Algorithms for Parity Games

Piotr Danilewski

May 15, 2008

- 4 聞 と 4 臣 と 4 臣 と

э

Parity Games

Why? Definition Winning condition

When can we win?

General observations Strategy representation Winning sets

Algorithms

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Why? Definition Winning condition

Practical use of Parity Games

- Modal µ-calculus model checking
- Synthesis and satisfiability checking for reactive systems
- Module checking

< 一型

< ∃ >

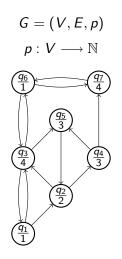
Why? Definition Winning condition

WHAT ARE PARITY GAMES?

*ロ * * @ * * 注 * * 注 *

Why? Definition Winning condition

Parity Graph



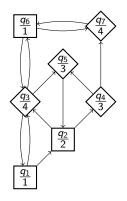
イロン イロン イヨン イヨン

Why? Definition Winning condition

Parity Game

Nodes assigned to players $\overline{\mathbb{A}}$ and $\widehat{\mathbb{B}}$.

 $V = V_A \cup V_B$



< ロ > < 同 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ >

э

Why? Definition Winning condition

Play

Play - infinite path $\pi = (v_0, v_1, v_2,)$

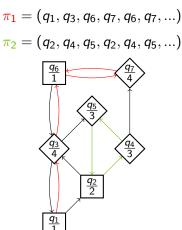


Image: Image:

- ∢ ≣ ▶

Why? Definition Winning condition

Winning condition

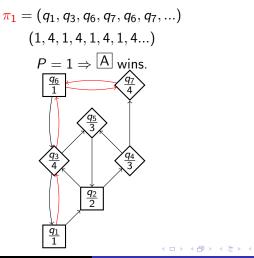
Let P denote minimal priority which repeats itself infinitely often.

- ▶ If P is odd then player A wins.
- If P is even then player \widehat{B} wins.

< ∃ →

Why? Definition Winning condition

Winning condition

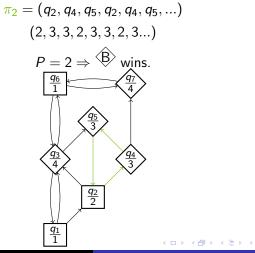


э

∃ >

Why? Definition Winning condition

Winning condition



Piotr Danilewski

Algorithms for Parity Games

General observations Strategy representation Winning sets

WHEN CAN WE WIN?

Piotr Danilewski Algorithms for Parity Games

General observations Strategy representation Winning sets

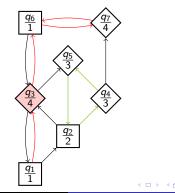
Ideal playing

We assume players do not make mistakes.

• π_1 is invalid under this assumption.

At vertex q_3 player B should have chosen q_5 .

▶ π_2 is valid. Choosing q_3 at q_2 would not help player A.



General observations Strategy representation Winning sets

Memoryless property

We do not have to know how we reached certain vertex in order to deduct how to play.

< 日 > < 同 > < 三 > < 三 >

э

General observations Strategy representation Winning sets

Memoryless strategy representation

Strategy does not change over time.

$$s_A : V_A \longrightarrow V$$

 $s_B : V_B \longrightarrow V$

 s_A , s_B point to successor picked by players A and B respectively.

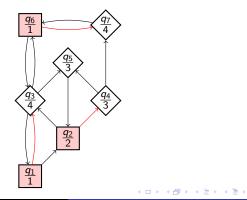
< 17 ▶

A =
 A =
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A
 A

General observations Strategy representation Winning sets

Memoryless strategy representation

$$s_{A}(v) := \left\{ egin{array}{l} q_{3} \ {\it if} \ v = q_{1} \ q_{4} \ {\it if} \ v = q_{2} \ q_{7} \ {\it if} \ v = q_{6} \end{array}
ight.$$

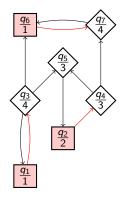


э

General observations Strategy representation Winning sets

Strategy graph

 G_A - Graph G were edges outgoing from V_A are limited to only those chosen by s_A .



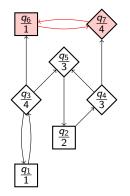
< 17 ▶

→ Ξ → < Ξ</p>

General observations Strategy representation Winning sets

Winning condition in strategy graph

Player A wins if for given vertex v all reachabe cycles in G_A are odd.



< 17 ▶

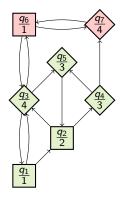
< ∃ >

-

General observations Strategy representation Winning sets

Winning sets partition

Every parity game graph can be partitioned into winning sets W_A and W_B .



< 日 > < 同 > < 三 > < 三 >

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Algorithms

*ロト *部ト *注ト *注ト

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

- Enrich graph with additional information and deduct the best strategy
- Choose some strategy and then improve it

< 日 > < 同 > < 三 > < 三 >

3

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

NAIVE ALGORITHM

・ロト ・回ト ・モト ・モト

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

The naive algorithm

- Consider all possible strategies s_A .
- Consider all possible counter-strategies s_B .
- Pick the best s_A

< □ > < 同 >

→ 3 → 4 3

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Time complexity of the naive algorithm

$$O\left(\prod_{v\in V}\deg v\right)$$

 $O\left(|V|^{|V|}\right)$

In case of full graph:

< 日 > < 同 > < 三 > < 三 >

э

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

JURDZINSKI'S ALGORITHM

*ロ * * @ * * 注 * * 注 *

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Parity Progress Measure

Store additional tuple \mathbb{N}^{c+1} at each vertex where c is maximal priority.

Comparison operators: $<_i, \leq_i, =_i, \geq_i, >_i$ - lexicographic operators on i + 1 first elements.

 $(2,3,0,0) >_1 (2,2,4,1)$ $(2,3,0,0) =_0 (2,2,4,1)$ $(0,1,0,0) <_1 (1,0,0,0)$

イロト イポト イラト イラト

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Parity Progress Measure

Function
$$\rho: V \longrightarrow \mathbb{N}^{c+1}$$
 is a *Parity Progress Measure* if:

$$\forall (v, w) \in E : \rho(v) \geq_{\rho(v)} \rho(w)$$

and if p(v) is odd

$$\forall (v, w) \in E : \rho(v) >_{\rho(v)} \rho(w)$$

<ロ> <同> <同> < 同> < 同>

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Parity Progress Measure

If Parity Progress Measure ρ exists then graph ${\it G}$ must have only even cycles.

Otherwise, let *i* be minimal odd priority in some cycle. Then:

$$\rho(\mathbf{v}_1) >_i \rho(\mathbf{v}_2) \geq_i \rho(\mathbf{v}_3) \dots \geq_i \rho(\mathbf{v}_1)$$

< 日 > < 同 > < 三 > < 三 >

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Parity Progress Measure

$$V_i := p^{-1}(i)$$

$M_G := \mathbb{N}_0 \times \mathbb{N}_{|V_1|} \times \mathbb{N}_0 \times \mathbb{N}_{|V_3|} \times \mathbb{N}_0 \times ... \times \mathbb{N}_{|V_c|}$

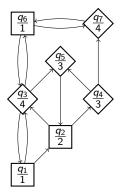
< 日 > < 同 > < 三 > < 三 >

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Codomain restriction

$$\left\{ \begin{array}{l} V_0 := \emptyset \\ V_1 := \{q_1, q_6\} \\ V_2 := \{q_2\} \\ V_3 := \{q_4, q_5\} \\ V_4 := \{q_3, q_7\} \end{array} \right.$$

 $M_G = \mathbb{N}_0 \times \mathbb{N}_2 \times \mathbb{N}_0 \times \mathbb{N}_2 \times \mathbb{N}_0$



<ロト <部ト < 注ト < 注ト

э

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Game Parity Progress Measure

So far - we worked on graphs with even cycles only. Now - we want to include all graphs and vertex assignment to players.

Add highest element T:

$$M_G^T = M_G \cup \{T\}$$

 ${\mathcal T}$ means we cannot fit any other value because we reach an odd cycle

< A

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Game Parity Progress Measure

Let $\rho: V \longrightarrow M_G^T$ be any function. Small progress function: $Prog(\rho, v, w) := \text{least } m \in M_G^T : m \ge_{p(v)} \rho(w)$, and inequality must be strict if p(v) is odd.

$$(0,1,0,0,0)$$
 $\frac{q_6}{1}$ $(0,1,0,1,0)$

 $Prog(\rho, q_6, q_7) = \{ \text{ least } m >_1 (0, 1, 0, 1, 0) \} = (0, 2, 0, 0, 0)$

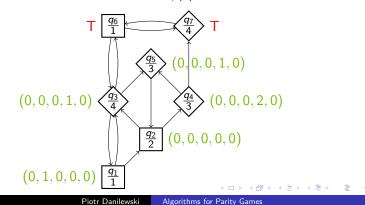
< 日 > < 同 > < 三 > < 三 >

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Game Parity Progress Measure

Game Parity Progress Measure is a function $\rho: V \longrightarrow M_G^T$ such that for all $v \in V$:

▶ $v \in V_A \Rightarrow \forall (v, w) \in E : \rho(v) \ge_{\rho(v)} Prog(\rho, v, w)$ ▶ $v \in V_B \Rightarrow \exists (v, w) \in E : \rho(v) \ge_{\rho(v)} Prog(\rho, v, w)$

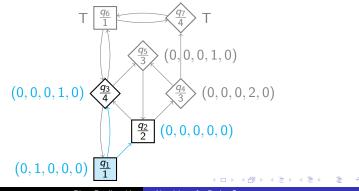


General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Game Parity Progress Measure

Game Parity Progress Measure is a function $\rho: V \longrightarrow M_G^T$ such that for all $v \in V$:

- $\blacktriangleright v \in V_A \Rightarrow \forall (v, w) \in E : \rho(v) \geq_{p(v)} Prog(\rho, v, w)$
- ► $v \in V_B \Rightarrow \exists (v, w) \in E : \rho(v) \ge_{\rho(v)} Prog(\rho, v, w)$

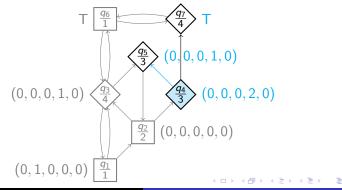


General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Game Parity Progress Measure

Game Parity Progress Measure is a function $\rho: V \longrightarrow M_G^T$ such that for all $v \in V$:

- $\blacktriangleright v \in V_A \Rightarrow \forall (v, w) \in E : \rho(v) \ge_{\rho(v)} Prog(\rho, v, w)$
- ► $v \in V_B \Rightarrow \exists (v, w) \in E : \rho(v) \ge_{\rho(v)} Prog(\rho, v, w)$



General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Strategy from Game Parity Progress Measure

Given ρ player B forms strategy $\textit{s}_{\textit{B}}$ by minimalising its value.

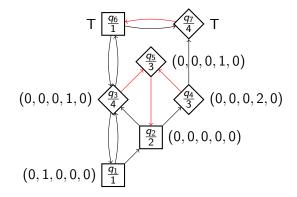
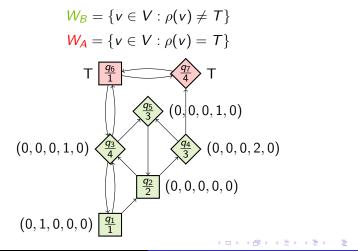


Image: Image:

→ Ξ →

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Winning sets from Game Parity Progress Measure If ρ is a *minimal* Game Parity Progress Measure:



General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Computing the minimal Game Parity Progress Measure

- Start by assigning $\rho(v) := (0, 0, ..., 0)$ to all vertices
- Increment each vertex which violates the Game Parity Progress Measure constraints

$$Lift(\rho, v)(u) := \begin{cases} \rho(u) \Leftarrow u \neq v \\ \max(\rho(v), \min_{(v,w) \in E} \operatorname{Prog}(\rho, v, w)) \Leftarrow u = v \in V_B \\ \max(\rho(v), \max_{(v,w) \in E} \operatorname{Prog}(\rho, v, w)) \Leftarrow u = v \in V_A \end{cases}$$

(日) (同) (三) (三)

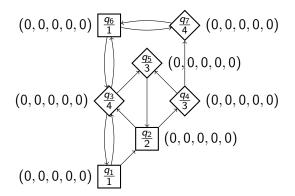
General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

RUN OF JURDZINSKI'S ALGORITHM

æ

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Run of Jurdzinski's algorithm (step 0)



< ロ > < 同 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ >

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Run of Jurdzinski's algorithm (step 1)

$$ift(\rho, q_1)(q_1) = \max\left((0, 0, 0, 0, 0), \max_{(v,w) \in E} Prog(\rho, q_1, w)\right)$$

$$Prog(\rho, q_1, w) := \text{least } m \in M_G^T : m >_1 \rho(w)$$

$$(0, 0, 0, 0, 0) \xrightarrow{q_1} (0, 0, 0, 0, 0, 0)$$

$$(0, 0, 0, 0, 0) \xrightarrow{q_2} (0, 0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_1} (0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_1} (0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_1} (0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_1} (0, 0, 0, 0, 0)$$

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Run of Jurdzinski's algorithm (step 2)

$$Lift(\rho, q_{5})(q_{5}) = \max\left((0, 0, 0, 0, 0), \min_{(v,w)\in E} Prog(\rho, q_{5}, w)\right)$$

$$Prog(\rho, q_{5}, w) := \text{least } m \in M_{G}^{T} : m >_{3} \rho(w)$$

$$(0, 0, 0, 0, 0) \xrightarrow{q_{6}} (0, 0, 0, 0, 0, 0)$$

$$(0, 0, 0, 0, 0) \xrightarrow{q_{3}} (0, 0, 0, 1, 0)$$

$$(0, 0, 0, 0, 0) \xrightarrow{q_{3}} (0, 0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 0, 0, 0, 0)$$

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Run of Jurdzinski's algorithm (step 3)

$$Lift(\rho, q_{6})(q_{6}) = \max\left((0, 0, 0, 0, 0), \max_{(v,w)\in E} Prog(\rho, q_{6}, w)\right)$$

$$Prog(\rho, q_{6}, w) := \text{least } m \in M_{G}^{T} : m >_{1} \rho(w)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{6}} (1, 0, 0, 0, 0) \xrightarrow{q_{7}} (0, 0, 0, 0, 0, 0)$$

$$(0, 0, 0, 0, 0) \xrightarrow{q_{3}} (0, 0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 0, 0, 0, 0)$$

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Run of Jurdzinski's algorithm (step 4)

$$Lift(\rho, q_{7})(q_{7}) = \max\left((0, 0, 0, 0, 0), \min_{(v,w)\in E} Prog(\rho, q_{7}, w)\right)$$

$$Prog(\rho, q_{7}, w) := \text{least } m \in M_{G}^{T} : m \geq_{4}\rho(w)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{6}} (0, 1, 0, 0, 0)$$

$$(0, 0, 0, 0, 0) \xrightarrow{q_{3}} (0, 0, 0, 1, 0)$$

$$(0, 0, 0, 0, 0) \xrightarrow{q_{3}} (0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 1, 0, 0, 0)$$

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Run of Jurdzinski's algorithm (step 5)

$$Lift(\rho, q_{4})(q_{4}) = \max\left((0, 0, 0, 0, 0), \min_{(v,w) \in E} Prog(\rho, q_{4}, w)\right)$$

$$Prog(\rho, q_{4}, w) := \text{least } m \in M_{G}^{T} : m >_{3}\rho(w)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{6}} (0, 1, 0, 0, 0, 0)$$

$$(0, 0, 0, 0, 0) \xrightarrow{q_{5}} (0, 0, 0, 1, 0)$$

$$(0, 0, 0, 0, 0) \xrightarrow{q_{3}} (0, 0, 0, 0, 2, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 0, 0, 0, 0, 0)$$

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Run of Jurdzinski's algorithm (step 6)

$$ift(\rho, q_{3})(q_{3}) = \max\left((0, 0, 0, 0, 0), \min_{(v,w)\in E} Prog(\rho, q_{3}, w)\right)$$

$$Prog(\rho, q_{3}, w) := \text{least } m \in M_{G}^{T} : m \geq_{4}\rho(w)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{5}} (0, 0, 0, 1, 0)$$

$$(0, 0, 0, 1, 0) \xrightarrow{q_{3}} (0, 0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 0, 0, 0, 0, 0)$$

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Run of Jurdzinski's algorithm (step 7)

$$Lift(\rho, q_{6})(q_{6}) = \max \left((0, 1, 0, 0, 0), \max_{(v,w) \in E} Prog(\rho, q_{6}, w) \right)$$

$$Prog(\rho, q_{6}, w) := \text{ least } m \in M_{G}^{T} : m >_{1} \rho(w)$$

$$(0, 2, 0, 0, 0) \xrightarrow{q_{6}} (q_{7}) (0, 1, 0, 0, 0)$$

$$(0, 0, 0, 1, 0) \xrightarrow{q_{5}} (0, 0, 0, 1, 0)$$

$$(0, 0, 0, 1, 0) \xrightarrow{q_{2}} (0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 1, 0, 0, 0)$$

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Run of Jurdzinski's algorithm (step 8)

$$ift(\rho, q_{7})(q_{7}) = \max\left((0, 1, 0, 0, 0), \min_{(v,w)\in E} Prog(\rho, q_{7}, w)\right)$$

$$Prog(\rho, q_{7}, w) := \text{least } m \in M_{G}^{T} : m \geq_{4} \rho(w)$$

$$(0, 2, 0, 0, 0) \xrightarrow{q_{5}} (0, 0, 0, 1, 0)$$

$$(0, 0, 0, 1, 0) \xrightarrow{q_{3}} (0, 0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 0, 0, 0, 0)$$

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Run of Jurdzinski's algorithm (step 9)

$$ift(\rho, q_{6})(q_{6}) = \max\left((0, 2, 0, 0, 0), \max_{(v,w)\in E} Prog(\rho, q_{6}, w)\right)$$

$$M_{G}^{T} = \mathbb{N}_{0} \times \mathbb{N}_{2} \times \mathbb{N}_{0} \times \mathbb{N}_{2} \times \mathbb{N}_{0} \cup \{T\}$$

$$T \xrightarrow{q_{6}} (0, 2, 0, 0, 0)$$

$$(0, 0, 0, 1, 0) \xrightarrow{q_{3}} (0, 0, 0, 1, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 0, 0, 0, 0)$$

$$(0, 1, 0, 0, 0) \xrightarrow{q_{1}} (0, 0, 0, 0, 0)$$

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

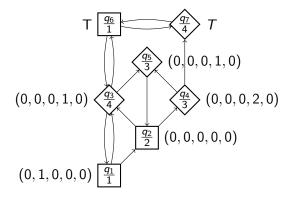
Run of Jurdzinski's algorithm (step 10)

$$ift(\rho, q_{7})(q_{7}) = \max\left((0, 2, 0, 0, 0), \min_{(v,w)\in E} Prog(\rho, q_{7}, w)\right)$$
$$M_{G}^{T} = \mathbb{N}_{0} \times \mathbb{N}_{2} \times \mathbb{N}_{0} \times \mathbb{N}_{2} \times \mathbb{N}_{0} \cup \{T\}$$
$$T \xrightarrow{q_{6}} (1, 0, 0, 0, 1, 0) \xrightarrow{q_{3}} (0, 0, 0, 1, 0) \xrightarrow{q_{3}} (0, 0, 0, 0, 0, 0) \xrightarrow{q_{3}} (0, 0, 0, 0, 0) \xrightarrow{q_{3}} (0, 0, 0, 0, 0, 0) \xrightarrow{q_{3}} (0, 0, 0, 0, 0, 0) \xrightarrow{q_{3}} (0, 0, 0, 0) \xrightarrow{q_{3}} (0, 0, 0, 0, 0) \xrightarrow{q_{3}} (0, 0, 0) \xrightarrow{q_{3}} (0, 0, 0, 0) \xrightarrow{q_$$

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Run of Jurdzinski's algorithm (step 11)

Observe Lift operation cannot perform any more changes. End of run.



< 17 >

- ₹ 🖹 🕨

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Space complexity of Jurdzinski's algorithm

O(c |V|)

Piotr Danilewski Algorithms for Parity Games

æ

<ロト <部ト < 注ト < 注ト

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms Complexity

Time complexity of Jurdzinski's algorithm

Time per single lift operation: $O(c \deg v)$ Total time:

$$O(\sum_{v \in V} |M_G| c \deg v) = O(c |E| |M_G|)$$

$$|M_{\mathcal{G}}| = \prod_{i=1}^{\left\lceil \frac{c}{2} \right\rceil} (|V_{2i-1}| + 1) < \left(\frac{\sum_{i=1}^{\left\lceil \frac{c}{2} \right\rceil} (|V_{2i-1}| + 1)}{\left\lceil \frac{c}{2} \right\rceil} \right)^{\left\lceil \frac{c}{2} \right\rceil} \le \left(\frac{|V|}{\left\lceil \frac{c}{2} \right\rceil} \right)^{\left\lceil \frac{c}{2} \right\rceil}$$

Finally

$$O\left(c\left|E\right|\left(\frac{|V|}{\left\lceil\frac{c}{2}\right\rceil}\right)^{\left\lceil\frac{c}{2}\right\rceil}\right)$$

(日) (同) (三) (三)

General approaches Naive algorithm Jurdzinski's algorithm **Other algorithms** Complexity

There are other algorithms, for example:

- McNaughton's algorithm
- Sven's algorithm

(日) (同) (三) (三)

э

General approaches Naive algorithm Jurdzinski's algorithm Other algorithms **Complexity**

We know that Parity Games problem

- ▶ is NP \cap co-NP
- ▶ is UP \cap co-UP
- it is unlikely to be NP-complete
- it is not known to be P

イロト イポト イヨト イヨト

э