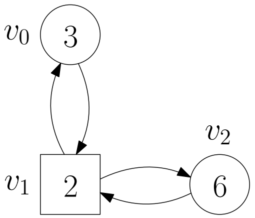
Partial solvers for generalized parity games

Véronique Bruyère (UMONS)
Jean-François Raskin (ULB)
Guillermo A. Pérez (UAntwerp)
Clément Tamines (UMONS)

September 18, 2019
Highlights '19
Warsaw, Poland

Game structure and play

Game structure $G = (V_0, V_1, E)$ and **priority function** $\alpha : V \to \{0, 1, \ldots, d\}$



Play: infinite path in G starting in v_0
→ sequence of priorities corresponding to π

Example:

$$\pi = v_0 \; v_1 \; v_2 \; (v_1 \; v_0)^\omega$$
$$\alpha(\pi) = 3 \; 2 \; 6 \; (2 \; 3)^\omega$$

→ play won by player 1

Objective for player $i \in \{0, 1\}$: set of plays $\Omega_i \subseteq V^\omega$

- $\Omega_0 = 0\mathsf{Parity}(\alpha)$: plays where maximum priority seen infinitely often is even
- $\Omega_1 = 1\mathsf{Parity}(\alpha)$: plays where maximum priority seen infinitely often is odd

Strategies: memoryless strategies required for both players to win

Complexity: NP ∩ coNP [2] and UP ∩ coUP [6], P membership is open problem

Reactive synthesis from LTL specifications

Parity games are intermediary steps in reactive synthesis from LTL specifications [7]

Goal : given an environment, synthetise a system that respects specifications (LTL)

Input : an LTL formula ϕ whose propositional variables are partitioned into inputs (controllable by the environment) and outputs (controlled by the system)

Classical algorithm :

- construct a deterministic parity automaton (DPA)
- the DPA can then be seen as a two player graph game
- winning condition in this game : parity acceptance condition of the DPA

Particularity: LTL formula $\phi = \phi_1 \wedge \cdots \wedge \phi_n$ is a conjunction of smaller formulas

- compositional approach: construct a DPA A_i for each subformula ϕ_i
- underlying game is then the product of the automata A_i and the winning condition is a conjunction (for Player 0) of parity conditions

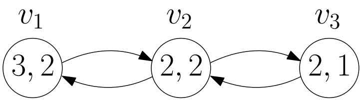
→ such games are generalized parity games

Generalized parity objective

Now consider $k \geq 1$ priority functions $\alpha_\ell : V \to \{0, \ldots, d_\ell\} \forall \ell \in \{1, \ldots, k\}$
$\to$ we consider several sequences of priorities corresponding to a play

Objective $\Omega_0 = \text{ConjEvenParity}(\alpha_1, \ldots, \alpha_k)$, conjunction of parity objectives
$\to$ maximum priority seen infinitely often according to every function is even

Opposite $\Omega_1 = \text{DisjOddParity}(\alpha_1, \ldots, \alpha_k)$, disjunction of parity objective
$\to$ maximum priority seen infinitely often according to some function is odd



$$\pi = v_1 \; (v_2 \; v_3)^\omega$$
$$\alpha_1(\pi) = 3 \; (2 \;\; 2)^\omega$$
$$\alpha_2(\pi) = 2 \; (2 \;\; 1)^\omega$$

Strategies: player 0 requires a finite-memory strategy to win

Complexity: coNP $-$ *complete* [1]

Partial solvers

Algorithms used to solve (generalized) parity games have exponential complexity

- introduction of incomplete algorithms that partially solve parity games
- benefit: partial solvers are polynomial-time algorithms [3, 4, 8]
- experimentally shown to behave well on random benchmarks and completely solve structured benchmarks from PGSolver [3]

Formally, a partial solver returns

- subsets of winning nodes $Z_0 \subseteq \mathrm{Win}(G, 0, \Omega_0)$ and $Z_1 \subseteq \mathrm{Win}(G, 1, \Omega_1)$
- a sub-game $G \setminus (Z_0 \cup Z_1)$ that is unsolved

Our contributions

- extend three partial solvers for parity games to generalized parity games
- combine partial solvers with recursive complete algorithms [9, 1]
- evaluate on benchmarks generated from LTL formulas

Let us now give intuitions on how two of them work !

Partial solver using Büchi games

A first very simple partial solver based on Büchi and safety objectives

Based on a simple remark for player 0 (symmetric for player 1)

- if player 0 can ensure to visit infinitely often an even priority
- without visiting a greater odd priority
- then he is winning for $0\mathrm{Parity}(\alpha)$

Given $p \in \{0, \ldots, d\}$ an even priority, we want to

- visit $U = \{v \in V \mid \alpha(v) = p\}$ infinitely often: $\mathrm{Buchi}(U)$
- avoid $U' = \{v \in V \mid \alpha(v)$ is an odd priority and $\alpha(v) > p\}$: $\mathrm{Safe}(U')$

Conjunction of Büchi and safety objectives

If $v \in \mathrm{Win}(G, 0, \mathrm{Buchi}(U) \cap \mathrm{Safe}(U'))$, then $v \in \mathrm{Win}(G, 0, 0\mathrm{Parity}(\alpha))$.

Variant : property holds with $\mathrm{CoBuchi}(U')$ instead of $\mathrm{Safe}(U')$

Partial solver using generalized Büchi games

Adaptation to generalized parity games based on generalized Büchi objective

Player 0 with the conjunction of parity objectives, on all dimensions ℓ he wants

- to ensure to visit infinitely often an even priority p_ℓ
- without visiting an odd priority greater than p_ℓ
- if he does, he is winning for $\mathsf{ConjEvenParity}(\alpha_1, \ldots, \alpha_k)$

→ this is Generalized Büchi objective using k sets $U_\ell = \{v \in V \mid \alpha_\ell(v) = p_\ell\}$

Player 1 with the disjunction of parity objectives, on some dimensions ℓ

- if he ensures $\mathsf{1Parity}(\alpha_\ell)$, he satisfies the disjunction

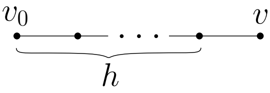
→ can apply same property as before dimension by dimension

Partial solver : Good Episode

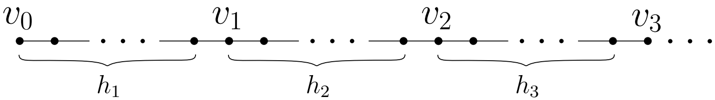
GoodEpSolver [4] for parity games : remark for player 0 (symmetric for player 1)

- if player 0 can ensure to visit some node v from v_0 : hv
- such that the maximum visited priority in h is even

→ h is a good episode



If a play consists in a succession of good episodes, it is won by player 0



Maximum priority in each h_j is even → objective $0\mathsf{Parity}(\alpha)$ is satisfied

To compute these good episodes, introduce an extended game structure $G \times M$
→ records the maximal visited priority by construction

- compute a fixpoint $\mathsf{GoodEp}_0(G, \alpha)$ based on attractor computations
- from the fixpoint, player 0 can ensure a succession of good episodes

Generalized Good episodes & other contributions

The generalized version computes a specific fixpoint depending on the player

- player 0: extended game structure $G \times M_1 \times \ldots \times M_k$
- player 1: previous approach : $\mathsf{GoodEp}_1(G, \alpha_\ell)$ for some dimension ℓ

Explicit construction and attractor computation in extended game $G \times M$ is costly
$\longrightarrow$ work symbolically using antichains-based algorithm to reduce required space

Additional contributions:

- we adapted a third partial solver [3] to generalized parity games
- partials solvers can be combined with the recursive algorithm for (generalized) parity games [9, 1] to obtain complete solvers
- implemented 6 algorithms and their combination with recursive algorithms
- evaluated on benchmarks generated from meaningful LTL specifications [5]

Results for parity games

240 benchmarks with mean size $|V|$ of around 46K and maximal size of 3157K, and a mean number d of priorities of 4.1 with a maximal number $d = 15$

Solver	Solved	T.O.	Fastest	Mean time (233)
Zielonka	240 (100%)	0	150 (62 %)	272 ms
Ziel&BuchiSolver	240 (100%)	0	89 (37 %)	480 ms
Ziel&GoodEpSolver	233 (97%)	7	0 (0%)	1272 ms
Ziel&LaySolver	238 (99%)	2	1 (1%)	587 ms
BuchiSolver	203 (84%) - 37	0	-	-
GoodEpSolver	233 (97%) - 0	7	-	-
LaySolver	232 (97%) - 6	2	-	-

- recursive algorithm is faster than partial solvers on average which was not observed on random graphs in [4]
- the combination of partial solvers with the recursive algorithm improves its performances in 90 cases over 240 (38%)

Results for generalized parity games

152 benchmarks with mean size $|V|$ of around 207K and maximal size of 709K, mean number of priority functions is 4.53 and maximum number is 17.

Solver	Solved	T.O.	Fastest	Mean-Time (87)
GenZielonka	128 (84%)	24	33 (25%)	66 ms
GenZiel&GenBuchiSolver	130 (86%)	22	72 (55%)	56 ms
GenZiel&GenGoodEpSolver	112 (74%)	40	24 (18%)	644 ms
GenZiel&GenLaySolver	110 (72%)	42	3 (2%)	1133 ms
GenBuchiSolver	110 (72%) - 20	22	-	-
GenGoodEpSolver	112 (74%) - 0	40	-	-
GenLaySolver	104 (68%) - 6	42	-	-

- some benchmarks can't be solved by generalized recursive algorithm or a partial solver alone, but can be solved by the combination of the generalized recursive algorithm with a partial solver
- the combination of partial solvers with the generalized recursive algorithm improves its performances in 99 cases over 132 (75%)

[1] K. Chatterjee, T. A. Henzinger, and N. Piterman.
Generalized parity games.
In *FOSSACS Proceedings*, volume 4423 of *Lecture Notes in Computer Science*, pages 153–167. Springer, 2007.

[2] E. A. Emerson and C. S. Jutla.
Tree automata, mu-calculus and determinacy (extended abstract).
In *FOCS Proceedings*, pages 368–377. IEEE Computer Society, 1991.

[3] M. Huth, J. H. Kuo, and N. Piterman.
Fatal attractors in parity games: Building blocks for partial solvers.
CoRR, abs/1405.0386, 2014.

[4] M. Huth, J. H. Kuo, and N. Piterman.
Static analysis of parity games: Alternating reachability under parity.
In *Semantics, Logics, and Calculi - Essays Dedicated to Hanne Riis Nielson and Flemming Nielson on the Occasion of Their 60th Birthdays*, volume 9560 of *Lecture Notes in Computer Science*, pages 159–177. Springer, 2016.

[5] S. Jacobs, R. Bloem, M. Colange, P. Faymonville, B. Finkbeiner, A. Khalimov, F. Klein, M. Luttenberger, P. J. Meyer, T. Michaud, M. Sakr, S. Sickert, L. Tentrup, and A. Walker.

The 5th reactive synthesis competition (SYNTCOMP 2018): Benchmarks, participants & results.

CoRR, abs/1904.07736, 2019.

[6] M. Jurdzinski.

Deciding the winner in parity games is in UP ∩ co-UP.

Inf. Process. Lett., 68(3):119–124, 1998.

[7] A. Pnueli and R. Rosner.

On the synthesis of a reactive module.

In *POPL Proceedings*, pages 179–190. ACM Press, 1989.

[8] S. Vester.

Winning cores in parity games.

In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 662–671, 2016.

[9] W. Zielonka.

Infinite games on finitely coloured graphs with applications to automata on infinite trees.

Theor. Comput. Sci., 200(1-2):135–183, 1998.