Graph Operations on Parity Games and Polynomial-Time Algorithms

Christoph Dittmann, Stephan Kreutzer, Alexandru I. Tomescu

Chair for Logic and Semantics, Technical University Berlin

September 11, 2012
GAMES Workshop, Napoli, Italia
Parity Games

- Parity games are polynomial-time equivalent to the model-checking problem of the modal $\mu$-calculus.
- Computing the winner of a parity game is in NP $\cap$ coNP.
Parity Games

- Parity games are polynomial-time equivalent to the model-checking problem of the modal $\mu$-calculus.
- Computing the winner of a parity game is in $\text{NP} \cap \text{coNP}$.

- Open problem: Is there a polynomial-time algorithm for computing the winner of a parity game?
Motivation and contributions

Parity games are known to be solvable in polynomial time on restricted classes of graphs, e.g.:

- graphs of bounded tree-width,
- directed clique-width, DAG-width, entanglement
Motivation and Contributions

Parity games are known to be solvable in polynomial time on restricted classes of graphs, e.g.:

- graphs of bounded tree-width,
- directed clique-width, DAG-width, entanglement

We study graph operations which preserve polynomial-time solvability in order to:

- construct larger classes of graphs
- identify new classes where parity games are solvable in polynomial time
- elucidate new combinatorial properties
Motivation and Contributions

Parity games are known to be solvable in polynomial time on restricted classes of graphs, e.g.:

- graphs of bounded tree-width,
- **directed** clique-width, DAG-width, entanglement

We study graph operations which preserve polynomial-time solvability in order to:

- construct larger classes of graphs
  - join of graphs, adding a single vertex, repeated pasting along vertices
- identify new classes where parity games are solvable in polynomial time
- elucidate new combinatorial properties
Motivation and contributions

Parity games are known to be solvable in polynomial time on restricted classes of graphs, e.g.:

- graphs of bounded tree-width,
- directed clique-width, DAG-width, entanglement

We study graph operations which preserve polynomial-time solvability in order to:

- construct larger classes of graphs
  - join of graphs, adding a single vertex, repeated pasting along vertices
- identify new classes where parity games are solvable in polynomial time
  - orientations of complete graphs, complete bipartite graphs, block graphs
- elucidate new combinatorial properties
Motivation and contributions

Parity games are known to be solvable in polynomial time on restricted classes of graphs, e.g.:

- graphs of bounded tree-width,
- directed clique-width, DAG-width, entanglement

We study graph operations which preserve polynomial-time solvability in order to:

- construct larger classes of graphs
  - join of graphs, adding a single vertex, repeated pasting along vertices
- identify new classes where parity games are solvable in polynomial time
  - orientations of complete graphs, complete bipartite graphs, block graphs
- elucidate new combinatorial properties
  - deciding the winning regions of a parity game is as hard as computing them
**Parity Games**

**Definition**

A parity game $P = (V, V_\square, V_\Diamond, E, \Omega)$ is a directed graph $(V, E)$ with a partitioning of the nodes $V = V_\square \cup V_\Diamond$ 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 $V_{\bigcirc}, V_{\square}$ determine whose turn it is.

**Definition**

A **play** is a maximal sequence of nodes $v_1, v_2, \ldots$ such that $(v_i, v_{i+1}) \in E$ for all $i$.

A play is **winning for Player $\bigcirc$** if the maximum priority that appears infinitely often is even or if the last node is in $V_{\square}$.
Parity Games

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

**Definition**

A play is a maximal sequence of nodes $v_1, v_2, \ldots$ such that $(v_i, v_{i+1}) \in E$ 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 $V_\square$.

Parity games are positionally determined.

We let $W_i(P) :=$ the winning region of player $i$, that is, the set of all nodes from which player $i$ has a positional winning strategy, $i \in \{\circ, \square\}$.

Solving a game means computing its winning regions.
GUIDING IDEA

We will prove that our graph operations guarantee that either:

- We can reduce the problem to solving a proper subgame $P' \subsetneq P$, or
- One winning region of $P$ is empty.
**GUIDING IDEA**

We will prove that our graph operations guarantee that either:

- We can reduce the problem to solving a proper subgame \( P' \subset P \), or
- One winning region of \( P \) is empty.

If this is the case, we can apply the following lemma:

**lemMA**

Let \( C \) be a **hereditary** class of parity games and assume that there is an \( O(n^c) \) time algorithm which for each \( P \in C \) returns either:

1. A proper subgame \( P' \) and \( W^*, W^* \subseteq V(P) \setminus V(P') \) with
   
   \[
   W_\bigcirc(P) = W^* \cup W_\bigcirc(P'),
   \]
   
   \[
   W_\square(P) = W^* \cup W_\square(P').
   \]

2. “\( W_\bigcirc(P) = \emptyset \) or \( W_\square(P) = \emptyset \)”.
**GUIDING IDEA**

We will prove that our graph operations guarantee that either:

- We can reduce the problem to solving a proper subgame $P' \subset P$, or
- One winning region of $P$ is empty.

If this is the case, we can apply the following lemma:

**Lemma**

Let $C$ be a **hereditary** class of parity games and assume that there is an $O(n^c)$ time algorithm which for each $P \in C$ returns either:

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

   $W^\bigcirc(P) = W^* \cup W^\bigcirc(P')$, 
   $W^\bigsquare(P) = W^* \cup W^\bigsquare(P')$.

2. "$W^\bigcirc(P) = \emptyset$ or $W^\bigsquare(P) = \emptyset$".

Then $C$ can be solved in time $O(n^{c+1})$. 
Deciding winning regions

**Problem: Deciding winning regions**

Given a parity game $P$ and $A \subseteq V(P)$, decide if $W_{\square}(P) = A$. 
Deciding winning regions

**Problem: Deciding winning regions**

Given a parity game $P$ and $A \subseteq V(P)$, decide if $W_2(P) = A$.

**Theorem**

Deciding winning regions is as hard as computing them, even if we can only decide whether $W_2(P) = V(P)$. 
JOINS

Game $P$ is a join of $P'$ and $P''$ if for every $i \in \{\circ, \Box\}$, every vertex of Player $i$ in $P'$ is connected with every vertex of Player $\bar{i}$ in $P''$. 
JOINS

Game $P$ is a join of $P'$ and $P''$ if for every $i \in \{\bigcirc, \Box\}$, every vertex of Player $i$ in $P'$ is connected with every vertex of Player $\bar{i}$ in $P''$. 

![Diagram showing the join of two games $P'$ and $P''$]
JOINS

Game $P$ is a join of $P'$ and $P''$ if for every $i \in \{\circ, \blacksquare\}$, every vertex of Player $i$ in $P'$ is connected with every vertex of Player $\bar{i}$ in $P''$. 

![Diagram showing the join of two graphs $P'$ and $P''$.]
JOINS

Game $P$ is a join of $P'$ and $P''$ if for every $i \in \{\circ, \square\}$, every vertex of Player $i$ in $P'$ is connected with every vertex of Player $\bar{i}$ in $P''$. 
JOINS

Game $P$ is a join of $P'$ and $P''$ if for every $i \in \\{\circ, \Box\}$, every vertex of Player $i$ in $P'$ is connected with every vertex of Player $\bar{i}$ in $P''$. 
JOINS

Game $P$ is a join of $P'$ and $P''$ if for every $i \in \{\circ, \square\}$, every vertex of Player $i$ in $P'$ is connected with every vertex of Player $i$ in $P''$.

- Other edges between $P'$ and $P''$ can be present or not
JOINS

Game $P$ is a join of $P'$ and $P''$ if for every $i \in \{\circ, \square\}$, every vertex of Player $i$ in $P'$ is connected with every vertex of Player $\tilde{i}$ in $P''$.

- Other edges between $P'$ and $P''$ can be present or not

- If $P'$ is a $\square$-player game having all edges and $P''$ is a $\circ$-player game having all edges, then $P$ is an orientation of a complete graph (i.e., tournament)
JOINS

Game $P$ is a **join** of $P'$ and $P''$ if for every $i \in \{\circ, \square\}$, every vertex of Player $i$ in $P'$ is connected with every vertex of Player $\tilde{i}$ in $P''$.

- Other edges between $P'$ and $P''$ can be present or not

- If $P'$ is a $\square$-player game having all edges and $P''$ is a $\circ$-player game having all edges, then $P$ is an **orientation of a complete graph** (i.e., tournament)
- If $P'$ and $P''$ are parity game without edges, and we add all edges between $P'$ and $P''$, then $P$ is an **orientation of a complete bipartite graph**
JOINS

**Definition**

Let $C, C'$ be two classes of parity games.

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

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

**Theorem**

If $C, C'$ are hereditary classes of parity games that can be solved in polynomial time, then all games $P \in \text{HalfJoin}(C)$ (in $\text{Join}(C, C')$, resp.) can be solved in polynomial time, provided that a decomposition of $P$ as a join is given.
**Main ingredient of the proof**
Let $P$ be a game and $u \in W_\square(P)$ and $v \in W_\circ(P)$.

In the join operation we require some arcs between vertices of different players.
- This restricts the winning regions.
**Main Ingredient of the Proof**

Let $P$ be a game and $u \in W □ (P)$ and $v \in W □ (P)$.

In the join operation we require some arcs between vertices of different players.

- This restricts the winning regions.
MAIN INGREDIENT OF THE PROOF

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

- In the join operation we require some arcs between vertices of different players
- This restricts the winning regions
**Main Ingredient of the Proof**

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

- In the join operation we require some arcs between vertices of different players
- This restricts the winning regions
**Main Ingredient of the Proof**

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
**Main Ingredient of the Proof**

Let $P$ be a game and $u \in W_\square(P)$ and $v \in W_\circ(P)$.

> Conclusion: There can be no arc between $u$ and $v$

- In the join operation we require some arcs between vertices of different players
- This restricts the winning regions
Algorithm for HalfJoin (1)

- $\text{attr}_i(A) :=$ the nodes from which Player $i$ can force the game to reach $A$
- $W_{\Box}(P \setminus \text{attr}_O(A)) \subseteq W_{\Box}(P)$
**Algorithm for HalfJoin (1)**

- \( \text{attr}_i(A) := \) the nodes from which Player \( i \) can force the game to reach \( A \)
- \( W_\square(P \setminus \text{attr}_\square(A)) \subseteq W_\square(P) \)

Suppose that \( P \) is a HalfJoin of a \( \square \)-player game \( P' \) with a game \( P'' \) in \( C \).

**Algorithm for solving HalfJoin(\( C \)) (Part 1)**

1. Solve \( P_1 := P \setminus \text{attr}_\square(V''_\square) \) (\( \square \)-player game).
2. If \( W_\square(P_1) \neq \emptyset \), return the subgame \( P \setminus \text{attr}_\square(W_\square(P_1)) \).

\[ P_1 = P \setminus \text{attr}_\square(V''_\square) \text{ is a single-player game} \]
Algorithm for HalfJoin (2)

- \(\text{attr}_i(A) := \) the nodes from which Player \(i\) can force the game to reach \(A\)
- \(W_\square(P \setminus \text{attr}_\square(A)) \subseteq W_\bigcirc(P)\)

Suppose that \(P\) is a HalfJoin of a \(\square\)-player game \(P'\) with a game \(P''\) in \(C\).

Algorithm for solving HalfJoin(\(C\)) (Part 2)

3. Solve \(P_2 := P \setminus \text{attr}_\square(V'_\square)\) (game from \(C\)).
4. If \(W_\bigcirc(P_2) \neq \emptyset\), return the subgame \(P \setminus \text{attr}_\bigcirc(W_\bigcirc(P_2))\).
Algorithm for HalfJoin (2)

- $\text{attr}_i(A) :=$ the nodes from which Player $i$ can force the game to reach $A$
- $W_\Diamond(P \setminus \text{attr}_\Box(A)) \subseteq W_\Diamond(P)$

Suppose that $P$ is a HalfJoin of a □-player game $P'$ with a game $P''$ in $C$.

Algorithm for solving HalfJoin($C$) (Part 2)

3. Solve $P_2 := P \setminus \text{attr}_\Box(V'_\Box)$ (game from $C$).
4. If $W_\Diamond(P_2) \neq \emptyset$, return the subgame $P \setminus \text{attr}_\Box(W_\Diamond(P_2))$.
5. Return "$W_\Diamond(P) = \emptyset$ or $W_\Box(P) = \emptyset$".

$P_2 = P \setminus \text{attr}_\Box(V'_\Box)$ is in $C$
**Algorithm for HalfJoin (2)**

- \( \text{attr}_i(A) := \text{the nodes from which Player } i \text{ can force the game to reach } A \)
- \( W_\square(P \setminus \text{attr}_\square(A)) \subseteq W_\square(P) \)

Suppose that \( P \) is a HalfJoin of a \( \square \)-player game \( P' \) with a game \( P'' \) in \( C \).

**Algorithm for Solving HalfJoin(\( C \)) (Part 2)**

3. Solve \( P_2 := P \setminus \text{attr}_\square(V'_\square) \) (game from \( C \)).
4. If \( W_\square(P_2) \neq \emptyset \), return the subgame \( P \setminus \text{attr}_\square(W_\square(P_2)) \).
5. Return "\( W_\square(P) = \emptyset \) or \( W_\square(P) = \emptyset \)".
PROOF FOR HalfJoin

Suppose that $P$ is a HalfJoin of a $\square$-player game $P$ with a game $P''$ in $C$.

Case 1: $W_\bigcirc(P) \cap P' \neq \emptyset$
**PROOF FOR HalfJoin**

Suppose that $P$ is a HalfJoin of a □-player game $P$ with a game $P''$ in $C$.

**Case 1:** $W_\bigcirc(P) \cap P' \neq \emptyset$

- $P_1 = P \setminus \text{attr}_\bigcirc(V''')$ is a □-player game

(a) Subcase 1a: $W_\Box(P_1) \neq \emptyset$

(b) Subcase 1b: $W_\Box(P_1) = \emptyset$
**Proof for HalfJoin**

Suppose that $P$ is a HalfJoin of a $\Box$-player game $P'$ with a game $P''$ in $C$.

**Case 2:** $W_\Box(P) \cap P' = \emptyset$.

$$P_2 = P \setminus \text{attr}_\Box(V'_\Box) \text{ is in } C$$

(a) Subcase 2a: $W_\Box(P_2) \neq \emptyset$

(b) Subcase 2b: $W_\Box(P_2) = \emptyset$
**Proof for Join**

For $\text{Join}(C, C')$, we proceed in exactly the same way.

The subgames will be in $\text{HalfJoin}(C)$ and in $\text{HalfJoin}(C')$, so they can be solved in polynomial time.
**Adding a Single Vertex**

Let

\[ \text{AddVertex}(C) := \{ P \mid P \text{ is a parity game and there exists a vertex } v \text{ such that } P \setminus \{v\} \in C \} \]
**Adding a Single Vertex**

Let

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

**Theorem**

If \(C\) is a hereditary class of parity games which can be solved in time \(O(n^c)\), then \(\text{AddVertex}(C)\) can be solved in time \(O(n^{c+1})\), assuming that the added vertex is part of the input.
**ADDING A SINGLE VERTEX**

Let

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

**Theorem**

If \( C \) is a hereditary class of parity games which can be solved in time \( O(n^c) \), then \( \text{AddVertex}(C) \) can be solved in time \( O(n^{c+1}) \), assuming that the added vertex is part of the input.

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

**Corollary**

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.
**Theorem**

If $C$ is a hereditary class of parity games that can be solved in polynomial time, then games in $\text{RepeatedPasting}(C)$ can be solved in polynomial time.
**Theorem**

If $C$ is a hereditary class of parity games that can be solved in polynomial time, then games in $\text{RepeatedPasting}(C)$ can be solved in polynomial time.

A **block graph** is an undirected graph whose 2-connected components are complete graphs.

**Corollary**

Parity games on any orientation of a block-graph can be solved in polynomial time.
Conclusions

We considered three graph operations and showed that they preserve the polynomial-time solvability of parity games:

- join of graphs, adding a single vertex, repeated pasting along vertices
CONCLUSIONS

We considered three graph operations and showed that they preserve the polynomial-time solvability of parity games:

▶ join of graphs, adding a single vertex, repeated pasting along vertices

By particularizing these constructions, we identified new classes of graphs on which parity games are solvable in polynomial-time:

▶ orientations of complete graphs, complete bipartite graphs, or block graphs
CONCLUSIONS

We considered three graph operations and showed that they preserve the polynomial-time solvability of parity games:

▶ join of graphs, adding a single vertex, repeated pasting along vertices

By particularizing these constructions, we identified new classes of graphs on which parity games are solvable in polynomial-time:

▶ orientations of complete graphs, complete bipartite graphs, or block graphs

Future work:

▶ search for other graph operations
▶ complete graphs and complete bipartite graphs have undirected clique-width 2; can we say something about any graph of undirected clique-width 2?
▶ study whether these graph operations preserve solvability of other games