

Algorithms for Model Checking (2IW55)

Lecture 13

The Small Progress Measures algorithm for Parity games

Background material:

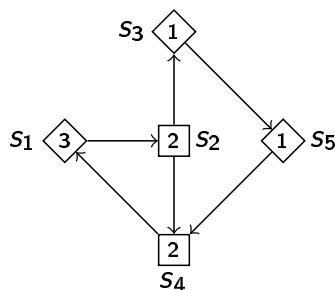
“Small Progress Measures for Solving Parity Games”, Marcin Jurdzinski

January 7, 2011

Parity games (recap)

2/19

Identify graph, priorities, owners, plays, and strategies in the following parity game.



Goal: compute winning sets

Relevant concepts:

- ▶ Divide and conquer
- ▶ Base: empty game
- ▶ Step: assemble winning sets/strategies from
 - winning sets/strategies of subgames
 - attractor strategy for one of players reaching set of nodes with minimal priority in the game
- ▶ Attractor set

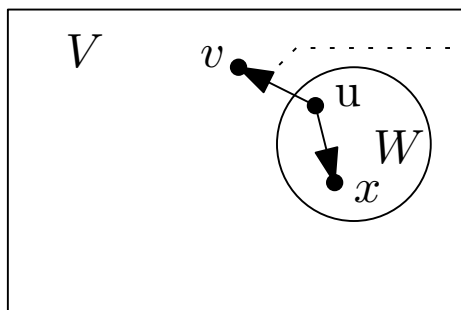
- ▶ Measure assigned to vertex contains, for odd priorities, maximal number of times this priority can be seen if player \square moves over the graph, until a vertex with smaller priority is seen
- ▶ Measure computed using fixed point iteration
- ▶ Characterise cycles reachable from each vertex
- ▶ Cycles can be used to decide winners

Let $G = (V, E, p, (V_\diamond, V_\square))$ be a parity game. A strategy $\varrho_\diamond: V_\diamond \rightarrow V$ is **closed** on a set $W \subseteq V$ if for all $v \in W$, we have:

- ▶ if $v \in V_\diamond$ then $\varrho_\diamond(v) \in W$, and
- ▶ if $v \in V_\square$ then $(v, w) \in E$ implies $w \in W$.

Each play **consistent** with strategy ϱ_\diamond **closed** on W , starting in W , **stays within W**

Illustration of closedness



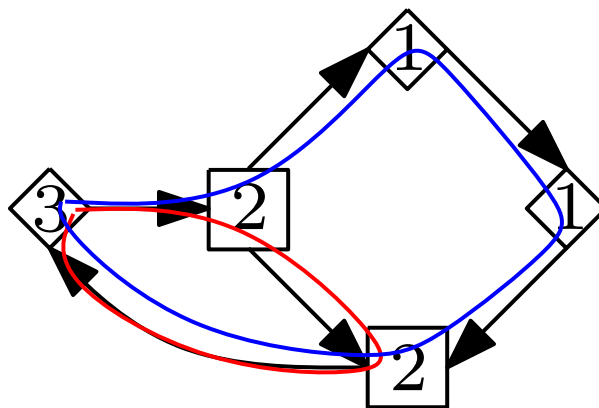
Edges (u, v) only for $u \in V_{\text{Even}}$, and only if there also is edge (u, x) for $x \in W$.

A **cycle** in a parity game is a path v_1, v_2, \dots, v_n , with $v_1 = v_n$

We say that a cycle v_1, v_2, \dots, v_n is

- ▶ an ***i*-cycle** if $i = \min\{p(v_j) \mid 1 \leq j \leq n\}$ (i is the smallest priority occurring on the cycle)
- ▶ an **even cycle** if it is an i -cycle, and i is even

Cycle (example)



Let:

- ▶ $G = (V, E, p, (V_\diamond, V_\square))$ be a parity game
- ▶ ϱ_\diamond be a strategy for player \diamond , **closed** on $W \subseteq V$

Define $G_{\varrho_\diamond}^W = (V', E, p, (V'_\diamond, V'_\square))$

- ▶ $V' = W$
- ▶ $V'_\diamond = V_\diamond \cap W$
- ▶ $V'_\square = V_\square \cap W$
- ▶ $E' = \{(v, w) \in E \mid v \in V'_\diamond \wedge w = \varrho_\diamond(v)\} \cup \{(v, w) \in E \mid v \in V'_\square\}$
- ▶ $p'(v) = p(v)$ for $v \in W$

ϱ_\diamond is **winning** for player \diamond from W if and only if **all cycles in $G_{\varrho_\diamond}^W$ are even**

- ▶ Measure assigned to vertex contains, for odd priorities, maximal number of times this priority can be seen if player \square moves over the graph, until a vertex with smaller priority is seen
- ▶ Measure computed using fixed point iteration
- ▶ **Characterise cycles reachable from each vertex**
- ▶ **Cycles can be used to decide winners**

Let $\alpha \in \mathbb{N}^d$ be a d -tuple of natural numbers

- ▶ we number its components from 0 to $d - 1$, i.e. $\alpha = (\alpha_0, \alpha_1, \dots, \alpha_{d-1})$,
- ▶ $<, \leq, =, \neq, \geq, >$ on tuples denote **lexicographic ordering**,
- ▶ $(n_0, n_1, \dots, n_k) \equiv_i (m_0, m_1, \dots, m_l)$ iff $(n_0, n_1, \dots, n_i) \equiv (m_0, m_1, \dots, m_i)$, for $\equiv \in \{<, \leq, =, \neq, \geq, >\}$
- ▶ Note that if $i > k$ or $i > l$, the tuples will be suffixed with 0s

- ▶ $(0, 1, 0, 1) =_0 (0, 2, 0, 1) \equiv (0) = (0) \equiv \text{true}$
- ▶ $(0, 1, 0, 1) <_1 (0, 2, 0, 1) \equiv (0, 1) < (0, 2) \equiv \text{true}$
- ▶ $(0, 1, 0, 1) \geq_3 (0, 2, 0, 1) \equiv (0, 1, 0, 1) \geq (0, 2, 0, 1) \equiv \text{false}$

Let $G = (V, E, p, (V_\diamond, V_\square))$ be a parity game, and let $d = \max\{p(v) \mid v \in V\} + 1$.

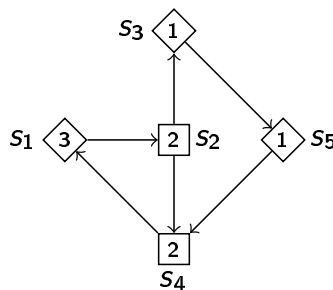
- ▶ For $i \in \mathbb{N}$, let $V_i = \{v \in V \mid p(v) = i\}$,
- ▶ Denote $n_i = |V_i|$, the number of vertices with priority i ,

Define $\mathbb{M}_G \subseteq \mathbb{N}^d$, such that it is the finite set of d -tuples, with:

- ▶ 0 on **even** positions
- ▶ Natural numbers $\leq n_i$ on **odd** positions i

\mathbb{M}_G (example)

Determine maximum value of \mathbb{M}_G for the following parity game:



Maximum value of \mathbb{M}_G is $(0, 2, 0, 1)$

$$\mathbb{M}_G = \{0\} \times \{0, 1, 2\} \times \{0\} \times \{0, 1\}$$

Recall: ϱ_\diamond is winning for player \diamond from W if and only if all cycles in $G_{\varrho_\diamond}^W$ are even

Idea: characterise vertices that can only reach even cycles.

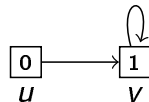
Definition (Parity progress measure)

Let $G = (V, E, p, (V_\diamond, V_\square))$ be a parity game. Function $\varrho: V \rightarrow \mathbb{N}^d$ is a parity progress measure for G if for all $(v, w) \in E$ it holds that:

- ▶ $\varrho(v) \succeq_{p(v)} \varrho(w)$ if $p(v)$ is even
- ▶ $\varrho(v) >_{p(v)} \varrho(w)$ if $p(v)$ is odd

There exists a parity progress measure for G iff all cycles in G are even

Parity progress measure (problem)



Problem: no parity progress measure can be assigned to these vertices, as parity progress measure only exists for even cycles. (Second clause requires $\varrho(v) >_1 \varrho(v)$)

Let $G = (V, E, p, (V_\diamond, V_\square))$ be a parity game.

Define $\mathbb{M}_G^\top = \mathbb{M}_G \cup \{\top\}$, such that:

- ▶ $m \{<, <_i\} \top$ for all $m \in \mathbb{M}_G$, and $m \{\neq, \neq_i\} \top$
- ▶ $\top =_i \top$ for all i .

Definition (Prog)

If $\varrho: V \rightarrow \mathbb{M}_G^\top$ and $(v, w) \in E$, then $Prog(\varrho, v, w)$ is the least $m \in \mathbb{M}_G^\top$, such that

- ▶ $m \geq_{p(v)} \varrho(w)$ if $p(v)$ is even,
- ▶ $m >_{p(v)} \varrho(w)$, or $m = \varrho(w) = \top$ if $p(v)$ is odd.

Prog (examples)

Let $\mathbb{M}_G = \{0\} \times \{0, 1, 2\} \times \{0\} \times \{0, 1\}$

Suppose $p(v) = 0$, $\varrho(w) = (0, 2, 0, 0)$. Then $Prog(\varrho, v, w) = (0, 0, 0, 0)$

Suppose $p(v) = 1$, $\varrho(w) = (0, 2, 0, 0)$. Then $Prog(\varrho, v, w) = \top$

Suppose $p(v) = 3$, $\varrho(w) = (0, 2, 0, 0)$. Then $Prog(\varrho, v, w) = (0, 2, 0, 1)$

Definition (Game parity progress measure)

Let $G = (V, E, p, (V_\diamond, V_\square))$ be a parity game. A function $\varrho: V \rightarrow \mathbb{M}_G^\top$ is a game parity progress measure if for all $v \in V$, it holds that:

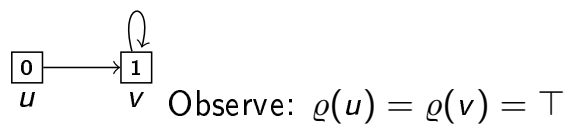
- ▶ if $v \in V_\diamond$, then $\exists_{(v,w) \in E} \varrho(v) \geq_{p(v)} \text{Prog}(\varrho, v, w)$
- ▶ if $v \in V_\square$, then $\forall_{(v,w) \in E} \varrho(v) \geq_{p(v)} \text{Prog}(\varrho, v, w)$

ϱ is least game parity progress measure

\implies

$\varrho(v) \neq \top$ iff all cycles reachable from vertex v are even.

Example



Measure can identify both even and odd reachable cycles.

- ▶ Measure assigned to vertex contains, for odd priorities, maximal number of times this priority can be seen if player \square moves over the graph, until a vertex with smaller priority is seen
- ▶ Measure computed using fixed point iteration
- ▶ Characterise cycles reachable from each vertex
- ▶ Cycles can be used to decide winners

Characterise game parity progress measure as **fixed point** of monotone operators in a finite complete lattice:

- ▶ a **least game parity progress measure** φ exists (Knaster-Tarski),
- ▶ computable by fixed point **iteration** (see Lecture 3, slide 13 for an algorithm),

Let $G = (V, E, p, (V_\diamond, V_\square))$, and $\varphi, \varrho: V \rightarrow \mathbb{M}_G^\top$.

- ▶ $\varphi \sqsubseteq \varrho$ if $\varphi(v) \leq \varrho(v)$ for all $v \in V$
- ▶ write $\varphi \sqsubset \varrho$ if $\varphi \sqsubseteq \varrho$ and $\varphi \neq \varrho$.

\sqsubseteq gives a **complete lattice** structure on the set of functions $V \rightarrow \mathbb{M}_G^\top$.

Define $Lift_v(\varrho)$ for $v \in V$ as follows:

$$Lift_v(\varrho) = \begin{cases} \varrho[v := \min\{Prog(\varrho, v, w) \mid (v, w) \in E\}] & \text{if } v \in V_{\diamond} \\ \varrho[v := \max\{Prog(\varrho, v, w) \mid (v, w) \in E\}] & \text{if } v \in V_{\square} \end{cases}$$

Observe:

- ▶ For every $v \in V$, $Lift_v$ is \sqsubseteq -monotone.
- ▶ A function $\varrho: V \rightarrow \mathbb{M}_G^T$ is a game parity progress measure if and only if $Lift_v(\varrho) \sqsubseteq \varrho$ for all $v \in V$.

The algorithm

Compute **least game parity progress measure** using fixed point approximation:

Algorithm (*ProgressMeasureLifting*)

```

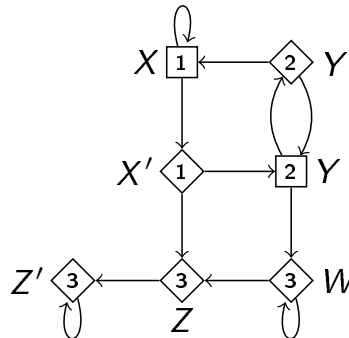
 $\varrho \leftarrow \lambda v \in V. (0, \dots, 0)$ 
while  $\varrho \sqsubset Lift_v(\varrho)$  for some  $v \in V$  do
   $\varrho \leftarrow Lift_v(\varrho)$ 
end while

```

Post condition:

- ▶ ϱ is least game parity progress measure
- ▶ $\{v \in V \mid \varrho(v) \neq \top\}$ is **winning set** for player \diamond

Consider parity game G :



Maximum value of \mathbb{M}_G is $(0, 2, 0, 3)$

Small progress measures (example) (1)

Initially: $\varrho \leftarrow \lambda v \in V. (0, 0, 0, 0)$, so

v	$\varrho(v)$
X	(0, 0, 0, 0)
X'	(0, 0, 0, 0)
Y	(0, 0, 0, 0)
Y'	(0, 0, 0, 0)
Z	(0, 0, 0, 0)
Z'	(0, 0, 0, 0)
W	(0, 0, 0, 0)

Step 2: $\varrho \leftarrow \text{Lift}_{\mathbf{X}}(\varrho) = \varrho[X := \max\{\text{Prog}(\varrho, X, X'), \text{Prog}(\varrho, X, X)\}] = \varrho[X := \max\{(0, 1, 0, 0), (0, 1, 0, 0)\}] = \varrho[X := (0, 1, 0, 0)]$

v	$\varrho(v)$
X	(0, 1, 0, 0)
X'	(0, 0, 0, 0)
Y	(0, 0, 0, 0)
Y'	(0, 0, 0, 0)
Z	(0, 0, 0, 0)
Z'	(0, 0, 0, 0)
W	(0, 0, 0, 0)

Step 3: $\varrho \leftarrow \text{Lift}_{\mathbf{X}}(\varrho) = \varrho[X := \max\{\text{Prog}(\varrho, X, X'), \text{Prog}(\varrho, X, X)\}] = \varrho[X := \max\{(0, 1, 0, 0), (0, 2, 0, 0)\}] = \varrho[X := (0, 2, 0, 0)]$

v	$\varrho(v)$
X	(0, 2, 0, 0)
X'	(0, 0, 0, 0)
Y	(0, 0, 0, 0)
Y'	(0, 0, 0, 0)
Z	(0, 0, 0, 0)
Z'	(0, 0, 0, 0)
W	(0, 0, 0, 0)

Step 4: $\varrho \leftarrow \text{Lift}_{\mathbf{X}}(\varrho) = \varrho[X := \max\{\text{Prog}(\varrho, X, X'), \text{Prog}(\varrho, X, X)\}] = \varrho[X := \max\{(0, 1, 0, 0), \top\}] = \varrho[X := \top]$

v	$\varrho(v)$
X	\top
X'	(0, 0, 0, 0)
Y	(0, 0, 0, 0)
Y'	(0, 0, 0, 0)
Z	(0, 0, 0, 0)
Z'	(0, 0, 0, 0)
W	(0, 0, 0, 0)

Step

5: $\text{Lift}_{\mathbf{Y}'}(\varrho) = \varrho[Y' := \min\{\text{Prog}(\varrho, Y', X), \text{Prog}(\varrho, Y', Y)\}] = \varrho[Y' := \min\{\top, (0, 0, 0, 0)\}] = \varrho[Y' := (0, 0, 0, 0)]$

$\text{Lift}_{\mathbf{Y}}(\varrho) = \varrho[Y := \max\{\text{Prog}(\varrho, Y, W), \text{Prog}(\varrho, Y, Y')\}] = \varrho[Y := \max\{(0, 0, 0, 0), (0, 0, 0, 0)\}] = \varrho[Y := (0, 0, 0, 0)]$

$\varrho \leftarrow \text{Lift}_{\mathbf{X}'}(\varrho) = \varrho[X' := \min\{\text{Prog}(\varrho, X', Y), \text{Prog}(\varrho, X', Z)\}] = \varrho[X' := \min\{(0, 1, 0, 0), (0, 1, 0, 0)\}] = \varrho[X' := (0, 1, 0, 0)]$

v	$\varrho(v)$
X	\top
X'	(0, 1, 0, 0)
Y	(0, 0, 0, 0)
Y'	(0, 0, 0, 0)
Z	(0, 0, 0, 0)
Z'	(0, 0, 0, 0)
W	(0, 0, 0, 0)

Step 6: $\varrho \leftarrow \text{Lift}_{\mathbf{Z}'}(\varrho) = \varrho[Z' := \min\{\text{Prog}(\varrho, Z', Z')\}] = \varrho[Z' := \min\{(0, 0, 0, 1)\}] = \varrho[Z' := (0, 0, 0, 1)]$

v	$\varrho(v)$
X	\top
X'	(0, 1, 0, 0)
Y	(0, 0, 0, 0)
Y'	(0, 0, 0, 0)
Z	(0, 0, 0, 0)
Z'	(0, 0, 0, 1)
W	(0, 0, 0, 0)

Step 7: $\varrho \leftarrow \text{Lift}_{Z'}(\varrho) = \varrho[Z' := \min\{\text{Prog}(\varrho, Z', Z')\}] = \varrho[Z' := \min\{(0, 0, 0, 2)\}] = \varrho[Z' := (0, 0, 0, 2)]$

v	$\varrho(v)$
X	⊥
X'	(0, 1, 0, 0)
Y	(0, 0, 0, 0)
Y'	(0, 0, 0, 0)
Z	(0, 0, 0, 0)
Z'	(0, 0, 0, 2)
W	(0, 0, 0, 0)

Step 8: $\varrho \leftarrow \text{Lift}_{Z'}(\varrho) = \varrho[Z' := \min\{\text{Prog}(\varrho, Z', Z')\}] = \varrho[Z' := \min\{(0, 0, 0, 3)\}] = \varrho[Z' := (0, 0, 0, 3)]$

v	$\varrho(v)$
X	⊥
X'	(0, 1, 0, 0)
Y	(0, 0, 0, 0)
Y'	(0, 0, 0, 0)
Z	(0, 0, 0, 0)
Z'	(0, 0, 0, 3)
W	(0, 0, 0, 0)

Step 9: $\varrho \leftarrow \text{Lift}(\varrho, Z') = \varrho[Z' := \min\{\text{Prog}(\varrho, Z', Z')\}] = \varrho[Z' := \min\{(0, 1, 0, 0)\}] = \varrho[Z' := (0, 1, 0, 0)]$

v	$\varrho(v)$
X	⊥
X'	(0, 1, 0, 0)
Y	(0, 0, 0, 0)
Y'	(0, 0, 0, 0)
Z	(0, 0, 0, 0)
Z'	(0, 1, 0, 0)
W	(0, 0, 0, 0)

Step 10: $\varrho \leftarrow \text{Lift}_{Z'}(\varrho) = \varrho[Z' := \min\{\text{Prog}(\varrho, Z', Z')\}] = \varrho[Z' := \min\{(0, 1, 0, 1)\}] = \varrho[Z' := (0, 1, 0, 1)]$

v	$\varrho(v)$
X	⊥
X'	(0, 1, 0, 0)
Y	(0, 0, 0, 0)
Y'	(0, 0, 0, 0)
Z	(0, 0, 0, 0)
Z'	(0, 1, 0, 1)
W	(0, 0, 0, 0)

Step 11*: Repeat lifting Z' even more often

$$\varrho \leftarrow \text{Lift}_{Z'}(\varrho) = \varrho[Z' := \min\{\text{Prog}(\varrho, Z', Z')\}] = \varrho[Z' := \min\{\top\}] = \varrho[Z' := \top]$$

v	$\varrho(v)$
X	\top
X'	$(0, 1, 0, 0)$
Y	$(0, 0, 0, 0)$
Y'	$(0, 0, 0, 0)$
Z	$(0, 0, 0, 0)$
Z'	\top
W	$(0, 0, 0, 0)$

Step 12: $\varrho \leftarrow \text{Lift}_Z(\varrho) = \varrho[Z := \min\{\text{Prog}(\varrho, Z, Z')\}] = \varrho[Z := \min\{\top\}] = \varrho[Z := \top]$

v	$\varrho(v)$
X	\top
X'	$(0, 1, 0, 0)$
Y	$(0, 0, 0, 0)$
Y'	$(0, 0, 0, 0)$
Z	\top
Z'	\top
W	$(0, 0, 0, 0)$

Step 13:

$$\varrho \leftarrow \text{Lift}_W(\varrho) = \varrho[W := \min\{\text{Prog}(\varrho, W, Z), \text{Prog}(\varrho, W, W')\}] = \varrho[W := \min\{\top, (0, 0, 0, 1)\}] = \varrho[W := (0, 0, 0, 1)]$$

v	$\varrho(v)$
X	\top
X'	$(0, 1, 0, 0)$
Y	$(0, 0, 0, 0)$
Y'	$(0, 0, 0, 0)$
Z	\top
Z'	\top
W	$(0, 0, 0, 1)$

Step 14*: Repeat lifting of W often

$$\varrho \leftarrow \text{Lift}_W(\varrho) = \varrho[W := \min\{\text{Prog}(\varrho, W, Z), \text{Prog}(\varrho, W, W')\}] = \varrho[W := \min\{\top, \top\}] = \varrho[W := \top]$$

v	$\varrho(v)$
X	\top
X'	$(0, 1, 0, 0)$
Y	$(0, 0, 0, 0)$
Y'	$(0, 0, 0, 0)$
Z	\top
Z'	\top
W	\top

Step 15: $\varrho \leftarrow \text{Lift}_Y(\varrho, Y) = \varrho[Y := \max\{\text{Prog}(\varrho, Y, W), \text{Prog}(\varrho, Y, Y')\}] = \varrho[Y := \max\{\top, (0, 0, 0, 0)\}] = \varrho[Y := \top]$

v	$\varrho(v)$
X	\top
X'	$(0, 1, 0, 0)$
Y	\top
Y'	$(0, 0, 0, 0)$
Z	\top
Z'	\top
W	\top

Step 16: $\varrho \leftarrow \text{Lift}_{X'}(\varrho) = \varrho[X' := \min\{\text{Prog}(\varrho, X', Z), \text{Prog}(\varrho, X', Y)\}] = \varrho[X' := \min\{\top, \top\}] = \varrho[X' := \top]$

v	$\varrho(v)$
X	\top
X'	\top
Y	\top
Y'	$(0, 0, 0, 0)$
Z	\top
Z'	\top
W	\top

Step 17: $\varrho \leftarrow \text{Lift}_{Y'}(\varrho) = \varrho[Y' := \min\{\text{Prog}(\varrho, Y', X), \text{Prog}(\varrho, Y', Y)\}] = \varrho[Y' := \min\{\top, \top\}] = \varrho[Y' := \top]$

v	$\varrho(v)$
X	\top
X'	\top
Y	\top
Y'	\top
Z	\top
Z'	\top
W	\top

ϱ is least game parity progress measure, and $\{v \in V \mid \varrho(v) \neq \top\} = \emptyset$ is winning set for player \diamond . Hence **player \square wins from all vertices**

Let $G = (V, E, p, (V_\diamond, V_\square))$ be a parity game, and $\varrho: V \rightarrow \mathbb{M}_G^\top$ be **least game parity progress measure**.

- ▶ Define strategy $\bar{\varrho}: V_\diamond \rightarrow V$ for player \diamond , by setting $\bar{\varrho}(v)$ to be a **successor w of $v \in V_\diamond$ that minimises $\varrho(w)$** .
- ▶ $\bar{\varrho}$ is a **winning strategy for player \diamond from $\{v \in V \mid \varrho(v) \neq \top\}$** .

As the winning set for player \diamond is empty, the strategy for player \diamond can be chosen **arbitrarily**

Strategy for player \square cannot be inferred directly (winning set *can* be determined), some tricks have to be applied...

Let $G = (V, E, p, (V_\diamond, V_\square))$ be a parity game; $n = |V|$, $e = |E|$, $d = \max\{p(v) \mid v \in V\}$.

Worst-case running time complexity:

$$\mathcal{O}(de \cdot \left(\frac{n}{\lfloor d/2 \rfloor}\right)^{\lfloor d/2 \rfloor})$$

Lowerbound on worst-case:

$$\Omega(\lceil n/d \rceil^{\lceil d/2 \rceil})$$

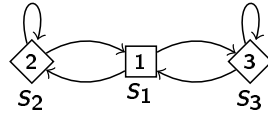
Let $G = (V, E, p, (V_\diamond, V_\square))$ be a parity game; $n = |V|$, $e = |E|$, $d = \max\{p(v) \mid v \in V\}$.

- ▶ Algorithm with best known upper bound: **Big step** algorithm due to Schewe, with complexity

$$\mathcal{O}(d \cdot n^{d/3})$$

- ▶ Also: **best known** upper bound for μ -calculus model checking
- ▶ Big step combines **recursive algorithm** with **small progress measures**;

Consider the following parity game:



- ▶ Compute the winning sets W_{\diamond} , W_{\square} for players \diamond and \square in this parity game using the small progress measures algorithm.
- ▶ Compare the solution with the solution obtained using the recursive algorithm and Gauß elimination.