Graph Operations on Parity Games and Polynomial-Time Algorithms

Christoph Dittmann, Stephan Kreutzer, Alexandru I. Tomescu[1]

Chair for Logic and Semantics, Technical University Berlin
christoph.dittmann@tu-berlin.de, stephan.kreutzer@tu-berlin.de,
alexandru.tomescu@mailbox.tu-berlin.de

September 19, 2012

[1] The third author gratefully acknowledges the support of the European Science Foundation, activity "Games for Design and Verification".

Parity Games

- Parity games are polynomial-time equivalent to the model-checking problem of the modal μ-calculus.
- Computing the winner of a parity game is in NP and coNP.

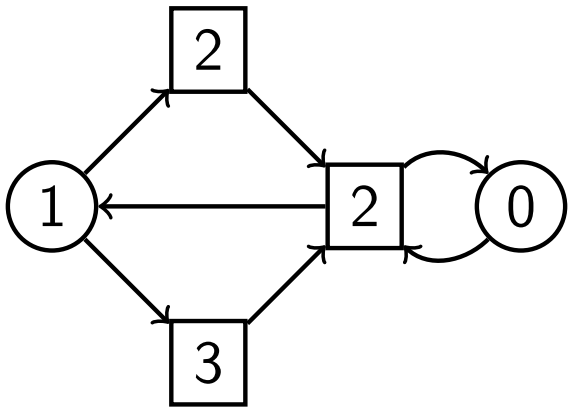
Parity Games

- Parity games are polynomial-time equivalent to the model-checking problem of the modal μ-calculus.
- Computing the winner of a parity game is in NP and coNP.
- Open problem: Is there a polynomial time algorithm for computing the winner of a parity game?

Parity Games

Definition 1

A *parity game* $P = (V, V_\bigcirc, V_\Box, E, \Omega)$ is a directed graph (V, E) with a partitioning of the nodes $V = V_\bigcirc \cup V_\Box$ and a priority function $\Omega : V \to \mathbb{N}$.

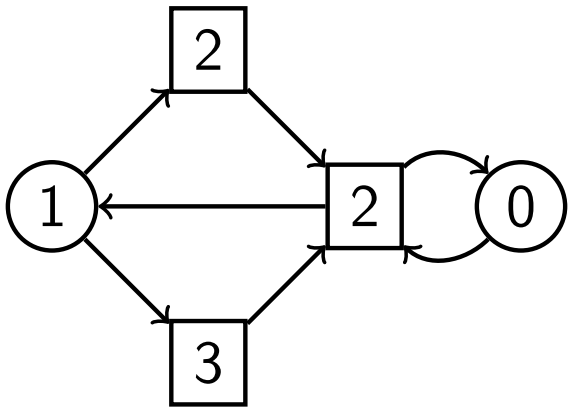


Parity Games

In a *play*, the players push a token along the edges of the graph. The sets $V_\bigcirc$, $V_\Box$ determine whose turn it is.

Definition 2

A *play* is a maximal sequence of nodes $v_1, v_2, \ldots$ such that $Ev_i v_{i+1}$ for all i. A play is *winning for Player* $\circ$ if the maximum priority that appears infinitely often is even or if the last node is in $V_\Box$.



Positional Determinacy

Theorem 3 (see e.g., [GTW02])

Parity games are positionally determined.

This implies that

$$V(P) = W_{\bigcirc}(P) \cup W_{\Box}(P),$$

where $W_i(P)$ is the *winning region* of player i, that is, the set of all nodes from which player i has a positional winning strategy.

Positional Determinacy

Theorem 3 (see e.g., [GTW02])

Parity games are positionally determined.

This implies that

$$V(P) = W_{\bigcirc}(P) \cup W_{\Box}(P),$$

where $W_i(P)$ is the *winning region* of player i, that is, the set of all nodes from which player i has a positional winning strategy.
Solving parity games is in NP and coNP.

Hereditary Classes

Definition 4

A class of graphs is *hereditary* if it is closed under induced subgraphs.

Hereditary Classes

Definition 4

A class of graphs is *hereditary* if it is closed under induced subgraphs.

Let $\mathcal{C}, \mathcal{C}'$ be hereditary classes of parity games solvable in polynomial time. Then the *join* of $\mathcal{C}$ and $\mathcal{C}'$ can be solved in polynomial time.

Reducing the Problem

- Solving a game means computing its winning regions.
- Idea: Reduce the problem of solving P to one of the problems:
  - A proper subgame $P' \subsetneq P$ and the winning regions for $P \setminus P'$, such that the winning regions of P' and P agree.
  - "One winning region of P is empty".
- This works according to the following lemma.

Half-Solving Parity Games

Lemma 5

Let $\mathcal{C}$ be a hereditary class of parity games and assume that there is an algorithm on $\mathcal{C}$ that runs in time $O(n^c)$ for some $c > 0$ and for each game $P \in \mathcal{C}$ returns one of the two following results.

1. *A proper subgame P' and $W_\bigcirc^*, W_\Box^* \subseteq V(P) \setminus V(P')$ with*

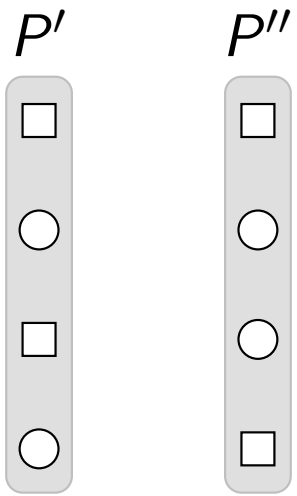
$$W_\bigcirc(P) = W_\bigcirc^* \cup W_\bigcirc(P'),$$
$$W_\Box(P) = W_\Box^* \cup W_\Box(P').$$

2. *"$W_\bigcirc(P) = \emptyset$ or $W_\Box(P) = \emptyset$".*

Then $\mathcal{C}$ can be solved in time $O(n^{c+1})$.

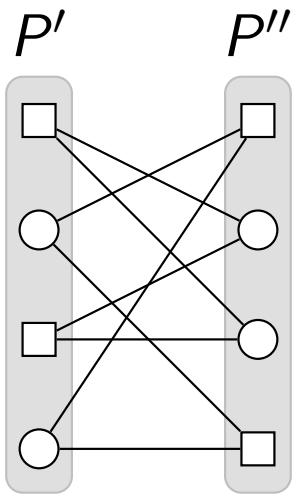
Joins

A join of two games adds an arc between all pairs of nodes of different players.



Joins

A join of two games adds an arc between all pairs of nodes of different players.



The lines represent required edges (in at least one direction).

Joins

Definition 6

Let $\mathcal{C}$, $\mathcal{C}'$ be two classes of parity games.

$$\text{Join}(\mathcal{C},\mathcal{C}') := \{P \mid P \text{ is a join of } P' \in \mathcal{C} \text{ and } P'' \in \mathcal{C}'\}$$

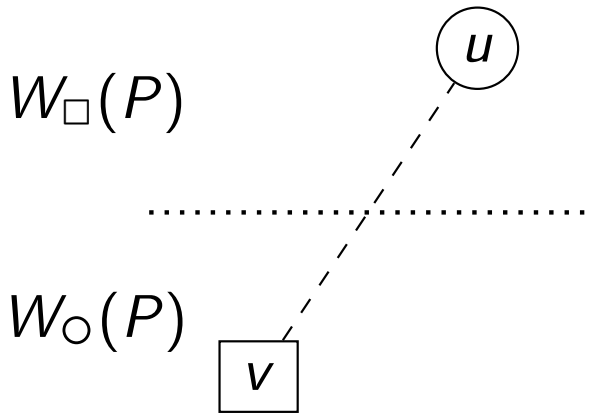
$$\text{HalfJoin}(\mathcal{C}) := \{P \mid P \text{ is a join of a single-player game } P' \text{ and a game } P'' \in \mathcal{C}\}$$

Theorem 7

If $\mathcal{C}$, $\mathcal{C}'$ are hereditary classes of parity games that can be solved in polynomial time, then all games $P \in \text{HalfJoin}(\mathcal{C})$ (in $\text{Join}(\mathcal{C},\mathcal{C}')$, resp.) can be solved in polynomial time, provided that a decomposition of P as a join is given.

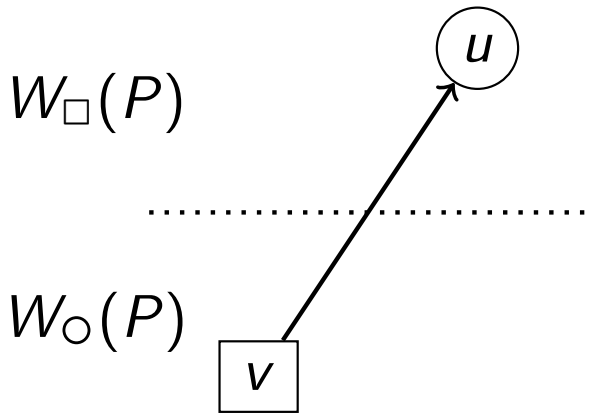
Proof for Join

Let P be a game and $u \in W_{\Box}(P)$ and $v \in W_{\bigcirc}(P)$.



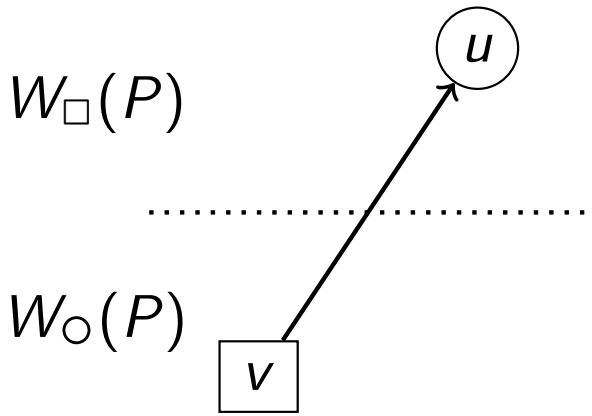
Proof for Join

Let P be a game and $u \in W_{\Box}(P)$ and $v \in W_{\bigcirc}(P)$.



Proof for Join

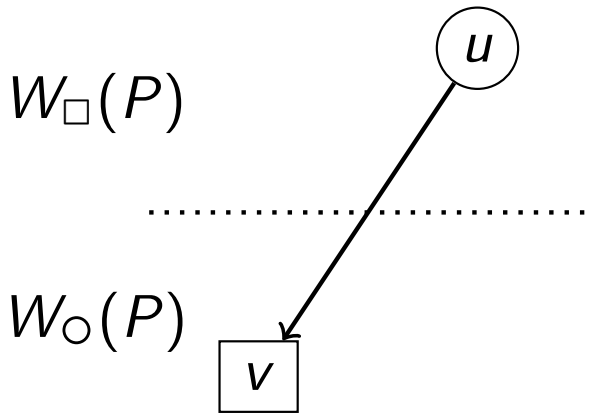
Let P be a game and $u \in W_{\Box}(P)$ and $v \in W_{\bigcirc}(P)$.



Impossible because
Player $\Box$ would escape to u

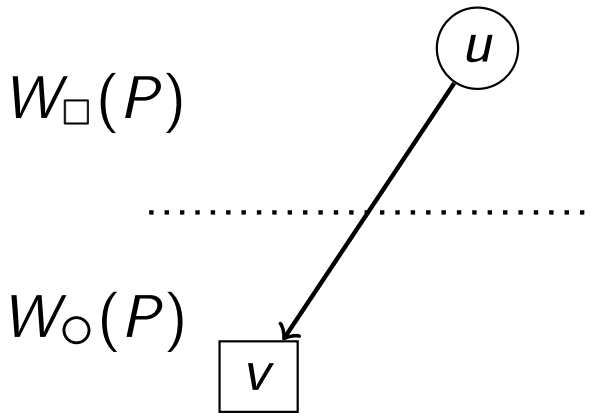
Proof for Join

Let P be a game and $u \in W_{\Box}(P)$ and $v \in W_{\bigcirc}(P)$.



Proof for Join

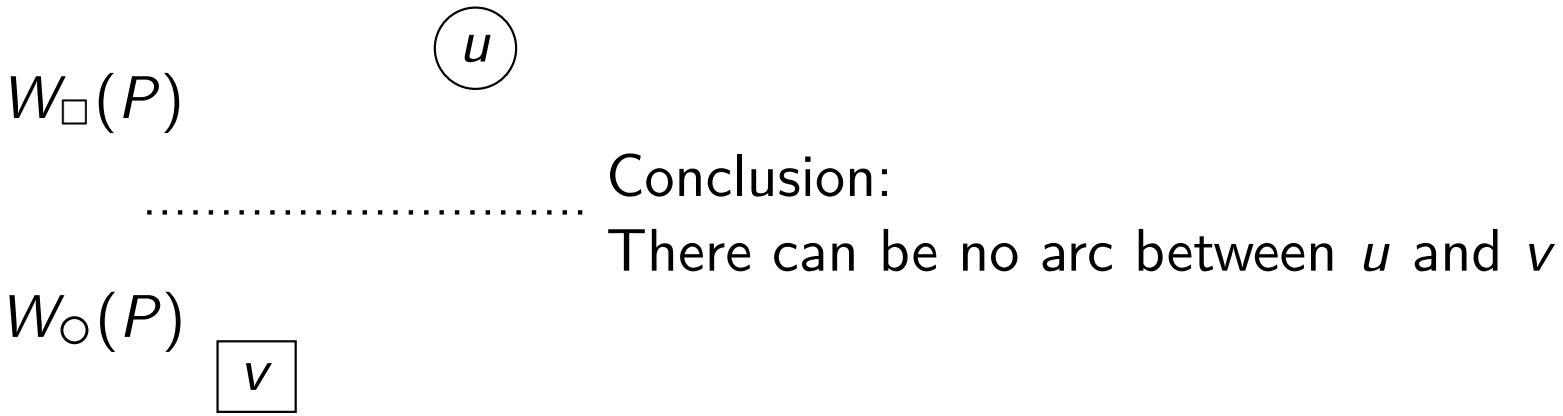
Let P be a game and $u \in W_\Box(P)$ and $v \in W_\bigcirc(P)$.



Impossible because
Player $\bigcirc$ would escape to v

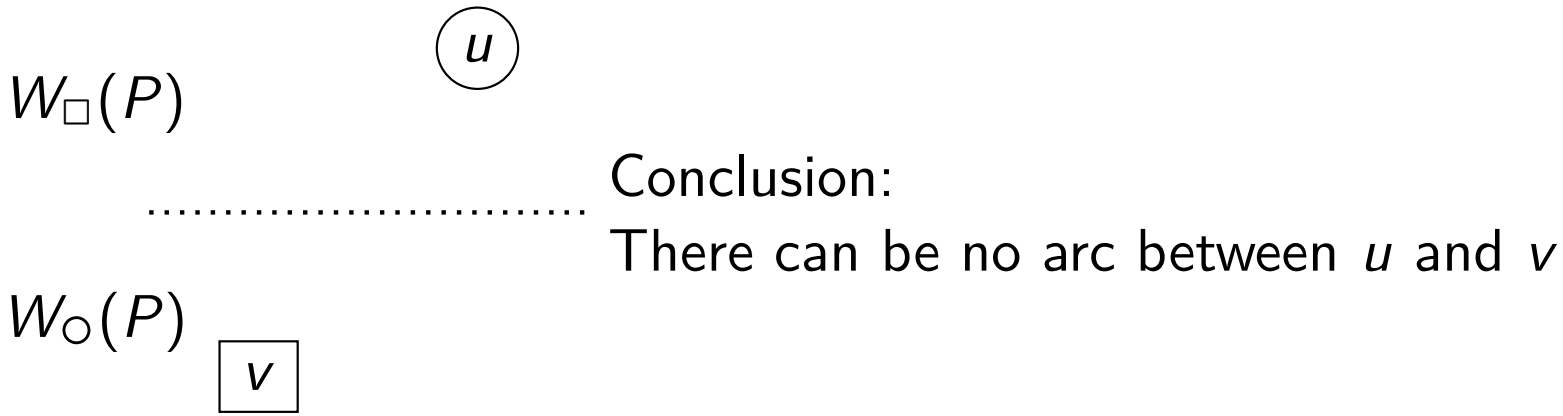
Proof for Join

Let P be a game and $u \in W_{\Box}(P)$ and $v \in W_{\bigcirc}(P)$.



Proof for Join

Let P be a game and $u \in W_\Box(P)$ and $v \in W_\bigcirc(P)$.



In the join operation we require some arcs between vertices of different players.
This restricts the winning regions.

Attractor Sets

Let P be a parity game and $A \subseteq V(P)$.

Definition 8

$\text{attr}_i(A)$ is the set of all nodes such that player i can force the game to reach a node in A.

We observe

$$W_{\bigcirc}(P \setminus \text{attr}_{\Box}(A)) \subseteq W_{\bigcirc}(P).$$

Proof for Join

Let P be a join of $P' \in \mathcal{C}$ and $P'' \in \mathcal{C}'$.

Algorithm for solving Join$(\mathcal{C}, \mathcal{C}')$

1. Solve $P_1 := P \setminus \text{attr}_\square(V_\square(P'))$.
2. If $W_\bigcirc(P_1) \neq \emptyset$, return the subgame $P \setminus \text{attr}_\bigcirc(W_\bigcirc(P_1))$.
3. Solve $P_2 := P \setminus \text{attr}_\bigcirc(V_\bigcirc(P''))$.
4. If $W_\square(P_2) \neq \emptyset$, return the subgame $P \setminus \text{attr}_\square(W_\square(P_2))$.
5. Return "$W_\bigcirc(P) = \emptyset$ or $W_\square(P) = \emptyset$".

Proof for Join

Let P be a join of $P' \in \mathcal{C}$ and $P'' \in \mathcal{C}'$.

Algorithm for solving Join$(\mathcal{C}, \mathcal{C}')$

1. Solve $P_1 := P \setminus \text{attr}_\square(V_\square(P'))$.
2. If $W_\bigcirc(P_1) \neq \emptyset$, return the subgame $P \setminus \text{attr}_\bigcirc(W_\bigcirc(P_1))$.
3. Solve $P_2 := P \setminus \text{attr}_\bigcirc(V_\bigcirc(P''))$.
4. If $W_\square(P_2) \neq \emptyset$, return the subgame $P \setminus \text{attr}_\square(W_\square(P_2))$.
5. Return "$W_\bigcirc(P) = \emptyset$ or $W_\square(P) = \emptyset$".

All steps need time $O(n^c)$ for some $c > 0$ because P_1, P_2 are in HalfJoin$(\mathcal{C})$ or HalfJoin$(\mathcal{C}')$. By Lemma 5, we get $O(n^{c+1})$ for the complete algorithm.

Proof for Join

Let P be a join of $P' \in \mathcal{C}$ and $P'' \in \mathcal{C}'$.

Proof for Join

Let P be a join of $P' \in \mathcal{C}$ and $P'' \in \mathcal{C}'$.
Observation: If $W_{\bigcirc}(P) \cap P'$ contains $\Box$-nodes, then $W_{\Box}(P) \cap P''$ cannot contain $\bigcirc$-nodes and vice-versa.

Proof for Join

Let P be a join of $P' \in \mathcal{C}$ and $P'' \in \mathcal{C}'$.
Observation: If $W_{\bigcirc}(P) \cap P'$ contains $\Box$-nodes, then $W_{\Box}(P) \cap P''$ cannot contain $\bigcirc$-nodes and vice-versa.

Case 1 $W_{\bigcirc}(P) \cap P'$ contains no $\Box$-nodes.

Case 2 $W_{\Box}(P) \cap P''$ contains no $\bigcirc$-nodes.

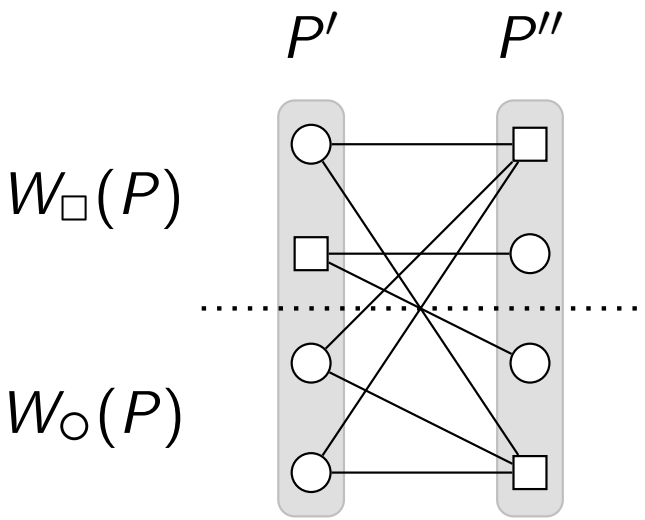
Proof for Join

Case 1 $W_{\bigcirc}(P) \cap P'$ contains no $\Box$-nodes.

Proof for Join

Case 1 $W_{\bigcirc}(P) \cap P'$ contains no $\square$-nodes.

Then all $\square$-nodes in P' are winning for Player $\square$.



Proof for Join

Hence it is enough to compute $W_\bigcirc(P_1)$ with

$$P_1 := P \setminus \mathrm{attr}_\Box(V_\Box(P'))$$

because we have

$$W_\bigcirc(P) = W_\bigcirc(P_1).$$

Proof for Join

Hence it is enough to compute $W_{\bigcirc}(P_1)$ with

$$P_1 := P \setminus \mathrm{attr}_{\Box}(V_{\Box}(P'))$$

because we have

$$W_{\bigcirc}(P) = W_{\bigcirc}(P_1).$$

$W_{\bigcirc}(P_1) = \emptyset$ implies $W_{\bigcirc}(P) = \emptyset$.

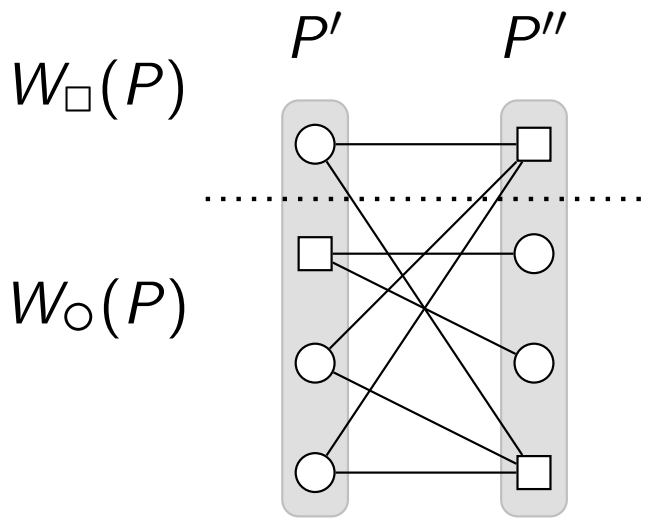
Proof for Join

Case 2 $W_{\square}(P) \cap P''$ contains no $\circ$-nodes.

Proof for Join

Case 2 $W_\square(P) \cap P''$ contains no $\circ$-nodes.

Then all $\circ$-nodes in P'' are winning for Player $\circ$.



Proof for Join

Hence it is enough to compute $W_\Box(P_2)$ with

$$P_2 := P \setminus \mathrm{attr}_\bigcirc(V_\bigcirc(P''))$$

because we have

$$W_\Box(P) = W_\Box(P_2).$$

Proof for Join

Hence it is enough to compute $W_\Box(P_2)$ with

$$P_2 := P \setminus \mathrm{attr}_\bigcirc(V_\bigcirc(P''))$$

because we have

$$W_\Box(P) = W_\Box(P_2).$$

$W_\Box(P_2) = \emptyset$ implies $W_\Box(P) = \emptyset$.

Proof for Join

If we are in Case 1, then $W_{\bigcirc}(P_1) = \emptyset$ implies $W_{\bigcirc}(P) = \emptyset$.
If we are in Case 2, then $W_{\Box}(P_2) = \emptyset$ implies $W_{\Box}(P) = \emptyset$.

Proof for Join

If we are in Case 1, then $W_{\bigcirc}(P_1) = \emptyset$ implies $W_{\bigcirc}(P) = \emptyset$.
If we are in Case 2, then $W_{\Box}(P_2) = \emptyset$ implies $W_{\Box}(P) = \emptyset$.

Conclusion, no matter if we are in Case 1 or Case 2:
If $W_{\bigcirc}(P_1) = W_{\Box}(P_2) = \emptyset$, then $W_{\bigcirc}(P) = \emptyset$ or $W_{\Box}(P) = \emptyset$.

Proof for HalfJoin

For HalfJoin($\mathcal{C}$), we proceed in exactly the same way. The subgames will be in $\mathcal{C}$ or single-player games, respectively, so they can be solved in polynomial time.

Adding a Single Vertex

Let

$$\text{AddVertex}(\mathcal{C}) := \{P \mid P \text{ is a parity game and there exists a vertex } v \text{ such that } P \setminus \{v\} \in \mathcal{C}\}.$$

Adding a Single Vertex

Let

$$\text{AddVertex}(\mathcal{C}) := \{P \mid P \text{ is a parity game and there exists a vertex } v \text{ such that } P \setminus \{v\} \in \mathcal{C}\}.$$

Theorem 9

If $\mathcal{C}$ is a hereditary class of parity games which can be solved in polynomial time, then $\text{AddVertex}(\mathcal{C})$ *can be solved in polynomial time assuming that the added vertex is part of the input.*

An Application: Apex Graphs

Definition 10

An *apex graph* is a graph G with a vertex v such that $G \setminus \{v\}$ is a planar graph.

Corollary 11

If the class of parity games on planar graphs can be solved in polynomial time, then the class of parity games on apex graphs can be solved in polynomial time.

Thank you for your attention

References

Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002.