Some Approaches to Rational Verification in Multiagent Systems

Muhammad Najib

University of Oxford, Oxford, UK
mnajib@cs.ox.ac.uk

Abstract

Recently, with the rapid advances of artificial intelligence, many researchers from verification community are starting to work on the analysis of systems composed of (semi)autonomous components known as multiagent systems. With this increasing interest, many concepts for reasoning about the behaviour of such systems are proposed. Among them is rational verification which is concerned with establishing whether a property can be sustained in a system composed of rational agents. In our research, we study different approaches to realise the notion of rational verification and strive for a concrete tool implementing the paradigm. We begin by introducing a formal framework as the foundation of our approaches. We then discuss the ability of current techniques/tools to perform rational verification and present some methods we have developed to expand the limits. We conclude with our ongoing work and possible future directions.

Introduction

We are interested in the verification of concurrent multiagent systems, in which processes are assumed as open systems. In this approach, a system is modelled as a *game*. System components are represented as *players* with their own *strategies*, possible computation runs are the *plays* of the game, and the required property of each player is specified with a *goal* that the player wants to satisfy. The usage of this game-theoretic approach gives rise to a natural question: *Does the system have a stable behaviour?* This question boils down to checking whether the strategies chosen by the players are in equilibrium [23]. This setting is such that, instead of model checking, we talk about *equilibrium checking* [33]. In fact, model checking is a special case of equilibrium checking, where cooperation is being forced to the players or the system is simply modelled as a single-player game.

Our ongoing work looks at the different approaches to perform the concept of rational verification. We begin by introducing a formal framework as the foundation of our approaches. We then discuss the ability of existing tools available to perform rational verification, including their limitations. Given this, we then present some methods we have proposed to expand the limits. We conclude with our future work and possible directions.

Framework

Let $(1, \dots, n)$ be the set of agents within a multiagent system. We assume that agents are nondeterministic reactive programs/modules. Nondeterminism means that agents can freely choose actions available to them without any authority telling them what to do. Reactive means that agents are nonterminating as long as the system is running. This general framework can be applied to different kinds of computational models, such as event structures [32], interpreted systems [9], concurrent games [2], or multiagent planning systems [4].

A *strategy* for agent i is a rule that defines how the agent makes choices throughout the run of the system. There are different ways to define strategy, but we assume that strategy is behavioural and generally can think of it as a function from what an agent can "see" to the choices available to them. We denote the set of stategies available to i by Σ_i. When each agent has selected a strategy, we have a *strategy profile* $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$. We assume that strategies are deterministic, thus each strategy profile induces a unique run denoted by $\rho(\vec{\sigma})$. We write $\rho \models \varphi$ to denote that run ρ satisfies φ. We now define agents *preferences* over runs of the system. We write $\rho_1 \succeq_i \rho_2$ to mean that an agent i with the goal γ_i prefers ρ_1 at least as much as ρ_2, thus formally $\rho_1 \succeq_i \rho_2$ if and only if $\rho_2 \models \gamma_i$ implies $\rho_1 \models \gamma_i$.

We then define the standard game theoretic concept of *Nash equilibrium*. Let $G = \langle (1, \ldots, n), (\gamma_1, \ldots, \gamma_n) \rangle$ be a multiagent system modelled as a game, and $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$ a strategy profile. We say $\vec{\sigma}$ is a Nash equilibrium of G if for all i and for all $\sigma'_i \in \Sigma_i$, we have $\rho(\sigma) \succeq_i \rho(\sigma_1, \ldots, \sigma'_i, \ldots, \sigma_n)$. We write $NE(G)$ to denote the set of Nash equilibria in G. With these definitions established, we can now address the main problems in rational verification.

The concept of rational verification can be regarded as a counterpart to classical verification with a more "restricted" condition. Given a multiagent system modelled as a (concurrent) game G (as well as a property φ and strategy profile $\vec{\sigma}$ for, respectively, Problem 2 and 3), we can capture the idea in these following decision problems [33].

Problem 1 (NE-Emptiness)**.** *Given a multiagent system G. Is it the case that $NE(G) \neq \emptyset$?*

Problem 2 (E/A-Nash)**.** *Given a multiagent system G and temporal formula φ. Is it the case that $\rho(\vec{\sigma}) \models \varphi$ in any/all $\vec{\sigma} \in NE(G)$?*

Problem 3 (NE-Membership)**.** *Given a multiagent system G and strategy profile $\vec{\sigma}$. Is it the case that $\vec{\sigma} \in NE(G)$?*

Now, we need a language to model the systems. For this, we use SRML [29] which is a strict subset of RML [1], the modelling language used in some established model checkers such as (Nu)SMV [19, 7], MOCHA [3], and PRISM [16]. An agent i is modelled as an SRML module $m_i = (\Phi_i, I_i, U_i)$, where Φ_i is a set of Boolean variables controlled by the agent, I_i a finite set of variable values initialisation commands, and U_i a set of variable values update commands. Due to space constraint, we will not discuss in detail the semantics of SRML here. However, it has been shown that SRML can model concurrent game-like systems such as Reactive Modules Games [11] and concurrent game structures [29].

Rational Verification with Existing Tools

The focus on open systems verification has led to the development of richer and more expressive formalisms. While LTL, CTL, and CTL* are adequately expressive for reasoning about computations of some systems which behaviour are completely deterministic, they are obviously not appropriate for systems with nondeterministic components. Alternating-time temporal logic (ATL*) is one of newer formalisms in which one can reason about time and strategic abilites of players [2]. However, ATL* is still not powerful enough to reason about Nash equilibria [1].

Strategy Logic (SL) is a more expressive logic and, in fact, stictly subsumes ATL*. We can quantify strategies explicitely, thus powerful enough to express Nash equilibria. BDD-based symbolic model checking for SL is implemented in MCMAS [17]. Despite the potential of SL, there is currently no explicit support to do rational verification in MCMAS. We have to manually devise a meta-algorithm performing rational verification in MCMAS input file. Recently, we proposed a prototype tool called EVE [22] that addresses Problem 1 by bridging

[1]There exists an extension of ATL (a strict subset of ATL*) called ATLES [31] that can express Nash equilibria, however, it is shown only in extensive games.

SRML with MCMAS. However, with great power comes great cost. Model checking with SL is hard, and the fragments that are useful for expressing the existence of Nash equilibra have undecidable satisfiability problem in perfect recall setting [20, 21]. Current implementation of SL in MCMAS that can answer Problem 1 is only under imperfect recall semantics.

Another existing (prototype) tool for rational verification is EAGLE [28]. This tool specifically addresses Problem 3. It uses CTL as its specification language and relies on two oracles: CTL model checker and CTL SAT solver. In EAGLE detailed report, it is shown that the bottleneck is the CTL satisfiability subroutine [27]. A closer tool to EAGLE is PRALINE [5], however, it focuses on synthesis/constructing strategies in Nash equilibrium.

Ongoing and Future Work

We are currently working on an approach to solve problems in rational verification via parity. One of the fundamental properties of parity games is the property of *memoryless determinacy*. The main idea is by transforming a concurrent game with temporal logic goals into a game with parity condition goals, we can exploit the memoryless determinacy of parity games to our advantage. Solving parity game played on a finite graph is in NP $\cap$ Co-NP, or more precisely UP $\cap$ Co-UP [8, 15]. In practice, there are many deterministic algorithms have been invented such as recursive method in [34], local μ-calculus model checking in [26], small progress algorithm [13, 12], strategy improvement algorithm [30, 24]. Although there is currently no polynomial algorithm found, subexponential [14, 25] and quasi-polynomial [6] algorithms have been proposed.

We have developed an algorithm that specifically solves Problem 1 and conjectured that it matches its theoretical bound shown in [10]. We are also working on extending it to be able to address Problem 2 and 3. While we are working on the theoretical side, a tool implementing the algorithm is also currently being built. The tool uses SRML syntax very close to the one described in [29] [2].

Other work that we are considering is optimising EAGLE with BDD-based CTL satisfiability [18] instead of tableaux-based currently used in EAGLE's library. Another thing is finding a way to directly inject SRML structure in EVE into BDD-based model checker, instead of going through MCMAS dedicated input language. With these works, we are hoping to be able to increase the tools perfomance.

For future work, we see a number of directions we can pursue. Although Nash equilibrium is one of the most widely used solution concepts in game theory, it would be a good idea to consider other solution concepts. It is also useful to extend the specifications beyond current assumptions, *e.g.*, quantitative and probabilistic, as well as stochastic setting so that more general agent's beliefs and preferences can be captured. Finally, from an implementation point of view, it would be helpful to have a more user-friendly interface for operating the tool such as via GUI.

References

[1] R. Alur and T. A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(11):7–48, July 1999.

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

[2] This SRML style is, in fact, also used in EVE.

[3] R. Alur, T. A. Henzinger, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Taşiran. Mocha: Modularity in model checking. In *CAV 1998: Tenth International Conference on Computer-aided Verification, (LNCS Volume 1427)*, pages 521–525. Springer-Verlag: Berlin, Germany, 1998.

[4] R. Brafman, C. Domshlak, Y. Engel, and M. Tennenholtz. Planning games. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, 2009.

[5] R. Brenguier. *PRALINE: A Tool for Computing Nash Equilibria in Concurrent Games*, pages 890–895. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[6] C. S. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan. Deciding parity games in quasipolynomial time. *STOCS*, 2017. To appear.

[7] A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *Proceedings of the 14th International Conference on Computer Aided Verification*, CAV '02, pages 359–364, London, UK, UK, 2002. Springer-Verlag.

[8] E. Emerson, C. S. Jutla, and A. Sistla. On model checking for the -calculus and its fragments. *Theoretical Computer Science*, 258(1):491 – 522, 2001.

[9] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. The MIT Press: Cambridge, MA, 1995.

[10] J. Gutierrez, P. Harrenstein, and M. Wooldridge. Iterated boolean games. *Inf. Comput.*, 242(C):53–79, June 2015.

[11] J. Gutierrez, P. Harrenstein, and M. Wooldridge. From model checking to equilibrium checking: Reactive modules for rational verification. *Artif. Intell.*, 248:123–157, 2017.

[12] K. Heljanko, M. Keinnen, M. Lange, and I. Niemel. Solving parity games by a reduction to sat. *Journal of Computer and System Sciences*, 78(2):430 – 440, 2012. Games in Verification.

[13] M. Jurdzinski. Small progress measures for solving parity games. In *Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science*, STACS '00, pages 290–301, London, UK, UK, 2000. Springer-Verlag.

[14] M. Jurdziński, M. Paterson, and U. Zwick. A deterministic subexponential algorithm for solving parity games. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 117–123, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics.

[15] M. Jurdziski. Deciding the winner in parity games is in up co-up. *Information Processing Letters*, 68(3):119 – 124, 1998.

[16] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[17] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: a model checker for the verification of multi-agent systems. In *CAV*, volume 5643 of *LNCS*, pages 682–688. Springer, 2009.

[18] W. Marrero. Using bdds to decide CTL. In *Tools and Algorithms for the Construction and Analysis of Systems, 11th International Conference, TACAS 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings*, pages 222–236, 2005.

[19] K. L. McMillan. *The SMV System*, pages 61–85. Springer US, Boston, MA, 1993.

[20] F. Mogavero, A. Murano, G. Perelli, and M. Y. Vardi. Reasoning about strategies: On the model-checking problem. *CoRR*, abs/1112.6275, 2011.

[21] F. Mogavero, A. Murano, and L. Sauro. On the boundary of behavioral strategies. In *LICS*, pages 263–272. IEEE Computer Society, 2013.

[22] M. Najib. A tool for temporal logic equilibrium analysis of concurrent games. Technical report, University of Oxford, 2016.

[23] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press: Cambridge, MA,

1994.

[24] S. Schewe. *An Optimal Strategy Improvement Algorithm for Solving Parity and Payoff Games*, pages 369–384. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[25] S. Schewe. Solving parity games in big steps. *Journal of Computer and System Sciences*, 84:243 – 262, 2017.

[26] P. Stevens and C. Stirling. *Practical model-checking using games*, pages 85–101. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[27] A. Toumi. Equilibrium checking in reactive modules games. Technical report, University of Oxford, 2015.

[28] A. Toumi, J. Gutierrez, and M. Wooldridge. *Theoretical Aspects of Computing - ICTAC 2015: 12th International Colloquium, Cali, Colombia, October 29-31, 2015, Proceedings*, chapter A Tool for the Automated Verification of Nash Equilibria in Concurrent Games, pages 583–594. Springer International Publishing, Cham, 2015.

[29] W. van der Hoek, A. Lomuscio, and M. Wooldridge. On the complexity of practical ATL model checking. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2006)*, Hakodate, Japan, 2005.

[30] J. Vöge and M. Jurdziński. *A Discrete Strategy Improvement Algorithm for Solving Parity Games*, pages 202–215. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[31] D. Walther, W. van der Hoek, and M. Wooldridge. Alternating-time temporal logic with explicit strategies. In *Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK '07, pages 269–278, New York, NY, USA, 2007. ACM.

[32] G. Winskel. *Event structures*, pages 325–392. Springer Berlin Heidelberg, Berlin, Heidelberg, 1987.

[33] M. Wooldridge, J. Gutierrez, P. Harrenstein, E. Marchioni, G. Perelli, and A. Toumi. Rational verification: From model checking to equilibrium checking. In *AAAI Conference on Artificial Intelligence*, Phoenix, Arizona, USA, 2016.

[34] W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1):135 – 183, 1998.