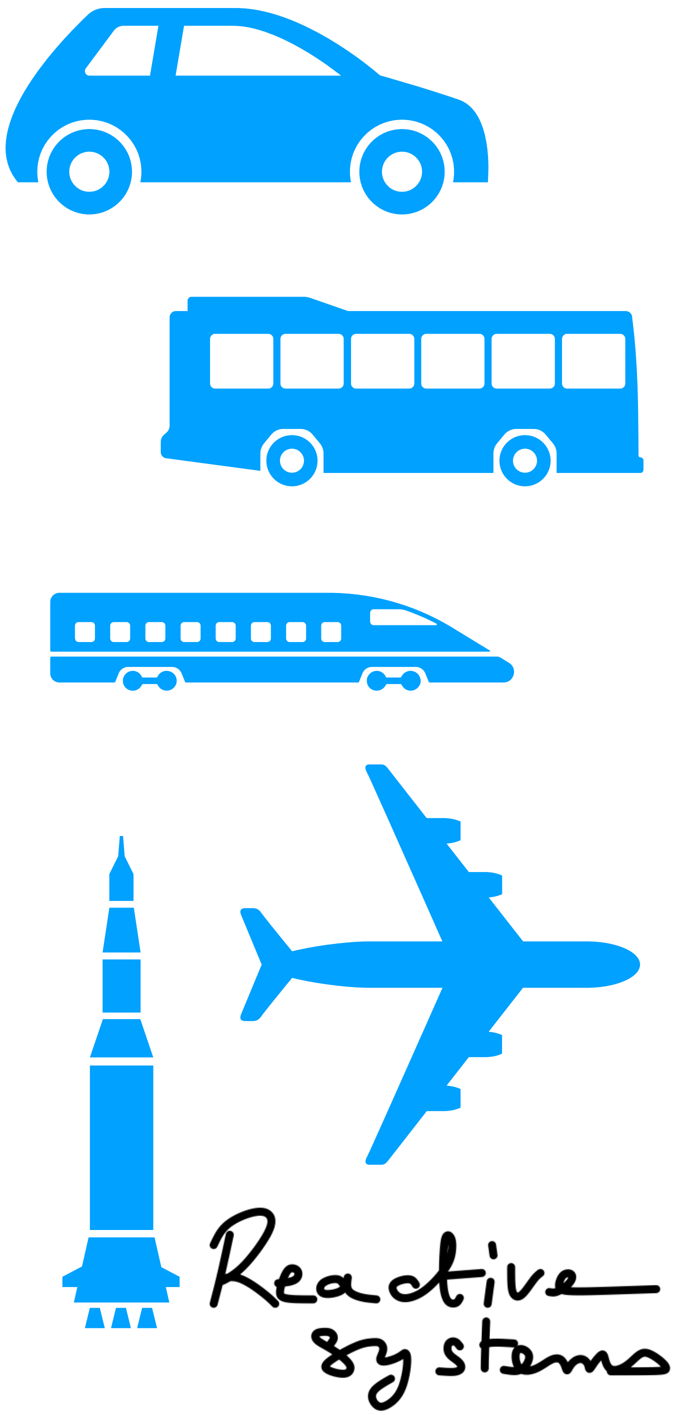
Quantitative Games on Graphs

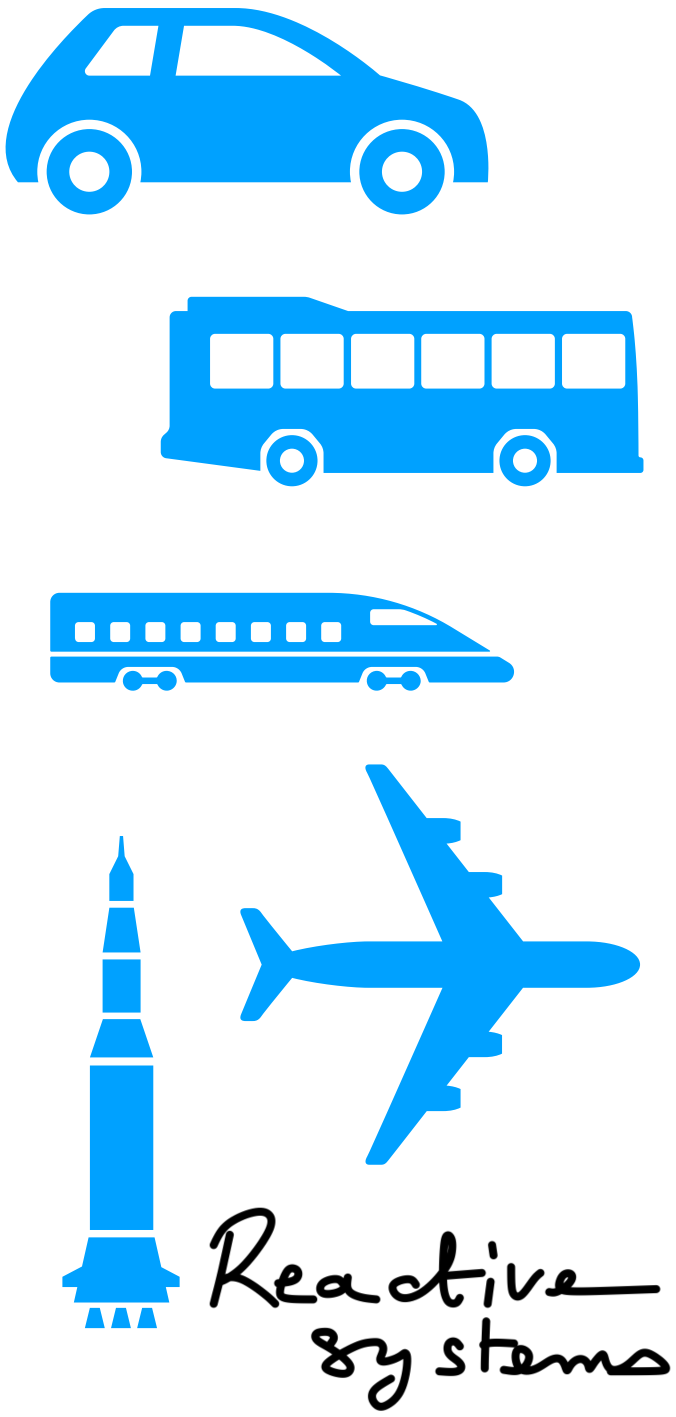
Benjamin Monmege, Aix-Marseille Université

Séminaire ENS Rennes

Games for synthesis

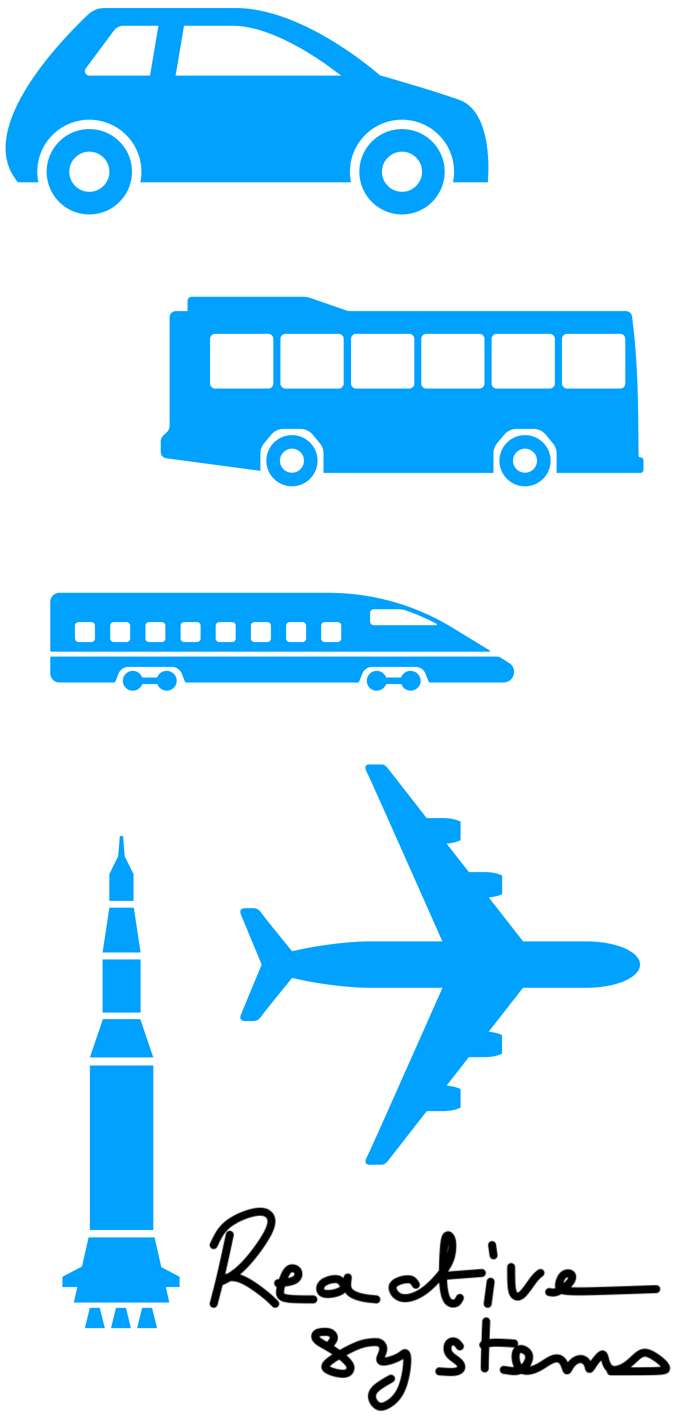


Games for synthesis

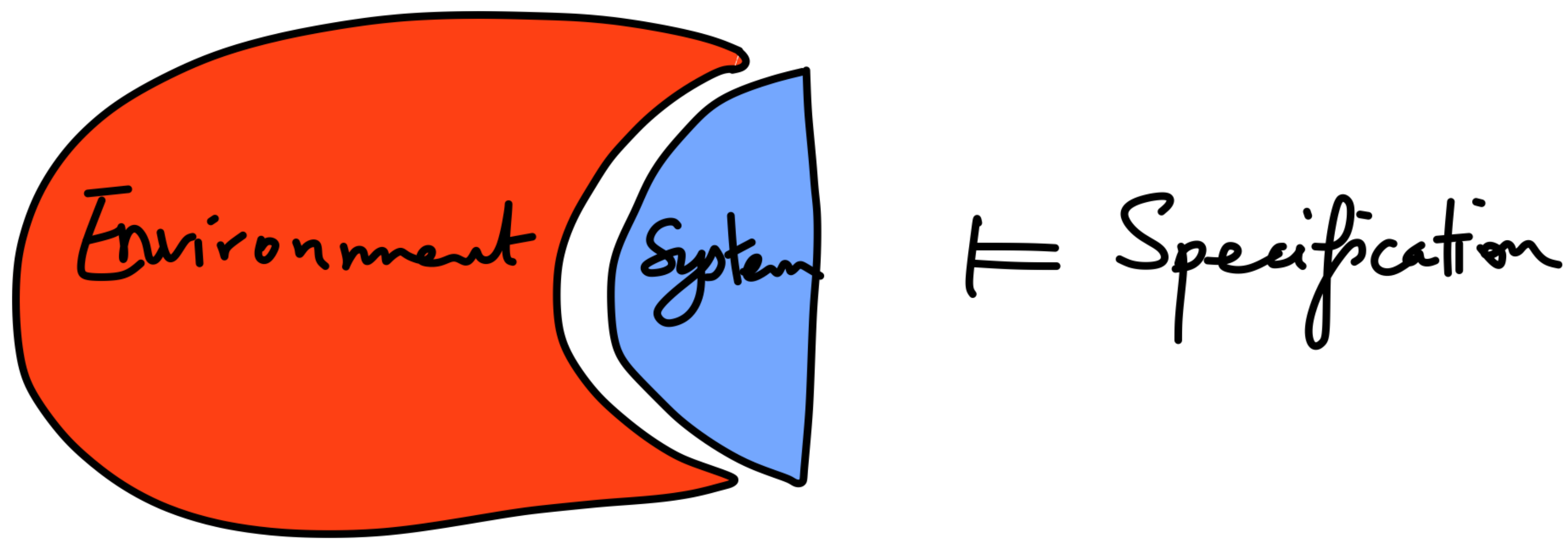


Crucial to make the critical programs **correct**

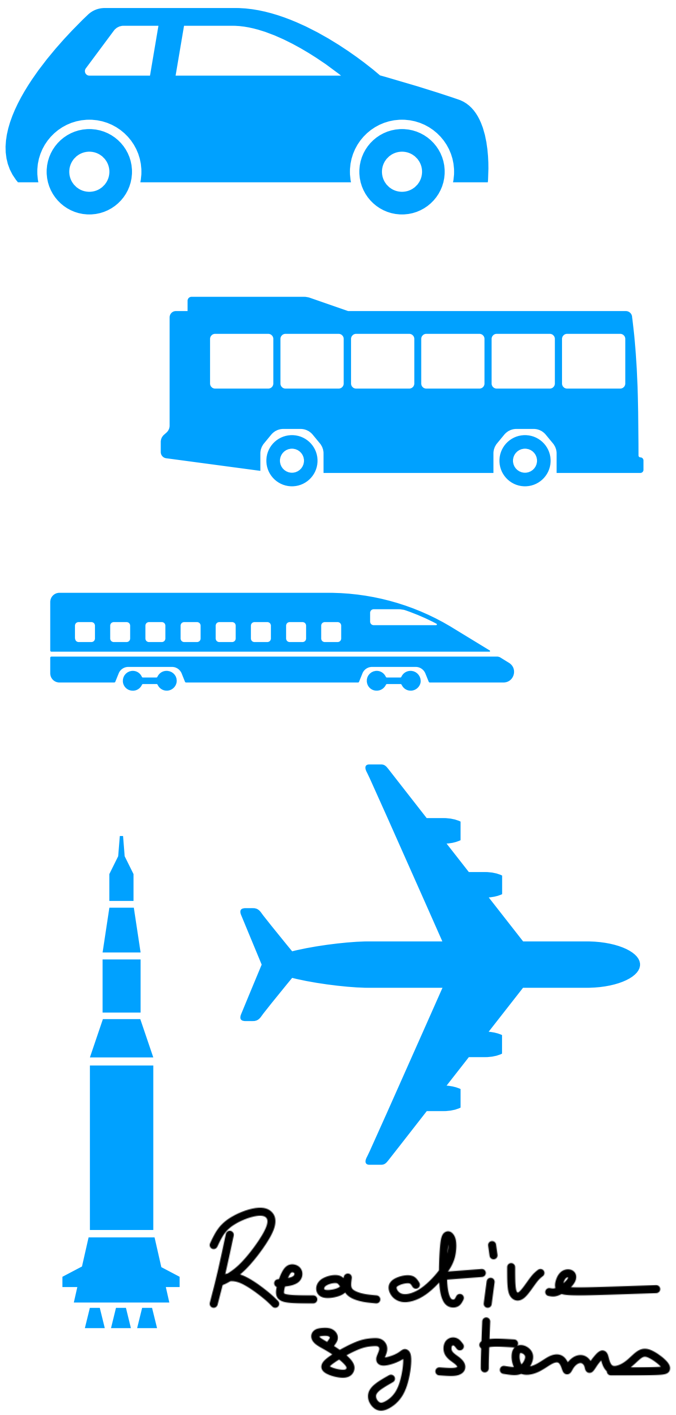
Games for synthesis



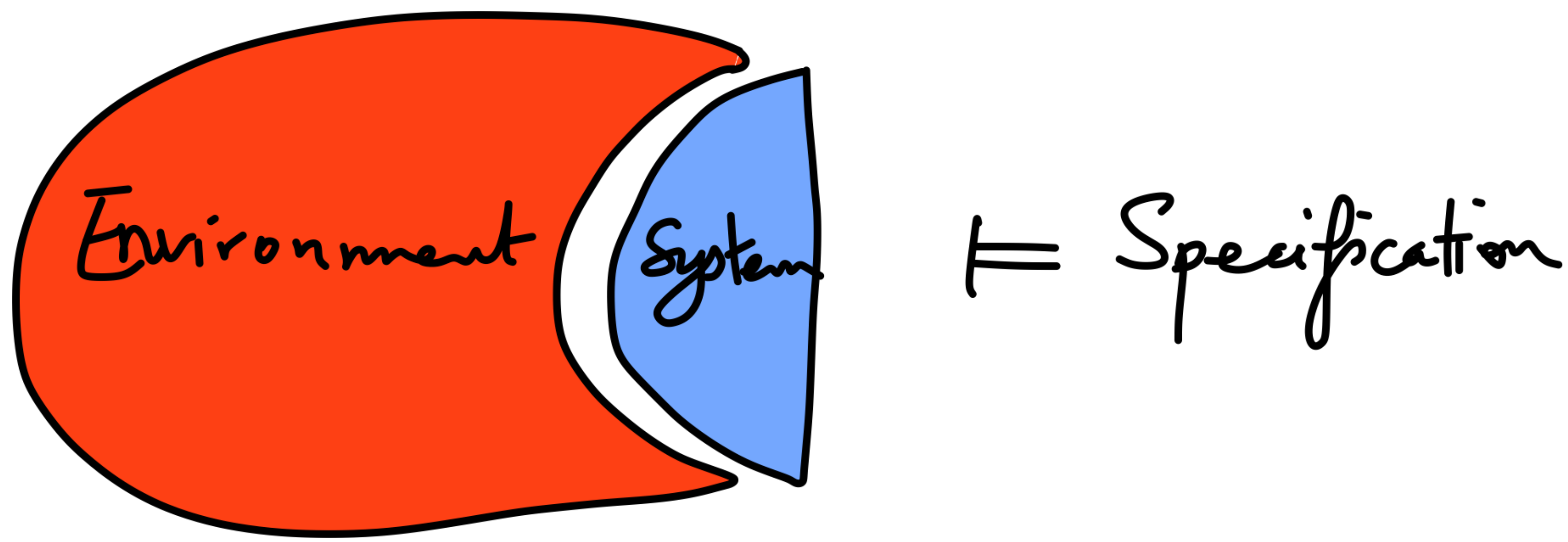
Crucial to make the critical programs **correct**



Games for synthesis

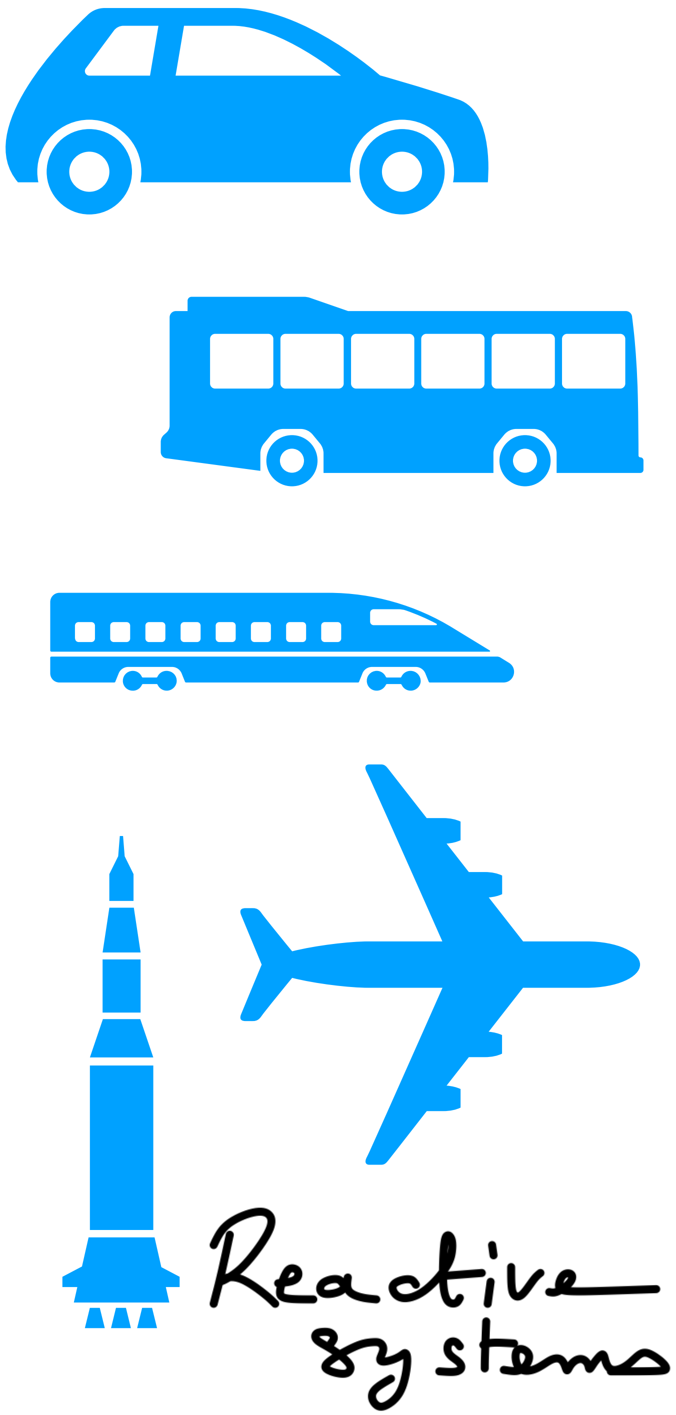


Crucial to make the critical programs **correct**

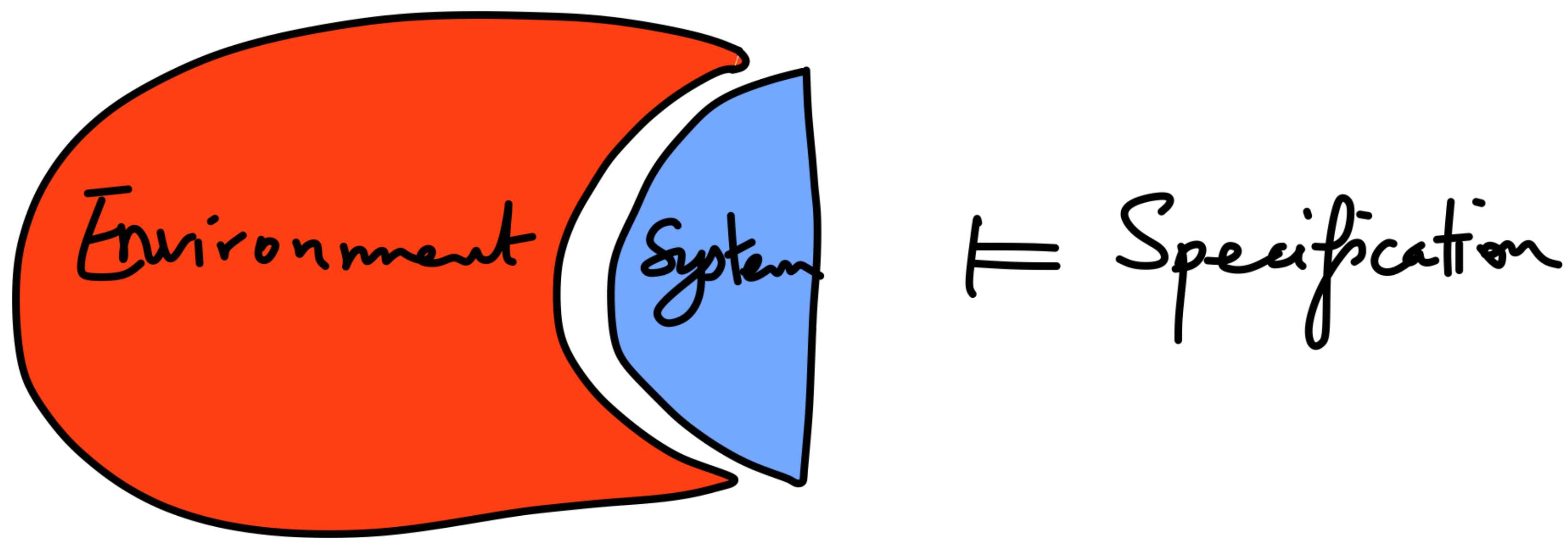


Instead of verifying an existing system...

Games for synthesis



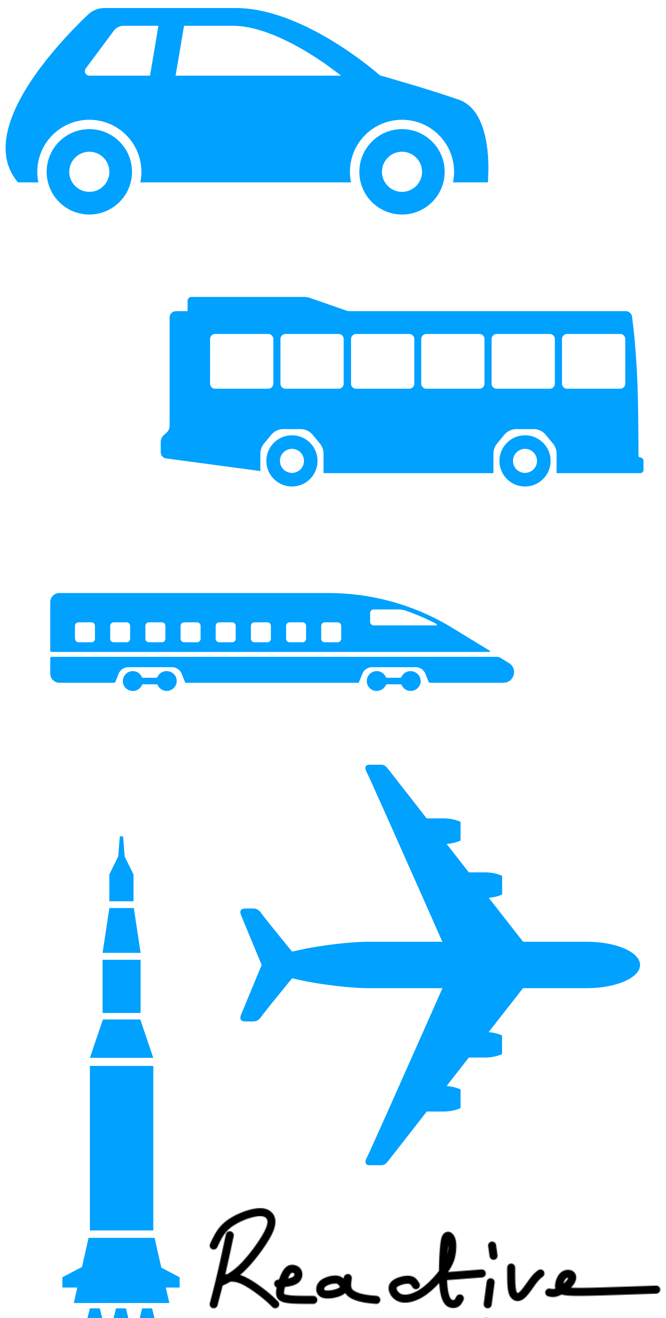
Crucial to make the critical programs **correct**



Instead of verifying an existing system...

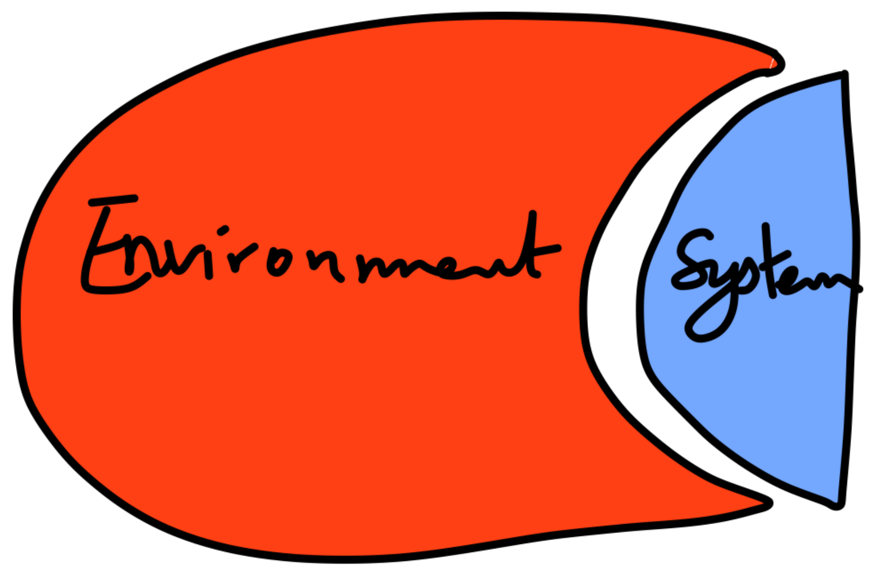
Synthesise a correct-by-design one!

Games for synthesis



Reactive systems

Crucial to make the critical programs **correct**



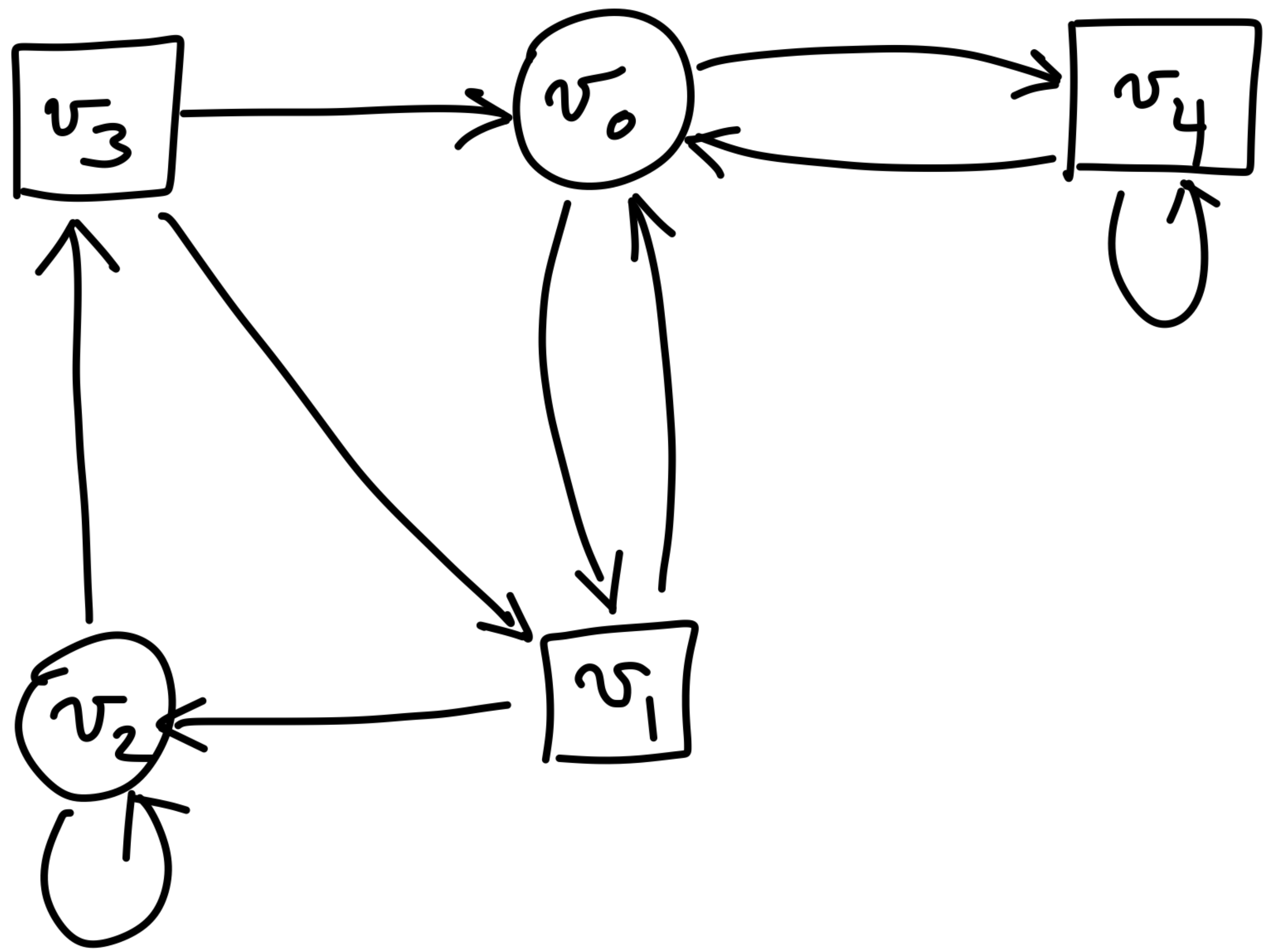
$\models$ Specification

Instead of verifying an existing system...

Synthesise a correct-by-design one!

Winning strategy = Correct system

2-player zero-sum games on graphs

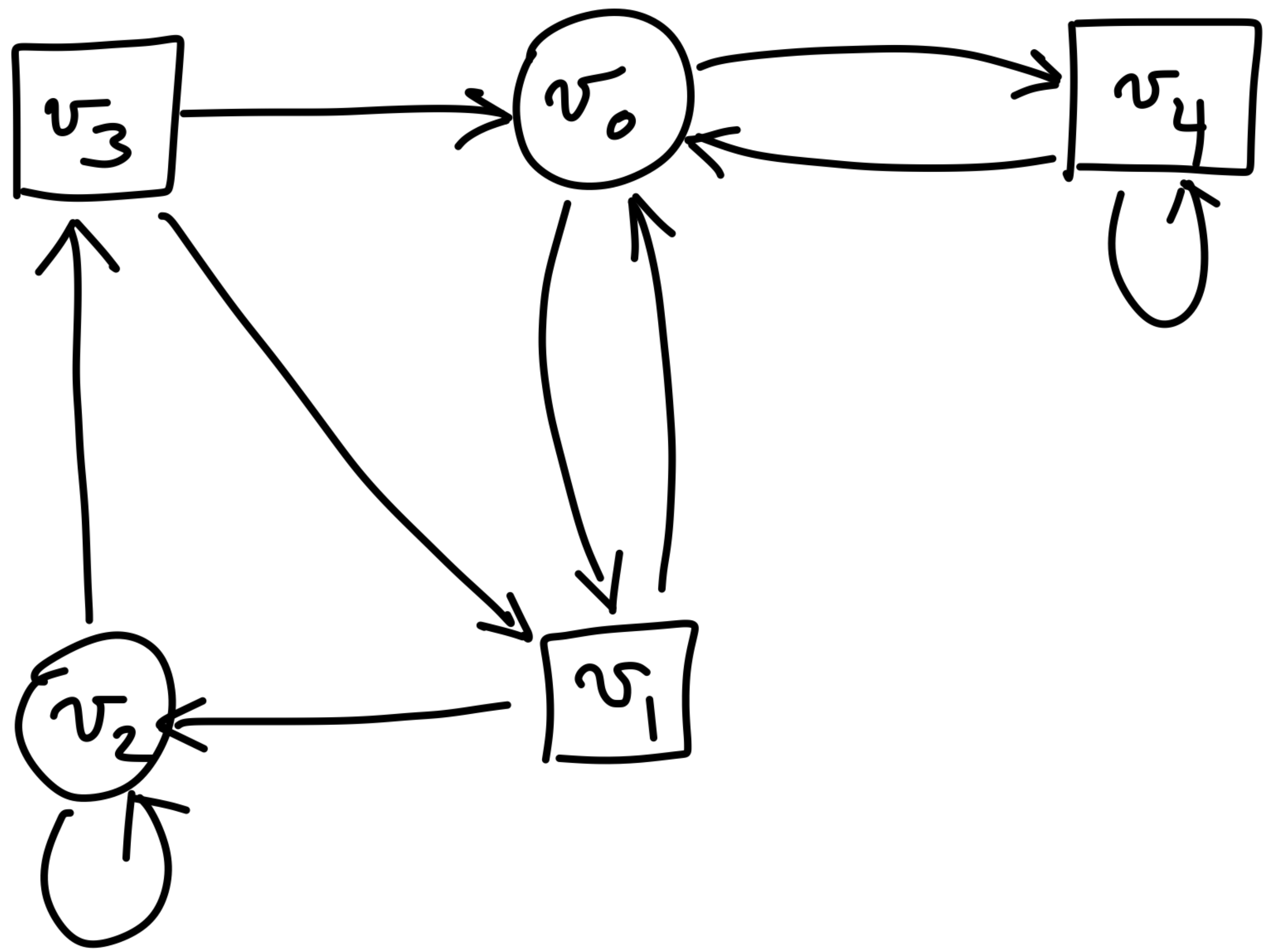


Finite directed graphs

Vertices of Player ◯

Vertices of Player □

2-player zero-sum games on graphs



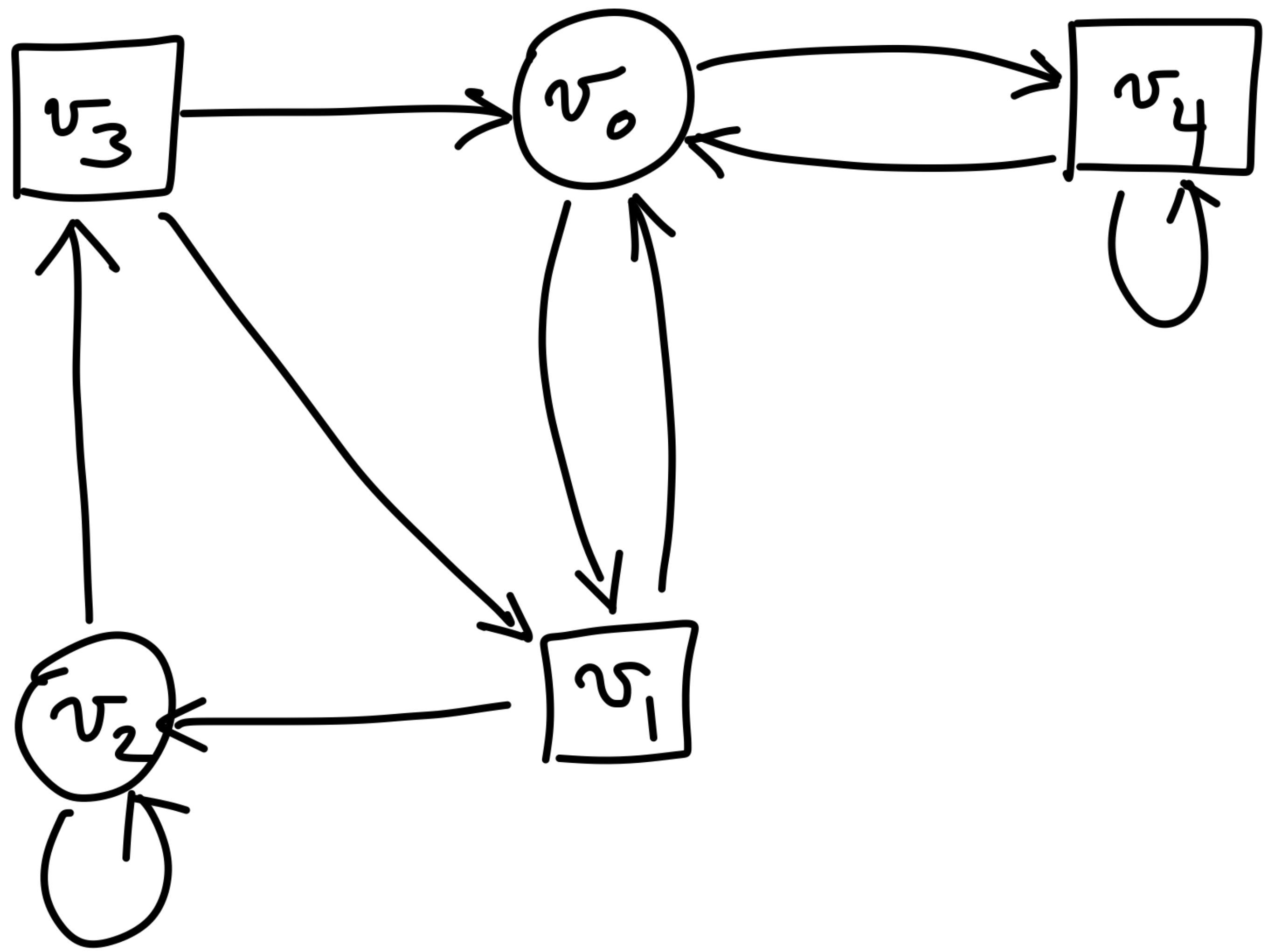
Finite directed graphs

Vertices of Player ◯

Vertices of Player □

Play: move a token along vertices

2-player zero-sum games on graphs



Finite directed graphs

Vertices of Player ◯

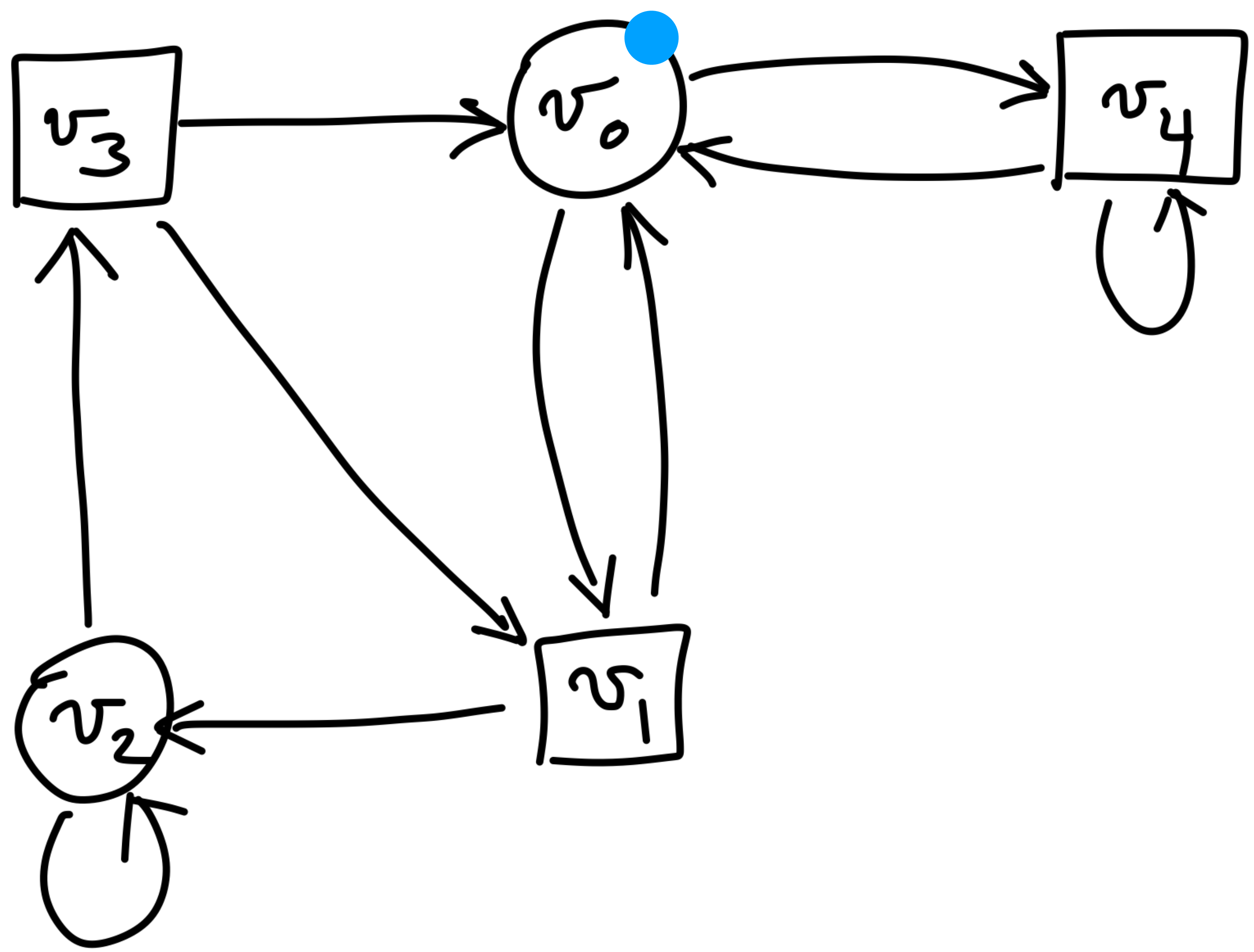
Vertices of Player □

Play: move a token along vertices

Infinite number of rounds

Outcome: infinite path

2-player zero-sum games on graphs



Finite directed graphs

Vertices of Player ◯

Vertices of Player □

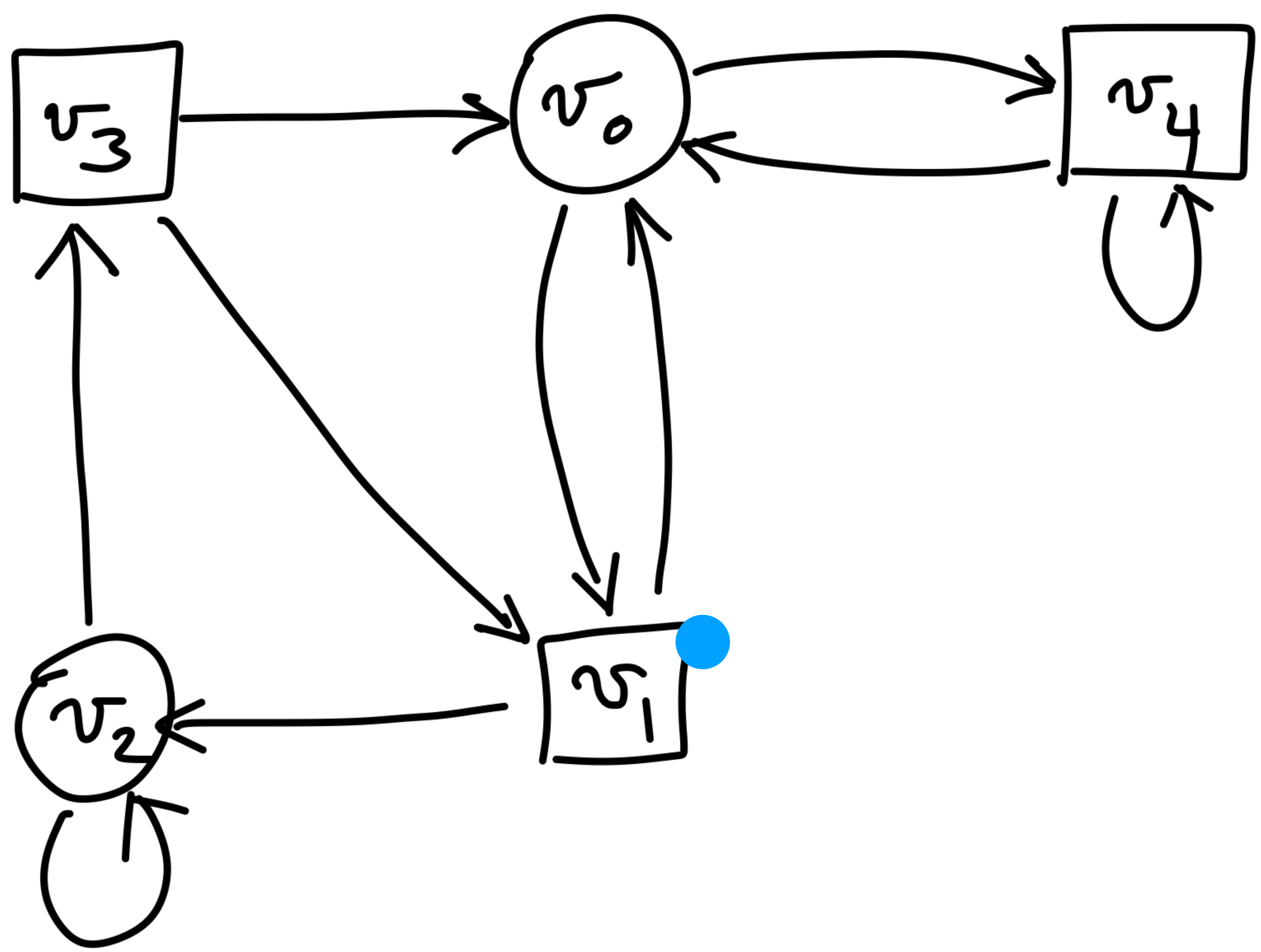
Play: move a token along vertices

Infinite number of rounds

Outcome: infinite path

v_0

2-player zero-sum games on graphs



Finite directed graphs
Vertices of Player ◯

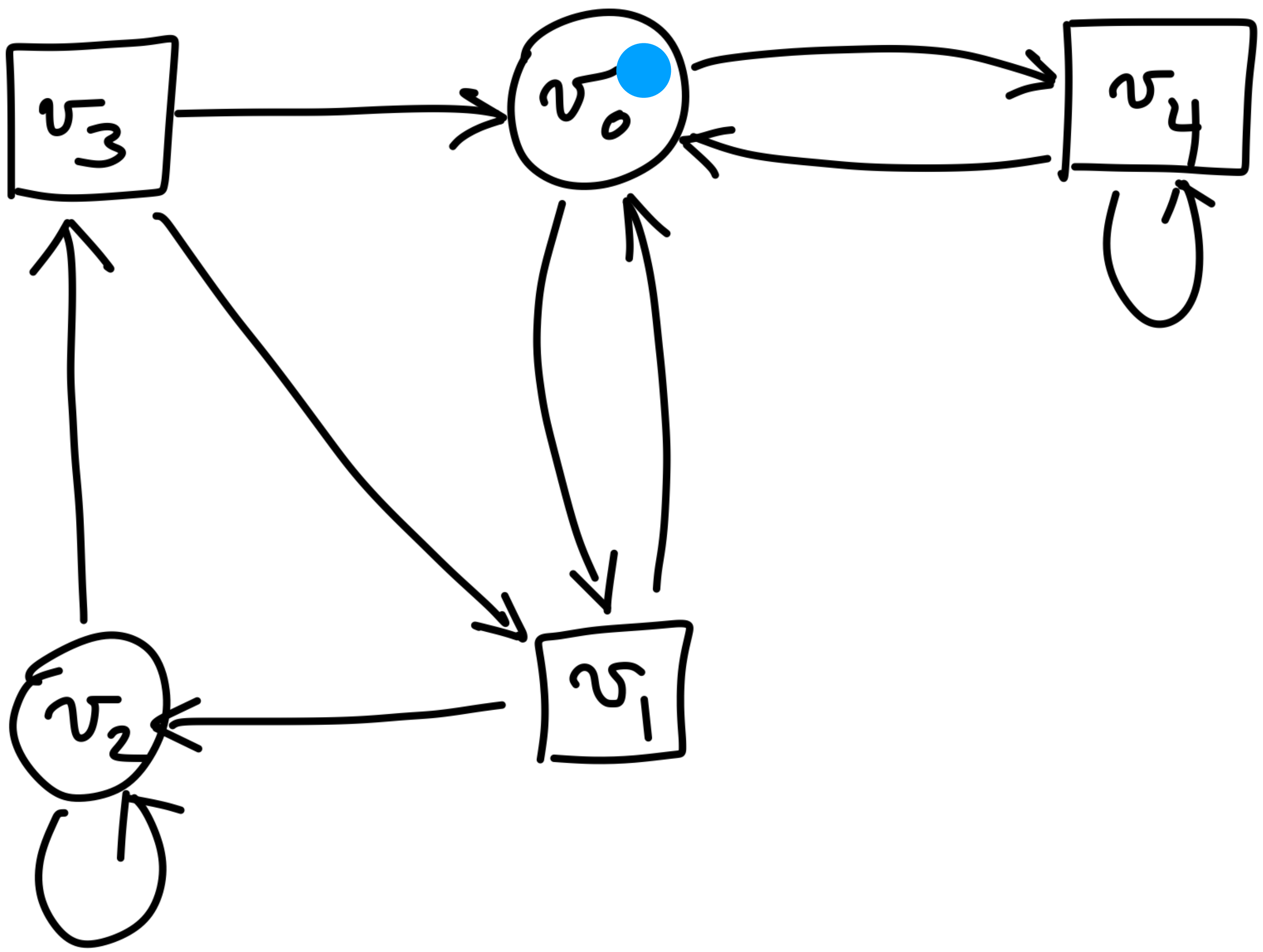
Vertices of Player □

Play: move a token along vertices

Infinite number of rounds
Outcome: infinite path

$v_0 \longrightarrow v_1$

2-player zero-sum games on graphs



Finite directed graphs
Vertices of Player ○

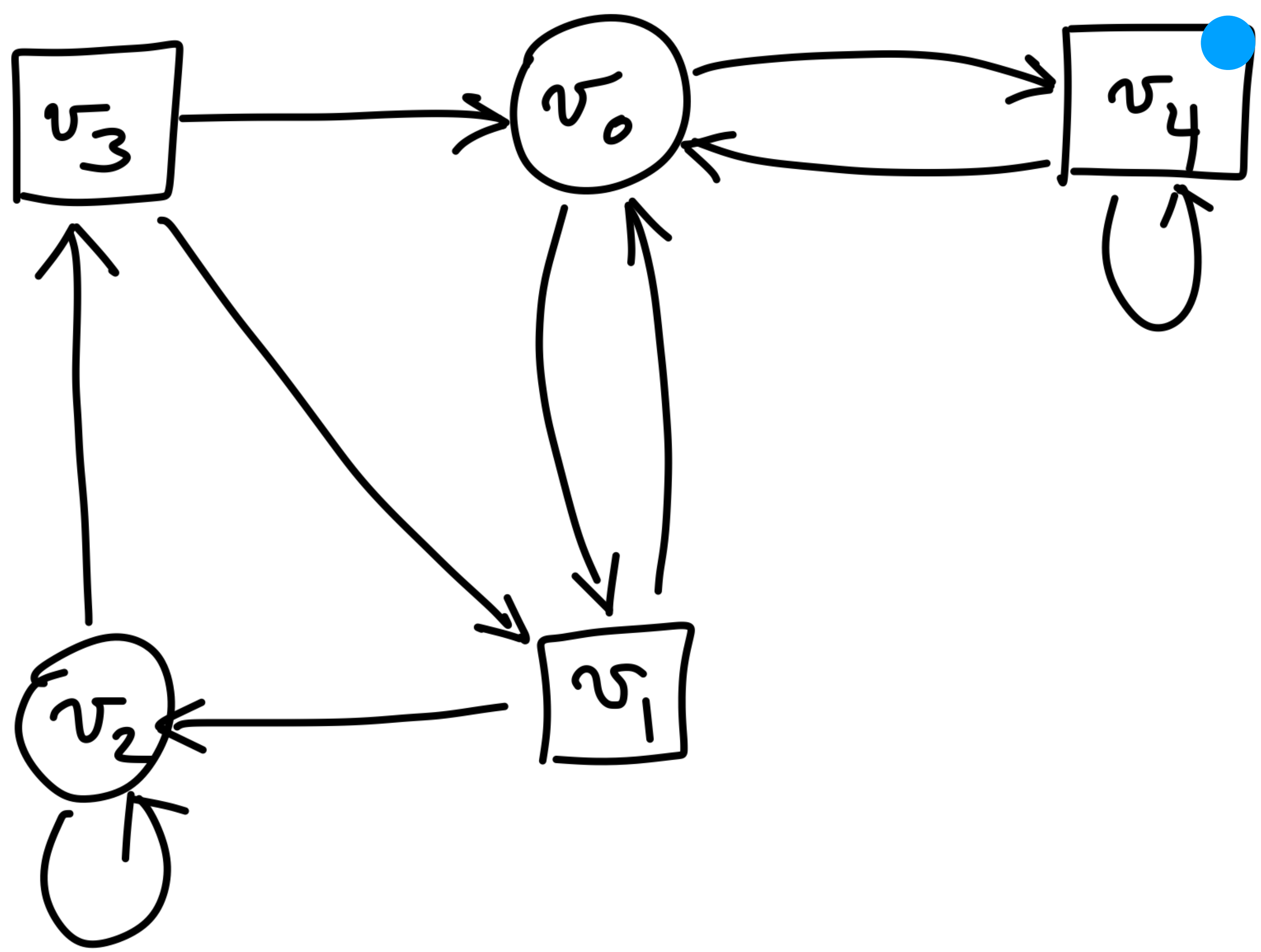
Vertices of Player □

Play: move a token along vertices

Infinite number of rounds
Outcome: infinite path

$$v_0 \longrightarrow v_1 \longrightarrow v_0$$

2-player zero-sum games on graphs



Finite directed graphs
Vertices of Player ◯

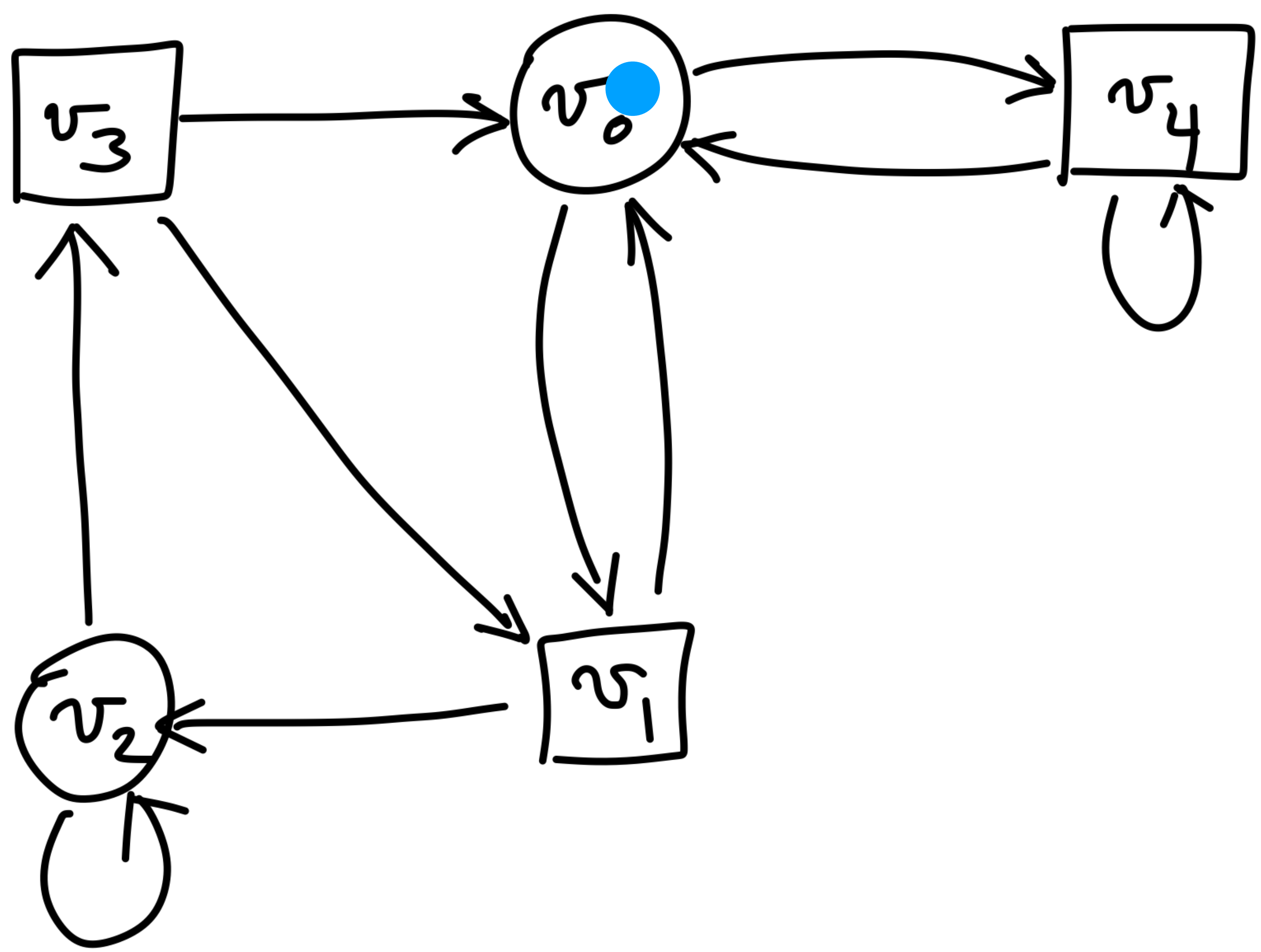
Vertices of Player □

Play: move a token along vertices

Infinite number of rounds
Outcome: infinite path

$$v_0 \longrightarrow v_1 \longrightarrow v_0 \longrightarrow v_4$$

2-player zero-sum games on graphs



Finite directed graphs
Vertices of Player ○

Vertices of Player □

Play: move a token along vertices

Infinite number of rounds
Outcome: infinite path

$$v_0 \longrightarrow v_1 \longrightarrow v_0 \longrightarrow v_4 \longrightarrow v_0 \ \cdots$$

Who is winning?

Who is winning?

$\mathrm{Win}_{\mathrm{O}} \subseteq V^{\omega}$ set of good outcomes for Player 1

Who is winning?

$\mathrm{Win}_{\bigcirc} \subseteq V^{\omega}$ set of good outcomes for Player 1

$\mathrm{Win}_{\square} = V^{\omega} \setminus \mathrm{Win}_{\bigcirc}$ (zero-sum game)

Who is winning?

$\mathrm{Win}_{\bigcirc} \subseteq V^{\omega}$ set of good outcomes for Player 1

$\mathrm{Win}_{\square} = V^{\omega} \backslash \mathrm{Win}_{\bigcirc}$ (zero-sum game)

Examples of winning conditions:

$\mathrm{Win}_{\bigcirc} = \{\pi \mid \pi \text{ visits } \text{Good}\}$ reachability

Who is winning?

$\mathrm{Win}_{\bigcirc} \subseteq V^{\omega}$ set of good outcomes for Player 1

$\mathrm{Win}_{\square} = V^{\omega} \setminus \mathrm{Win}_{\bigcirc}$ (zero-sum game)

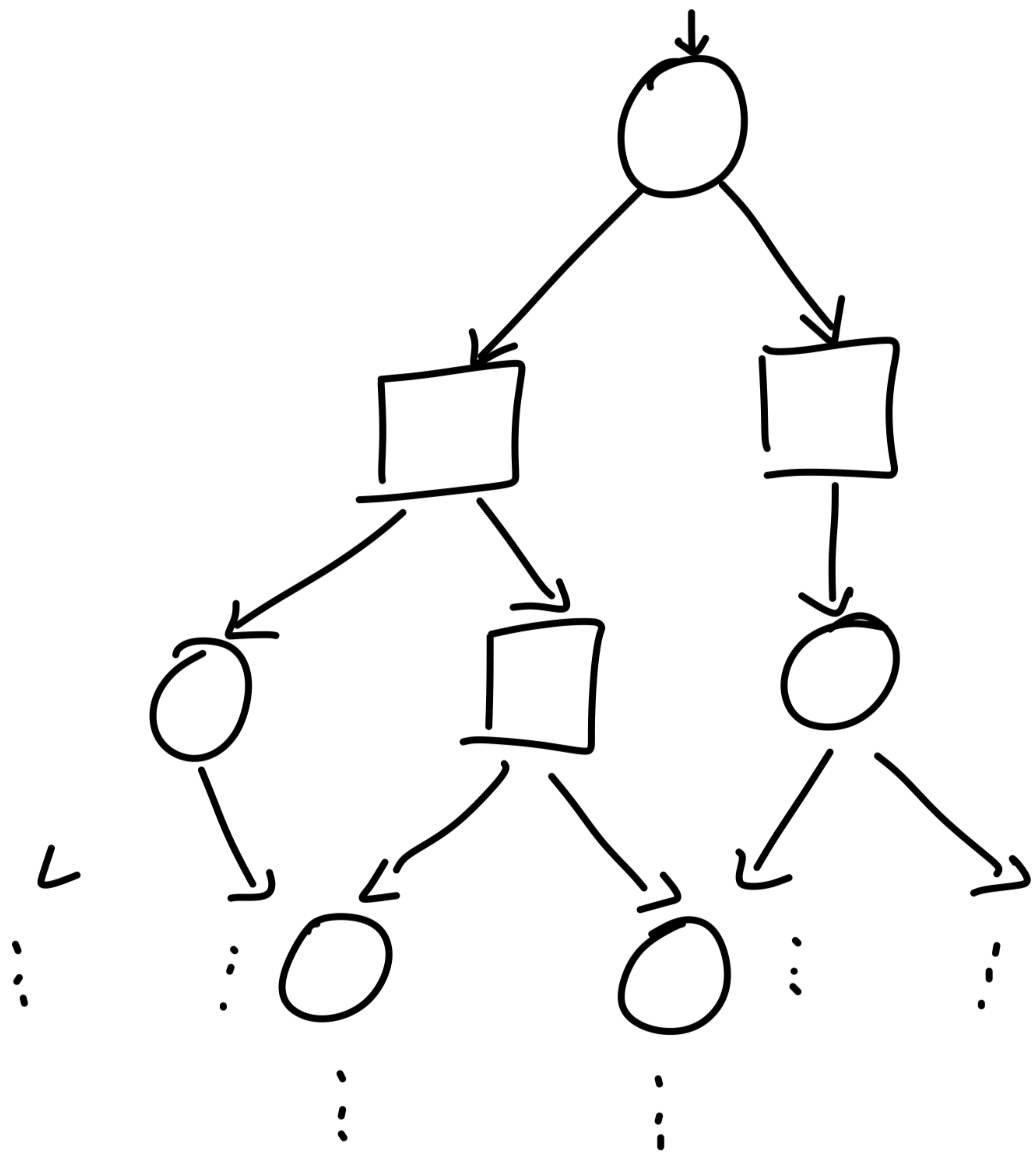
Examples of winning conditions:

$\mathrm{Win}_{\bigcirc} = \{\pi \mid \pi \text{ visits } \text{Good}\}$ reachability

$\mathrm{Win}_{\bigcirc} = \{\pi \mid \pi \text{ visits } \text{Good} \text{ infinitely often}\}$ Büchi

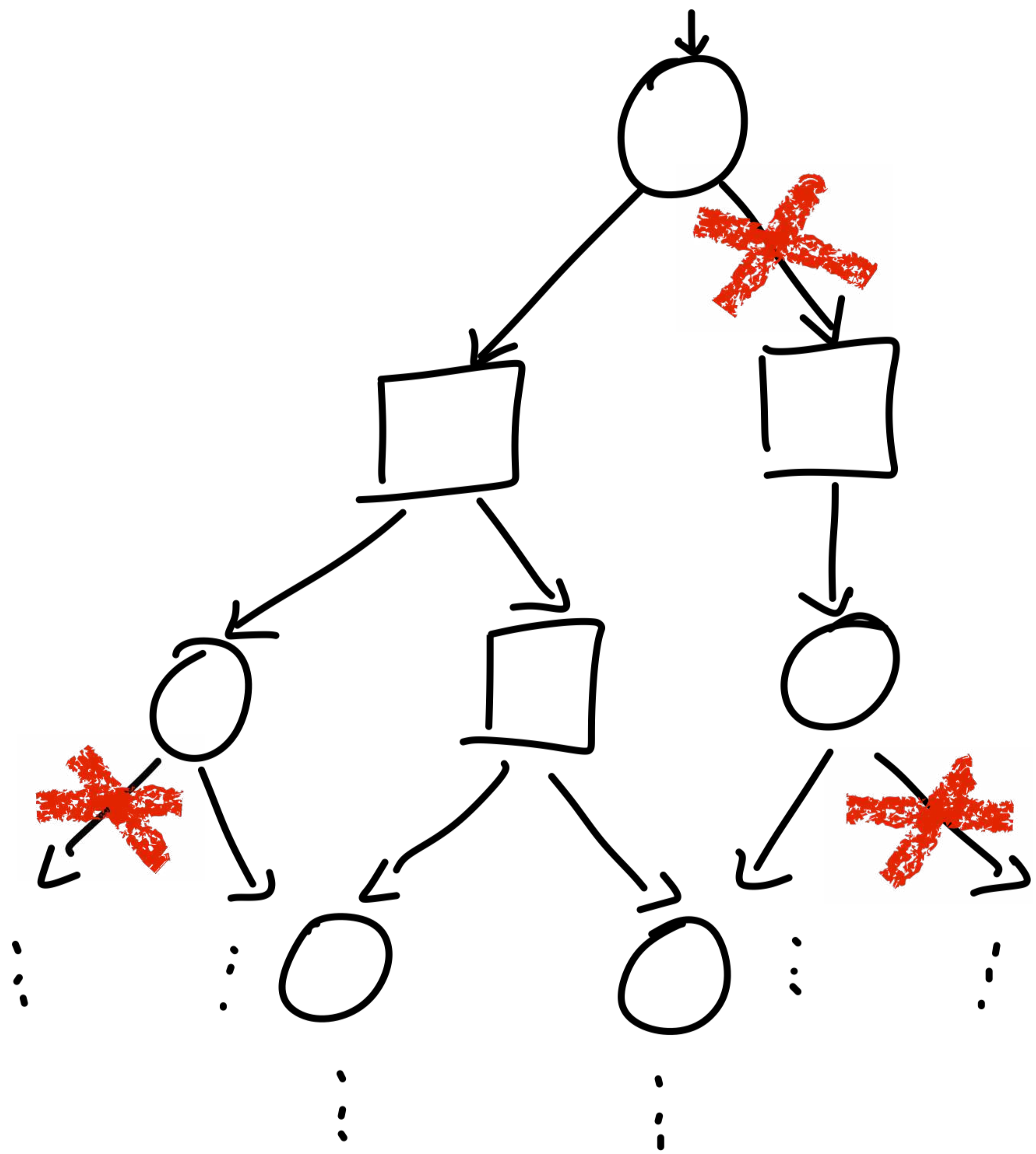
Strategies

Unfolding of the game graph:



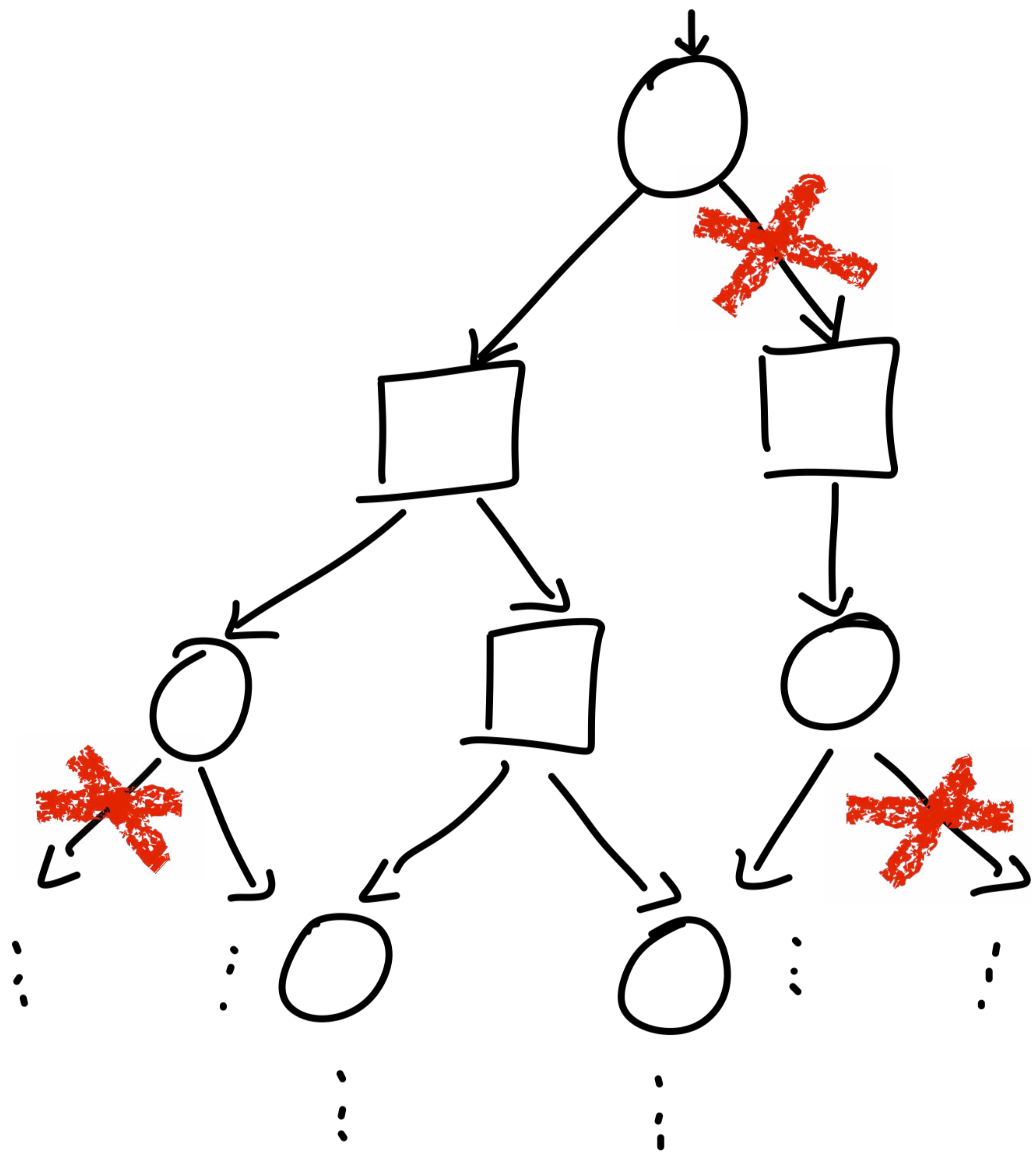
Strategies

Unfolding of the game graph:



Strategies

Unfolding of the game graph:

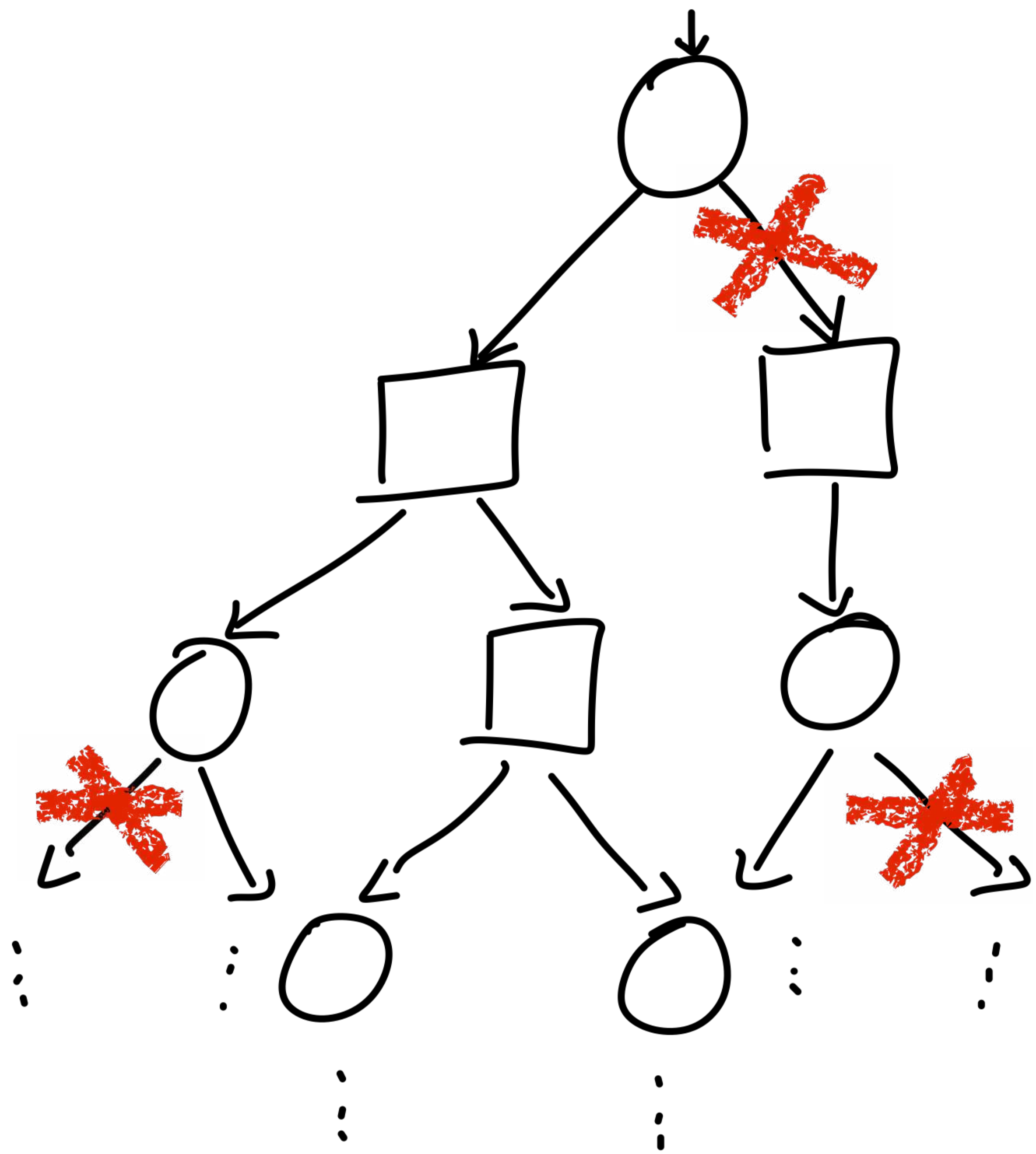


Strategy for Player ◯: one choice in each node of Player ◯ in unfolding

$$\sigma_{\mathrm{O}}: V^* V_{\mathrm{O}} \to E$$

Strategies

Unfolding of the game graph:



Strategy for Player ◯: one choice in each node of Player ◯ in unfolding

$$\sigma_{\mathrm{O}}\colon V^{*}V_{\mathrm{O}} \to E$$

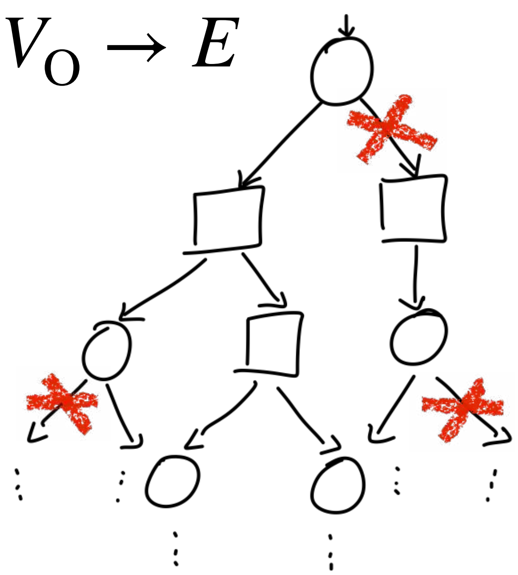
Strategy is **winning** if **all paths** of the resulting tree are winning

Types of strategies

Types of strategies

Strategy (infinite memory)

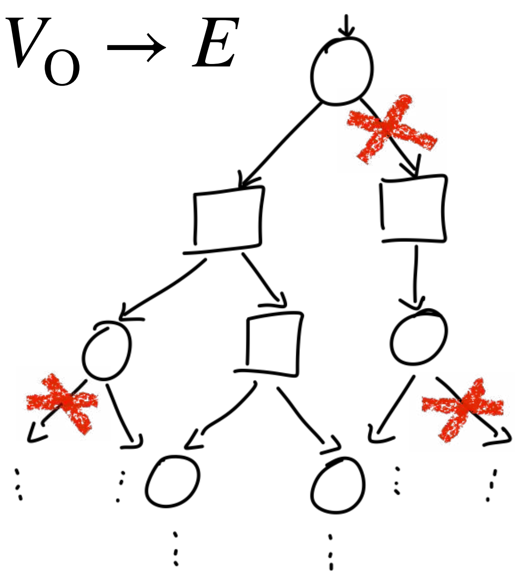
$\sigma_{\mathrm{O}}\colon V^*V_{\mathrm{O}} \to E$



Types of strategies

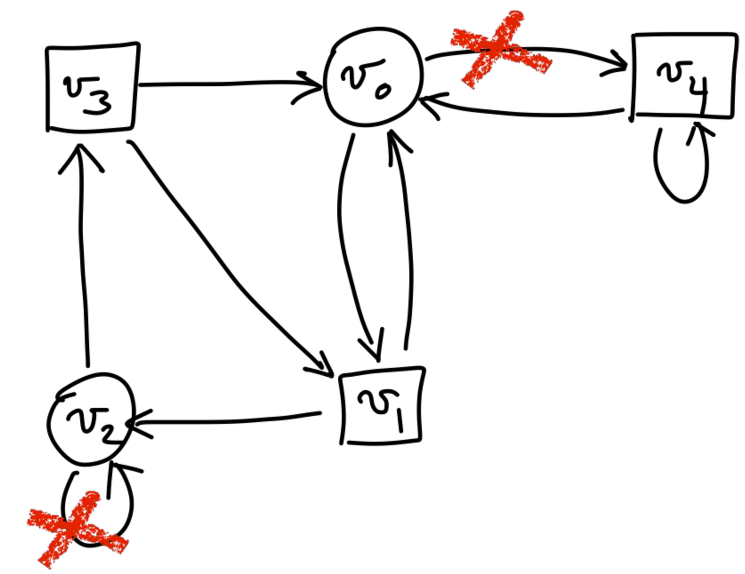
Strategy (infinite memory)

$\sigma_O \colon V^*V_O \to E$



Memoryless/positional strategy

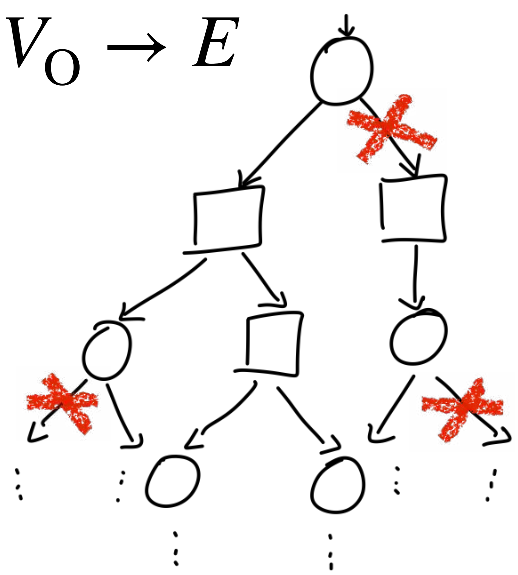
$\sigma_O \colon V_O \to E$



Types of strategies

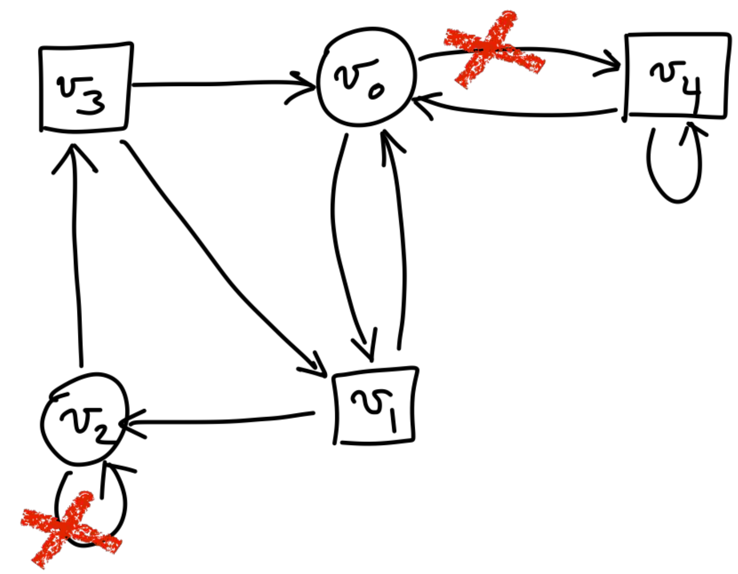
Strategy (infinite memory)

$\sigma_{\mathrm{O}}: V^*V_{\mathrm{O}} \to E$



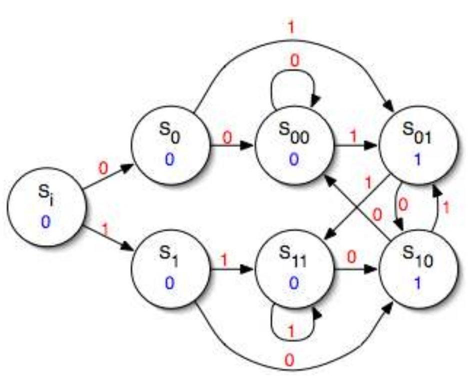
Memoryless/positional strategy

$\sigma_{\mathrm{O}}: V_{\mathrm{O}} \to E$



Finite memory strategy

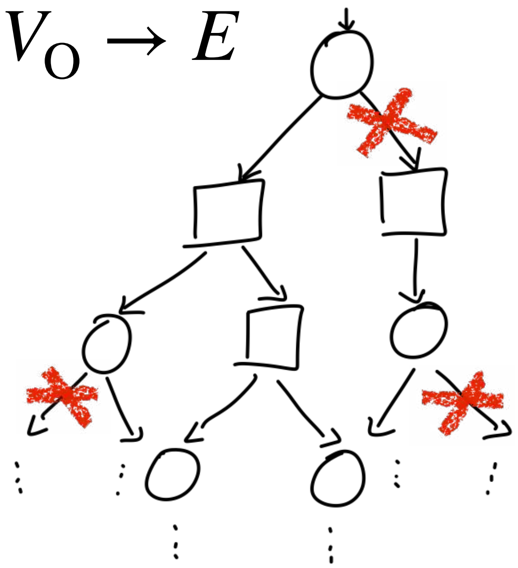
$\sigma_{\mathrm{O}}: V^*V_{\mathrm{O}} \to E$ representable with a Moore machine



Types of strategies

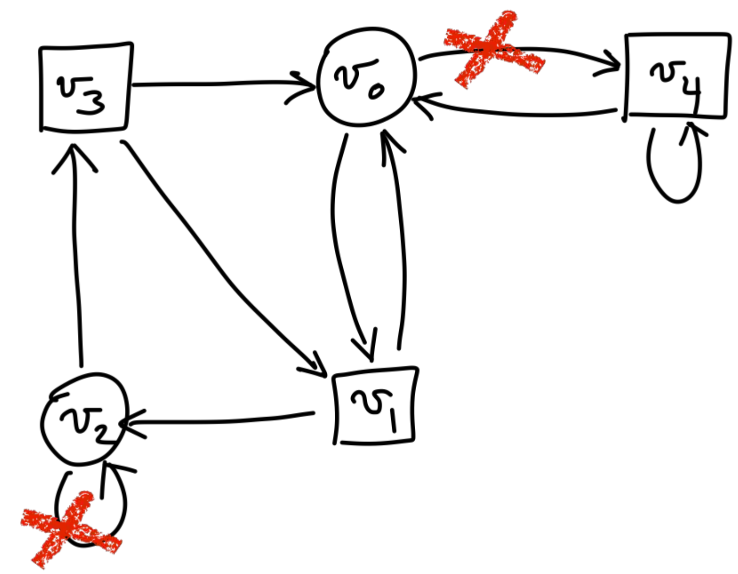
Strategy (infinite memory)

$\sigma_O : V^*V_O \to E$



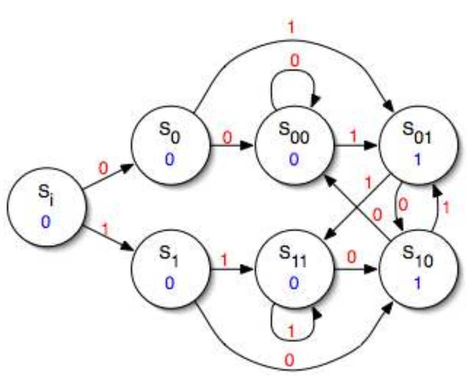
Memoryless/positional strategy

$\sigma_O : V_O \to E$



Finite memory strategy

$\sigma_O : V^*V_O \to E$ representable with a Moore machine



Randomised strategy

$\sigma_O : V^*V_O \to \mathrm{Distr}(E)$

Decision problem

Given a game graph G and a winning condition $\mathrm{Win}_{\bigcirc}$ decide if Player $\bigcirc$ has a winning strategy.

Decision problem

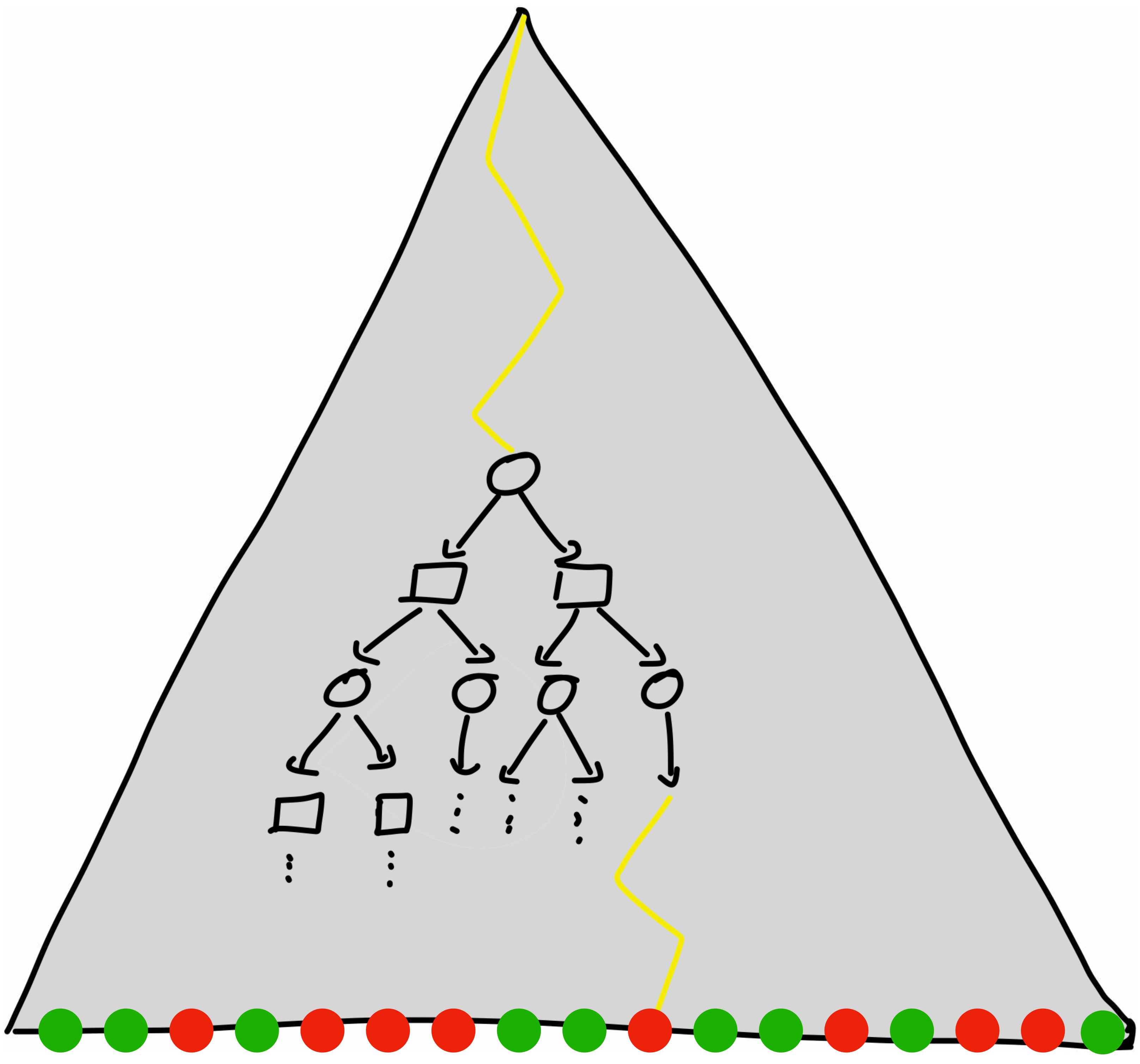
Given a game graph G and a winning condition $\mathrm{Win}_{\bigcirc}$
decide if Player $\bigcirc$ has a winning strategy.

What about Player $\square$?
Determinacy (true in a large class of objectives, e.g. all ω-regular objectives)

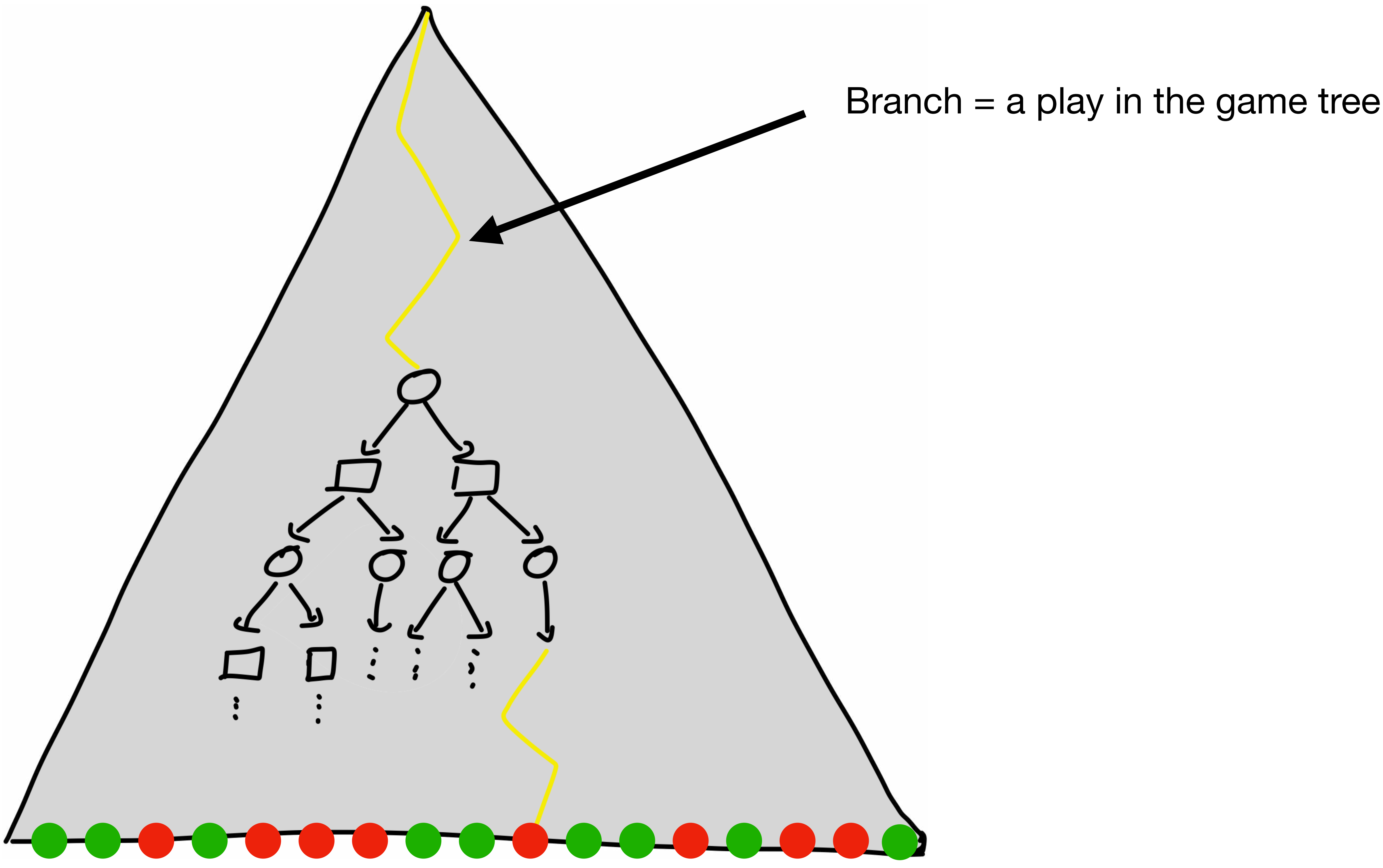
either Player $\bigcirc$ has a winning strategy for $\mathrm{Win}_{\bigcirc}$

or Player $\square$ has a winning strategy for $\mathrm{Win}_{\square} = V^{\omega} \setminus \mathrm{Win}_{\bigcirc}$

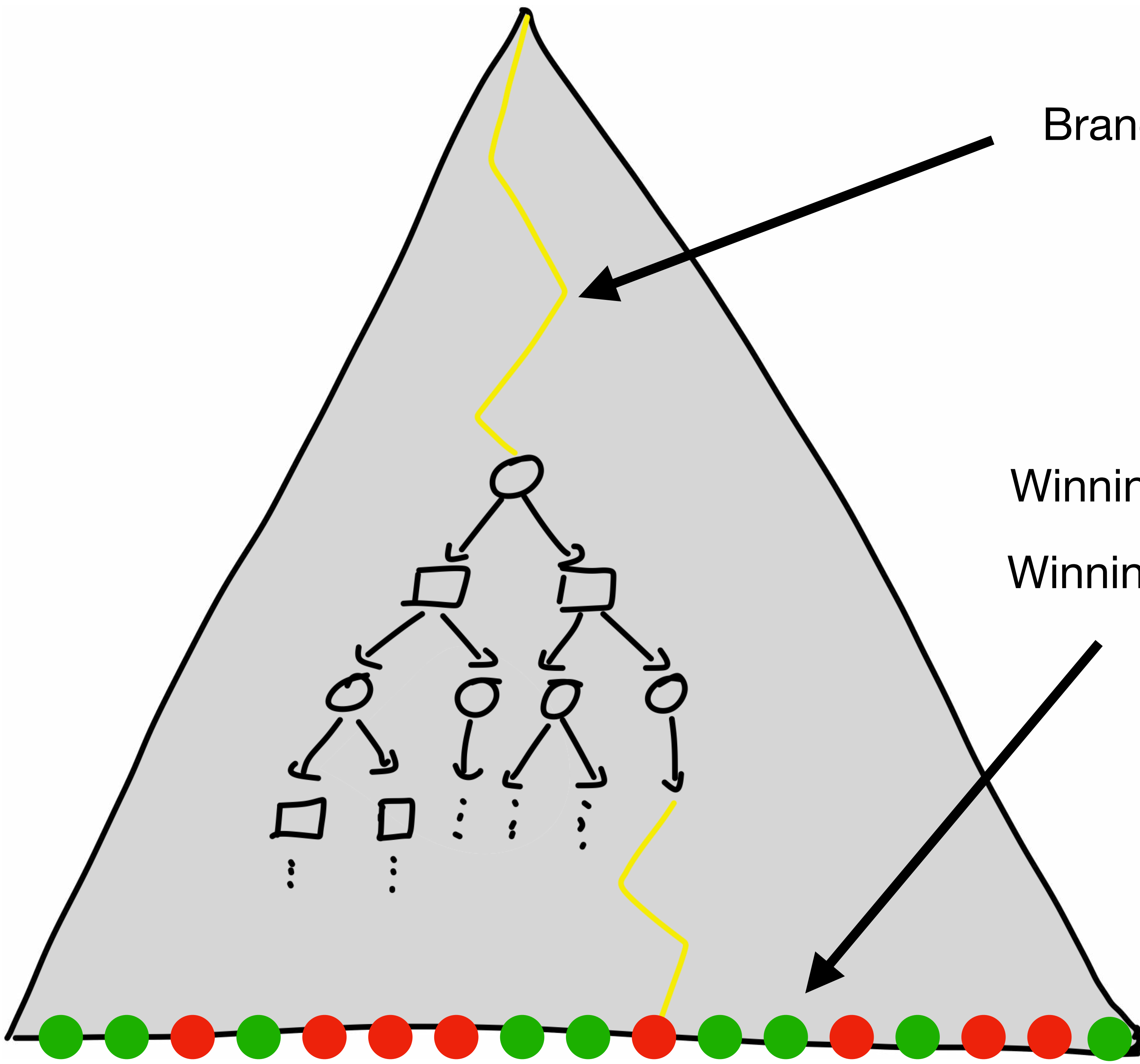
Example: finite trees



Example: finite trees



Example: finite trees

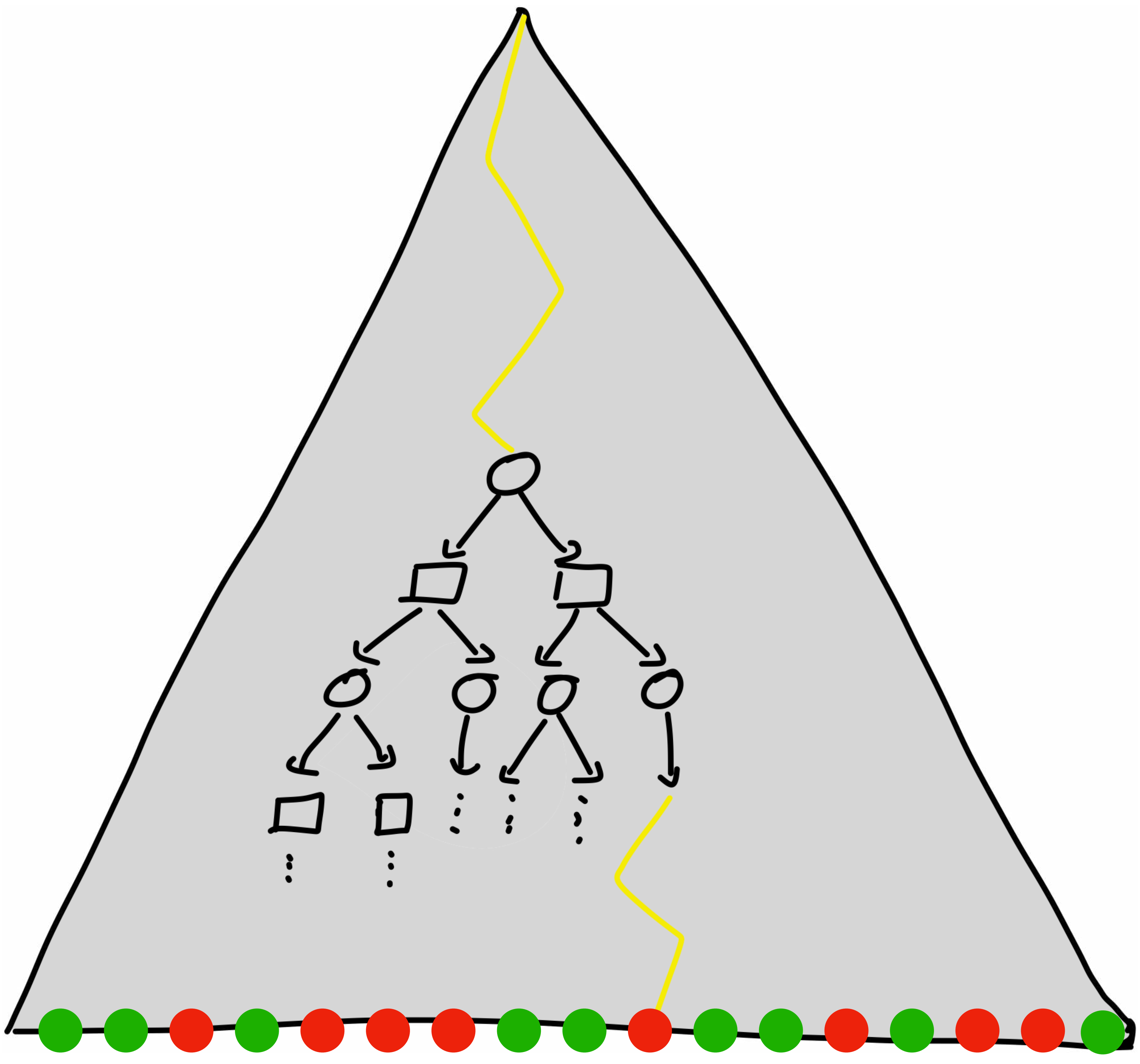


Branch = a play in the game tree

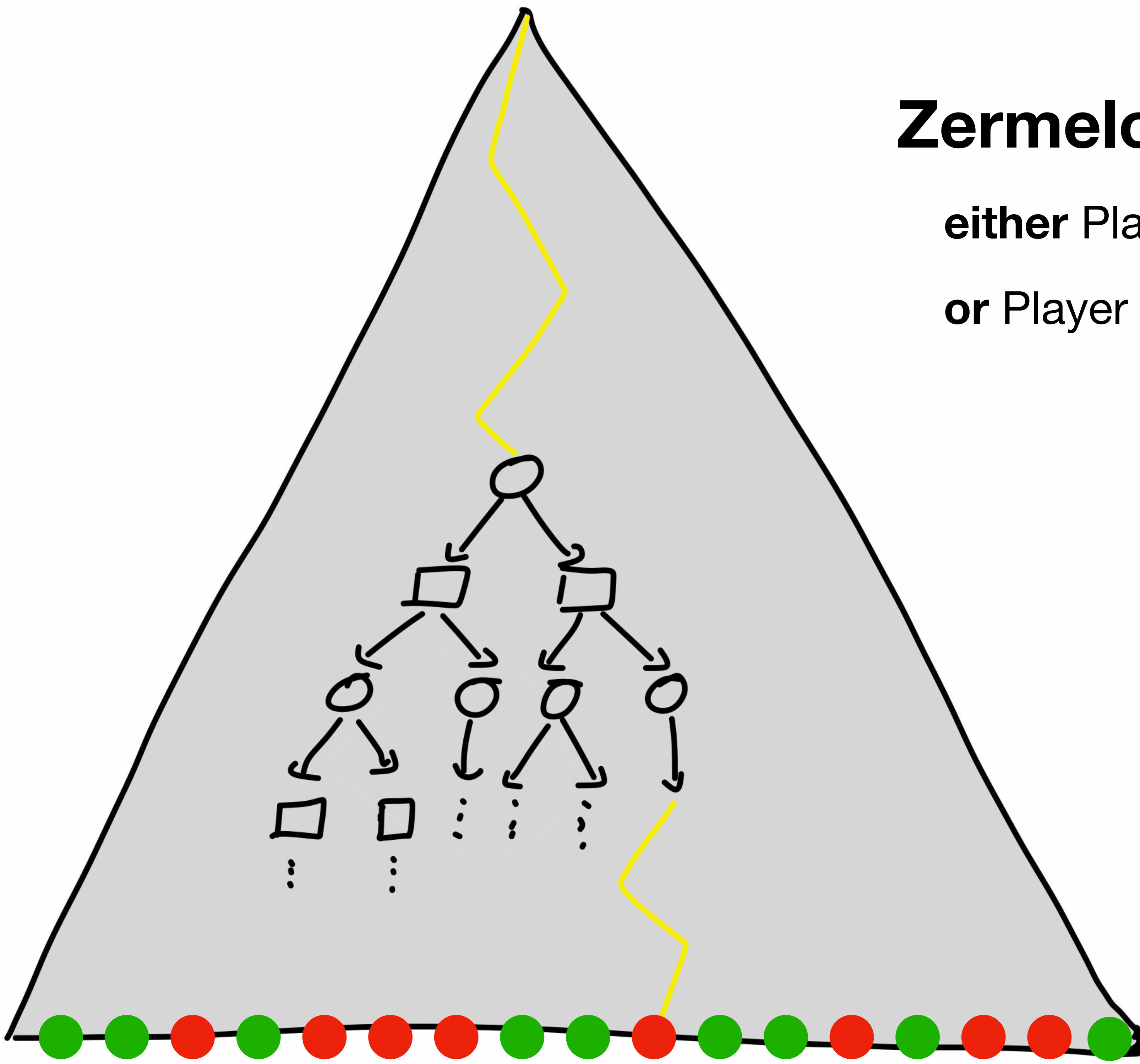
Winning for Player ○ if ends in ●

Winning for Player □ if ends in ●

Example: finite trees



Example: finite trees

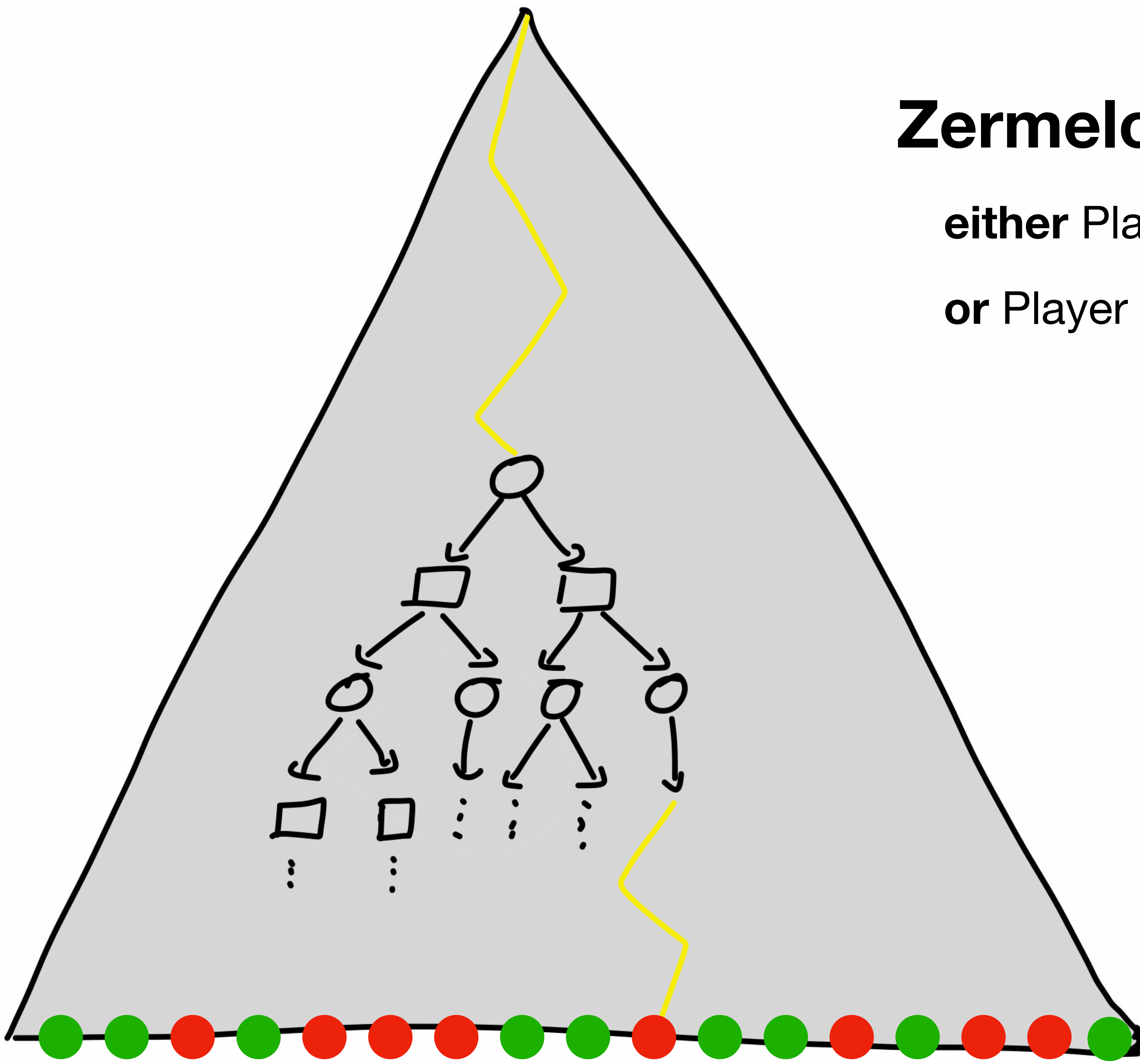


Zermelo's theorem

either Player has a strategy to force

or Player has a strategy to force

Example: finite trees



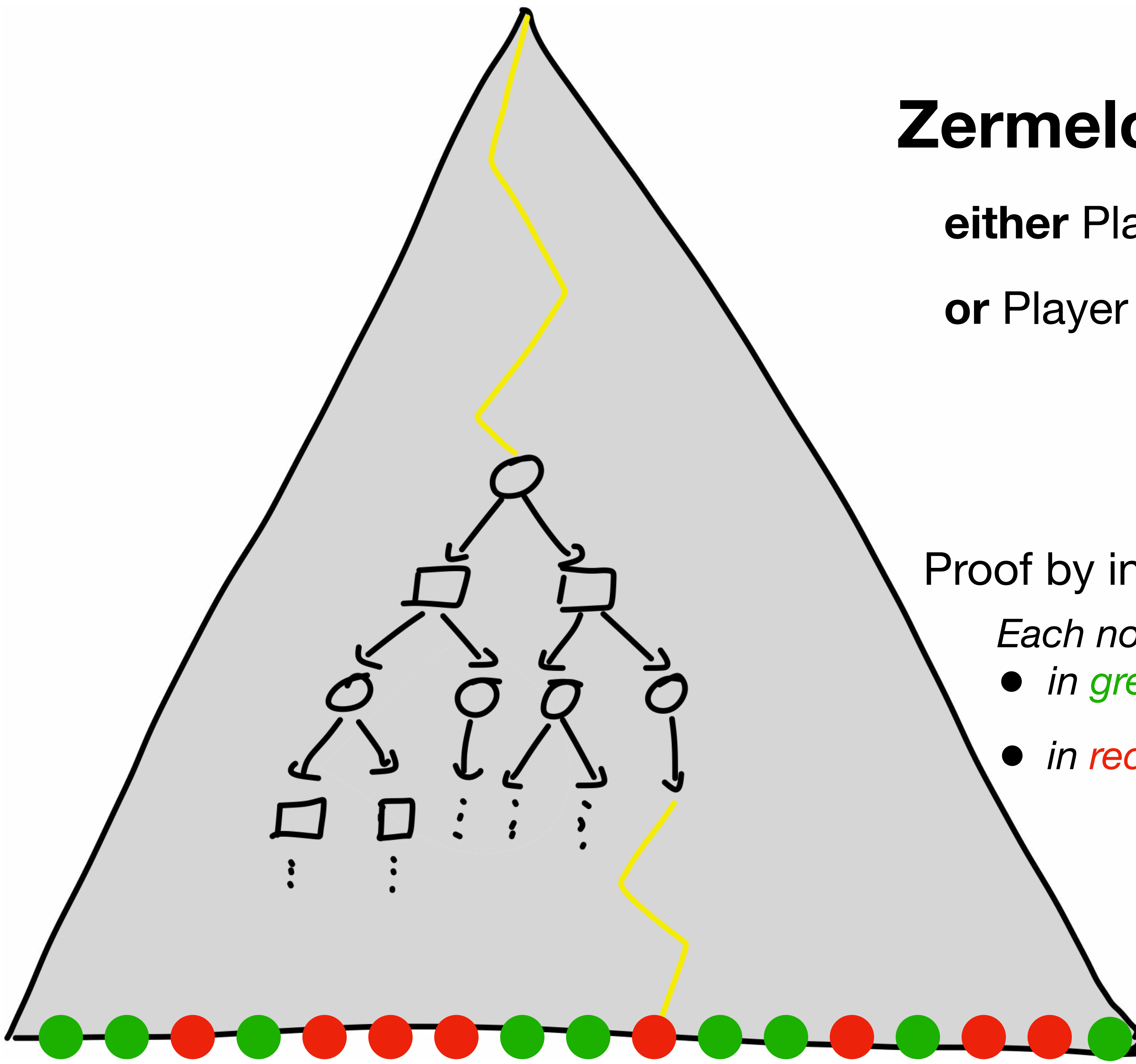
Zermelo's theorem

either Player ○ has a strategy to force ●

or Player □ has a strategy to force ●

= **determinacy**

Example: finite trees



Zermelo's theorem

either Player ◯ has a strategy to force 🟢

or Player □ has a strategy to force 🔴

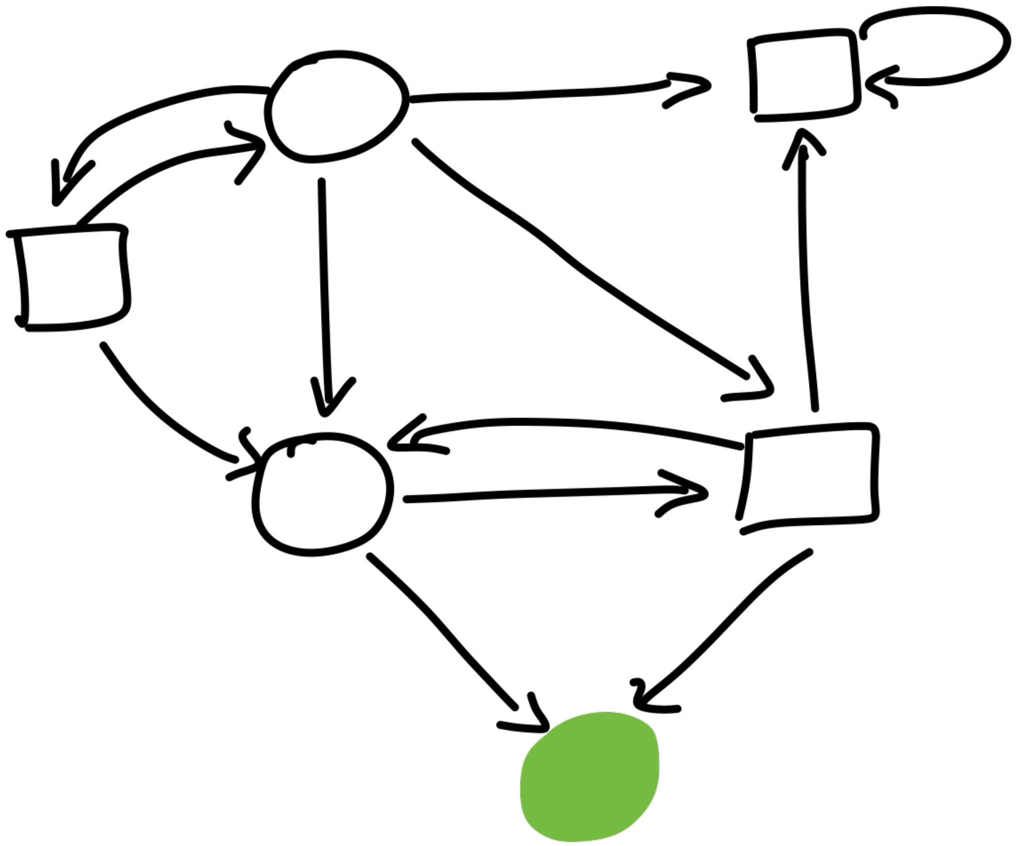
= determinacy

Proof by induction on the depth of the tree

Each node can be labelled bottom-up:

- *in green if Player ◯ can force 🟢 from there*
- *in red if Player □ can force 🔴 from there*

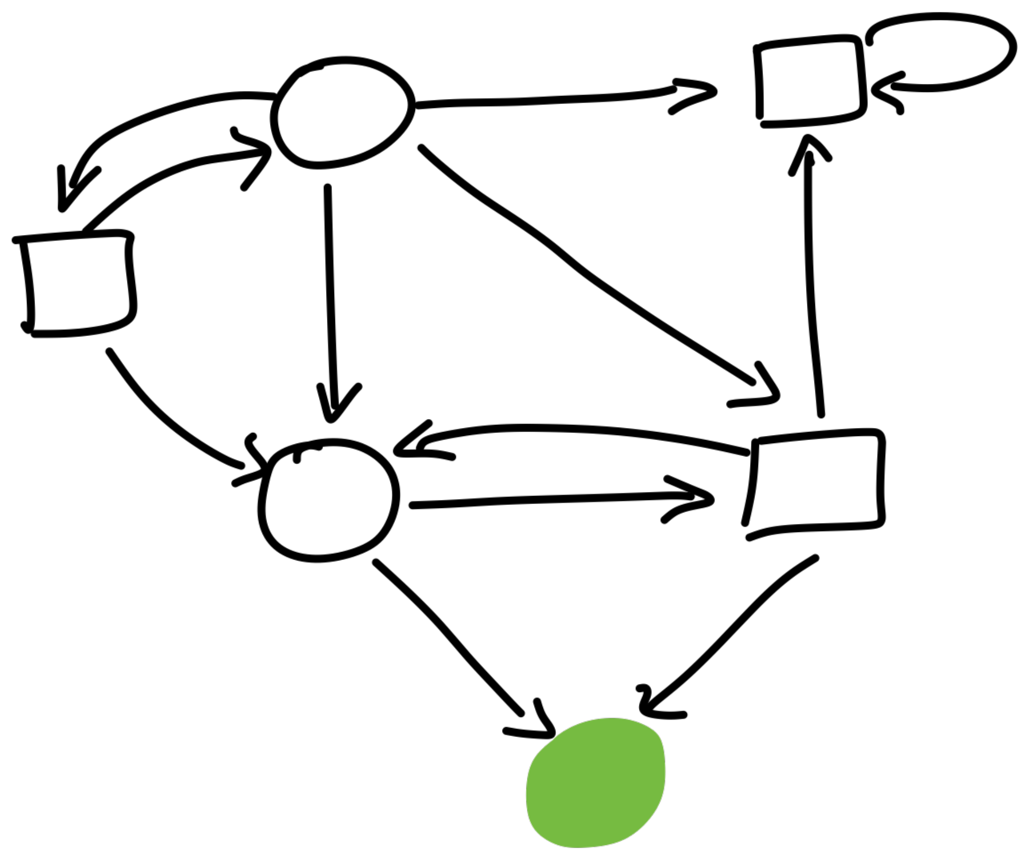
Example: reachability in graphs



$$\mathrm{Win}_{\bigcirc} = \{\pi \mid \pi \text{ visits } \text{Good}\}$$

$$\mathrm{Win}_{\square} = \{\pi \mid \pi \text{ avoids } \text{Good}\}$$

Example: reachability in graphs

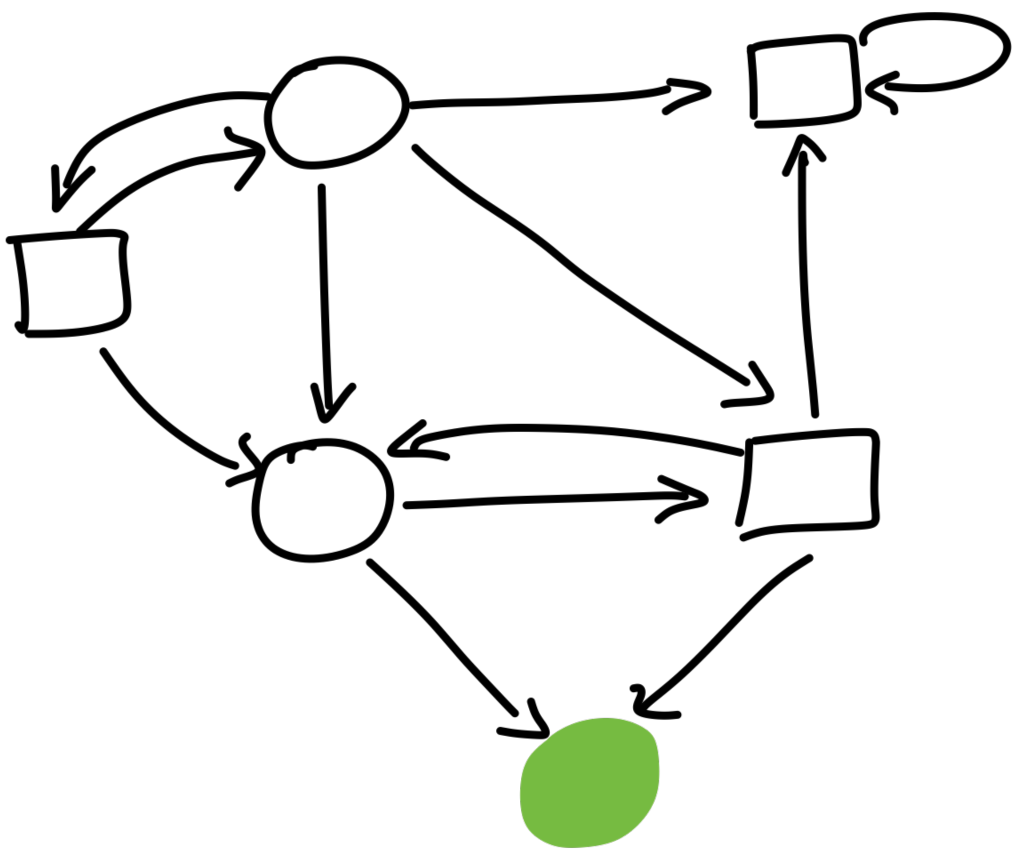


$$\mathrm{Win}_{\bigcirc} = \{\pi \mid \pi \text{ visits } \textcolor{green}{\mathrm{Good}}\}$$

$$\mathrm{Win}_{\square} = \{\pi \mid \pi \text{ avoids } \textcolor{green}{\mathrm{Good}}\}$$

Apply the same bottom-up rule…

Example: reachability in graphs



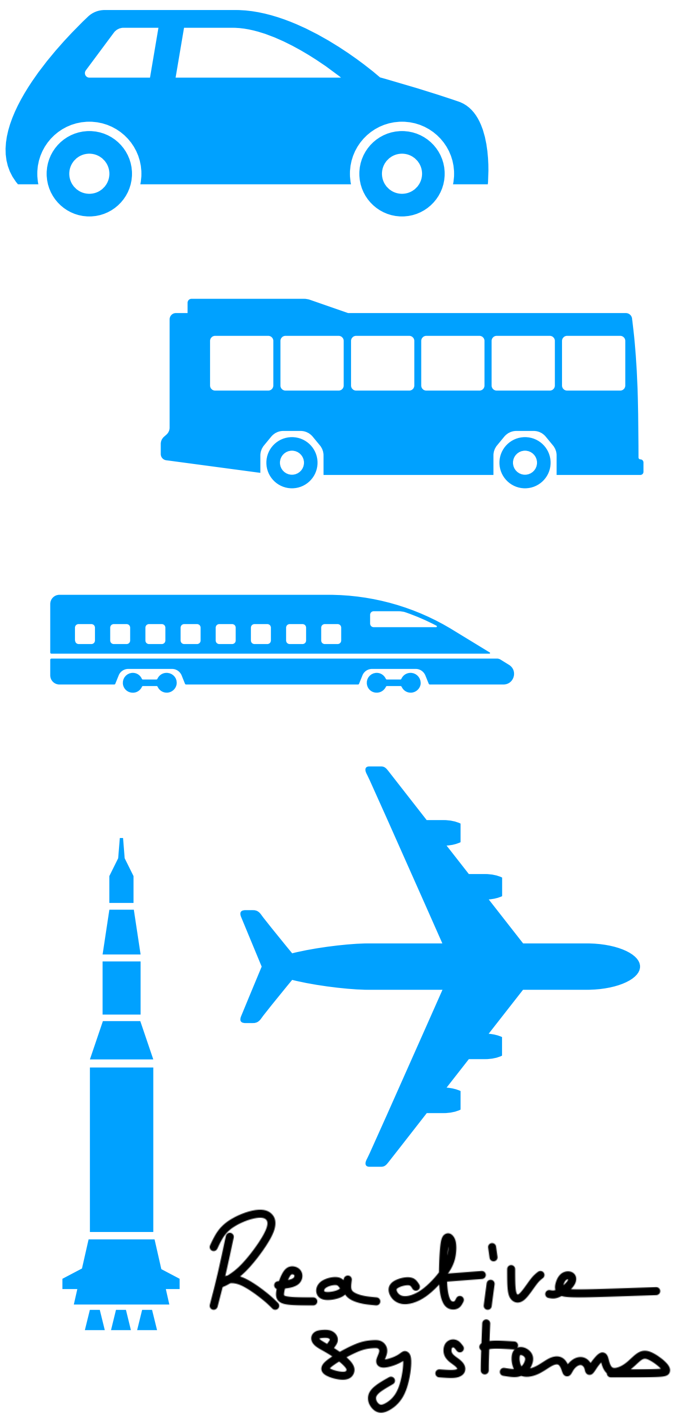
$$\mathrm{Win}_{\bigcirc} = \{\pi \mid \pi \text{ visits } \text{Good}\}$$

$$\mathrm{Win}_{\square} = \{\pi \mid \pi \text{ avoids } \text{Good}\}$$

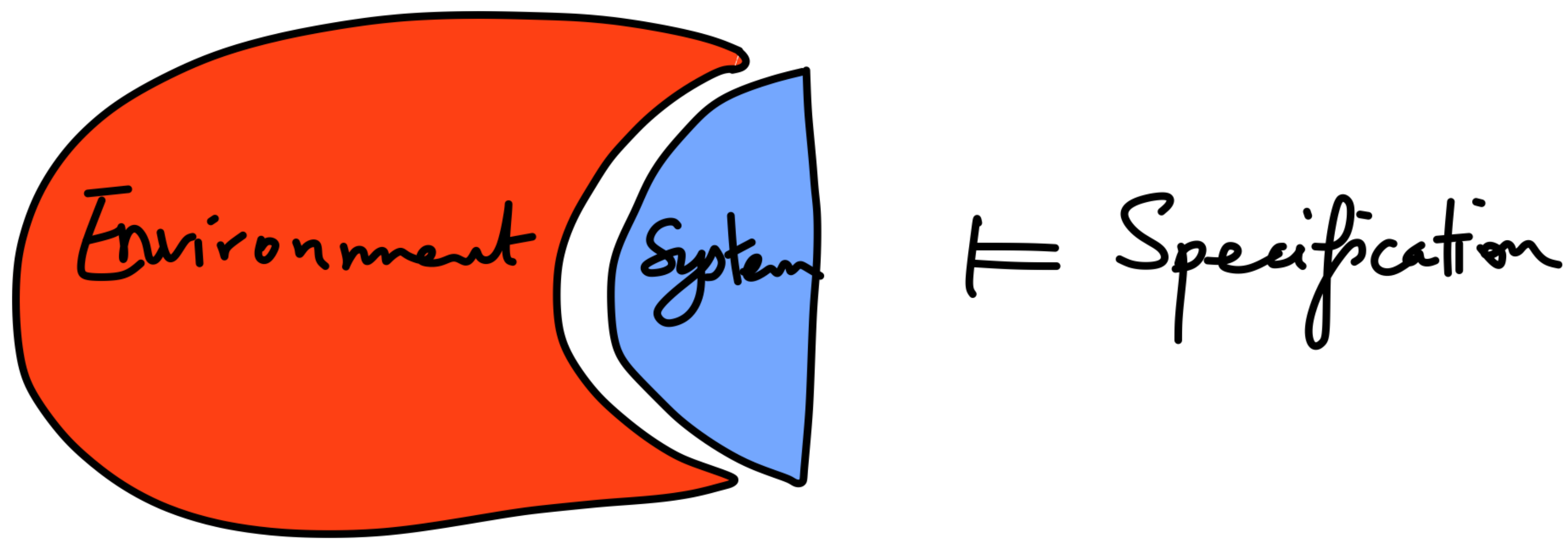
Apply the same bottom-up rule…

…to decide the winner and find winning strategies

Games for synthesis



Crucial to make the critical programs **correct**

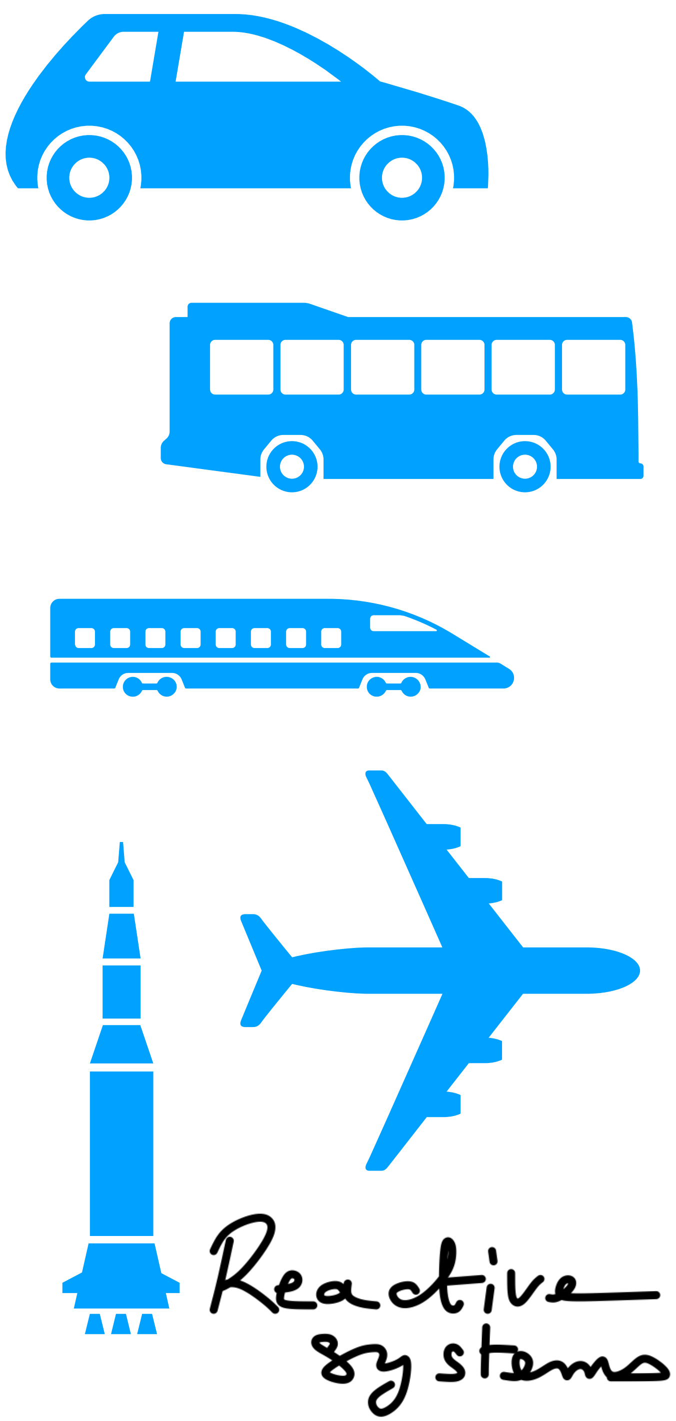


Instead of verifying an existing system...

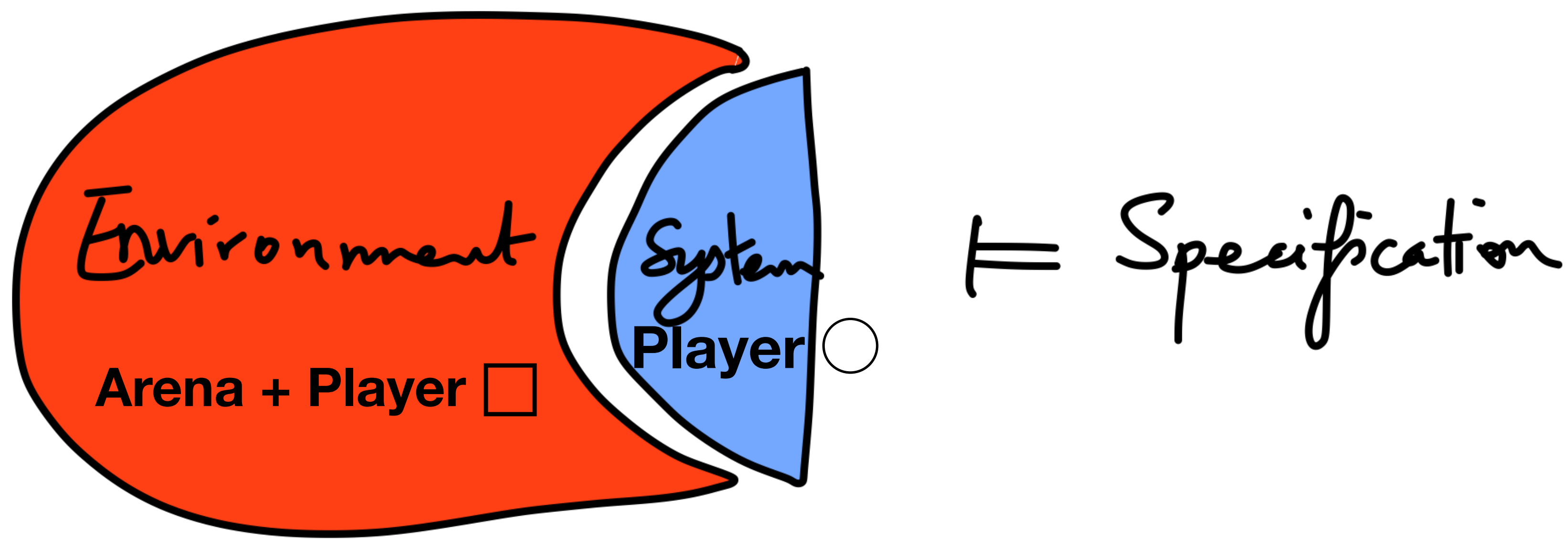
Synthesise a correct-by-design one!

Winning strategy = Correct system

Games for synthesis



Crucial to make the critical programs **correct**

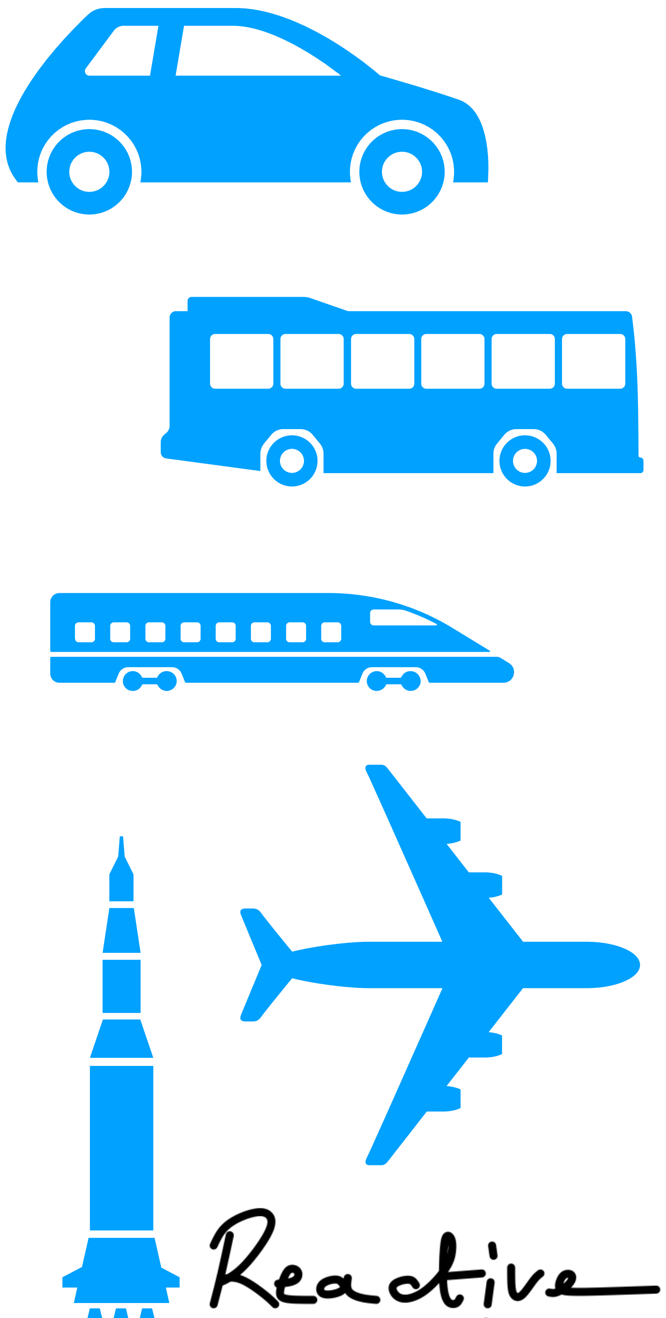


Instead of verifying an existing system...

Synthesise a correct-by-design one!

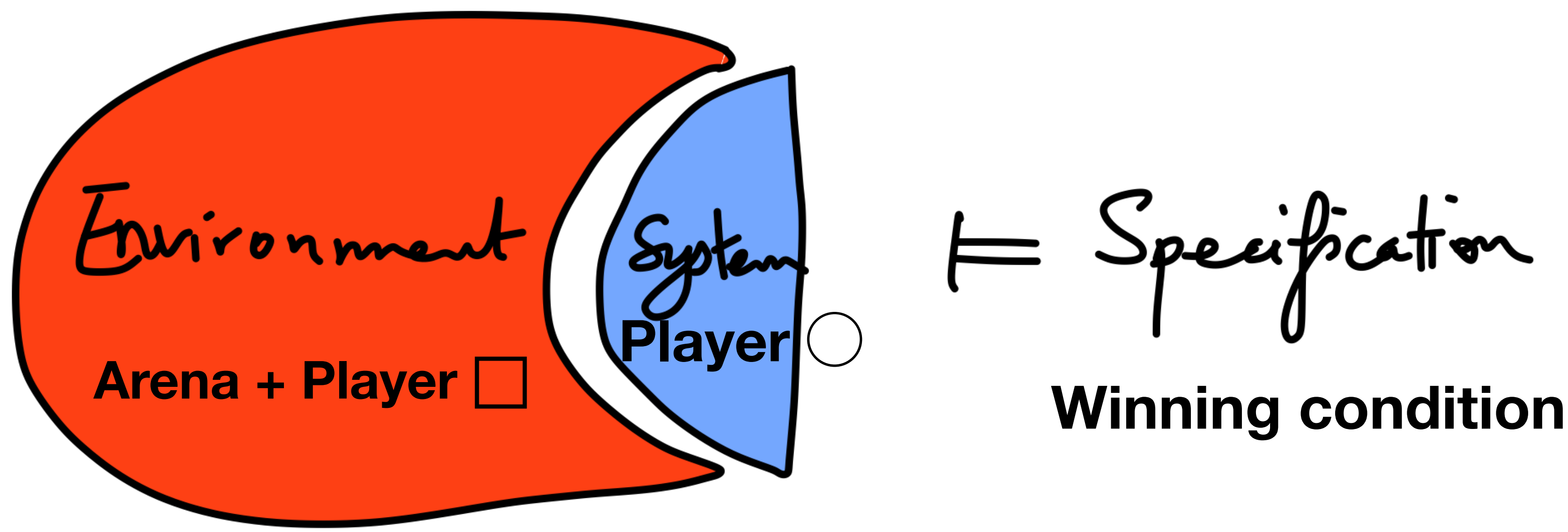
Winning strategy = Correct system

Games for synthesis



Reactive systems

Crucial to make the critical programs **correct**

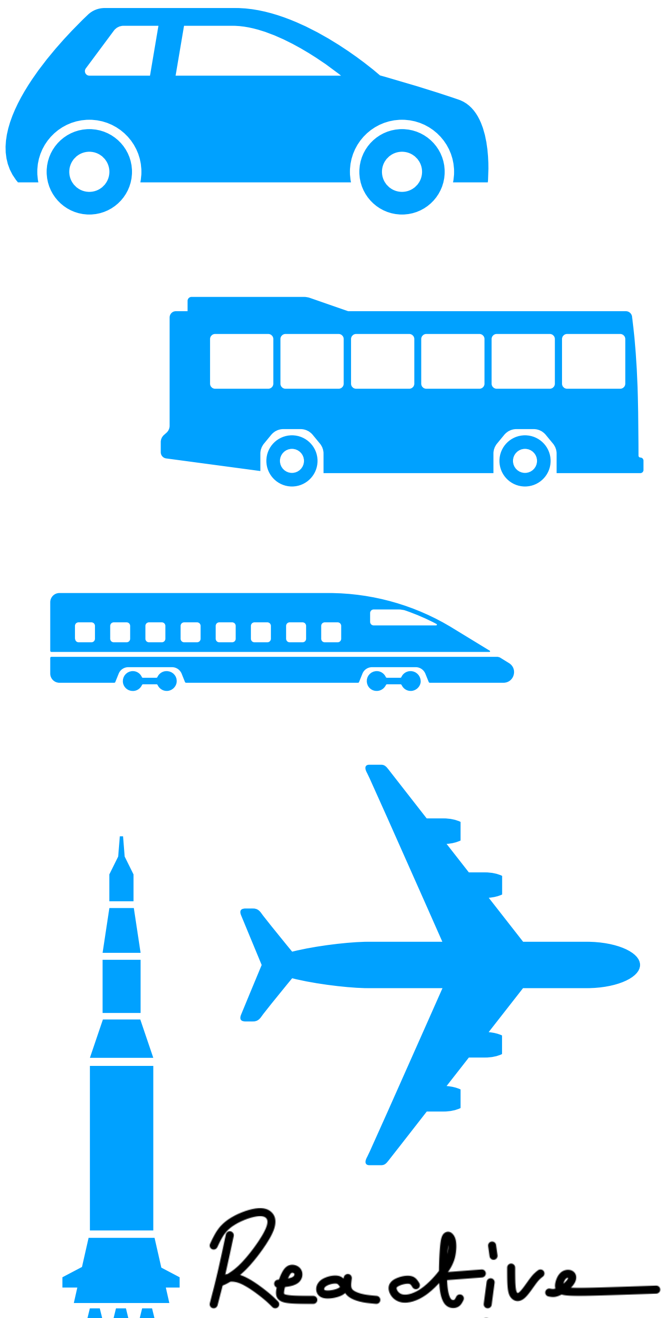


Instead of verifying an existing system...

Synthesise a correct-by-design one!

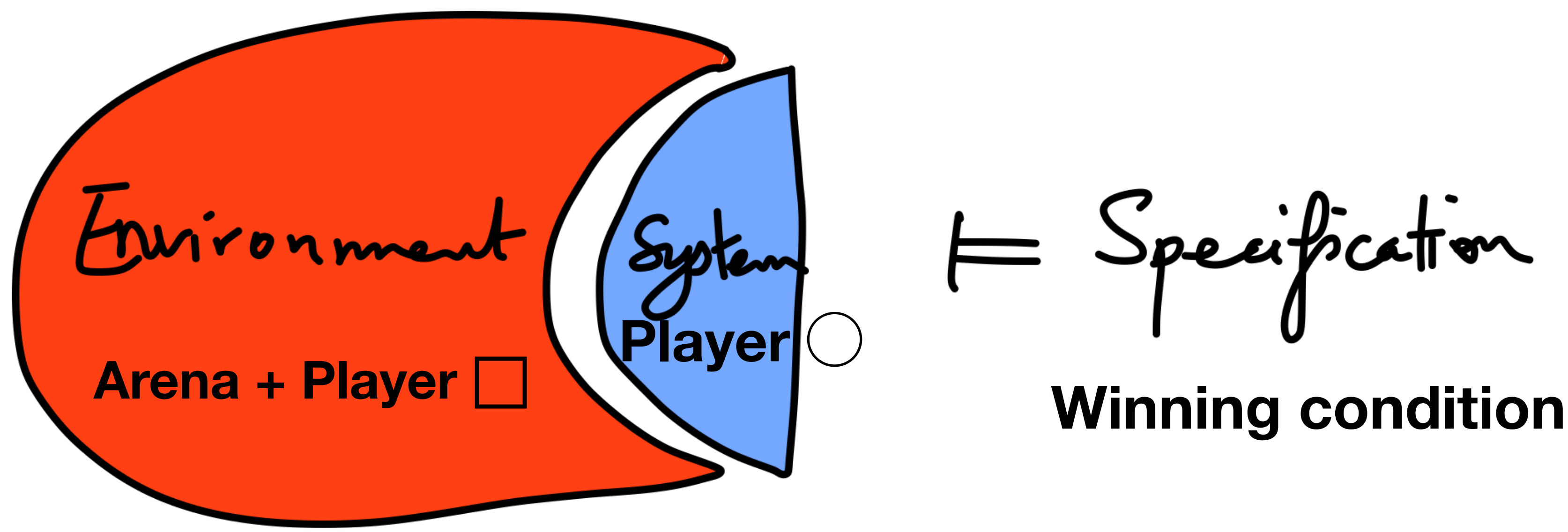
Winning strategy = Correct system

Games for synthesis



Reactive systems

Crucial to make the critical programs **correct**



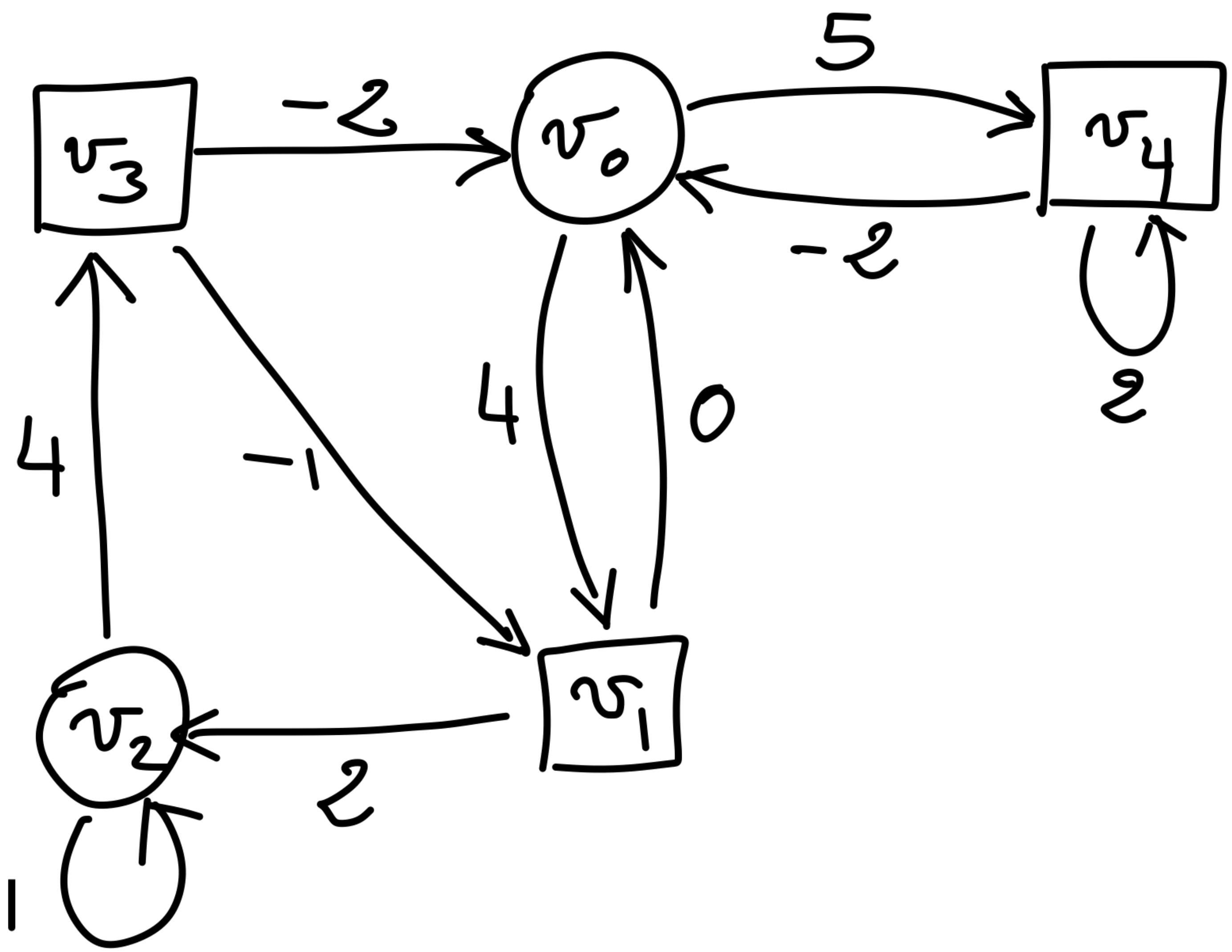
Instead of…

What if several *winning* strategies for Player ○?

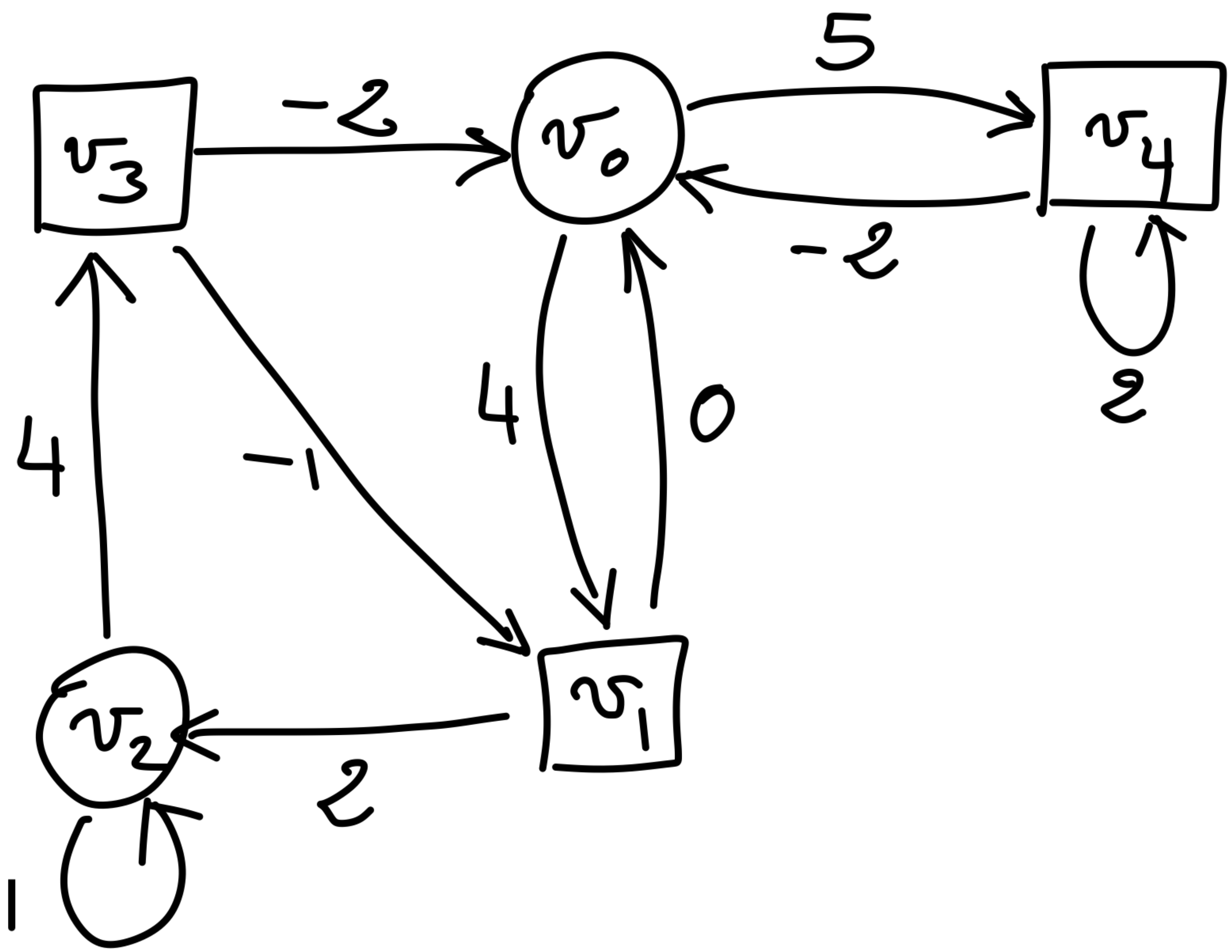
Need for a quality measure, to choose the *best* one…

…ng strategy = Correct system

Quantitative games on graphs

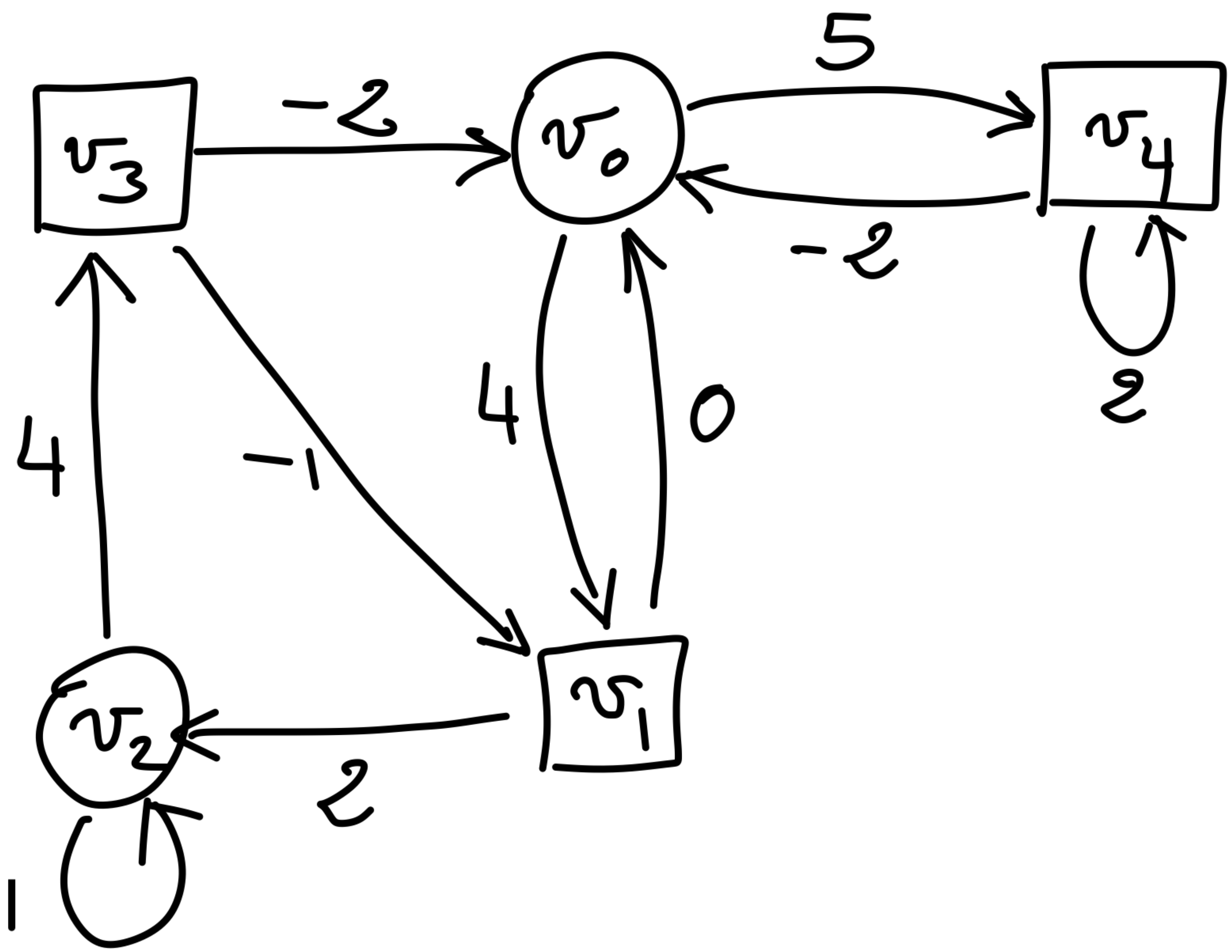


Quantitative games on graphs



Weighted graph: weights=rewards

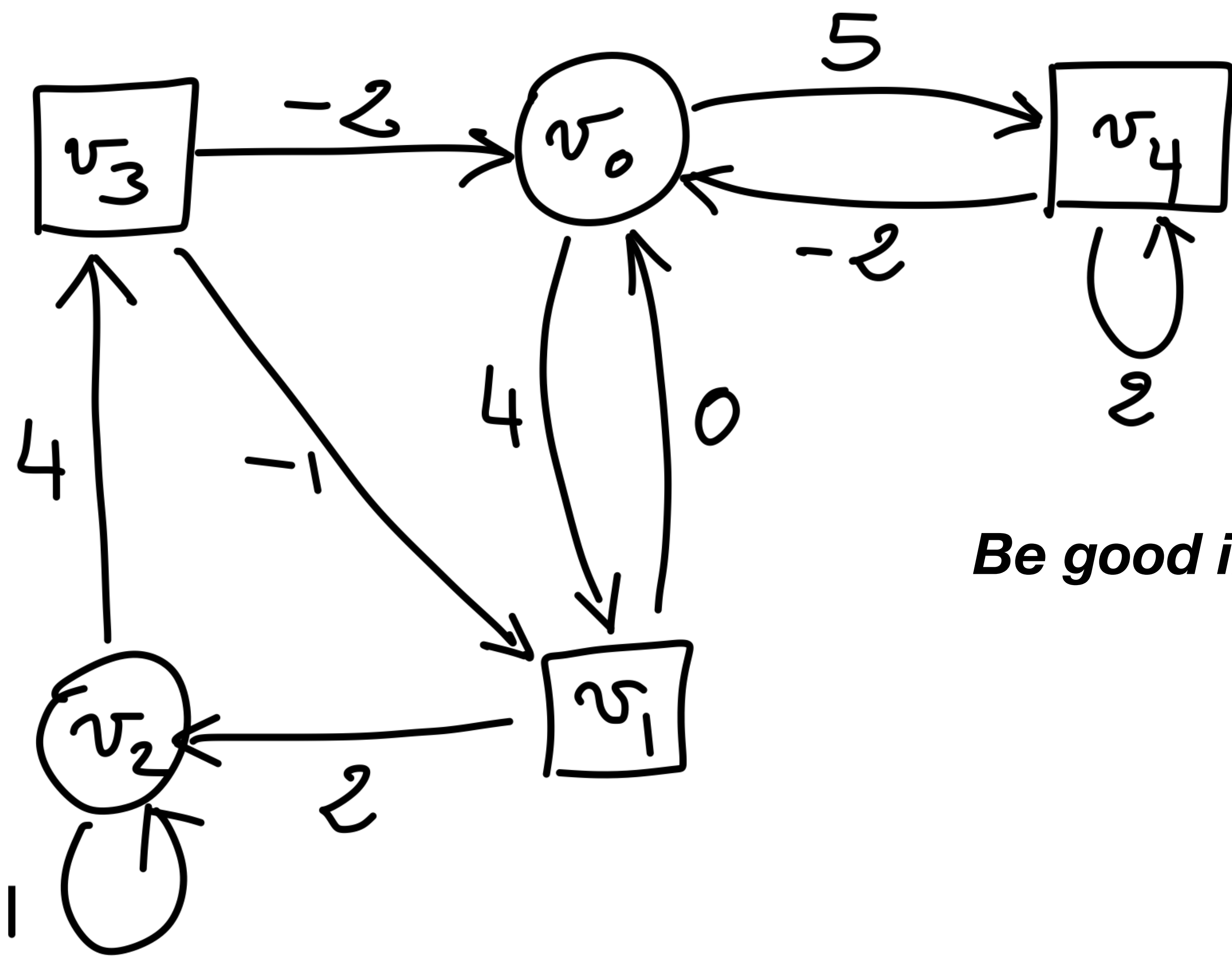
Quantitative games on graphs



Weighted graph: weights=rewards

$$v_0 \xrightarrow{4} v_1 \xrightarrow{0} v_0 \xrightarrow{5} v_4 \xrightarrow{-2} v_0 \ \cdots$$

Quantitative games on graphs

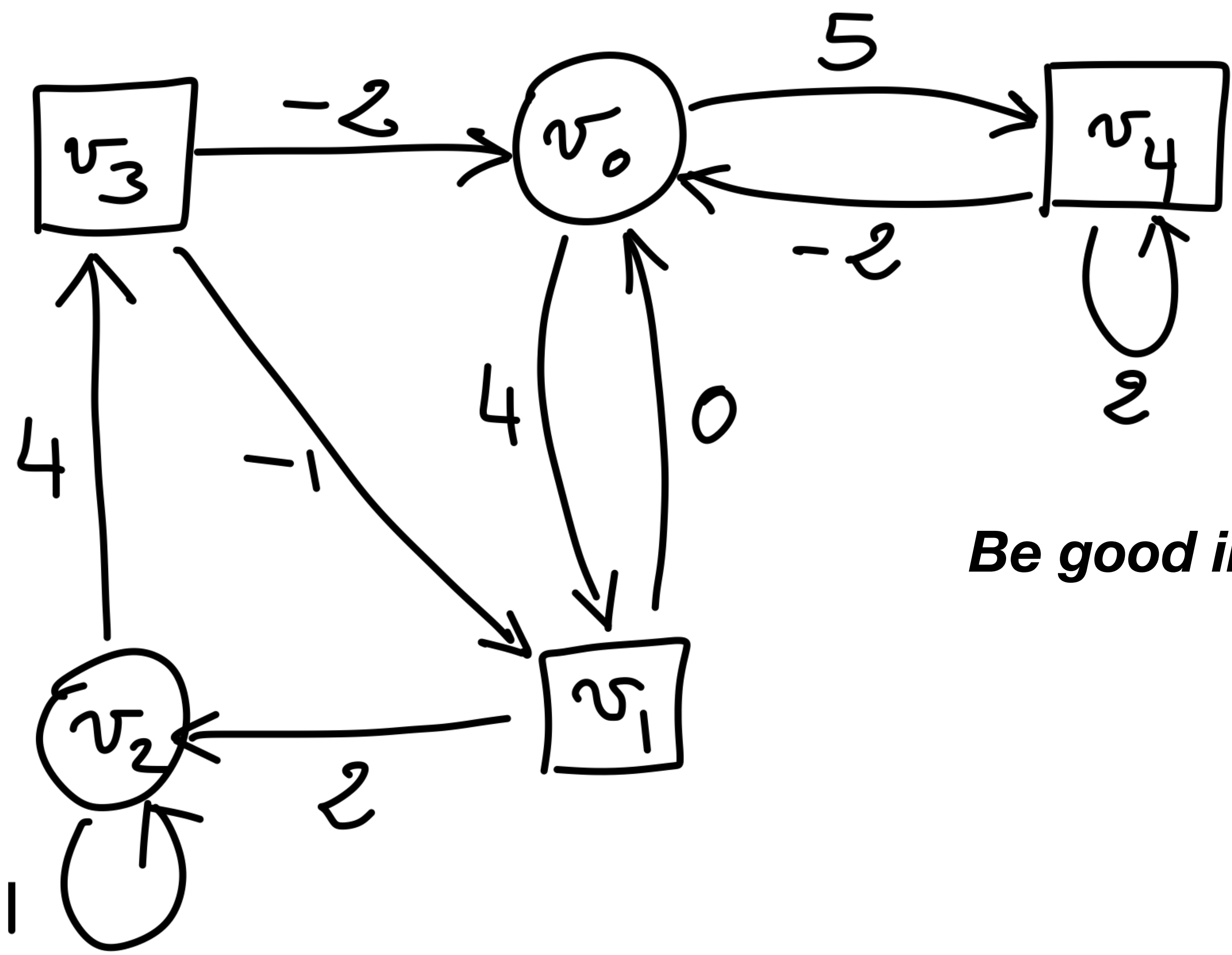


Weighted graph: weights=rewards

Be good in total: *total-payoff* $\sum_{i=0}^{\infty} r_i$

$$v_0 \xrightarrow{4} v_1 \xrightarrow{0} v_0 \xrightarrow{5} v_4 \xrightarrow{-2} v_0 \ \ldots$$

Quantitative games on graphs



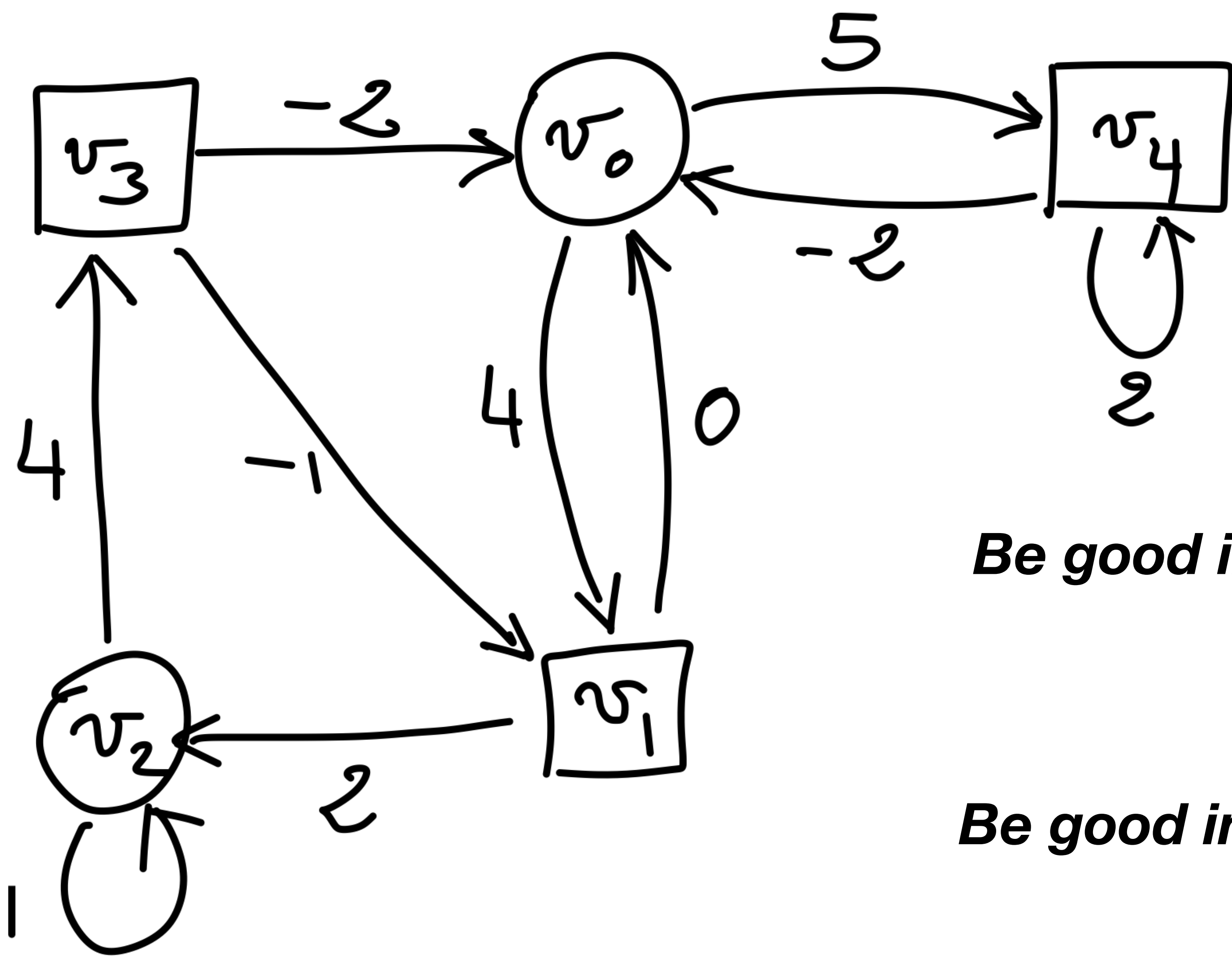
Weighted graph: weights=rewards

Be good in total: *total-payoff* $\sum_{i=0}^{\infty} r_i$

may not exist…

$$v_0 \xrightarrow{4} v_1 \xrightarrow{0} v_0 \xrightarrow{5} v_4 \xrightarrow{-2} v_0 \;\ldots$$

Quantitative games on graphs



Weighted graph: weights=rewards

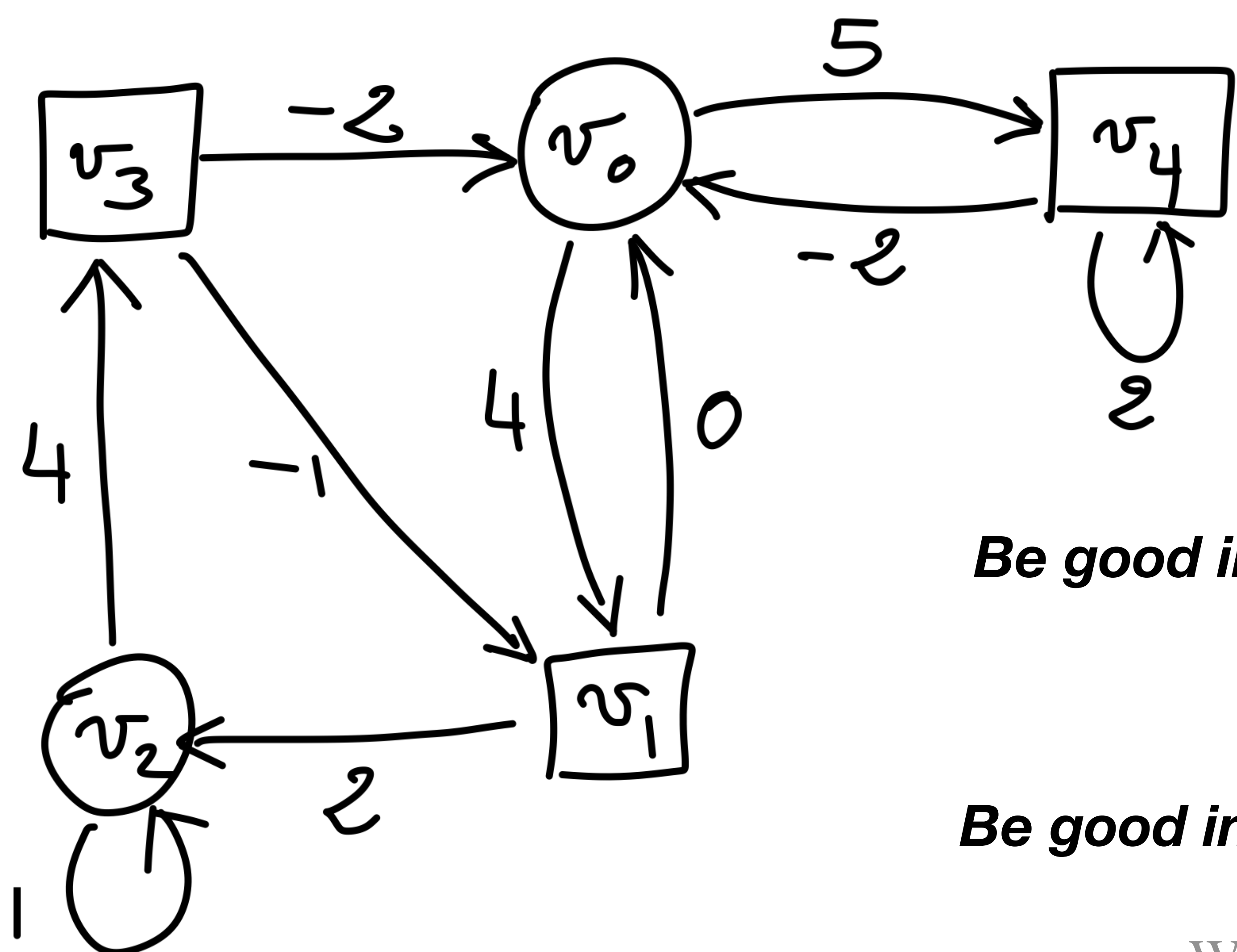
Be good in total: *total-payoff* $\sum_{i=0}^{\infty} r_i$

may not exist…

Be good in average: *mean-payoff* $\liminf_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} r_i$

$$v_0 \xrightarrow{4} v_1 \xrightarrow{0} v_0 \xrightarrow{5} v_4 \xrightarrow{-2} v_0 \ \cdots$$

Quantitative games on graphs



Weighted graph: weights=rewards

Be good in total: *total-payoff* $\sum_{i=0}^{\infty} r_i$

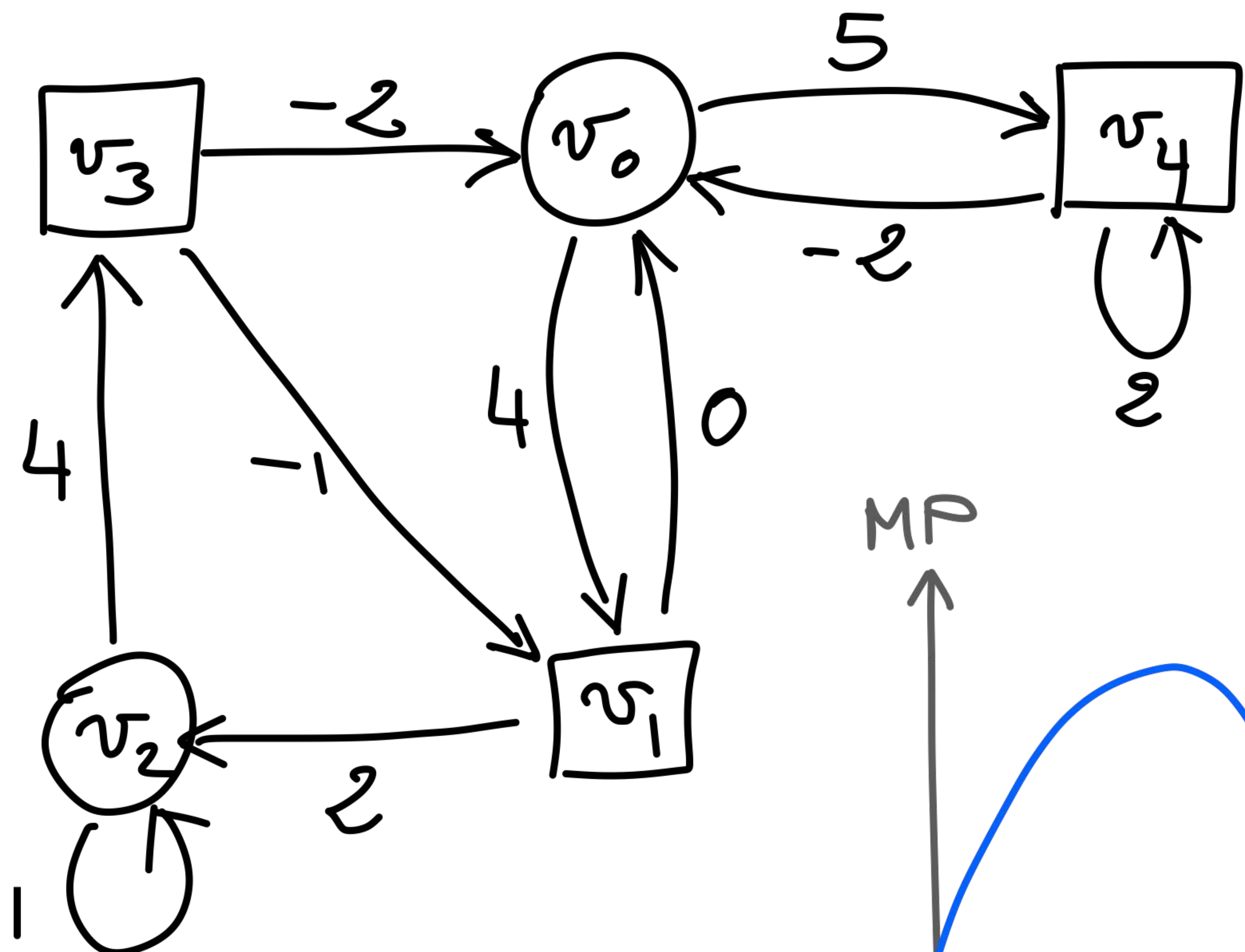
may not exist…

Be good in average: *mean-payoff* $\liminf_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} r_i$

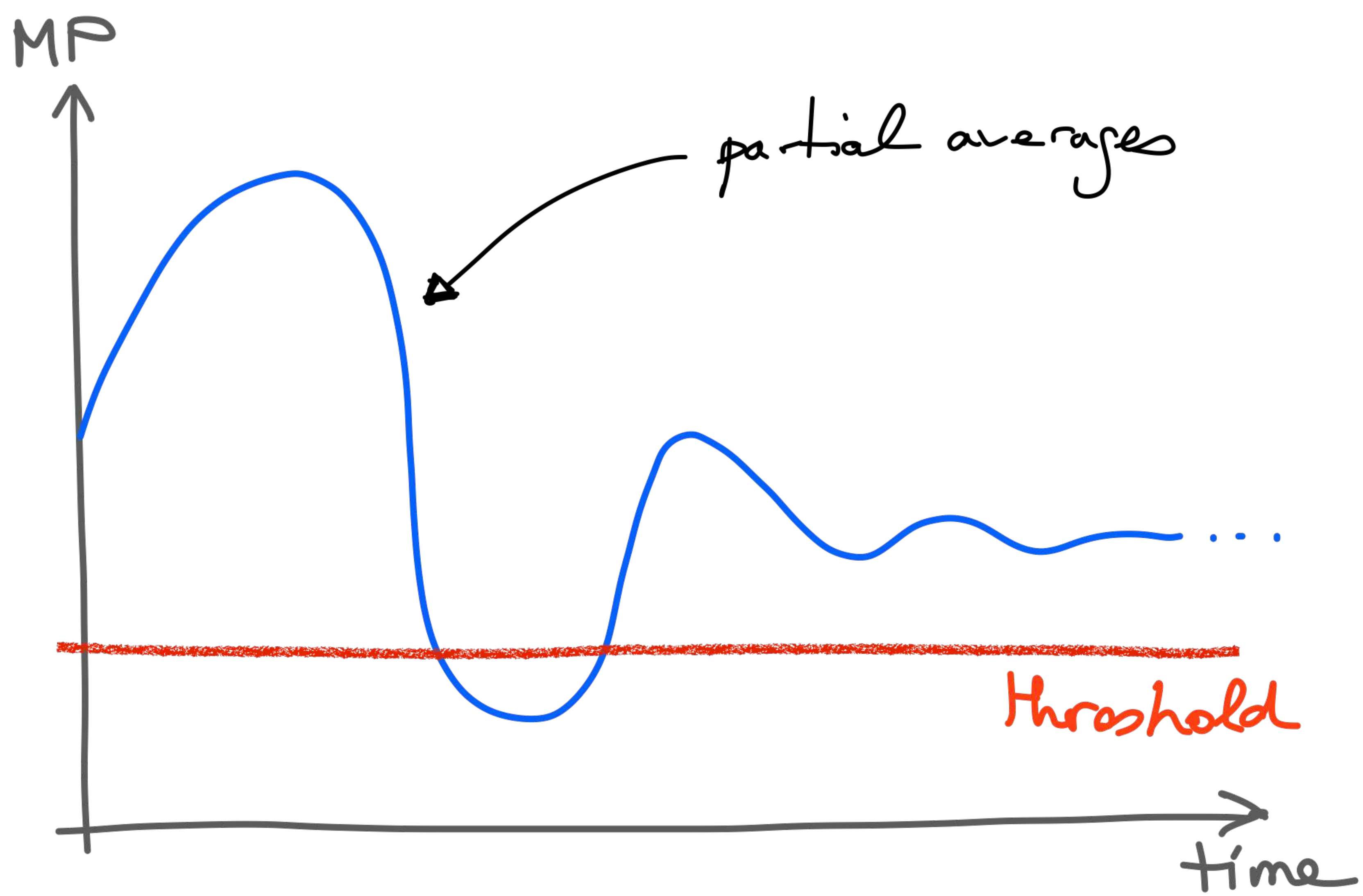
$\mathrm{Win}_{\mathrm{O}} = \{\pi \mid \mathrm{MP}(\pi) \geq c\}$ not ω-regular…

$$v_0 \xrightarrow{4} v_1 \xrightarrow{0} v_0 \xrightarrow{5} v_4 \xrightarrow{-2} v_0 \;\ldots$$

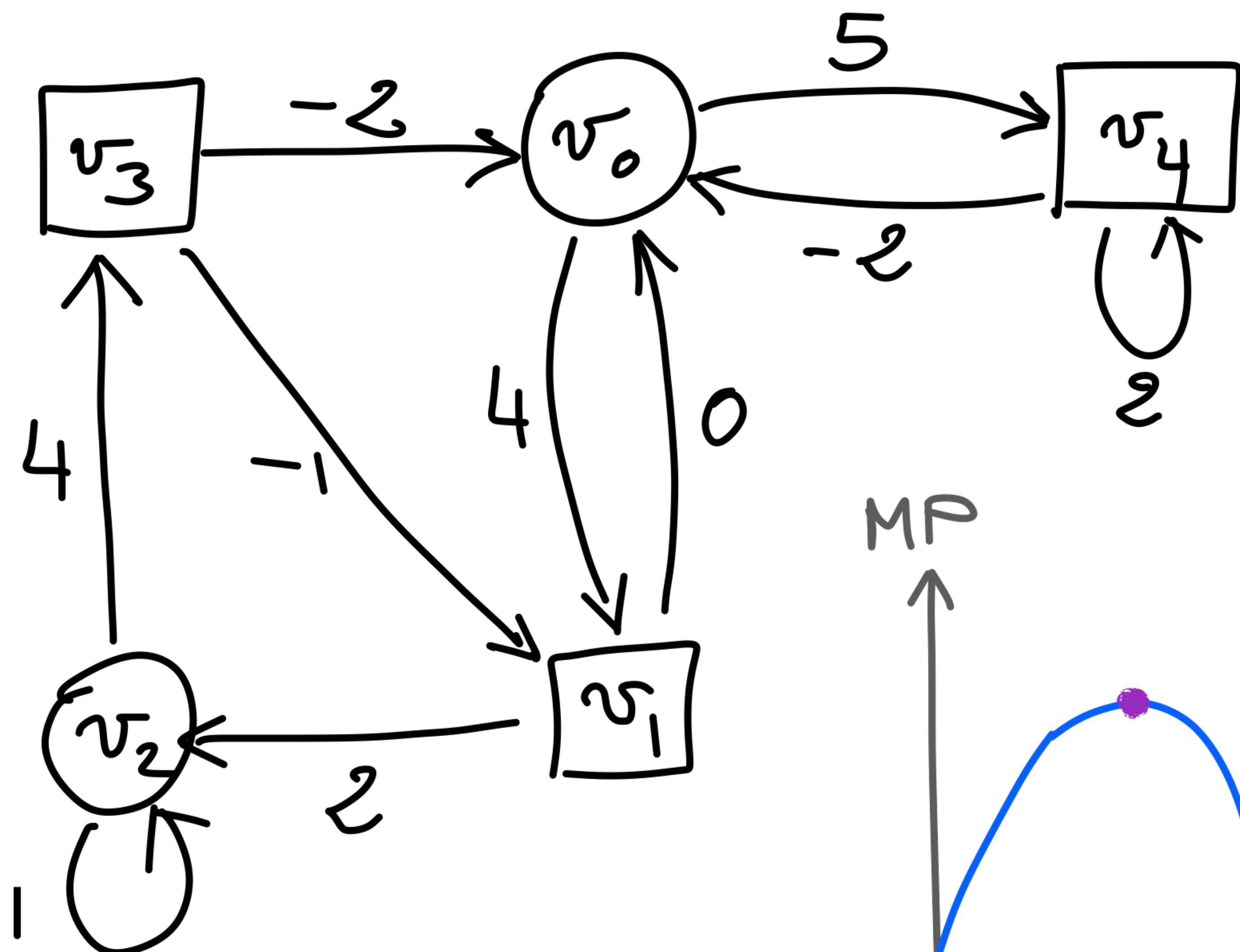
Mean-payoff games



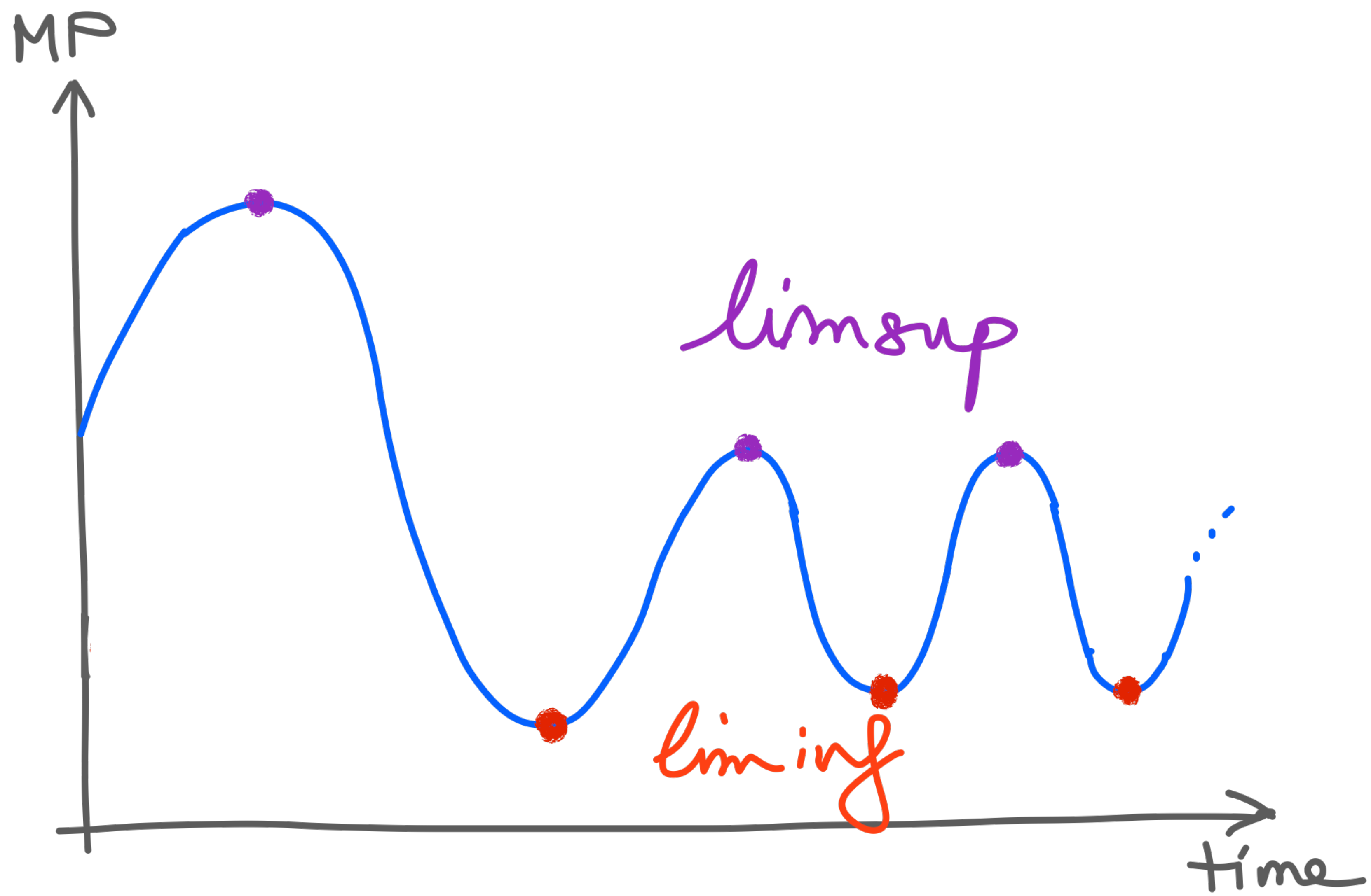
Be good in average: $\liminf_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} r_i$



Mean-payoff games



Be good in average: $\liminf_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n-1} r_i$



Mean-payoff games

Greatest mean-payoff that Player ◯ can guarantee:

$$\mathrm{Val}_{\mathrm{O}}(v) = \inf_{\sigma_\square} \sup_{\sigma_{\mathrm{O}}} \mathrm{MP}(\mathrm{play}(v, \sigma_{\mathrm{O}}, \sigma_\square))$$

Mean-payoff games

Greatest mean-payoff that Player ◯ can guarantee:

$$\mathrm{Val}_{\mathrm{O}}(v) = \inf_{\sigma_\square} \sup_{\sigma_{\mathrm{O}}} \mathrm{MP}(\mathrm{play}(v, \sigma_{\mathrm{O}}, \sigma_\square))$$

Smallest mean-payoff that Player □ can guarantee:

$$\mathrm{Val}_{\square}(v) = \sup_{\sigma_{\mathrm{O}}} \inf_{\sigma_\square} \mathrm{MP}(\mathrm{play}(v, \sigma_{\mathrm{O}}, \sigma_\square))$$

Mean-payoff games

Greatest mean-payoff that Player ◯ can guarantee:

$$\mathrm{Val}_{\mathrm{O}}(v) = \inf_{\sigma_\square} \sup_{\sigma_{\mathrm{O}}} \mathrm{MP}(\mathrm{play}(v, \sigma_{\mathrm{O}}, \sigma_\square))$$

Smallest mean-payoff that Player □ can guarantee:

$$\mathrm{Val}_\square(v) = \sup_{\sigma_{\mathrm{O}}} \inf_{\sigma_\square} \mathrm{MP}(\mathrm{play}(v, \sigma_{\mathrm{O}}, \sigma_\square))$$

Theorem (Ehrenfeucht-Mycielski 1979, Zwick-Paterson 1997)

1. Mean-payoff games are determined: $\forall v \quad \mathrm{Val}_{\mathrm{O}}(v) = \mathrm{Val}_\square(v) =: \mathrm{Val}(v)$

2. Both players have *optimal* memoryless strategies:

$$\exists \sigma^*_{\mathrm{O}}\ \forall v \quad \inf_{\sigma_\square} \mathrm{MP}(\mathrm{play}(v, \sigma^*_{\mathrm{O}}, \sigma_\square)) = \mathrm{Val}(v)$$

$$\exists \sigma^*_\square\ \forall v \quad \sup_{\sigma_{\mathrm{O}}} \mathrm{MP}(\mathrm{play}(v, \sigma_{\mathrm{O}}, \sigma^*_\square)) = \mathrm{Val}(v)$$

3. The winner, with respect to a fixed threshold, can be decided in NP ∩ co-NP.

1. Mean-payoff games are determined

$$\mathrm{Val}_{\square}(v) = \sup_{\sigma_{\mathrm{O}}} \inf_{\sigma_{\square}} \mathrm{MP}(\mathrm{play}(v, \sigma_{\mathrm{O}}, \sigma_{\square})) \qquad \inf_{\sigma_{\square}} \sup_{\sigma_{\mathrm{O}}} \mathrm{MP}(\mathrm{play}(v, \sigma_{\mathrm{O}}, \sigma_{\square})) = \mathrm{Val}_{\mathrm{O}}(v)$$

1. Mean-payoff games are determined

$$\mathrm{Val}_{\square}(v) = \sup_{\sigma_{\mathrm{O}}} \inf_{\sigma_{\square}} \mathrm{MP}(\mathrm{play}(v, \sigma_{\mathrm{O}}, \sigma_{\square})) \leq \inf_{\sigma_{\square}} \sup_{\sigma_{\mathrm{O}}} \mathrm{MP}(\mathrm{play}(v, \sigma_{\mathrm{O}}, \sigma_{\square})) = \mathrm{Val}_{\mathrm{O}}(v)$$

1. Mean-payoff games are determined

$$\mathrm{Val}_{\square}(v) = \sup_{\sigma_{\mathrm{O}}} \inf_{\sigma_{\square}} \mathrm{MP}(\mathrm{play}(v, \sigma_{\mathrm{O}}, \sigma_{\square})) \leq \inf_{\sigma_{\square}} \sup_{\sigma_{\mathrm{O}}} \mathrm{MP}(\mathrm{play}(v, \sigma_{\mathrm{O}}, \sigma_{\square})) = \mathrm{Val}_{\mathrm{O}}(v)$$

Determinacy (inequality ≥) can be restated as:

$\forall \alpha$ **either** Player ◯ has a strategy to force a MP $\geq \alpha$

or Player □ has a strategy to force a MP $< \alpha$

First-cycle game

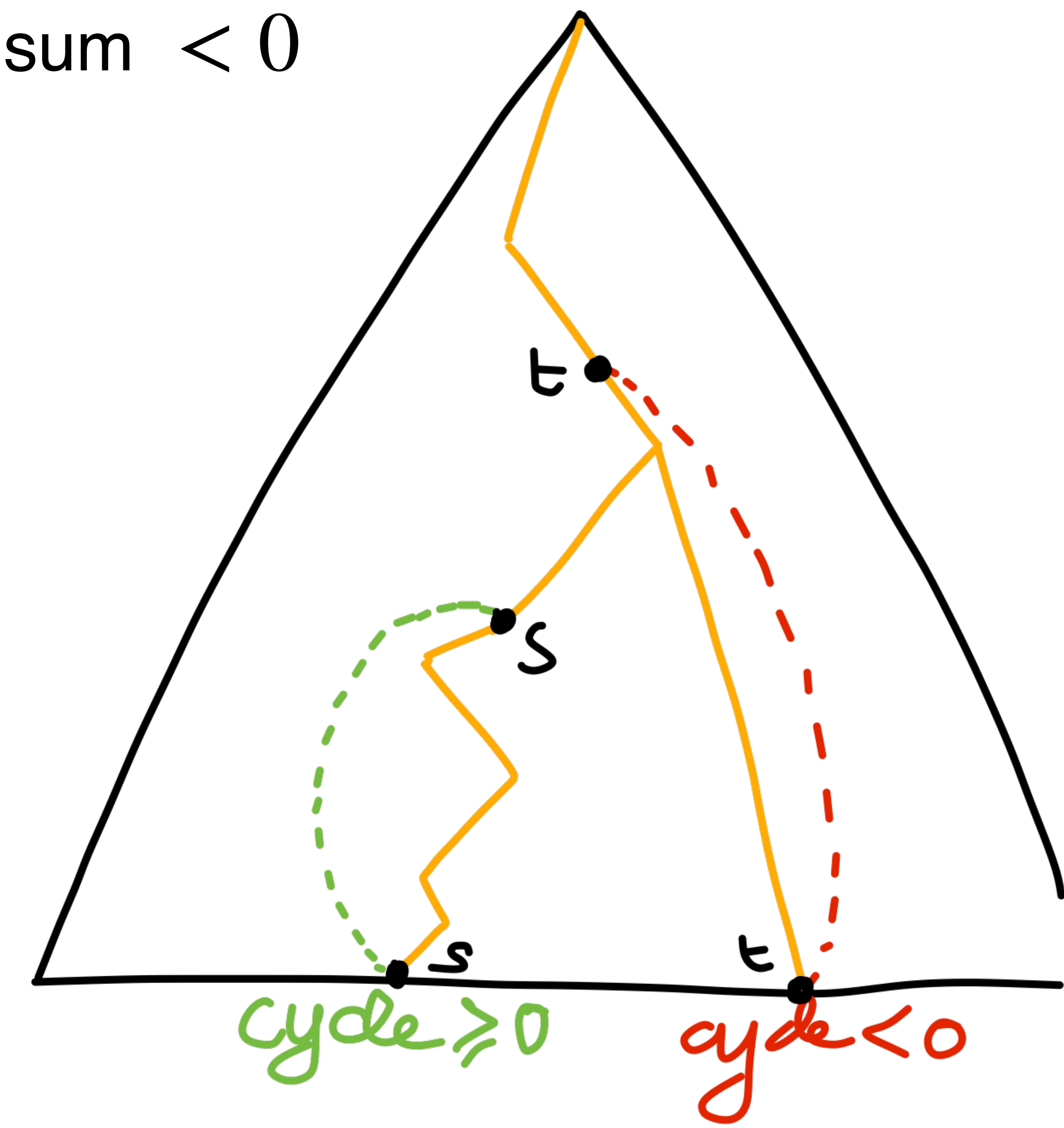
Unfold the weighted graph up to a first repetition of vertex:

- a leaf is winning for Player ◯ if the cycle has a sum ≥ 0
- a leaf is winning for Player □ if the cycle has a sum < 0

First-cycle game

Unfold the weighted graph up to a first repetition of vertex:

- a leaf is winning for Player ◯ if the cycle has a sum ≥ 0
- a leaf is winning for Player □ if the cycle has a sum < 0

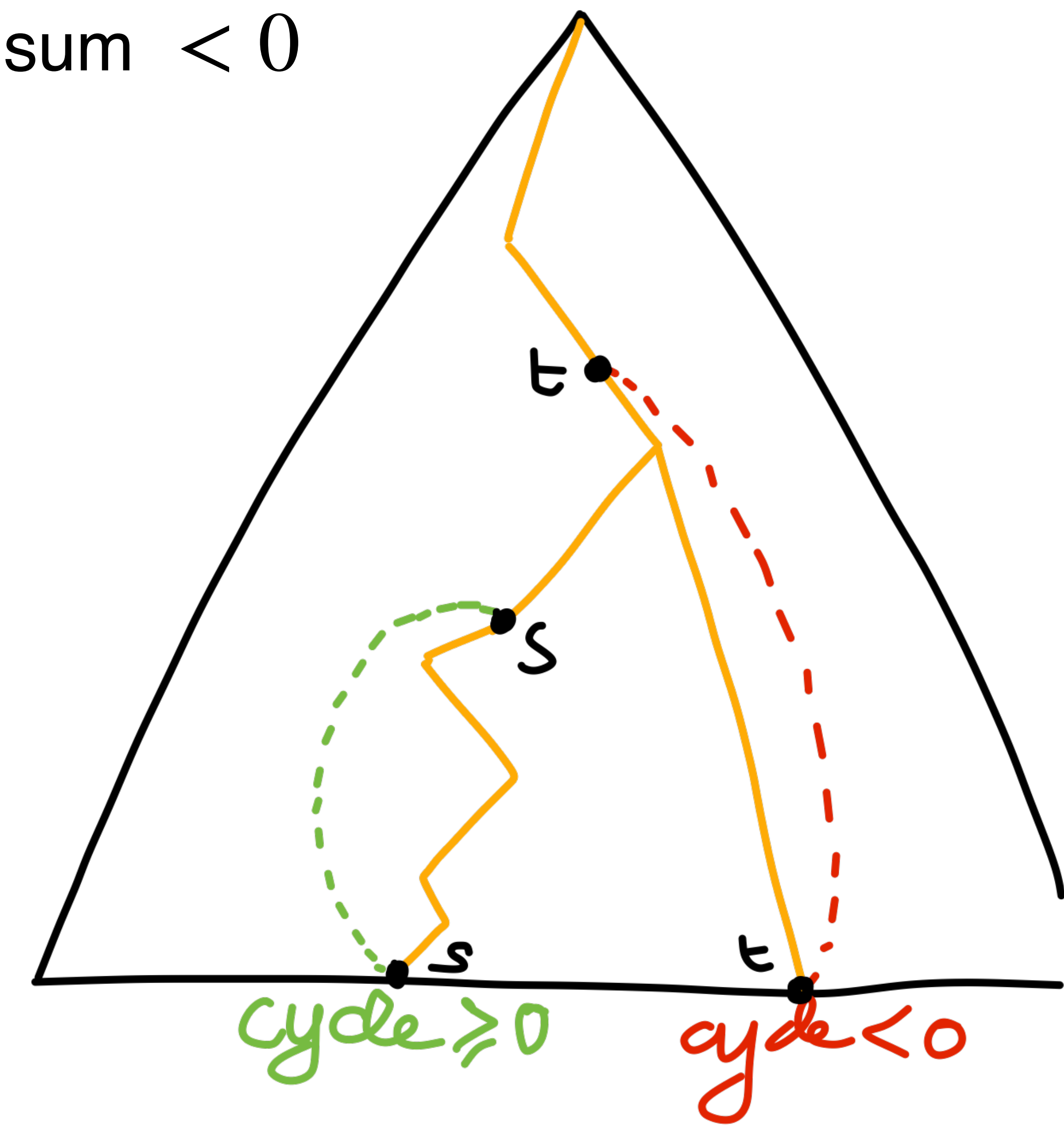


First-cycle game

Unfold the weighted graph up to a first repetition of vertex:

- a leaf is winning for Player ◯ if the cycle has a sum ≥ 0
- a leaf is winning for Player □ if the cycle has a sum < 0

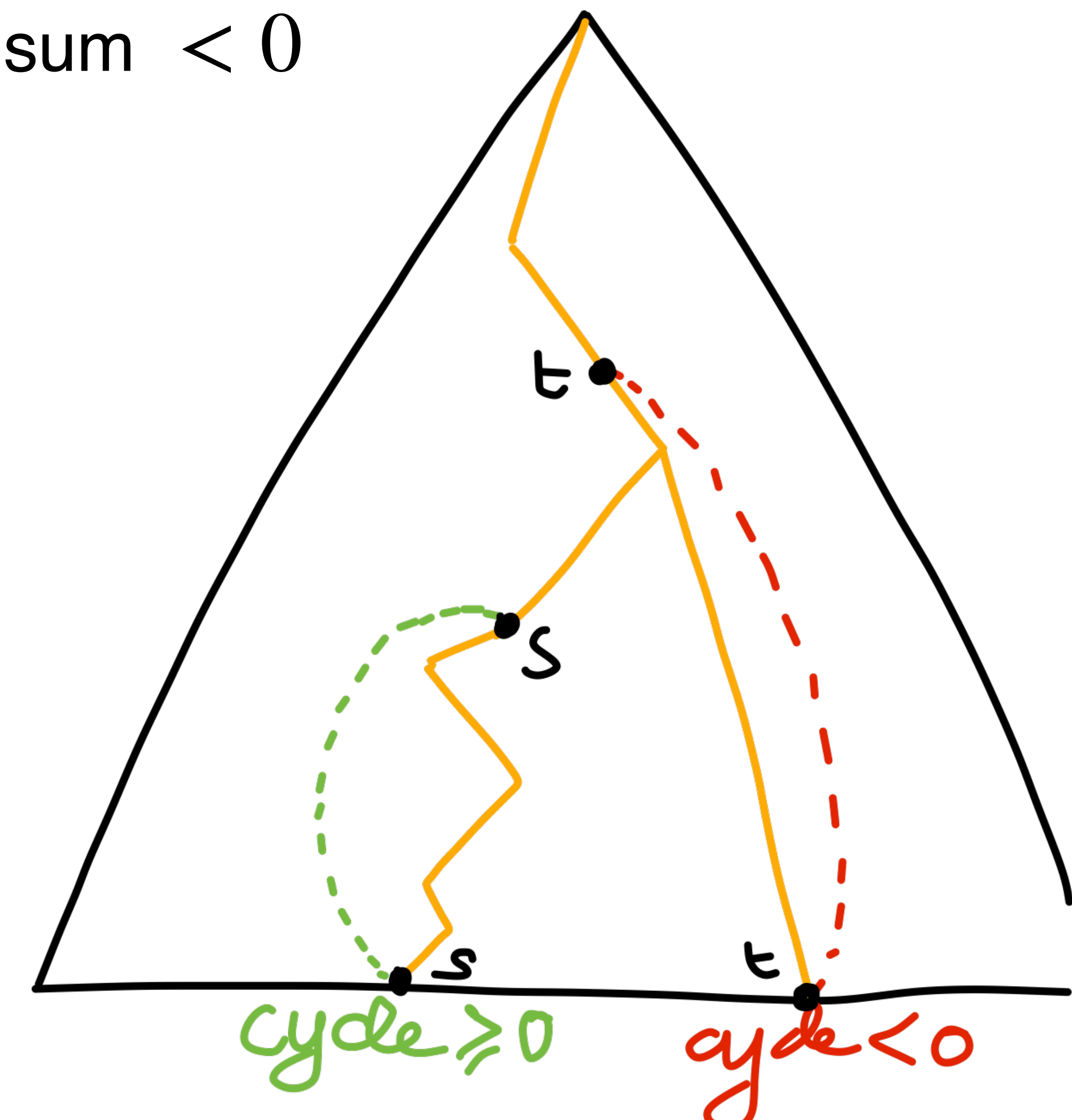
By Zermelo's theorem:
either Player ◯ can force non-negative cycles
or Player □ can force negative cycles



First-cycle game

Unfold the weighted graph up to a first repetition of vertex:

- a leaf is winning for Player ◯ if the cycle has a sum ≥ 0
- a leaf is winning for Player □ if the cycle has a sum < 0



By Zermelo's theorem:
either Player ◯ can force non-negative cycles

or Player □ can force negative cycles

transfer of strategies

either Player ◯ has a <u>memoryless</u> strategy to force a MP ≥ 0

or Player □ has a <u>memoryless</u> strategy to force a MP < 0

Mean-payoff games

Theorem (Ehrenfeucht-Mycielski 1979, Zwick-Paterson 1997)

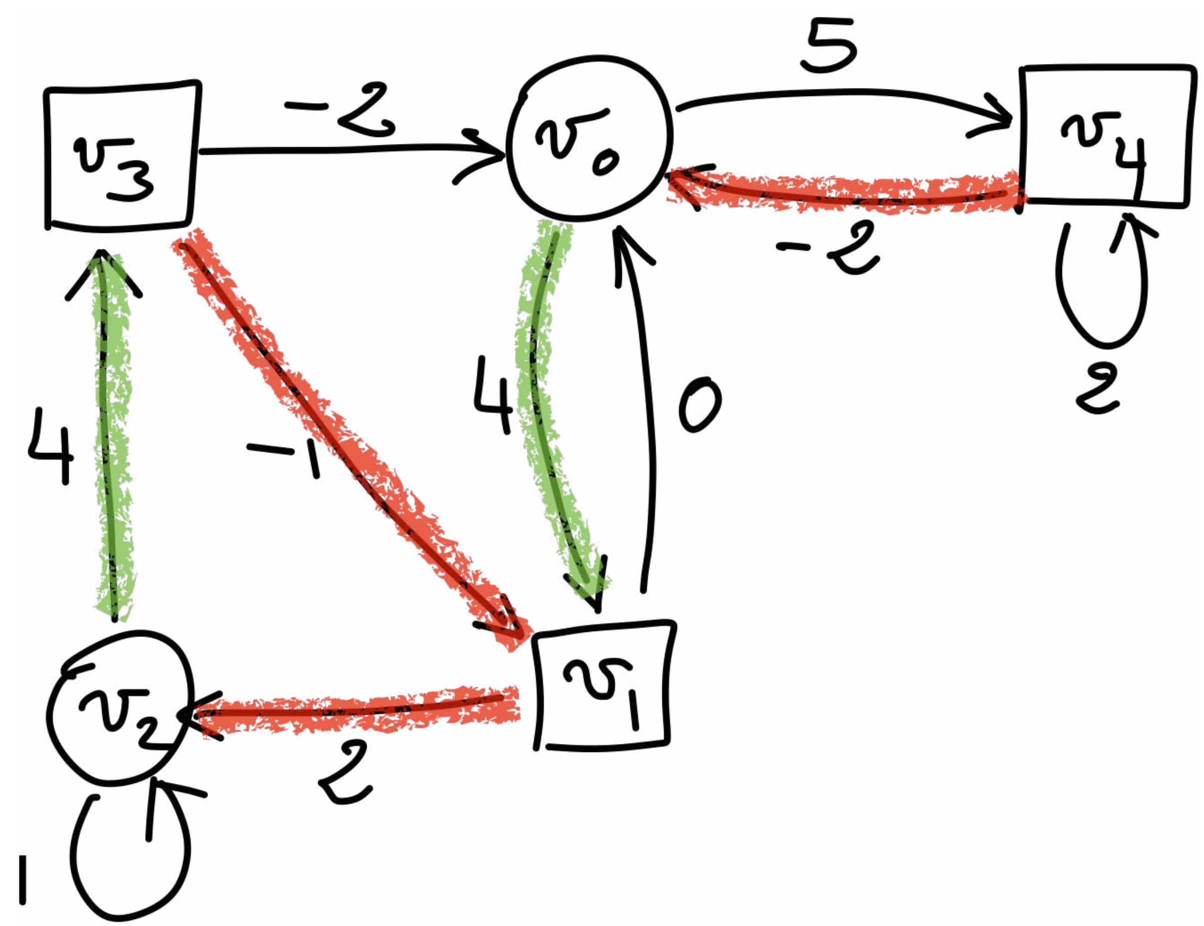
1. Mean-payoff games are determined: $\forall v \quad \mathrm{Val}_{\bigcirc}(v) = \mathrm{Val}_{\square}(v) =: \mathrm{Val}(v)$

2. Both players have *optimal* memoryless strategies:

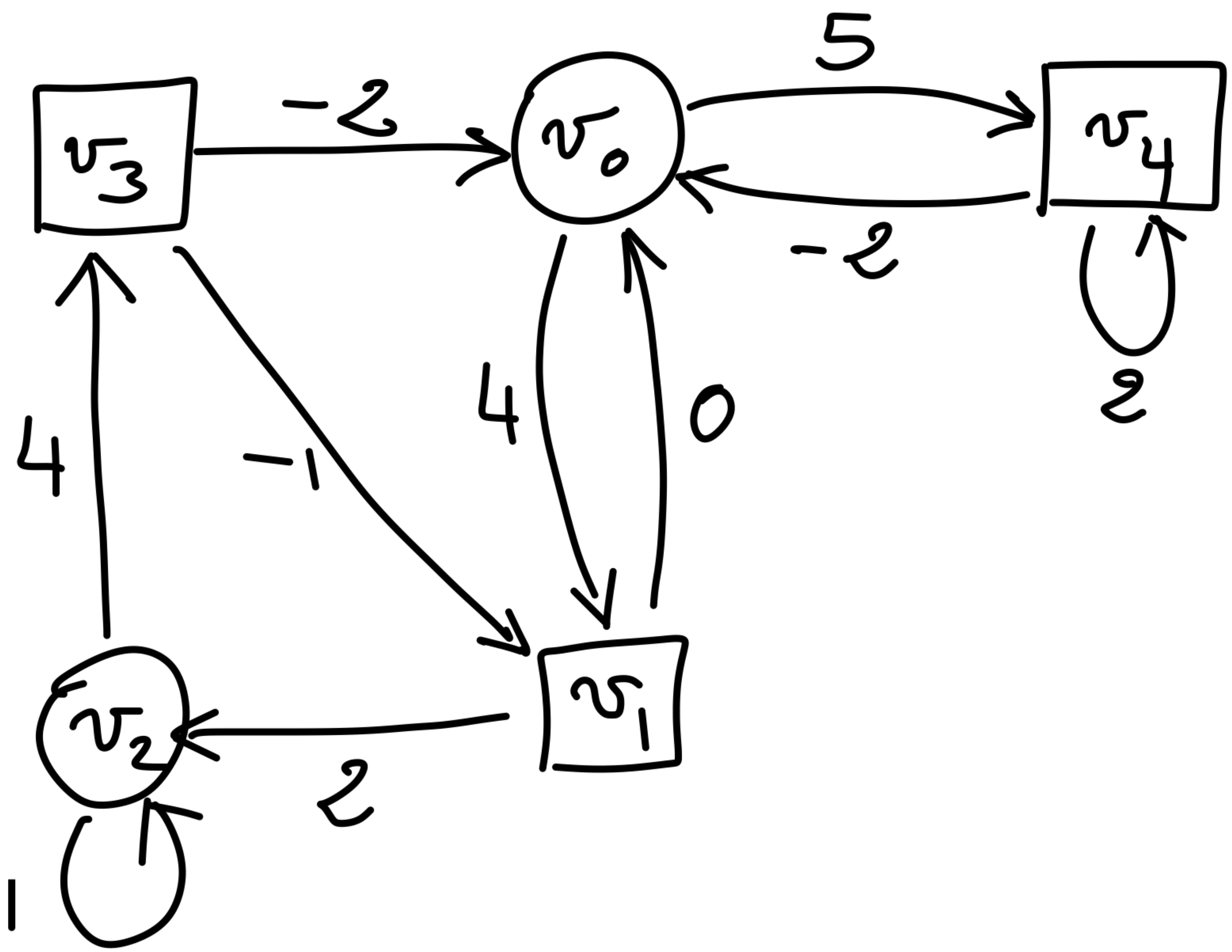
$$\exists \sigma_{\bigcirc}^* \, \forall v \quad \inf_{\sigma_{\square}} \mathrm{MP}(\mathrm{play}(v, \sigma_{\bigcirc}^*, \sigma_{\square})) = \mathrm{Val}(v)$$

$$\exists \sigma_{\square}^* \, \forall v \quad \sup_{\sigma_{\bigcirc}} \mathrm{MP}(\mathrm{play}(v, \sigma_{\bigcirc}, \sigma_{\square}^*)) = \mathrm{Val}(v)$$

3. The winner, with respect to a fixed threshold, can be decided in NP ∩ co-NP.



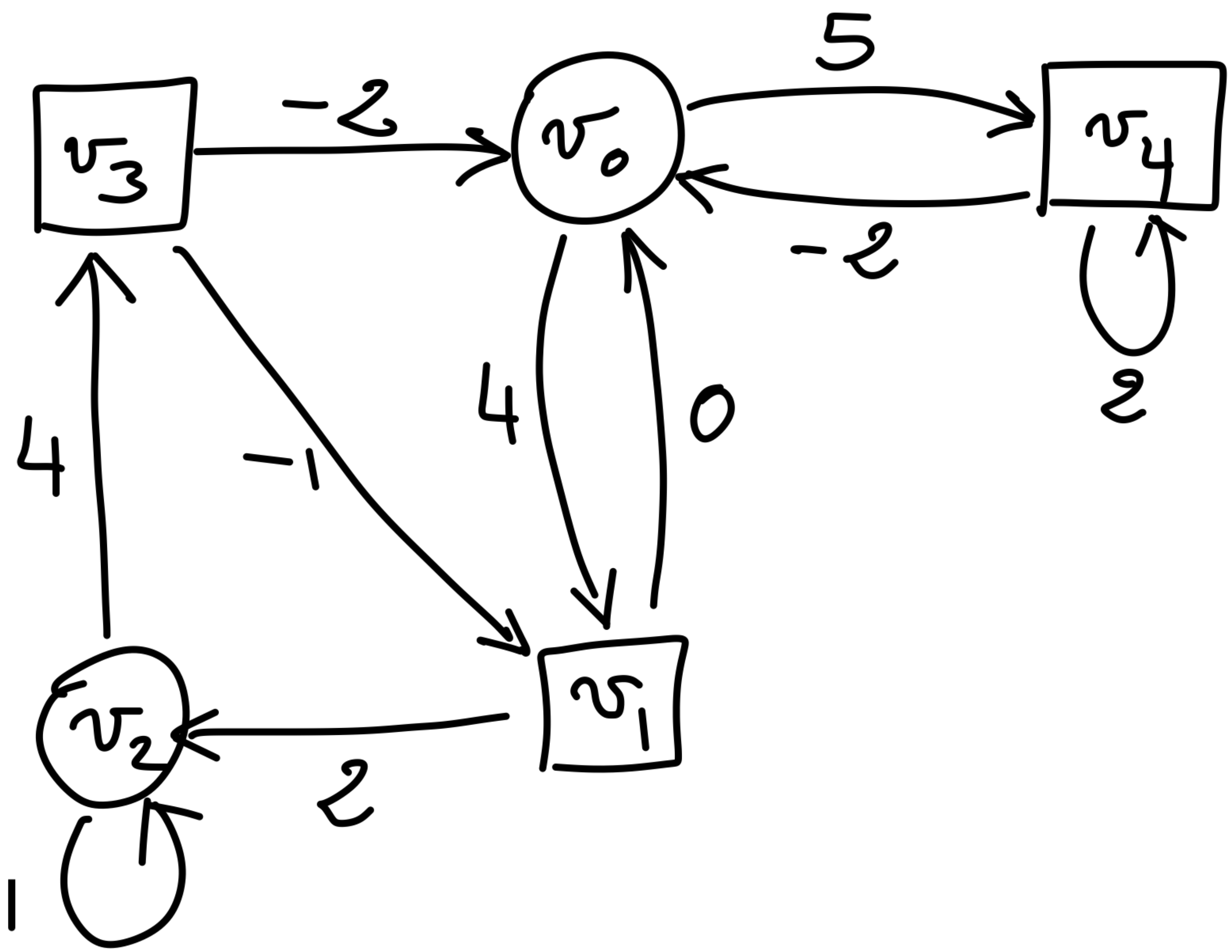
Discounted-payoff games



Be good soon enough: $(1 - \lambda) \sum_{i=0}^{\infty} \lambda^i r_i$

$0 < \lambda < 1$

Discounted-payoff games



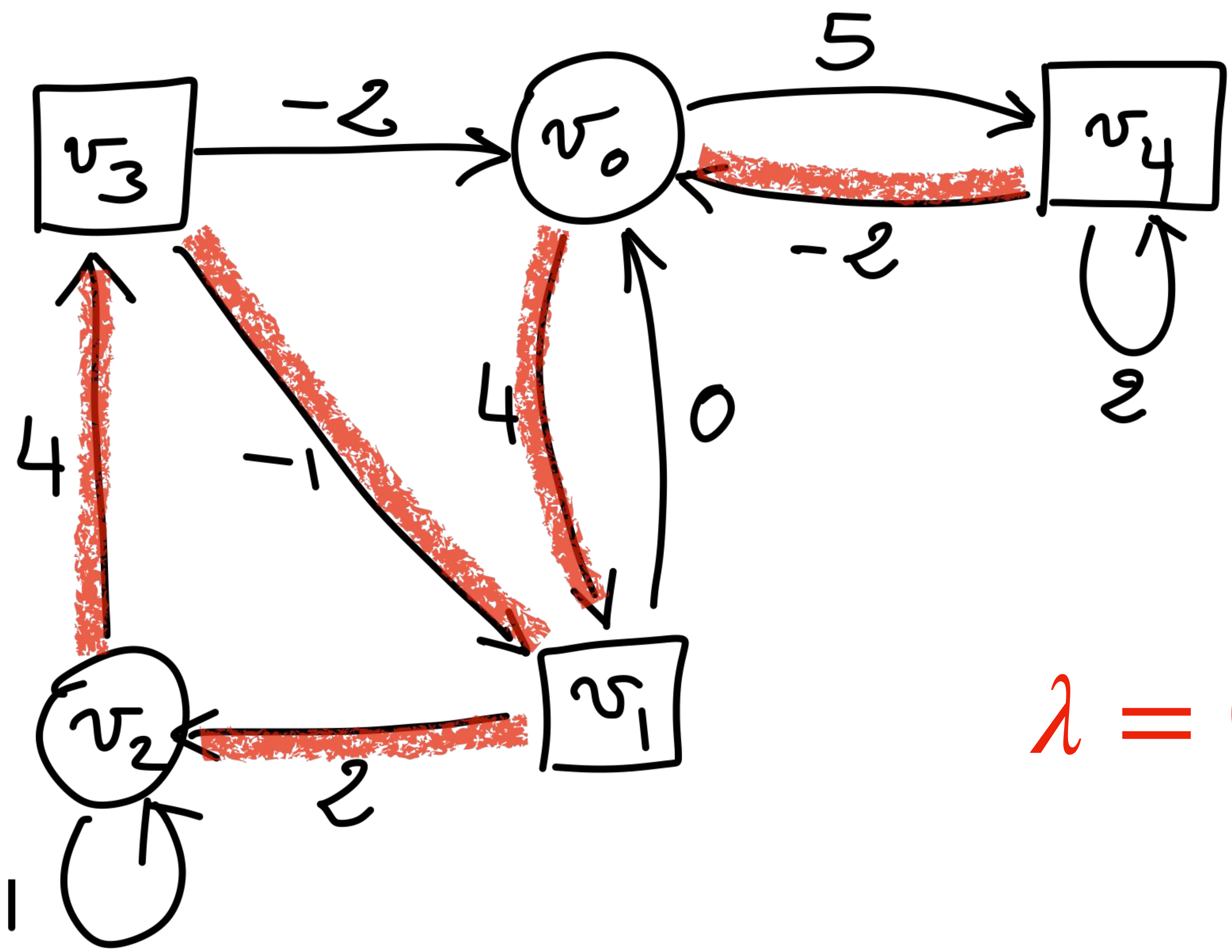
Be good soon enough: $(1-\lambda)\sum_{i=0}^{\infty} \lambda^i r_i$

$0 < \lambda < 1$

When $\lambda \to 0$ only prefixes matter

When $\lambda \to 1$ DP looks a lot like MP

Discounted-payoff games



Be good soon enough: $(1-\lambda)\sum_{i=0}^{\infty}\lambda^i r_i$

$$0 < \lambda < 1$$

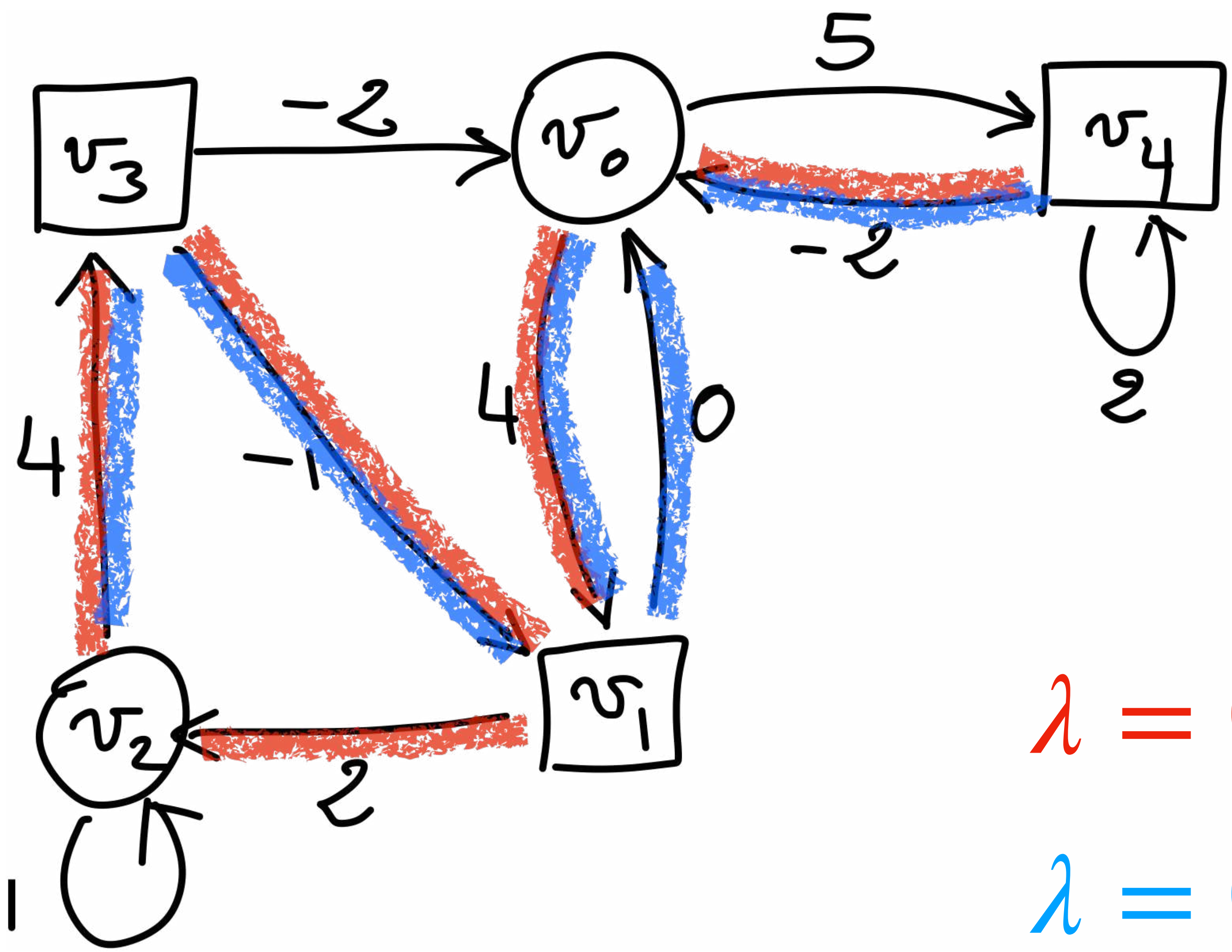
When $\lambda \to 0$ only prefixes matter

When $\lambda \to 1$ DP looks a lot like MP

$$\lambda = 0.9$$

same strategy as for MP

Discounted-payoff games



Be good soon enough: $(1-\lambda)\sum_{i=0}^{\infty}\lambda^i r_i$

$0 < \lambda < 1$

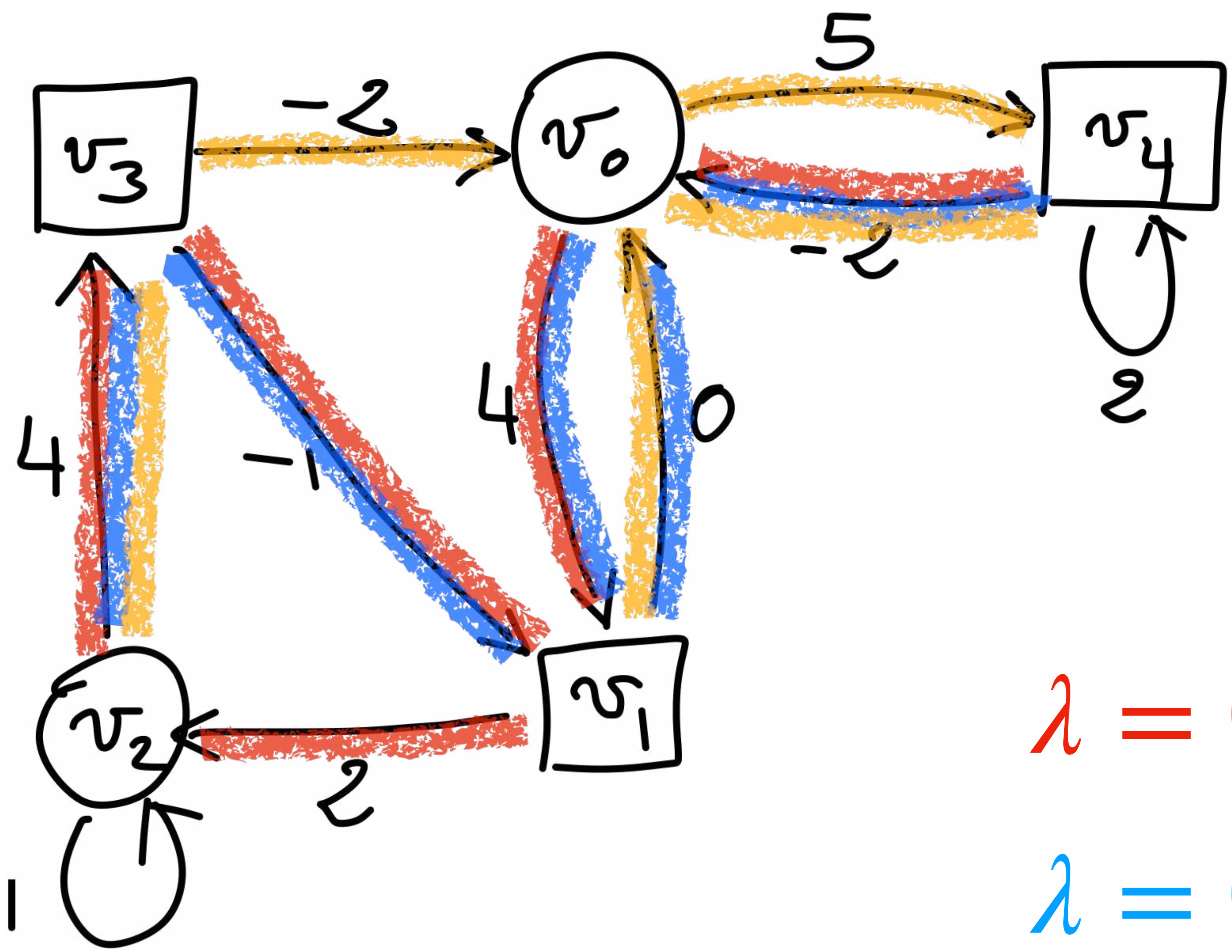
When $\lambda \to 0$ only prefixes matter

When $\lambda \to 1$ DP looks a lot like MP

$\lambda = 0.9$ *same strategy as for MP*

$\lambda = 0.5$

Discounted-payoff games



Be good soon enough: $(1-\lambda)\sum_{i=0}^{\infty}\lambda^i r_i$

$0 < \lambda < 1$

When $\lambda \to 0$ only prefixes matter

When $\lambda \to 1$ DP looks a lot like MP

$\lambda = 0.9$ *same strategy as for MP*

$\lambda = 0.5$

$\lambda = 0.1$

Memoryless determinacy

Theorem (Zwick-Paterson 1997)

1. Discounted-payoff games are determined: $\forall v \quad \mathrm{Val}_{\bigcirc}(v) = \mathrm{Val}_{\Box}(v) =: \mathrm{Val}(v)$
2. Both players have *optimal* memoryless strategies:
$$\exists \sigma^*_{\bigcirc}\ \forall v \quad \inf_{\sigma_{\Box}} \mathrm{DP}_\lambda(\mathrm{play}(v, \sigma^*_{\bigcirc}, \sigma_{\Box})) = \mathrm{Val}(v)$$
$$\exists \sigma^*_{\Box}\ \forall v \quad \sup_{\sigma_{\bigcirc}} \mathrm{DP}_\lambda(\mathrm{play}(v, \sigma_{\bigcirc}, \sigma^*_{\Box})) = \mathrm{Val}(v)$$
3. The winner, with respect to a fixed threshold, can be decided in NP ∩ co-NP.

Proof: finite horizon

$$F(x)_v = \begin{cases} \max_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\square} \end{cases}$$

Proof: finite horizon

$$F(x)_v = \begin{cases} \max_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\square} \end{cases}$$

$$F : \mathbf{R}^V \to \mathbf{R}^V$$

contraction mapping

Proof: finite horizon

$$F(x)_v = \begin{cases} \max_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\square} \end{cases}$$

$$F : \mathbf{R}^V \to \mathbf{R}^V$$

contraction mapping

By Banach theorem, unique fixed point

$$F(x^*) = x^*$$

Proof: finite horizon

$$F(x)_v = \begin{cases} \max_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\Box} \end{cases}$$

$$F : \mathbf{R}^V \to \mathbf{R}^V$$

contraction mapping

By Banach theorem, unique fixed point

$$F(x^*) = x^*$$

$$x^* = \lim_{n\to\infty} F^n(\mathbf{0})$$

Proof: finite horizon

$$F(x)_v = \begin{cases} \max_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\Box} \end{cases}$$

$$F : \mathbf{R}^V \to \mathbf{R}^V$$

contraction mapping

By Banach theorem, unique fixed point

$$F(x^*) = x^*$$

$$x^* = \lim_{n\to\infty} F^n(\mathbf{0})$$

following strategies dictated by $F(x^*) = x^*$

$$\mathrm{Val}_{\bigcirc}(v) \le x^*_v \le \mathrm{Val}_{\Box}(v)$$

Proof: finite horizon

$$F(x)_v = \begin{cases} \max_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\Box} \end{cases}$$

$$F : \mathbf{R}^V \to \mathbf{R}^V$$

contraction mapping

By Banach theorem, unique fixed point

$$F(x^*) = x^*$$

$$x^* = \lim_{n\to\infty} F^n(\mathbf{0})$$

following strategies dictated by $F(x^*) = x^*$

$$\mathrm{Val}_{\bigcirc}(v) \leq x^*_v \leq \mathrm{Val}_{\Box}(v)$$

always true

$$\mathrm{Val}_{\Box}(v) \leq \mathrm{Val}_{\bigcirc}(v)$$

Proof: finite horizon

$$F(x)_v = \begin{cases} \max_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\Box} \end{cases}$$

$$F: \mathbf{R}^V \to \mathbf{R}^V$$

contraction mapping

By Banach theorem, unique fixed point

$$F(x^*) = x^*$$

$$x^* = \lim_{n\to\infty} F^n(\mathbf{0})$$

following strategies dictated by $F(x^*) = x^*$

$$\mathrm{Val}_{\bigcirc}(v) \le x^*_v \le \mathrm{Val}_{\Box}(v)$$

always true

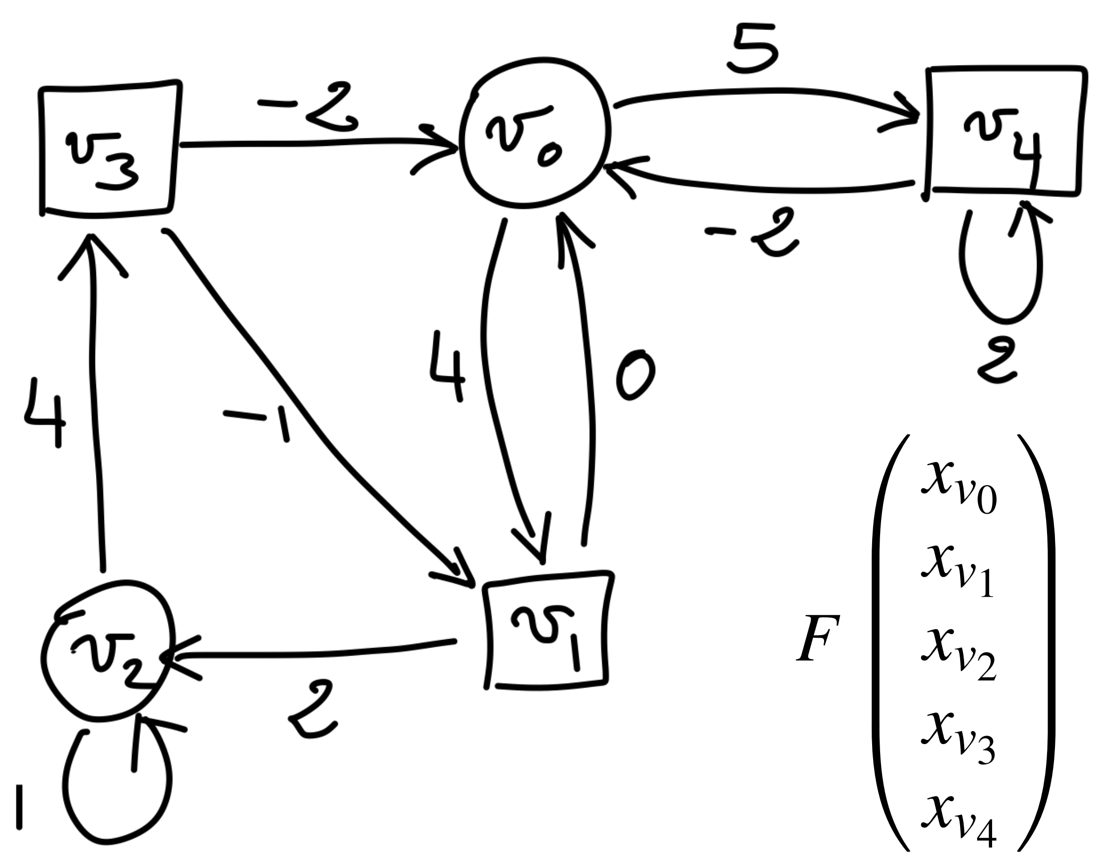
$$\mathrm{Val}_{\Box}(v) \le \mathrm{Val}_{\bigcirc}(v)$$

$$x^* = \mathrm{Val}$$

Memoryless determinacy

Theorem (Zwick-Paterson 1997)

1. Discounted-payoff games are determined: $\forall v \quad \mathrm{Val}_{\bigcirc}(v) = \mathrm{Val}_{\Box}(v) =: \mathrm{Val}(v)$
2. Both players have *optimal* memoryless strategies:
$$\exists \sigma^*_{\bigcirc}\ \forall v \quad \inf_{\sigma_\Box} \mathrm{DP}_\lambda(\mathrm{play}(v, \sigma^*_{\bigcirc}, \sigma_\Box)) = \mathrm{Val}(v)$$
$$\exists \sigma^*_{\Box}\ \forall v \quad \sup_{\sigma_{\bigcirc}} \mathrm{DP}_\lambda(\mathrm{play}(v, \sigma_{\bigcirc}, \sigma^*_\Box)) = \mathrm{Val}(v)$$
3. The winner, with respect to a fixed threshold, can be decided in NP ∩ co-NP.



$$F\begin{pmatrix} x_{v_0} \\ x_{v_1} \\ x_{v_2} \\ x_{v_3} \\ x_{v_4} \end{pmatrix} = \begin{pmatrix} \max\left(4(1-\lambda)+\lambda x_{v_1}, (1-\lambda)5+\lambda x_{v_4}\right) \\ \min\left(\lambda x_{v_0}, 2(1-\lambda)+\lambda x_{v_2}\right) \\ \max\left((1-\lambda)+\lambda x_{v_2}, 4(1-\lambda)+\lambda x_{v_3}\right) \\ \min\left(-2(1-\lambda)+\lambda x_{v_0}, -(1-\lambda)+\lambda x_{v_1}\right) \\ \min\left(-2(1-\lambda)+\lambda x_{v_0}, 2(1-\lambda)+\lambda x_{v_4}\right) \end{pmatrix}$$

How to compute optimal values?

$$F(x)_v = \begin{cases} \max_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\square} \end{cases}$$

$$x^* = \lim_{n\to\infty} F^n(\mathbf{0})$$

How to compute optimal values?

$$F(x)_v = \begin{cases} \max_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\Box} \end{cases}$$

$$x^* = \lim_{n\to\infty} F^n(\mathbf{0})$$

When to stop the computation, supposing every weight is rational?

How to compute optimal values?

$$F(x)_v = \begin{cases} \max_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\Box} \end{cases}$$

$$x^* = \lim_{n\to\infty} F^n(\mathbf{0})$$

When to stop the computation, supposing every weight is rational?

1. If $\lambda = a/b$ is rational, then x^*_v is rational too, of denominator $D = b^{O(|V|^2)}$

How to compute optimal values?

$$F(x)_v = \begin{cases} \max_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\square} \end{cases}$$

$$x^* = \lim_{n\to\infty} F^n(\mathbf{0})$$

When to stop the computation, supposing every weight is rational?

1. If $\lambda = a/b$ is rational, then x_v^* is rational too, of denominator $D = b^{O(|V|^2)}$
2. If K is big enough (*polynomial* in $|V|$, *exponential* in λ), then $\|F^K(\mathbf{0}) - \mathrm{Val}\|_\infty \leq 1/2D$

How to compute optimal values?

$$F(x)_v = \begin{cases} \max_{(v,v') \in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v') \in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\Box} \end{cases}$$

$$x^* = \lim_{n \to \infty} F^n(\mathbf{0})$$

When to stop the computation, supposing every weight is rational?

1. If $\lambda = a/b$ is rational, then x_v^* is rational too, of denominator $D = b^{O(|V|^2)}$
2. If K is big enough (*polynomial* in $|V|$, *exponential* in λ), then $\|F^K(\mathbf{0}) - \mathrm{Val}\|_\infty \leq 1/2D$
3. Use a rounding procedure to deduce Val from $F^K(\mathbf{0})$

How to compute optimal values?

$$F(x)_v = \begin{cases} \max_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[(1-\lambda)r(v,v') + \lambda x_{v'}] & \text{if } v \in V_{\Box} \end{cases}$$

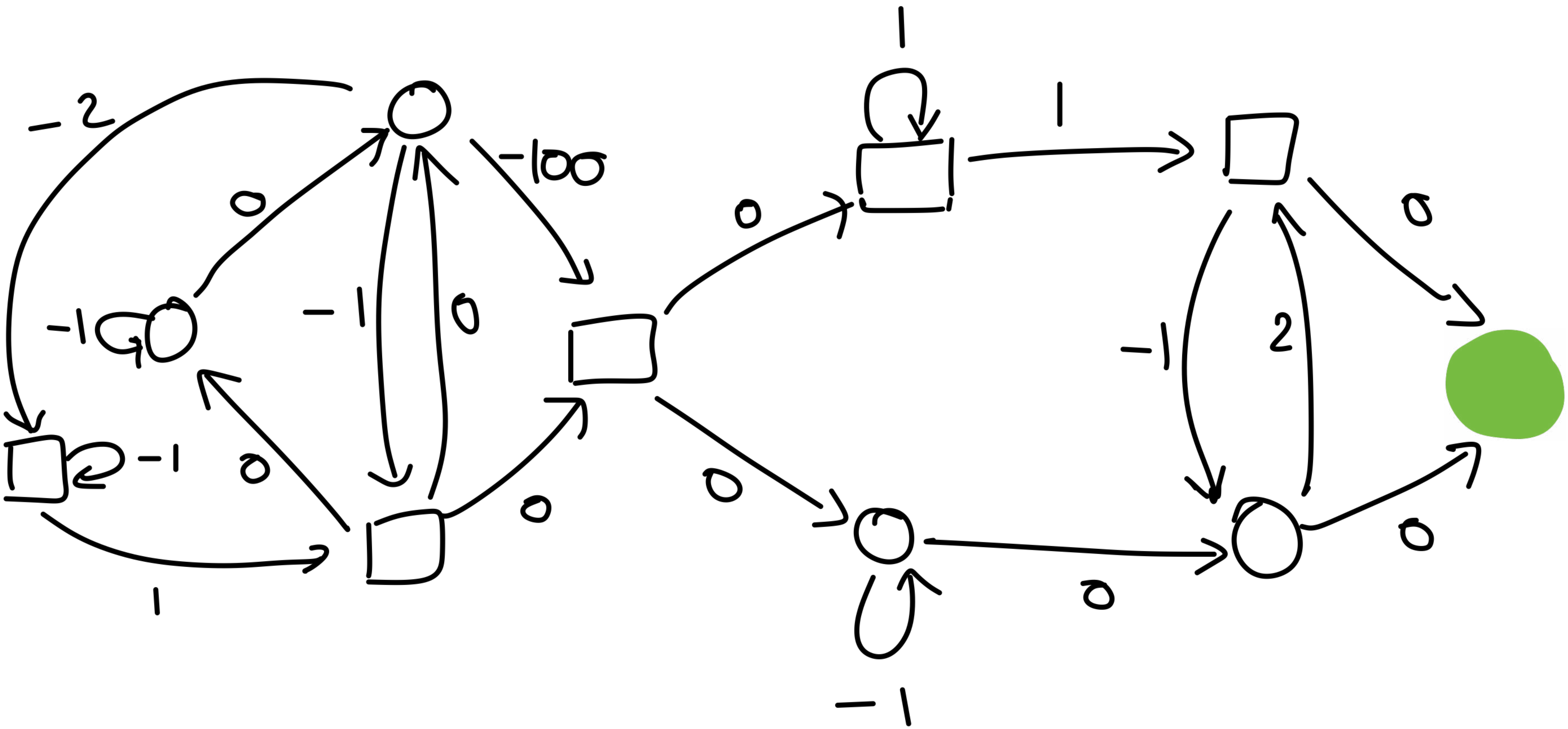
$$x^* = \lim_{n\to\infty} F^n(\mathbf{0})$$

When to stop the computation, supposing every weight is rational?

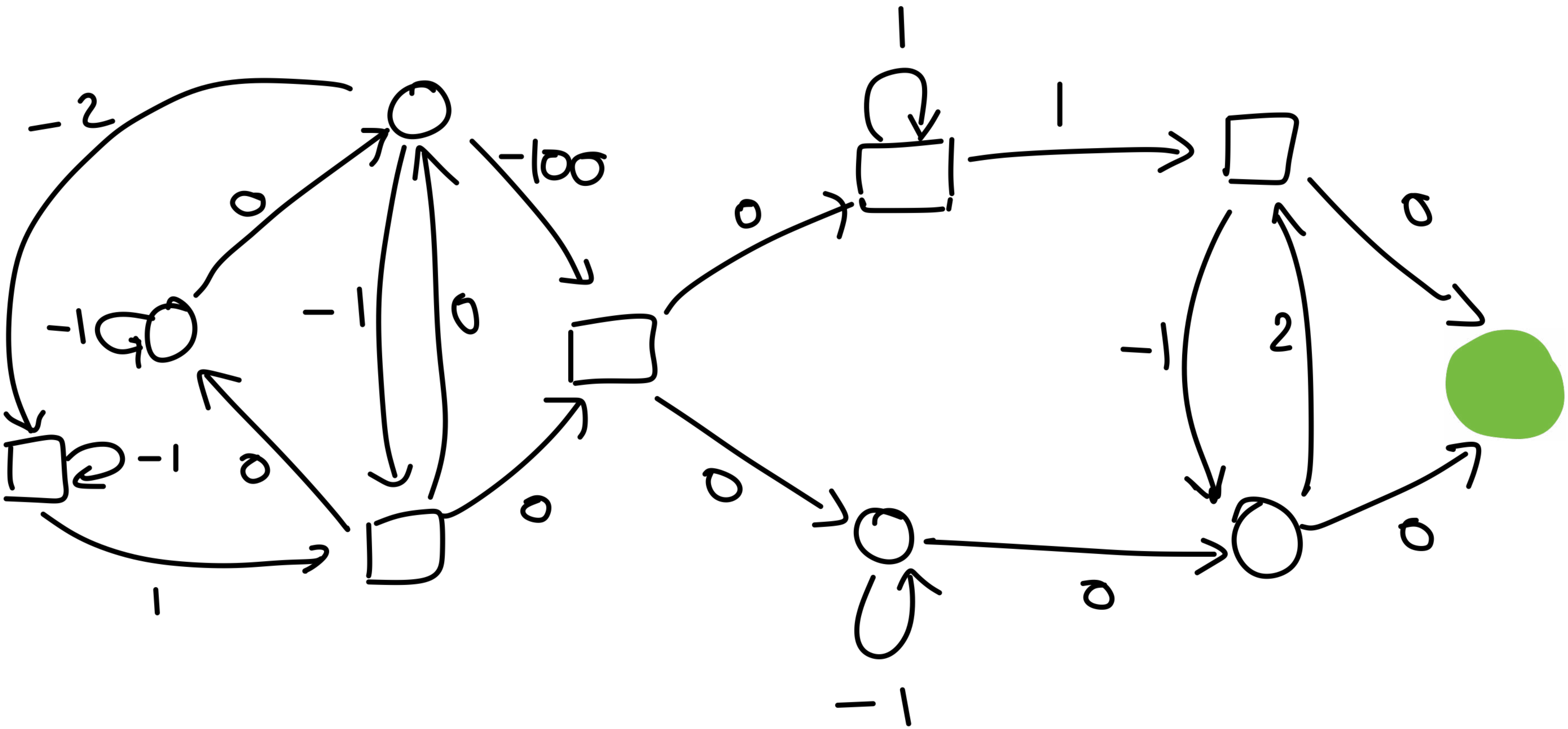
1. If $\lambda = a/b$ is rational, then x^*_v is rational too, of denominator $D = b^{O(|V|^2)}$
2. If K is big enough (*polynomial* in $|V|$, *exponential* in λ), then $\|F^K(\mathbf{0}) - \mathrm{Val}\|_\infty \leq 1/2D$
3. Use a rounding procedure to deduce Val from $F^K(\mathbf{0})$

Pseudo-polynomial algorithm

Shortest-path games



Shortest-path games



Player □ wants to reach the target with the smallest weight

Player ◯ wants to avoid the target, and if not possible, maximise the weight to the target

Non-negative case

Theorem (Khachiyan *et al* 2008)

1. Shortest-path games are determined: $\forall v \quad \mathrm{Val}_{\bigcirc}(v) = \mathrm{Val}_{\square}(v) =: \mathrm{Val}(v)$
2. Both players have *optimal* memoryless strategies:

$$\exists \sigma_{\bigcirc}^{*} \; \forall v \quad \inf_{\sigma_{\square}} \mathrm{DP}_{\lambda}(\mathrm{play}(v, \sigma_{\bigcirc}^{*}, \sigma_{\square})) = \mathrm{Val}(v)$$

$$\exists \sigma_{\square}^{*} \; \forall v \quad \sup_{\sigma_{\bigcirc}} \mathrm{DP}_{\lambda}(\mathrm{play}(v, \sigma_{\bigcirc}, \sigma_{\square}^{*})) = \mathrm{Val}(v)$$

3. The winner, with respect to a fixed threshold, can be decided in polynomial time.

Non-negative case

Theorem (Khachiyan *et al* 2008)

1. Shortest-path games are determined: $\forall v \quad \mathrm{Val}_{\bigcirc}(v) = \mathrm{Val}_{\Box}(v) =: \mathrm{Val}(v)$

2. Both players have *optimal* memoryless strategies:

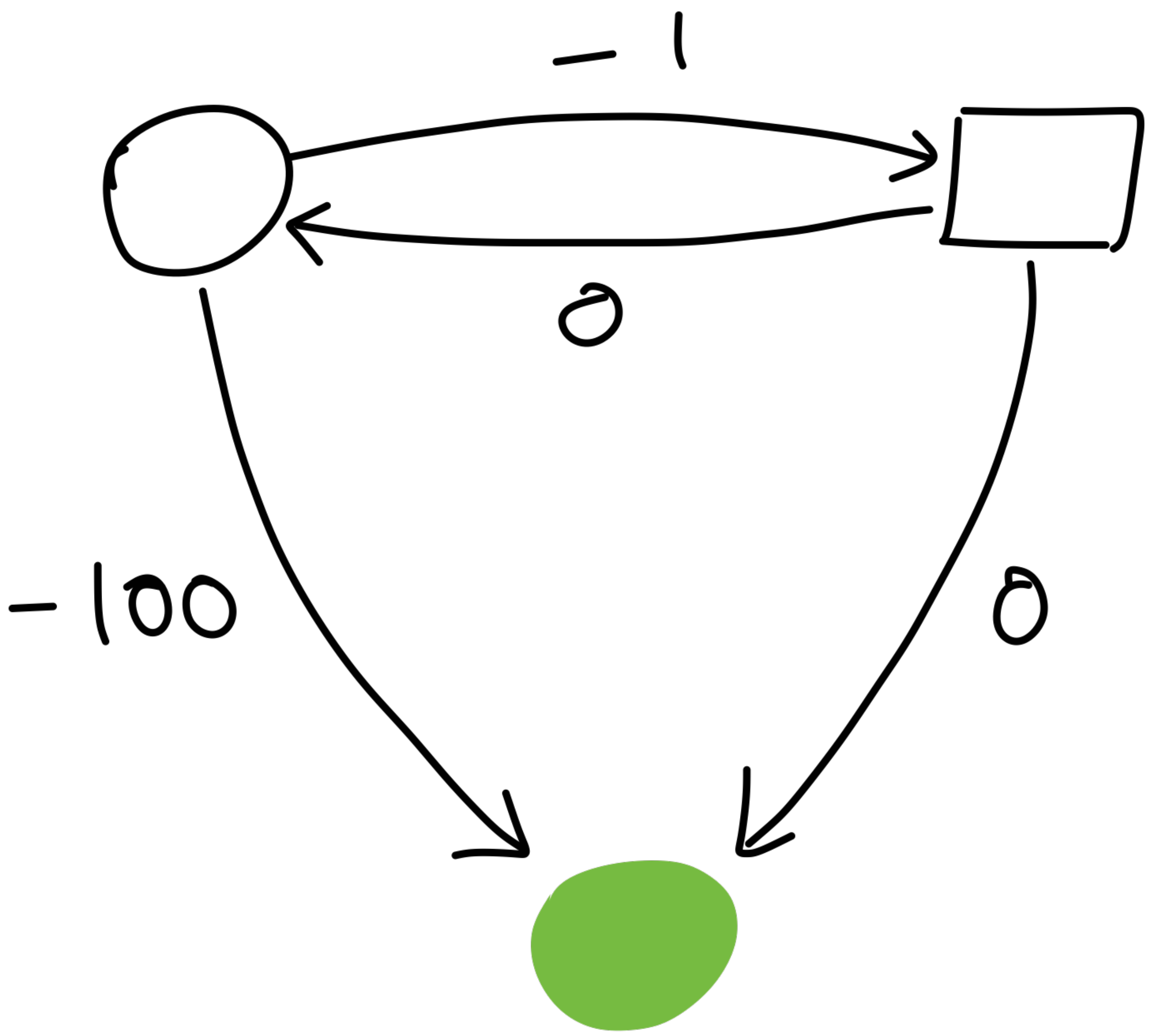
$$\exists \sigma_{\bigcirc}^{*}\ \forall v \quad \inf_{\sigma_{\Box}} \mathrm{DP}_{\lambda}(\mathrm{play}(v, \sigma_{\bigcirc}^{*}, \sigma_{\Box})) = \mathrm{Val}(v)$$

$$\exists \sigma_{\Box}^{*}\ \forall v \quad \sup_{\sigma_{\bigcirc}} \mathrm{DP}_{\lambda}(\mathrm{play}(v, \sigma_{\bigcirc}, \sigma_{\Box}^{*})) = \mathrm{Val}(v)$$

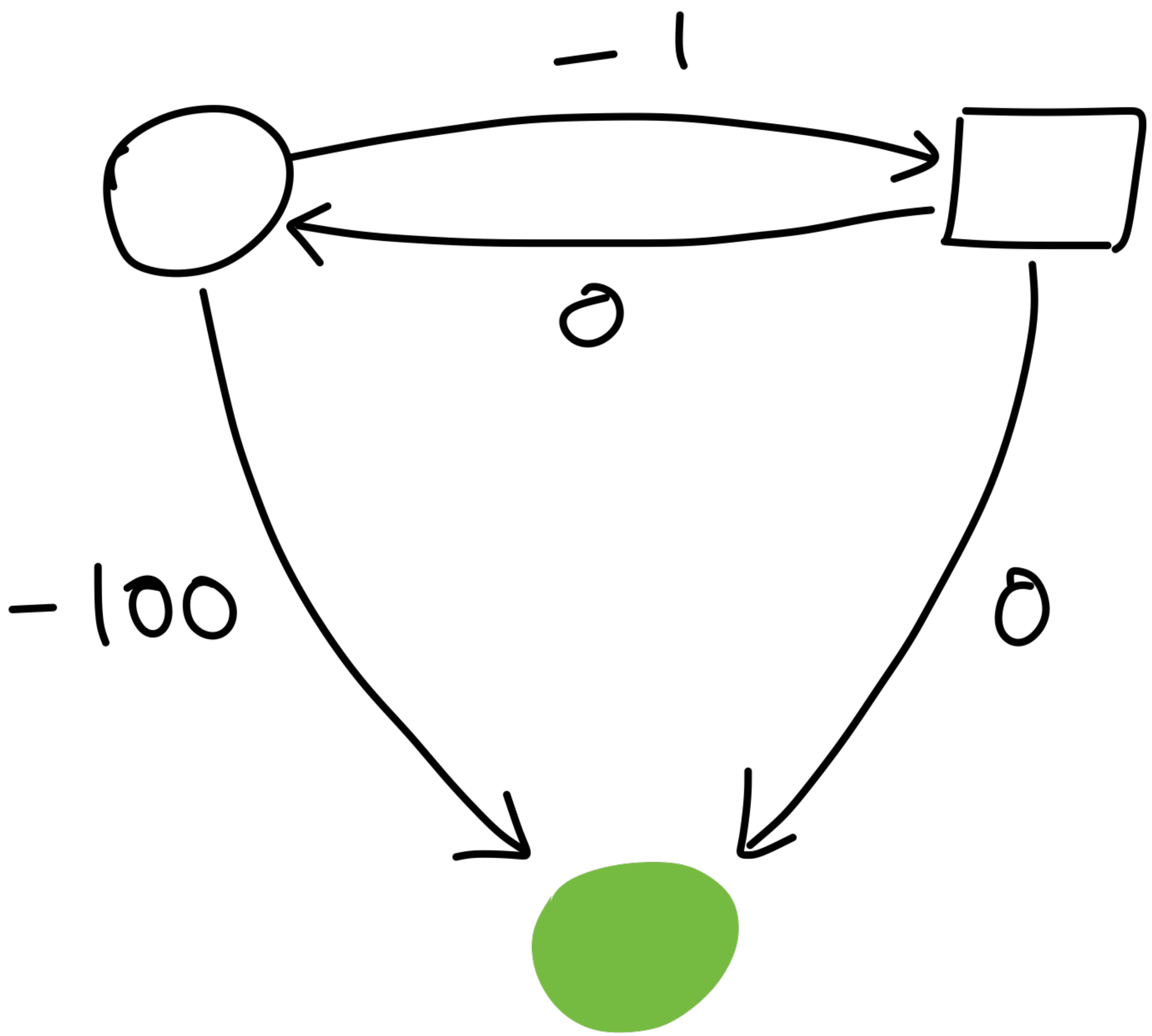
3. The winner, with respect to a fixed threshold, can be decided in polynomial time.

Adaptation of Dijkstra's shortest-path algorithm from graphs to games…

Negative weights



Negative weights



Player ☐ needs memory to play optimally!

Non-negative case

Theorem (Brihaye, Geeraerts, Haddad, Monmege 2015)

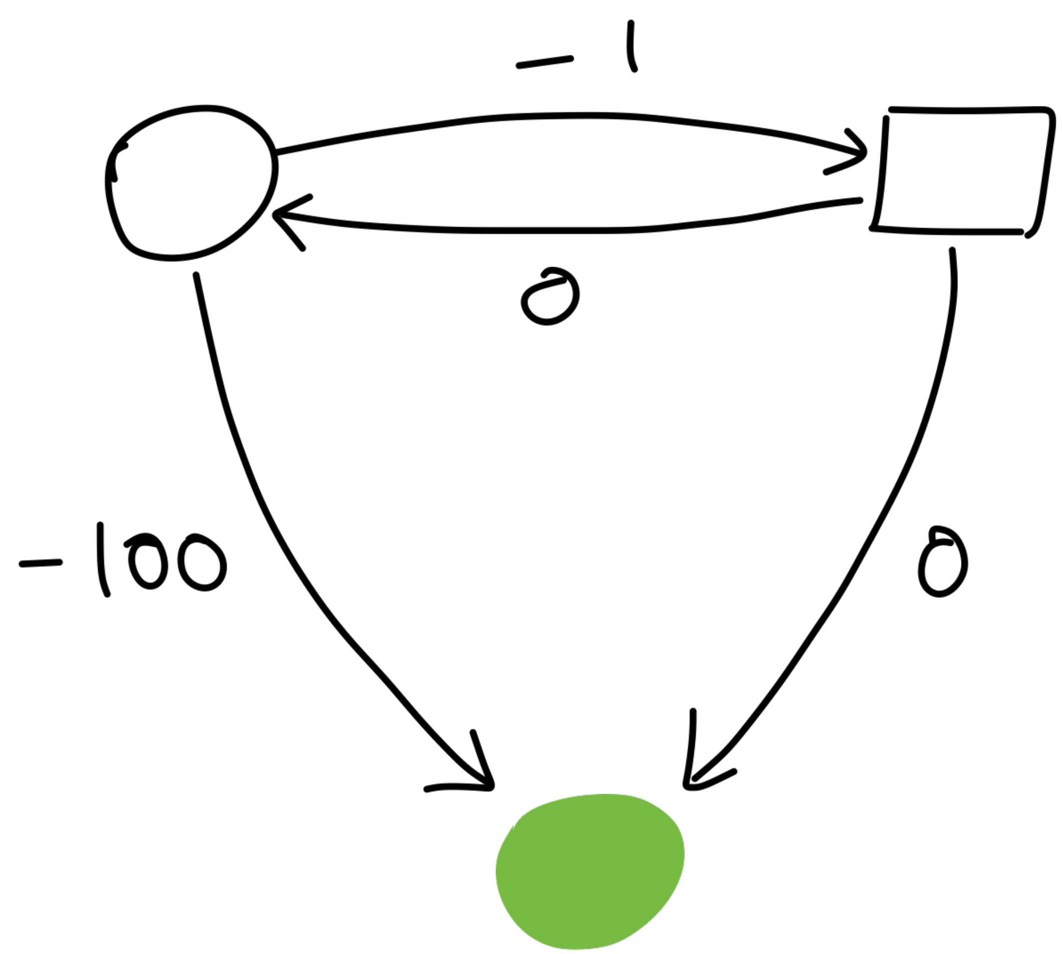
1. Shortest-path games are determined: $\forall v \quad \mathrm{Val}_{\bigcirc}(v) = \mathrm{Val}_{\square}(v) =: \mathrm{Val}(v)$
2. Both players have *optimal* ~~memoryless~~ strategies:

   $\exists \sigma^*_{\bigcirc}\ \forall v \quad \inf_{\sigma_\square} \mathrm{DP}_\lambda(\mathrm{play}(v, \sigma^*_{\bigcirc}, \sigma_\square)) = \mathrm{Val}(v)$ —> *memoryless*

   $\exists \sigma^*_{\square}\ \forall v \quad \sup_{\sigma_{\bigcirc}} \mathrm{DP}_\lambda(\mathrm{play}(v, \sigma_{\bigcirc}, \sigma^*_\square)) = \mathrm{Val}(v)$ —> *may require finite memory*
3. The winner, with respect to a fixed threshold, can be decided in pseudo-polynomial time.

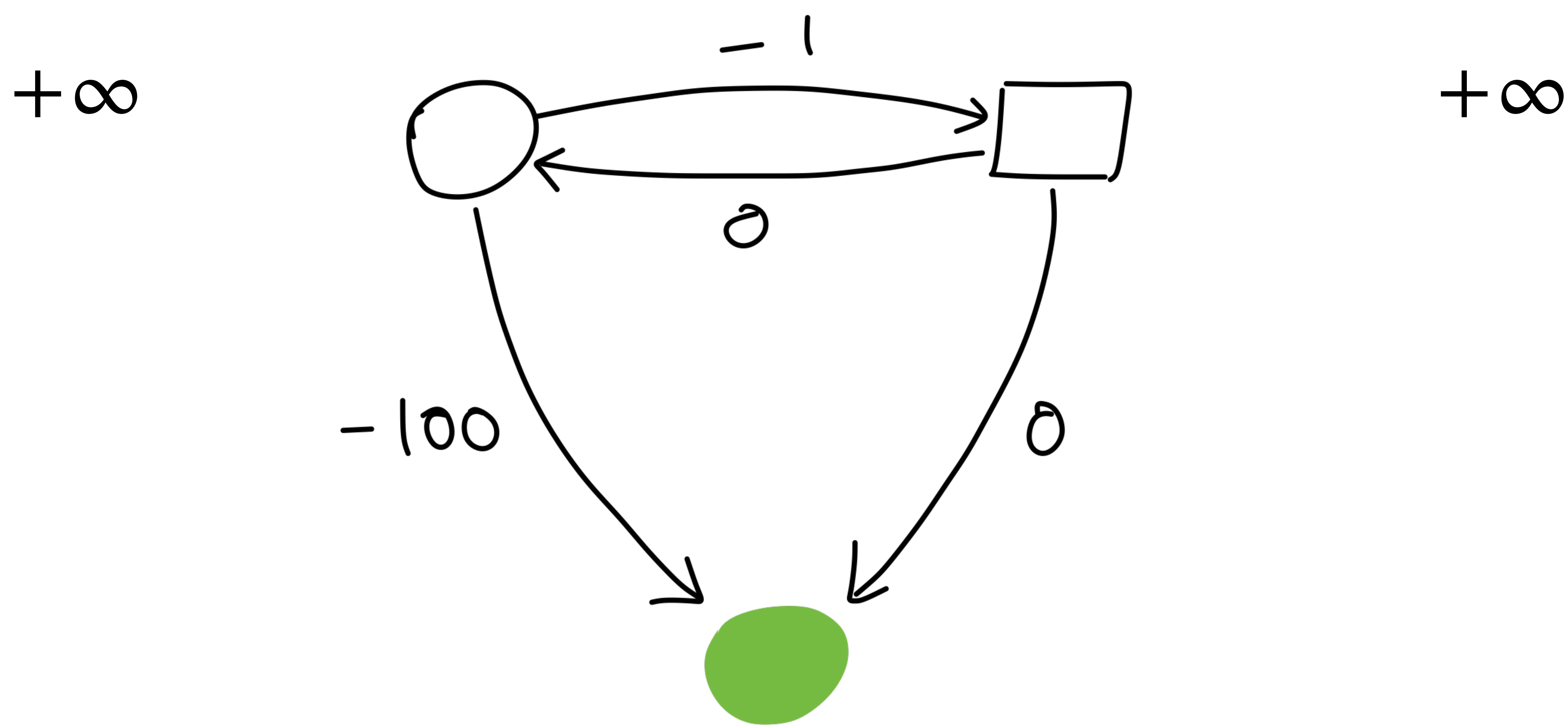
Computation of the optimal values

$$F(x)_v = \begin{cases} 0 & \text{if } v \in V_{\text{target}} \\ \max_{(v,v')\in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\square} \end{cases}$$



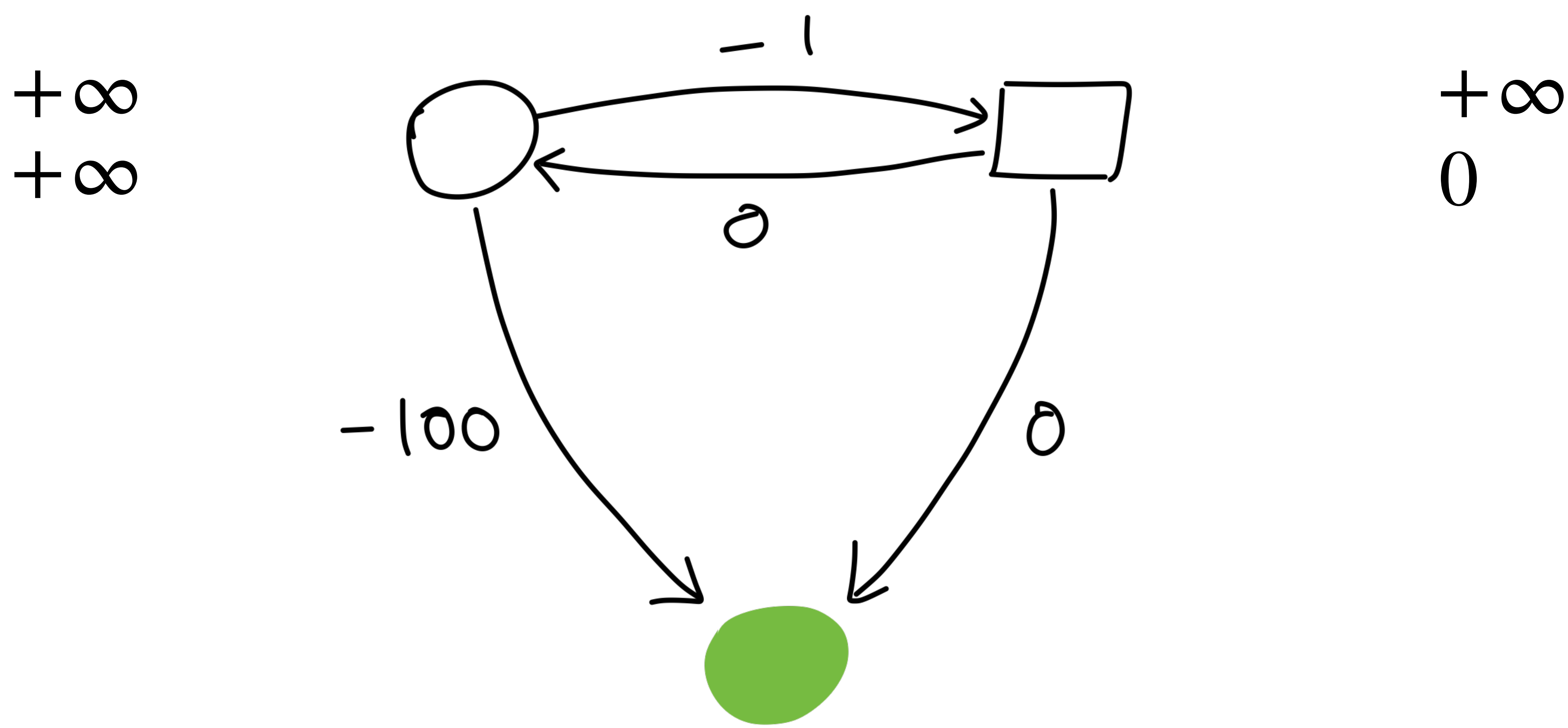
Computation of the optimal values

$$F(x)_v = \begin{cases} 0 & \text{if } v \in V_{\text{target}} \\ \max_{(v,v')\in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\square} \end{cases}$$



Computation of the optimal values

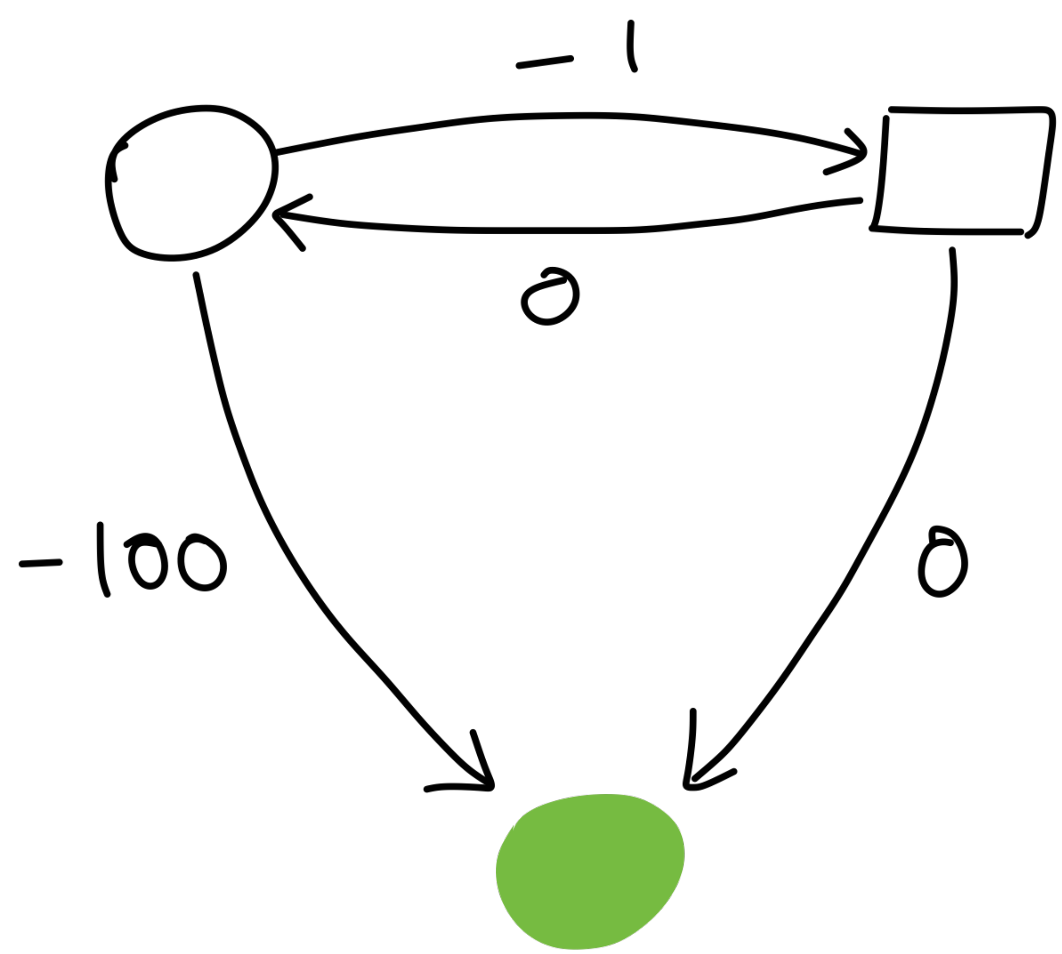
$$F(x)_v = \begin{cases} 0 & \text{if } v \in V_{\text{target}} \\ \max_{(v,v')\in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\square} \end{cases}$$



Computation of the optimal values

$$F(x)_v = \begin{cases} 0 & \text{if } v \in V_{\text{target}} \\ \max_{(v,v')\in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\square} \end{cases}$$

$+\infty$
$+\infty$
-1

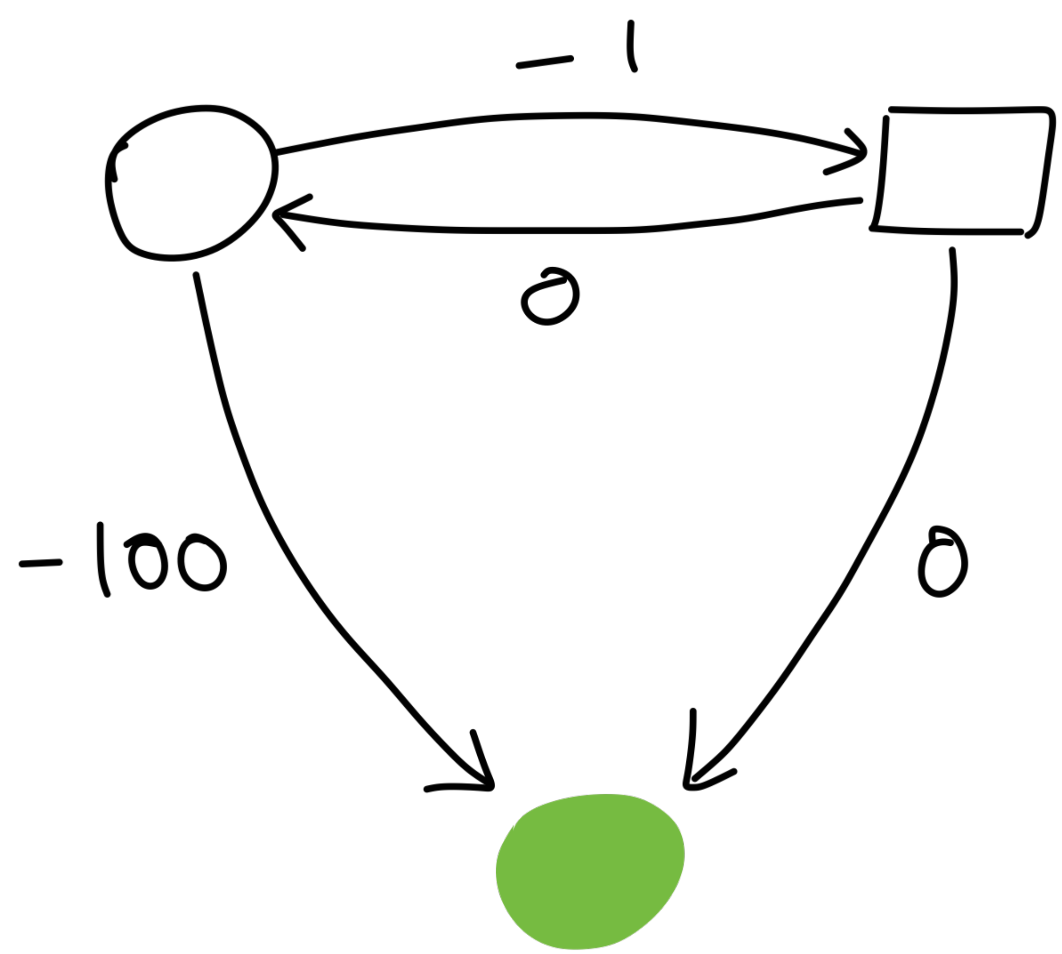


$+\infty$
0
0

Computation of the optimal values

$$F(x)_v = \begin{cases} 0 & \text{if } v \in V_{\text{target}} \\ \max_{(v,v')\in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\text{O}} \\ \min_{(v,v')\in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\square} \end{cases}$$

$+\infty$
$+\infty$
-1
-1

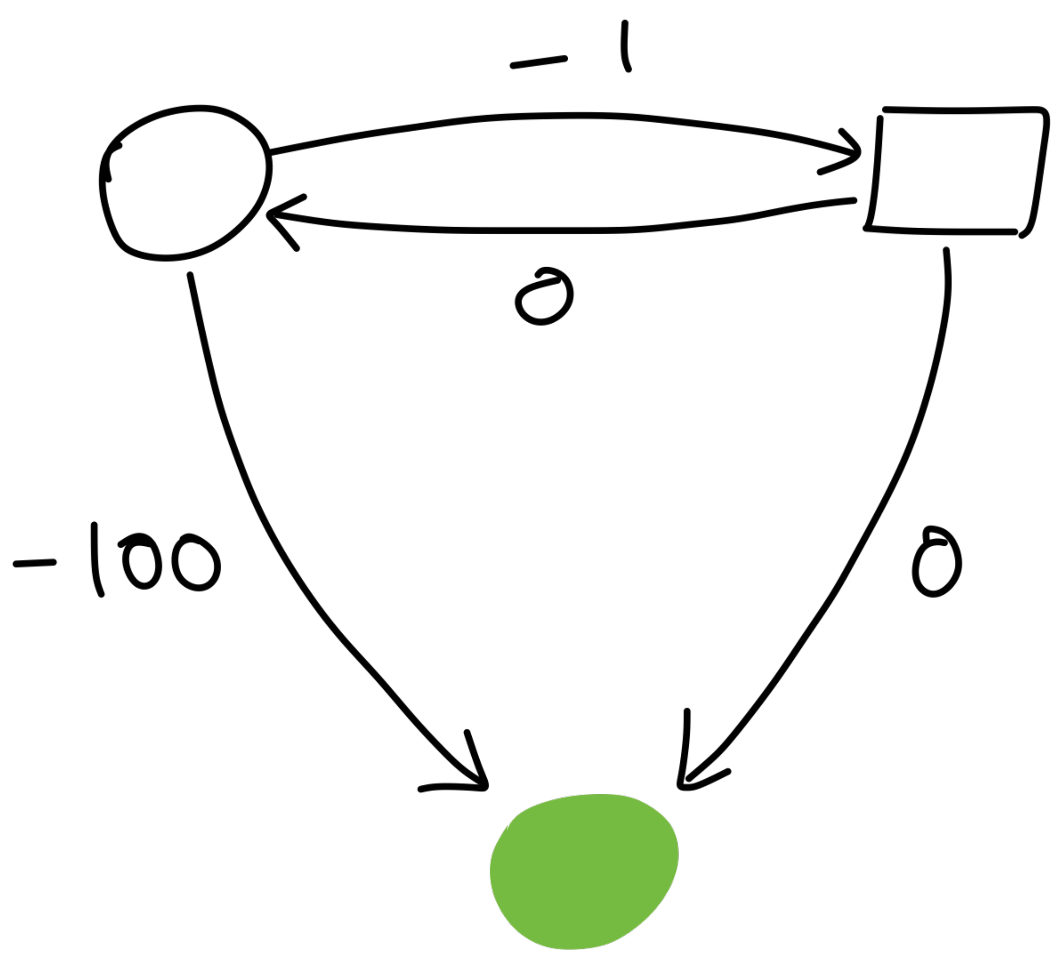


$+\infty$
0
0
-1

Computation of the optimal values

$$F(x)_v = \begin{cases} 0 & \text{if } v \in V_{\text{target}} \\ \max_{(v,v')\in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\square} \end{cases}$$

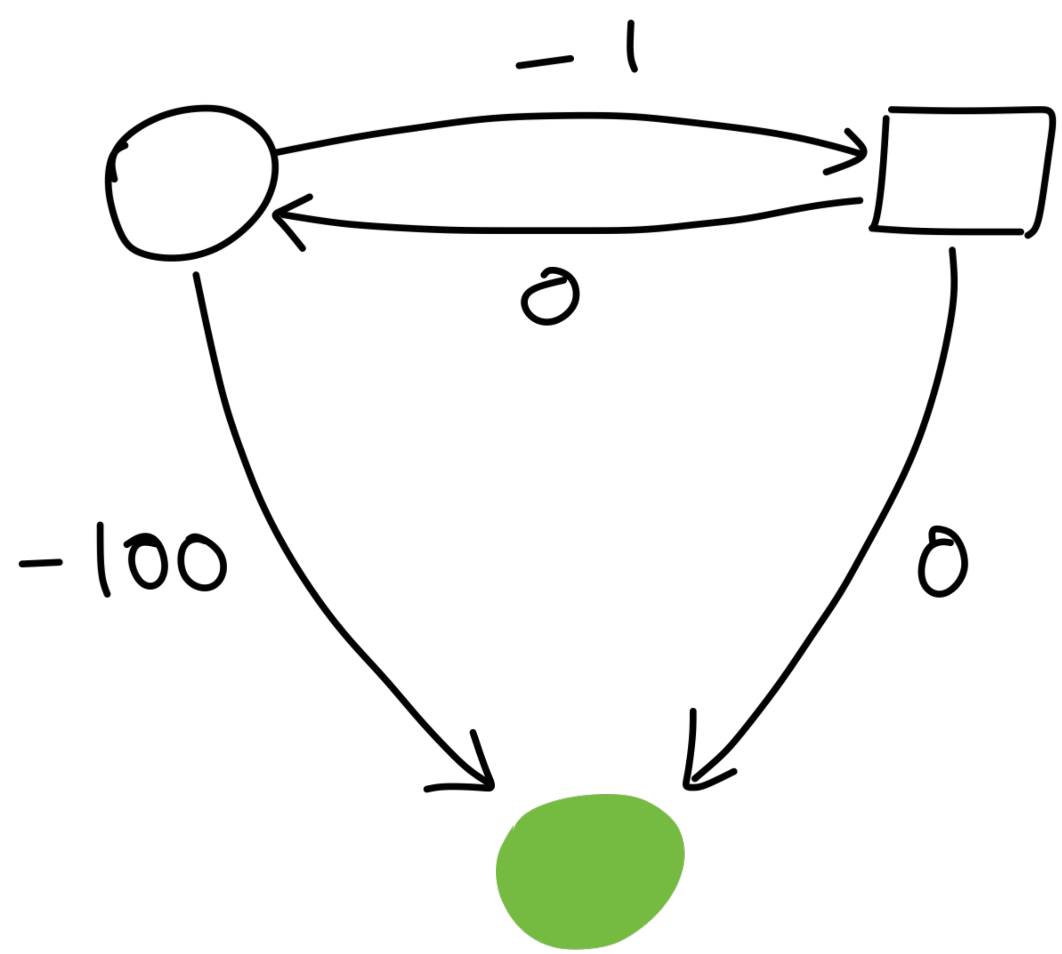
$+\infty$	$+\infty$
$+\infty$	0
-1	0
-1	-1
-2	-1



Computation of the optimal values

$$F(x)_v = \begin{cases} 0 & \text{if } v \in V_{\text{target}} \\ \max_{(v,v')\in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\square} \end{cases}$$

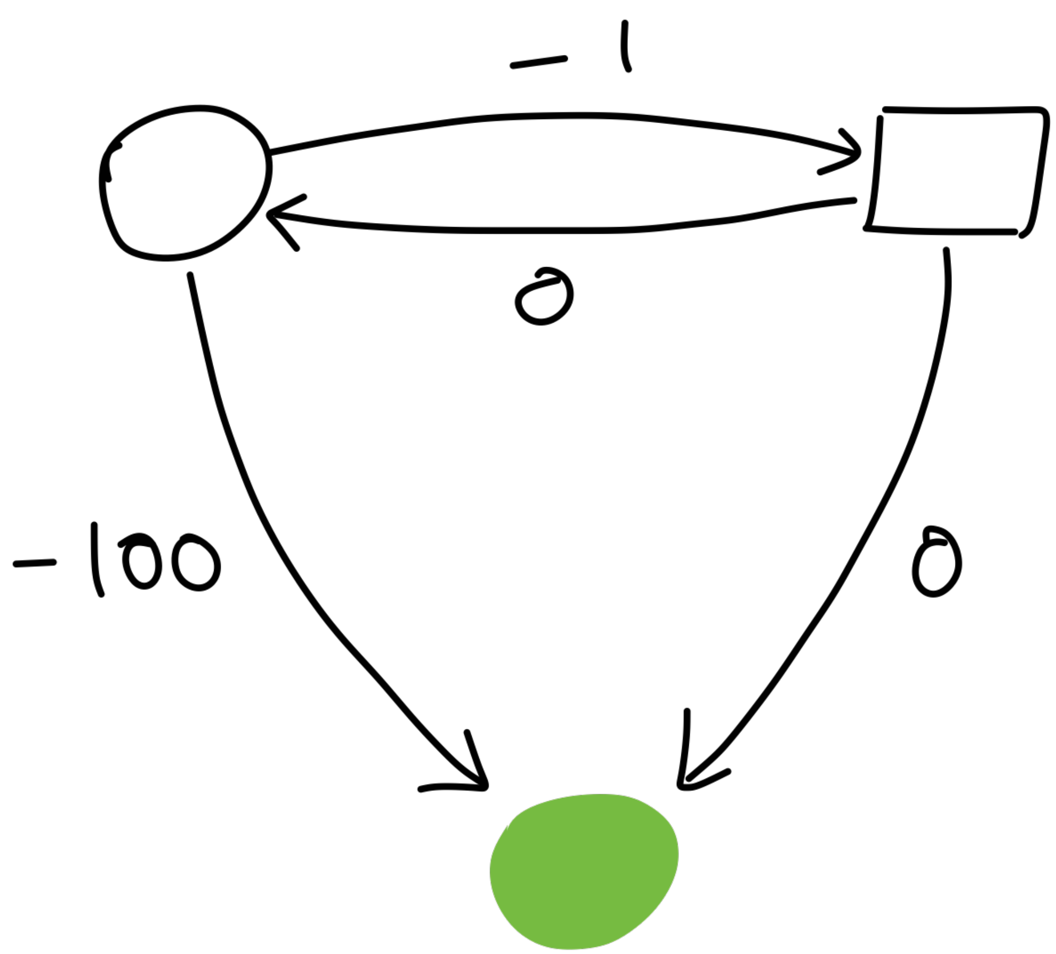
$+\infty$	$+\infty$
$+\infty$	0
-1	0
-1	-1
-2	-1
...	...
-100	-100



Computation of the optimal values

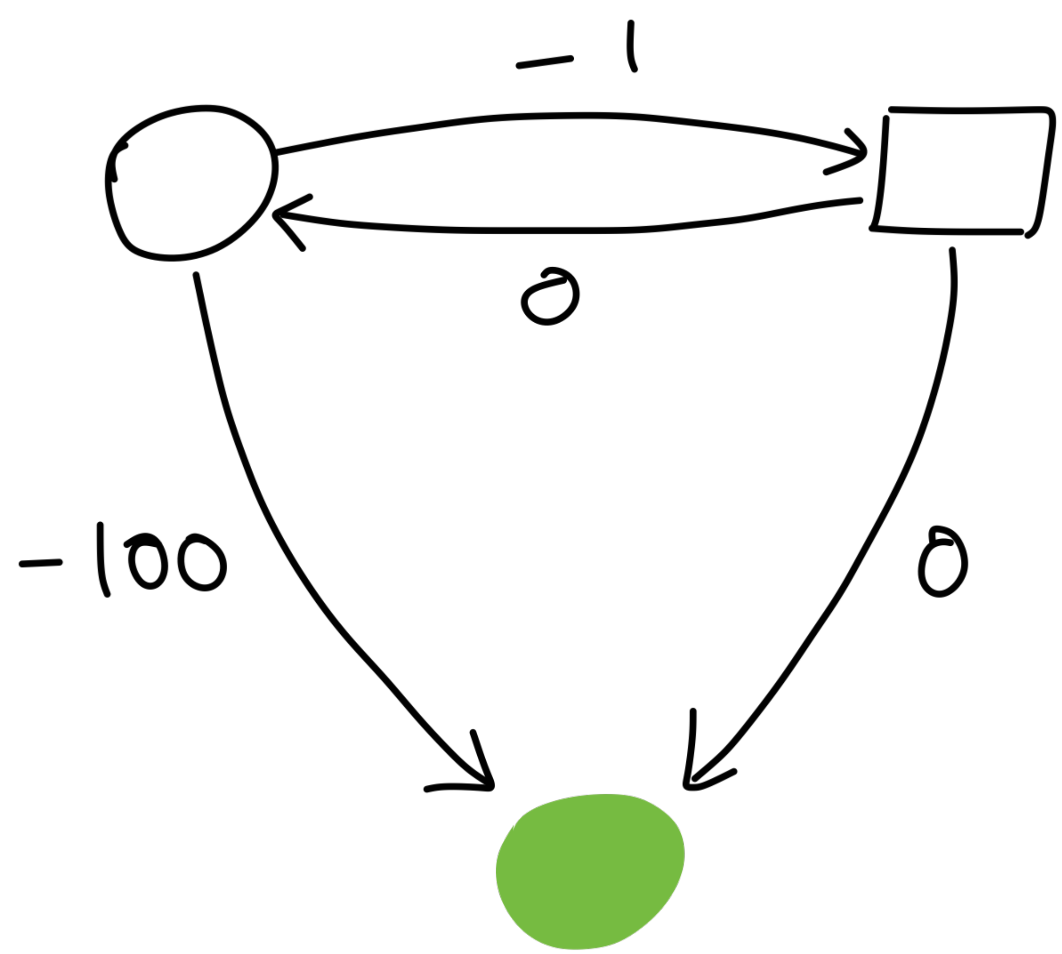
$$F(x)_v = \begin{cases} 0 & \text{if } v \in V_{\text{target}} \\ \max_{(v,v') \in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v') \in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\Box} \end{cases}$$

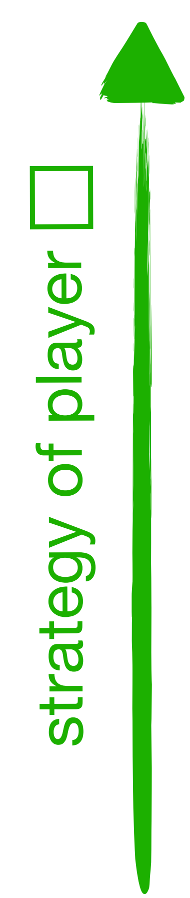
$+\infty$	$+\infty$
$+\infty$	0
-1	0
-1	-1
-2	-1
...	...
-100	-100
-100	-100



Computation of the optimal values

$$F(x)_v = \begin{cases} 0 & \text{if } v \in V_{\text{target}} \\ \max_{(v,v')\in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\bigcirc} \\ \min_{(v,v')\in E}[r(v,v') + x_{v'}] & \text{if } v \in V_{\square} \end{cases}$$





$+\infty$	$+\infty$
$+\infty$	0
-1	0
-1	-1
-2	-1
...	...
-100	-100
-100	-100

Non-negative case

Theorem (Brihaye, Geeraerts, Haddad, Monmege 2015)

1. Shortest-path games are determined: $\forall v \quad \mathrm{Val}_{\bigcirc}(v) = \mathrm{Val}_{\square}(v) =: \mathrm{Val}(v)$
2. Both players have *optimal* ~~memoryless~~ strategies:

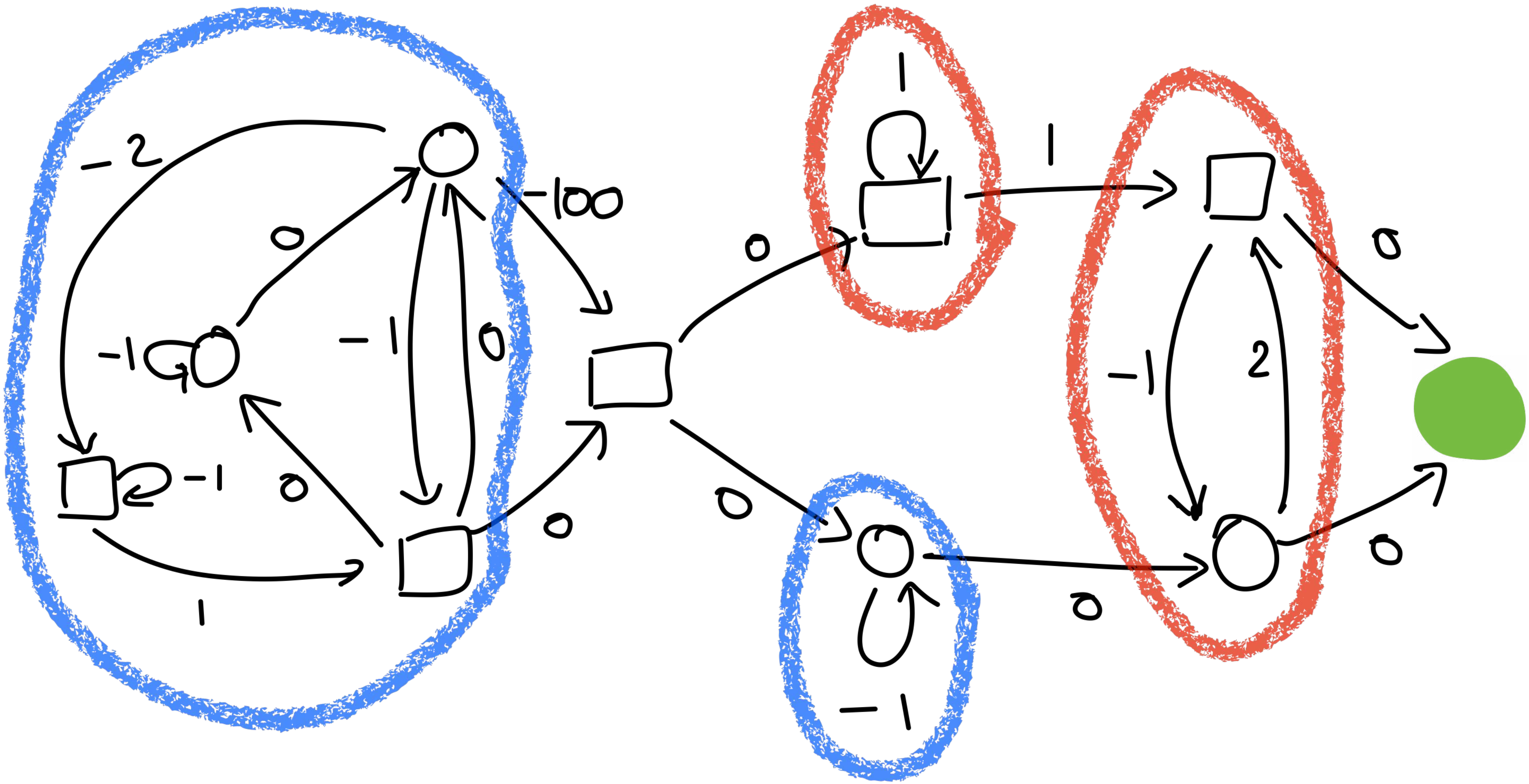
$$\exists \sigma_{\bigcirc}^{*}\ \forall v \quad \inf_{\sigma_{\square}} \mathrm{DP}_{\lambda}(\mathrm{play}(v, \sigma_{\bigcirc}^{*}, \sigma_{\square})) = \mathrm{Val}(v) \qquad \text{—> } \textit{memoryless}$$

$$\exists \sigma_{\square}^{*}\ \forall v \quad \sup_{\sigma_{\bigcirc}} \mathrm{DP}_{\lambda}(\mathrm{play}(v, \sigma_{\bigcirc}, \sigma_{\square}^{*})) = \mathrm{Val}(v) \qquad \text{—> } \textit{may require finite memory}$$

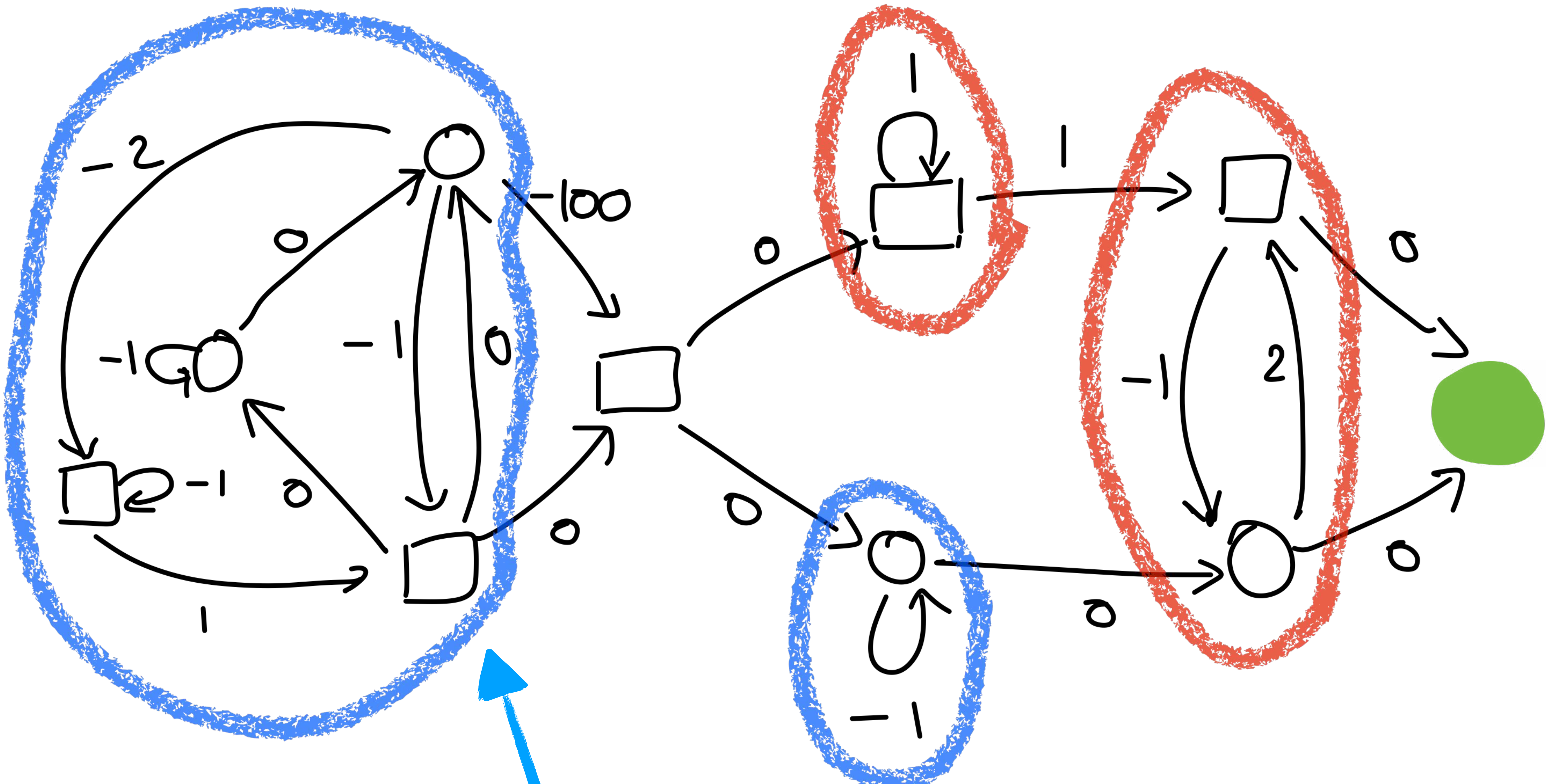
3. The winner, with respect to a fixed threshold, can be decided in pseudo-polynomial time.

Polynomial wrt |V|
Polynomial wrt weights encoded in unary

Interesting fragment?



Interesting fragment?



only case where pseudo-polynomial complexity…

Divergent weighted games

No cycles of weight = 0

Divergent weighted games

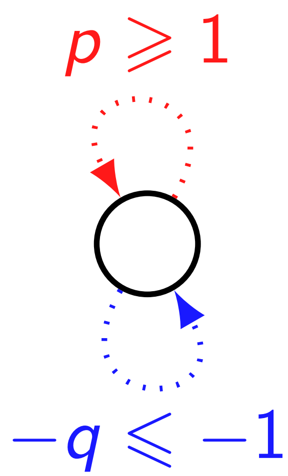
No cycles of weight = 0

Characterisation (Busatto-Gaston, Monmege, Reynier 2017)
All cycles in an SCC have the same sign.

Divergent weighted games

No cycles of weight = 0

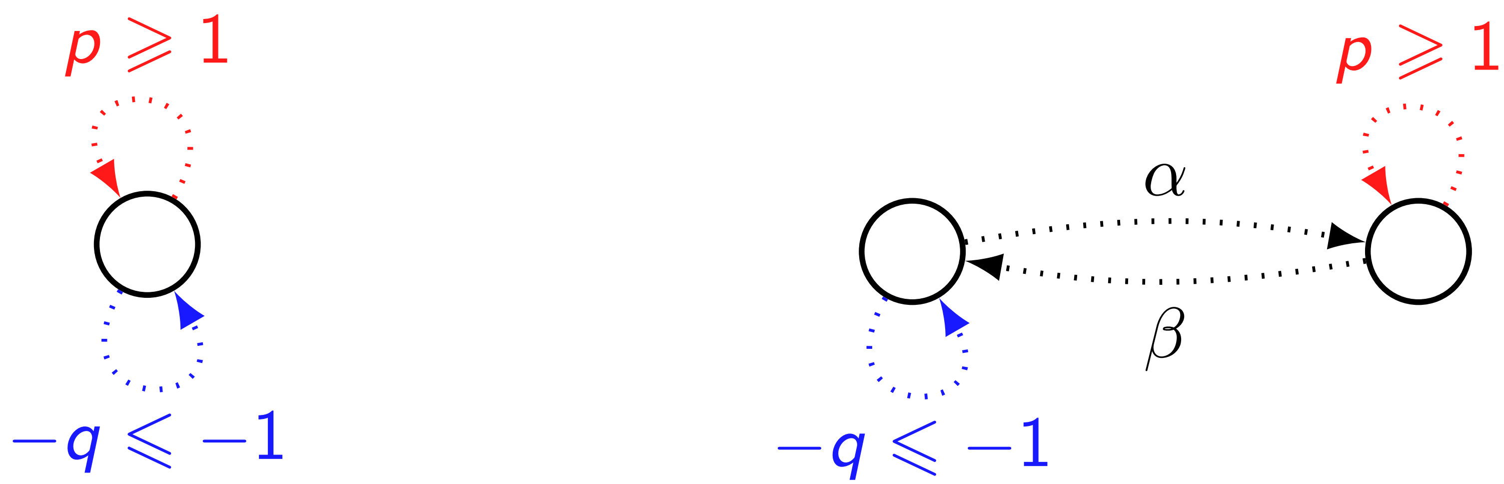
Characterisation (Busatto-Gaston, Monmege, Reynier 2017)
All cycles in an SCC have the same sign.



Divergent weighted games

No cycles of weight = 0

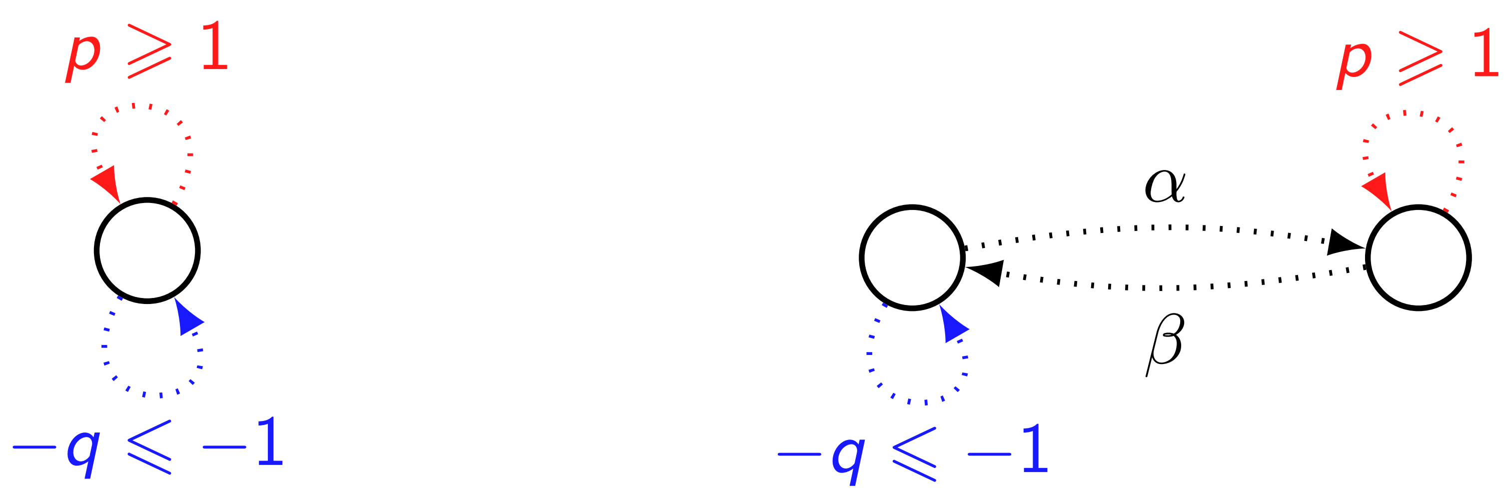
Characterisation (Busatto-Gaston, Monmege, Reynier 2017)
All cycles in an SCC have the same sign.



Divergent weighted games

No cycles of weight = 0

Characterisation (Busatto-Gaston, Monmege, Reynier 2017)
All cycles in an SCC have the same sign.



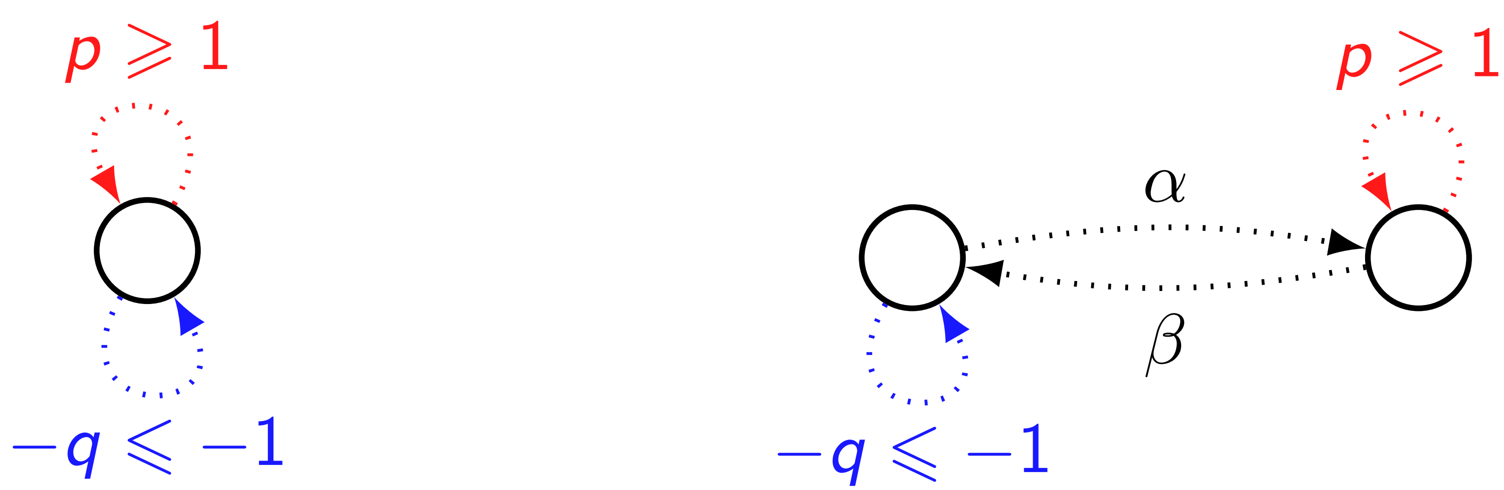
In positive SCCs, value iteration algorithm converges in polynomial time.
In negative SCCs :

1. outside the attractor of Player ◯ —> value $-\infty$
2. value iteration algorithm starting from $-\infty$ (instead of $+\infty$) converges in polynomial time

Divergent weighted games

No cycles of weight = 0

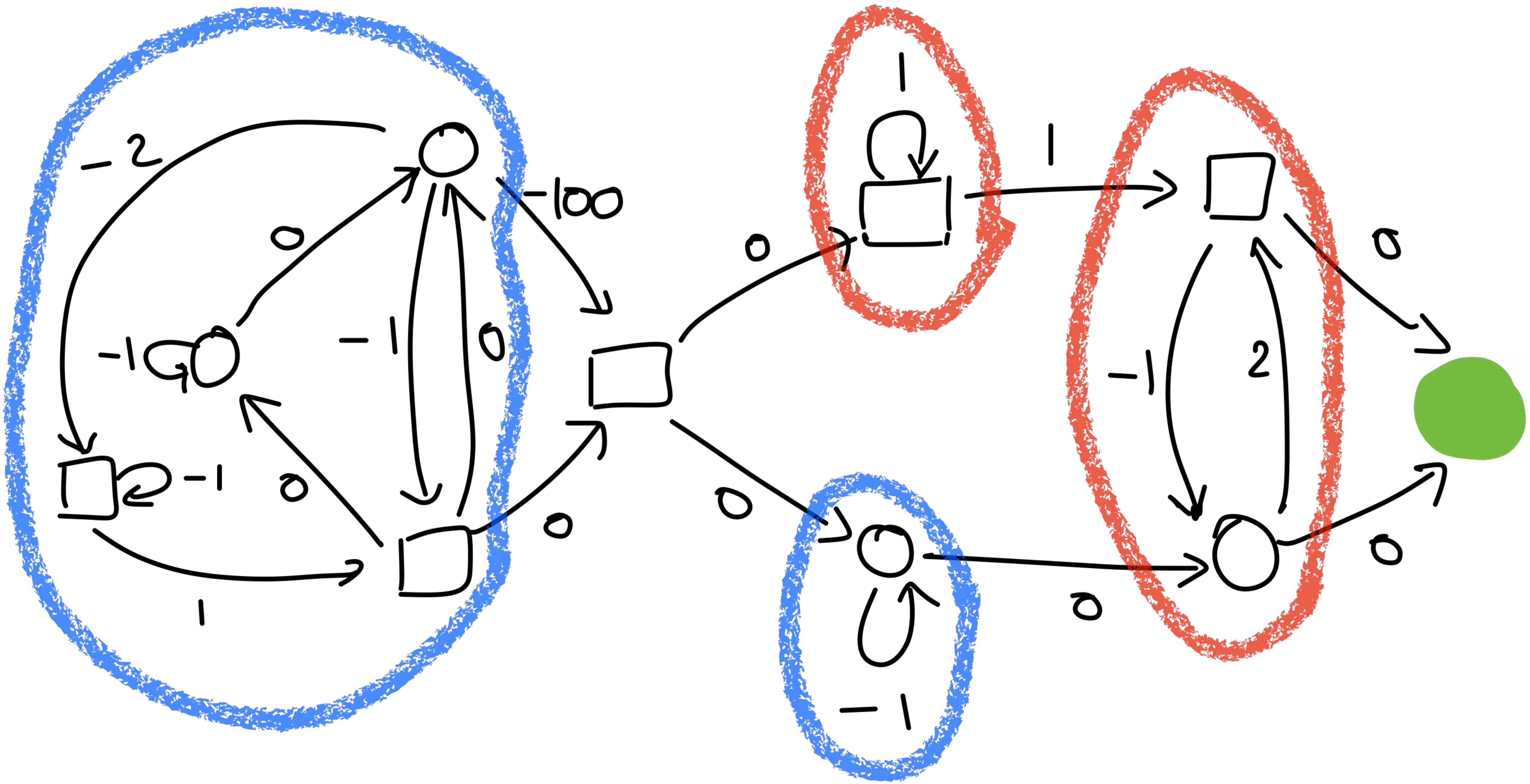
Characterisation (Busatto-Gaston, Monmege, Reynier 2017)
All cycles in an SCC have the same sign.



In positive SCCs, value iteration algorithm converges in polynomial time.
In negative SCCs :

1. outside the attractor of Player ◯ —> value $-\infty$
2. value iteration algorithm starting from $-\infty$ (instead of $+\infty$) converges in polynomial time

Theorem (Busatto-Gaston, Monmege, Reynier 2017)
Optimal values/strategies in divergent weighted games are computable in polynomial time.



-2
-100
0
-1
-1
0
-1
0
1
0
0
1
1
0
0
-1
2
0
0
0
-1

$$\boxed{\text{Environment}} \quad \| \quad \boxed{\text{Controller??}} \quad \models \quad \text{Spec}$$

Two-player game

$\boxed{\text{Environment}} \quad \| \quad \boxed{\text{Controller??}} \quad \models \quad \text{Spec}$

Two-player game

Among all *valid* controllers, choose a *cheap/efficient* one

Two-player **weighted** game

$\boxed{\text{Environment}} \quad \| \quad \boxed{\text{Controller??}} \quad \models \quad \text{Spec}$

Two-player game

Among all *valid* controllers, choose a *cheap/efficient* one

Two-player **weighted** game

Additional difficulty: **negative weights**

$\Longrightarrow$ to model production/consumption of resources

$$\boxed{\text{Environment}} \quad \| \quad \boxed{\text{Controller??}} \quad \models \quad \text{Spec}$$

Two-player game

Real-time requirements/environment $\Longrightarrow$ real-time controller

Two-player **timed** game

Among all *valid* controllers, choose a *cheap/efficient* one

Two-player **weighted** timed game

Additional difficulty: **negative weights**

$\Longrightarrow$ to model production/consumption of resources

	Peak-hour 	Offpeak-hour 
	15 c€/kWh	12 c€/kWh
rate:	total power $\times$ 15 c€/h	total power $\times$ 12 c€/h

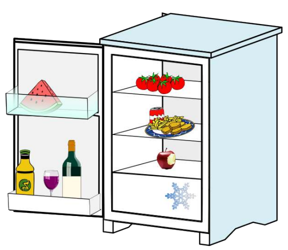
	Peak-hour 	Offpeak-hour 
	15 c€/kWh	12 c€/kWh
rate:	total power $\times$ 15 c€/h	total power $\times$ 12 c€/h

states to record which device is on/off: computation of the total power

	Peak-hour 	Offpeak-hour 
	15 c€/kWh	12 c€/kWh
rate:	total power $\times$ 15 c€/h	total power $\times$ 12 c€/h

states to record which device is on/off: computation of the total power

Power consumption:

-  100W (1.5 c€/h in peak-hour, 1.2 c€/h in offpeak-hour)
-  2500W (37.5 c€/h in peak-hour, 30 c€/h in offpeak-hour)
-  2000W (24 c€/h in offpeak-hour)

	Peak-hour 	Offpeak-hour 	Solar panels 
	15 c€/kWh	12 c€/kWh	Reselling: 20 c€/kWh
rate:	total power $\times$ 15 c€/h	total power $\times$ 12 c€/h	-0.5×20 c€/h

	Peak-hour 	Offpeak-hour 	Solar panels 
	15 c€/kWh	12 c€/kWh	Reselling: 20 c€/kWh
rate:	total power $\times$ 15 c€/h	total power $\times$ 12 c€/h	-0.5×20 c€/h

states to record which device is on/off: computation of the total power

	Peak-hour 	Offpeak-hour 	Solar panels 
	15 c€/kWh	12 c€/kWh	Reselling: 20 c€/kWh
rate:	total power $\times$ 15 c€/h	total power $\times$ 12 c€/h	-0.5×20 c€/h

states to record which device is on/off: computation of the total power

Environment: user profile, weather profile  / 
Controller: chooses contract (discrete cost for the monthly subscription) and exact consumption (what, when...)

	Peak-hour 	Offpeak-hour 	Solar panels 
	15 c€/kWh	12 c€/kWh	Reselling: 20 c€/kWh
rate:	total power $\times$ 15 c€/h	total power $\times$ 12 c€/h	-0.5×20 c€/h

states to record which device is on/off: computation of the total power

Environment: user profile, weather profile  / 
Controller: chooses contract (discrete cost for the monthly subscription) and exact consumption (what, when...)

Goal: optimise the energy consumption based on the cost

	Peak-hour 	Offpeak-hour 	Solar panels 
	15 c€/kWh	12 c€/kWh	Reselling: 20 c€/kWh
rate:	total power $\times$ 15 c€/h	total power $\times$ 12 c€/h	-0.5×20 c€/h

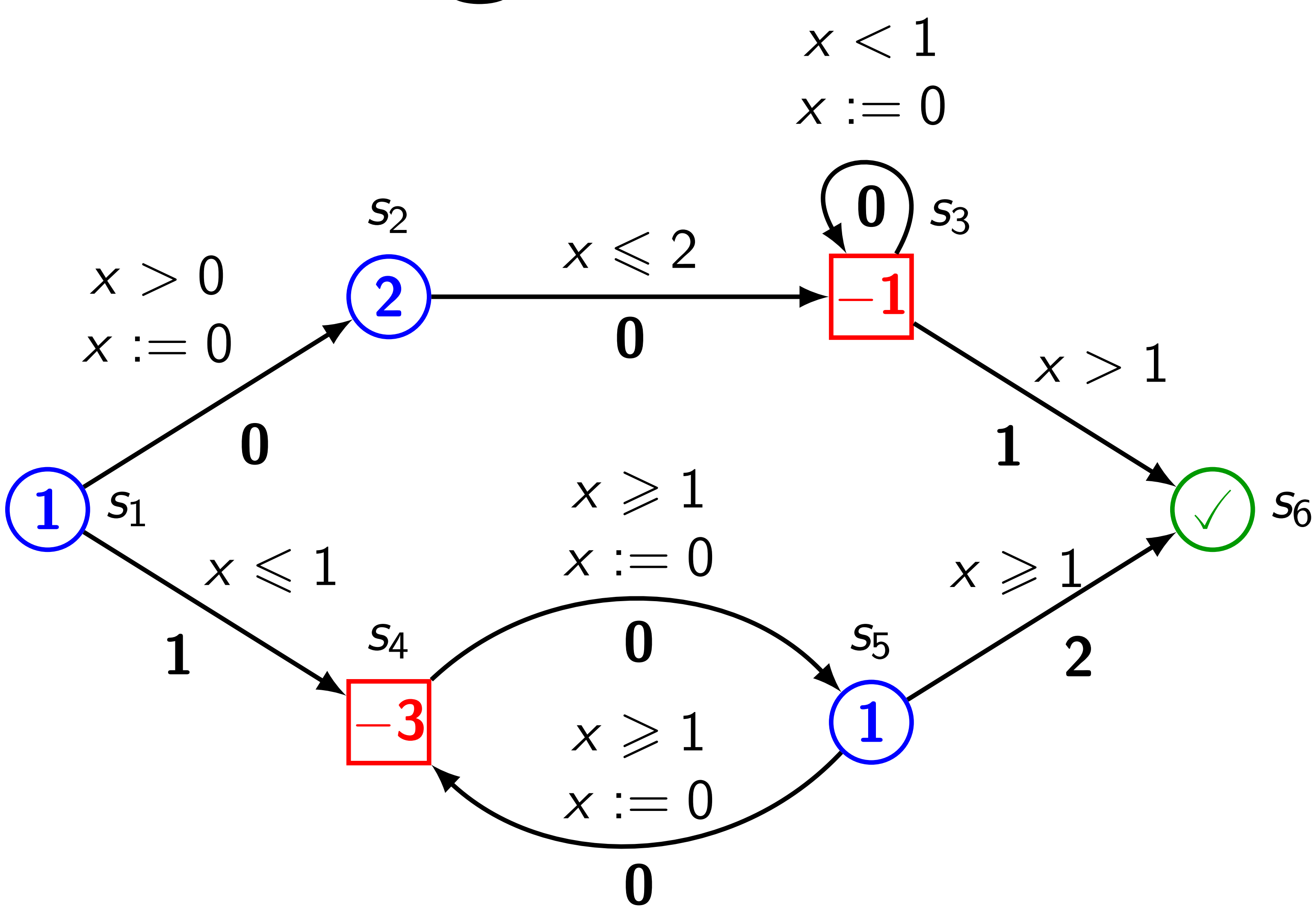
states to record which device is on/off: computation of the total power

Environment: user profile, weather profile  / 
Controller: chooses contract (discrete cost for the monthly subscription) and exact consumption (what, when...)

Goal: optimise the energy consumption based on the cost

Solution 1 : discretisation of time, resolution via a *weighted game*
Solution 2 : thin time behaviours, resolution via a *weighted timed game*

Weighted timed games

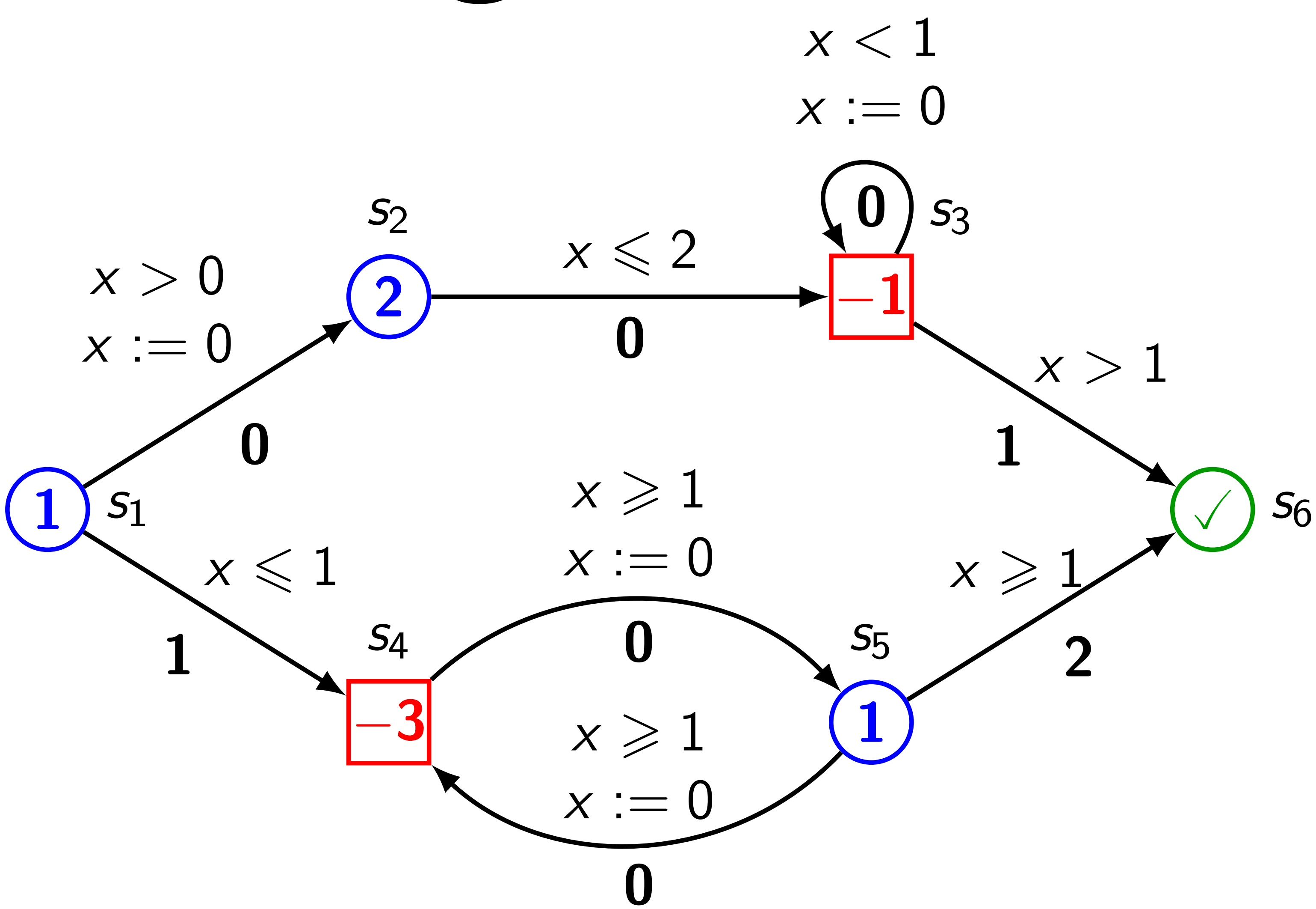


Timed automaton
with state partition between
2 players
+ reachability objective
+ linear rates on states
+ discrete weights on
transitions

$$(s_1, 0) \xrightarrow{0.4, \searrow} (s_4, 0.4) \xrightarrow{0.6, \rightarrow} (s_5, 0) \xrightarrow{1.5, \leftarrow} (s_4, 0) \xrightarrow{1.1, \rightarrow} (s_5, 0) \xrightarrow{2, \nearrow} (\checkmark, 2)$$

$$\mathbf{1} \times 0.4 + \mathbf{1} \quad - \mathbf{3} \times 0.6 + \mathbf{0} \quad + \mathbf{1} \times 1.5 + \mathbf{0} \quad - \mathbf{3} \times 1.1 + \mathbf{0} \quad + \mathbf{1} \times 2 + \mathbf{2} \qquad = 1.8$$

Weighted timed games



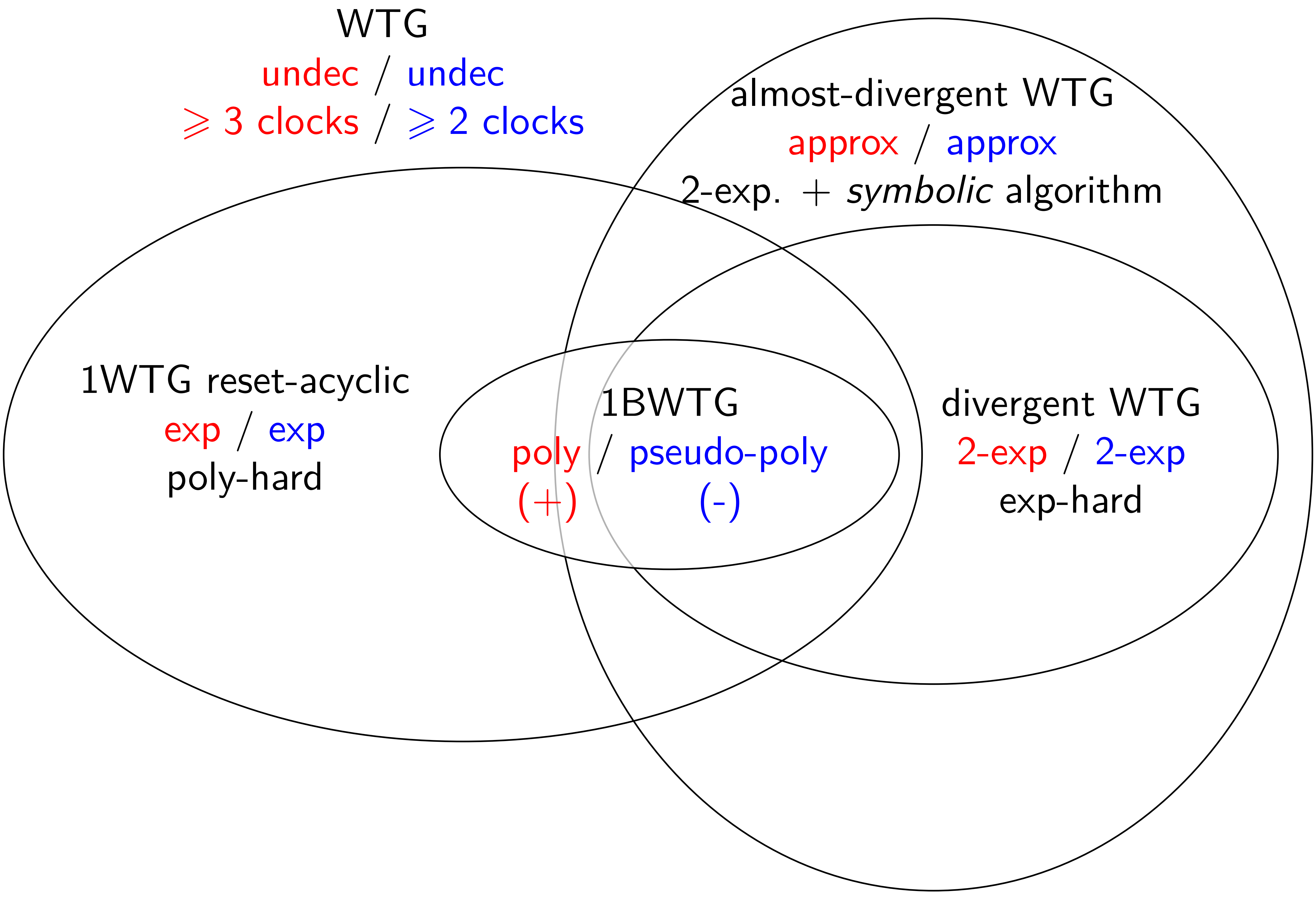
Timed automaton
with state partition between
2 players
$+$ reachability objective
$+$ linear rates on states
$+$ discrete weights on
transitions

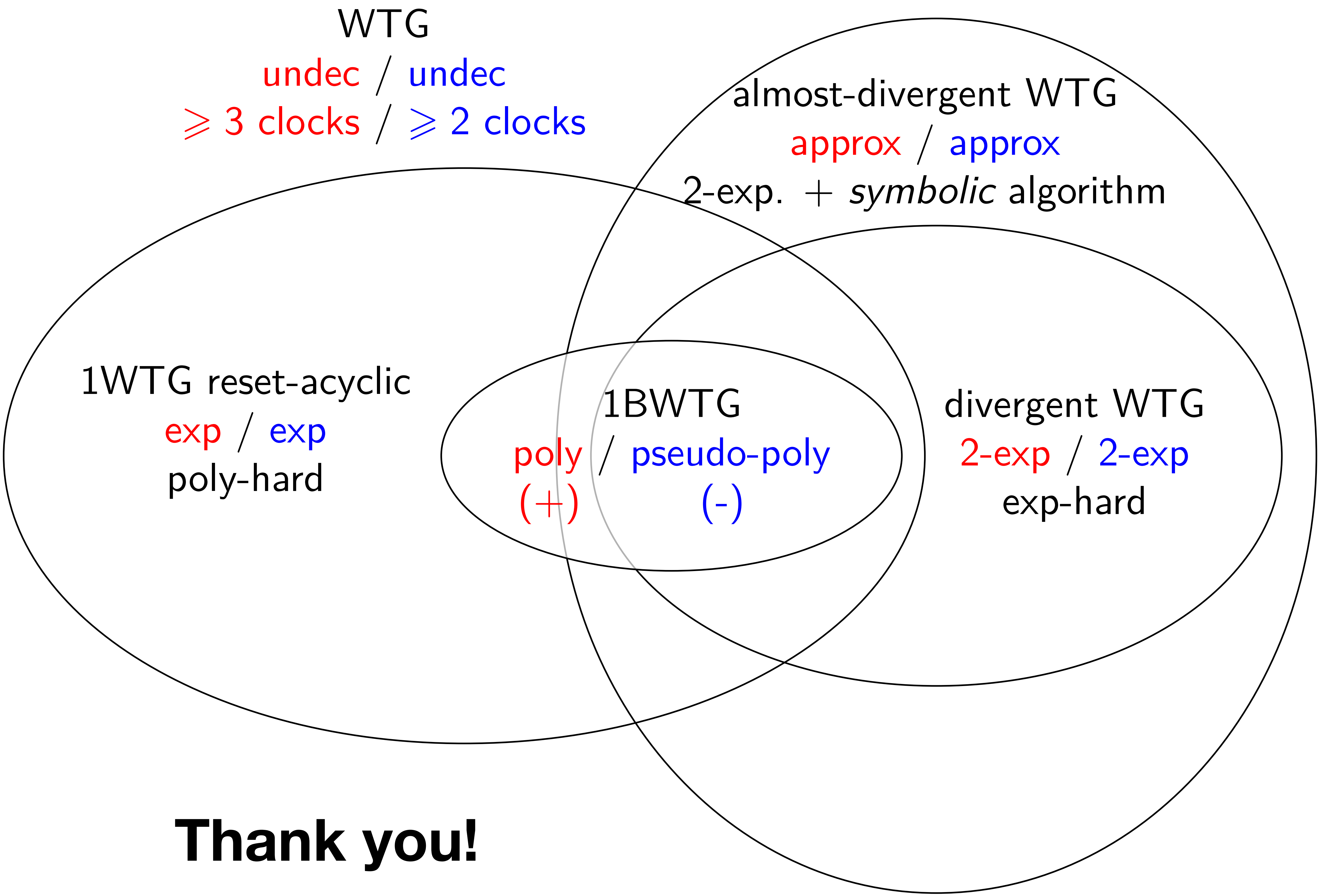
$$(s_1, 0) \xrightarrow{0.4, \searrow} (s_4, 0.4) \xrightarrow{0.6, \rightarrow} (s_5, 0) \xrightarrow{1.5, \leftarrow} (s_4, 0) \xrightarrow{1.1, \rightarrow} (s_5, 0) \xrightarrow{2, \nearrow} (\checkmark, 2)$$

$$\mathbf{1} \times 0.4 + \mathbf{1} \quad -\mathbf{3} \times 0.6 + \mathbf{0} \quad +\mathbf{1} \times 1.5 + \mathbf{0} \quad -\mathbf{3} \times 1.1 + \mathbf{0} \quad +\mathbf{1} \times 2 + \mathbf{2} \qquad = 1.8$$

$$(s_1, 0) \xrightarrow{0.2, \nearrow} (s_2, 0) \xrightarrow{0.9, \rightarrow} (s_3, 0.9) \xrightarrow{0.2, \circlearrowleft} (s_3, 0) \xrightarrow{0.9, \circlearrowleft} (s_3, 0) \quad \cdots$$

$$\mathbf{1} \times 0.2 + \mathbf{0} \quad +\mathbf{2} \times 0.9 + \mathbf{0} \quad -\mathbf{1} \times 0.2 + \mathbf{0} \quad -\mathbf{1} \times 0.9 + \mathbf{0} \quad \cdots \qquad = +\infty$$


WTG
undec / undec
⩾ 3 clocks / ⩾ 2 clocks
almost-divergent WTG
approx / approx
2-exp. + *symbolic* algorithm
1WTG reset-acyclic
exp / exp
poly-hard
1BWTG
poly / pseudo-poly
(+) (-)
divergent WTG
2-exp / 2-exp
exp-hard


WTG
undec / undec
$\geqslant 3$ clocks / $\geqslant 2$ clocks
almost-divergent WTG
approx / approx
2-exp. + *symbolic* algorithm
1WTG reset-acyclic
exp / exp
poly-hard
1BWTG
poly / pseudo-poly
(+) (-)
divergent WTG
2-exp / 2-exp
exp-hard

Thank you!