



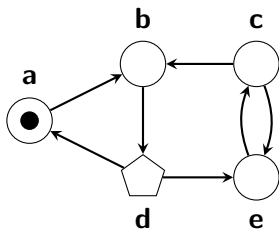
Attracting Tangles to Solve Parity Games

Tom van Dijk (JKU Linz)

CAV, 17 July 2018

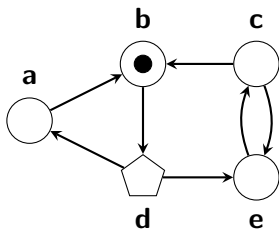
Parity Games

- A parity game is played on a **directed graph**
- Two players: **Even** \circ and **Odd** \diamond
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



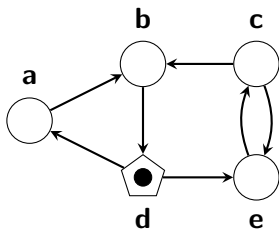
Parity Games

- A parity game is played on a **directed graph**
- Two players: **Even** \circ and **Odd** \diamond
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



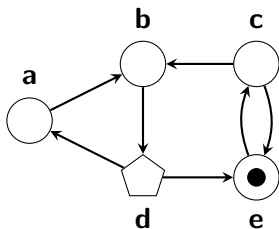
Parity Games

- A parity game is played on a **directed graph**
- Two players: **Even** \circ and **Odd** \diamond
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



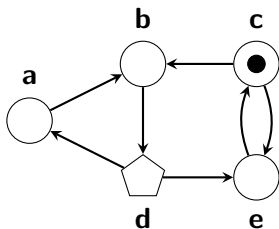
Parity Games

- A parity game is played on a **directed graph**
- Two players: **Even** \circ and **Odd** \diamond
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



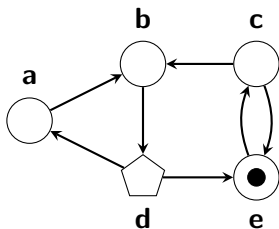
Parity Games

- A parity game is played on a **directed graph**
- Two players: **Even** \circ and **Odd** \diamond
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



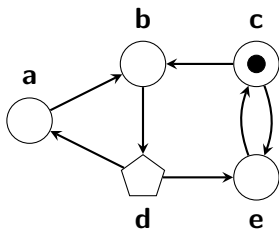
Parity Games

- A parity game is played on a **directed graph**
- Two players: **Even** \circ and **Odd** \diamond
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



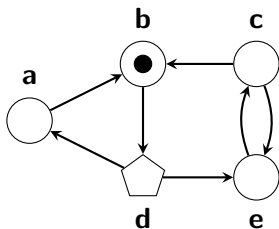
Parity Games

- A parity game is played on a **directed graph**
- Two players: **Even** \circ and **Odd** \diamond
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



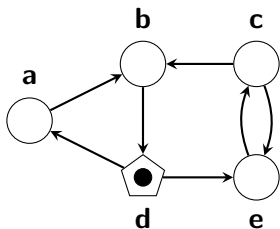
Parity Games

- A parity game is played on a **directed graph**
- Two players: **Even** \circ and **Odd** \diamond
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



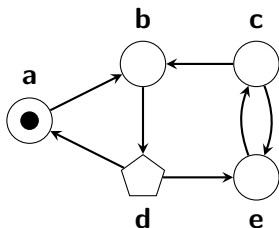
Parity Games

- A parity game is played on a **directed graph**
- Two players: **Even** \circ and **Odd** \diamond
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



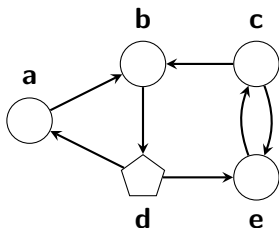
Parity Games

- A parity game is played on a **directed graph**
- Two players: **Even** \circ and **Odd** \diamond
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



Parity Games

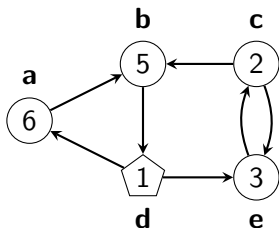
- A parity game is played on a **directed graph**
- Two players: **Even** \circ and **Odd** \diamond
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



How do we determine who wins a play?

Parity Games

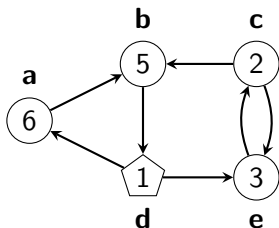
- A parity game is played on a **directed graph**
- Two players: **Even** \circ and **Odd** \diamond
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



- Each vertex has a **priority** $\{0, 1, 2, \dots, d\}$
- **Highest priority seen infinitely often** determines winner
- Player Even wins if this number is even

Parity Games

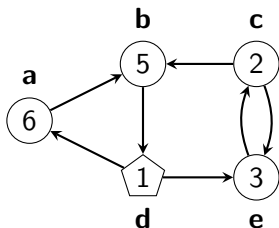
- A parity game is played on a **directed graph**
- Two players: **Even** \circ and **Odd** \diamond
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



How do we determine who wins a vertex?

Parity Games

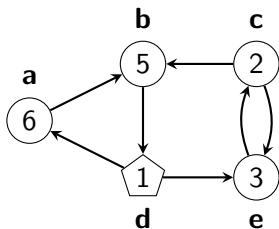
- A parity game is played on a **directed graph**
- Two players: **Even** \circ and **Odd** \diamond
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



A player **wins a vertex** if it has a **strategy** to win all plays from that vertex

Parity Games

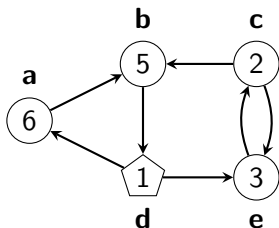
- A parity game is played on a **directed graph**
- Two players: **Even** \circ and **Odd** \diamond
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



Which vertices are won by which player?

Parity Games

- A parity game is played on a **directed graph**
- Two players: **Even** \circ and **Odd** \diamond
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



Player Odd wins all vertices with strategy $\{ \mathbf{d} \rightarrow \mathbf{e} \}$

Parity Games

Known facts of parity games

- Some vertices are won by Even, some vertices are won by Odd
- The winner has a **memoryless strategy** to win

Memoryless strategy

“If I always play from v to w , then I win all plays from v ”

Parity Games

Known facts of parity games

- Some vertices are won by Even, some vertices are won by Odd
- The winner has a **memoryless strategy** to win

Memoryless strategy

“If I always play from v to w , then I win all plays from v ”

Solving a parity game

- Determine the winner of each vertex
- Compute the strategy for each player

Why do we want to solve parity games?

- As expressive as nested least and greatest fixpoint operators
- Polynomial-time equivalent to:
 - modal μ -calculus model-checking
 - solving Boolean Equation Systems
- Backend for LTL model checking and LTL synthesis

Why do we want to solve parity games?

- As expressive as nested least and greatest fixpoint operators
- Polynomial-time equivalent to:
 - modal μ -calculus model-checking
 - solving Boolean Equation Systems
- Backend for LTL model checking and LTL synthesis

Open question: Is solving parity games in **P**?

- The problem is in **NP** \cap **co-NP**
- The problem is in **UP** \cap **co-UP**
- It is believed a polynomial solution exists

(Incomplete list of) published algorithms

McNaughton/Zielonka	$\mathcal{O}(e \cdot n^d), \mathcal{O}(2^n)$	1998
Small Progress Measures	$\mathcal{O}(d \cdot e \cdot (n/d)^{d/2})$	1998
Strategy Improvement	$\mathcal{O}(n \cdot e \cdot 2^e)$	2000
Dominion Decomposition	$\mathcal{O}(n^{\sqrt{n}})$	2006
Big Step	$\mathcal{O}(e \cdot n^{d/3})$	2007
APT	$\mathcal{O}(n^d)$	2016
Priority Promotion	Exponential	2016
Quasi-Polynomial (multiple)	$\mathcal{O}(n^{6+\log d})$	2016 – 2018
Tangle Learning	(tbd)	2018

Attractor computation

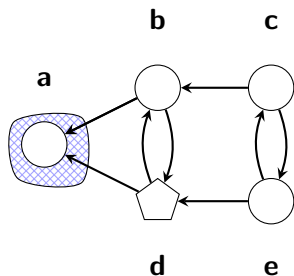
Compute all vertices from which player α can ensure arrival in a target set

Start with the target set A , then iteratively add vertices to A :

- All vertices of α with an edge to A
- All vertices of $\bar{\alpha}$ with only edges to A

Example of attractor computation

Computing the \heartsuit -attractor to **a**

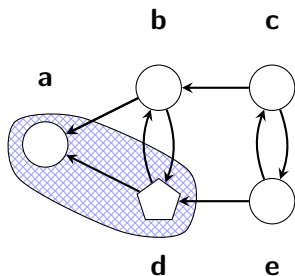


Initial set: $\{\mathbf{a}\}$

Can attract: **d** but not **b**

Example of attractor computation

Computing the \heartsuit -attractor to **a**

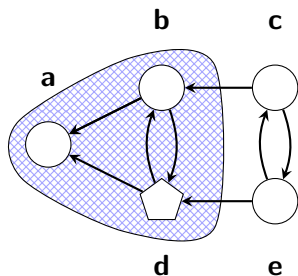


Current set: $\{\mathbf{a}, \mathbf{d}\}$

Can attract: **b** but not **e**

Example of attractor computation

Computing the \heartsuit -attractor to **a**

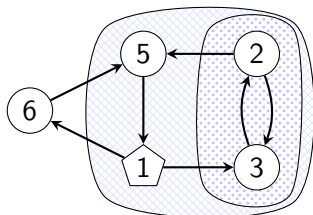


Current set: $\{\mathbf{a}, \mathbf{b}, \mathbf{d}\}$

Can attract: neither **c** nor **e**

The notion of a **tangle**

- A tangle is a strongly connected subgraph, such that all plays that stay in the tangle are won by one player
- Therefore the other player must leave the subgraph



The notion of a tangle

- A tangle is a strongly connected subgraph, such that all plays that stay in the tangle are won by one player
- Therefore the other player must leave the subgraph

The role of tangles in parity game solving algorithms

- Many algorithms implicitly explore tangles
- They often explore the same tangles over and over again
- This leads to an exponential number of steps

Main contribution

Tangles can be used with attractor computation!

The loser **must leave** the tangle

Thus we can attract vertices of a tangle together

Tangle learning

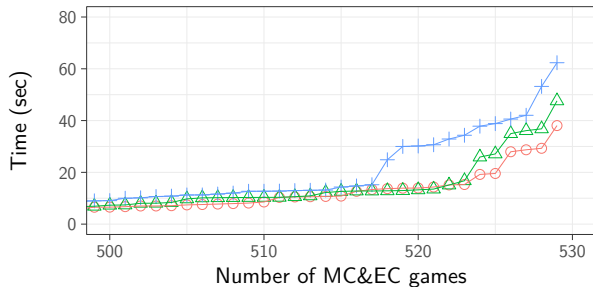
- Extend attractor computation to attract tangles
- Use extended attractor computation to decompose the game
 - Compute attractor set to highest priority
 - Remove this attractor set from the game
- Analyse decomposition to compute new tangles
- Refine decomposition with new tangles
- Repeat this until the game is solved

Empirical evaluation

- Evaluated using [Oink](#) (TACAS 2018)
- Benchmarks
 - Model checking and equivalence checking games [Keiren 2015]
 - Random games
 - Random games with max out-degree 2
- Runtimes in seconds, timeout 20 minutes

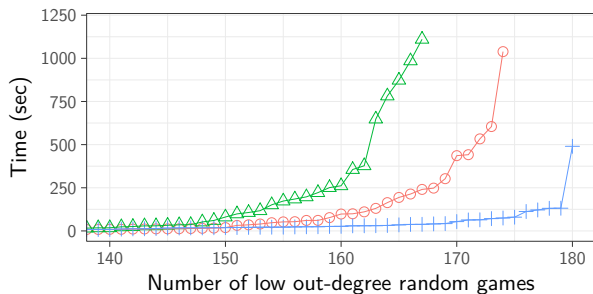
Solver	MC&EC	Random	Random (low degree)	
	time	time	time	timeouts
priority promotion	503	21	12770	6
recursive	576	21	23119	13
tangle learning	808	21	2281	0

Empirical evaluation



Solver

- priority promotion
- △ recursive
- + tangle learning



Conclusions

- Fast moving field with great progress in the last few years
- Existing algorithms implicitly explore tangles repeatedly
- Tangles can be used with attractor computation
- Tangle learning explicitly remembers tangles
- See for yourself: <https://www.github.com/trolando/oink>

Solver	MC&EC	Random	Random (low degree)	
	time	time	time	timeouts
priority promotion	503	21	12770	6
recursive	576	21	23119	13
tangle learning	808	21	2281	0