# Algorithms for Model Checking (2IW55)
# Lecture 7:
# Solving parity games recursively

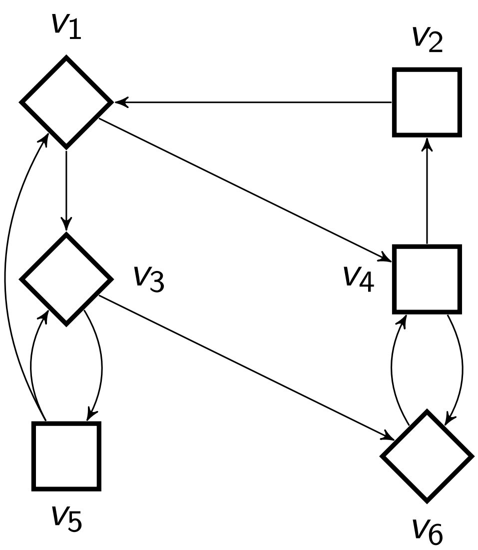
Maciej Gazda
September 30, 2013

using some of Jeroen Keiren's slides

/ Department of Mathematics and Computer Science

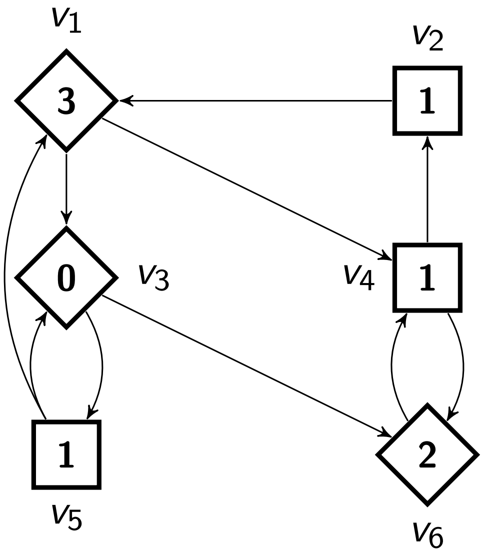


## Parity games




- 2 players: $\Diamond$ (Even) and $\Box$ (Odd)
- every node has an owner ($V = V_\Diamond \cup V_\Box$)
- moving token infinitely often; owner chooses the next state
- play = infinite path through the game

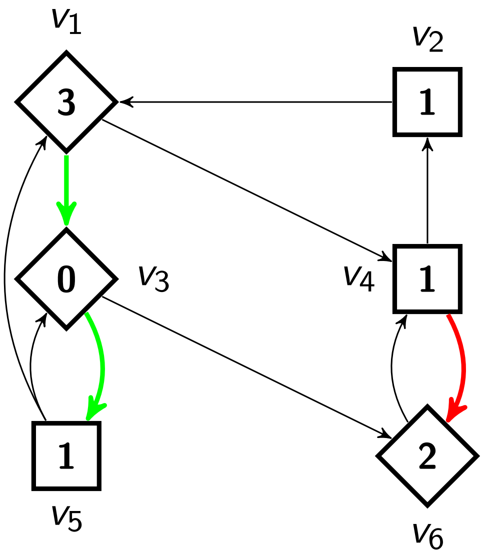
# Parity games




- 2 players: $\diamond$ (Even) and $\square$ (Odd)
- every node has an owner ($V = V_\diamond \cup V_\square$)
- moving token infinitely often; owner chooses the next state
- play = infinite path through the game
- nodes labelled with natural numbers (priorities)
- winner of the play: depends on the minimal priority occurring infinitely often (even or odd?)



# Parity games




- strategy
- winning strategy
- memoryless strategy
- winning partition

# Goal

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

- There is a unique partition $(W_\Diamond, W_\Box)$ of $V$ such that:
  - $\Diamond$ has winning strategy $\varrho_\Diamond$ from $W_\Diamond$, and
  - $\Box$ has winning strategy $\varrho_\Box$ from $W_\Box$.

**Goal of parity game algorithms**

Compute partitioning $(W_\Diamond, W_\Box)$ with strategies $\varrho_\Diamond$ and $\varrho_\Box$ of $V$, such that $\varrho_\Diamond$ is winning for player $\Diamond$ from $W_\Diamond$ and $\varrho_\Box$ is winning for player $\Box$ from $W_\Box$.



# Notation

Let $G = (V, E, p, (V_\Diamond, V_\Box))$ be a parity game.
We use the following notation:

- $\bigcirc \in \{\Diamond, \Box\}$
- $\overline{\Diamond}$ is $\Box$, $\overline{\Box}$ is $\Diamond$
- $G \setminus U$ is parity game $G$ restricted to the vertices outside $U$. Formally $G \setminus U = (V', E', p', (V'_\Diamond, V'_\Box))$, with
  - $V' = V \setminus U$,
  - $E' = E \cap (V' \times V')$,
  - $p'(v) = p(v)$ for $v \in V \setminus U$,
  - $V'_\Diamond = V_\Diamond \setminus U$, and
  - $V'_\Box = V_\Box \setminus U$
- $G \cap U$ defined similarly

## Closed strategies and sets

Let $G = (V, E, p, (V_\Diamond, V_\Box))$ be a parity game. A strategy $\varrho_\Diamond : V_\Diamond \to V$ 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 $v \in V_\Box$ then $(v, w) \in E$ implies $w \in W$.

Each play consistent with strategy $\varrho_\Diamond$ closed on $W$, starting in $W$, stays within $W$

A set $W$ is $\Diamond$-closed [resp. $\Box$-closed], if there is a strategy of player $\Diamond$ [resp. $\Box$] that is closed on $W$.



## Dominions

Let $W_\bigcirc$ be a winning region.

**Definition**

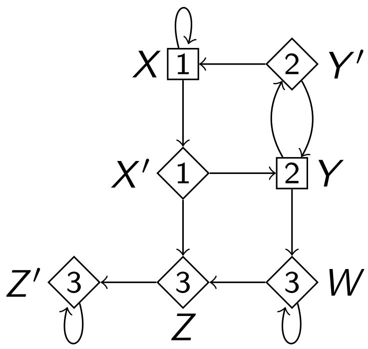
$D \subseteq W_\bigcirc$ is a dominion of $\bigcirc$, if $\bigcirc$ has memoryless strategy $\varrho_\bigcirc$ that is:

- winning for $\bigcirc$ from all $v \in D$
- closed on $D$

## Example of dominions

**Example**

Consider parity game $G$:




- $\{X\}$, $\{Z', Z, W\}$ are $\Box$-dominions
- Note that $\{Z, W\}$ is not a dominion
- Why is $\{Y, Y'\}$ not a dominion

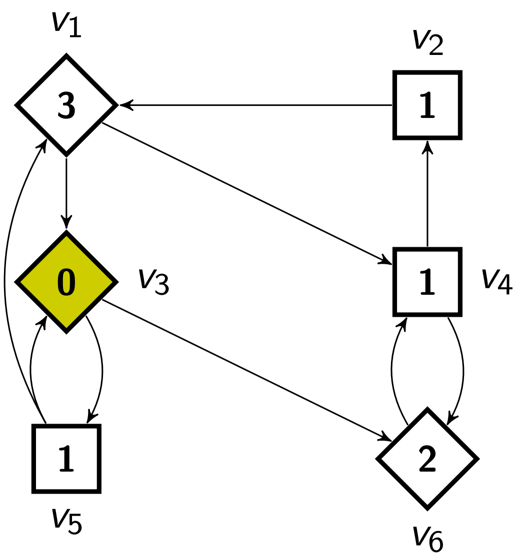
# Attractor sets

The attractor set for $\bigcirc$ and set $U \subseteq V$ is the set of vertices such that $\bigcirc$ can force any play to reach $U$.

**Definition**

$$
\begin{aligned}
Attr^0_{\bigcirc}(G,U) &= U \\
Attr^{k+1}_{\bigcirc}(G,U) &= Attr^k_{\bigcirc}(G,U) \\
&\quad \cup \{v \in V_{\bigcirc} \mid \exists v' \in V : (v,v') \in E \\
&\qquad \wedge v' \in Attr^k_{\bigcirc}(G,U)\} \\
&\quad \cup \{v \in V_{\overline{\bigcirc}} \mid \forall v' \in V : (v,v') \in E \\
&\qquad \implies v' \in Attr^k_{\bigcirc}(G,U)\}) \\
Attr_{\bigcirc}(G,U) &= \bigcup_{k \in \mathbb{N}} Attr^k_{\bigcirc}(G,U)
\end{aligned}
$$

## Attractor: "forced" reachability



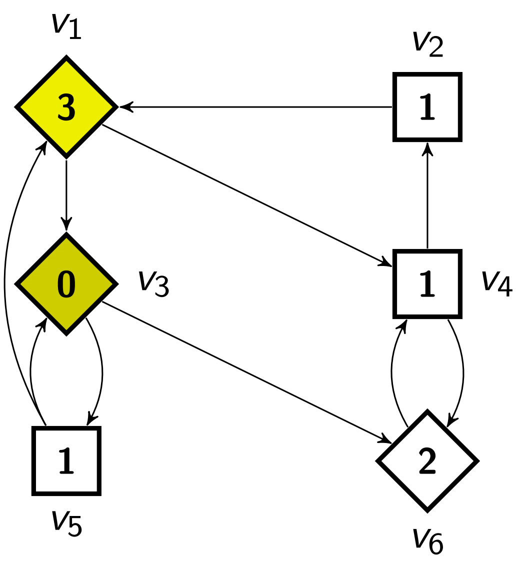

$Attr_P(B)$: vertices from which player $P$ can force the play to reach set $B$

Consider $Attr_\diamond(G, \{v_3\})$

$$Attr^0_\diamond(G, \{v_3\}) = \{v_3\}$$



## Attractor: "forced" reachability

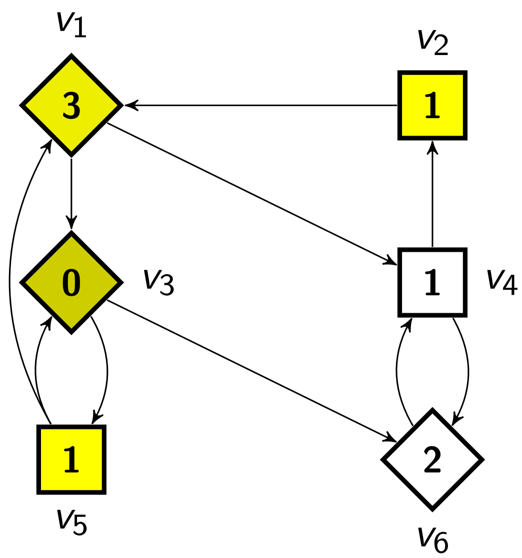



$Attr_P(B)$: vertices from which player $P$ can force the play to reach set $B$

Consider $Attr_\diamond(G, \{v_3\})$

$$Attr^0_\diamond(G, \{v_3\}) = \{v_3\}$$
$$Attr^1_\diamond(G, \{v_3\}) = \{v_3, v_1\}$$

# Attractor: "forced" reachability




$Attr_P(B)$: vertices from which player $P$ can force the play to reach set $B$

Consider $Attr_\diamond(G, \{v_3\})$

$$
\begin{array}{rcl}
Attr^0_\diamond(G, \{v_3\}) & = & \{v_3\} \\
Attr^1_\diamond(G, \{v_3\}) & = & \{v_3, v_1\} \\
Attr^2_\diamond(G, \{v_3\}) & = & \{v_3, v_1, v_2, v_5\}
\end{array}
$$

Time: $O(|V| + |E|)$



# Properties

Let $D$ be a $\bigcirc$-dominion in $G$, then:

- there is a strategy $\varrho_\bigcirc$ such that $\bigcirc$ wins on $D$;
- $\bigcirc$ can always choose to stay in $D$;
- $\overline{\bigcirc}$ cannot leave $D$;
- $A = Attr^\bigcirc(G, D)$ is a $\bigcirc$-dominion;
- $\bigcirc$ cannot leave $V \setminus A$

# Recursive algorithm (intuition)

Divide and conquer

- Base: empty game
- Step:
  - identify a proper subgame (with at least one node less)
  - compute a dominion in the subgame
  - remove the dominion and solve the remainder of the original game
  - assemble winning sets/strategies from winning sets/strategies of subgames



# Recursive algorithm (McNaughton '93, Zielonka '98)

Recursively solve a parity game: *Recursive*$(G)$. Returns partitioning $(W_\Diamond, W_\Box)$ such that $\Diamond$ wins from $W_\Diamond$, and $\Box$ wins from $W_\Box$.

$m \leftarrow \min\{p(v) \mid v \in V\}^\dagger$
$h \leftarrow \max\{p(v) \mid v \in V\}^\dagger$
**if** $h - m = 0$ **then**
  **if** $h$ is even **then**
    **return** $(V, \emptyset)$
  **else**
    **return** $(\emptyset, V)$
  **end if**
**end if**
$\bigcirc \leftarrow \begin{cases} \Diamond & \text{if m is even} \\ \Box & \text{otherwise} \end{cases}$
$U \leftarrow \{v \in V \mid p(v) = m\}$
$A \leftarrow Attr^{\bigcirc}(G, U)$
$(W'_\Diamond, W'_\Box) \leftarrow Recursive(G \setminus A)$
**if** $W'_{\overline{\bigcirc}} = \emptyset$ **then**
  $W_\bigcirc \leftarrow A \cup W'_\bigcirc$
  $W_{\overline{\bigcirc}} \leftarrow \emptyset$
**else**
  $B \leftarrow Attr^{\overline{\bigcirc}}(G, W'_{\overline{\bigcirc}})$
  $(W_\Diamond, W_\Box) \leftarrow Recursive(G \setminus B)$
  $W_{\overline{\bigcirc}} \leftarrow W_{\overline{\bigcirc}} \cup B$
**end if**
**return** $(W_\Diamond, W_\Box)$

$\dagger$: we assume that min and max return $-1$ if called on an empty set

# Zielonka's Recursive Algorithm

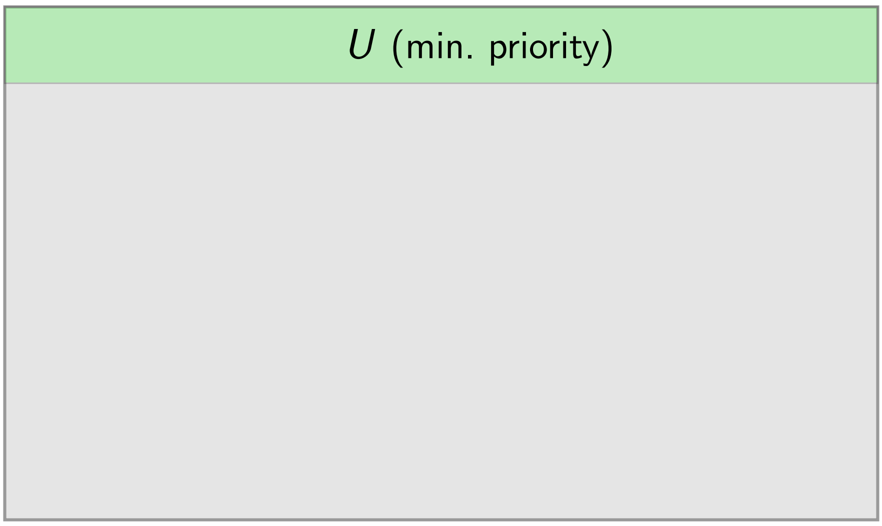
Assume that the minimal priority in $G$ is even.

$G$



# Zielonka's Recursive Algorithm

Assume that the minimal priority in $G$ is even.

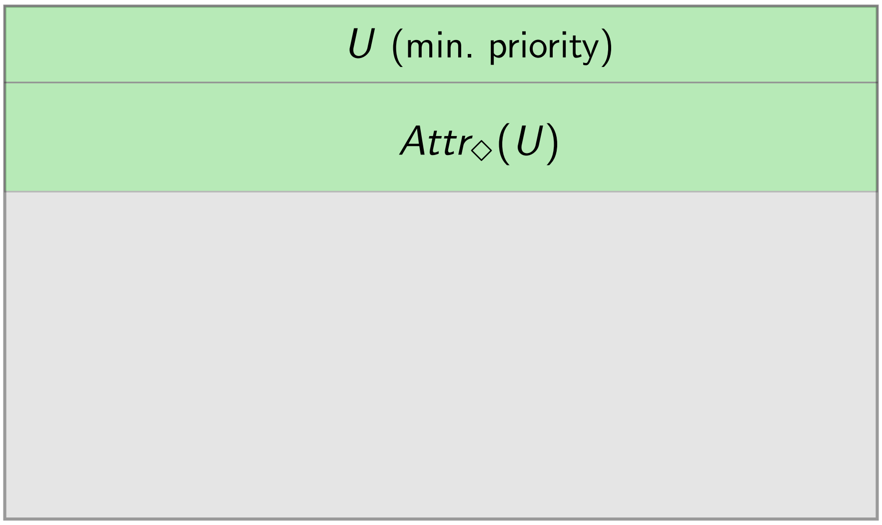



$G$

line 11

# Zielonka's Recursive Algorithm

Assume that the minimal priority in $G$ is even.



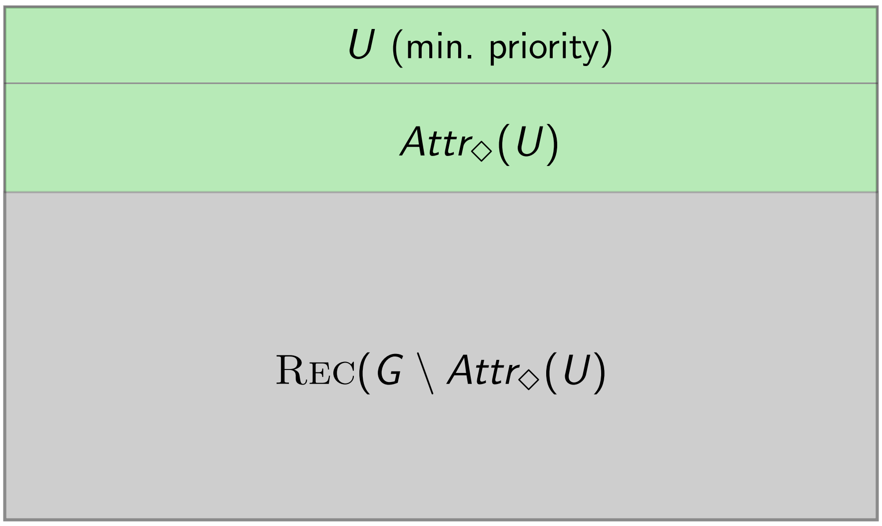

$G$

line 12



# Zielonka's Recursive Algorithm

Assume that the minimal priority in $G$ is even.

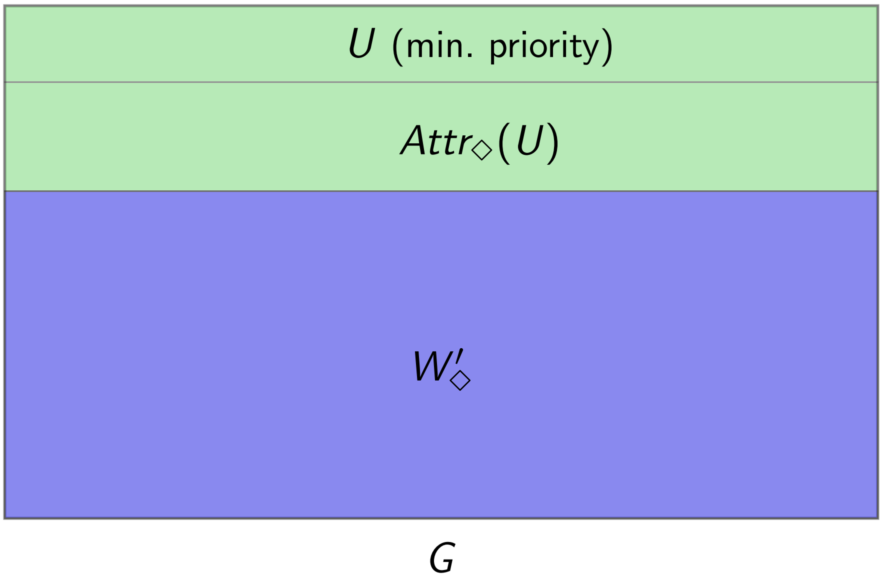



$G$

line 13

# Zielonka's Recursive Algorithm

Assume that the minimal priority in $G$ is even.

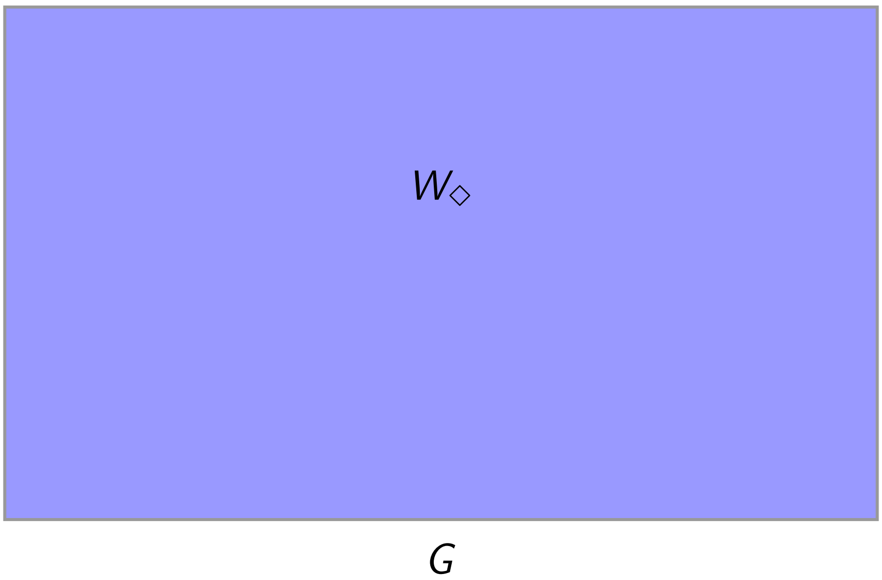



line 14 (case $W'_{\Box} = \emptyset$)



# Zielonka's Recursive Algorithm

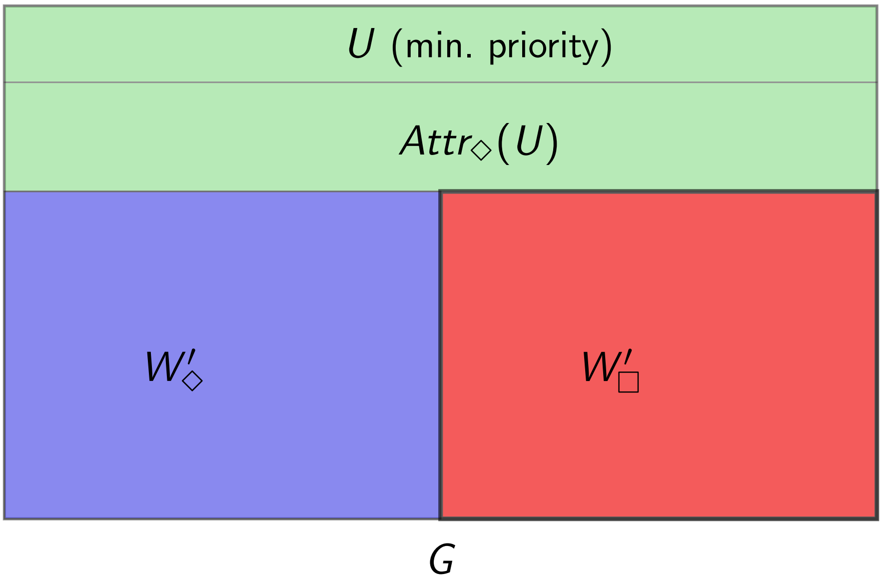
Assume that the minimal priority in $G$ is even.




line 15, 16 & 22 (case $W'_{\Box} = \emptyset$)

## Zielonka's Recursive Algorithm

Assume that the minimal priority in $G$ is even.

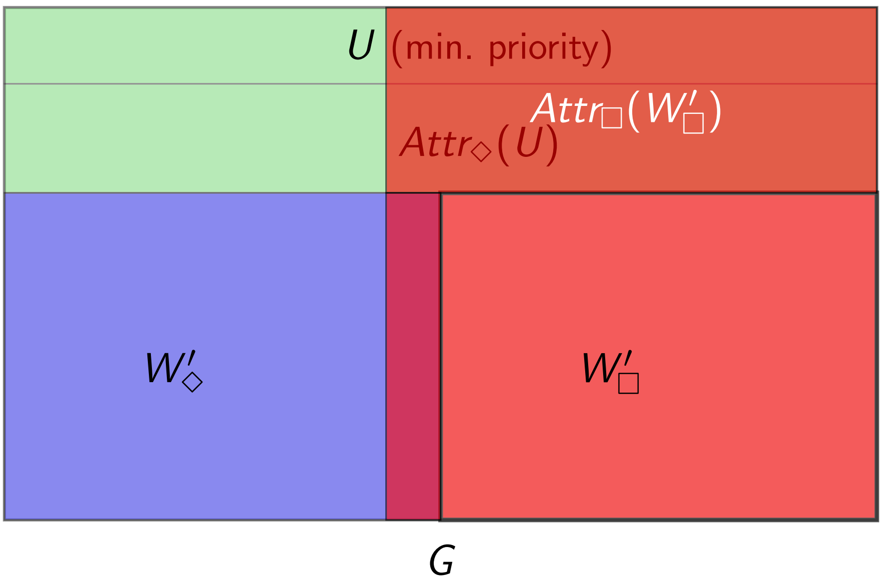



line 17 (case $W'_{\Box} \neq \emptyset$)



## Zielonka's Recursive Algorithm

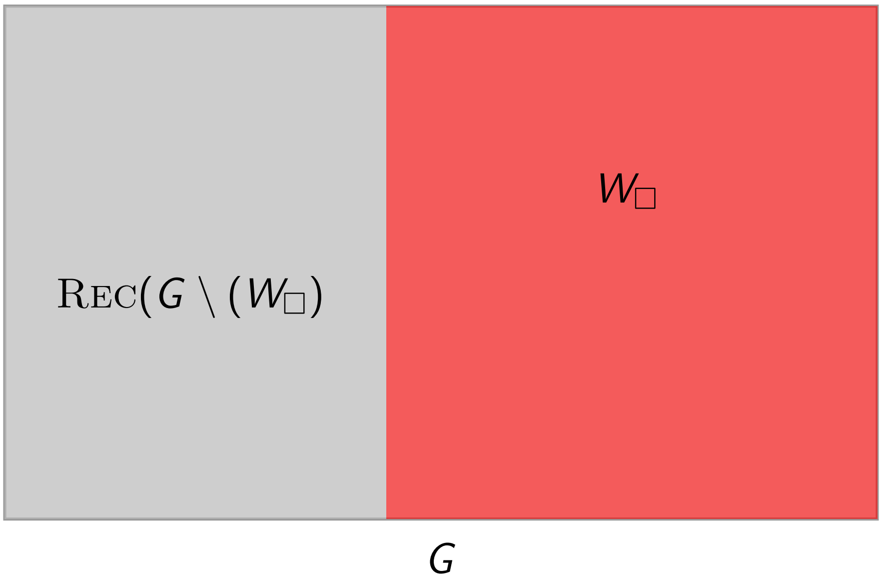
Assume that the minimal priority in $G$ is even.




line 18

# Zielonka's Recursive Algorithm

Assume that the minimal priority in $G$ is even.

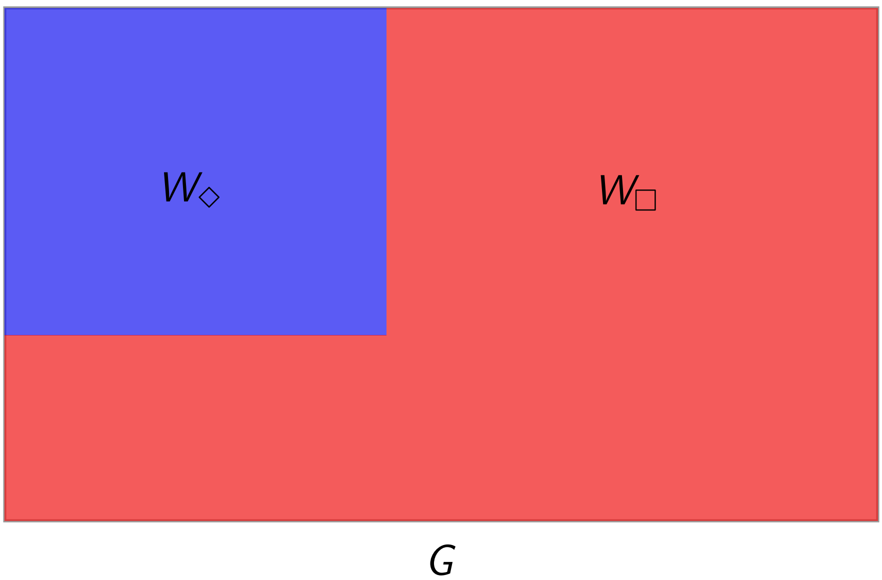



line 19

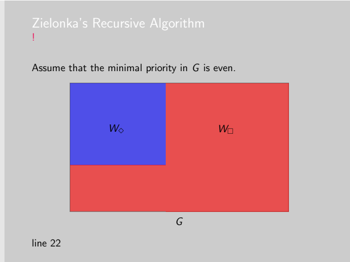


# Zielonka's Recursive Algorithm

Assume that the minimal priority in $G$ is even.



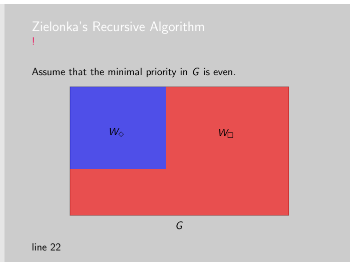

line 22




## Observations

- Lines 1-9: base case, straightforward.
- Lines 10-13: try to establish a dominion. Two cases:
  - Lines 12-15: ($\bigcirc$ wins all): $\bigcirc$ wins in $G \setminus A$, then $\bigcirc$ wins all of $G$, since if $\overline{\bigcirc}$ visits $A$, then $\bigcirc$ plays towards $U$ using attractor, visiting $A$ infinitely often, hence $m$ infinitely often. If $A$ not visited, game stays in $G \setminus A$.
  - Lines 16-20: ($\overline{\bigcirc}$-dominion found): $W'_{\overline{\bigcirc}}$ is a $\overline{\bigcirc}$-dominion in $G \setminus A$. Since $\bigcirc$ cannot leave $G \setminus A$ also $W'_{\overline{\bigcirc}}$ is $\overline{\bigcirc}$-dominion in $G$. Then solve remaining game recursively and fix solution, compose strategies.

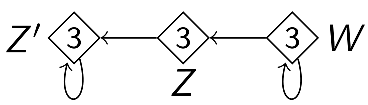


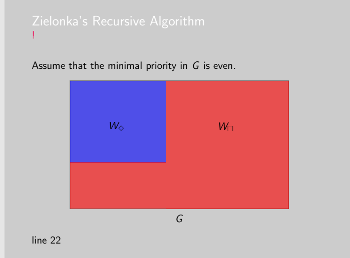



## Exercise

Apply the recursive algorithm to the following parity game $G$

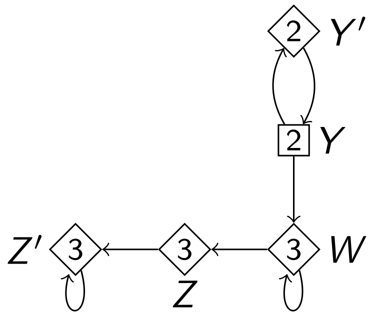
$m \leftarrow 3$
$h \leftarrow 3$
**return** $(\emptyset, \{W, Z, Z'\})$





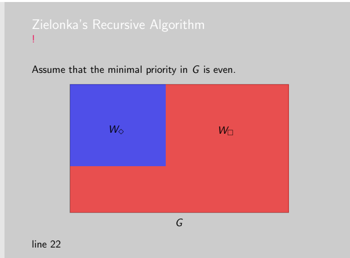

## Exercise

Apply the recursive algorithm to the following parity game $G$



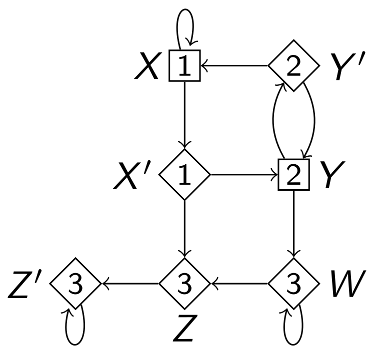

1: $m \leftarrow 2$
2: $h \leftarrow 3$
3: $\ldots$
10: $\bigcirc \leftarrow \diamond$
11: $U \leftarrow \{v \in V \mid p(v) = 2\} = \{Y, Y'\}$
12: $A \leftarrow Attr^{\diamond}(G, U) = \{Y, Y'\}$
13: $(W'_\diamond, W'_\Box) \leftarrow Recursive(G \setminus \{Y, Y'\}) = (\emptyset, \{Z, Z', W\})$
14: **if** $W'_\Box = \emptyset$ **then**
15: $\ldots$
15: **else**
16: $B \leftarrow Attr^{\Box}(G, W'_\Box) = \{Y, Y', Z, Z', W\}$
17: $(W_\diamond, W_\Box) \leftarrow Recursive(G \setminus B) = (\emptyset, \emptyset)$
18: $W_\Box \leftarrow W_\Box \cup B = B = \{Y, Y', Z, Z', W\}$
19: **end if**
20: **return** $(W_\diamond, W_\Box) = (\emptyset, \{Y, Y', Z, Z', W\})$

2013-10-21
Algorithms for Model Checking (2IW55) Lecture 7:Solving parity games recursively
└Solving parity games
└Zielonka's Recursive Algorithm




## Example ($Recursive(G)$)

Consider parity game $G$:




1: $m \leftarrow 1$
2: $h \leftarrow 3$
3: $\ldots$
10: $\bigcirc \leftarrow \Box$
11: $U \leftarrow \{v \in V \mid p(v) = 1\} = \{X, X'\}$
12: $A \leftarrow Attr^{\Box}(G, U) = \{X, X'\}$
13: $(W'_\diamond, W'_\Box) \leftarrow Recursive(G \setminus \{X, X'\}) = (\emptyset, \{Y, Y', Z, Z', W\})$
14: **if** $W'_\diamond = \emptyset$ **then**
15: $W_\Box \leftarrow A \cup W'_\Box = \{X, X', Y, Y', Z, Z', W\}$
16: $W_\diamond \leftarrow \emptyset$
17: **else**
18: $\ldots$
19: **end if**
20: **return** $(W_\diamond, W_\Box) = (\emptyset, \{X, X', Y, Y', Z, Z', W\})$

So, player $\Box$ wins from all vertices!

# Complexity

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

Worst-case running time complexity

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

Lowerbound on worst-case:

$$\Omega(2^{n/3})$$





Complexity

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

Worst-case running time complexity

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

Lowerbound on worst-case:

$\Omega(2^{n/3})$

## Worst-case analysis

Recurrence $T(n,d)$ defined as $\begin{cases} T(0,0) & = 1 \\ T(n+1, d+1) & = T(n,d) + T(n, d+1) + e \end{cases}$

Solve recurrence using substitution method. Guess solution $T(n,d) = \mathcal{O}(en^d)$. We prove that $T(n,d) \leqslant cen^d$ for an appropriate constant $c > 0$ by induction on $n$ and $d$.

- $n = d = 0$. $T(0,0) \leqslant ce$, $c \geqslant 1$
- $n > 0, d > 0$. IH: assume $T(n, d+1) \leqslant cen^{d+1}$ and $T(n,d) \leqslant cen^d$. Substitute into recurrence:

$$\begin{aligned} T(n+1, p+1) &= T(n,d) + T(n, d+1) + e \\ &\leqslant cen^d + cen^{d+1} + e \\ &\leqslant^\dagger ce(n^d + n^{d+1} + 1) \\ &= ce(n^d + n \cdot n^d + 1) \\ &= ce((n+1)n^d + 1) \\ &\leqslant ce((n+1)(n+1)^d) \\ &= ce((n+1)^{d+1}) \end{aligned}$$

$\dagger : c \geqslant 1$.

## The Quest for an Efficient PG Solving Algorithm

- **Recursive algorithm** [McNaughton 1993, Zielonka 1998] $O(n^c)$
- Small Progress Measures [Jurdziński, 2000] $O(n^{c/2})$
- subexponential algorithm [Jurdziński, Paterson and Zwick, 2006] $O(n^{\sqrt{n}})$
- bigstep [Schewe, 2007] $O(n^{c/3})$
- strategy improvement algorithms [e.g. Voege & Jurdziński]: superpolynomial in worst case

Computational status: $NP \cap \text{co}NP$.



## Wrap up

- Recursive algorithm:
  - Divide and conquer
  - Dominions
  - Attractor sets
  - $\mathcal{O}(en^d)$
  - Exponential examples available