Give Me a Hand: How to Use Model Checking for Multi-Agent Systems to Help Runtime Verification and Vice Versa

Angelo Ferrando[1,*], Vadim Malvone[2]

[1]University of Genoa, Italy
[2]Telecom Paris, France

Abstract

In this paper, we review the history of model checking and runtime verification on multi-agent systems by recalling the results obtained in the two research areas. Then, we present some past, present and future directions to combine these techniques in the two possible sides, that is by using model checking for multi-agent systems to solve runtime verification problems and vice versa.

Keywords

LaTeX class, paper template, paper formatting, CEUR-WS

1. Introduction

Software systems cannot be trusted. Even though this sounds as a bold statement, it is most of the time the case. A software system, to be considered trustworthy, has to offer some guarantees to the end user. Amongst these guarantees, correctness is by far one of the most challenging to demonstrate. However, existent solutions to proof software correctness mainly focus on monolithic systems. So, systems that do not usually present any kind of autonomy, nor distribution, whatsoever. Unfortunately (in some sense), nowadays, artificial intelligent systems can be found everywhere. Thus, techniques to tackle their verification in order to establish their correctness also need to be revised (and adapted).

When we talk about software verification, we mainly refer to standard approaches such as: testing, simulation, and formal verification. Testing and simulation have one main issue: they can detect errors but can not determine their absence. To overcome this problem, *formal verification* results to be very useful. This approach provides a formal-based methodology to model systems, specify properties, and verify that a system satisfies a given specification.

In formal verification, the specification is usually based on temporal logics. The latter can describe the order of events without introducing the time explicitly. In temporal logics, we mainly distinguish between linear- and branching-time logics, which reflect the underlying

IPS-RiCeRcA-SPIRIT 2022: 10th Italian Workshop on Planning and Scheduling, RiCeRcA Italian Workshop, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy.

*Corresponding author.
angelo.ferrando@unige.it (A. Ferrando); vadim.malvone@telecom-paris.fr (V. Malvone)
0000-0002-8711-4670 (A. Ferrando); 0000-0001-6138-4229 (V. Malvone)

nature of the time we consider. The most popular temporal logics are LTL (linear-time temporal logic) [34], CTL (computation tree logic) [14], and their extension CTL^* [18]. An outstanding development in the area of temporal logics has been the discovery of algorithmic methods to verify properties of finite-state systems represented by Kripke structures [28]. Hence, the formal verification of a system modelled by a Kripke structure M with respect to a temporal logic specification φ can be rephrased as "Is M a model of φ?", which explains the name *model checking* (MC), as it was coined by Clarke and Emerson in [14].

Naturally, model checking is not the only existent formal verification technique. Specifically, another verification approach focused on a more dynamic perspective (execution of the system rather than verification of an abstraction of the latter) is called Runtime Verification [32].

Runtime verification (RV) is being pursued as a lightweight verification technique bridging static verification techniques, such as MC, and testing. One of the main distinguishing features of RV is due to its nature of being performed at runtime, which opens up the possibility to act whenever incorrect behavior of a software system is detected. A fault is defined as the deviation between the current behavior and the expected behavior of the system [32, 16]. A fault might lead to a failure, but not necessarily. An error, on the other hand, is a mistake made by a human that results in a fault and possibly in a failure. Runtime verification is the discipline of computer science that deals with the study, development, and application of those verification techniques that allow checking whether a run of a system under scrutiny satisfies or violates a given correctness property.

In [20], we presented an initial vision on possible overlapping of RV and MC, especially in the area of Multi-Agent Systems (MAS); which are distributed systems comprised of intelligent components (called agents) that can be deployed to solve complex tasks in an autonomous fashion (which may, or may not, involve cooperation and communication amongst the agents). We focused on MAS for their implicit complexity, and consequently hard verification problem. Nonetheless, in [20] we mainly focused on the verification aspects of combining RV and MC. Instead, in here, we present some novel results in that direction, but we also focus on the synergy between RV and MAS. Thus, not only focusing on how RV can be deployed to verify MAS, but also on how the verification for MAS can be exploited to support RV.

In the rest of the paper we first discuss the state of the art of model checking and runtime verification for MAS (Section 2). Then, in Section 3 we present our results to use runtime verification to reduce the model checking complexity on MAS and conclude with future directions. Finally, in Section 4 we present model checking techniques to help runtime verification and conclude with some future works.

2. State of the art

Model Checking for MAS. One of the most important developments in this field is *Alternating-Time Temporal Logic* (ATL), introduced by Alur, Henzinger, and Kupferman [4]. Such a logic allows to reason about strategies of agents having the satisfaction of temporal goals as payoff criterion. More formally, it is obtained as a generalization of CTL, in which the existential E and the universal A *path quantifiers* are replaced with *strategic modalities* of the form $\langle\langle\Gamma\rangle\rangle$ and $[\![\Gamma]\!]$, where Γ is a set of *agents*. Despite its expressiveness, ATL suffers

from the strong limitation that strategies are treated only implicitly in the semantics of such modalities. This restriction makes the logic less suited to formalize several important solution concepts, such as the *Nash Equilibrium*. These considerations led to the introduction of *Strategy Logic* (SL) [33], a more powerful formalism for strategic reasoning. As a key aspect, this logic treats strategies as *first-order objects* that can be determined by means of the existential $\exists x$ and universal $\forall x$ quantifiers, which can be respectively read as *"there exists a strategy x"* and *"for all strategies x"*. Notably, a strategy is a generic conditional plan that at each step of the game prescribes an action. With more detail, there are two main classes of strategies: memoryless and memoryful. In the former case, agents choose an action by considering only the current game state while, in the latter case, agents choose an action by considering the full 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 the others, Strategy Logic with Simple-Goals [6] considers SL formulas in which strategic operators, bindings operators, and temporal operators are coupled. It has been shown that Strategy Logic with Simple-Goals strictly subsume ATL and its MC problem is P-Complete, as it is for ATL. To conclude this section, we want to focus on a key aspect in MAS: the agents' visibility. Specifically, we distinguish between *perfect* and *imperfect* information games [35]. The former corresponds to a basic setting in which every agent has full knowledge about the game. However, in real-life scenarios it is common to have situations in which agents have to play without having all relevant information at hand. In computer science these situations occur for example when some variables of a system are internal/private and not visible to an external environment [29, 13]. In game models, the imperfect information is usually modelled by setting an indistinguishability relation over the states of the game [29, 35]. This feature deeply impacts on the MC complexity. For example, ATL becomes undecidable in the context of imperfect information and memoryful strategies [17]. To overcome this problem, some works have either focused on an approximation to perfect information [8, 11] or developed new notions on strategies [7, 26, 27, 10, 12, 9].

Runtime Verification for MAS. In [2], the authors presented a framework to verify at runtime agent interaction protocols (AIP). The formalism used in this work allows the introduction of variables, that are then used to constrain the expected behavior in a more expressive way. In [19], the same authors proposed an approach to verify at runtime AIP using multiple monitors. This is obtained by decentralizing the global specification (specified as a Trace Expression [1]), which is used to represent the global protocol, into partial specifications denoting the single agents' perspective. In [5, 36], other works on runtime verification of agent interactions are proposed, and in [30] a framework for dynamic adaptive MAS (DAMS-RV) based on an adaptive feedback loop is presented. Other approaches to MAS RV include the proposals spin-off from the SOCS project where the SCIFF computational logic framework [3] is used to provide the semantics of social integrity constraints. To model MAS interaction, expectation-based semantics specifies the links between observed and expected events, providing a means to test runtime

conformance of an actual conversation with respect to a given interaction protocol [38]. Similar work has been performed using commitments [15].

3. Runtime verification gives a hand to Model checking for MAS

The model checking problem for ATL giving a generic MAS is known to be undecidable. Nonetheless, decidable fragments exist. Indeed, model checking ATL under perfect information is PTIME-complete [4], while under imperfect information and imperfect recall is PSPACE [37]. Unfortunately, MAS usually have imperfect information, and when memory is needed to achieve the goals, the resulting model checking problem becomes undecidable. Given the relevance of the imperfect information setting, even partial solutions to the problem are useful.

The only existent work on exploiting RV to formally verify the strategic behaviour in MAS is [24, 22]. In such work, given an ATL formula φ and a model of MAS M, the procedure extracts all the sub-models of M with perfect information that satisfy a sub-formula of φ. Then, runtime monitors are used to check if the remaining part of φ can be satisfied at execution time. If this is the case, we conclude at runtime the satisfaction of φ for the corresponding system execution. Note that, this does not imply that the system satisfies φ, indeed future executions may violate φ. The formal result over φ only concerns the current execution, and how it has behaved in it. However, the following preservation results holds.

Lemma 1. *Given a model M and an ATL formula φ, for any history h of M starting in s_I, we have that:*

$$Monitor_{\varphi_{LTL}}(h) = \top \;\Rightarrow\; M, s_I \models \varphi_{Ag}$$

$$Monitor_{\varphi_{LTL}}(h) = \bot \;\Rightarrow\; M, s_I \not\models \varphi_{\emptyset}$$

where φ_{LTL} is the variant of φ where all strategic operators are removed, φ_{Ag} is the variant of φ where all strategic operators are converted into $\langle\langle Ag \rangle\rangle$, $\varphi_{\emptyset}$ is the variant of φ where all strategic operators are converted into $\langle\langle \emptyset \rangle\rangle$.

Lemma 1 shows a preservation result from RV to ATL model checking that needs to be discussed. If our monitor returns true we have two possibilities:

1 The procedure found an under-approximation sub-model in which the original formula φ is satisfied then it can conclude the verification procedure by using RV only by checking that the atom representing φ holds in the initial state of the history h given in input;
2 A sub-formula φ' is satisfied in an under-approximation sub-model and at runtime the formula φ_{Ag} holds on the history h given in input.

While case (1) gives a preservation result for the formula φ given in input, case (2) checks formula φ_{Ag} instead of φ. That is, it substitutes Ag as coalition for all the strategic operators of φ but the ones in φ'. So, our procedure approximates the truth value by considering the case in which all the agents in the game collaborate to achieve the objectives not satisfied in

the model checking phase. That is, while in [8, 11] the approximation is given in terms of information, in [7] is given in terms of memory of strategies, and in [21] the approximation is given by generalizing the logic, here we give results by approximating the coalitions. So, the main limitation of this approach concerns this aspect. Furthermore, we recall that the procedure produces always results, even partial. This aspect is strongly relevant in concrete scenario in which there is the necessity to have some sort of verification results. For example, in the context of swarm robots, with this procedure we can verify macro properties such as "the system works properly" since we are able to guarantee fully collaboration between agents because this property is relevant and desirable for each agent in the game. The same reasoning described above, can be applied in a complementary way for the case of over-approximation sub-models and the falsity.

Note that this is the first attempt of using runtime verification to verify strategic properties on MAS. Thus, even though the solution might not be optimal, it is a milestone for the corresponding lines of research. Additional works will be done to improve the technique and, above all, its implementation. For instance, we are planning to extend this work by considering a more predictive flavour.

4. Model checking for MAS gives a hand to Runtime verification

Runtime Verification is built on the assumption of perfect information over the system, that is, the monitor checking the system can perceive everything. Unfortunately, this is not always the case, especially when the system under analysis contains rational/autonomous components and is deployed in real-world environments with possibly faulty sensors. In [23], an extension of the standard Runtime Verification of Linear Temporal Logic properties to consider scenarios with imperfect information is presented; along with all the engineering steps necessary to update the verification pipeline. Moreover, a corresponding implementation is proposed and applied to a case study involving robotic systems. In particular, [23] defines the notion of imperfect information w.r.t. the monitor's visibility over the system, and then re-engineers the LTL monitor's synthesis pipeline to recognise such visibility information.

In [23], imperfect information is specified by means of a indistinguishability relation $\sim$ over the atomic propositions. Intuitively, given two atomic propositions p and q, it is said that they are indistinguishable if and only if $p \sim q$.

To handle the verification process in the imperfect information scenario, the standard RV approach needs to be extended. This extension is based on the notion of duplication of the atoms. The latter is used in order to make the truth value of each atomic proposition explicit.

The new monitor is defined as follows.

Definition 1 (Monitor with imperfect information). *Given an LTL formula φ and a visible*

history h_v, a monitor with imperfect information is so defined:

$$Monitor^{v}_{\varphi}(h_v) = \begin{cases} \top & h_v \in \mathcal{L}(\varphi) \wedge h_v \notin \mathcal{L}(\neg\varphi) \wedge h_v \notin \mathcal{L}(\otimes\varphi) \\ \bot & h_v \notin \mathcal{L}(\varphi) \wedge h_v \in \mathcal{L}(\neg\varphi) \wedge h_v \notin \mathcal{L}(\otimes\varphi) \\ uu & h_v \notin \mathcal{L}(\varphi) \wedge h_v \notin \mathcal{L}(\neg\varphi) \wedge h_v \in \mathcal{L}(\otimes\varphi) \\ ?_{\not\perp} & h_v \in \mathcal{L}(\varphi) \wedge h_v \notin \mathcal{L}(\neg\varphi) \wedge h_v \in \mathcal{L}(\otimes\varphi) \\ ?_{\not\top} & h_v \notin \mathcal{L}(\varphi) \wedge h_v \in \mathcal{L}(\neg\varphi) \wedge h_v \in \mathcal{L}(\otimes\varphi) \\ ? & h_v \in \mathcal{L}(\varphi) \wedge h_v \in \mathcal{L}(\neg\varphi) \wedge h_v \in \mathcal{L}(\otimes\varphi) \end{cases}$$

Where $\mathcal{L}(\varphi)$ is the language of histories satisfying φ, $\mathcal{L}(\neg\varphi)$ is the language of histories violating φ, and finally, $\mathcal{L}(\otimes\varphi)$ is the language of histories making φ undefined (because of lack of information).

In what follows, we provide two preservation results from the monitor with imperfect information to the one with perfect information.

Lemma 2. *Given a finite history σ, a monitor with its visibility $Monitor^{v}_{\varphi}(h)$, and a general monitor $Monitor_{\varphi}(h)$, we have that:*

$$\textit{if } Monitor^{v}_{\varphi}(h_v) = \top \textit{ then } Monitor_{\varphi}(h) = \top$$
$$\textit{if } Monitor^{v}_{\varphi}(h_v) = \bot \textit{ then } Monitor_{\varphi}(h) = \bot$$

The above results can be extended to consider multi-monitors and solved by using formal verification for MAS. In fact, even though a single monitor does not have access to all the information it needs, multiple monitors might. As in standard distributed RV [25], we are planning to explore this idea as well, but focusing on the theoretical foundations of information sharing amongst monitors. Specifically, we are going to show how this can be specified as a multi-agent problem; where each monitor is denoted as an agent with the goal of gathering all the information needed to carry out the verification of its formal property. By representing the information sharing as a multi-agent problem, we could gain from different viewpoints. In particular, we could exploit existing techniques, used for strategic reasoning [31], to guide the information sharing amongst the monitors. Moreover, by denoting monitors as agents, we would recognise their autonomy on what concern the sharing of private information. For instance, a monitor could be the only one with the access to a certain resource, and it is not realistic to assume it will freely share such information. This because the act of sharing information is not free of charge, since it requires to both consume computation time and bandwidth. For this, and other reasons, it is of paramount importance to take into consideration the cost of sharing. We are going to show how such cost can be ported into the multi-agent problem, and how it can guide the selection of the information sharing strategy of the agents (*i.e.*, the monitors).

In particular, we are going to study the theoretical foundation of information sharing amongst runtime monitors. We will tackle this by porting the problem into a multi-agent setting. By dosing so, we apply existing formal verification techniques, such as model checking [31], to exploit the strategic reasoning of the agents to overcome the information sharing problem. We will not only present theoretical results, but we will also propose an implementation prototype, as a proof of concept.

References

[1] D. Ancona, A. Ferrando, and V. Mascardi. Comparing trace expressions and linear temporal logic for runtime verification. In *TPFM*, pages 47–64, 2016.

[2] D. Ancona, A. Ferrando, and V. Mascardi. Parametric runtime verification of multiagent systems. In *AAMAS*, pages 1457–1459, 2017.

[3] M. Alberti, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. The *S*ciff abductive proof-procedure. In *AI*IA*, pages 135–147, 2005.

[4] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-Time Temporal Logic. *JACM*, 49(5):672–713, 2002.

[5] N. Bakar and A. Selamat. Runtime verification of multi-agent systems interaction quality. In *ACIIDS*, pages 435–444, 2013.

[6] F. Belardinelli, W. Jamroga, D. Kurpiewski, V. Malvone, and A. Murano. Strategy logic with simple goals: Tractable reasoning about strategies. In IJCAI, pages 88–94, 2019.

[7] F. Belardinelli, A. Lomuscio, and V. Malvone. Approximating perfect recall when model checking strategic abilities. In *KR2018*, pages 435–444, 2018.

[8] F. Belardinelli, A. Lomuscio, and V. Malvone. An abstraction-based method for verifying strategic properties in multi-agent systems with imperfect information. In *AAAI*, 2019.

[9] F. Belardinelli, A. Lomuscio, V. Malvone, and E. Yu. Approximating perfect recall when model checking strategic abilities: Theory and applications. *JAIR*, 73:897–932, 2022.

[10] F. Belardinelli, A. Lomuscio, A. Murano, and S. Rubin. Verification of multi-agent systems with public actions against strategy logic. *AIJ*, 285:103302, 2020.

[11] F. Belardinelli and V. Malvone. A three-valued approach to strategic abilities under imperfect information. In *KR*, pages 89–98, 2020.

[12] R. Berthon, B. Maubert, A. Murano, S. Rubin and, M. Y. Vardi. Strategy Logic with Imperfect Information. *TOCL*, 22(1):1–51, 2021.

[13] R. Bloem, K. Chatterjee, S. Jacobs, and R. Könighofer. Assume-guarantee synthesis for concurrent reactive programs with partial information. In *TACAS*, pages 517–532, 2015.

[14] E.M. Clarke and E.A. Emerson. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *LP*, pages 52–71, 1981.

[15] F. Chesani, P. Mello, M. Montali, and P. Torroni. Commitment tracking via the reactive event calculus. In *IJCAI*, pages 91–96, 2009.

[16] N. Delgado, A. Gates, and S. Roach. A taxonomy and catalog of runtime software-fault monitoring tools. *IEEE TSE*, 30(12):859–872, 2004.

[17] C. Dima and F.L. Tiplea. Model-checking ATL under Imperfect Information and Perfect Recall Semantics is Undecidable. Technical report, arXiv, 2011.

[18] E.A. Emerson and J.Y. Halpern. "Sometimes" and "Not Never" Revisited: On Branching Versus Linear Time. *JACM*, 33(1):151–178, 1986.

[19] A. Ferrando, D. Ancona, and V. Mascardi. Decentralizing MAS monitoring with decamon. In *AAMAS*, pages 239–248, 2017.

[20] A. Ferrando and V. Malvone. Combine model checking and runtime verification in multi-agent systems. In ICTCS, pages 302–310, 2021.
[21] A. Ferrando and V. Malvone. Towards the verification of strategic properties in multi-agent systems with imperfect information. *CoRR*, abs/2112.13621, 2021.
[22] A. Ferrando and V. Malvone. Strategy RV: A Tool to Approximate ATL Model Checking under Imperfect Information and Perfect Recall. In AAMAS, pages 1764–1766, 2021.
[23] A. Ferrando and V. Malvone. Runtime verification with imperfect information through indistinguishability relations. In SEFM, pages 335–351, 2022.
[24] A. Ferrando and V. Malvone. Towards the combination of model checking and runtime verification on multi-agent systems. In PAAMS, pages 140–152, 2022.
[25] A. Francalanza, J. A. Pérez, and C. Sánchez. Runtime verification for decentralised and distributed systems. In LRVIAT, pages 176–210, 2018.
[26] W. Jamroga, V. Malvone, and A. Murano. Natural strategic ability. *AIJ*, 277:103170, 2019.
[27] W. Jamroga, V. Malvone, and A. Murano. Natural Strategic Ability under Imperfect Information. In *AAMAS*, 962–970, 2019.
[28] S.A. Kripke. Semantical Considerations on Modal Logic. *APF*, 16:83–94, 1963.
[29] O. Kupferman and M.Y. Vardi. Module checking revisited. In *CAV*, pages 36–47, 1997.
[30] Y. J. Lim, G. Hong, D. Shin, E. Jee, and D.-H- Bae. A runtime verification framework for dynamically adaptive multi-agent systems. In *BigComp*, pages 509–512, 2016.
[31] A. Lomuscio and F. Raimondi. Model checking knowledge, strategies, and games in multi-agent systems. In *AAMAS*, pages 161–168, 2006.
[32] M. Leucker and C. Schallhart. A brief account of runtime verification. *JLAP*, 78(5):293 – 303, 2009.
[33] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. Reasoning About Strategies: On the Model-Checking Problem. *ACM TCL*, 15(4): 34:1-34:47, 2014.
[34] A. Pnueli. The Temporal Logic of Programs. In *FCS*, pages 46–57, 1977.
[35] J. H. Reif. The complexity of two-player games of incomplete information. *JCSS*, 29(2):274–301, 1984.
[36] C. Roungroongsom and D. Pradubsuwun. Formal verification of multi-agent system based on jade: A semi-runtime approach. In *RAICT*, pages 297–306, 2015.
[37] P.Y. Schobbens. Alternating-Time Logic with Imperfect Recall. 85(2):82–93, 2004.
[38] P. Torroni, P. Yolum, M. P. Singh, M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, and P. Mello. Modelling interactions via commitments and expectations. In *HRMASSDOM*, 2009.