Winning Cores in Parity Games

Steen Vester

DTU Compute

October 22, 2015

Outline

Outline

Why is parity game solving important?

Solving parity games is **polynomial-time equivalent** to

- μ-calculus model-checking
- Solving boolean equation systems
- Emptiness of parity tree automata on infinite binary trees.

Various problems are **reducible to parity games**, e.g.

- Satisfiability problems
- Model-checking problems
- Synthesis problems

(though, not necessarily by polynomial-time reductions)

Complexity Status

It is unknown whether $\textsc{ParityGame}$ is in $\textsc{PTime}$.
We know:

- $\textsc{Paritygame}$ is in $\mathrm{NP} \cap \textsc{co}\text{-}\mathrm{NP}$ implying
  - If it is NP-complete then $\mathrm{NP} = \textsc{co}\text{-}\mathrm{NP}$
  - If it is not solvable in $\textsc{PTime}$ then $\mathsf{P} \neq \mathsf{NP}$
- For a fixed maximal color d, it is in $\textsc{PTime}$

Existing Algorithms

The best current algorithms for solving parity games are

- Zielonkas Recursive algorithm $O(n^d)$ and $O(2^n)$ [Zielonka, 1998]
- Small Progress Measures $O(d \cdot m \cdot (n/d)^{d/2})$ [Jurdzinski, 1998]
- Strategy Improvement $O(n \cdot m \cdot 2^m)$ [Vöge and Jurdzinski, 2000]
- Dominion Decomposition $O(n^{\sqrt{n}})$ [Jurdzinski et al., 2006]
- Big Step Algorithms $O(m \cdot n^{d/3})$ [Schewe, 2007]

where n is the number of states, m is the number of transitions, d is the maximal color of the game.
Note: $d \leq n$

Contributions

We introduce and study **winning cores**
They are interesting because they provide

1. **knowledge** about parity games
2. a **new direction** for solving parity games
3. a polynomial-time **approximation algorithm** for solving parity games

Outline

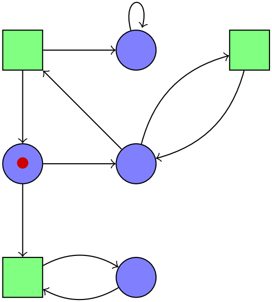
A game graph

Player 0 state

Player 1 state

Transition

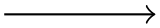
Current state



A game graph

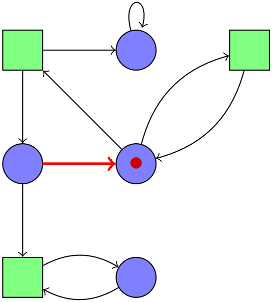
Player 0 state

Player 1 state

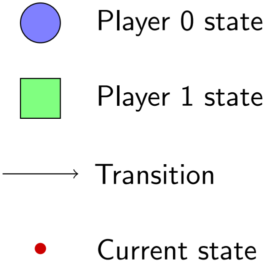


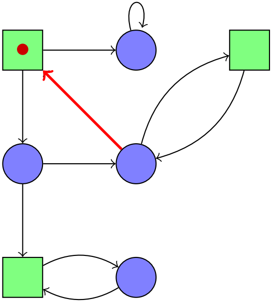
Transition

Current state



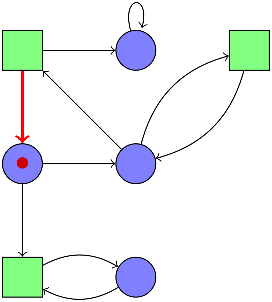
A game graph





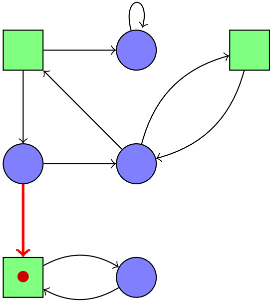
A game graph

- Player 0 state
- Player 1 state
- Transition
- Current state



A game graph

- Player 0 state
- Player 1 state
- Transition
- Current state



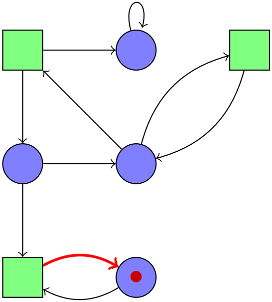
A game graph

Player 0 state

Player 1 state

Transition

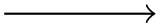
Current state



A game graph

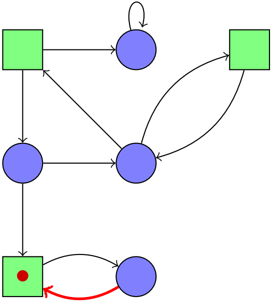
Player 0 state

Player 1 state



Transition

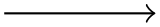
Current state



A game graph

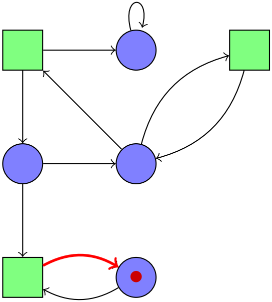
Player 0 state

Player 1 state



Transition

Current state



A Parity Game

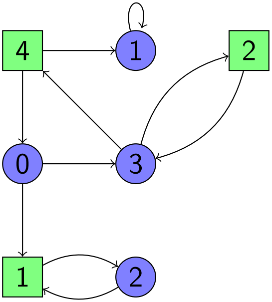
Player 0 state

Player 1 state

Transition

Current state

$0, \ldots, d$ **Colors**



- **Player 0** wants the largest color infinitely often visited is **even**
- **Player 1** wants the largest color infinitely often visited is **odd**

Determinacy

A game is **determined** if for every state s either

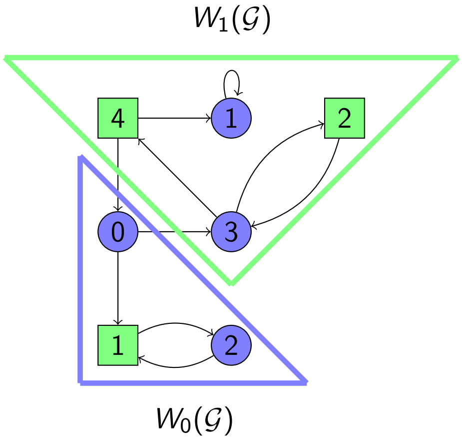
- Player 0 can ensure winning from s or
- Player 1 can ensure winning from s

Theorem ([Ehrenfeucht and Mycielski, 1979])

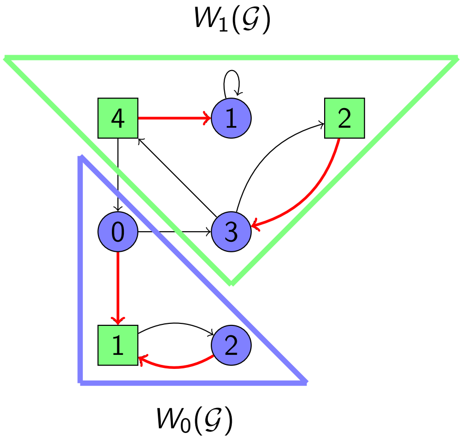
Parity games are determined

$\mathcal{G}$: A parity game
$W_j(\mathcal{G})$: Set of winning states for player j

Determinacy Example



Determinacy Example



Solving parity games

<u>ParityGame</u>
Input: A parity game $\mathcal{G}$
Output: $W_0(\mathcal{G}), W_1(\mathcal{G})$

Outline

Dominating sequences

A sequence $\rho = s_0 s_1 \dots$ of states is

- 0-dominating if $\max_{i>0}(c(s_i))$ is even
- 1-dominating if $\max_{i>0}(c(s_i))$ is odd

Note: Initial state does not count

$$1 \rightarrow 4 \rightarrow 3 \rightarrow 4 \rightarrow 3$$

0-dominating

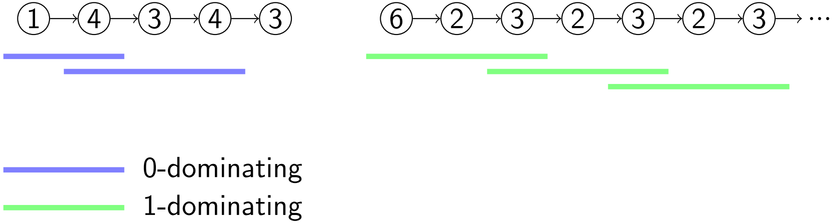
$$6 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 3 \rightarrow \cdots$$

1-dominating

Consecutive j-dominating sequences

A sequence $\rho = s_0 s_1 \ldots$ begins with k consecutive j-dominating sequences if $\exists i_0 < i_1 \ldots < i_k$ such that

- $i_0 = 0$
- $\rho_{i_\ell} \rho_{i_\ell + 1} \ldots \rho_{i_{\ell+1}}$ is j-dominating for all $0 \leq \ell < k$

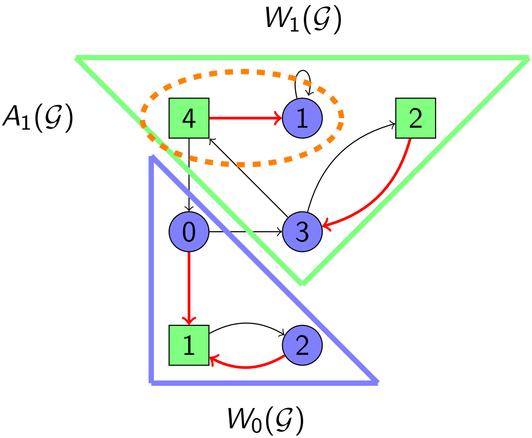


Winning core

Winning core $A_j(\mathcal{G})$ for player j in game $\mathcal{G}$:

Set of **states** from which player j can **force** the play to begin with an **infinite number** of consecutive j-dominating sequences.

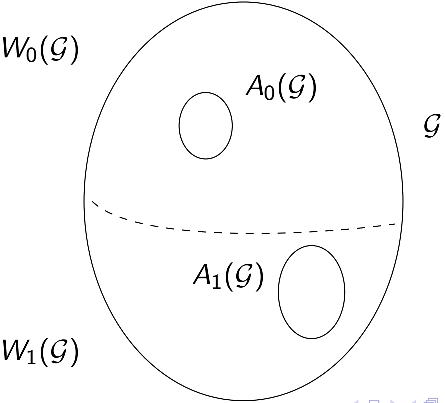
Winning core example



Properties of winning cores

Theorem

- $A_j(\mathcal{G}) \subseteq W_j(\mathcal{G})$
- $A_j(\mathcal{G}) = \emptyset \Leftrightarrow W_j(\mathcal{G}) = \emptyset$



Winning cores and dominions

A *j-dominion* D is a set of states so player j can make sure that both

1. the play stays in D and
2. that player j wins the play

Interestingly, the winning core $A_j(\mathcal{G})$ is not necessarily a j-dominion.

Complexity of computing winning cores

Theorem

There is a **polynomial-time reduction** *from* $\textsc{ParityGame}$ *to computing winning cores and vice versa*

Corollary

- *Computing winning cores is in* $\mathrm{NP} \cap \textsc{co}\text{-}\mathrm{NP}$
- *Computing winning cores is in* P *if and only if* $\textsc{ParityGame}$ *is in* P

Computing winning regions using winning cores

ParityGameSolver($\mathcal{G}$):
$A \leftarrow \textsc{WinningCore}(\mathcal{G}, 0)$
$B \leftarrow \textsc{WinningCore}(\mathcal{G}, 1)$
if $A = \emptyset$ and $B = \emptyset$ **then**
    return $(\emptyset, \emptyset)$
end if
$A' = \mathsf{Attr}_0(\mathcal{G}, A)$
$B' = \mathsf{Attr}_1(\mathcal{G}, B)$
$(W_0, W_1) \leftarrow \textsc{ParityGameSolver}(\mathcal{G} \setminus (A' \cup B'))$
return $(A' \cup W_0, B' \cup W_1)$

Note: If $\textsc{WinningCore}(\mathcal{G}, j)$ returns a subset of $A_j(\mathcal{G})$ then $\textsc{ParityGameSolver}(\mathcal{G})$ returns subsets of the winning regions

An underapproximation algorithm

WinningCoreApp$(\mathcal{G}, j)$:
$A \leftarrow S$
$A' \leftarrow \emptyset$
while $A \neq A'$ **do**
  $A' \leftarrow A$
  $A \leftarrow \{s \mid \text{Player } j \text{ can ensure a } j\text{-dominating sequence ending in } A'\}$
end while
return A

Note: Returns subset of $A_j(\mathcal{G})$
Combined with previous slide, gives **underapproximations** of **winning regions** in time $O(d \cdot n^2 \cdot (n+m))$ and $O(n+m+d)$ space.

Quality of approximation algorithm

No guarantees on the quality of underapproximations :(
But:

1. It is **easy to check** whether entire winning region is returned
2. Preliminary **experimental results** on
   - Random games
   - Difficult benchmark games
   - A few verification cases

   are **promising** both w.r.t.
   - **(Quality)** All benchmark games and verification cases solved completely. High ratio of random games solved
   - **(Running time)** It outperforms existing algorithms in most cases

Experiments - overview

- Algorithm implemented in PgSolver framework in OCaml
- PgSolver has implementations of state-of-the-art algorithms for comparison
- Performed on an Intel(R) Core(TM) i7-4610M Processor (2.90 GHz)

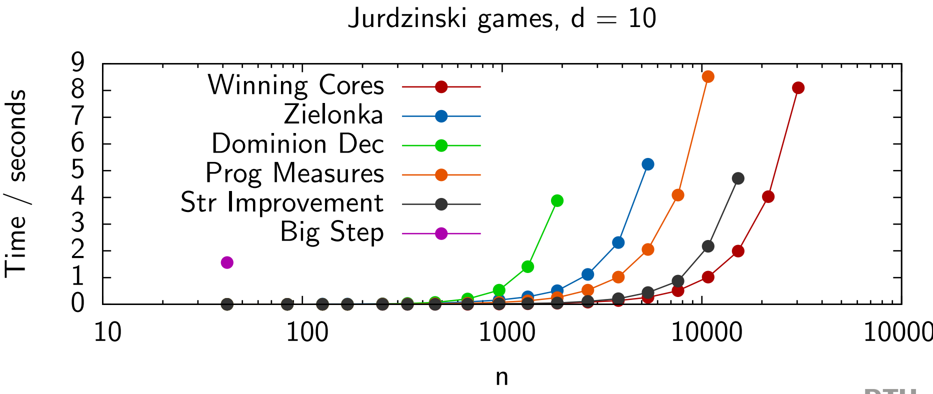
Experimental results - random games

Ratio (in %) of games where the algorithm did not return the **entire winning region**

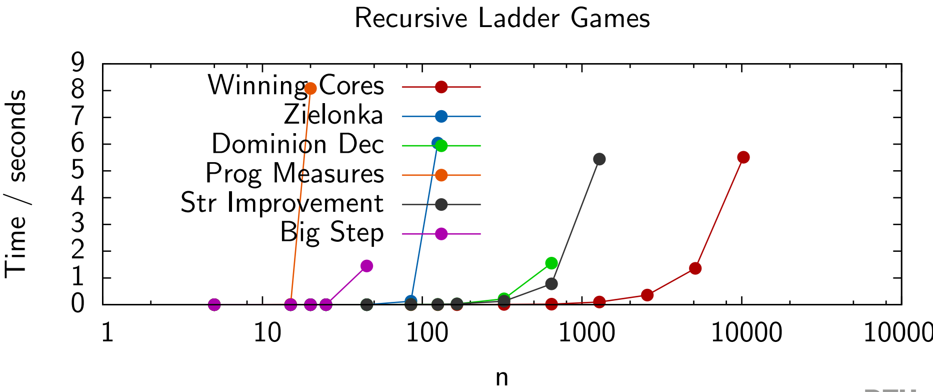
	$d = 4$			$d = \lceil \sqrt{n} \rceil$			$d = n$		
$n \backslash b$	2	5	10	2	5	10	2	5	10
100	0.11	0.20	0.00	0.80	0.04	0.00	1.37	0.05	0.00
1000	0.01	0.00	0.00	2.79	0.00	0.00	4.66	0.00	0.00

n is the number of states, d is the number of colors and b is the out-degree.

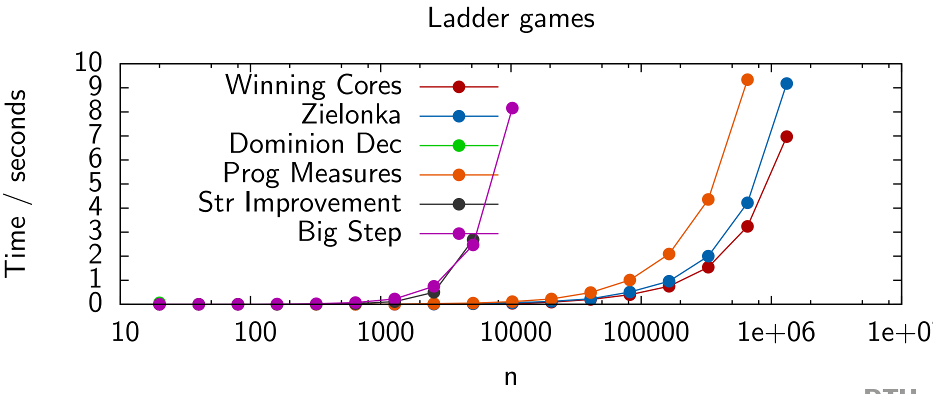
Experimental results - hard games



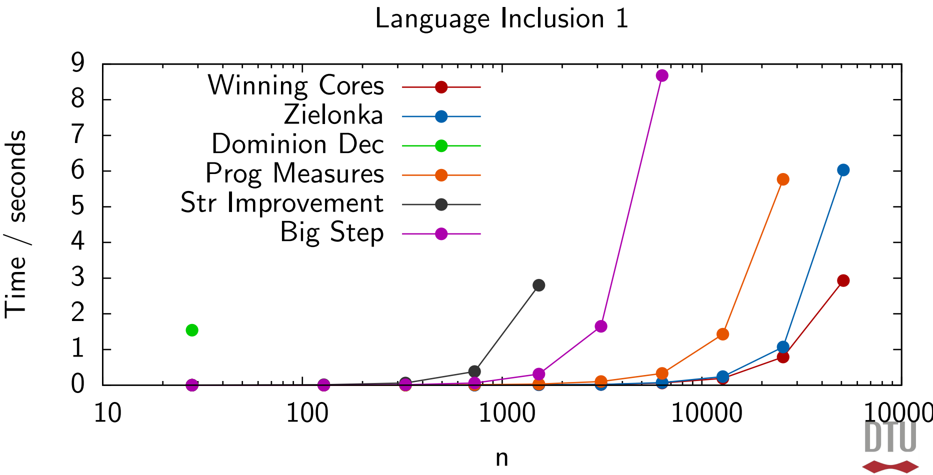
Experimental results - hard games



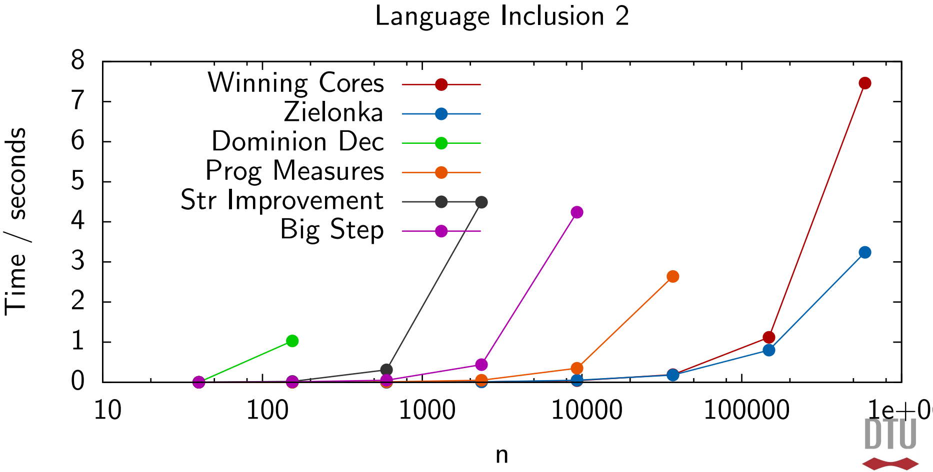
Experimental results - hard games



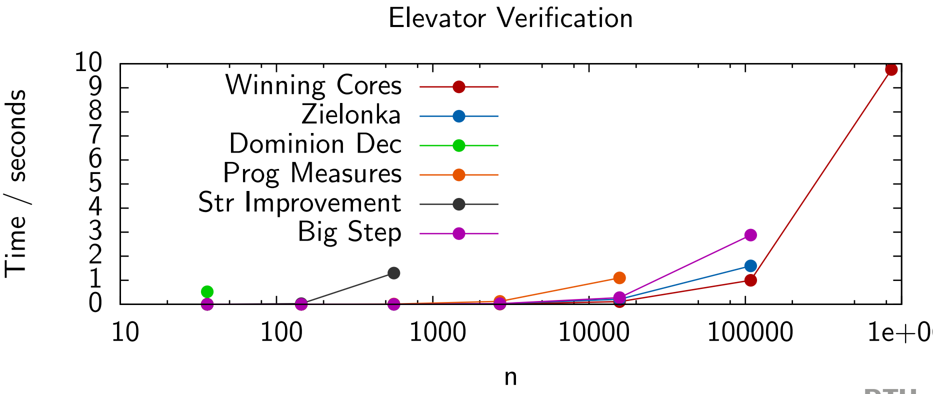
Experimental results - verification case studies



Experimental results - verification case studies



Experimental results - verification case studies



Outline

Summary

We have

- Introduced winning cores
- Shown interesting properties
- Provided an approximation algorithm for parity games
- Shown promising initial experimental results

Open questions

- For which games does the approximation algorithm give correct results?
- Do winning cores have further interesting properties?
- Are there fast deterministic algorithms for computing winning cores?
- Can winning cores be computed in polynomial time?

Bibliography

Ehrenfeucht, A. and Mycielski, J. (1979).
Positional strategies for mean payoff games.
International Journal of Game Theory, 8(2):109–113.

Jurdzinski, M. (1998).
Deciding the winner in parity games is in UP \cap co-up.
Inf. Process. Lett., 68(3):119–124.

Jurdzinski, M., Paterson, M., and Zwick, U. (2006).
A deterministic subexponential algorithm for solving parity games.
In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 117–123.

Schewe, S. (2007).
Solving parity games in big steps.
In *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science, 27th International Conference, New*