Development of Natural Strategies in Strategic Logics

Marco Aruta[1], Vadim Malvone[2] and Aniello Murano[1]

[1]Università degli Studi di Napoli Federico II, Italy
[2]Télécom Paris, France

Abstract

In the last two decades, both *Alternating-time Temporal Logic* (*ATL*) and *Strategy Logic (SL)* has been proved to be very useful in modeling strategic reasoning for *Multi-Agent Systems* (*MAS*). However, these logics struggle to capture the bounded rationality inherent in human decision-making processes. To overcome these limitations, *Natural Alternating-time Temporal Logic* (*NatATL*) and *Natural Strategy Logic (NatSL)* have been recently introduced. As respectively extensions of *ATL* and *SL*, these natural variants incorporate bounded memory constraints into agents' strategies, which allows to resemble human cognitive limitations. In this paper, we discuss a novel verification implementation for *NatATL* and *NatSL* specifications, both for memoryless strategies and strategies with recall. This research project aims to transform theoretical advancements into a practical verification framework, enabling comprehensive analysis and validation of strategic reasoning in complex multi-agent environments. Our novel tool paves the way for applications in areas such as explainable AI and human-in-the-loop systems, highlighting both *NatATL* and *NatSL* substantial potential.

Keywords

Multi-Agent Systems, Natural Strategies, Alternating-time Temporal Logic, Strategy Logic

1. Introduction

Multi-Agent Systems (MAS) have gained significant attention in recent years due to their ability to model and analyze complex systems composed of multiple interacting agents [2, 3]. These agents can be either human or artificial, and their interactions can range from cooperative to adversarial. Strategic reasoning plays a crucial role in understanding and predicting agent behavior and in designing systems that exhibit desired properties [4].

Over the past twenty years, formal verification works on *MAS* have flourished, with practical applications developed by both theorists and practitioners [5]. One of the most successful logics for representing agent conduct is *Alternating-time Temporal Logic* (*ATL*) [6], which uses a strategic operator $\langle\langle A\rangle\rangle\phi$ to indicate that a coalition of agents A has a strategy capable of enforcing ϕ regardless of the actions of all other agents.

Recently, *NatATL* [7] has been introduced as a logic to better capture how humans naturally strategize. *NatATL* is a variant of ATL that updates the strategic operator $\langle\langle A\rangle\rangle\phi$ with a bounded version $\langle\langle A\rangle\rangle^{\leq k}\phi$, where $k \in \mathbb{N}$ denotes the complexity bound. *NatATL* has the potential to be useful in various AI applications, such as human-in-the-loop systems and explainable AI. Despite the potential of *NatATL*, it has never been implemented in a tool.

Building on the success of *NatATL*, a more expressive logic called *Natural Strategy Logic (NatSL)* [8] has been proposed. *NatSL* extends traditional *Strategy Logic (SL)* by allowing the explicit quantification of strategies for both existential and universal agents. This logic enables the verification of both cooperative and adversarial strategies, while incorporating bounded natural strategies, which reflect more realistic constraints such as memory limitations [9]. As well as is intended in *NatATL* and *NatSL*, strategies are structured as lists of condition-action rules, mimicking the decision-making process of agents with limited resources. This extension allows for reasoning about scenarios where agents adopt human-like, simple strategies, striking a balance between complexity and expressive power. As a result,

AI4CC-IPS-RCRA-SPIRIT 2024: International Workshop on Artificial Intelligence for Climate Change, Italian Workshop on Planning and Scheduling, RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy. November 25-28th, 2024, Bolzano, Italy [1].
marco.aruta@unina.it (M. Aruta); vadim.malvone@telecom-paris.fr (V. Malvone); aniello.murano@unina.it (A. Murano)

NatSL is particularly well-suited for verifying strategic behaviors in complex multi-agent systems where both memory and computational resources are constrained.

So, both *NatATL* and *NatSL* aim to provide frameworks that better capture human-like decision-making and strategy formulation, making them particularly relevant for real-world applications such as human-centered AI and autonomous systems. These logics are powerful tools in the ongoing effort to bridge the gap between formal strategic reasoning and practical applications in multi-agent environments.

This paper discusses an ongoing work on practical verification design for *NatATL* and *NatSL* logics, both in the context of agents' strategies with or without recall. Our NatATL verification segment has already been successfully integrated into *VITAMIN* [10], an open-source model checker designed for verifying MAS that supports a variety of specifications, including Alternating-time Temporal Logic (ATL) [6], ATL with Fuzzy functions (ATLF) [11], Resource-Bounded ATL (RB-ATL) [12, 13], Resource Action-based Bounded ATL [14], Capacity ATL (CapATL) [15], Obstruction Logic (OL) [16], and Obstruction ATL (OATL) [17].

The implementation of *NatATL* model checking includes the following features: (i) *Strategies Generation*: The first step involves generating all strategies for each coalition whose complexity does not exceed a fixed value; (ii) *Model Pruning*: This phase involves projecting the strategies generated in the previous phase to simplify the model; (iii) *Model Checking CTL*: Since the pruning phase has made the problem independent from the coalition due to the projection of the strategy coalition, it is now possible to use CTL model checking. This approach allows the synthesis of optimal strategies that lead to a solution.

In addition to *NatATL*, we discuss novel approaches for the implementation of model checking in *NatSL*. The key innovation of *NatSL* lies in its ability to verify strategies that involve both existential and universal quantifiers over agents' behaviors, accounting for both cooperation and adversarial settings. Our implementation supports two distinct approaches: (i) *Alternated Strategy Generation*: In this approach, we first generate natural strategies for existential agents. If no solution is found, we proceed by generating all possible strategies for universal agents, iterating between the two until a solution is reached. This approach ensures a thorough exploration of the strategic space but may introduce computational overhead. This approach significantly reduces space computational complexity. (ii) *Sequential Strategy Generation*: This approach involves first generating all strategies for existential agents. Only if no solution is found we generate strategies for universal agents in a single step, allowing early termination when no further solutions are possible. This method significantly reduces time computational complexity by minimizing redundant strategy generation and verification steps. These approaches extend the capabilities of *NatSL*, enabling the verification of more intricate agent interactions and providing a robust framework for handling both cooperative and adversarial behaviors in multi-agent systems.

Outline. The rest of the paper is organized as follows. Section 2 recalls the main notions of natural strategies, NatATL and NatSL. Section 3 presents the novel engineering contribution of our tool. Finally, we conclude in Section 4.

2. Background

Concurrent Game Structure. Formally, a *CGS* consists of a tuple $S = \langle Agt, Q, AP, \pi, d, \delta \rangle$ with the following components: (i) $Agt \geq 1$ is a natural number indicating the number of agents; (ii) Q is a finite set of states; (iii) AP is finite set of propositions; (iv) π is a labelling function such that for each state q, $\pi(q) \subseteq AP$ is a set of true propositions at q; (v) $d(a, q)$ indicates for each agent a and state q, the set of actions available in the state q for a. A *move vector* at q consist of a tuple $\langle j_1, \ldots, j_{Agt} \rangle$ such that $j_a \in d(a, q)$ for each agent a; (vi) δ is a transition function that for each state q and each move vector, returns a state that results from q if each agent a chooses the move in the vector.

Natural Strategies. Given a CGS S and a set of agents, a general strategy for an agent in a *MAS* is a function that determines a move for each agent based on every finite prefix of a computation. For a specific state q, a coalition A, and a set of strategies for A, the *outcomes* from q are all possible future sequences of states from q that the coalition A can enforce by following their strategies. *Natural strategies* are expressed using a rule-based system, where each rule comprises a condition and an action. A memoryless strategy (*nr*-strategy) differs from a strategy with recall (*nR*-strategy) in that the former defines its conditions using Boolean formulas over AP—considering current state information only. In contrast, the latter describes its conditions through regular expressions over AP, incorporating the history of states (i.e., the sequence of states that has occurred so far). The complexity of natural strategies is gauged by the overall size of the conditions' representation.

NatATL. Natural Alternating-time Temporal Logic (*NatATL*)[9] is a logic for natural strategic ability that enhances *ATL* by integrating human reasoning constraints, thereby expanding its applicability and effectiveness. *NatATL* effectively addresses usability issues related to the functional requirements of reactive systems. It considers situations where multiple agents, each with their own goals and capabilities, can take actions concurrently. *NatATL* syntax derives from substituting the modality $\langle\langle A\rangle\rangle$ in *ATL* with the bounded strategic modality $\langle\langle A\rangle\rangle^{\leq k}$. Intuitively, $\langle\langle A\rangle\rangle^{\leq k}\gamma$ says that a coalition of agents ($A \subseteq Agt$) has a collective natural strategy of size at most k to enforce property γ. Similar to *ATL*, *NatATL* formulas predicate over a set of atomic propositions AP and uses classical temporal operators. Therefore, the *NatATL* language can be delineated by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle^{\leq k} X\varphi \mid \langle\langle A\rangle\rangle^{\leq k} \varphi\, U \varphi \mid \langle\langle A\rangle\rangle^{\leq k} \varphi\, R \varphi$$

where p is an atomic proposition, k is the complexity bound, φ is a composed formula and, X, U, and R are temporal operators and stand for "next", "until", and "release", respectively. For a coalition A, s_A denotes a collective natural strategy. The complexity of a natural collective strategy is the sum of the individual strategies' complexities. For further insights into the semantics and theoretical aspects, refer to [9].

NatSL. The NatSL extension adds a new layer of strategic reasoning by differentiating between existential and universal agents in the logical formula [8]. In essence, the formula now consists of two types of quantifiers: existential quantifiers (denoted by $\exists$) and universal quantifiers (denoted by $\forall$). Each quantifier is associated with a set of agents and their corresponding strategies. The goal of NatSL verification is to ensure that the existential agents can find a winning strategy, taking into account that the universal agents may counteract with their strategies. The syntax of Natural Strategy Logic (NatSL) extends standard Strategy Logic by incorporating natural strategies and their complexities. The formula syntax can be defined as follows:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists^{\leq k} s_a \varphi \mid \forall^{\leq k} s_a \varphi \mid (a, s_a)\varphi \mid X\varphi \mid \varphi\, U \varphi \mid \varphi\, R \varphi$$

The intuitive reading of the operators is as follows: $\exists^{\leq k} s_a \varphi$ means that there exists a strategy with complexity less or equal than k for agent a such that φ holds; $(a, s_a)\varphi$ means that when strategy s_a is assigned to agent a, φ holds; X, U, and R are the usual temporal operators.

3. Current Implementation

The implementation chapter of our work focuses on the development of a module designed to enhance the verification capabilities of both NatATL and NatSL formal logics within MAS, considering both cases with and without recall. This module operates within the VITAMIN framework of an existing verification tool and integrates various components to process and evaluate models representing systems of agents, states, and actions. The module's primary function is to verify strategic properties expressed in a logical formalism by transforming the input model and applying standard model checking techniques.

3.1. NatATL System Architecture

The architecture of the tool is built around three core components: *strategy generation, model transformation, and model checking*. These components interact in a pipeline, ensuring that the input model is progressively refined through each stage of the process. The input consists of a representation of the system as a graph, where nodes correspond to states and edges represent possible transitions based on actions taken by agents. Additionally, a logical formula that encapsulates the desired property is provided for verification.

The tool begins by generating possible strategies for the agents involved. These strategies determine how agents may act collectively or individually in the given system. Once a strategy is selected, the model is pruned—this means removing transitions that are not compatible with the chosen strategy, effectively simplifying the model for further analysis. This pruned model is then subjected to the verification of the logical formula using model checking algorithms. A comprehensive overview of the entire NatATL verification process is shown in Fig. 1.

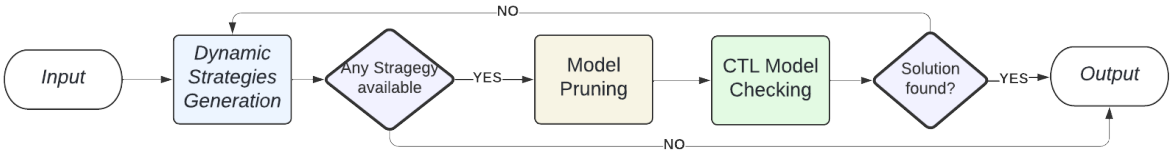

Figure 1: *NatATL* verification process

Strategy Generation. The generation of strategies is essential for exploring the behavior of agents within the system. This process is highly complex due to the number of possible strategies that agents can adopt, depending on their actions and the states they occupy. To manage this complexity, the system incrementally generates strategies, prioritizing those with lower complexity before expanding to more involved strategies if necessary. This ensures computational efficiency and streamlines the verification process by focusing on strategies that are more likely to produce a result.

Model Transformation. Once a strategy is selected, the system undergoes a transformation process where the original model is pruned. This involves removing transitions that do not align with the current strategy, which simplifies the system's structure and reduces the number of states and transitions that need to be considered during verification. This transformation can be performed for different types of strategies, including memoryless natural strategies, where actions depend solely on the current state, and natural strategies with recall, where actions can depend on the history of states. For strategies with recall, the system is transformed into a tree structure, which better represents the history of states and transitions. This allows the model to capture more complex behaviors over time and enables more accurate verification of strategies that rely on past information.

Model Checking. The final step involves verifying whether the pruned model satisfies the given logical formula. The system employs a tree-based structure to evaluate the formula, breaking it down into smaller components that can be checked independently. Standard CTL model checking algorithms are used to determine whether the formula holds in the pruned model. If the formula is satisfied, the system identifies the set of states that meet the formula's requirements. If not, the system iterates through additional strategies, repeating the pruning and verification process until a solution is found or all strategies are exhausted.

Computational Considerations. The complexity of the overall process is governed by both the strategy generation and the model checking phases. The strategy generation involves a combinatorial explosion of possible actions, but optimizations are applied to mitigate this issue by focusing on simpler

strategies first. The model transformation and pruning are computationally efficient, operating within the constraints of the system's size, while the model checking algorithm leverages well-established techniques with polynomial complexity. This ensures that the system can handle reasonably large models without excessive computational overhead.

3.2. NatSL System Architecture

The implementation of NatSL is currently in its early stages, particularly in comparison to the more mature implementation of *NatATL*. At present, the approach focuses on a limited case where strategies target only a single goal, and quantifiers over existential and universal agents are not alternated. This simplification allows us to concentrate on the foundational elements of NatSL, while ensuring that the core verification process is feasible.

In its current form, the implementation primarily handles scenarios where the existential agents aim to achieve a specific goal, while the universal agents act in opposition, trying to prevent the goal from being reached. This non-alternating quantifier setup simplifies the strategy generation process, as it removes the need to consider more complex, nested strategies that arise when existential and universal quantifiers are alternated.

Despite these limitations, the NatSL implementation is designed to provide a foundation for future extensions. We plan to gradually expand its scope to handle more intricate agent interactions and alternating quantifiers. As a first step, we introduce two distinct approaches for strategy generation and verification, each offering different trade-offs in terms of computational efficiency.

Approach 1: Alternated Strategy Generation. The first approach to NatSL verification proceeds in an alternated manner:

1. *Existential Strategy Generation*: The system first generates a natural strategy for the existential agents, similar to how it operates in NatATL. If the generated strategy passes model checking (i.e., the model satisfies the logical formula after pruning the model based on the strategy), the process terminates, and the system returns true.

2. *Universal Strategy Generation*: If no winning strategy is found for the existential agents, the system pauses further existential strategy generation and shifts to generating all possible strategies for the universal agents. The universal agents' strategies are then applied, and the system checks if any of these strategies lead to a false result after pruning and model checking. If any universal strategy results in false, the process returns to generating the next existential strategy.

3. *Iterative Process*: This iterative process continues, alternating between generating existential strategies and verifying them against all possible universal strategies. The process stops once either a valid existential strategy is found, or all strategies have been exhausted.

This approach, while simple, is less efficient in terms of computational time complexity because it requires generating and verifying all universal strategies after each existential strategy. The back-and-forth nature of this process increases the computational load, especially as the number of agents and strategies grows. The diagram below (Fig. 2) will visualize how the system alternates between generating strategies for existential and universal agents in this approach.

Approach 2: Sequential Strategy Generation. The second approach improves time computational efficiency by restructuring the strategy generation process. The key idea is to first complete the entire NatATL verification before moving to universal strategy generation:

1. *Complete Existential Strategy Generation*: The system generates all existential strategies first, performing pruning and model checking for each strategy. By the way, if a winning existential

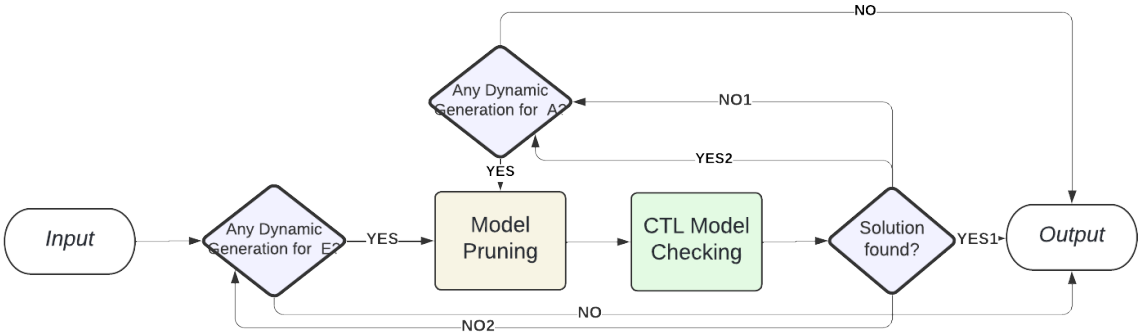

Figure 2: *NatSL* verification process

strategy is found, the whole process terminates with true, stopping further analysis, as the rest of existential strategies are not generated.

2. *Universal Strategy Generation*: If no existential strategy leads to a solution, the system generates all strategies for the universal agents in a single step. The universal strategies are then applied to the pruned models. If any universal strategy fails (i.e., results in false after pruning and model checking), the process terminates early with false, analyzing the next pruned tree - if there is any, otherwise the process is finished.

3. Exit Condition: The key time-efficiency gain in this approach is the early exit condition in the analysis of existential strategies. In fact, as soon as a solution is found in the existential analysis, the algorithm terminates without generating further universal strategies. This drastically reduces the computational time overhead compared to the first approach.

To better illustrate this approach, a diagram is included to the reader below (Fig. 3) showing how this technique streamlines the first approach process by separating its strategy generation phases.

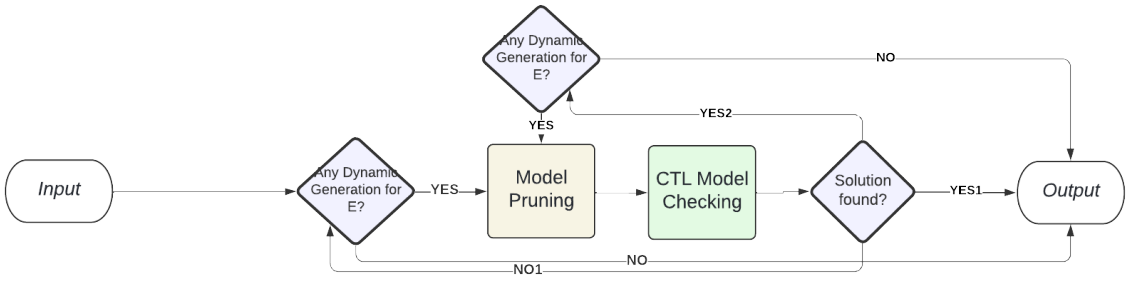

Figure 3: *NatSL* verification process

Computational Considerations. From an engineering standpoint, Approach 2 is more timely-efficient because it requires fewer strategy generations and model checks. By generating all existential strategies upfront and only generating universal strategies if no solution is found for the existential ones, this approach minimizes unnecessary recomputation and reduces the overall time complexity of the verification process. However, during the generation of existential strategies, it must store each pruned model to use it later when universal strategies are invoked. This results in higher memory usage, making this approach less suitable in terms of memory-space compared to the first approach, which, although slower, is more space-efficient (as each pruned model is used on the fly).

4. Conclusions

In this paper, we discussed a practical implementation for *NatATL* and *NatSL*, along with methods for efficiently generating natural strategies, both for memoryless strategies and strategies with recall. Our tool enhances strategic reasoning in *MAS* by addressing the limitations of traditional *ATL* and *SL* in capturing human decision-making processes. For *NatATL*, we have successfully integrated the verification module into the *VITAMIN* model checker, optimizing both space and computational complexity by employing a strategy generation process that progresses from minimal complexity and only increases when necessary. This segmentation significantly improves performance, particularly in complex systems where strategies with recall are required. Similarly, our proposed approaches for *NatSL* focus on optimizing both memory usage and computational time. The two approaches, *Alternated Strategy Generation* and *Sequential Strategy Generation*, offer different trade-offs between space and time efficiency. The former ensures comprehensive exploration of strategic possibilities, while the latter minimizes overhead by reducing redundant strategy generation. These techniques are designed to handle both cooperative and adversarial agent behaviors, paving the way for broader applications of natural strategies in practical settings.

While our implementation of *NatATL* represents a fully integrated solution in an existing model checker, the development of *NatSL* is still a work in progress. Several steps remain before a complete integration of *NatSL* into a verification tool can be achieved. Future work will focus on refining the current approaches to improve both time and memory efficiency and ensuring compatibility with large-scale multi-agent systems. Additionally, we aim to extend the functionality of the tool to support the entire *NatSL* logic and ensuring more complex agent interactions. By continuing to optimize *NatSL* and enhancing its integration into practical verification tools, we hope to advance the capabilities of strategic reasoning in *MAS*, with potential applications spanning areas such as AI-driven decision-making and human-centered autonomous systems.

References

[1] D. Aineto, R. De Benedictis, M. Maratea, M. Mittelmann, G. Monaco, E. Scala, L. Serafini, I. Serina, F. Spegni, E. Tosello, A. Umbrico, M. Vallati (Eds.), Proceedings of the International Workshop on Artificial Intelligence for Climate Change, the Italian workshop on Planning and Scheduling, the RCRA Workshop on Experimental evaluation of algorithms for solving problems with combinatorial explosion, and the Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy (AI4CC-IPS-RCRA-SPIRIT 2024), co-located with 23rd International Conference of the Italian Association for Artificial Intelligence (AIxIA 2024), CEUR Workshop Proceedings, CEUR-WS.org, 2024.

[2] M. Wooldridge, An Introduction to Multiagent Systems, John Wiley & Sons, 2009.

[3] W. Jamroga, Logical Methods for Specification and Verification of Multi-Agent Systems, Institute of Computer Science, Polish Academy of Sciences, 2015.

[4] F. Mogavero, A. Murano, G. Perelli, M. Y. Vardi, Reasoning about strategies: On the model-checking problem, ACM Trans. Comput. Log. 15 (2014) 34:1–34:47.

[5] A. Lomuscio, H. Qu, F. Raimondi, Mcmas: An open-source model checker for the verification of multi-agent systems, Int. J. Softw. Tools Technol. Transf. 19 (2017) 9–30.

[6] O. K. Rajeev Alur, Thomas A. Henzinger, Alternating-time temporal logic, J. ACM 49 (2002) 672–713.

[7] W. Jamroga, V. Malvone, A. Murano, Reasoning about natural strategic ability, in: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, 2017, pp. 714–722.

[8] F. Belardinelli, W. Jamroga, V. Malvone, M. Mittelmann, A. Murano, L. Perrussel, Reasoning about human-friendly strategies in repeated keyword auctions, in: P. Faliszewski, V. Mascardi, C. Pelachaud, M. E. Taylor (Eds.), 21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022, Auckland, New Zealand, May 9-13, 2022, International Foun-

dation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2022, pp. 62–71. URL: https://www.ifaamas.org/Proceedings/aamas2022/pdfs/p62.pdf. doi:10.5555/3535850.3535859.

[9] W. Jamroga, V. Malvone, A. Murano, Natural strategic ability under imperfect information, in: 18th International Conference on Autonomous Agents and Multiagent Systems AAMAS 2019, IFAAMAS, 2019.

[10] A. Ferrando, V. Malvone, Vitamin: A compositional framework for model checking of multi-agent systems, arXiv preprint arXiv:2403.02170 (2024).

[11] A. Ferrando, G. Luongo, V. Malvone, A. Murano, Theory and practice of quantitative atl, in: R. Arisaka, V. S. Anguix, S. Stein, R. Aydogan, L. van der Torre, T. Ito (Eds.), PRIMA 2024: Principles and Practice of Multi-Agent Systems - 25th International Conference, Kyoto, Japan, November 18-24, 2024, Proceedings, volume to appear of *Lecture Notes in Computer Science*, Springer, 2024.

[12] H. N. Nguyen, N. Alechina, B. Logan, A. Rakib, Alternating-time temporal logic with resource bounds, J. Log. Comput. 28 (2018) 631–663.

[13] A. Ferrando, V. Malvone, Hands-on VITAMIN: A compositional tool for model checking of multi-agent systems, in: M. Alderighi, M. Baldoni, C. Baroglio, R. Micalizio, S. Tedeschi (Eds.), Proceedings of the 25th Workshop "From Objects to Agents", Bard (Aosta), Italy, July 8-10, 2024, volume 3735 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 148–160.

[14] D. Catta, A. Ferrando, V. Malvone, Resource action-based bounded atl: a new logic for mas to express a cost over the actions, in: R. Arisaka, V. S. Anguix, S. Stein, R. Aydogan, L. van der Torre, T. Ito (Eds.), PRIMA 2024: Principles and Practice of Multi-Agent Systems - 25th International Conference, Kyoto, Japan, November 18-24, 2024, Proceedings, volume to appear of *Lecture Notes in Computer Science*, Springer, 2024.

[15] G. Ballot, V. Malvone, J. Leneutre, Y. Laarouchi, Strategic reasoning under capacity-constrained agents, in: M. Dastani, J. S. Sichman, N. Alechina, V. Dignum (Eds.), Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024, Auckland, New Zealand, May 6-10, 2024, International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2024, pp. 123–131. URL: https://dl.acm.org/doi/10.5555/3635637.3662859. doi:10.5555/3635637.3662859.

[16] D. Catta, J. Leneutre, V. Malvone, Obstruction logic: A strategic temporal logic to reason about dynamic game models, in: K. Gal, A. Nowé, G. J. Nalepa, R. Fairstein, R. Radulescu (Eds.), 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*, IOS Press, 2023, pp. 365–372. URL: https://doi.org/10.3233/FAIA230292. doi:10.3233/FAIA230292.

[17] D. Catta, J. Leneutre, V. Malvone, A. Murano, Obstruction alternating-time temporal logic: A strategic logic to reason about dynamic models, in: M. Dastani, J. S. Sichman, N. Alechina, V. Dignum (Eds.), Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024, Auckland, New Zealand, May 6-10, 2024, International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2024, pp. 271–280. URL: https://dl.acm.org/doi/10.5555/3635637.3662875. doi:10.5555/3635637.3662875.