Verifying and Designing Equilibria in Multi-Agent Systems*

Muhammad Najib

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

Abstract

Verifying equilibria involves checking which temporal logic properties will hold in some "stable" runs of a system composed of multiple agents which are assumed to behave rationally and strategically in pursuit of individual objectives. This paradigm is called *rational verification*, and can be regarded as a counterpart to (classical) model checking for multi-agent systems. In this talk, I will propose a practically amenable technique for rational verification which relies on a reduction to the solution of a collection of parity games. This approach has been implemented in a tool called EVE. I will also talk about some cases in which the problem of rational verification is computationally tractable. In particular, it is possible to reduce the complexity from 2EXPTIME to fixed-parameter tractable. Finally, I will also introduce a concept called *equilibrium design* which is concerned in the design of incentives so that a desirable equilibrium is obtained.

Rational Verification

In the *rational verification* problem, we desire to check which temporal logic properties are satisfied by the system/game *in equilibrium*, that is, assuming players select strategies that form a Nash equilibrium. A little more formally, let $P_1, \ldots, P_n$ be the agents in our concurrent/multi-agent system, and let $\mathrm{NE}(P_1, \ldots, P_n)$ denote the set of all computation runs of the system that could be generated by agents selecting strategies that form a Nash equilibrium. Finally, let φ be a temporal logic formula. Then, in the rational verification problem, we want to know whether for some/every run $\pi \in \mathrm{NE}(P_1, \ldots, P_n)$ we have $\pi \models \varphi$.

Let $(1, \ldots, n)$ 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 [17], interpreted systems [4], concurrent games [1], or multiagent planning systems [3].

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 $\rho_1 \models \gamma_i$.

*Based on author's DPhil/PhD thesis: Rational Verification in Multi-Agent Systems.

We then define the standard game theoretic concept of *Nash equilibrium*. Let $\mathcal{G} = \langle (1, \ldots, n), (\gamma_1, \ldots, \gamma_n) \rangle$ 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 $\mathcal{G}$ 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 $\text{NE}(\mathcal{G})$ to denote the set of Nash equilibria in $\mathcal{G}$. 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 $\mathcal{G}$ (as well as a property φ for Problem 2), we can capture the idea in these following decision problems [18, 8].

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

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

Proposed Approach. The technique we develop to solve above-mentioned problems consists of three steps. First, we build a Parity game $\mathcal{G}_{\text{PAR}}$ from an input LTL game $\mathcal{G}_{\text{LTL}}$. Then—using a characterisation of Nash equilibrium that separates players in the game into those that achieve their goals in a Nash equilibrium (the "winners", W) and those that do not achieve their goals (the "losers", L)—for each set of players in the game, we eliminate nodes and paths in $\mathcal{G}_{\text{PAR}}$ which cannot be a part of a Nash equilibrium, thus producing a modified Parity game, $\mathcal{G}_{\text{PAR}}^{-L}$. Finally, in the third step, we use Streett automata on infinite words to check if the obtained Parity game witnesses the existence of a Nash equilibrium.

The procedure presented above runs in doubly exponential time, matching the *optimal* upper bound of the problem. In the first step we obtain a doubly exponential blowup. The underlying structure $\mathcal{M}$ of the obtained Parity game $\mathcal{G}_{\text{PAR}}$ is doubly exponential in the size of the goals of the input LTL game $\mathcal{G}_{\text{LTL}}$, but the priority functions set $(\alpha_i)_{i \in \text{N}}$ is only (singly) exponential. Then, in the second step, reasoning takes only polynomial time in the size of the underlying concurrent game structure of $\mathcal{G}_{\text{PAR}}$, but exponential time in both the number of players and the size of the priority functions set. Finally, the third step takes only polynomial time, leading to an overall 2EXPTIME complexity.

Implementation. The procedure above is implemented in a tool called EVE[1]. Each system component (agent/player) in EVE is represented as a SRML *module*, which consists of an *interface* that defines the name of the module and lists a non-empty set of Boolean variables controlled by the module, and a set of *guarded commands*, which define the choices available to the module at each state. We refer to [16] for further details on the semantics of SRML. In addition, we associate each module with a goal, which is specified as an LTL formula.

EVE[2] is available online as webservice from `http://eve.cs.ox.ac.uk/eve`. EVE takes as input a concurrent and multi-agent system described in SRML, with player goals and a property φ to be checked specified in LTL. For Non-Emptiness, EVE returns "YES" (along with a set of winning players W) if the set of NE in the system is not empty, and returns "NO" otherwise. For E-Nash (A-Nash), EVE returns "YES" if φ holds on *some* (*all*) NE of the system, and "NO" otherwise. EVE also returns a witness for each "YES" instance as a synthesised strategy profile.

[1] Originally appeared in the proceedings of ATVA'18 [10].
[2] EVE is open-source and the code can be obtained from `https://github.com/eve-mas/eve-parity`

Tractable Rational Verification

Rational verification has been studied for a number of settings, including iterated Boolean games, reactive modules games, and concurrent game structures [7, 8, 6, 9]. In all cases, the problem is 2EXPTIME-complete. Rational verification is also closely related to rational synthesis, which is also 2EXPTIME-complete both in the Boolean case [5] and with rational environments [13]. All of the above cases only consider perfect information.

In this section[3], I address the high-complexity issue of rational verification and provide complexity results that greatly improve on the 2EXPTIME-complete result of the general case. In particular, we consider games where the goals of players are represented as either $\mathsf{GR}(1)$ *formulae* (an important fragment of LTL that can express most response properties of a concurrent and reactive system [2]), or *mean-payoff utility functions* (one of the most studied reward and quality measures used in games for automated formal verification). In each case, we study the rational verification problem for system specifications φ given as $\mathsf{GR}(1)$ formulae and as LTL formulae, with respect to system models that are formally represented as concurrent game structures [1]. Our main results show that in the cases above mentioned, the 2EXPTIME result can be dramatically improved, to settings where rational verification can be solved in polynomial space, NP, or even in polynomial time if the number of players in the game is assumed to be fixed.

Designing Equilibrium

Mechanism design is the problem of designing a game such that, if players behave rationally, then a desired outcome will be obtained [15]. In this section[4], I present a new concept related to mechanism design that we call *equilibrium design*. Equilibrium design is the design of subsidy schemes (incentives) for concurrent games, so that a desired outcome (a Nash equilibrium in the game) can be obtained. We model agents as synchronously executing concurrent processes, with each agent receiving an integer payoff for every state the overall system visits; the overall payoff an agent receives over an infinite computation path is then defined to be the mean payoff over this path. While agents (naturally) seek to maximise their individual mean payoff, the designer of the subsidy scheme wishes to see some temporal logic formula satisfied, either on some or on every Nash equilibrium of the game.

We assume that the designer – an external principal – has a finite budget that is available for making subsidies, and this budget can be allocated across agent/state pairs. By allocating this budget appropriately, the principal can incentivise players away from some states and towards others. Since the principal has some temporal logic goal formula, it desires to allocate subsidies so that players are rationally incentivised to choose strategies so that the principal's temporal logic goal formula is satisfied in the path that would result from executing the strategies. Following [14], we identify two variants of the principal's mechanism design problem, which we refer to as Weak Implementation and Strong Implementation. For these two problems, we consider goals specified by LTL formulae or $\mathsf{GR}(1)$ formulae [2] and classify the complexity of the problem. We then go on to study variations of these two problems, for example considering *optimality* and *uniqueness* of solutions, and show that the complexities of all such problems lie within the polynomial hierarchy, thus making them potentially amenable to efficient practical implementations.

[3] To appear in the proceedings of IJCAI'19 [12].
[4] To appear in the proceedings of CONCUR'19 [11].

References

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

[2] Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. *Journal of Computer and System Sciences*, 78(3):911–938, 2012.

[3] R. Brafman, C. Domshlak, Y. Engel, and M. Tennenholtz. Planning games. 2009.

[4] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. 1995.

[5] Dana Fisman, Orna Kupferman, and Yoad Lustig. Rational Synthesis. In *TACAS*, volume 6015 of *LNCS*, pages 190–204. Springer, 2010.

[6] Julian Gutierrez, Paul Harrenstein, and Michael Wooldridge. Expresiveness and Complexity Results for Strategic Reasoning. In *CONCUR*, volume 42 of *LIPIcs*, pages 268–282. Schloss Dagstuhl, 2015.

[7] Julian Gutierrez, Paul Harrenstein, and Michael Wooldridge. Iterated Boolean Games. *Information and Computation*, 242:53–79, 2015.

[8] Julian Gutierrez, Paul Harrenstein, and Michael Wooldridge. From model checking to equilibrium checking: Reactive modules for rational verification. *Artificial Intelligence*, 248(Supplement C):123 – 157, 2017.

[9] Julian Gutierrez, Paul Harrenstein, and Michael Wooldridge. Reasoning about Equilibria in Game-like Concurrent Systems. *Annals of Pure and Applied Logic*, 168(2):373–403, 2017.

[10] Julian Gutierrez, Muhammad Najib, Giuseppe Perelli, and Michael Wooldridge. Eve: A tool for temporal equilibrium analysis. In Shuvendu K. Lahiri and Chao Wang, editors, *Automated Technology for Verification and Analysis*, pages 551–557, Cham, 2018. Springer International Publishing.

[11] Julian Gutierrez, Muhammad Najib, Giuseppe Perelli, and Michael Wooldridge. Equilibrium design for concurrent games. In *CONCUR*, 2019. To Appear.

[12] Julian Gutierrez, Muhammad Najib, Giuseppe Perelli, and Michael Wooldridge. On computational tractability for rational verification. In *IJCAI*, 2019. To Appear.

[13] Orna Kupferman, Giuseppe Perelli, and Moshe Vardi. Synthesis with Rational Environments. *Annals of Mathematics and Artificial Intelligence*, 78(1):3–20, 2016.

[14] M. Wooldridge and U. Endriss and S. Kraus and J. Lang. Incentive engineering for Boolean games. *Artificial Intelligence*, 195:418 – 439, 2013.

[15] M.J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[16] W. van der Hoek, A. Lomuscio, and M. Wooldridge. On the complexity of practical ATL model checking. 2005.

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

[18] Michael Wooldridge, Julian Gutierrez, Paul Harrenstein, Enrico Marchioni, Giuseppe Perelli, and Alexis Toumi. Rational verification: From model checking to equilibrium checking. In *AAAI Conference on Artificial Intelligence*, Phoenix, Arizona, USA, 2016.