current state of the system.
- $o : St \times Ac^{|Ag|} \to St$ is a partial function defined for all joint actions $\alpha \in d(1,s) \times \cdots \times d(n,s)$ available in each state s
- $\Delta_1, \ldots, \Delta_{|Ag|}$ are probability distributions over action subsets (capacities) for each agent ($\Delta_a \in \mathcal{D}(\mathcal{P}(Ac))$).

For the rest of the article, we consider a general CGS-SA $\mathcal{M}$ of the form $\mathcal{M} = (St, Ag, Ac, \Pi, \pi, d, o, \Delta_1, \ldots, \Delta_n)$, with n agents ($n \in \mathbb{N}$). A *path* ρ in $\mathcal{M}$ is an infinite sequence of alternating states and joint actions:

$$\rho = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} s_2 \xrightarrow{\alpha_2} \cdots$$

where, for each index $i \in \mathbb{N}$, the joint action $\alpha_i \in d(1,s_i) \times \cdots \times d(n,s_i)$ represents the action selected by each agent in state s_i, and the transition function satisfies $o(s_i, \alpha_i) = s_{i+1}$. Moreover, for an agent a and a joint action α, $\alpha[a]$ denotes the action of agent a in α. Given such a path ρ, we write $\rho[i] = s_i$ to denote the i-th state on the path, and $\rho_{\leq i}$ to refer to the finite prefix of the path up to (and including) state s_i, along with all preceding joint actions. A *capacity assignment* is a function $\kappa : Ag \to \mathcal{P}(Ac)$ that assigns to each agent a subset of actions: its capacity. A capacity assignment is called *complete* if it provides a capacity for every agent in Ag. We denote by Γ be the set of all complete capacity assignments with positive probability. Given a finite path $\rho = s_0 \xrightarrow{\alpha_0} \cdots \xrightarrow{\alpha_{i-1}} s_i$, we define the set of capacity assignments *consistent* with the history as:

$$\mathcal{K}(\rho) = \left\{ \kappa \in \Gamma \,\middle|\, \begin{array}{l} \forall j \in \{0, \ldots, i-1\}, \\ \forall a \in Ag,\ \alpha_j[a] \in \kappa(a) \end{array} \right\}.$$

This means $\mathcal{K}(\rho)$ contains all assignments in which every agent has only performed actions that are allowed by their assigned capacity.

The probability distributions $\Delta_1, \ldots, \Delta_n$ are assumed independent. As such, they induce a probability measure $\mathbb{P}$ over complete capacity assignments such that, for $K \subseteq \Gamma$, we have $\mathbb{P}(K) = \sum_{\kappa \in K} \prod_{a \in Ag} \Delta_a(\kappa(a))$. It is the probability that the complete capacity assignment is among K.

A *strategy* is a function f which takes as input a complete capacity assignment κ and a finite path (called *history*) ρ and returns an action $f(\kappa, \rho)$. A *strategy assignment* σ is a partial function that assigns strategies to agents such that, for any finite path ρ and any complete capacity assignment κ, the strategy assigned to an agent a (of its domain) returns an action $\alpha = \sigma(a)(\kappa, \rho)$ verifying $\alpha \in d(a, s_i) \cap \kappa(a)$, where s_i is the last state of ρ. That is, agent a's strategy selects a legal action at state s_i that is also consistent with the agent's assigned capacity $\kappa(a)$. We say that σ is a strategy assignment for a coalition of agents Y when the domain of σ is Y. A strategy assignment is *complete* when its domain is the set of all agents. Moreover, a strategy assignment σ for Y is *uniform* iff, for all $a \in Y$, finite path ρ, and $\kappa, \kappa' \in \Gamma$, we have $\kappa(a) = \kappa'(a)$ implies $\sigma(\kappa, \rho) = \sigma(\kappa', \rho)$. It means that the agents use the same action whatever the capacity of other agents, encoding the fact that agents do not know others' capacity.

For a state s_0, a complete strategy assignment σ, and a complete capacity assignment κ, the *outcome path* $\text{out}(\sigma, \eta, \kappa)$ is the unique infinite path $\rho = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} s_2 \cdots$ such that, for all agents a and $j \in \mathbb{N}$ such that $j \geq i$, the joint action α_j satisfies $\alpha_j[a] = \sigma(a)(\kappa, \rho_{\leq j})$. In words, $\text{out}(\sigma, \eta, \kappa)$ yields the unique infinite path that extends the finite prefix η, progressing via the transition function o according to the joint actions selected by the agents' strategies under the complete capacity assignment κ. This outcome reflects how the system would evolve if each agent acts according to its assigned strategy and capacity.

The next section presents our novel logic to express CGS-SA's properties.

4 PROBABILISTIC ALTERNATING-TIME TEMPORAL LOGIC WITH STOCHASTIC ABILITIES

To overcome the limitations of CapATL and ATL-SA, we introduce *Probabilistic Alternating-Time Temporal Logic with Stochastic Abilities* (PATL-SA), building upon the strengths of both logics. It incorporates stochastic agent capacities where each agent is associated with a probability distribution over possible profiles and enables reasoning about these capacities in light of interaction history. Unlike ATL-SA, which models uncertainty via probabilistic capacities but does not track capacities' probabilities over time explicitly, PATL-SA allows defenders to assess the likelihood of an agent possessing a given capacity as the system evolves. This is inspired from CapATL which allows reasoning about feasible capacities for

a given history, but considers deteministic capacities only. This expressiveness makes PATL-SA particularly well-suited for modeling adversarial environments such as cybersecurity, where defenders must adapt to evolving threats and incomplete information. PATL-SA provides the tools to represent both probabilistic variability and inference-driven adaptation in agent reasoning, which are essential for modern systems like adaptive honeypots.

4.1 Syntax

Given a set Ag of agents, a set Π of atomic propositions, and a set Ac of actions, the syntax of PATL-SA formulas is defined as follows:

$$\phi ::= l \mid \neg\phi \mid \phi \wedge \phi \mid \langle Y \rangle^{\bowtie p}\psi \mid \mathsf{P}^{\bowtie p}(\varphi)$$
$$\psi ::= \mathsf{X}\phi \mid \phi \,\mathsf{U}\, \phi \mid \phi \,\mathsf{R}\, \phi$$
$$\varphi ::= a \mapsto c \mid \neg\varphi \mid \varphi \wedge \varphi$$

Where $l \in \Pi$ is an atomic proposition, $a \in Ag$ is an agent, $Y \subseteq Ag$ is a coalition of agents, and $c \subseteq Ac$ is a capacity, i.e., a set of actions. The comparison operator $\bowtie$ belongs to the set $\{<, \leq, \geq, >\}$.

This syntax extends ALT-SA's syntax with the operator $\mathsf{P}^{\bowtie p}(\varphi)$ and the subformulas φ. Intuitively, the strategy formula $\langle Y \rangle^{\bowtie p}\psi$ expresses that the coalition Y has a strategy to ensure that the temporal goal ψ holds with probability $\bowtie p$. The operator $\mathsf{P}^{\bowtie p}(\varphi)$ evaluates whether the probability that a capacity formula φ holds—given the current history—compares to p with $\bowtie$. Temporal operators include $\mathsf{X}\phi$ (next), $\phi_1 \,\mathsf{U}\, \phi_2$ (until), and $\phi_1 \,\mathsf{R}\, \phi_2$ (release), following standard temporal logic conventions. The atomic capacity formula $a \mapsto c$ asserts that agent a's current capacity is exactly the capacity (i.e., action set) c. The Boolean operators $\rightarrow$, $\vee$, and $\leftrightarrow$ are defined as usual from $\neg$ and $\wedge$. The Boolean values $\top$ and $\bot$ denote respectively true and false. Moreover, the "finally" operator is $\mathsf{F}\phi := \top \,\mathsf{U}\, \phi$ and the "globally" operator is $\mathsf{G}\phi := \bot \,\mathsf{R}\, \phi$.

Example. The following formula illustrates the behavior of the defender D in ensuring system security:

$$\langle D \rangle^{\geq 0.9}\, \mathsf{F}\big(\neg\text{Hacked} \wedge (\mathsf{P}^{\geq 0.8}(A \mapsto \text{Advanced}) \vee \mathsf{P}^{\geq 0.8}(A \mapsto \text{Beginner}))\big)$$

This formula expresses that the defender D can enforce, with at least 90% probability, a future state in which the system is not hacked and the attacker A is identified (with at least 80% probability) as either an advanced or a beginner attacker. This captures both the goal of security preservation and adversary profiling with probabilistic confidence.

4.2 Semantics

Let $\mathcal{M}$ be such a structure, ρ be an infinite path, κ be a complete capacity assignment, and $i \in \mathbb{N}$. PATL-SA semantic is defined inductively through the following satisfaction relation:

- $(\mathcal{M}, \rho, i, \kappa) \models l$ iff $l \in \pi(\rho[i])$
- $(\mathcal{M}, \rho, i, \kappa) \models a \mapsto c$ iff $\kappa(a) = c$
- $(\mathcal{M}, \rho, i, \kappa) \models \mathsf{P}^{\bowtie p}(\varphi)$ iff

  $$\mathbb{P}\big(\{\, \kappa' \in \tau \mid (\mathcal{M}, \rho, i, \kappa') \models \varphi \,\} \mid \mathcal{K}(\rho_{\leq i})\big) \bowtie p$$

  where $\mathcal{K}(\rho_{\leq i}) \subseteq \tau$ is the set of capacity assignments consistent with the observed history $\rho_{\leq i}$.
- $(\mathcal{M}, \rho, i, \kappa) \models \langle Y \rangle^{\bowtie p}\psi$ iff there exists a uniform strategy assignment σ_Y for the coalition Y such that for all strategy assignments $\sigma_{\overline{Y}}$ for the remaining agents $\overline{Y} = Ag \setminus Y$, we have:

  $$\mathbb{P}\big(\{\kappa' \in \tau \mid (\mathcal{M}, \text{out}(\sigma, \rho[i], \kappa'), 1, \kappa') \models \psi\}\big) \bowtie p$$

  where $\sigma = \sigma_Y \oplus \sigma_{\overline{Y}}$ denotes the complete strategy assignment combining strategy assignments with disjoint domain.
- $(\mathcal{M}, \rho, i, \kappa) \models \neg\vartheta$ iff $(\mathcal{M}, \rho, i, \kappa) \not\models \vartheta$
- $(\mathcal{M}, \rho, i, \kappa) \models \vartheta_1 \wedge \vartheta_2$ iff $(\mathcal{M}, \rho, i, \kappa) \models \vartheta_1$, and $(\mathcal{M}, \rho, i, \kappa) \models \vartheta_2$.
- $(\mathcal{M}, \rho, i, \kappa) \models \phi_1 \mathsf{U} \phi_2$ iff there exists $j \in \mathbb{N}$ with $j \geq i$ such that $(\mathcal{M}, \rho, j, \kappa) \models \phi_2$ and for all $k \in \mathbb{N}$ with $i \leq k < j$, we have $(\mathcal{M}, \rho, k, \kappa) \models \phi_1$
- $(\mathcal{M}, \rho, i, \kappa) \models \phi_1 \mathsf{R} \phi_2$ iff either
  - for all $j \geq i$, $(\mathcal{M}, \rho, j, \kappa) \models \phi_2$, or
  - there exists $j \geq i$ such that $(\mathcal{M}, \rho, j, \kappa) \models \phi_1 \wedge \phi_2$, and for all k with $i \leq k < j$, we have $(\mathcal{M}, \rho, k, \kappa) \models \phi_2$

Note that for a PATL-SA formula ϕ, the satisfaction relation $(\mathcal{M}, \rho, i, \kappa) \models \phi$ depends only on $\mathcal{M}$, the prefix $\rho_{\leq i}$, and the formula ϕ. Hence, we may write $(\mathcal{M}, \rho_{\leq i}) \models \phi$. In particular, for a given state $s \in St$, $(\mathcal{M}, s) \models \phi$ is well defined. PATL-SA's semantic extends ATL-SA's uniform semantic but we could also extends ATL-SA's distributed semantic in similar ways (where agents in the strategic coalition share their assigned capacity).

In the next section, we study the PATL-SA *model-checking problem*, which takes as input a CGS-SA $\mathcal{M}$, an initial state $s \in St$, and a PATL-SA formula ϕ, and returns whether $(\mathcal{M}, s) \models \phi$.

5 PATL-SA MODEL CHECKING

We denote by MC the global model-checking procedure for PATL-SA formulas. It recursively eliminates nested strategic subformulas by replacing each inner occurrence with a fresh atomic proposition and then delegates temporal verification to the auxiliary procedure MCTemp. Theorem 5.1 shows that the function MCTemp from Algorithm 2 (using functions SuccN, SuccU, and SuccR from Algorithm 1 and MCNoStrat and MCCap from Algorithm 2) enables checking that a winning strategy exists for a PATL-SA strategy formula without nested strategy subformulas, given a subset of complete capacity assignments for which the strategy wins with probability 1.

Although the structure of our model-checking algorithm resembles that of ATL-SA, PATL-SA introduces a probabilistic evaluation phase (MCNoStrat and MCCap) that computes conditional probabilities over capacity assignments consistent with the observed history. This additional layer allows evaluating probability constraints such as $\mathsf{P}^{\geq p}(\varphi)$ within temporal goals, which CapATL cannot express.

Theorem 5.1. *Let s be a state of a CGS-SA $\mathcal{M}$, $K \subseteq \Gamma$, $Y \subseteq Ag$, and ψ be a temporal formula without strategy subformulas. The following propositions are equivalent:*

1. *There is a uniform strategy assignment σ_Y for Y such that, for all strategy assignment $\sigma_{Ag\setminus Y}$ for $Ag \setminus Y$ and all $\kappa \in K$, we have:*
$$(\mathcal{M}, \mathrm{out}(\sigma_Y \oplus \sigma_{Ag\setminus y}, s, \kappa), 1, \kappa) \models \psi$$
2. $\mathsf{MCTemp}(\mathcal{M}, s, Y, \psi) = \top$.

Proof. The proof relates to the similar theorem in ATL-SA (Ballot et al., 2025b) where the second proposition is the satisfaction of an ATL formula on a specific CGS. The algorithms from this theorem perform the explicit ATL model checking on the CGS from (Ballot et al., 2025b)'s proof. The intuition in SuccN, SuccU, and SuccR is that, after termination check, agents in Y commit to a action for each of their possible capacities (the possible capacity assignments is the argument K in the algorithms). This commitments are the functions $\{f_y\}_{y\in Y}$, which model the uniformity of Y's strategy assignment (c.f. (Ballot et al., 2025b)). Then, for all possible opponents response and capacity assignments in K, the temporal formula is verified recursively (c.f. (Alur et al., 2002) for ATL model checking). In SuccU and SuccR, the argument K is updated as the subset of complete capacity assignments which are consistent with the transition and where Y performs the committed action. The argument K_p keeps track of the set of all consistent capacity assignments (whatever the commitment of Y) and allows verifying subformulas of the form $\mathsf{P}^{\bowtie p}(\varphi)$. Indeed, there are calls to MCNoStrat for non-strategic subformula, which is sound because it performs simple Boolean verification and computes the conditional probability as expected. □

As in ATL-SA, we can reduce the analysis to formulas where $\bowtie \in \{\geq, >\}$ (c.f. (Ballot et al., 2025b)). Indeed, let $\overline{\bowtie} \in \{<, \leq, \geq, >\}$ such that $p_1 \bowtie p_2$ iff $p_2 \overline{\bowtie} p_1$, for all $p_1, p_2 \in [0,1]$, and let $\overline{\mathsf{X}\,\phi} = \mathsf{X}(\neg\phi)$, $\overline{\phi_1 \,\mathsf{U}\, \phi_2} = (\neg\phi_1)\ \mathsf{R}\ (\neg\phi_2)$, and $\overline{\phi_1 \,\mathsf{R}\, \phi_2} = (\neg\phi_1)\ \mathsf{U}\ (\neg\phi_2)$, which is the negation of a temporal formula. Then, the formulas $\langle Y\rangle^{\bowtie p}\psi$ and $\langle Y\rangle^{\overline{\bowtie}(1-p)}\overline{\psi}$ are equivalent. We define the function MC from Algorithm 2, which implements the model checking for any PATL-SA formula. This function inductively checks nested strategy subformulas with a call to MCTemp and update the structure and formula to remove strategic operators. By Theorem 5.1, the function MC is a sound model-checking algorithm for PATL-SA. This leads to the following Theorem 5.2.

Theorem 5.2. *PATL-SA model checking for formulas of the form $\langle Y\rangle^{\bowtie p}\psi$ without nested strategy formulas is* NEXPTIME*-complete. For general formulas, it is in Δ_2^{E}.*[1]

Proof. The lower bounds come from ATL-SA model checking (Ballot et al., 2025b). For the upper bound, notice that MC can choose K nondeterministically and the rest of its execution (without the call to MCTemp) is polynomial. Calls to MCTemp take an exponential time with respect to $\mathcal{M}$ and ϕ. Indeed, suppose we call SuccU from MCTemp. The last arguments K_m in the m^{th} recursive call to SuccU can be encoded as $K_m = \{\kappa \in K_1 \mid \forall a \in Ag, \kappa(a) \in C_a^m\}$, for some sets of capacities $C_1^m, \ldots, C_n^m$ with positive probabilities for each respective agent (and the same holds for the second last argument). Consequently, the number of different arguments for SuccU is exponential. Moreover, the loops of SuccU are exponential so SuccU executes in exponential time overall, and so does MCTemp. □

To conclude this section, while PATL-SA extends ATL-SA with a new operator to reason about the probability that agents have some capacity according to the history, the model-checking complexity remain the same as ATL-SA. The next section presents a cybersecurity illustration which demonstrates PATL-SA's applicability.

[1] Δ_2^{E} is the class of problems that can be solved in polynomial time with calls to an NEXPTIME oracle.

Algorithm 1 Temporal formula handlers with a global table U and R.

function SUCCN($\mathcal{M}, s, Y, \phi, K$)
  for all $a \in Ag$ **do** $C_a \leftarrow \{\kappa(a) \mid \kappa \in K\}$
  for all $\{f_y\}_{y \in Y}$ where, for all $y \in Y$, $f_y : C_y \to Ac$ s.t. $f_y(c) \in d(a,s) \cap c$ **do**
    valid $\leftarrow \top$
    for all $\kappa \in K$ and **all** $\{\alpha_z\}_{z \in \overline{Y}}$ s.t. for all $z \in \overline{Y}$, $\alpha_z \in d(z,s) \cap \kappa(z)$ **do**
      for all $y \in Y$ **do** $\alpha_y \leftarrow f_y(\kappa(y))$
      $s' \leftarrow o(s, \alpha_1, \ldots, \alpha_n)$; $K' \leftarrow \{\kappa \in K \mid \forall a \in Ag, \alpha_a \in \kappa(a)\}$
      if $\neg$MCNoStrat$(\mathcal{M}, s', K'_p, \phi)$ **then** valid $\leftarrow \bot$; **break**
    if valid **then return** $\top$
  return $\bot$
end function

function SUCCU($\mathcal{M}, s, Y, \phi_1, \phi_2, K_p, K$)
  if $U[s, Y, \phi_1, \phi_2, K_p, K]$ exists **then return** $U[s, Y, \phi_1, \phi_2, K_p, K]$
  if MCNoStrat$(\mathcal{M}, s, K_p, \phi_2)$ **then save in** U **and return** $\top$
  if $\neg$MCNoStrat$(\mathcal{M}, s, K_p, \phi_1)$ **then save in** U **and return** $\bot$
  $U[s, Y, \phi_1, \phi_2, K_p, K] \leftarrow \top$
  for all $a \in Ag$ **do** $C_a \leftarrow \{\kappa(a) \mid \kappa \in K\}$
  for all $\{f_y\}_{y \in Y}$ where, for all $y \in Y$, $f_y : C_y \to Ac$ s.t. $f_y(c) \in d(a,s) \cap c$ **do**
    valid $\leftarrow \top$
    for all $\kappa \in K$ and **all** $\{\alpha_z\}_{z \in \overline{Y}}$ s.t. for all $z \in \overline{Y}$, $\alpha_z \in d(z,s) \cap \kappa(z)$ **do**
      for all $y \in Y$ **do** $\alpha_y \leftarrow f_y(\kappa(y))$
      $s' \leftarrow o(s, \alpha_1, \ldots, \alpha_n)$
      $K'_p \leftarrow \{\kappa' \in K_p \mid \forall a \in Ag, \alpha_a \in \kappa'(a)\}$
      $K' \leftarrow \{\kappa' \in K \mid \forall y \in Y, f_y(\kappa(y)) = \alpha_y \text{ and } \forall z \in \overline{Y}, \alpha_z \in \kappa'(z)\}$
      if $\neg$SuccU$(\mathcal{M}, s', Y, \phi_1, \phi_2, K'_p, K')$ **then** valid $\leftarrow \bot$; **break**
    if valid $= \top$ **then save in** U **and return** $\top$
  save in U **and return** $\bot$
end function

function SUCCR($\mathcal{M}, s, Y, \phi_1, \phi_2, K_p, K_{\langle\rangle}$)
  if $R[s, Y, \phi_1, \phi_2, K_p, K]$ exists **then return** $R[s, Y, \phi_1, \phi_2, K_p, K]$
  if $\neg$MCNoStrat(s, K_p, ϕ_2) **then save in** R **and return** $\bot$
  if MCNoStrat(s, K_p, ϕ_1) **then save in** R **and return** $\top$
  $R[s, Y, \phi_1, \phi_2, K_p, K] \leftarrow \bot$
  for all $a \in Ag$ **do** $C_a \leftarrow \{\kappa(a) \mid \kappa \in K\}$
  for all $\{f_y\}_{y \in Y}$ where, for all $y \in Y$, $f_y : C_y \to Ac$ s.t. $f_y(c) \in d(a,s) \cap c$ **do**
    valid $\leftarrow \top$
    for all $\kappa \in K$ and **all** $\{\alpha_z\}_{z \in \overline{Y}}$ s.t. for all $z \in \overline{Y}$, $\alpha_z \in d(z,s) \cap \kappa(z)$ **do**
      for all $y \in Y$ **do** $\alpha_y \leftarrow f_y(\kappa(y))$
      $s' \leftarrow o(s, \alpha_1, \ldots, \alpha_n)$
      $K'_p \leftarrow \{\kappa' \in K_p \mid \forall a \in Ag, \alpha_a \in \kappa'(a)\}$
      $K' \leftarrow \{\kappa' \in K \mid \forall y \in Y, f_y(\kappa(y)) = \alpha_y \text{ and } \forall z \in \overline{Y}, \alpha_z \in \kappa'(z)\}$
      if $\neg$SuccR$(s', Y, \phi_1, \phi_2, K'_p, K')$ **then** valid $\leftarrow \bot$; **break**
    if valid $= \top$ **then save in** R **and return** $\top$
  save in R **and return** $\bot$
end function

Algorithm 2 PATL-SA model-checking algorithms.

function MCNoStrat($\mathcal{M}, s, K, \phi$)
  if $\phi = l$ **then return** whether $l \in \pi(s)$
  if $\phi = \neg\phi'$ **then return** $\neg$MCNoStrat($\mathcal{M}, s, K, \phi'$)
  if $\phi = \phi_1 \wedge \phi_2$ **then return** MCNoStrat($\mathcal{M}, s, K, \phi_1$) $\wedge$ MCNoStrat($\mathcal{M}, s, K, \phi_2$)
  if $\phi = \mathsf{P}^{\bowtie}(\phi)$ **then**
    $p_1 \leftarrow 0$; $p_2 \leftarrow 0$
    for all $\kappa' \in K$ **do**
      if MCCap(κ', ϕ) **then** $p_1 = p_1 + \prod_{a \in Ag} \Delta_a(\kappa'(a))$
      $p_2 = p_2 + \prod_{a \in Ag} \Delta_a(\kappa'(a))$
    return whether $p_1/p_2 \bowtie p$
end function

function MCCap(κ, ϕ)
  if $\phi = a \mapsto c$ **then return** whether $c = \kappa(a)$
  if $\phi = \neg\phi'$ **then return** $\neg$MCCap(κ, ϕ')
  if $\phi = \phi_1 \wedge \phi_2$ **then return** MCCap(κ, ϕ_1) $\wedge$ MCCap(κ, ϕ_2)
end function

function MCTemp($\mathcal{M}, s, Y, \psi, K$)
  if $\psi = \mathsf{X}\phi'$ **then return** SuccN($\mathcal{M}, s, Y, \phi', K$)
  if $\psi = \phi_1 \,\mathsf{U}\, \phi_2$ **then return** SuccU($\mathcal{M}, s, Y, \phi_1, \phi_2, \Gamma, K$)
  if $\psi = \phi_1 \,\mathsf{R}\, \phi_2$ **then return** SuccR($\mathcal{M}, s, Y, \phi_1, \phi_2, \Gamma, K$)
end function

function MC($\mathcal{M}, s, \phi$)
  if there is an innermost strategic subformula ϕ_s in ϕ **then**
    if $\phi_s = \langle Y \rangle^{\bowtie p}\psi$ with $\bowtie \in \{\leq, <\}$ **then** $\phi_s \leftarrow \langle Y \rangle^{\overline{\bowtie}(1-p)}\overline{\psi}$
    if $\phi_s = \langle Y \rangle^{\bowtie p}\psi$ with $\bowtie \in \{>, \geq\}$ **then**
      for all state s in $\mathcal{M}$ **do**
        if there is $K \subseteq \Gamma$ s.t. $\mathbb{P}(K) \bowtie p$ and MCTemp($\mathcal{M}, s, Y, \psi, K$) **then**
          give label l_{ϕ_s} to s in $\mathcal{M}'$
      $\phi' \leftarrow$ replace ϕ_s by l_{ϕ_s} in ϕ
      return MC($\mathcal{M}', s, \phi'$)
  return MCNoStrat($\mathcal{M}, s, \Gamma, \cdot, \phi$)
end function

6 CYBERSECURITY ILLUSTRATION

We now demonstrate PATL-SA on a cybersecurity case study. The goal is to illustrate how probabilistic capacity inference supports attacker profiling and adaptive defense synthesis. To demonstrate the power of formal verification in cybersecurity, we model a corporate network defended by an *adaptive honeypot system*, which dynamically mimics real services and introduces decoy vulnerabilities to engage and profile attackers. The interaction between attacker and defender is captured as a stochastic game, and we verify key security objectives in PATL-SA.

6.1 System Model

The adaptive honeypot model progresses through a finite set of network configurations as the defender and attacker interact. Initially, the system resides in state I, representing the untouched network before any engagement. Depending on the defender's decoy choice and the attacker's probing, the game may transition to one of the P_i states, where service s_i appears vulnerable. If the attacker attempts an exploit on these services, they may enter the intermediate state E_{13}, reflecting partial penetration efforts on P_1 or P_2. At this point, the defender can lure the attacker further into the honeypot, moving them into the winning decoy state E_W, where a long-running fake exploit (W) fully engages the adversary. Finally, the model includes a true compromise state H, indicating that the attacker has breached real assets. Figure 1 shows these states and their interconnections. Each state's labels are their respective name.

6.2 Verification Objectives

The following formulas are presented as illustrative examples, demonstrating the expressiveness and efficiency of PATL-SA in capturing key security and intelligence goals; they are not intended as prescriptive objectives for any specific deployment.

1. **Attacker Identification.** The honeypot can infer the attacker skill level with 80% confidence:

$$\phi_{id} = \mathsf{P}^{\geq 0.8}(A \mapsto \text{Beg}) \vee \mathsf{P}^{\geq 0.8}(A \mapsto \text{Int}) \vee P^{\geq 0.8}(A \mapsto \text{Adv})$$

2. **Strategic property.** There exists a defender strategy that, with probability at least 90%, eventually both keeps the system safe ($\neg H$) and reaches the profiling objective of ϕ_{id}.

$$\phi = \langle D \rangle^{\geq 0.9} \mathsf{F}\big(\neg H \wedge \phi_{id}\big)$$

By model-checking these properties on the adaptive honeypot CGS-SA (for instance, checking whether the initial state I from Figure 1 satisfies $I \models \phi$), we obtain quantitative guarantees guiding the design of dynamic deception strategies in real-world cybersecurity defenses.

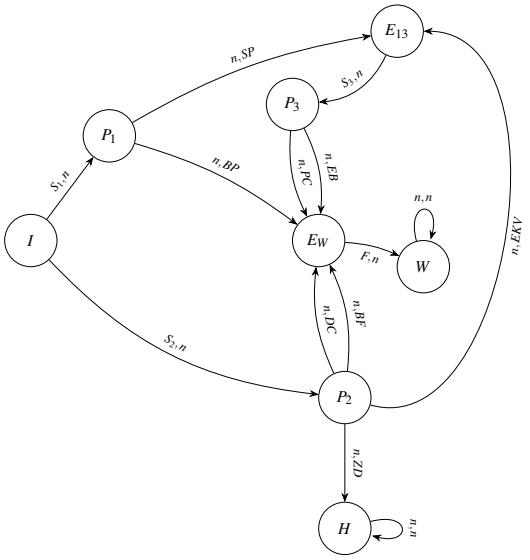

Figure 1: CGS-SA for the cybersecurity use case.

Agent	Cap	Proba	Actions
A	Beg	0.50	BP, BF, DC, n
	Inter	0.35	BP, BF, DC, SP, PC, EKV, n
	Adv	0.15	BP, BF, DC, SP, PC, EKV, ZD, EB, n
D	HP	1	S_1, S_2, S_3, F, n

Table 1: Agent capacities and available actions for the cybersecurity case study.

The following details the interpretation of the actions listed in Table 1 and depicted in Figure 1.

- **Attacker actions:**
  - **BP**: Basic probe (network scan or service fingerprinting).
  - **BF**: Brute-force authentication attempt.
  - **DC**: Interaction with a decoy asset (click or download).
  - **SP**: Spear-phishing.
  - **PC**: Privilege-escalation command.
  - **EKV**: Exploitation of a known vulnerability.
  - **EB**: Exploit bypass to evade honeypot detection.
  - **ZD**: Zero-day exploit.
  - **n**: No operation.

- **Defender actions:**
  - $\mathbf{S_1}$: Spear-phishing decoy.
  - $\mathbf{S_2}$: Fake login page with a known CVE.
  - $\mathbf{S_3}$: Encrypted-channel decoy.
  - **F**: Fake long-running exploit success.
  - **n**: No action.

7 CONCLUSION

This article introduces PATL-SA as an extension of ATL-SA (Ballot et al., 2025b) with an operator to reason about the posterior probability that agents have some capacity after observing a history. We study the model-checking complexity and prove its NEXPTIME-completeness for formulas of the form $\langle Y \rangle^{\bowtie p} \psi$ without nested strategy formulas. For general formulas, it is in Δ_2^{E}. These complexity results are consistent with those of ATL-SA while we gain in expressibility. We showcase PATL-SA's applicability in a cybersecurity example where we verify attacker attribution objectives for an adaptive honeypot. While demonstrated in a cyber-defense scenario, PATL-SA offers a general tool for reasoning about agents whose abilities and intentions unfold probabilistically over time.

Future works include extending the model to imperfect information structures. Strategic logics are notoriously undecidable in general in this context (Dima and Ţiplea, 2011). However, limiting agents memory or imperfect information can help retrieving decidability (Catta et al., 2025; Belardinelli et al., 2023; Belardinelli et al., 2022b). Another extension would consider probability on transitions as well as capacities, as in (Chen and Lu, 2007). Then, we could see the impact of stochastic strategies in this context. Finally, we plan to implement this logic within a model checker such as `VITAMIN`(Ferrando and Malvone, 2025a; Ferrando and Malvone, 2025b), which has already been successfully applied to a wide range of strategic reasoning logics, including natural strategic logics (Jamroga et al., 2019; Belardinelli et al., 2022a) and obstruction logics (Catta et al., 2023; Catta et al., 2024b).

REFERENCES

Ågotnes, T. (2006). Action and knowledge in alternating-time temporal logic. *Synth.*, 149(2):375–407.

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

Ballot, G., Malvone, V., Leneutre, J., and Laarouchi, Y. (2024). Strategic reasoning under capacity-constrained agents. In *Proceedings of AAMAS 2024*, Auckland, New Zealand.

Ballot, G., Malvone, V., Leneutre, J., and Ma, J. (2025a). Strategic reasoning with capacity-constrained agents and imperfect information. *Proceedings of ECAI 2025*.

Ballot, G., Malvone, V., Leneutre, J., Ma, J., and Leslous, M. (2025b). Alternating-time temporal logic with stochastic abilities. In *Proceedings of AAMAS 2025*, pages 214–222.

Belardinelli, F., Ferrando, A., and Malvone, V. (2023). An abstraction-refinement framework for verifying strategic properties in multi-agent systems with imperfect information. *Artif. Intell.*, 316:103847.

Belardinelli, F., Jamroga, W., Malvone, V., Mittelmann, M., Murano, A., and Perrussel, L. (2022a). Reasoning about human-friendly strategies in repeated keyword auctions. In Faliszewski, P., Mascardi, V., Pelachaud, C., and Taylor, M. E., editors, *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022, Auckland, New Zealand, May 9-13, 2022*, pages 62–71. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS).

Belardinelli, F., Jamroga, W., Mittelmann, M., and Murano, A. (2024). Verification of stochastic multi-agent systems with forgetful strategies. In *AAMAS 2024*, pages 160–169.

Belardinelli, F., Lomuscio, A., Malvone, V., and Yu, E. (2022b). Approximating perfect recall when model checking strategic abilities: Theory and applications. *J. Artif. Intell. Res.*, 73:897–932.

Belardinelli, F., Lomuscio, A., Murano, A., and Rubin, S. (2017). Verification of multi-agent systems with imperfect information and public actions. In *AAMAS 2017*, pages 1268–1276.

Belardinelli, F., Lomuscio, A., Murano, A., and Rubin, S. (2020). Verification of multi-agent systems with public actions against strategy logic. *Artificial Intelligence*, 285:103302.

Berthon, R., Katoen, J.-P., Mittelmann, M., and Murano, A. (2024). Natural strategic ability in stochastic multi-agent systems. In *AAAI 2024*, pages 17308–17316.

Brihaye, T., Bruyere, V., and Raskin, J.-F. (2005). On optimal timed strategies. In *FORMATS 2005*, pages 49–64.

Catta, D., Ferrando, A., and Malvone, V. (2024a). Resource action-based bounded ATL: A new logic for MAS to express a cost over the actions. In Arisaka, R., Sánchez-Anguix, V., Stein, S., Aydogan, R., van der Torre, L., and Ito, T., editors, *PRIMA 2024: Principles and Practice of Multi-Agent Systems - 25th International Conference, Kyoto, Japan, November 18-24, 2024, Proceedings*, volume 15395 of *Lecture Notes in Computer Science*, pages 206–223. Springer.

Catta, D., Ferrando, A., and Malvone, V. (2025). Reasoning about decidability of strategic logics with imperfect information and perfect recall strategies. *J. Artif. Intell. Res.*, 82:777–817.

Catta, D., Leneutre, J., and Malvone, V. (2023). Obstruction logic: A strategic temporal logic to reason about dynamic game models. In Gal, K., Nowé, A., Nalepa, G. J., Fairstein, R., and Radulescu, R., editors, *ECAI 2023 - 26th European Conference on Artificial Intelligence, September 30 - October 4, 2023, Kraków, Poland - Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023)*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 365–372. IOS Press.

Catta, D., Leneutre, J., Malvone, V., and Murano, A. (2024b). Obstruction alternating-time temporal logic: A strategic logic to reason about dynamic models. In Dastani, M., Sichman, J. S., Alechina, N., and Dignum, V., editors, *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024, Auckland, New Zealand, May 6-10, 2024*, pages 271–280. International Foundation for Autonomous Agents and Multiagent Systems / ACM.

Chen, T. and Lu, J. (2007). Probabilistic alternating-time temporal logic and model checking algorithm. In *FSKD 2007*, pages 35–39.

David, A., Jensen, P. G., Larsen, K. G., Legay, A., Lime, D., Sø rensen, M., and Taankvist, J. (2014). On time with minimal expected cost! In *ATVA 2014*, pages 129–145.

Dima, C. and Ţiplea, F. L. (2011). Model-checking atl under imperfect information and perfect recall is undecidable. *arXiv:1102.4225*.

Ferrando, A., Luongo, G., Malvone, V., and Murano, A. (2024). Theory and practice of quantitative ATL. In Arisaka, R., Sánchez-Anguix, V., Stein, S., Aydogan, R., van der Torre, L., and Ito, T., editors, *PRIMA 2024: Principles and Practice of Multi-Agent Systems - 25th International Conference, Kyoto, Japan, November 18-24, 2024, Proceedings*, volume 15395 of *Lecture Notes in Computer Science*, pages 231–247. Springer.

Ferrando, A. and Malvone, V. (2025a). VITAMIN: A compositional framework for model checking of multiagent systems. In Rocha, A. P., Steels, L., and van den Herik, H. J., editors, *Proceedings of the 17th International Conference on Agents and Artificial Intelligence, ICAART 2025 - Volume 1, Porto, Portugal, February 23-25, 2025*, pages 648–655. SCITEPRESS.

Ferrando, A. and Malvone, V. (2025b). VITAMIN: verification of A multi agent system. In Das, S., Nowé, A., and Vorobeychik, Y., editors, *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025, Detroit, MI, USA, May 19-23, 2025*, pages 3023–3025. International Foundation for Autonomous Agents and Multiagent Systems / ACM.

Herzig, A., Lorini, E., and Walther, D. (2013). Reasoning about actions meets strategic logics. In *LORI 2013*, pages 162–175.

Huang, X., Su, K., and Zhang, C. (2012). Probabilistic atl of incomplete information and perfect recall. In *AAAI 2012*, pages 765–771.

Iqbal, S. and Pearson, J. (2017). A goal-based movement model for continuous multi-agent tasks. arXiv:1702.07319.

Jamroga, W., Malvone, V., and Murano, A. (2019). Natural strategic ability. *Artif. Intell.*, 277.

Jamroga, W. and van der Hoek, W. (2004). Agents that know how to play. *Fundamenta Informaticae*, 63:185–219.

Mittelmann, M., Maubert, B., Murano, A., and Perrussel, L. (2023). Formal verification of bayesian mechanisms. In *AAAI 2023*, pages 11621–11629.

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

Nguyen, H. N. and Rakib, A. (2019). Probabilistic resource-bounded atl. In *AAMAS 2019*, pages 2141–2143.

Schnoor, H. (2010). Strategic planning for probabilistic games with incomplete information. In *AAMAS 2010*, pages 1057–1064.

Schobbens, P.-Y. (2004). Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85:82–93.

van der Hoek, W. and Wooldridge, M. (2002). Tractable multiagent planning for epistemic goals. In *AAMAS 2002*, pages 1167–1174.

Walther, D., van der Hoek, W., and Wooldridge, M. (2007). Alternating-time temporal logic with explicit strategies. In *TARK 2007*, pages 269–278.

Ågotnes, T., Goranko, V., and Jamroga, W. (2007). Alternating-time temporal logics with irrevocable strategies. In *Proceedings of the 11th conference on Theoretical aspects of rationality and knowledge*, TARK '07, pages 15–24, New York, NY, USA. Association for Computing Machinery.