Formal verification with games on graphs

Andrea Manini[1]

Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Milan, 20133, Italy andrea.manini@polimi.it

Abstract. When checking the correctness of software systems, games on graphs have emerged as a viable alternative or supplementary tool to other more renowned and well-established techniques, such as model checking. In this PhD research, the application of games on graphs (and game theory in general) to formal verification and testing of software systems is studied from a theoretical and practical perspective. An extensive review of the existing literature has identified a significant connection between a particular class of games, known as parity games, and the domain of formal verification. This observation underpins the relevance and potential impact of further exploring this area. Given that this PhD research is still in its early stages, this paper aims to present the findings related to parity games that have been discovered thus far, adding insights and preliminary directions on how the research will proceed.

Keywords: game theory · formal verification · testing · model checking.

1 Motivations and Goal

Modern software systems offer exceptional and highly advanced functionalities. Our society and our life is nowadays entirely based on digital devices and, for this reason, the behavioral correctness of these systems must be guaranteed in every phase of their development. Numerous automated techniques, such as software testing [19] and model checking [6], have been developed to achieve this goal.
It may not be widely known that the correctness of software systems can be verified through games. The term *games* here should not be intended in its broadest sense; rather, the types of games leveraged in this context are mathematical constructs that intersect with automata theory and logic.
This PhD research aims at enhancing the current State of the Art of games used in formal verification and testing of software systems. This involves reviewing the current and past literature and finding possible pitfalls or ways to enhance the most recent findings.
Since, at the current time of writing, this PhD research is still in its early stages, the following sections will provide only a description of what has been discovered thus far, along with some hints on how the work will proceed.

2 Groundings

Games used in formal verification and software testing have strong theoretical foundations in mathematics, logic, and automata theory [11]. In particular, these

games differ from those in classical game theory since they are played on a graph acting as an arena between two players. During a play, both players move a fictitious token between different vertexes of the arena. Based on particular winning conditions, the main goal is to determine who is going to be the winner given an initial starting vertex of the arena.

An exhaustive and recent classification of the most common classes of games on graphs can be found in [10], e.g., reachability games, parity games, games with payoffs, stochastic games, and timed games.

The most intriguing and worth investigating class of games identified to date seems to be parity games. They have a deep connection with modal μ-calculus [7] and, for this reason, are tightly connected to the formal verification domain.

Concerning some practical applications, it has been proven that solving parity games is polynomial-time equivalent to μ-calculus model checking [9] and to the emptiness problem for nondeterministic automata on infinite trees with parity acceptance conditions. It has also been proven that parity games can be transformed in polynomial time into different classes of games, e.g., into mean-payoff games or simple stochastic games. Whether transformations in the opposite direction exist is currently not known.

Furthermore, parity games have also been used in the synthesis of controllers for discrete event systems [5], and for automatically extracting temporal interface specifications for Java classes [4].

From a more theoretical perspective, solving parity games is one of the few known problems to be both in NP $\cap$ coNP [9] and UP $\cap$ coUP [12]. Despite belonging to these complexity classes, the existence of a polynomial-time algorithm for solving parity games is believed to be feasible. Still, none has been found after decades of research.

Several algorithms have been developed to solve parity games. Oink [1] is a C++ tool implementing a wide range of algorithms for solving parity games. Experimental results on time and resources used by these implementations are summarized in [8]. It turns out that Zielonka's algorithm [20], despite having an exponential theoretical complexity, outperforms other quasi-polynomial time algorithms, being the fastest in practice. For this reason, it is one of the most studied and enhanced algorithms for solving parity games (please refer to [16], [17], and [3] for some examples).

As of our best knowledge, the only other relevant tool for solving parity games is PGSolver [2]. To be precise, PGSolver is not properly a tool but a library written in OCaml collecting algorithms for solving parity games.

Of course, parity games and games on graphs are not the only examples of game theory application to formal verification and testing. For instance, [15] presents an incomplete framework for transforming mutation testing into a so-called *mutation game* and solving it by finding mixed-strategy Nash equilibria.

3 Contribution

The contribution of this PhD research will consider all the findings reported in the previous section and the ones yet to be discovered. Due to the popularity of parity games, it looks promising to start from this class of games.

Modern computing infrastructures now rely on heterogeneous architectures. From a performance perspective, the energy efficiency wall has been reached, making Moore's law practically ineffective [18]. Consequently, to achieve performance improvements, parallel and multi-core solutions must be employed.
The main theme of this PhD research will be the study and implementation of current (and potentially new) games on graphs algorithms using modern parallel programming paradigms and architectures, with a particular focus on parity games. As viable parallel programming candidates, it is worth mentioning pthread, openMP, MPI (Message Passing Interface), and GPU (Graphics Processing Unit) computing, e.g., as it has been done in [3].
Since parity games also have practical applications, it would be interesting to discover new domains in which they can be beneficial. Another (albeit ambitious) goal could be to refine the complexity class in which parity games belong.
Artifacts would also be produced during this PhD research. A new tool (or framework) for solving parity games will certainly be developed. This tool will feature detailed and coherent documentation, along with a user-friendly interface to ensure ease of use.
This PhD research will not solely focus on parity games or games in general. Other problems that may arise or will be discovered in the formal verification or testing domain could also be tackled.

As a final note, this PhD research aims at maximizing engagement and interest of future readers. Contribution to (completely) unrelated domains from formal verification and testing could also be pursued. Talking about real games, this could entail achieving outcomes akin to those proven in [13] and [14], where the classical Windows game Minesweeper has been proven to be both NP-complete and Turing-complete.

4 Research Plan Status

At the current time of writing, this PhD research is focusing on a literature review of automata theory, logic, software testing, formal verification, and games on graphs. The next step will be the analysis of suitable parallel programming paradigms and architectures on which to implement algorithms for solving parity games. The following is a tentative definition of the whole PhD research plan.
The first year will be devoted to reviewing existing literature, possibly finding new algorithms for solving parity games (or other games on graphs), and finding suitable architectures for implementing such algorithms. An initial development of the tool or framework could also be initiated.

The second year will encompass the development of a parity games solver or a framework for solving parity games using parallel programming paradigms and architectures. A comprehensive literature review will also be conducted to ensure up-to-date findings are integrated and to gather innovative ideas.
The third year will focus on completing the aforementioned tool or framework (or even some more theoretical results) while completing the thesis writing.

References

1. https://github.com/trolando/oink
2. https://www.uni-kassel.de/eecs/en/tifm/software/pgsolver
3. Aggarwal, S., de la Banda, A.S., Yang, L., Gutierrez, J.: A matrix-based approach to parity games. In: Tools and Algorithms for the Construction and Analysis of Systems: 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France, April 22–27, 2023, Proceedings, Part I. pp. 666–683. Springer-Verlag, Berlin, Heidelberg (2023)
4. Alur, R., Černý, P., Madhusudan, P., Nam, W.: Synthesis of interface specifications for java classes. SIGPLAN Not. **40**(1), 98–109 (jan 2005)
5. Arnold, A., Vincent, A., Walukiewicz, I.: Games for synthesis of controllers with partial observation. Theoretical Computer Science **303**(1), 7–34 (2003)
6. Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press (2008)
7. Blackburn, P., van Benthem, J., Wolter, F. (eds.): Handbook of Modal Logic. Elsevier (2006)
8. van Dijk, T.: Oink: An Implementation and Evaluation of Modern Parity Game Solvers, pp. 291–308. Springer International Publishing (2018)
9. Emerson, E.A., Jutla, C.S., Sistla, A.P.: On model checking for the μ-calculus and its fragments. Theoretical Computer Science **258**(1), 491–522 (2001)
10. Fijalkow, N., Bertrand, N., Bouyer-Decitre, P., Brenguier, R., Carayol, A., Fearnley, J., Gimbert, H., Horn, F., Ibsen-Jensen, R., Markey, N., Monmege, B., Novotný, P., Randour, M., Sankur, O., Schmitz, S., Serre, O., Skomra, M.: Games on graphs (2023)
11. Grädel, E., Thomas, W., Wilke, T. (eds.): Automata logics, and infinite games: a guide to current research. Springer-Verlag, Berlin, Heidelberg (2002)
12. Jurdziński, M.: Deciding the winner in parity games is in up ∩ co-up. Information Processing Letters **68**(3), 119–124 (1998)
13. Kaye, R.: Minesweeper is np-complete. The Mathematical Intelligencer **22** (2000)
14. Kaye, R.: Infinite versions of minesweeper are turing complete (2007), https://api.semanticscholar.org/CorpusID:126129473
15. Lee, J.: Game Theoretic Framework to Mutation Testing, pp. 159–164 (01 2020)
16. Liu, Y., Duan, Z., Tian, C.: An improved recursive algorithm for parity games. In: 2014 Theoretical Aspects of Software Engineering Conference. pp. 154–161 (2014)
17. Parys, P.: Parity games: Zielonka's algorithm in quasi-polynomial time (2019)
18. Rupp, K.: (Feb 2018), https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/
19. Young, M., Pezze, M.: Software Testing and Analysis: Process, Principles and Techniques. John Wiley & Sons, Inc., Hoboken, NJ, USA (2005)
20. Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theoretical Computer Science **200**(1), 135–183 (1998)