Distractions in Parity Games

Tom van Dijk

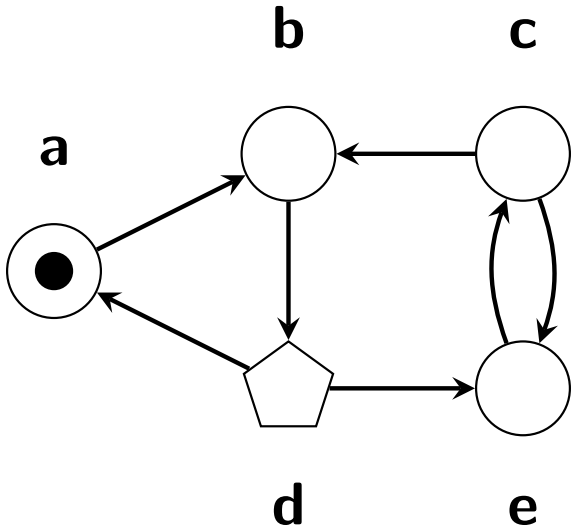
Highlights of Logic, Games and Automata 2019

Overview

- Goal: understanding parity game algorithms
- Tangles in parity game algorithms (*last year*)
  - Parity games algorithm reason how a player must play from **A** to **B**
  - Tangles are the underlying structure when a player cannot stay somewhere (that is, inside the tangle) and must move towards B instead.
- Distractions in parity game algorithms

Parity Games

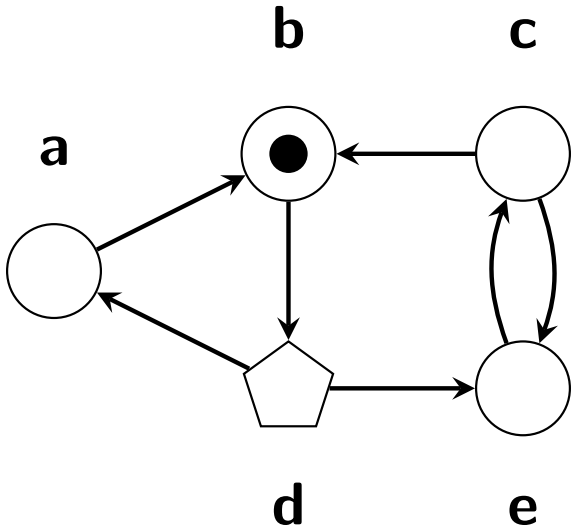
- A parity game is played on a directed graph
- Two players: Even ○ and Odd ⬠
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



The play π: **a**

Parity Games

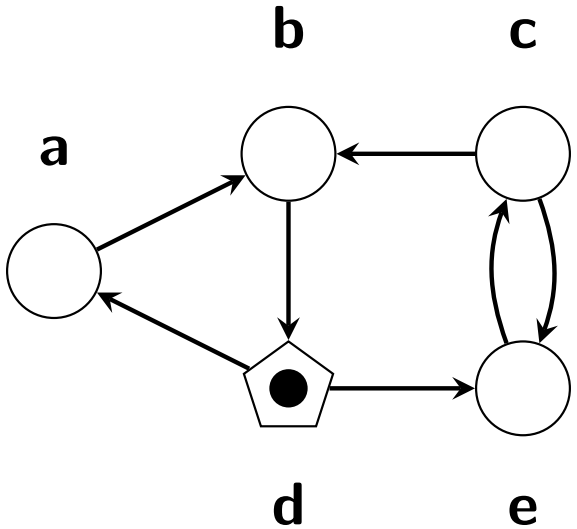
- A parity game is played on a directed graph
- Two players: Even ○ and Odd ⬠
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



The play π: **a b**

Parity Games

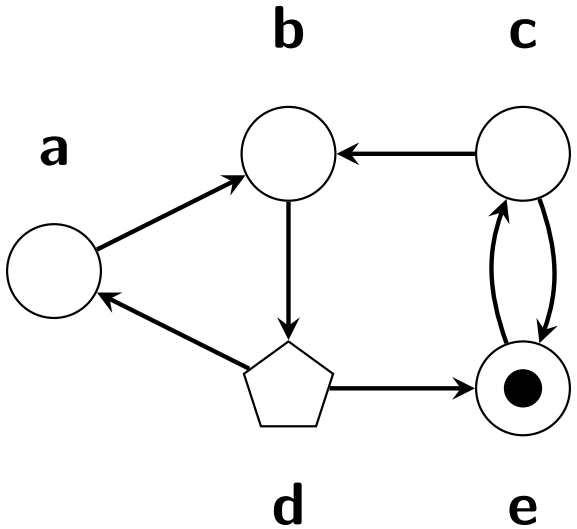
- A parity game is played on a directed graph
- Two players: Even ○ and Odd ⬠
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



The play π: **a b d**

Parity Games

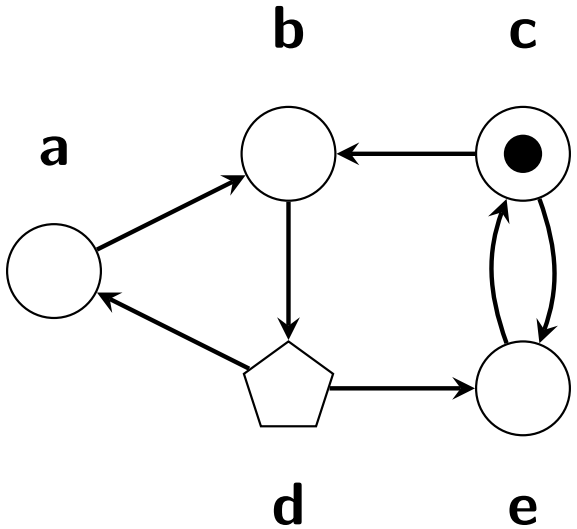
- A parity game is played on a directed graph
- Two players: Even ○ and Odd ⬠
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



The play π: **a b d e**

Parity Games

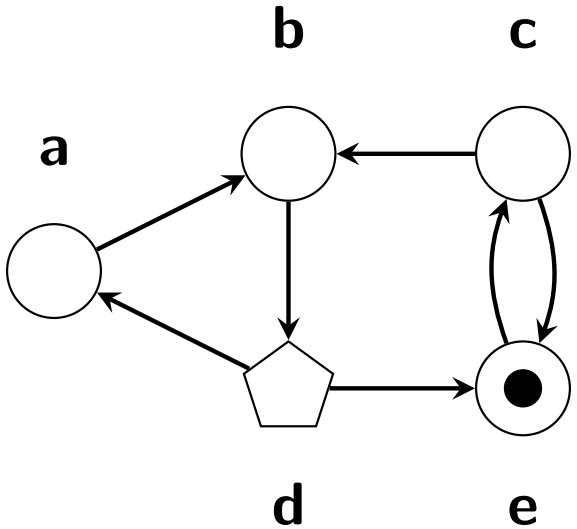
- A parity game is played on a directed graph
- Two players: Even ○ and Odd ⬠
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



The play π: **a b d e c**

Parity Games

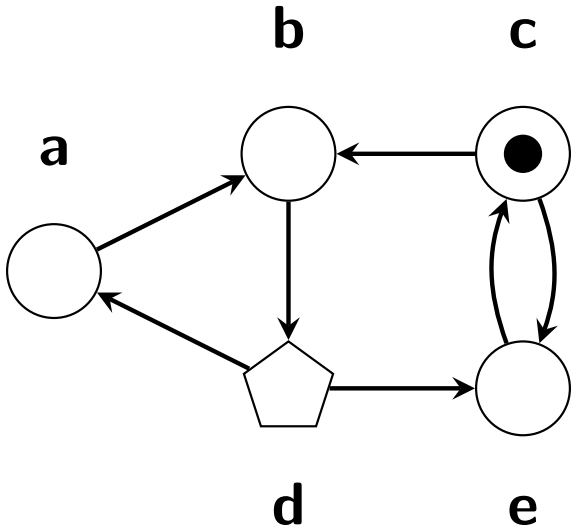
- A parity game is played on a directed graph
- Two players: Even ○ and Odd ⬠
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



The play π: **a b d e c e**

Parity Games

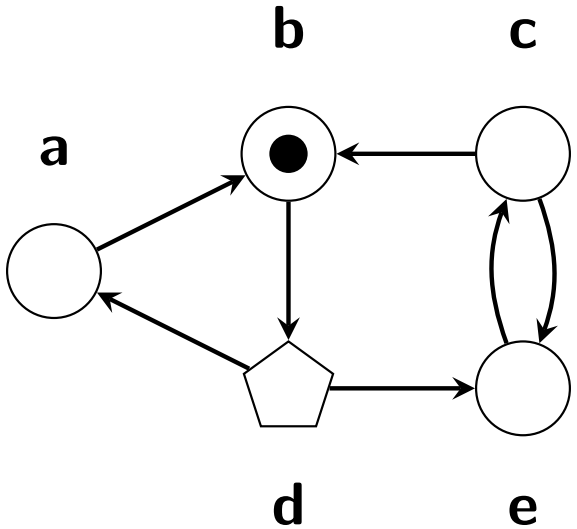
- A parity game is played on a directed graph
- Two players: Even ○ and Odd ⬠
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



The play π: **a b d e c e c**

Parity Games

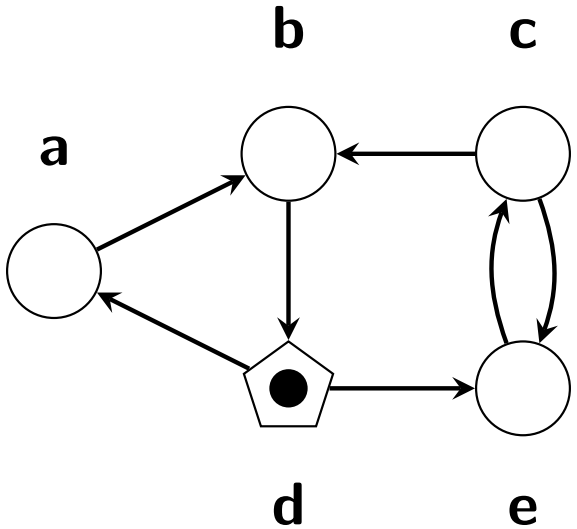
- A parity game is played on a directed graph
- Two players: Even ○ and Odd ⬠
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



The play π: **a b d e c e c b**

Parity Games

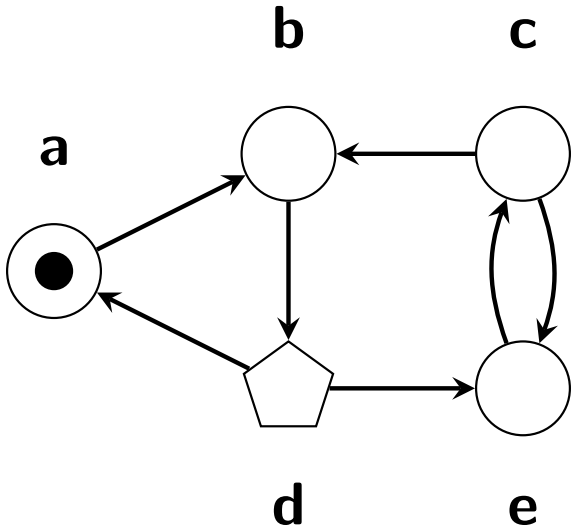
- A parity game is played on a directed graph
- Two players: Even ○ and Odd ⬠
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



The play π: **a b d e c e c b d**

Parity Games

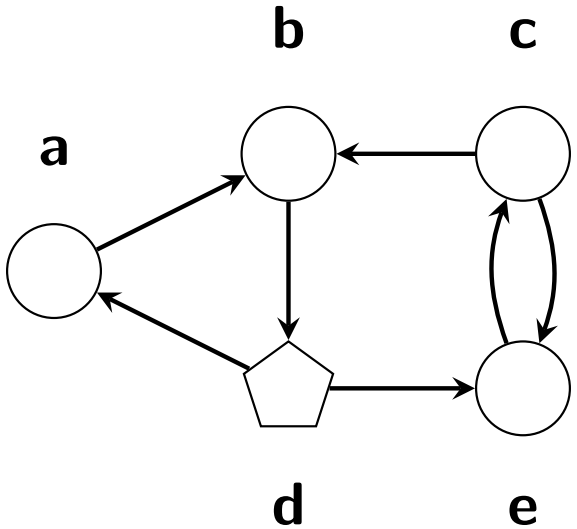
- A parity game is played on a directed graph
- Two players: Even ○ and Odd ⬠
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



The play π: **a b d e c e c b d a**

Parity Games

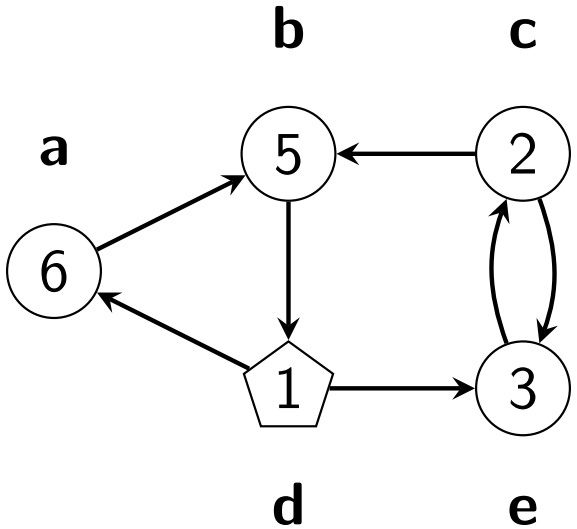
- A parity game is played on a directed graph
- Two players: Even ○ and Odd ⬠
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



How do we determine who wins a play?

Parity Games

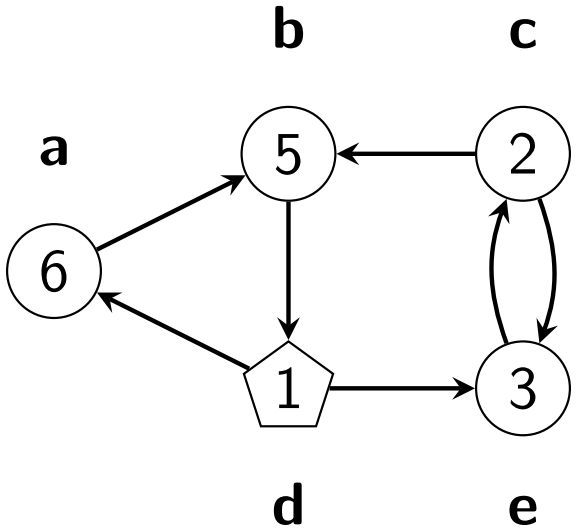
- A parity game is played on a directed graph
- Two players: Even ○ and Odd ⬠
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



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

Parity Games

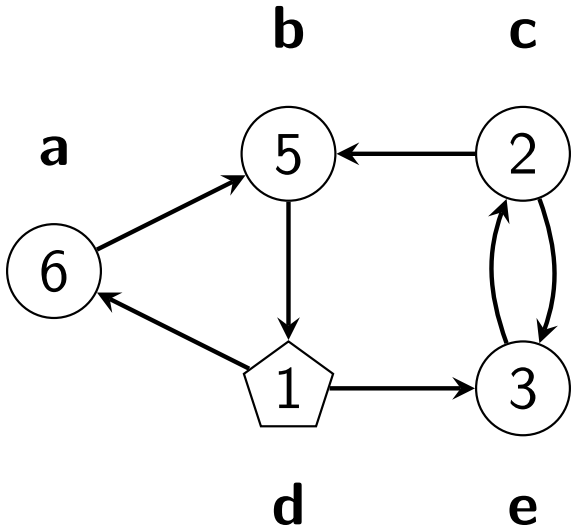
- A parity game is played on a directed graph
- Two players: Even ○ and Odd ⬠
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



How do we determine who wins a vertex?

Parity Games

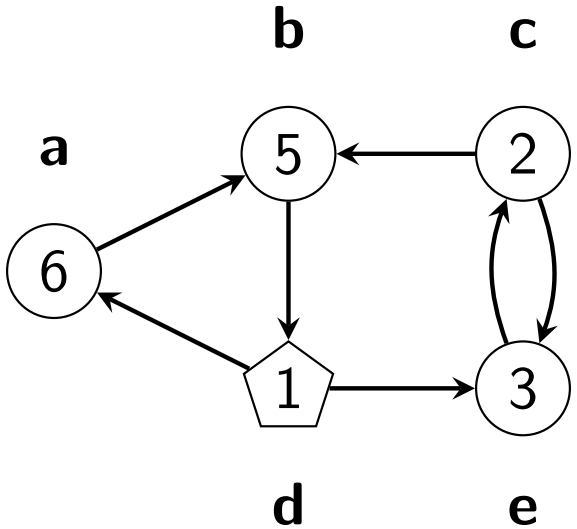
- A parity game is played on a directed graph
- Two players: Even ○ and Odd ⬠
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



A player wins if they have a strategy
such that they win all plays consistent with the strategy.
All cycles in the graph induced by the strategy are Even/Odd.

Parity Games

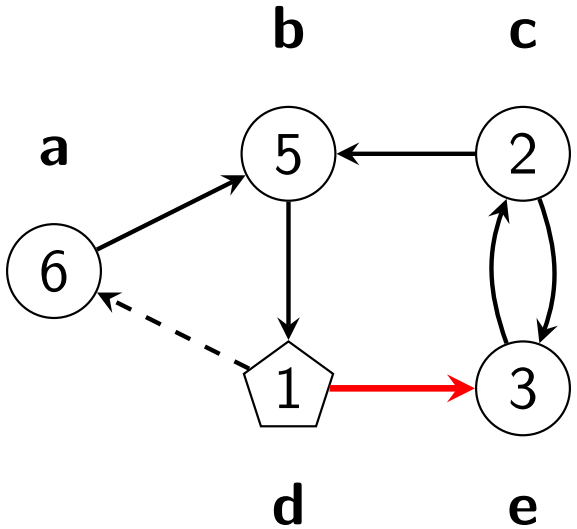
- A parity game is played on a directed graph
- Two players: Even ○ and Odd ⬠
- The players move a token along the edges of the graph
- Each vertex is owned by one player who chooses a successor



Which vertices are won by which player?

Parity Games

- A parity game is played on a directed graph
- Two players: Even ○ and Odd ⬠
- 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 $\{\mathbf{d} \to \mathbf{e}\}$

Parity Games

Basic facts of parity games

- Partition into $W_{\bigcirc}$ (won by Even) and $W_{\pentagon}$ (won by Odd)
- The winner α has a positional strategy $\sigma_\alpha : V_\alpha \to V$
- All plays consistent in W_α with σ_α are won by α
  That is: player α wins all cycles in $W_\alpha[\sigma_\alpha]$

Solving a parity game

- Compute the winning regions $W_{\bigcirc}$ and $W_{\pentagon}$
- Compute the winning strategies $\sigma_{\bigcirc}$ and $\sigma_{\pentagon}$

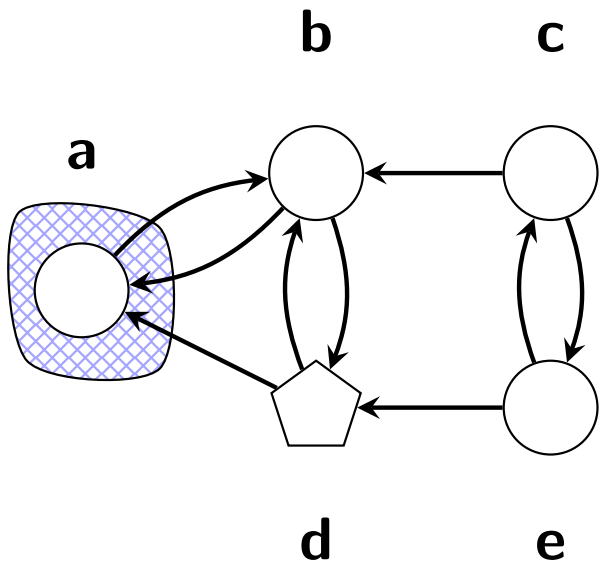
Parity Games

Attractor computation

- "Reachability but in a game"
- Compute all vertices from which player $\alpha \in \{\bigcirc, \pentagon\}$ can ensure arrival in a given target set
- Start with the target set Z, then iteratively add:
  - all vertices of α that can play to Z
  - all vertices of $\overline{\alpha}$ that can only play to Z

Example of attractor computation

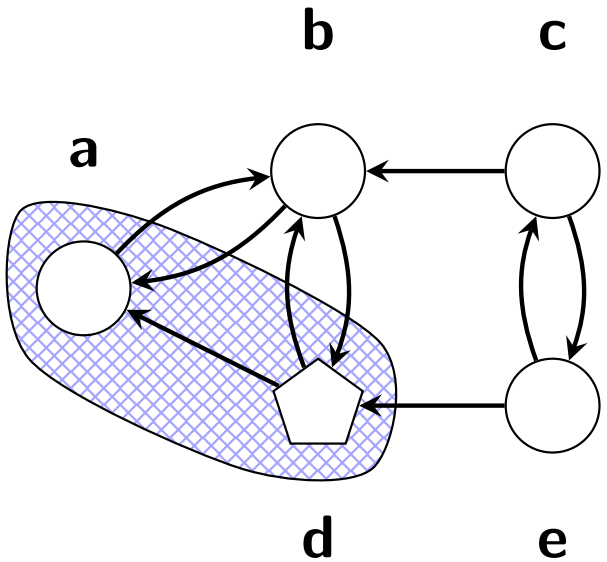
Computing the ⬠-attractor to **a**



Initial set: $\{\mathbf{a}\}$
Can attract: **d** but not **b**

Example of attractor computation

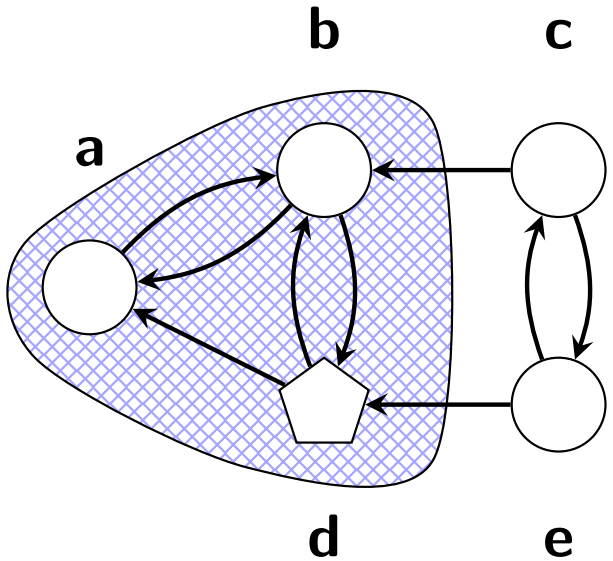
Computing the ⬠-attractor to **a**



Current set: $\{\mathbf{a}, \mathbf{d}\}$
Can attract: **b** but not **e**

Example of attractor computation

Computing the ⬠-attractor to **a**



Current set: $\{\mathbf{a}, \mathbf{b}, \mathbf{d}\}$
Can attract: neither **c** nor **e**

Tangles

Definition

A tangle is

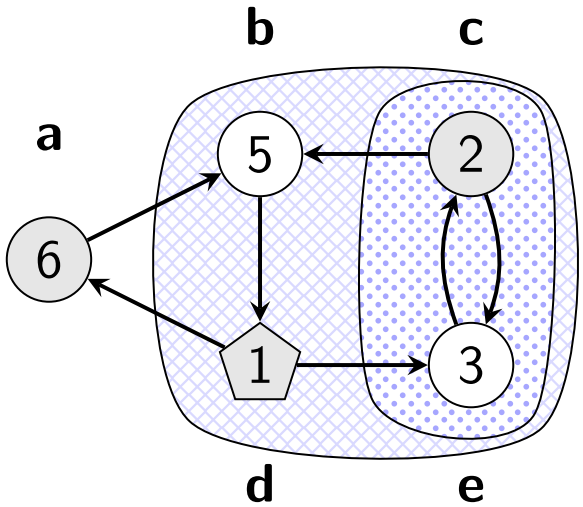
- a pair $T = (U, \sigma)$ where
  - $U \subseteq V$ is a nonempty set of vertices
  - $\sigma : U_\alpha \to U$ is a strategy for player $\alpha := \mathsf{pr}(U) \bmod 2$

  such that
- player α wins all cycles in the induced subgame $\mathcal{Z}[U, \sigma]$
- the induced subgame $\mathcal{Z}[U, \sigma]$ is strongly connected

Tangles

Tangle

A tangle is a strongly connected subgame for which one player has a strategy to win all cycles in the subgame.



A 5-dominion with a 5-tangle and a 3-tangle

Tangles

Tangle

A tangle is a strongly connected subgame for which one player has a strategy to win all cycles in the subgame.

Properties

- Player $\overline{\alpha}$ **must** escape (or lose inside the tangle)
- Player $\overline{\alpha}$ can reach all vertices of the tangle
  (and can thus choose which escape to take)
- Tangles can be nested (subtangles when player $\overline{\alpha}$ can avoid vertices)

Tangles and dominions (winning regions)

- A closed tangle (no escapes) is a dominion
- A dominion decomposes into a hierarchy of tangles

Tangles

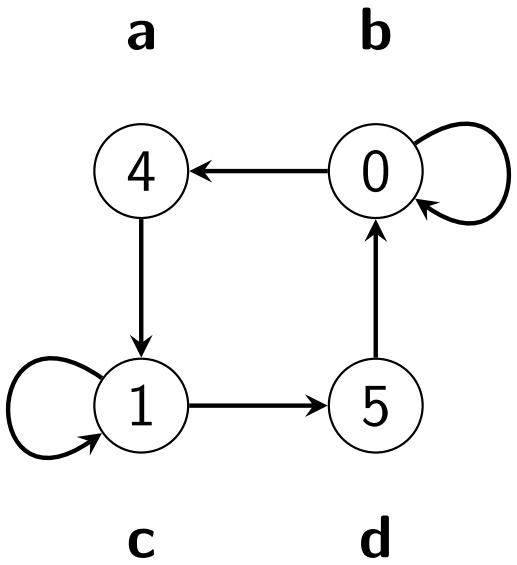
Tangles affect every algorithm

- If the algorithm concludes that the opponent must play from A to B, but the opponent could avoid A by staying in some subgame, then this subgame must contain tangles.
  1. The opponent avoids some high value region
  2. Then finds that staying in a lower region is losing
  3. Chooses the lowest (least bad) escape
- Algorithms often explore the same tangle many times!

Examples

- Recursive algorithm: the lower regions won by the opponent
- Priority promotion: the closed regions contain tangles
- Strategy iteration: "best response" stays in tangles
- Progress measures: slowly increase value until an escape is taken

Distractions



- **Find the winning strategy for player Even.**
- A random vertex inside the winning region?
  If you play from **c** to **c** you lose.
- Always play to the highest reachable even vertex?
  If you play from **b** to **a** you lose.
- Vertex **a** is a distraction for player Even

Distractions

Intuition

A distraction for α is a vertex with an α-priority that the opponent $\overline{\alpha}$ can win if player α always tries to visit it.

Player α must *sometimes/always* avoid the distraction.

Definition

A distraction for player α is a vertex v with an α-priority p, such that if player α always plays to reach v along paths of priorities $\leq p$, then player $\overline{\alpha}$ wins v and all vertices that reach v.

(Along these paths to v, player $\overline{\alpha}$ either must or chooses to play to v.)

Distractions

Distraction

A distraction for α is a vertex with an α-priority that the opponent $\overline{\alpha}$ can win if player α always tries to visit it.

Player α must *sometimes/always* avoid the distraction.

Two types

1. Player $\overline{\alpha}$ can attract v **to higher $\overline{\alpha}$-priorities**.
   Every play that infinitely often visits v, also infinitely often visits a vertex with a higher $\overline{\alpha}$-priority (or stays in a losing tangle).

   Sometimes or always avoid.

2. Player $\overline{\alpha}$ can attract v **to an $\overline{\alpha}$-dominion**.
   Every play that visits v reaches the $\overline{\alpha}$-dominion.
   Distractions inside this $\overline{\alpha}$-dominion are Type 1!

   Always avoid.

Distractions

Distraction

A distraction for α is a high value vertex v with an α-priority that player $\overline{\alpha}$ can win if player α always tries to visit it.

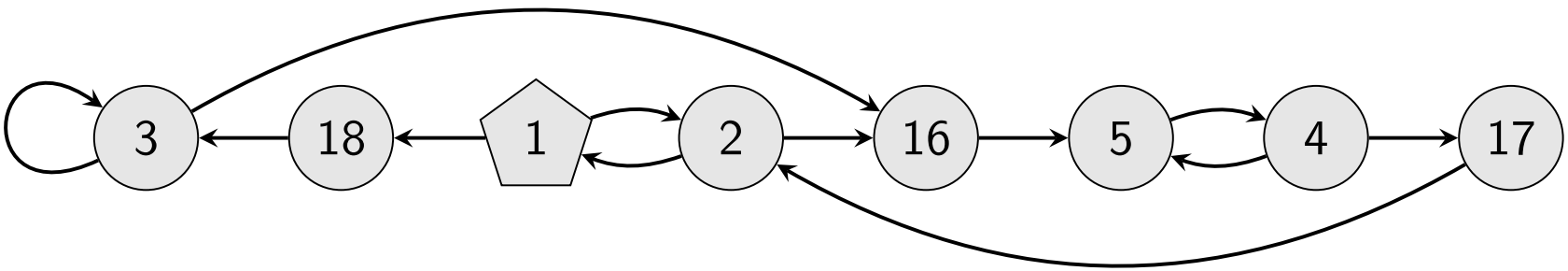
More distracting terminology

- Trivial distractions
  The opponent attracts v to higher $\overline{\alpha}$-priority vertices directly
- Non-trivial distractions
  The opponent attracts v to higher $\overline{\alpha}$-priority vertices via tangles
- Devious distractions
  A distraction for α that is actually in the winning region of α

  Player α must *sometimes* avoid v.
- Fatal distractions

  Player α must *always* avoid v.

Distractions



Vertex **16** is a non-trivial, devious distraction.
Play $\mathbf{2} \to \mathbf{1}$ instead of $\mathbf{2} \to \mathbf{16}$ to win!
But also play $\mathbf{3} \to \mathbf{16}$!

Distractions

Three way to identify distractions

Every algorithm must identify the distractions and remember the strategy to defeat devious distractions.

1. By showing that the opponent $\overline{\alpha}$ attracts the distraction.
   (Zielonka recursive, priority promotion, basic tangle learning)
   - Requires identifying lower distractions before higher distractions
   - Lower distractions attract again after identifying higher distraction
2. By preferring vertices with a better *progressive path value*.
   (Strategy iteration, small progress measures, progressive tangle learning)
   - Requires identifying lower distractions before higher distractions
   - Lower distractions attract again after identifying higher distraction
3. Same as 2, but using distance-to-$\overline{\alpha}$ instead of priority.
   (The quasi-polynomial algorithms)
   - Distractions are attracted to $\overline{\alpha}$'s regions
   - Related: how often the player dominates along the progressive path.
   - Does not require identifying lower distractions before higher distractions

Tangles and distractions

Quasi-polynomial algorithms and distractions

- Look at "how many times α dominates beyond this vertex"
- Similar to "distance to opponent region"
- Also a path-progressive measure, but stronger

Shortcomings of the quasi-polynomial algorithms

- They are not aware of tangles (!!)
- They are short-sighted, not aware of even trivial distractions.
  (they do not identify distractions based on opponent attraction)
- They seem to involve a LOT of repetition
- They solve the entire game, instead of returning a dominion.
- **So there is still hope!**