Formal Verification using Parity Games

Mathias Nygaard Justesen

Technical University of Denmark, Kongens Lyngby, Denmark
s123152@student.dtu.dk

While many problems can be reduced to solving parity games, see [FL10] for example, verification frameworks using parity game solvers as a backend technology seem quite unexplored. In this abstract we report an initial attempt at building an infrastructure for a verification framework, which so far captures model checking for the modal μ-calculus.

At least two toolsets, mCRL2 and LTSmin, reduce the model-checking problem to parity game solving, but they both do so by encoding the problem as a parameterized boolean equation system (PBES) [CGK+13, KvdP14]. We take a more direct, game-based approach, first proposed by Colin Stirling [Sti96, BS06], where the problem is not encoded as a PBES, but instead the system is modelled as a labelled transition system and the properties are specified as a formula in the modal μ-calculus. This is an interesting logic, because it subsumes other widely used modal logics, such as LTL and CTL [Koz83].

1 Basic Notions

Parity games. A parity game is played by two players, called Player 0 and Player 1, on a directed graph in which all nodes are labelled with priorities. Formally, a *parity game* is a tuple $G = (V, V_0, V_1, E, \Omega)$, where (V, E) forms a directed, total graph. Player 0 controls the nodes in V_0, and Player 1 controls the nodes in V_1, such that $V = V_0 \cup V_1$ and $V_0 \cap V_1 = \emptyset$. The priority function $\Omega : V \to \mathbb{N}$ assigns a natural number to each node, called the *priority* of the node.

The game starts in a node $v_0 \in V$ and an infinite sequence of nodes is constructed as follows. If the play so far has yielded a finite sequence of nodes $v_0 v_1 \dots v_i$ and v_i is in V_j, then Player j selects a node w, such that $(v, w) \in E$, and the play continues with the sequence $v_0 v_1 \dots v_i w$. The winner of the play $v_0 v_1 v_2 \dots$ is Player 0, if the highest priority that occurs infinitely often is even, otherwise Player 1 wins.

A *strategy* for Player j is a function $\sigma : V^* V_j \to V$ that maps every initial play $v_0 v_1 \dots v_i$ ending a node $v_i \in V_j$ to a successor node v_{i+1}, such that $(v_i, v_{i+1}) \in E$.

An important property of parity games is that they exhibit *positional determinacy* [Kü02], i.e., the set V of nodes in an arbitrary game G can be divided into two *winning regions*, W_0 and W_1, such that Player j can win every game that starts in a node $v \in W_j$ by following a *winning strategy*. Moreover, a winning strategy is also positional, i.e., there exists a function $\sigma' : V_j \to V$ such that $\sigma(v_0 \dots v_i) = \sigma'(v_i)$ for $v_i \in V_j$. Thus, an algorithm for solving parity games should compute the winning regions and the positional winning strategies.

Labelled transition systems. A (P, A)-*labelled transition system* (LTS) is a tuple $\mathbb{S} = (S, V, R)$, such that S is the set of states; $V : P \to \mathcal{P}(S)$ is a *valuation*, i.e., $V(p)$ is the set of states where p is true; and $R = \{R_a \subseteq S \times S \mid a \in A\}$ contains the action-labelled relations of the system.

Modal μ-calculus. Given a set of proposition letters P and a set of actions A, the collection of formulas φ in modal μ-calculus are defined by the following grammar

$$\varphi ::= \top \mid \bot \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle a \rangle \varphi \mid [a]\varphi \mid \mu x.\varphi \mid \nu x.\varphi$$

where $p, x \in P$ and $a \in A$. Informally, $\top$ and $\bot$ are true and false, respectively; $\neg$, $\wedge$, and $\vee$ are the usual Boolean operators; $\langle a \rangle$ and $[a]$ are *modal operators*; and μ and ν are *fixpoint operators*.

The semantics of the modal μ-calculus can be defined game-theoretically in terms of a so-called *evaluation game* [Ven08]. It is helpful to think of the game in terms of proving or disproving a formula φ. Player 0 is trying to prove φ, and Player 1 is trying to disprove it. The problem of determinining the winner of an evaluation game can be reduced to finding the winning regions of a parity game. Furthermore, the model checker can construct examples and counter-examples from the winning strategies.

2 The Framework

The overall goal is to investigate the use of parity games for formal verification. Thus, an efficient solver is a critical part of a verification framework. Therefore, in the first version we use PGSolver [FL14] as the backend solver, because it allows for experimentation with different algorithms. Furthermore, a model checker for modal μ-calculus is implemented. This implementation is based on an on-the-fly generation of a parity game on the basis of a labelled transition system and a formula in modal μ-calculus.

Parity games of a few million nodes with average degree $d = 3$ can be solved in less than a minute[1], so speed is not an immediate concern. However, memory is quickly exhausted as the LTS or the formula becomes complicated, since the game graph consists of $O(|S| \cdot |Sfor(\varphi)|)$ nodes.[2] One approach to overcome this problem is solving parity games symbolically, which leads to a more compact representation [BEKR09, KvdP14]. For a simple one-node LTS with three actions, the current implementation can check formulas with eleven alternating fixpoints in a few seconds, by generating and solving a parity game of 1.2 million nodes. For such formulas the state space grows exponentially with the depth of the formula, so a formula with twelve alternating fixpoints results in a parity game of 3.7 million nodes, which exhausts the memory of the parity game solver.

We have also considered a new type of algorithm due to Steen Vester, in which parity games are considered in a certain normal form, where the game is strictly turn-based, and where Player 0 only controls nodes of even parity and Player 1 only controls nodes of odd parity. The algorithms in consideration exploit these restrictions in order to simplify the solving process. Considering parity games in this normal form is viable, because it is possible to transform any parity game to one in normal form in linear time, such that no player has any additional strategic advantages and the winning regions are preserved. In practice, the normal-form algorithms perform worse than the state-of-the-art algorithm due to Zielonka [Zie98, FL09], but they perform well on games that they are theoretically well suited for, i.e., dense game graphs that are already in normal form (need no transformation). This is promising for the development of other specialized algorithms. These may not perform better in general, but in special cases of interest, e.g., an algorithm that exploits the tree-like structure of parity games constructed from evaluation games.

Overall, this is promising for using parity games for other verification techniques, e.g., controller synthesis, which amounts to finding the winning strategies in a parity game [RW89].

[1] Benchmarks carried out on a machine with four 3.5 GHz Intel Core i5 processors and 8 GB RAM space. The implementation does not support prallel computations, hence, the benchmarks was only run on one processor.

[2] $Sfor(\varphi)$ denotes the subformulas of φ.

References

[BEKR09] Marco Bakera, Stefan Edelkamp, Peter Kissmann, and Clemens D. Renner. Solving μ-calculus parity games by symbolic planning. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Lect. Notes Comput. Sci*, 5348:15–33, 2009.

[BS06] Julian Bradfield and Colin Stirling. Modal mu-calculi. *Handbook of modal logic*, 3:721 – 756, 2006.

[CGK+13] Sjoerd Cranen, Jan Friso Groote, Jeroen JA Keiren, Frank PM Stappers, Erik P de Vink, Wieger Wesselink, and Tim AC Willemse. An overview of the mCRL2 toolset and its recent advances. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 199–213. Springer, 2013.

[FL09] Oliver Friedmann and Martin Lange. Solving parity games in practice. In Zhiming Liu and Anders P. Ravn, editors, *Automated Technology for Verification and Analysis*, volume 5799 of *Lecture Notes in Computer Science*, pages 182–196. Springer Berlin Heidelberg, 2009.

[FL10] Oliver Friedmann and Martin Lange. Local strategy improvement for parity game solving. *arXiv preprint arXiv:1006.1409*, 2010.

[FL14] Oliver Friedmann and Martin Lange. The PGSolver collection of parity game solvers — version 3. URL: `https://github.com/tcsprojects/pgsolver/raw/master/doc/pgsolver.pdf`, 2014.

[Koz83] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333 – 354, 1983. Special Issue Ninth International Colloquium on Automata, Languages and Programming (ICALP) Aarhus, Summer 1982.

[KvdP14] Gijs Kant and Jaco van de Pol. Generating and solving symbolic parity games. In *Proceedings 3rd Workshop on GRAPH Inspection and Traversal Engineering, GRAPHITE 2014, Grenoble, France, 5th April 2014.*, pages 2–14, 2014.

[Kü02] Ralf Küsters. Memoryless determinacy of parity games. In Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors, *Automata Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*, pages 95–106. Springer Berlin Heidelberg, 2002.

[RW89] P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, Jan 1989.

[Sti96] Colin Stirling. *Modal and temporal logics for processes*. Springer, 1996.

[Ven08] Yde Venema. Lectures on the modal μ-calculus. *Institute for Logic, Language and Computation, University of Amsterdam*, 2008.

[Zie98] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1–2):135 – 183, 1998.