# Towards a Compositional and User-friendly tool for Multi-Agent Systems Verification

Angelo Ferrando[1], Vadim Malvone[2]

[1]University of Modena and Reggio Emilia, Italy
[2]Télécom Paris, France

**Abstract**

Verifying software and hardware systems presents considerable challenges due to their inherent complexity, often making thorough verification impractical. The transition from monolithic architectures to Multi-Agent Systems (MAS) exacerbates these issues, demanding more sophisticated verification tools. Existing solutions, such as MCMAS and STV, fall short in terms of modularity, flexibility, and usability. This paper highlights these limitations and suggests future directions for the development of verification tools for MAS. Then, it discusses VITAMIN (VerIficaTion of A MultI-ageNt system), a formal verification framework aimed at addressing these shortcomings by supporting diverse logics and model formalisms while also focusing on ease of use.

**Keywords**

Model Checking, Multi-Agent Systems, Verification Tools

## 1. Introduction

The correctness of systems is crucial in both hardware and software design, particularly for critical systems, where failure is unacceptable. Critical systems refer to those where malfunctions are not tolerable. The primary approaches for software verification include testing, simulation, and formal verification. The key limitation of testing and simulation is that while they can identify errors, they cannot guarantee their absence. To address this issue, *formal verification* proves to be highly effective. This approach offers a formalised methodology for modeling systems, specifying properties, and ensuring that a system meets a given specification.

In formal verification, specifications are typically based on temporal logics. These logics describe the sequence of events without explicitly mentioning time. Temporal logics are primarily categorised into linear-time and branching-time logics, each reflecting different conceptions of time. The most commonly used temporal logics include LTL (Linear-Time Temporal Logic) [2], CTL (Computation Tree Logic) [3], and its extension CTL* [4]. A significant advancement in the field of temporal logics has been the development of algorithmic methods for verifying properties of finite-state systems represented by Kripke structures [5]. Thus, the formal verification of a system modelled by a Kripke structure $M$ with respect to a temporal logic specification $\varphi$ can be restated as: "Is $M$ a model of $\varphi$?". This encapsulates the concept of *model checking* (MC), a term introduced by Clarke and Emerson in [3].

Two primary approaches are used to carry out model checking in practice. The first approach involves using classical ad-hoc algorithms. For instance, PSpace-Complete recursive algorithms are employed to solve model checking problems for LTL. Similarly, a linear algorithm has been developed for CTL. The second approach involves a systematic use of the automata-theoretic method for infinite objects. Specifically, this involves translating a temporal logic formula $\varphi$ into an automaton. Consequently, the model checking problem is reduced to solving the emptiness problem of the intersection between the automaton representing the system and the automaton for the complement of the property.

*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].*
angelo.ferrando@unimore.it (A. Ferrando); vadim.malvone@telecom-paris.fr (V. Malvone)
https://angeloferrando.github.io/ (A. Ferrando); https://vadimmalvone.github.io/ (V. Malvone)
0000-0002-8711-4670 (A. Ferrando); 0000-0001-6138-4229 (V. Malvone)

Initially, model checking was applied primarily to closed systems, where behaviour is entirely determined by internal states. However, these techniques have limited practical utility today, as most systems are open and interact continuously with other systems. To address this challenge, model checking was extended to Multi-Agent Systems (MAS).

In the design and verification of MAS, temporal logics have recently become central to strategic reasoning [6, 7, 8, 9, 10, 11]. In particular, classical temporal logics have been extended to accommodate properties specific to MAS. One of the most significant advancements in this area is *Alternating-Time Temporal Logic* (ATL), introduced by Alur, Henzinger, and Kupferman [6]. This logic enables reasoning about agents' strategies with the satisfaction of temporal goals as the payoff criterion. More formally, ATL is a generalisation of CTL in which the existential E and universal A *path quantifiers* are replaced by *strategic modalities* of the form $\langle\!\langle \Gamma \rangle\!\rangle$ and $[\![\Gamma]\!]$, where $\Gamma$ represents a set of *agents*. Despite its expressiveness, ATL has a significant limitation: strategies are only implicitly handled in the semantics of its modalities. This restriction makes the logic less suitable for formalising several important solution concepts, such as the *Nash Equilibrium*. These considerations led to the development of *Strategy Logic* (SL) [12, 9], a more advanced formalism for strategic reasoning. A key feature of this logic is that it treats strategies as *first-order objects*, which can be quantified using existential $\exists x$ and universal $\forall x$ quantifiers. These quantifiers can be interpreted as *"there exists a strategy x"* and *"for all strategies x"*, respectively. Notably, in SL [9], a strategy is defined as a generic conditional plan that specifies an action at each step of the MAS. In more detail, there are two main types of strategies: memoryless and memoryful. In the case of memoryless strategies, agents select an action based solely on the current game state. Conversely, memoryful strategies involve agents making decisions based on the entire history of the game. Therefore, this plan is not intrinsically glued to a specific agent, but an explicit binding operator $(a, x)$ allows to link an agent $a$ to the strategy associated with a variable $x$. Unfortunately, the high expressivity of SL comes at a price. Indeed, it has been proved that the model-checking problem for SL becomes non-elementary complete and the satisfiability undecidable. To gain back elementariness, several fragments of SL have been considered. Among various approaches, Strategy Logic One-Goal focuses on SL formulas in a specific prenex normal form, which involves a single temporal goal at a time. Here, a goal is defined as a sequence of bindings followed by a temporal logic formula. It has been demonstrated that Strategy Logic One-Goal strictly subsumes ATL$^*$, and its model-checking problem is 2Exptime-Complete, the same complexity class as for ATL$^*$. Additionally, Strategy Logic with Simple-Goals [13] deals with SL formulas where strategic operators, binding operators, and temporal operators are combined. It has been shown that Strategy Logic with Simple-Goals strictly subsumes ATL, and its model-checking problem is P-Complete, similar to ATL.

To conclude this section, we focus on a crucial aspect in MAS: the visibility of agents. Specifically, we distinguish between *perfect* and *imperfect* information MAS [14]. In perfect information, every agent has complete knowledge of the MAS. However, in real-world scenarios, agents often operate without full access to all relevant information. In computer science, such situations arise, for example, when some system variables are internal or private and not visible to the external environment [15, 16]. In MAS models, imperfect information is typically represented by an indistinguishability relation over the MAS's states [15, 14, 17]. This means that, during the evolution of the MAS, some agents may not be able to precisely determine their current state but can only observe a set of indistinguishable states. As a result, these agents cannot base their strategies on the exact state of the MAS, which implies that they can only use the same move within indistinguishable sets, or that some perfect information strategies are no longer applicable. This characteristic significantly impacts the complexity of model checking. For instance, ATL becomes undecidable in the context of imperfect information and memoryful strategies [18]. To address this challenge, some research has focused on approximations to perfect information [19, 20, 21], developed concepts of bounded memory [22, 23], or provided hybrid techniques [24, 25].

## 2. Limitations of Current Tools for MAS Verification

As mentioned before, the verification of MAS presents significant challenges, surpassing the complexity of traditional monolithic systems. MAS are characterised by the autonomous behaviour of agents, their interactions, and the strategic reasoning required, making formal verification in these systems a demanding task. Two prominent tools used for this purpose are MCMAS [26] and STV [27]. Despite their widespread use, both tools have notable limitations, especially when applied outside academic settings. In this section, we explore these limitations, focusing on modularity, user accessibility, and maintainability.

### 2.1. MCMAS: A Research-Centric Tool with Limitations

MCMAS, the *Model Checker for Multi-Agent Systems* [26], has long been a foundational tool in MAS verification, particularly in academic research. Its early development as a proof-of-concept model checker made it a preferred tool for researchers. However, several limitations restrict its application beyond academia, particularly in industrial contexts.

**Lack of Modularity.** The most significant limitation of MCMAS is its lack of modularity. Its verification process is rigid and hard-coded, which makes it difficult to adapt or extend the tool for new logics and models. The absence of a modular architecture has resulted in a monolithic codebase that is hard to maintain and prone to bugs. Contributions from different research groups over time have led to inconsistencies, as the lack of a unified development approach has complicated the implementation of new features.

**Inadequate Documentation.** MCMAS also suffers from a lack of thorough documentation. External documentation is scarce, leaving new developers or users without clear guidance. The internal documentation, when available, is often insufficient and primarily useful only for those directly involved in its development. This documentation gap significantly hinders the tool's accessibility, making it difficult for new researchers to install, execute, or extend the tool without encountering problems—compounded by the need for external tools like Eclipse[1] to run it effectively.

**Narrow Research-Only Focus.** Designed primarily as a research tool, MCMAS's limitations are evident in its inability to support a wider variety of logics and models required for real-world MAS verification. The tool was not intended for practical, large-scale application beyond academia. This narrow focus limits MCMAS's scalability and adaptability for real-world verification tasks, which demand flexible architectures capable of addressing complex and diverse MAS scenarios.

### 2.2. STV: A Specialised Tool with Limited Extensibility

STV, *StraTegic Verifier* [27], another model checker for multi-agent systems, offers a graphical interface that sets it apart from MCMAS. However, while this interface makes STV more user-friendly, the tool faces its own set of limitations in terms of scalability, adaptability, and industry application.

**Limited Scope and Extensibility.** STV was specifically developed for addressing narrow theoretical research problems. Its design reflects this focus, limiting its ability to be extended by other researchers or applied in broader contexts. This lack of extensibility hampers STV's development into a more versatile tool that could support multiple logics and models necessary for comprehensive MAS verification.

[1] https://www.eclipse.org/

**Unsuitable for Industrial Use.** Another significant limitation of STV is its unsuitability for industrial use. The tool was not designed for environments where users may lack expertise in formal verification techniques. Industrial contexts require tools that can be operated by non-experts, yet STV remains inaccessible to this broader audience due to its reliance on specialised formal methods knowledge. As a result, STV fails to meet the practical needs of industry users, where ease of use and scalability are critical.

**Documentation and Usability Challenges.** Like MCMAS, STV also suffers from insufficient documentation, both for developers and end-users. Although it provides a graphical interface, its usability is still limited by the need for expertise in formal methods. The lack of clear documentation and tutorials exacerbates the challenge, making it difficult for those unfamiliar with formal verification techniques to use the tool effectively.

### 2.3. Broader Challenges in MAS Verification

The limitations observed in both MCMAS and STV reflect the broader challenges inherent in the verification of multi-agent systems. The transition from monolithic systems to MAS intensifies the verification complexities, as MAS involve not only traditional system behaviours but also intricate agent interactions and strategic reasoning. While tools like MCMAS and STV have advanced theoretical research, they fall short in meeting the practical requirements of scalable, modular, and user-friendly verification solutions.

**Modularity and Extensibility.** A key challenge in MAS verification is the need for modular tools that can accommodate new models and logics without compromising the integrity of existing modules. Both MCMAS and STV lack the modularity required for this extensibility. Future verification tools must be designed with modular architectures that allow for the addition of new features in a way that preserves the correctness and stability of the existing system.

**Accessibility for Non-Experts.** Another critical challenge is making MAS verification accessible to users who are not experts in formal methods. Neither MCMAS nor STV has been designed with ease of use in mind, as both tools require a deep understanding of formal verification techniques. For broader adoption, particularly in industrial settings, future tools must include user-friendly interfaces and comprehensive documentation, along with tutorials that guide non-expert users through the verification process.

## 3. Moving Beyond the Limitations

While MCMAS and STV have been crucial in advancing MAS verification research, their limitations—particularly the lack of modularity, extensibility, and accessibility—underscore the need for the next generation of MAS verification tools. Addressing these shortcomings is essential for creating tools that bridge the gap between academic research and industrial application.

To overcome the limitations of MCMAS and STV, the development of future MAS verification tools must focus on the following key areas:

- **Modularity:** A modular architecture is critical for enabling the independent development of new logics and models without affecting the existing system. This ensures that the tool remains adaptable and scalable over time.
- **User Accessibility:** Future tools must be designed with user-friendly interfaces and thorough documentation that can support both experts and non-experts in formal methods. This will help broaden the tool's usability in industrial contexts.

- **Industry Scalability:** Verification tools must be optimised for use in industrial settings, where users may lack formal verification expertise. This requires developing features that simplify the generation of formal models and properties, along with incorporating optimisation techniques to manage the execution complexity of large-scale MAS verification tasks.

By focusing on these areas, future MAS verification tools can address the weaknesses of existing solutions, unlocking new possibilities for both research and practical application in industry.

To open MAS verification to industrial contexts, we must eliminate the weaknesses of current model checkers. This involves creating a modular tool with a user-friendly interface and comprehensive documentation for future developers, all while ensuring the correctness of the core verification engine. Importantly, this modular system would allow for new developments to be treated as black boxes, guaranteeing that future extensions do not compromise the integrity of existing functionality. Finally, simplifying the execution complexity and incorporating optimisation techniques—many of which have already been introduced in our research—will be key to making MAS verification viable for large-scale, real-world applications.

**Towards VITAMIN.** Given the limitations of MCMAS and STV, it is evident that future MAS verification tools must prioritise modularity, user accessibility, and scalability to meet the diverse needs of both academia and industry. Recent research, such as that by [28, 29], outlines a promising path forward with a novel framework called VITAMIN, designed to enhance the modularity, extensibility, and user-friendliness of MAS verification frameworks. This approach emphasises a compositional model-checking architecture, enabling independent modules tailored to specific logics and models. Currently, VITAMIN supports a variety of specifications, including Alternating-time Temporal Logic (ATL) [30], ATL with Fuzzy functions (ATLF) [31], Natural ATL (NatATL) [32], Natural SL (NatSL) [33], Resource-Bounded ATL (RB-ATL) [34, 29], Resource Action-based Bounded ATL [35], Capacity ATL (CapATL) [36], Obstruction Logic (OL) [37], and Obstruction ATL (OATL) [38]. Additionally, VITAMIN's documentation is designed to support both developers and end-users, facilitating its use and extension. Adopting such a framework will allow the next generation of verification tools to not only address the rigidity of current solutions like MCMAS and STV but also integrate new advances in MAS verification without compromising the integrity of existing modules. This shift towards a more adaptable and scalable architecture has the potential to bridge the gap between academic research and real-world application, leading to more robust, industry-ready MAS verification tools.

## 4. Conclusions and Future Work

In this paper, we have explored the limitations of the current model checkers, MCMAS and STV, in the context of MAS verification. Both tools have made significant contributions to academic research, but their lack of modularity, limited extensibility, and insufficient documentation hinder their broader adoption, particularly in industrial contexts. MCMAS, while widely recognised for its theoretical contributions, suffers from a rigid, hard-coded architecture and documentation issues that prevent it from evolving into a versatile tool. STV, though offering a graphical interface, faces similar limitations in scope and usability, making it difficult to apply in environments that lack formal verification expertise.

To address these challenges, we pointed out to VITAMIN, a novel verification tool designed to overcome the weaknesses of MCMAS and STV. VITAMIN's modularity enables it to support multiple logics and models in a scalable manner, without compromising on ease of use or flexibility. By allowing independent modules for each formalism, VITAMIN ensures that future developments can be incorporated without affecting the correctness of existing components. This makes it suitable for both academic research and practical applications, including industrial contexts where ease of use is essential.

Future efforts will concentrate on expanding VITAMIN's capabilities in several critical areas. First, the framework will be extended to handle more complex case studies, ensuring its scalability for real-world MAS applications, including autonomous vehicles and security protocols. Second, feedback

from the broader MAS community will be solicited, particularly on the usability of VITAMIN in both academic and industrial contexts. This feedback will guide further improvements to the tool's interface, documentation, and modular architecture. Third, optimising the execution complexity of VITAMIN will be a critical area for future research. Incorporating advanced optimisation techniques, such as those explored in prior work, will help reduce the computational overhead of verifying large-scale systems. Further development on automated model and property generation techniques will be carried out, which will simplify the verification process for non-experts, making VITAMIN more accessible to a wider range of users.

By addressing these future directions, we believe that VITAMIN has the potential to become the next-generation MAS verification tool, bridging the gap between academic research and industrial applications, and providing a robust, scalable solution for formal verification in multi-agent 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] A. Pnueli, The Temporal Logic of Programs., in: Foundation of Computer Science'77, IEEE Computer Society, 1977, pp. 46–57.

[3] E. Clarke, E. Emerson, Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic., in: Logic of Programs'81, LNCS 131, Springer, 1981, pp. 52–71.

[4] E. Emerson, J. Halpern, "Sometimes" and "Not Never" Revisited: On Branching Versus Linear Time., Journal of the ACM 33 (1986) 151–178.

[5] S. Kripke, Semantical Considerations on Modal Logic., Acta Philosophica Fennica 16 (1963) 83–94.

[6] R. Alur, T. Henzinger, O. Kupferman, Alternating-Time Temporal Logic., Journal of the ACM 49 (2002) 672–713.

[7] W. Jamroga, W. van der Hoek, Agents that Know How to Play., Fundamenta Informaticae 63 (2004) 185–219.

[8] K. Chatterjee, T. Henzinger, N. Piterman, Strategy Logic., Information and Computation 208 (2010) 677–693.

[9] F. Mogavero, A. Murano, M. Vardi, Reasoning About Strategies., in: Foundations of Software Technology and Theoretical Computer Science'10, LIPIcs 8, Leibniz-Zentrum fuer Informatik, 2010, pp. 133–144.

[10] E. Lorini, A Dynamic Logic of Agency II: Deterministic DLA, Coalition Logic, and Game Theory., Journal of Logic, Language, and Information' 19 (2010) 327–351.

[11] J. van Eijck, PDL as a Multi-Agent Strategy Logic., in: Theoretical Aspects of Rationality and Knowledge'13, 2013, pp. 206–215.

[12] K. Chatterjee, T. Henzinger, N. Piterman, Strategy Logic., in: Concurrency Theory'07, LNCS 4703, Springer, 2007, pp. 59–73.

[13] F. Belardinelli, W. Jamroga, V. Malvone, A. Murano, Strategy logic with simple goals: Tractable reasoning about strategies, in: 28th International Joint Conference on Artificial Intelligence (IJCAI 2019), 2019, pp. 88–94.

[14] J. H. Reif, The complexity of two-player games of incomplete information, JCSS 29 (1984) 274–301.

[15] O. Kupferman, M. Vardi, Module checking revisited, in: CAV '96, volume 1254 of *LNCS*, Springer-Verlag, 1997, pp. 36–47.

[16] R. Bloem, K. Chatterjee, S. Jacobs, R. Könighofer, Assume-guarantee synthesis for concurrent reactive programs with partial information, in: TACAS, 2015, pp. 517–532.
[17] A. Pnueli, R. Rosner, Distributed reactive systems are hard to synthesize, in: FOCS, 1990, pp. 746–757.
[18] C. Dima, F. Tiplea, Model-checking ATL under Imperfect Information and Perfect Recall Semantics is Undecidable., Technical Report, arXiv, 2011.
[19] F. Belardinelli, A. Lomuscio, V. Malvone, An abstraction-based method for verifying strategic properties in multi-agent systems with imperfect information, in: Proceedings of AAAI, 2019.
[20] F. Belardinelli, V. Malvone, A three-valued approach to strategic abilities under imperfect information, in: Proceedings of the 17th International Conference on Knowledge Representation and Reasoning, 2020, pp. 89–98.
[21] F. Belardinelli, A. Ferrando, V. Malvone, An abstraction-refinement framework for verifying strategic properties in multi-agent systems with imperfect information, Artif. Intell. 316 (2023) 103847. URL: https://doi.org/10.1016/j.artint.2022.103847. doi:10.1016/j.artint.2022.103847.
[22] F. Belardinelli, A. Lomuscio, V. Malvone, Approximating perfect recall when model checking strategic abilities, in: KR2018, 2018, pp. 435–444.
[23] F. Belardinelli, A. Lomuscio, V. Malvone, E. Yu, Approximating perfect recall when model checking strategic abilities: Theory and applications, J. Artif. Intell. Res. 73 (2022) 897–932. URL: https://doi.org/10.1613/jair.1.12539. doi:10.1613/jair.1.12539.
[24] A. Ferrando, V. Malvone, Towards the combination of model checking and runtime verification on multi-agent systems, in: F. Dignum, P. Mathieu, J. M. Corchado, F. de la Prieta (Eds.), Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation. The PAAMS Collection - 20th International Conference, PAAMS 2022, L'Aquila, Italy, July 13-15, 2022, Proceedings, volume 13616 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 140–152. URL: https://doi.org/10.1007/978-3-031-18192-4_12. doi:10.1007/978-3-031-18192-4_12.
[25] A. Ferrando, V. Malvone, Towards the verification of strategic properties in multi-agent systems with imperfect information, in: N. Agmon, B. An, A. Ricci, W. Yeoh (Eds.), Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023, ACM, 2023, pp. 793–801. URL: https://dl.acm.org/doi/10.5555/3545946.3598713. doi:10.5555/3545946.3598713.
[26] 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. URL: https://doi.org/10.1007/s10009-015-0378-x. doi:10.1007/S10009-015-0378-X.
[27] D. Kurpiewski, W. Jamroga, M. Knapik, STV: model checking for strategies under imperfect information, in: E. Elkind, M. Veloso, N. Agmon, M. E. Taylor (Eds.), Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019, International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 2372–2374. URL: http://dl.acm.org/citation.cfm?id=3332116.
[28] A. Ferrando, V. Malvone, VITAMIN: A compositional framework for model checking of multi-agent systems, CoRR abs/2403.02170 (2024). URL: https://doi.org/10.48550/arXiv.2403.02170. doi:10.48550/ARXIV.2403.02170. arXiv:2403.02170.
[29] 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. URL: https://ceur-ws.org/Vol-3735/paper_12.pdf.
[30] R. Alur, T. A. Henzinger, O. Kupferman, Alternating-time temporal logic, J. ACM 49 (2002) 672–713. URL: https://doi.org/10.1145/585265.585270. doi:10.1145/585265.585270.
[31] 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.

[32] W. Jamroga, V. Malvone, A. Murano, Natural strategic ability, Artif. Intell. 277 (2019).
[33] F. Belardinelli, W. Jamroga, V. Malvone, M. Mittelmann, A. Murano, L. Perrussel, Reasoning about human-friendly strategies in repeated keyword auctions, in: AAMAS 2022, 2022, pp. 62–71.
[34] H. N. Nguyen, N. Alechina, B. Logan, A. Rakib, Alternating-time temporal logic with resource bounds, J. Log. Comput. 28 (2018) 631–663.
[35] 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.
[36] 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.
[37] 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.
[38] 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.