Mean-payoff Games: Extensions and Variations

LCCC Workshop
Lund

J.-F. Raskin
Université Libre de Bruxelles, U.L.B.

based on joint works with K. Chatterjee (IST Austria), L. Doyen (ENS Cachan)
T. Henzinger (IST Austria), A. Rabinovitch (U Tel Aviv), M. Randour (U Mons) and Y. Velner (U Tel Aviv)

Plan

- **Synthesis** and 2 player zero-sum **games** on graphs
- Mean-payoff games (as an example of quantitative games):
  - **Mean-payoff games** in 1 dimension
  - **Extensions:** k dimensions - recent results
  - **Variations: window objectives**
    - Motivations and definitions
    - Algorithm for 1 dimension
    - Overview of results
- Conclusion

Games for Synthesis (of Reactive Systems)

☞ support the design process with **automatic synthesis**

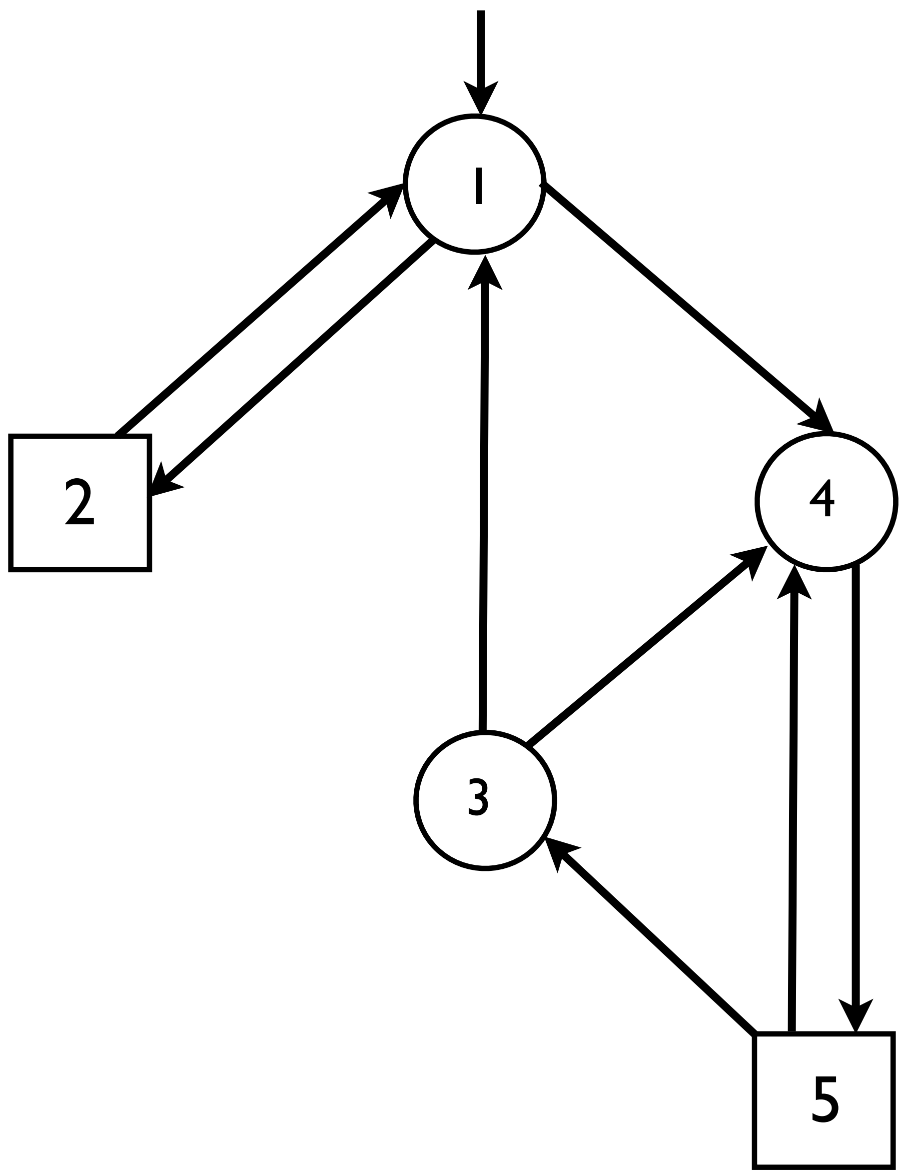
$$\text{Env} \;||\; \textbf{?} \;\models\; \psi$$

- ➢Sys is constructed by an algorithm
- ➢Sys is **correct** by construction
- ➢Underlying theory: **2-player zero-sum games**
- ➢Env is **adversarial** (worst-case assumption)

Winning strategy = Correct Sys

Preliminaries:
2-player Zero-sum Games on Graphs

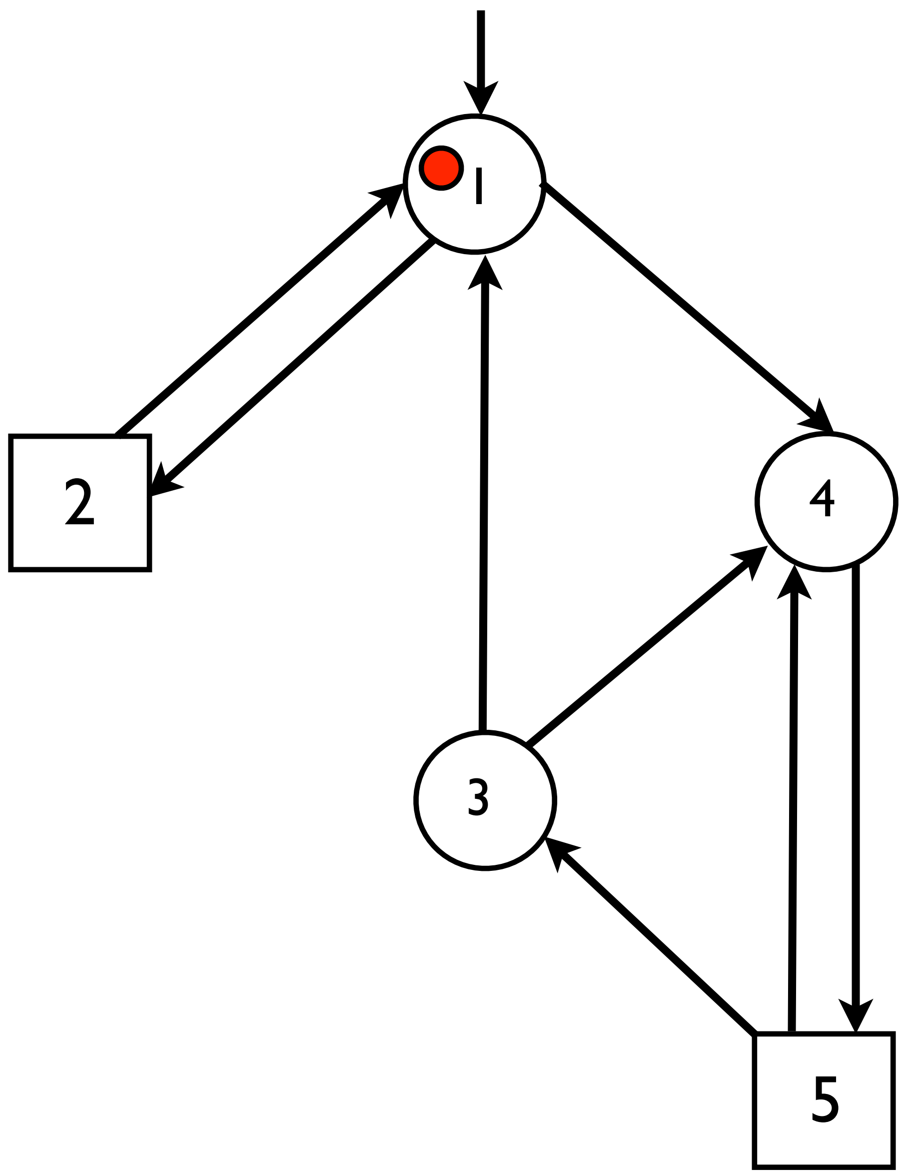
2-player Zero-sum Games on Graphs



(Finite) directed graph

Two types of vertices
(Player 1 and Player 2 vertices)

2-player Zero-sum Games on Graphs



How to play ?

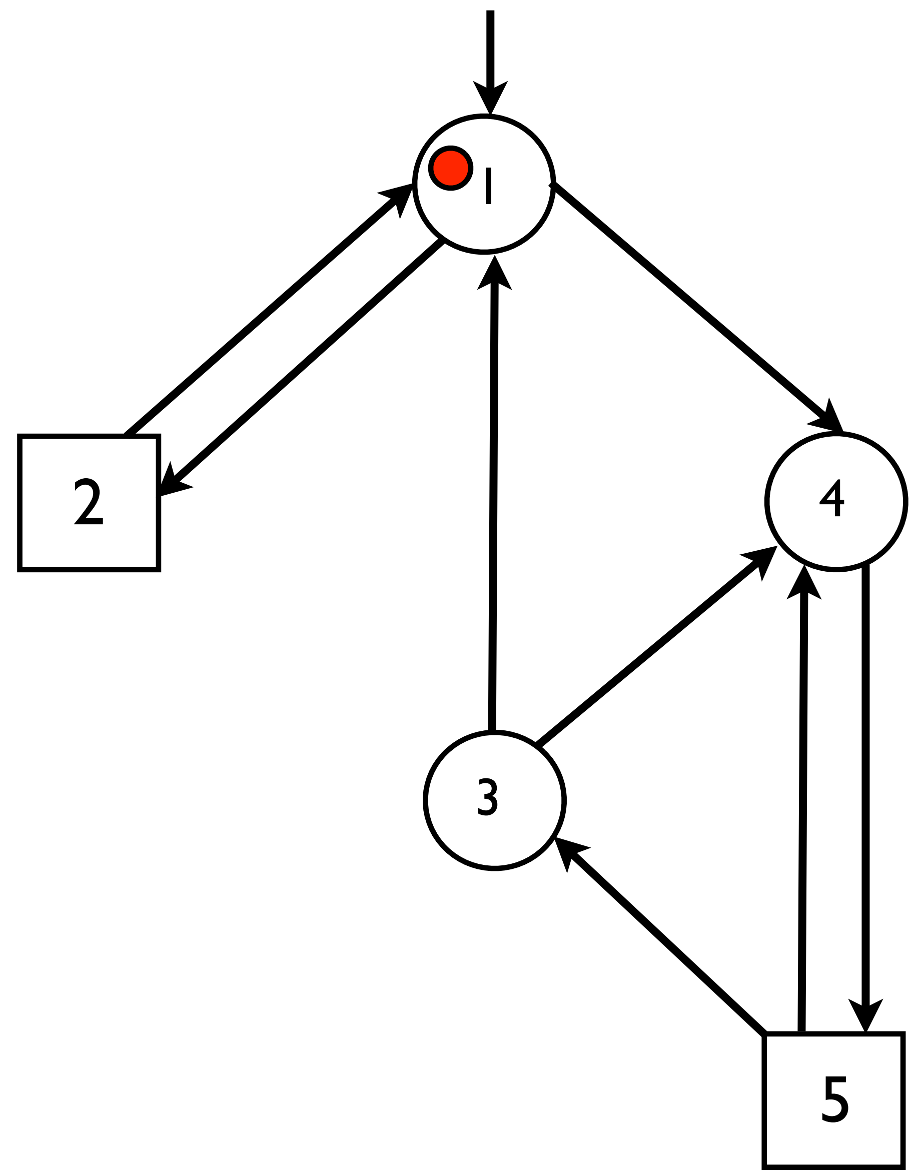
One token is placed on initial vertex

Players play for an infinite number of rounds:

- in each round: the player that owns the vertex moves the token to an adjacent vertex

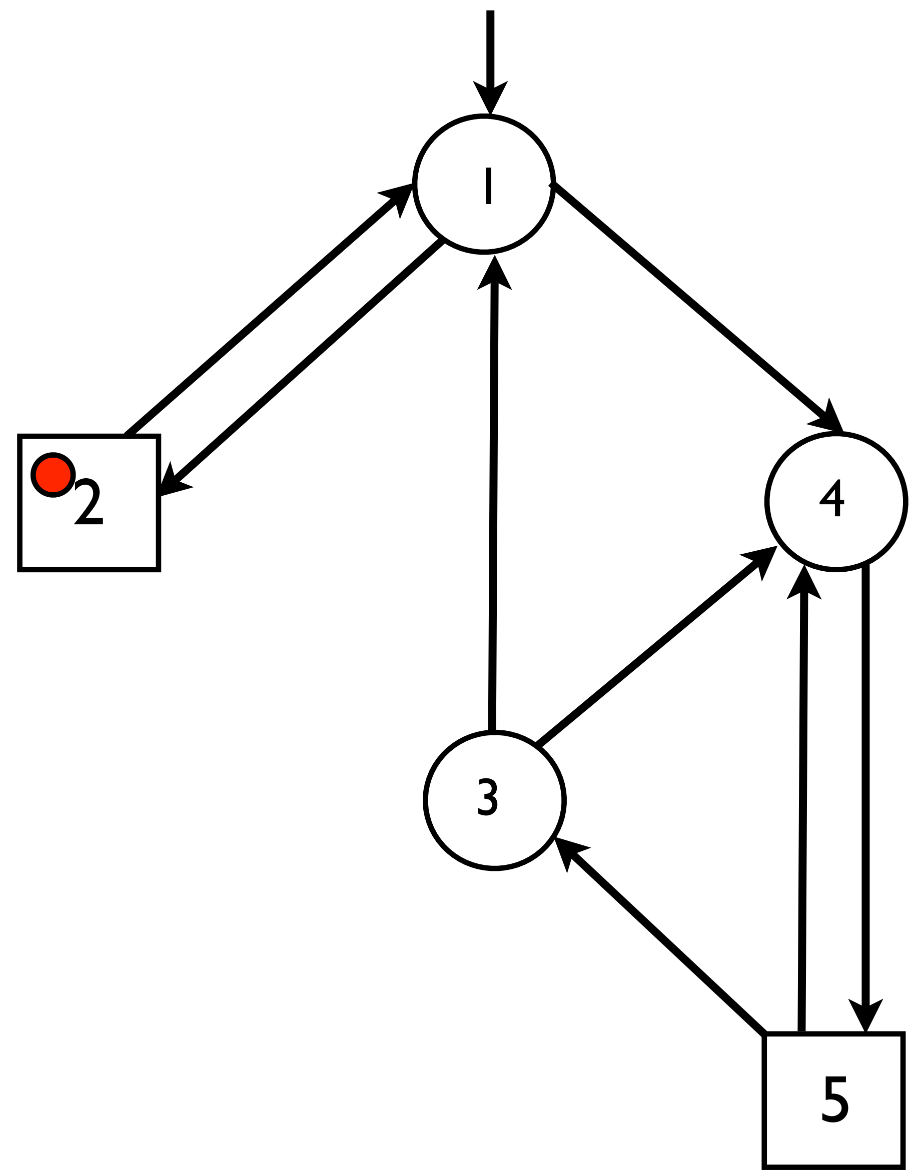
Outcome=infinite path

2-player Zero-sum Games on Graphs



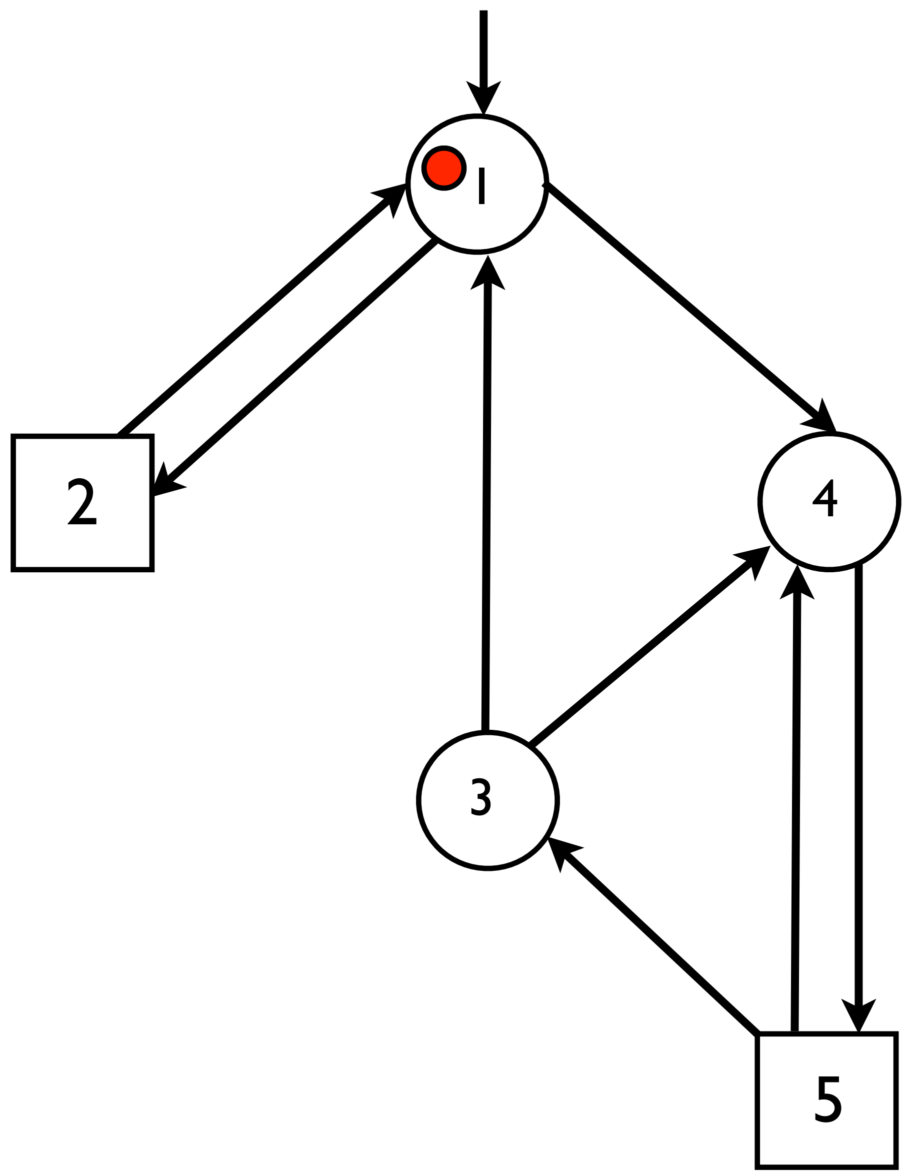
1

2-player Zero-sum Games on Graphs



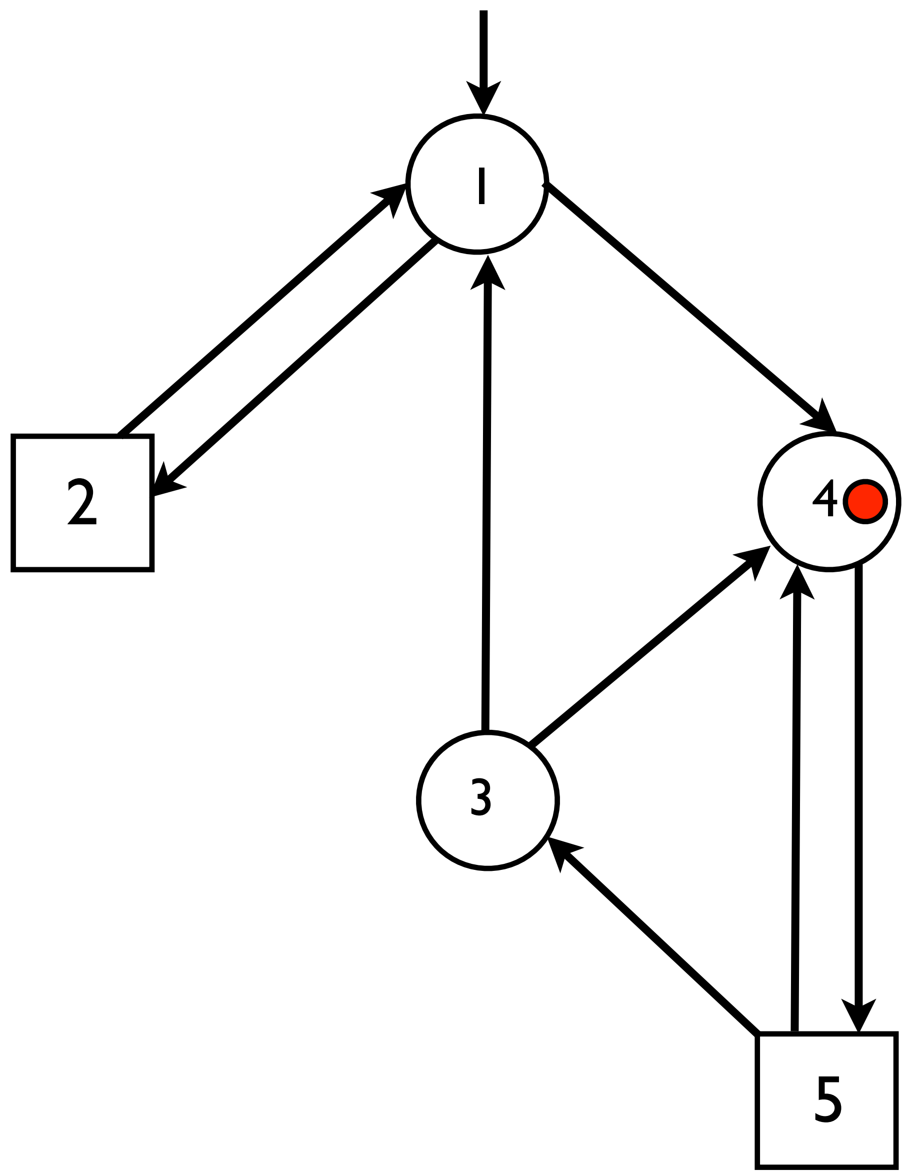
1 → 2

2-player Zero-sum Games on Graphs



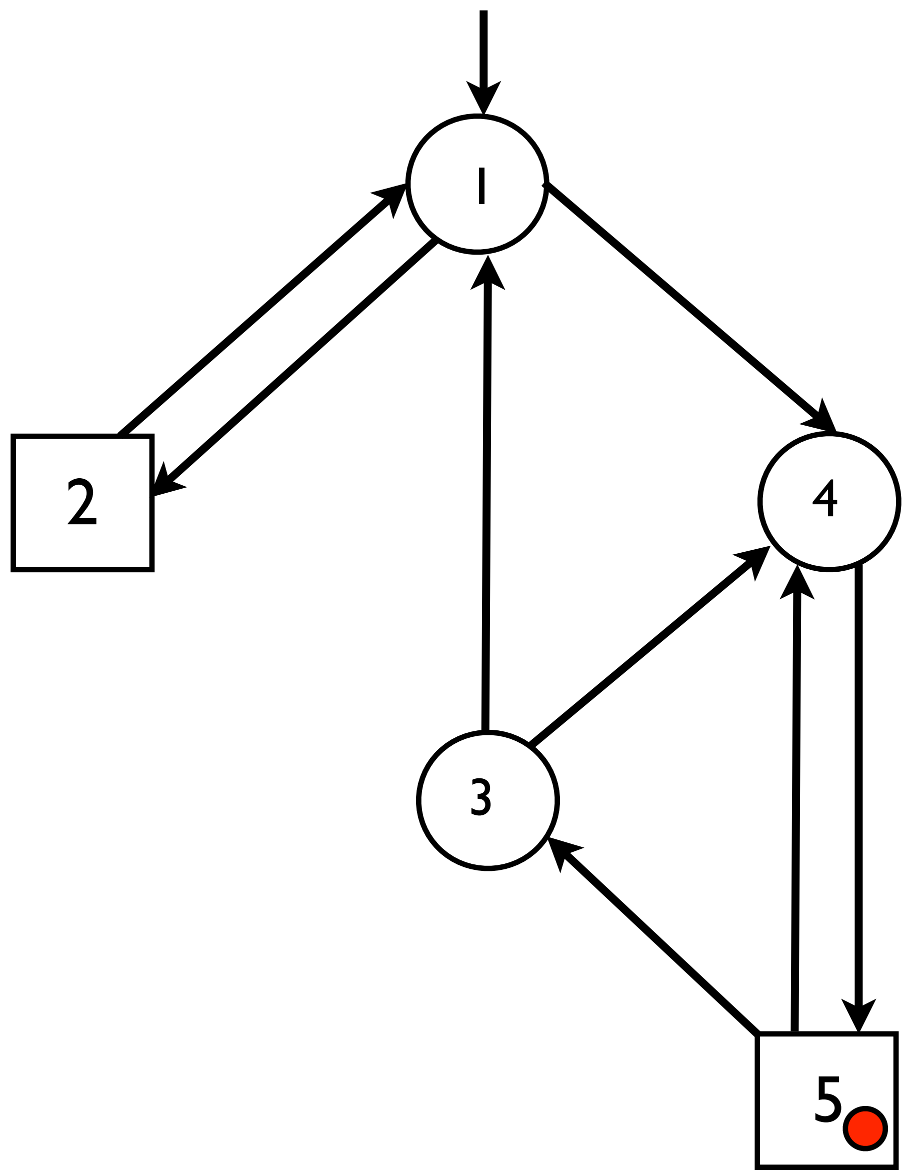
1 → 2 → 1

2-player Zero-sum Games on Graphs



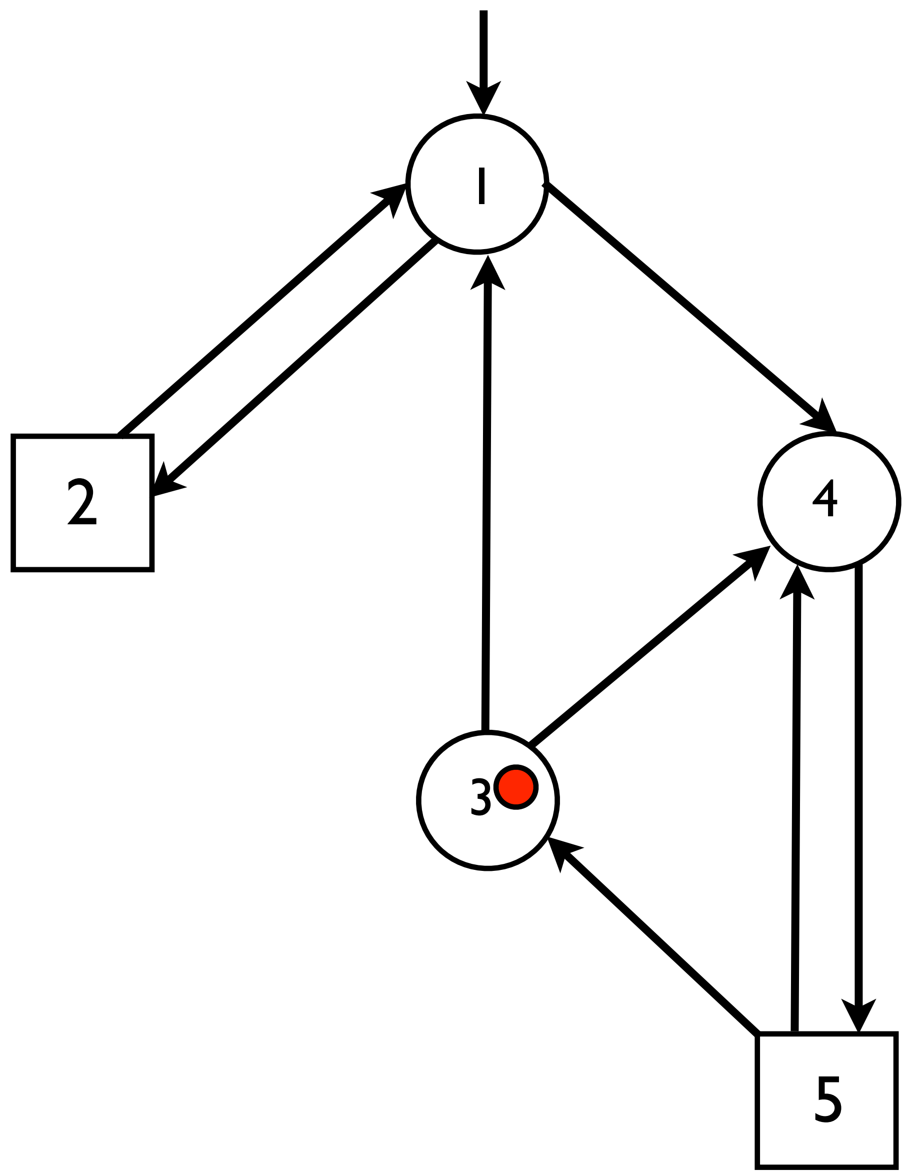
1 → 2 → 1 → 4

2-player Zero-sum Games on Graphs



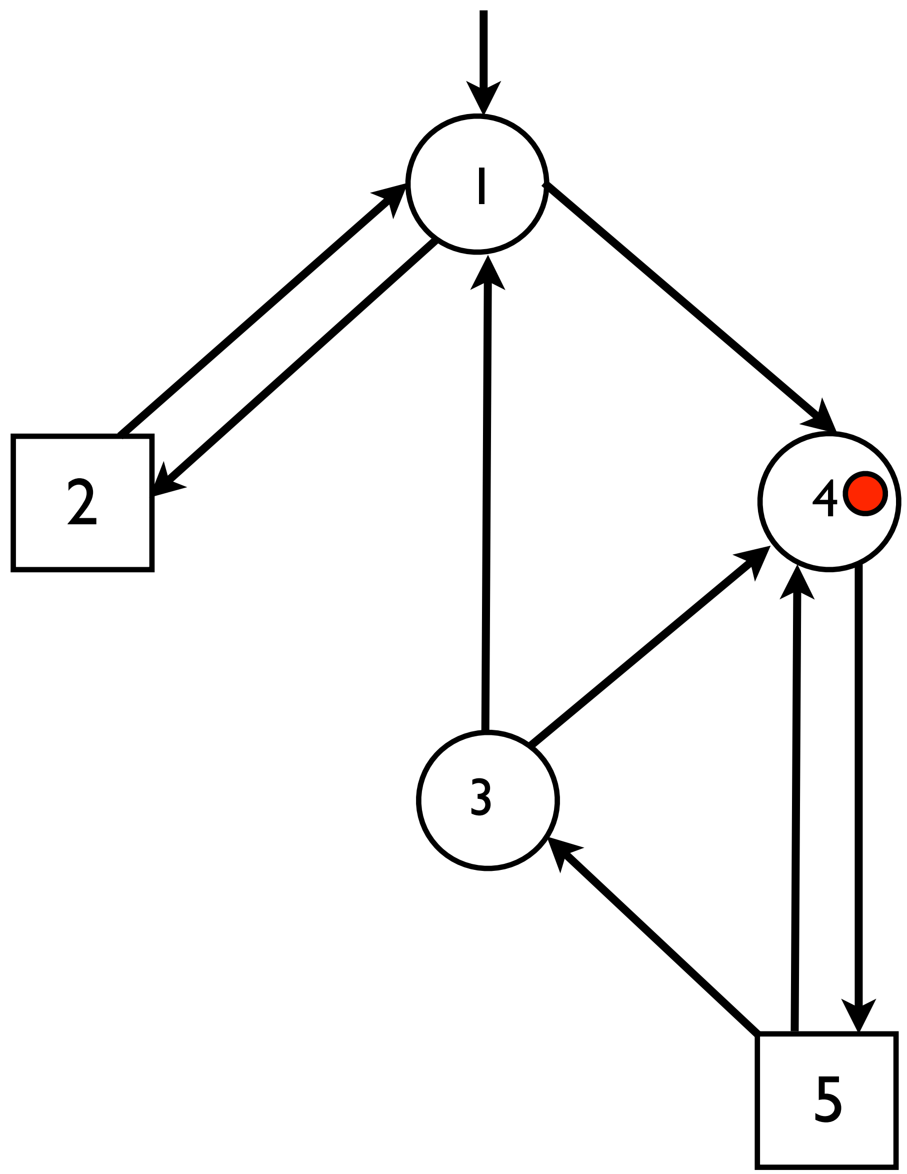
1 → 2 → 1 → 4 → 5

2-player Zero-sum Games on Graphs



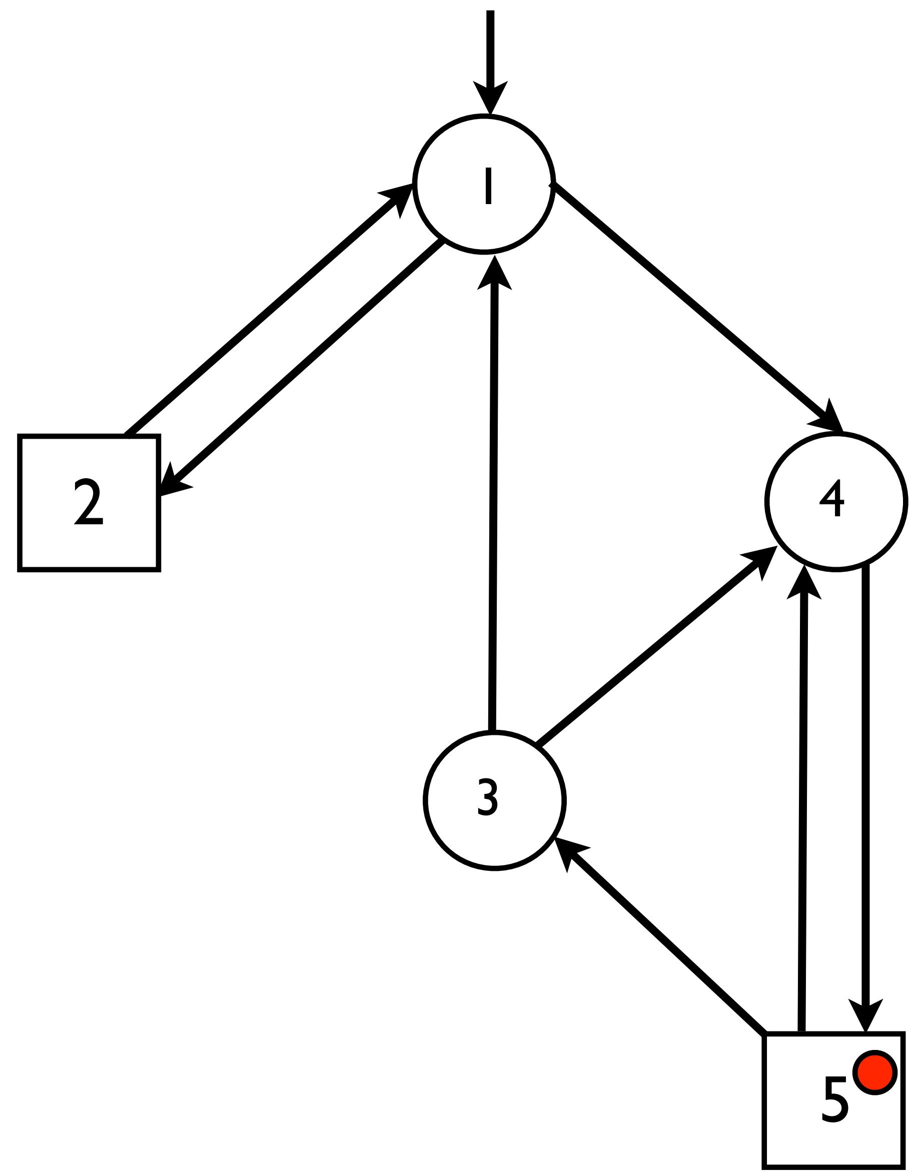
$1 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 3$

2-player Zero-sum Games on Graphs



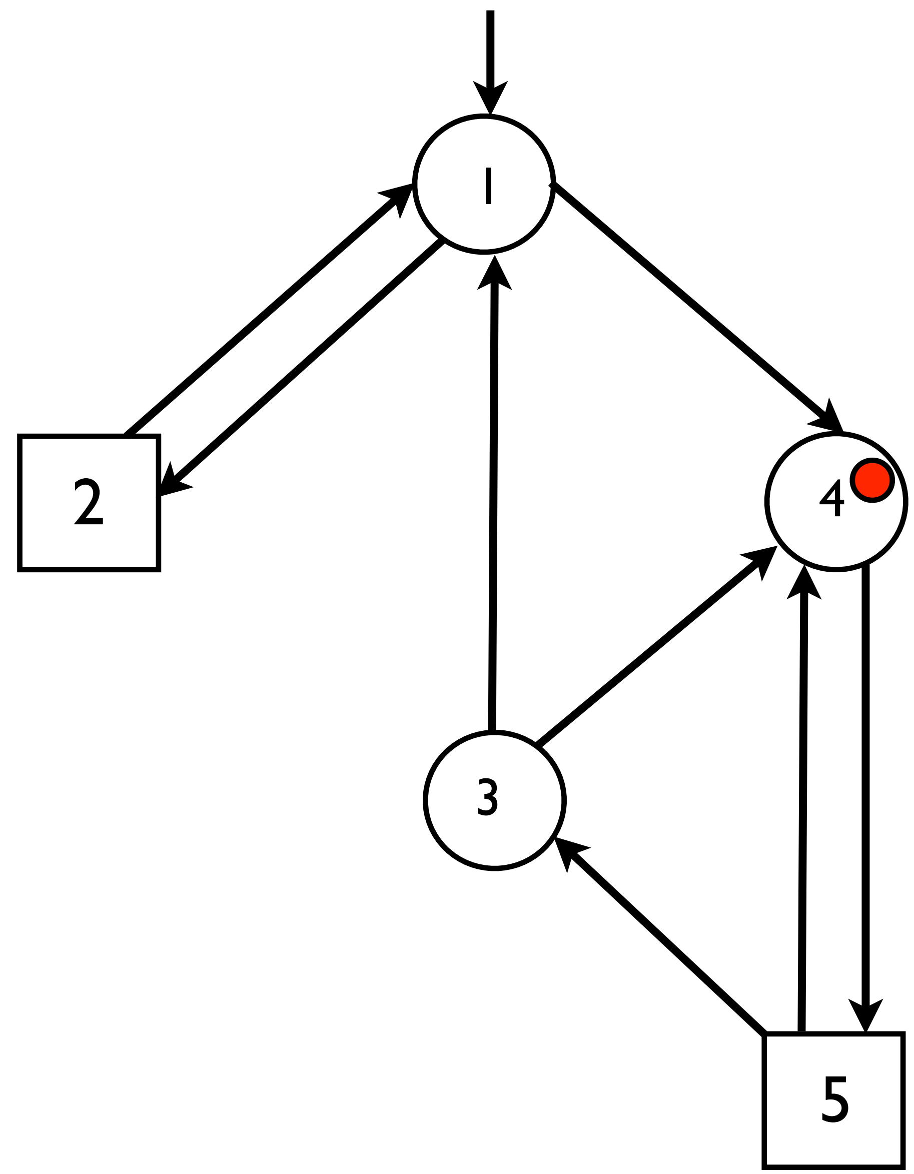
$1 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 3 \rightarrow 4$

2-player Zero-sum Games on Graphs



$1 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 3 \rightarrow 4 \rightarrow 5$

2-player Zero-sum Games on Graphs



$1 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 4 \rightarrow \ldots$

2-player Zero-sum Games on Graphs

Who is winning ? winning conditions

☛ $Win_1 \subseteq V^\omega$: Set of good outcomes (paths) for Player 1

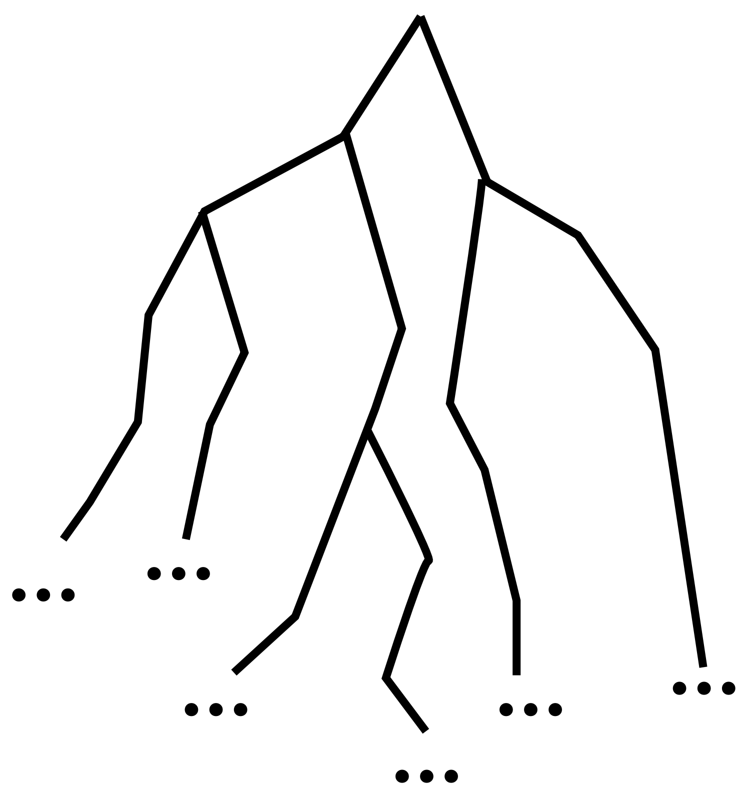
☛ $Win_2 = V^\omega \setminus Win_1$ (zero sum)

Examples of winning conditions:

- $Win_1 = \{ \pi \mid \pi$ visits Good $\}$
  Reachability winning condition
- $Win_1 = \{ \pi \mid \pi$ visits Good infinitely often $\}$
  Büchi winning condition

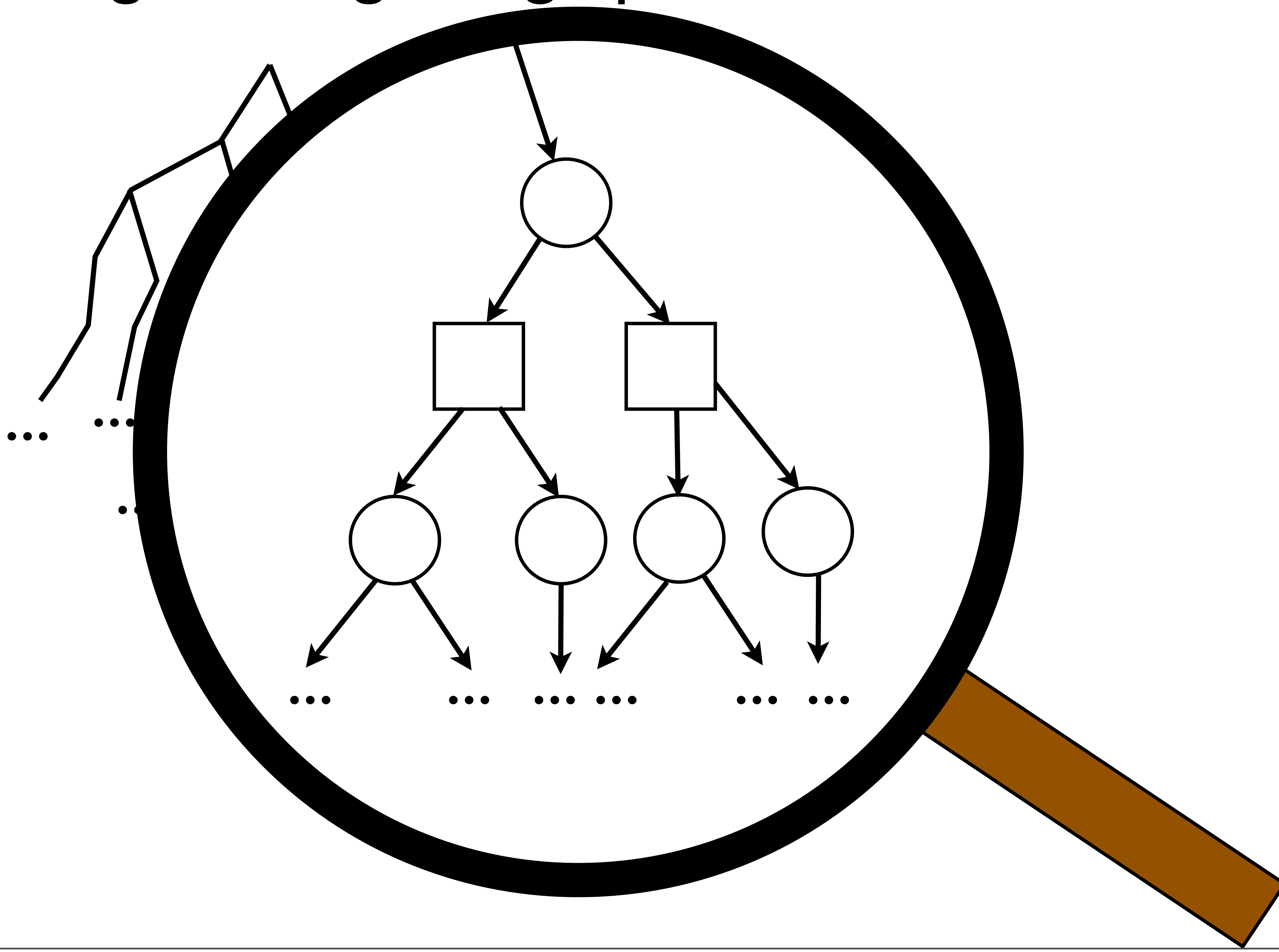
Strategies

Unfolding of the game graph



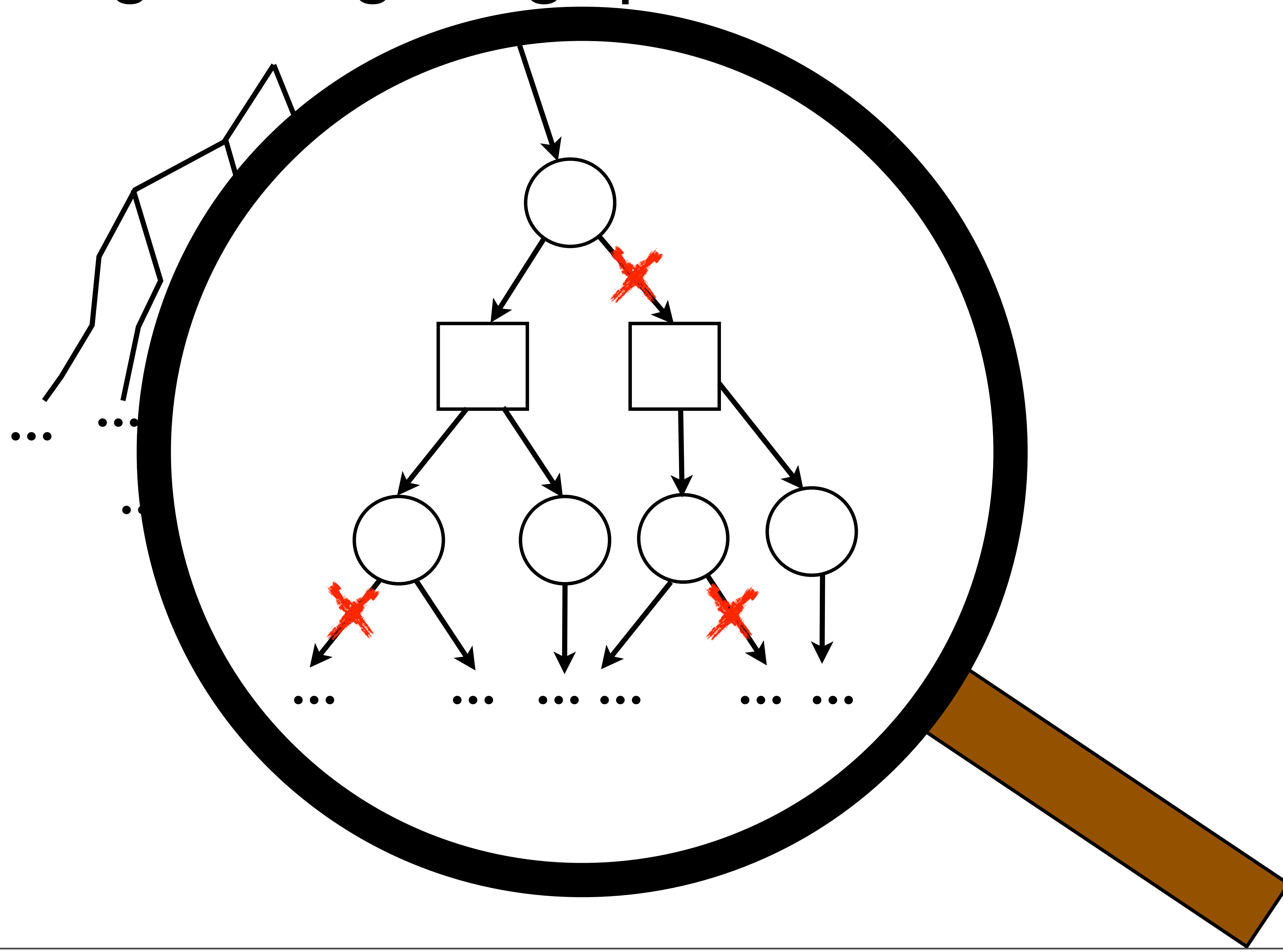
Strategies

Unfolding of the game graph

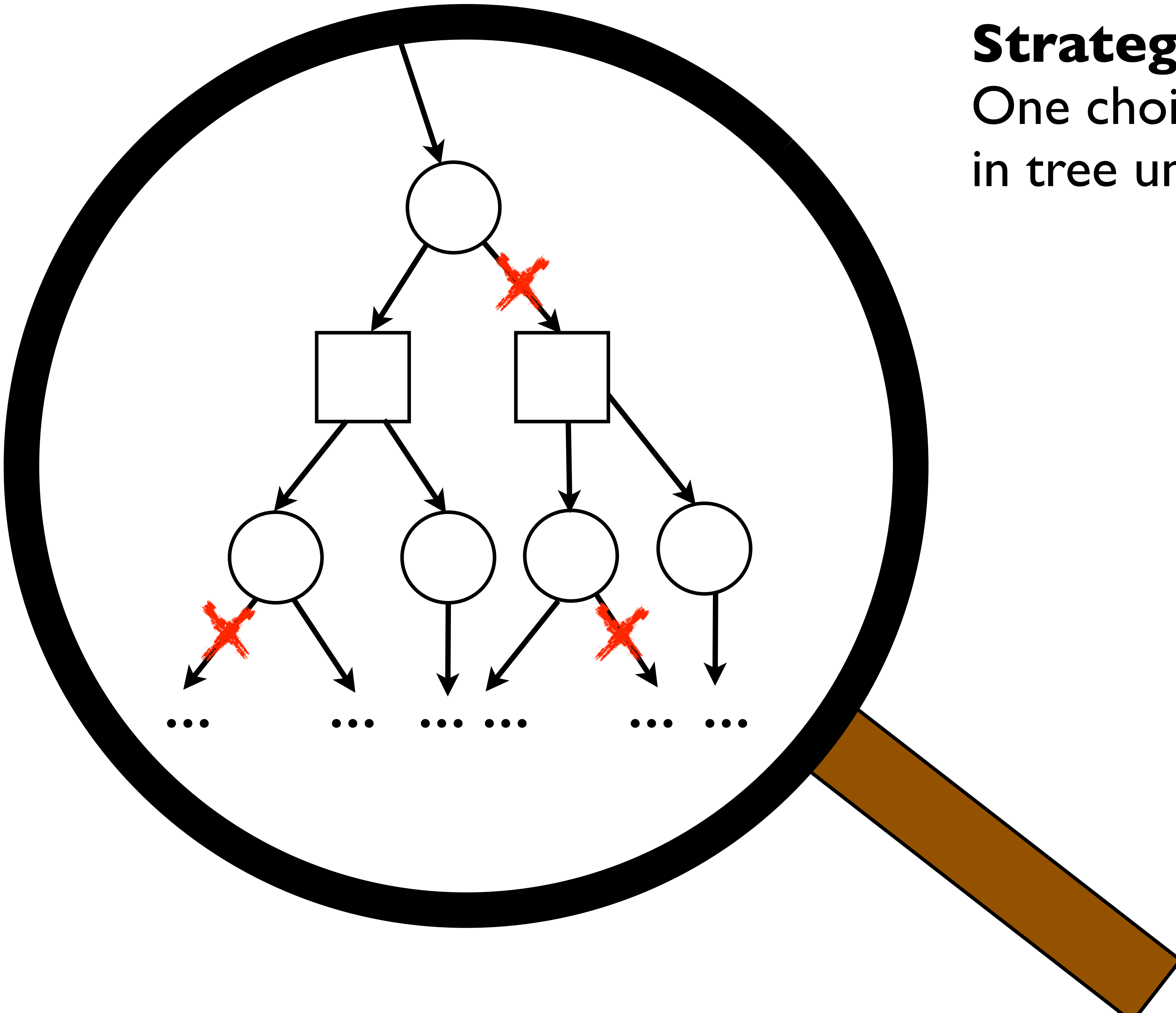


Strategies

Unfolding of the game graph

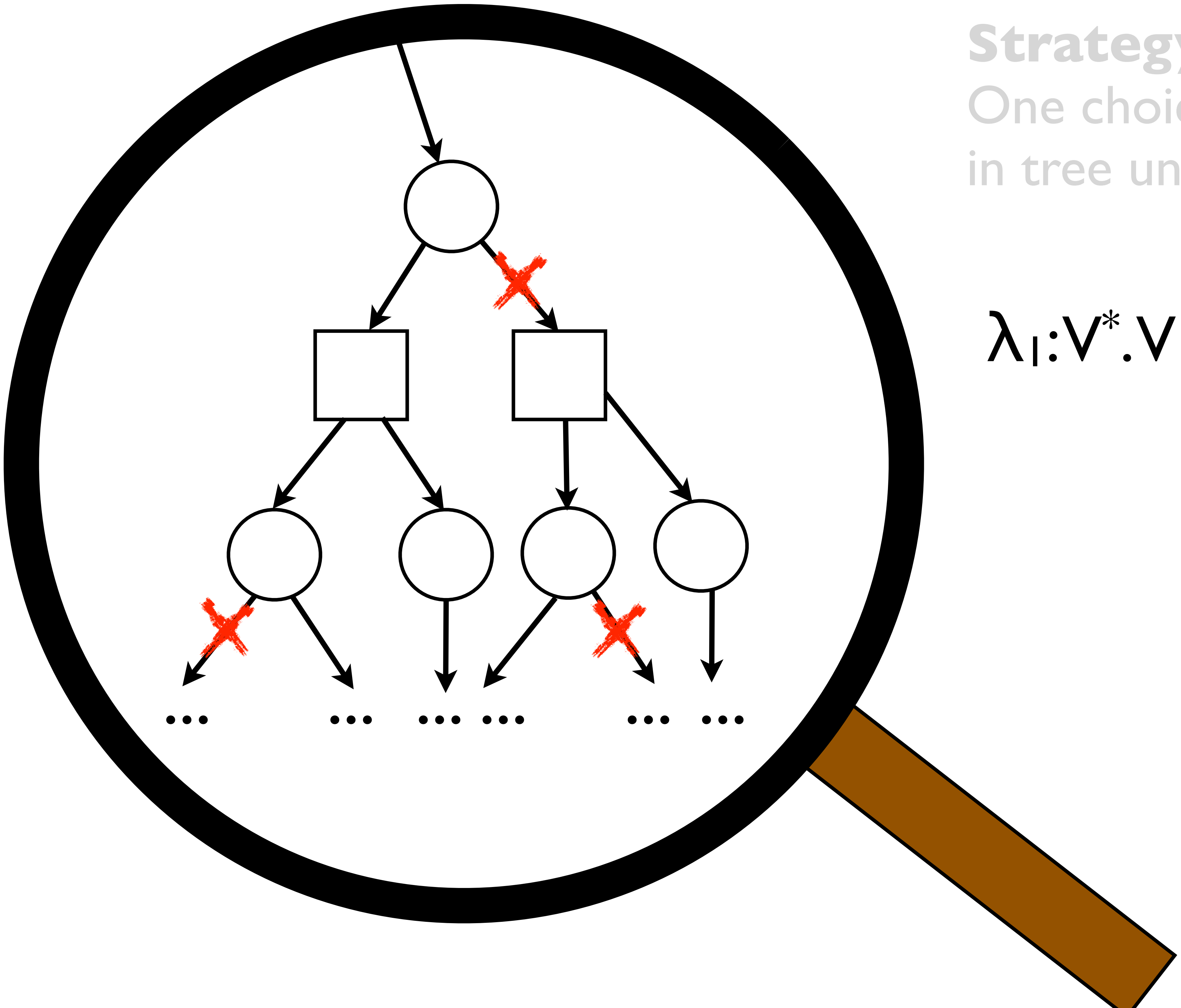


Strategies



Strategy for Player 1 =
One choice in each node of Player 1 in tree unfolding

Strategies



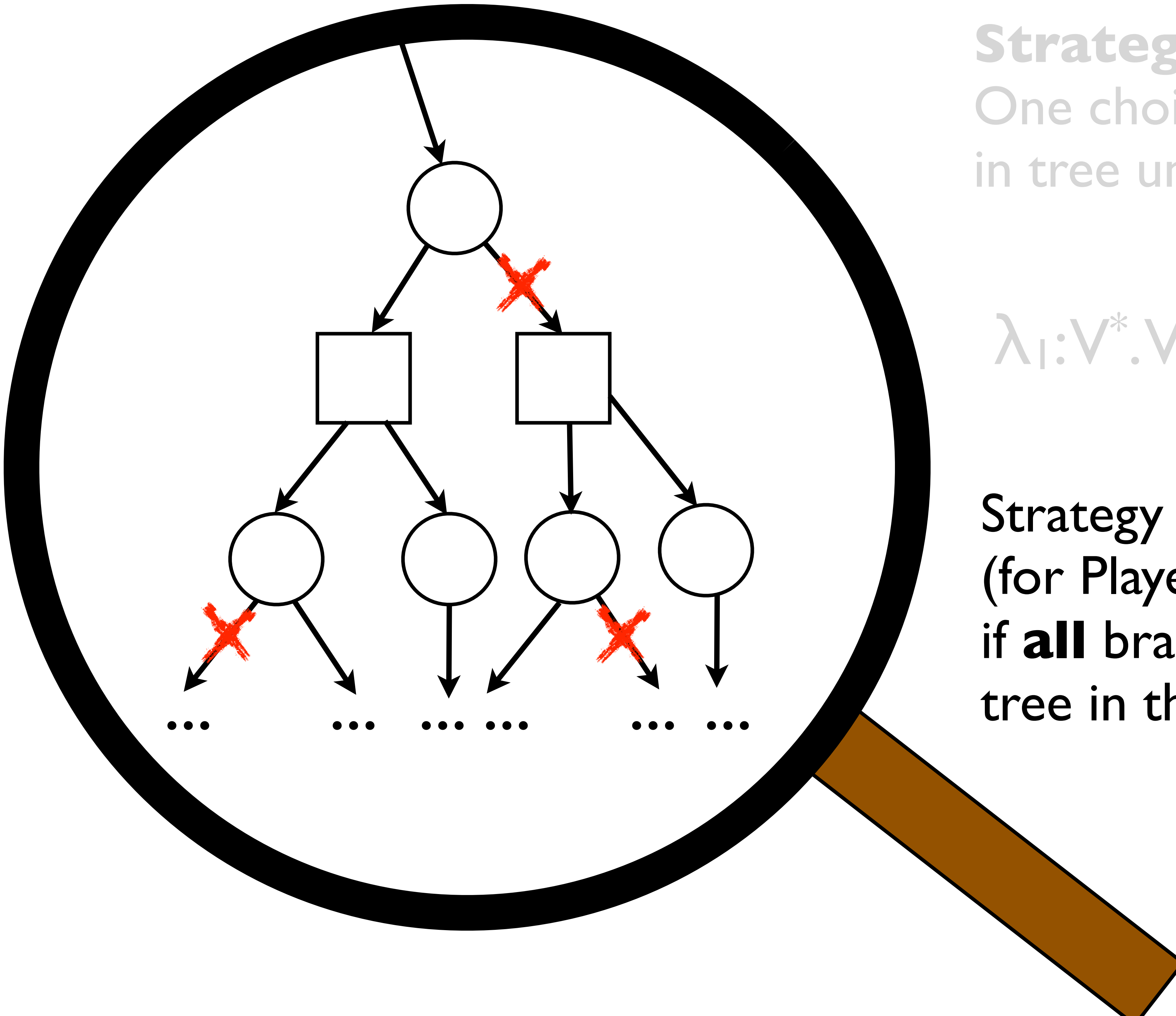
Strategy for Player I =
One choice in each node of Player I in tree unfolding

$\lambda_I : V^*.V_I \rightarrow \text{edge}$

Strategies

Strategy for Player I =
One choice in each node of Player I in tree unfolding

$\lambda_I : V^*.V_I \to \text{edge}$

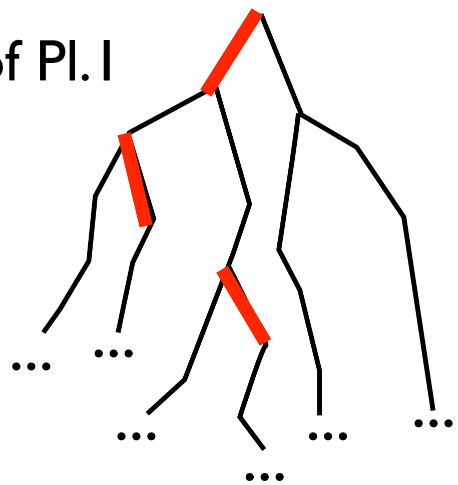


Strategy is **winning** (for Player I), if **all** branches of the resulting tree in the winning condition

Types of strategies

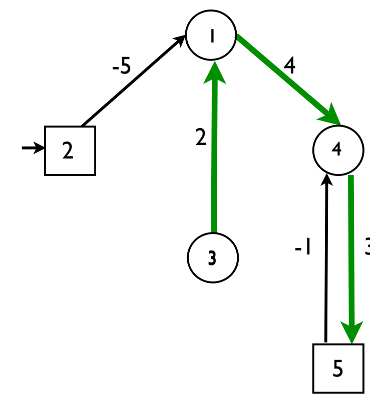
(Player 1) strategy:

$\lambda_1 : V^*.V_1 \rightarrow \text{edge}$.
Σ_1=set of strategies of Pl.1



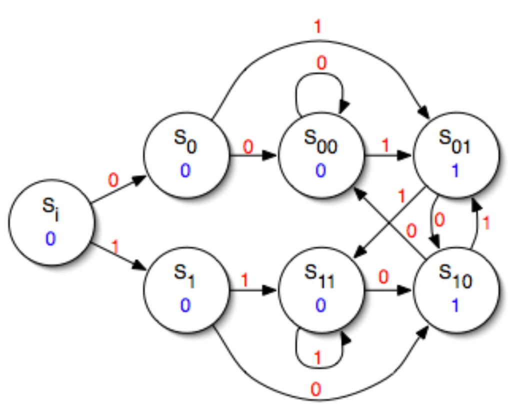
Memoryless strategy:

$\lambda_{1,\mathbf{m}} : V_1 \rightarrow \text{edge}$.
$\Sigma_{1,\mathbf{m}}$=set of memoryless strategies of Player1



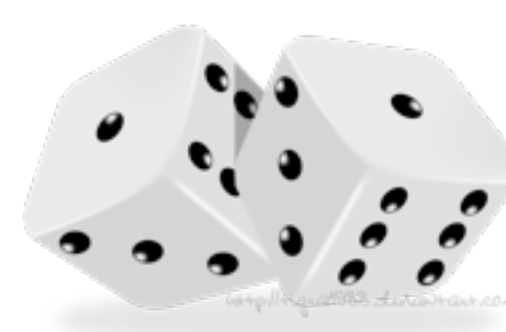
Finite-memory strategy:

$\lambda_{1,\mathbf{f}} : V^*.V_1 \rightarrow \text{edge}$ but regular (Moore machine)
$\Sigma_{1,\mathbf{f}}$=set of finite memory strategies of Player 1



Randomized strategy:

$\lambda_{1,\mathbf{m}} : V^*.V_1 \rightarrow$ **Dist**(edge).
$\Sigma_{1,\mathbf{m}}$=set of randomized strategies of Player1

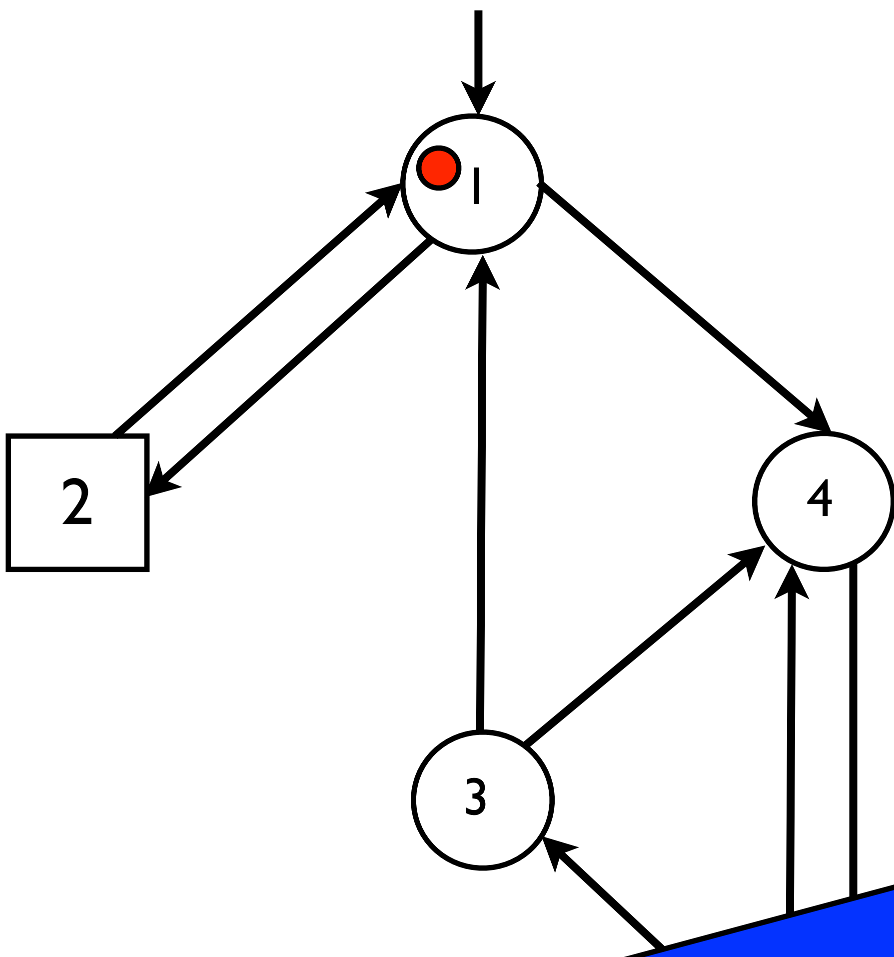


Decision Problem

Given:

- a game graph G
- a winning condition $Win_1 \subseteq V^\omega$
  - **decide** if Player 1 has a **winning strategy**
  - **determinacy**:
    either Player 1 has a winning for Win_1
    or Player 2 has a winning strategy for $Win_2 = V^\omega \setminus Win_1$
    It is true for a large class of objectives,
    e.g. ω-regular objectives

2-player Zero-sum Games on Graphs



One token is placed on initial vertex

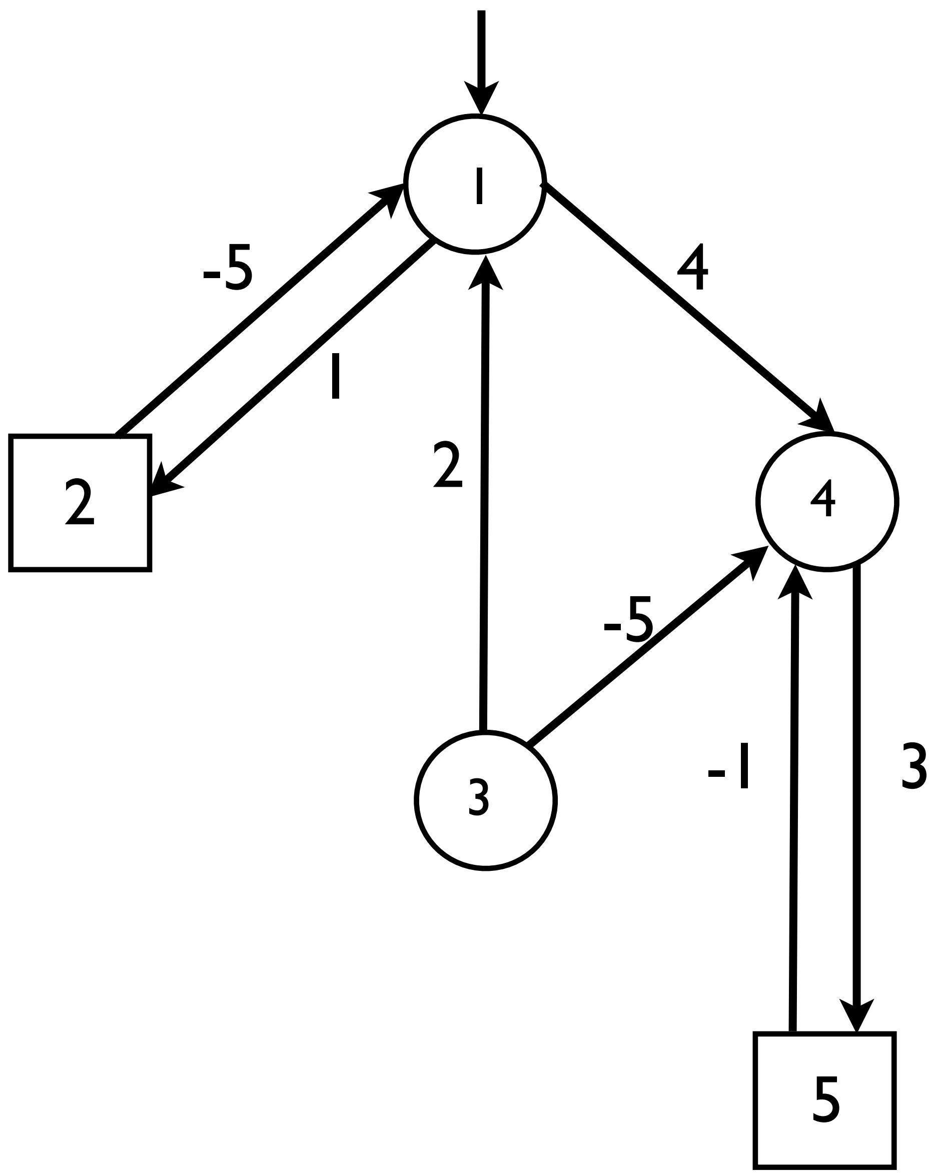
We play an infinite number of rounds:

- in each round: the player that owns the vert[...] the tok[...]

Classical model for the synthesis of reactive systems:
Player I=system and Player II=environment
For embedded systems, we need **quantities** !

Quantitative Objectives: Mean-Payoff Games

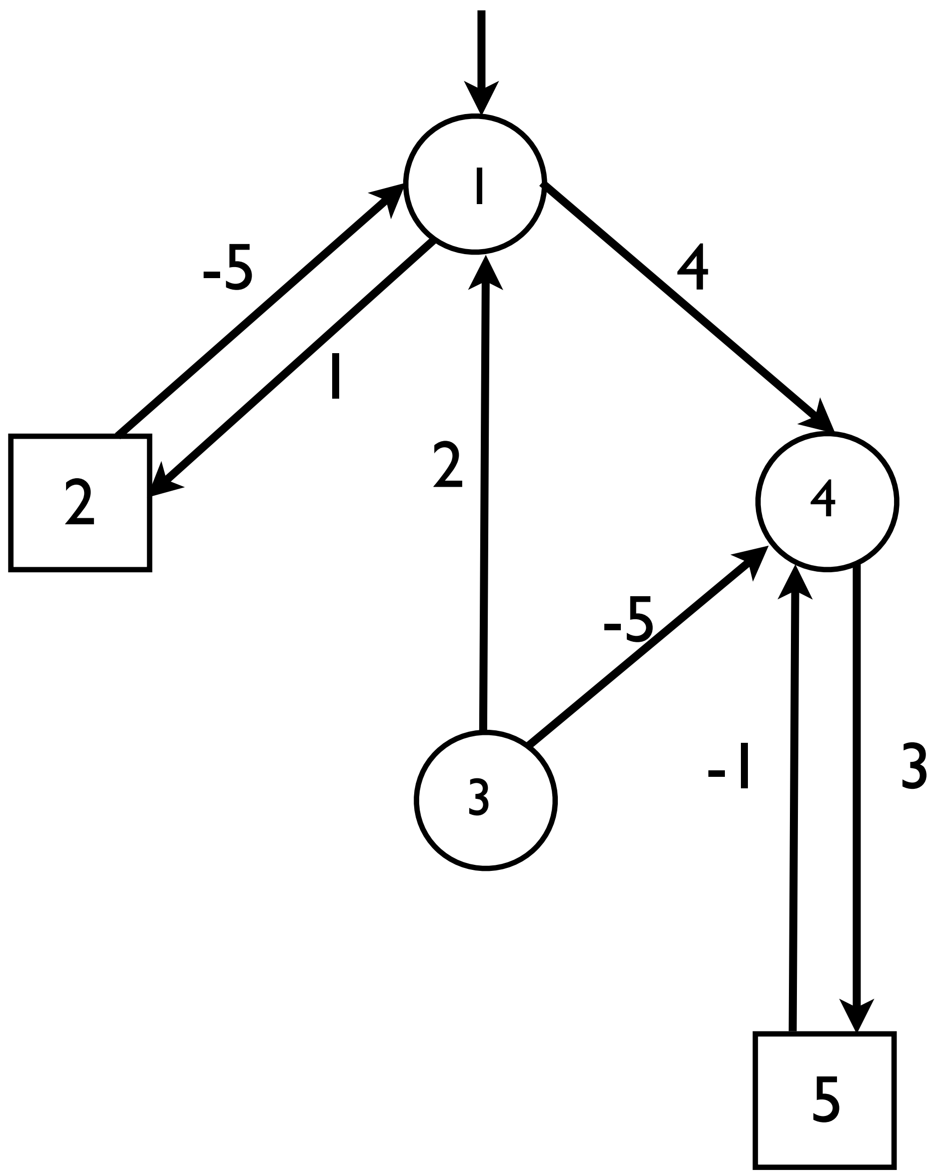
Mean-Payoff Games [EM79]



defined on

weighted directed graphs

Mean-Payoff Games [EM79]



○ : positions of **maximizer=system**

□ : positions of **minimizer=environment**

Edges are labelled with rewards

$$\begin{array}{ccccccc} (1,4) & (4,5) & (5,4) & \ldots & (4,5) & (5,4) & \ldots\ldots \\ 4 & 3 & -1 & & 3 & -1 & \ldots\ldots \end{array} \quad = \text{play}$$

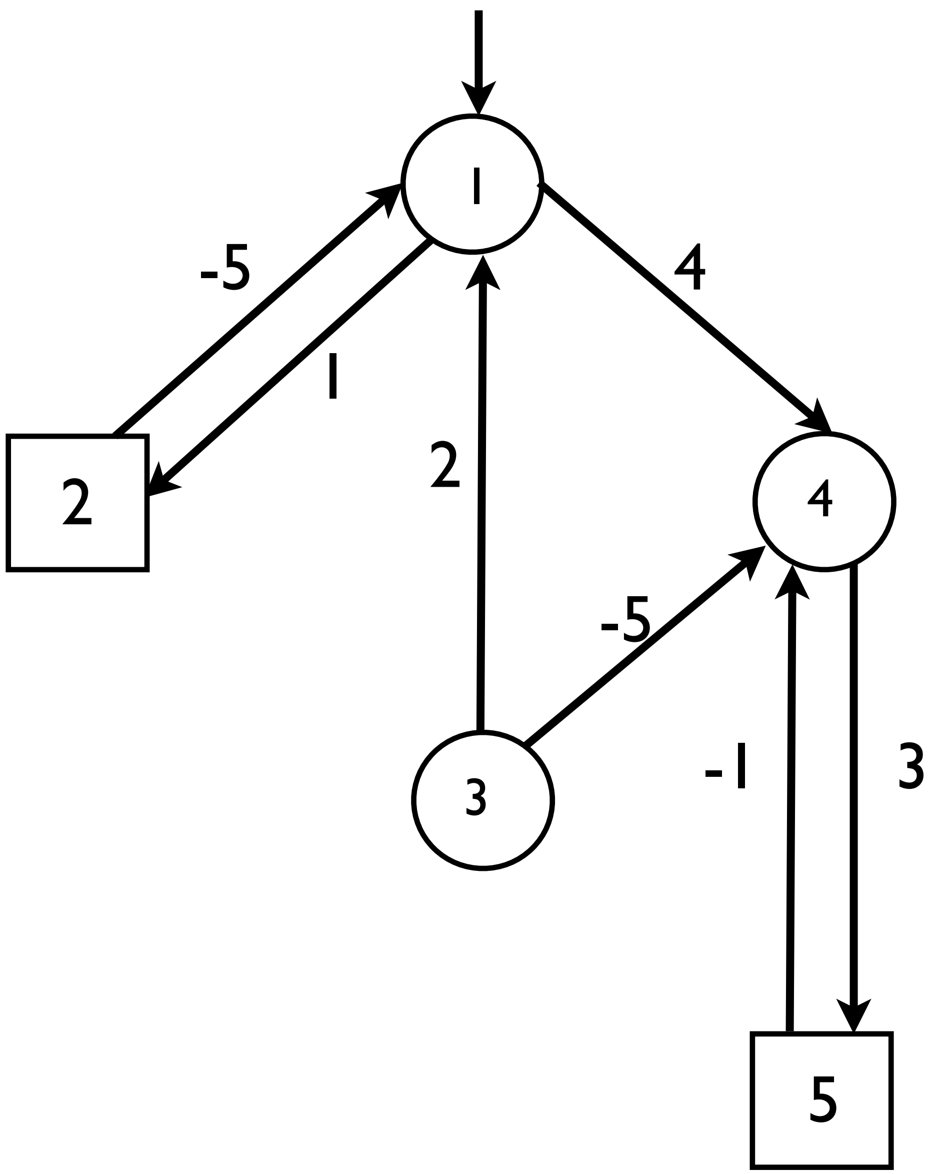
$= \mathbf{Lim\ Sup}_{n\to+\infty} \sum_{i=1,i=n} r_i / n$

$= \mathbf{MP}((1,4)\ (4,5)\ (5,4)\ \ldots\ (4,5)\ (5,4)\ \ldots\ldots) = 1$

Win={ play | MP(play)≥**c**}

Note: **not** ω-regular.

Mean-Payoff Games [EM79]

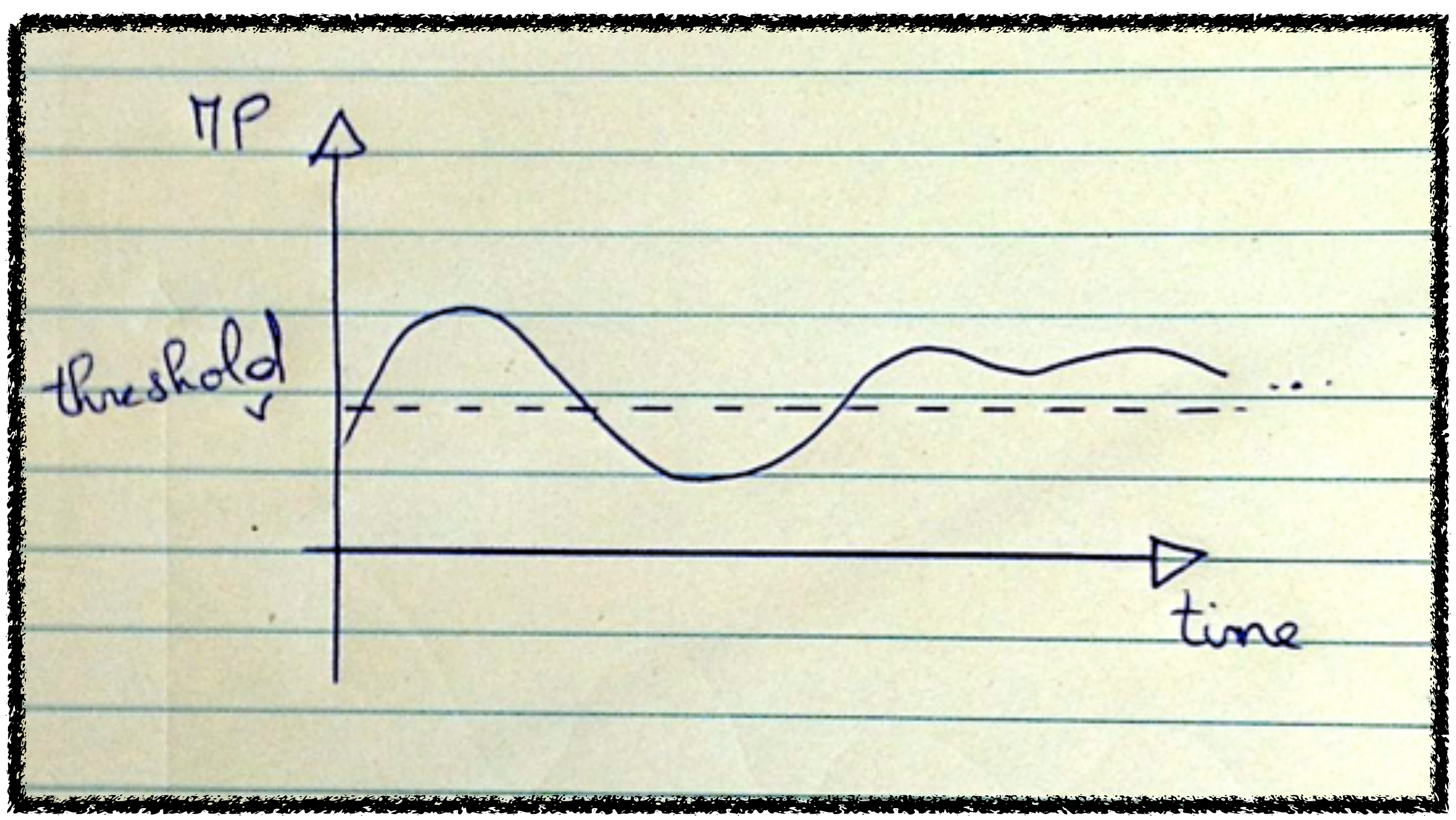


: positions of **maximizer=system**

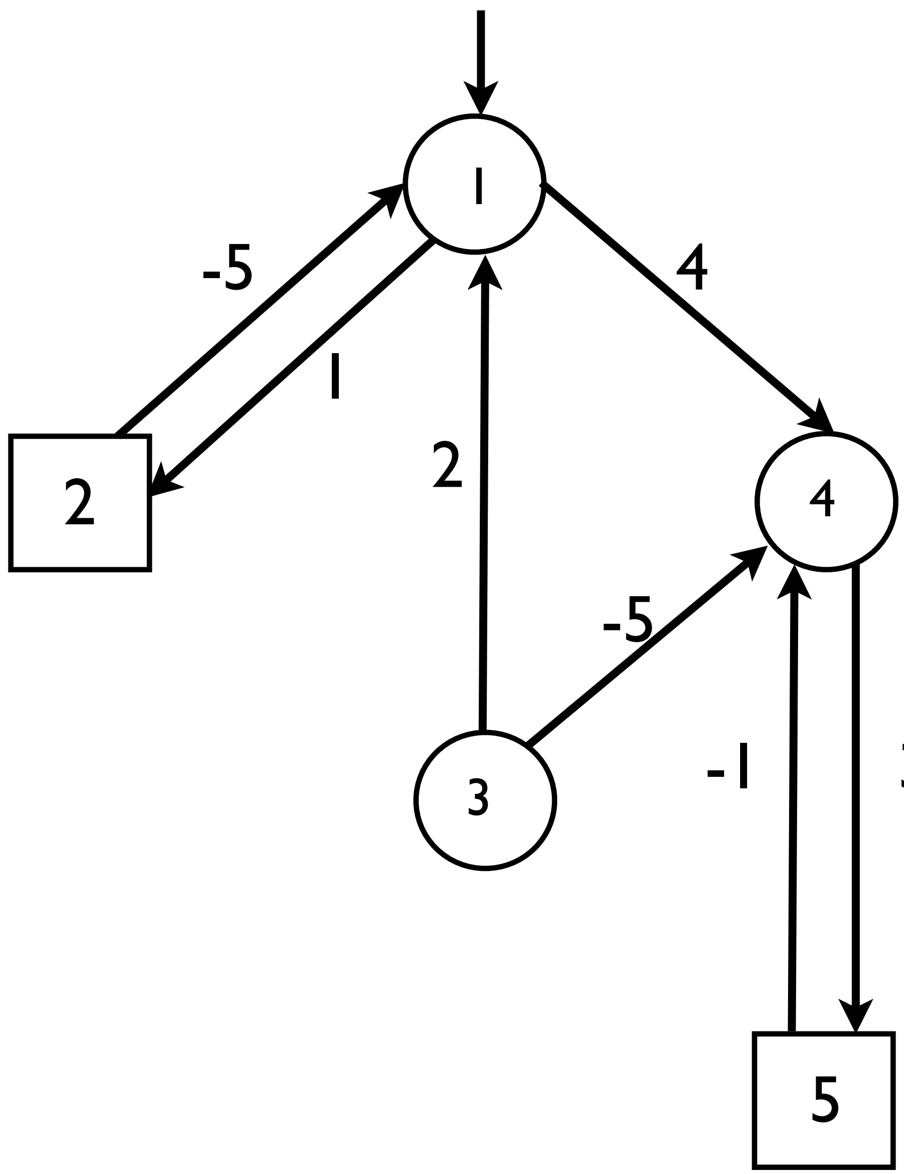
: positions of **minimizer=environment**

Edges are labelled with rewards

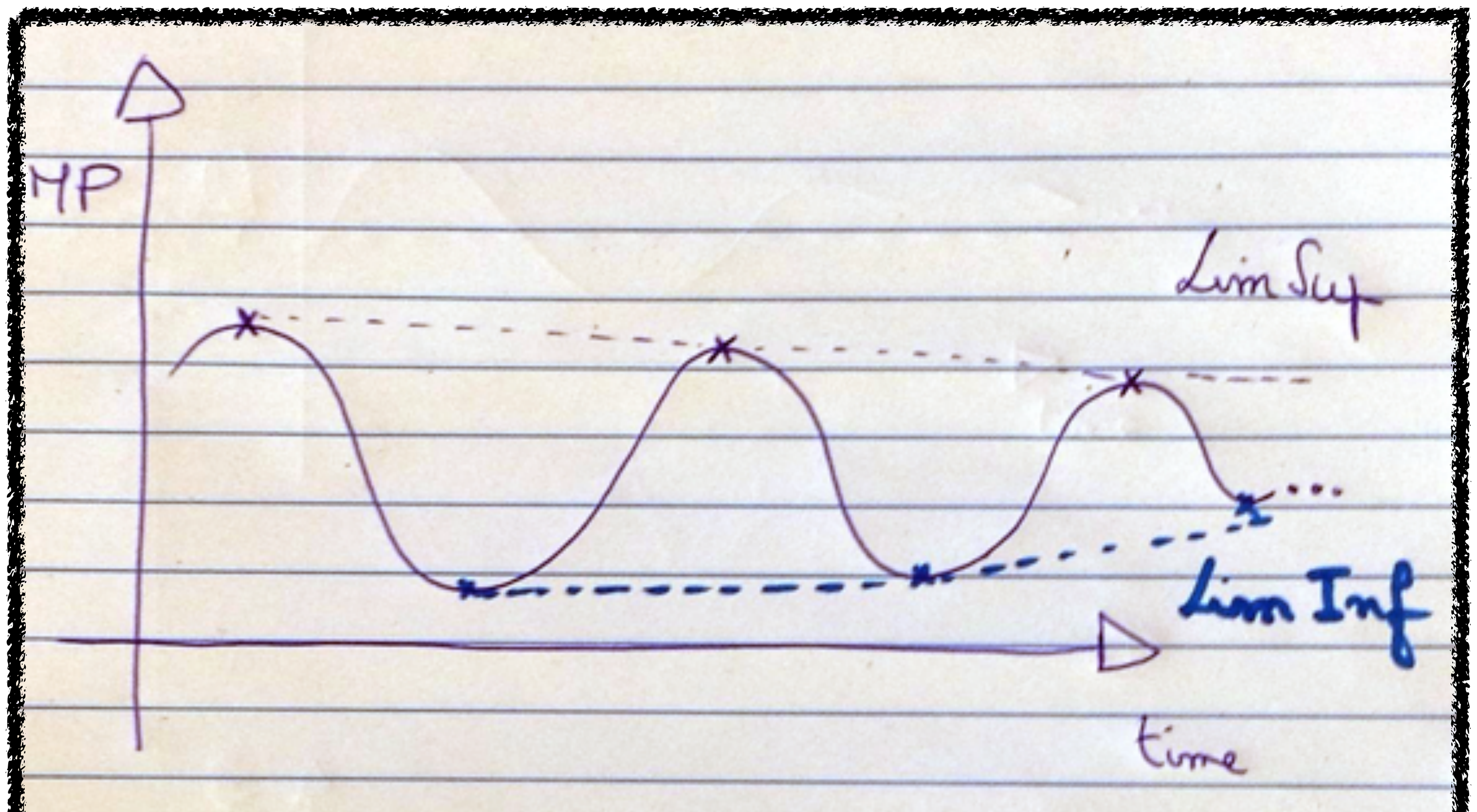
(1,4) (4,5) (5,4) ... (4,5) (5,4) .... =play
4 3 -1 3 -1 ....



Mean-Payoff Games [EM79]



Lim Sup - Lim Inf do not define the same set of plays



Mean-payoff Games

Theorem [EM79,Jur98,ZP97,GZ09]

(*i*) *In mean-payoff games, the two players can play optimally with* ***memoryless strategies***

(*ii*) *The winner can be decided in* **NP∩coNP** *and in* ***pseudo-polynomial*** *time*

Rem: Those results hold no matter if lim sup/lim inf is used in the definition and the winner is the same for the two definitions

Mean-payoff Games

Theorem [EM79,Jur98,ZP97,GZ09]

(i) *In mean-payoff games, the two players can play optimally with **memoryless strategies***

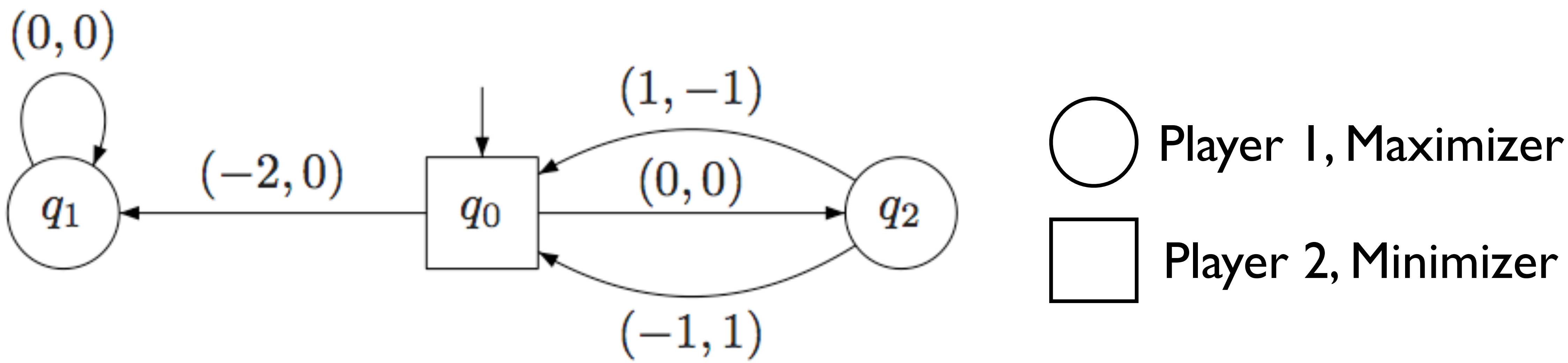
(ii) *The winner can be decided in* **NP∩coNP** *and in **pseudo-polynomial** time*

Rem: Those results hold no matter if lim sup/li
definition and the winner is the same f

Open question: are MP games solvable in PTime ?

Quantitative Objectives: Multi-dimension Extensions

Multi-dim. Mean-Payoff Games (MMPs)



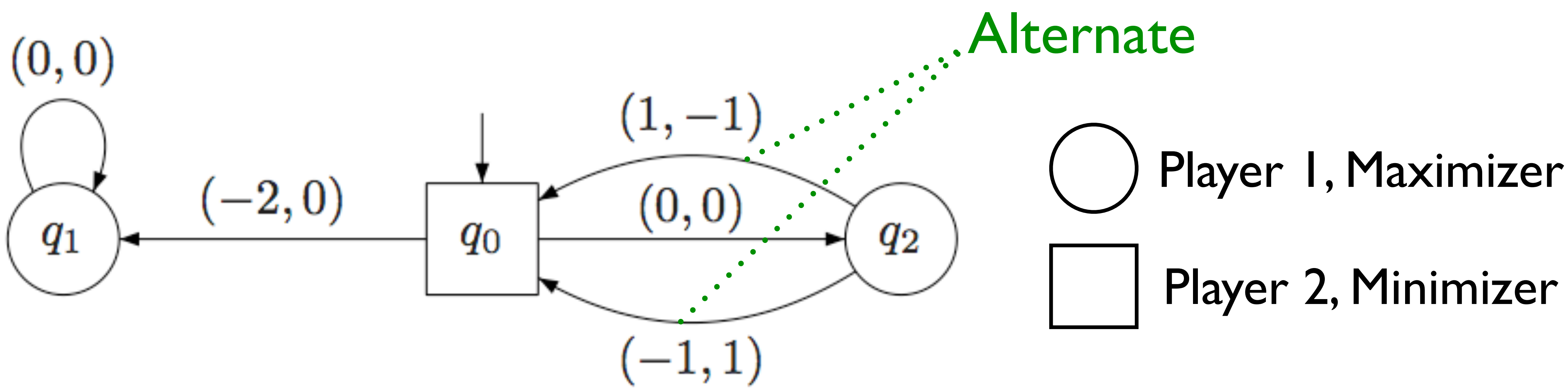
? $\exists\lambda_1$ s. t. **Outcome**$(q_0,\lambda_1) \models \mathrm{MP}_{\mathrm{inf.}} \geq (0,0)$

Multi-dimension objectives (for Player 1)

=

conjunction of one-dimension objectives

Multi-dim. Mean-Payoff Games (MMPs)

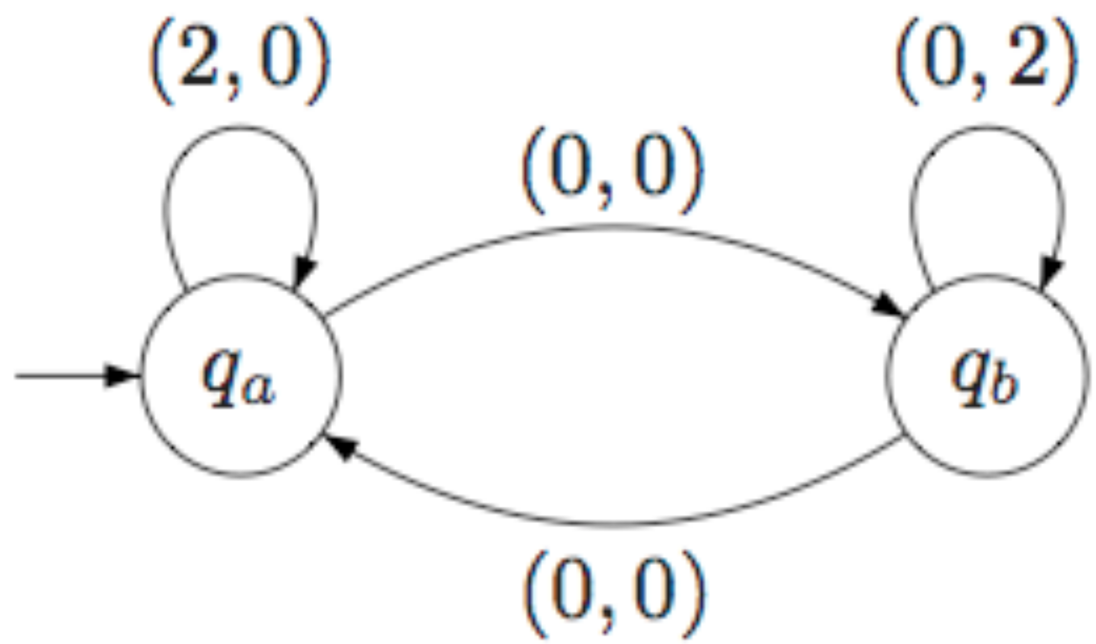


? $\exists\lambda_I$ s. t. **Outcome**$(q_0,\lambda_I) \vDash MP_{inf.} \geq (0,0)$

- Player I has a winning strategy.
- Player I may need **infinite** memory !
- Player II can play memoryless

gMPGs - Infinite Memory

To play optimally gMPGs, infinite memory is necessary

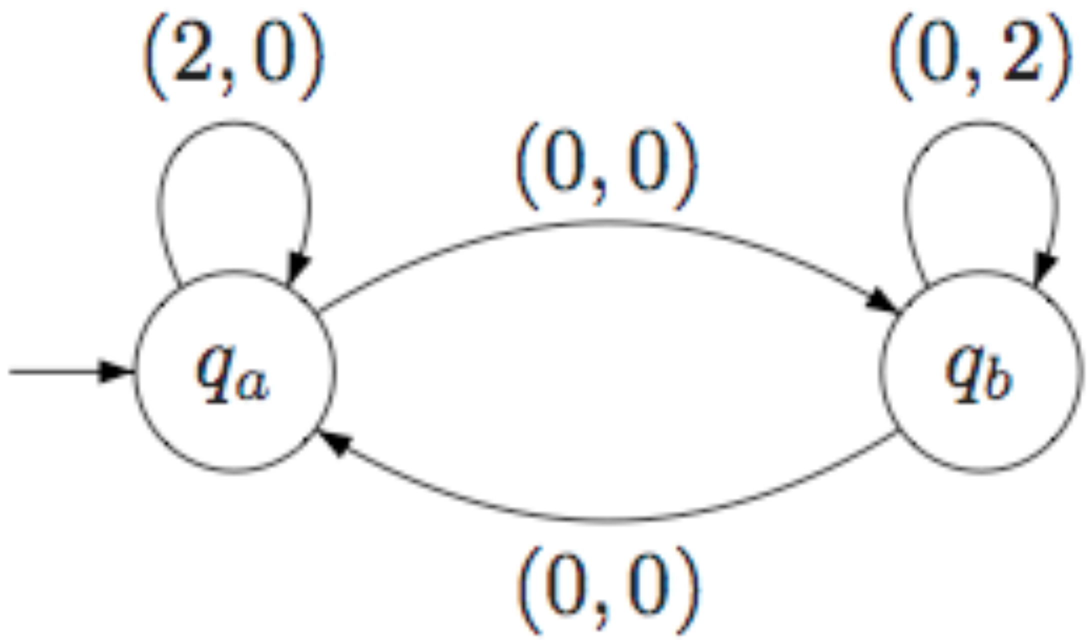


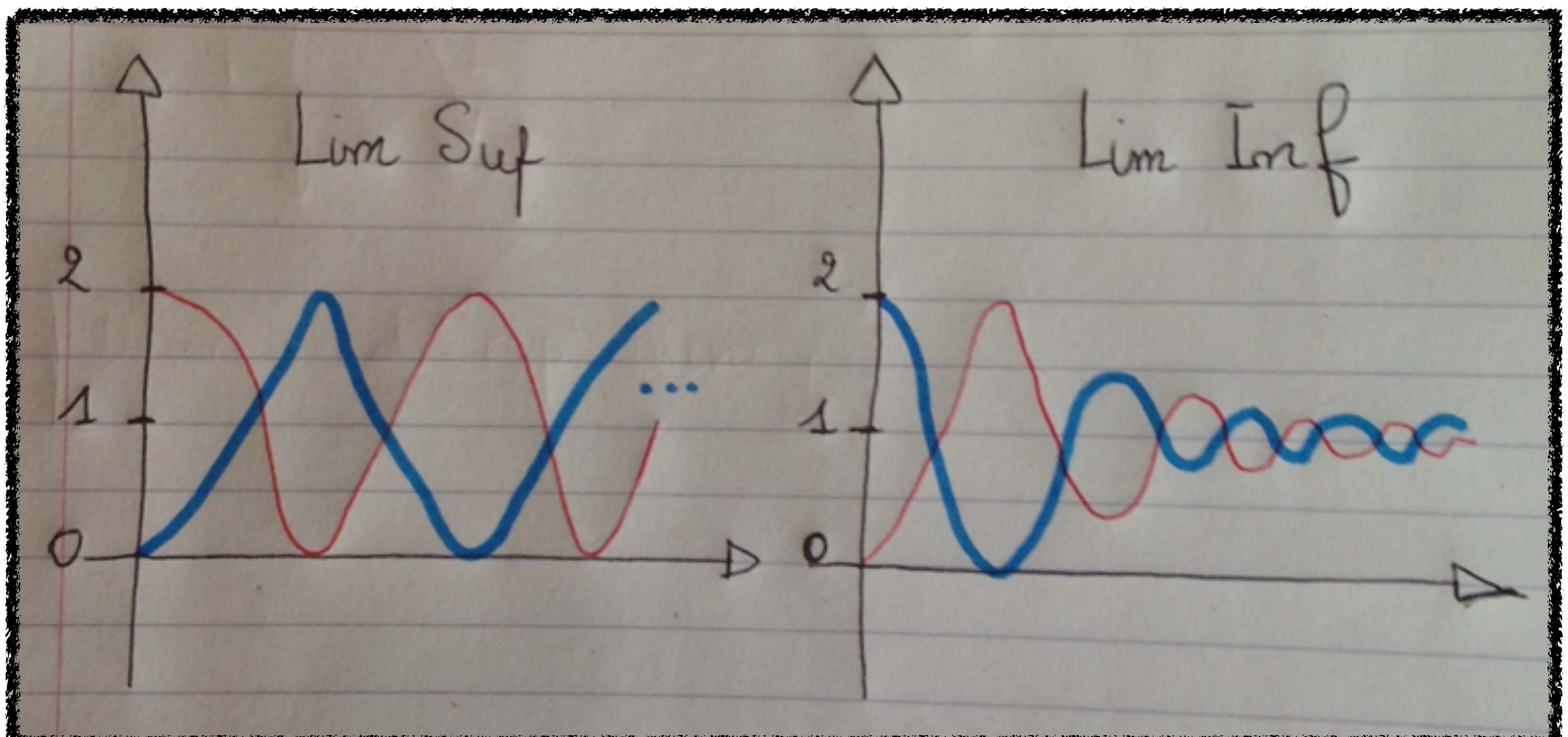
✦(2, 2) for Lim Sup MP

✦(1,1) is achievable for Lim Inf MP

✦None of the two is achievable with finite memory

gMPGs - Infinite Memory





Results for Extensions

	Opt. Stg. Player 1	Opt. Stg. Player 2	Complexity Decision
MP	Memoryless	Memoryless	NP∩coNP
MMPG - **Sup**	**Infinite**	Memoryless	NP∩coNP
MMPG - **Inf**	**Infinite**	Memoryless	coNP-C
MMPG - **Mix**	**Infinite**	Memoryless	coNP-C

Results for Extensions

	Opt. Stg. Player 1	Opt. Stg. Player 2	
MP	Memoryless		NP∩coNP
MMPG - **Su**		Memoryless	NP∩coNP
MMPG - **Inf**	**Infinite**	Memoryless	coNP-C
MMPG - **Mix**	**Infinite**	Memoryless	coNP-C

ε-optimality acheivable with finite memory strategies

Variations on MP: Window Objectives

Window Objectives

- Space for new definitions as classical objectives have **drawbacks**:
  ① complexity of MP is open
  ② MMP is sensitive to lim inf. vs. lim sup.

➡ **Window objectives**: look at the payoff through a local finite window sliding over the play

- **conservative approximations** of MP
- ensure good properties within a bounded time horizon
- algorithms and complexity results:
  - **PTIME-C** for 1 dim. fixed and polynomial window size
  - EXPTIME-C for k dim. fixed window size
  - ... and more

Window Objectives - Definitions

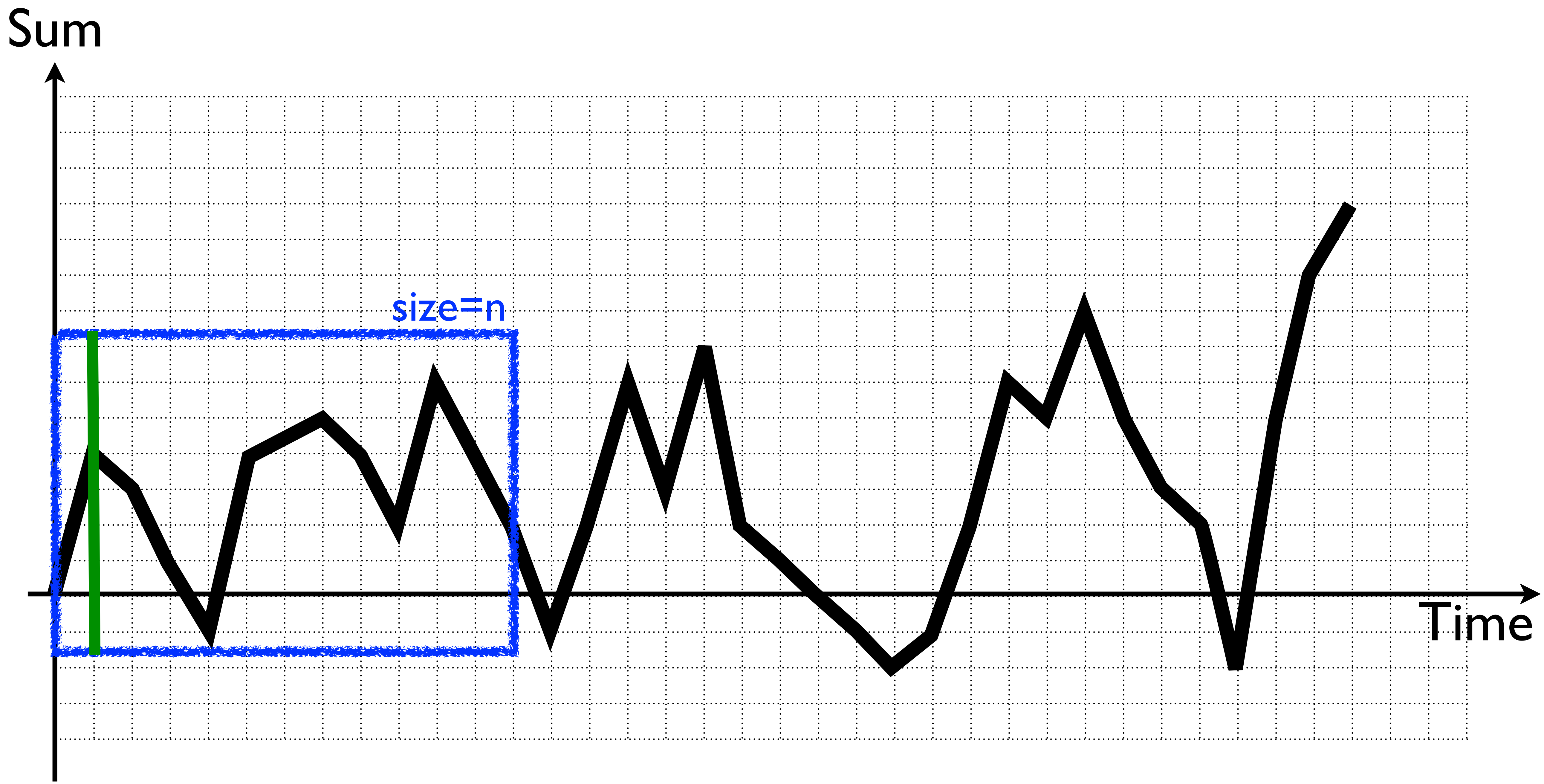
Idea: look at payoffs through a local finite window

$\Longrightarrow$ mean should be above zero **within** window size

Window Objectives - Definitions

Idea: look at payoffs through a local finite window

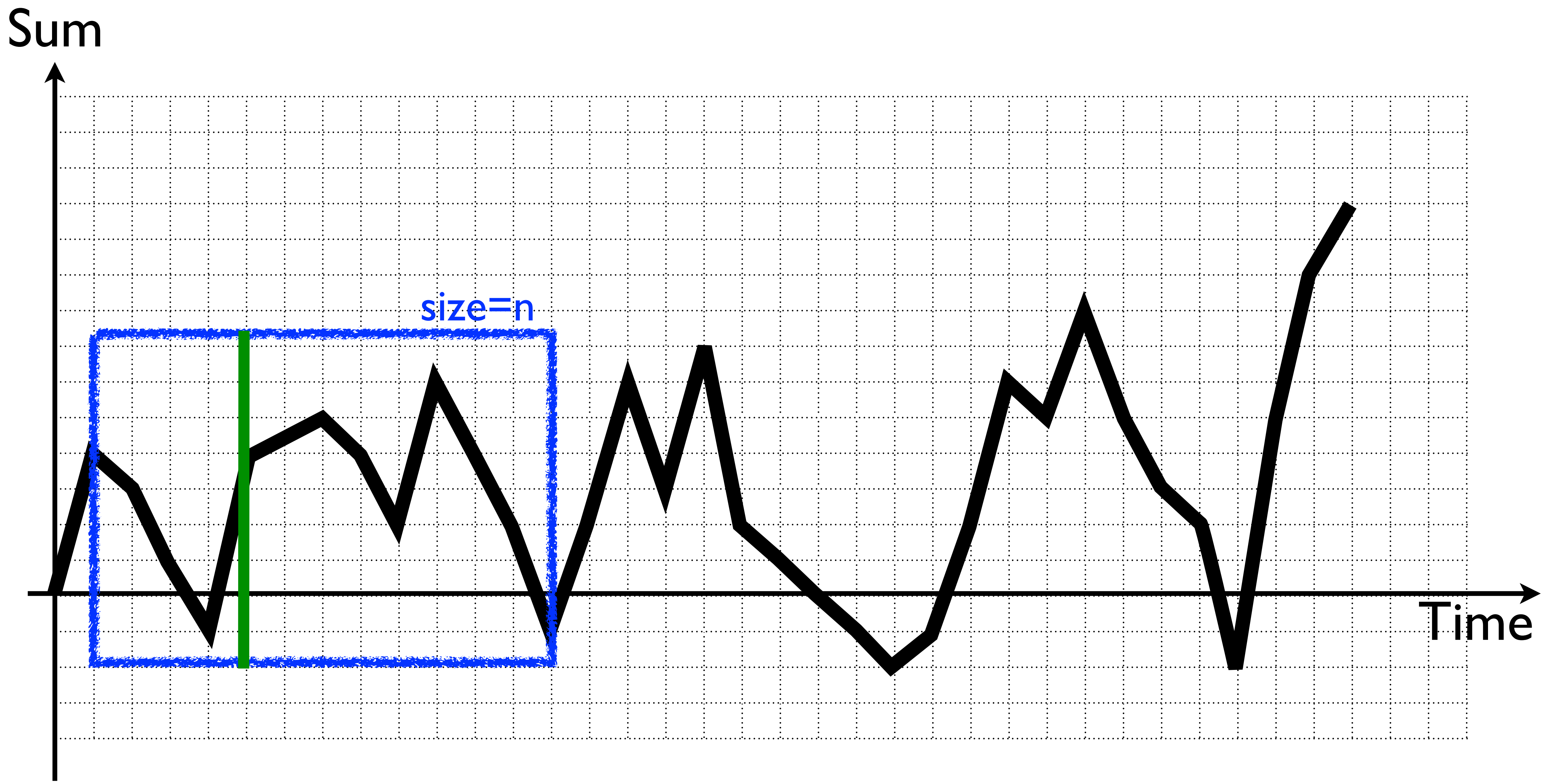
$\Longrightarrow$ mean should be above zero **within** window size



Window Objectives - Definitions

Idea: look at payoffs through a local finite window

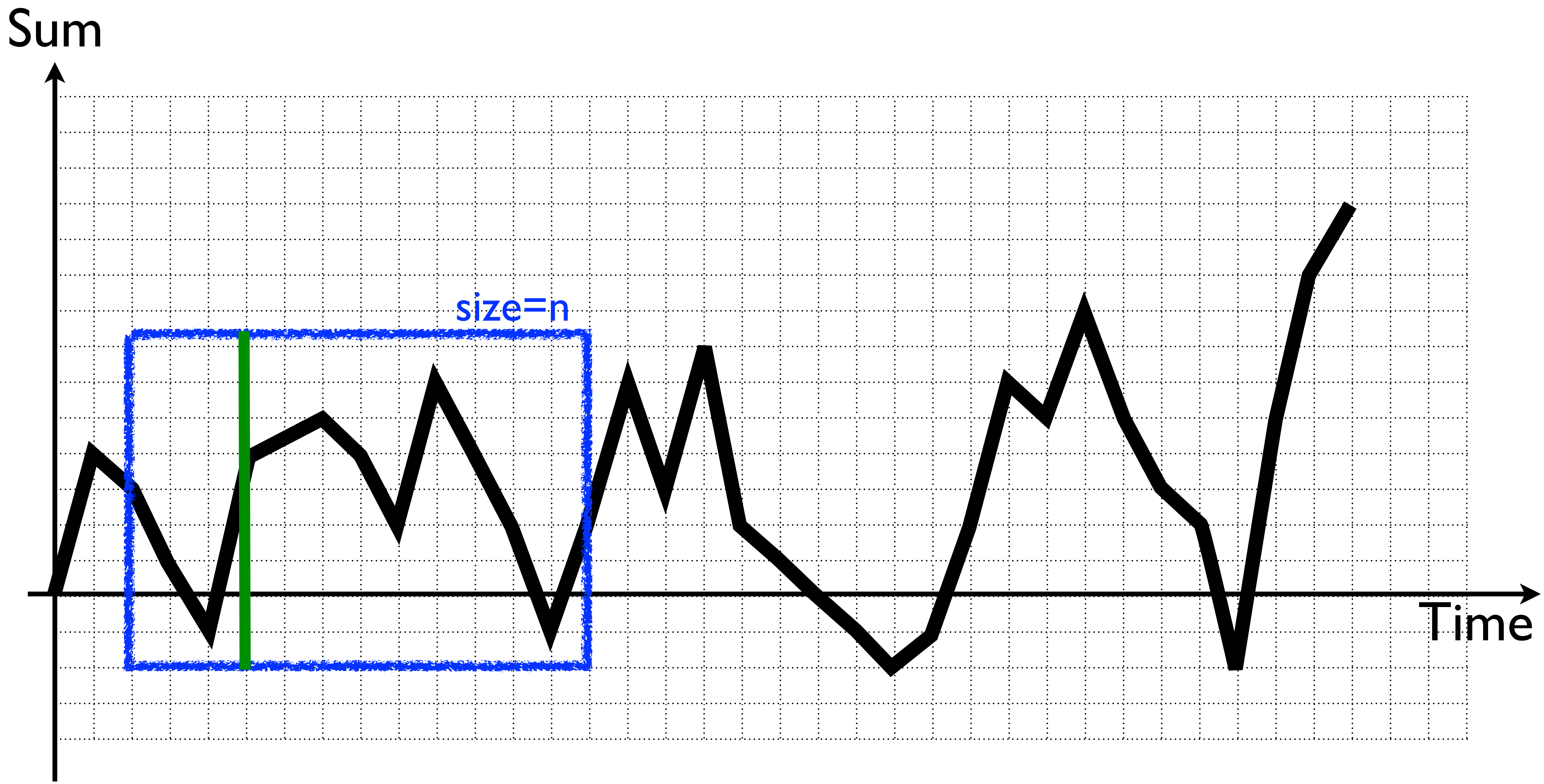
$\Longrightarrow$ mean should be above zero **within** window size



Window Objectives - Definitions

Idea: look at payoffs through a local finite window

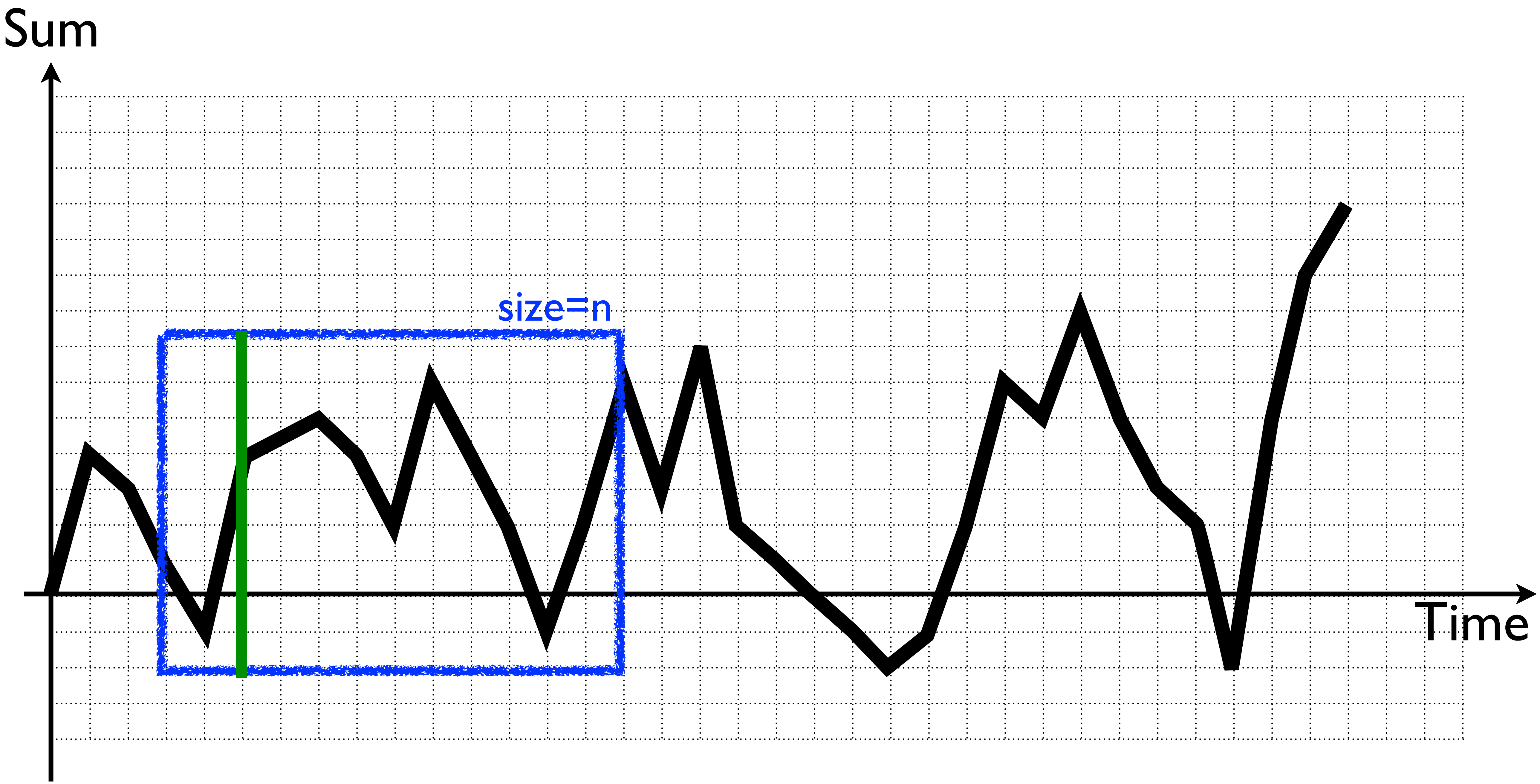
$\Longrightarrow$ mean should be above zero **within** window size



Window Objectives - Definitions

Idea: look at payoffs through a local finite window

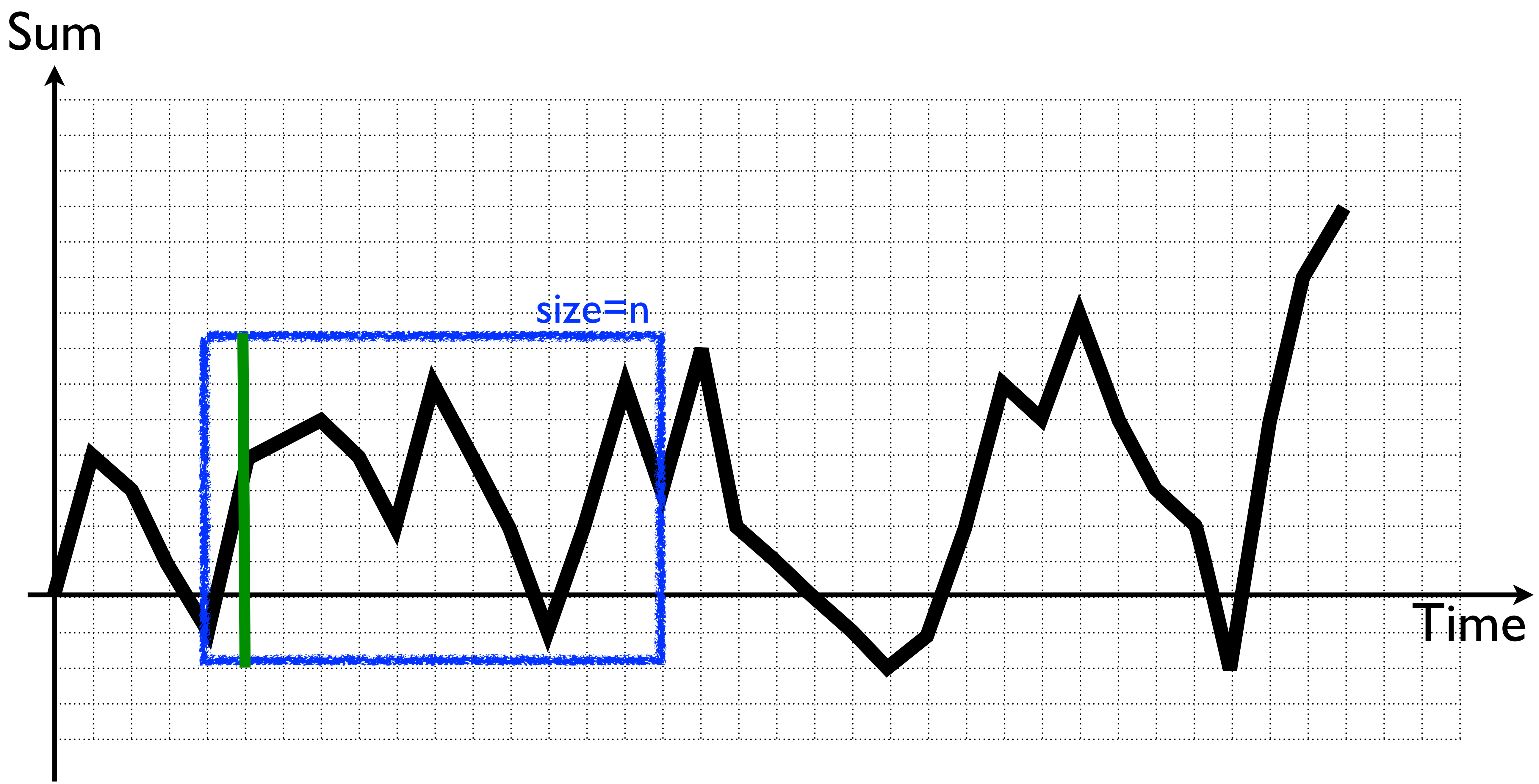
$\Longrightarrow$ mean should be above zero **within** window size



Window Objectives - Definitions

Idea: look at payoffs through a local finite window

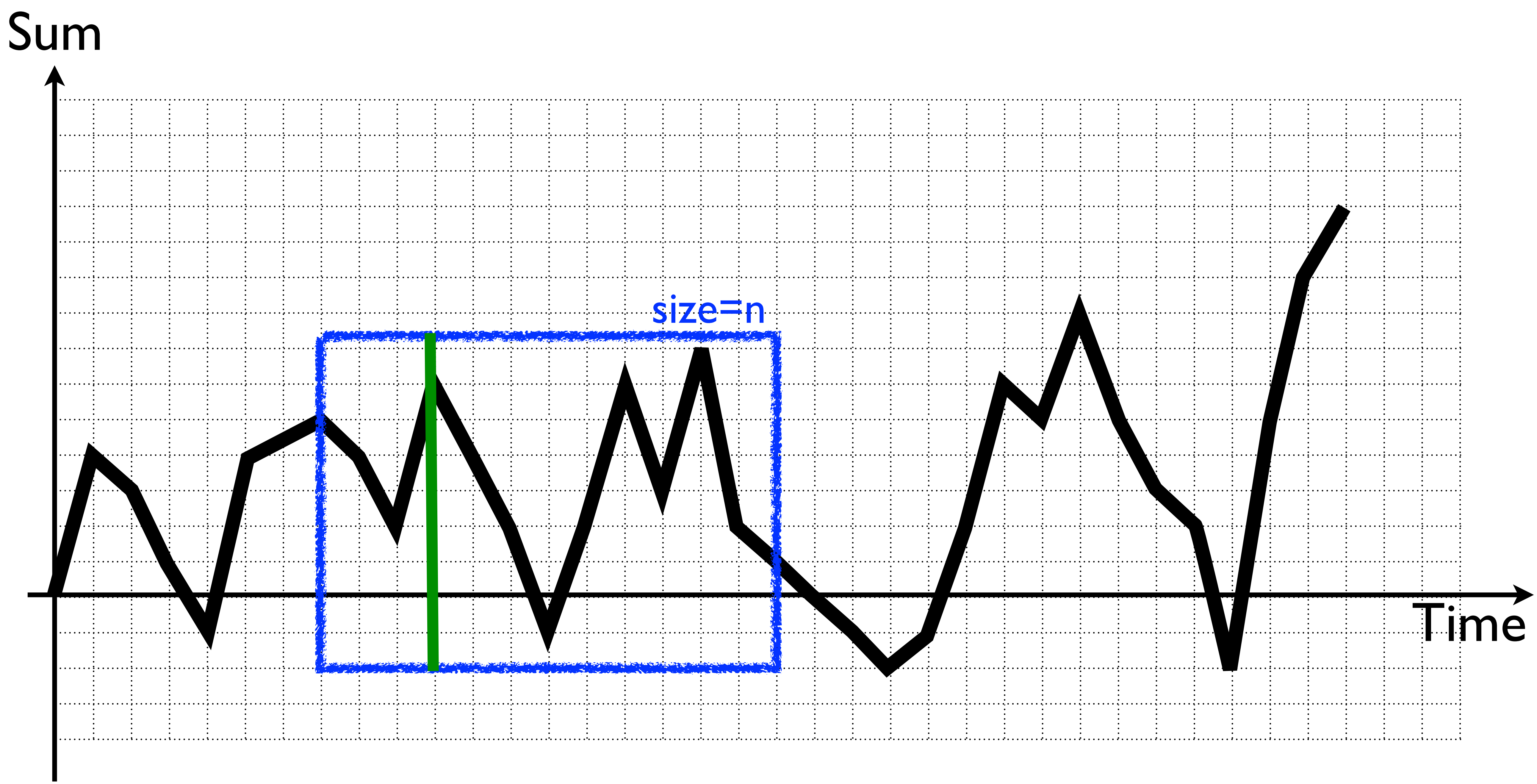
$\Longrightarrow$ mean should be above zero **within** window size



Window Objectives - Definitions

Idea: look at payoffs through a local finite window

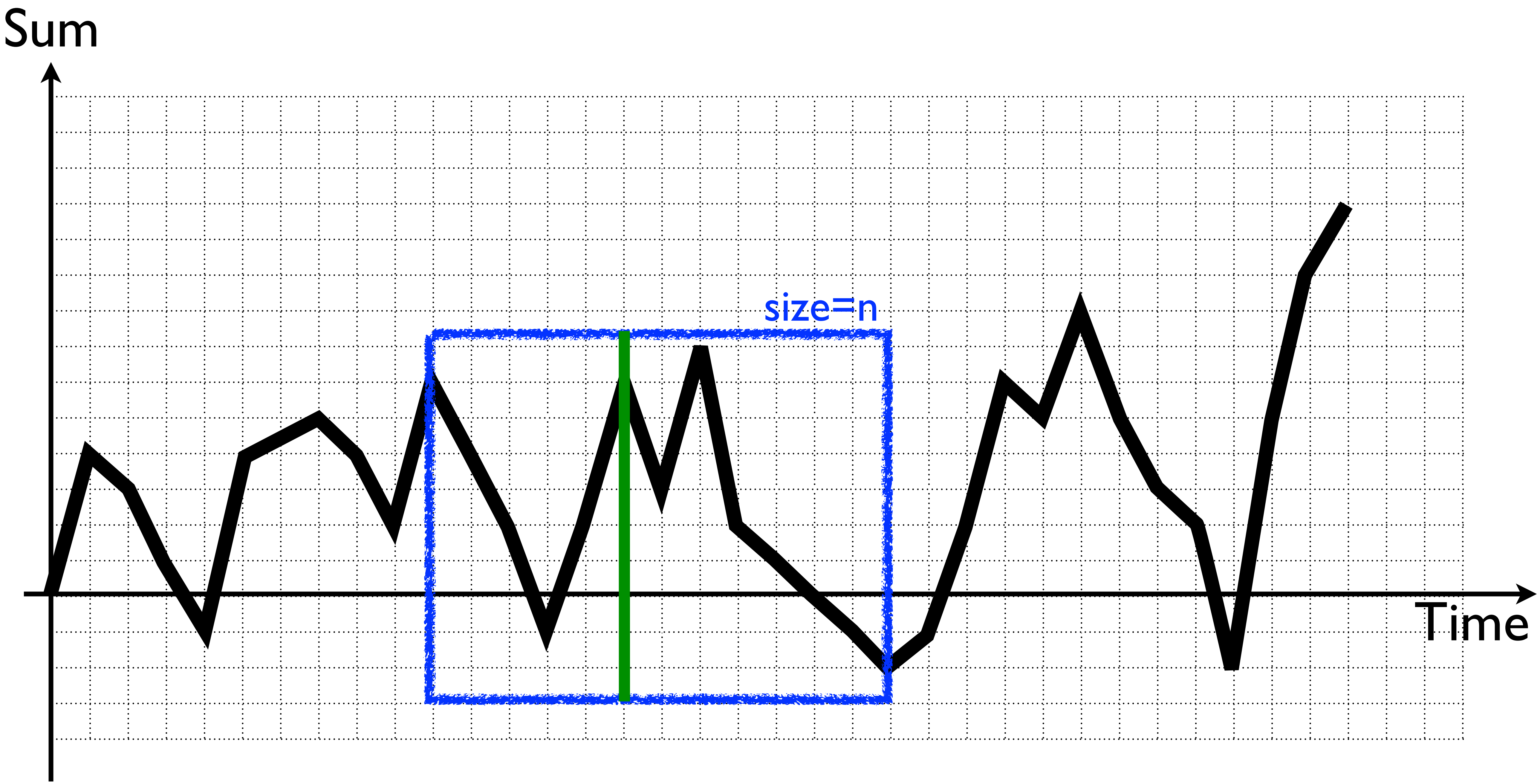
$\Longrightarrow$ mean should be above zero **within** window size



Window Objectives - Definitions

Idea: look at payoffs through a local finite window

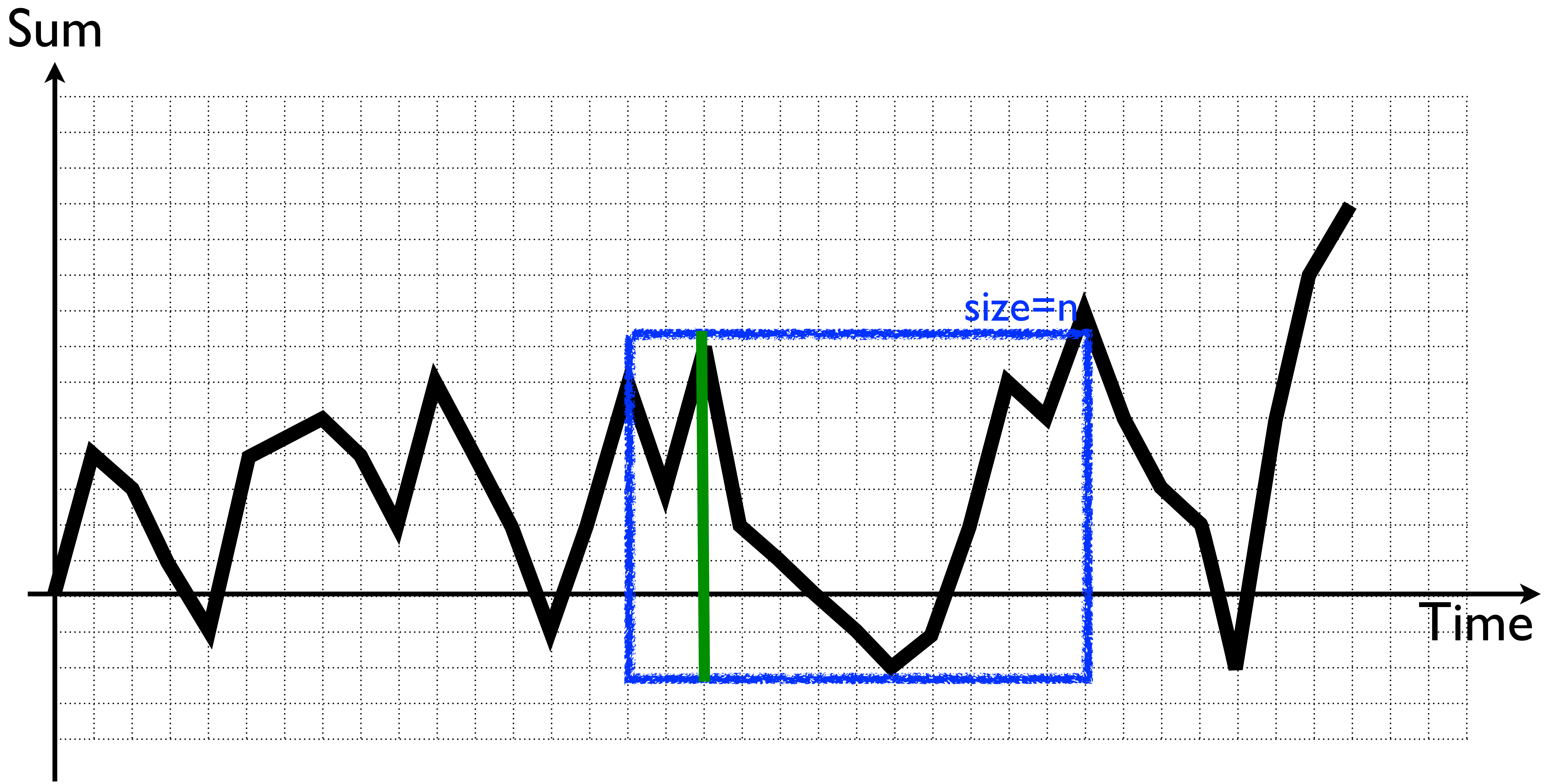
$\Longrightarrow$ mean should be above zero **within** window size



Window Objectives - Definitions

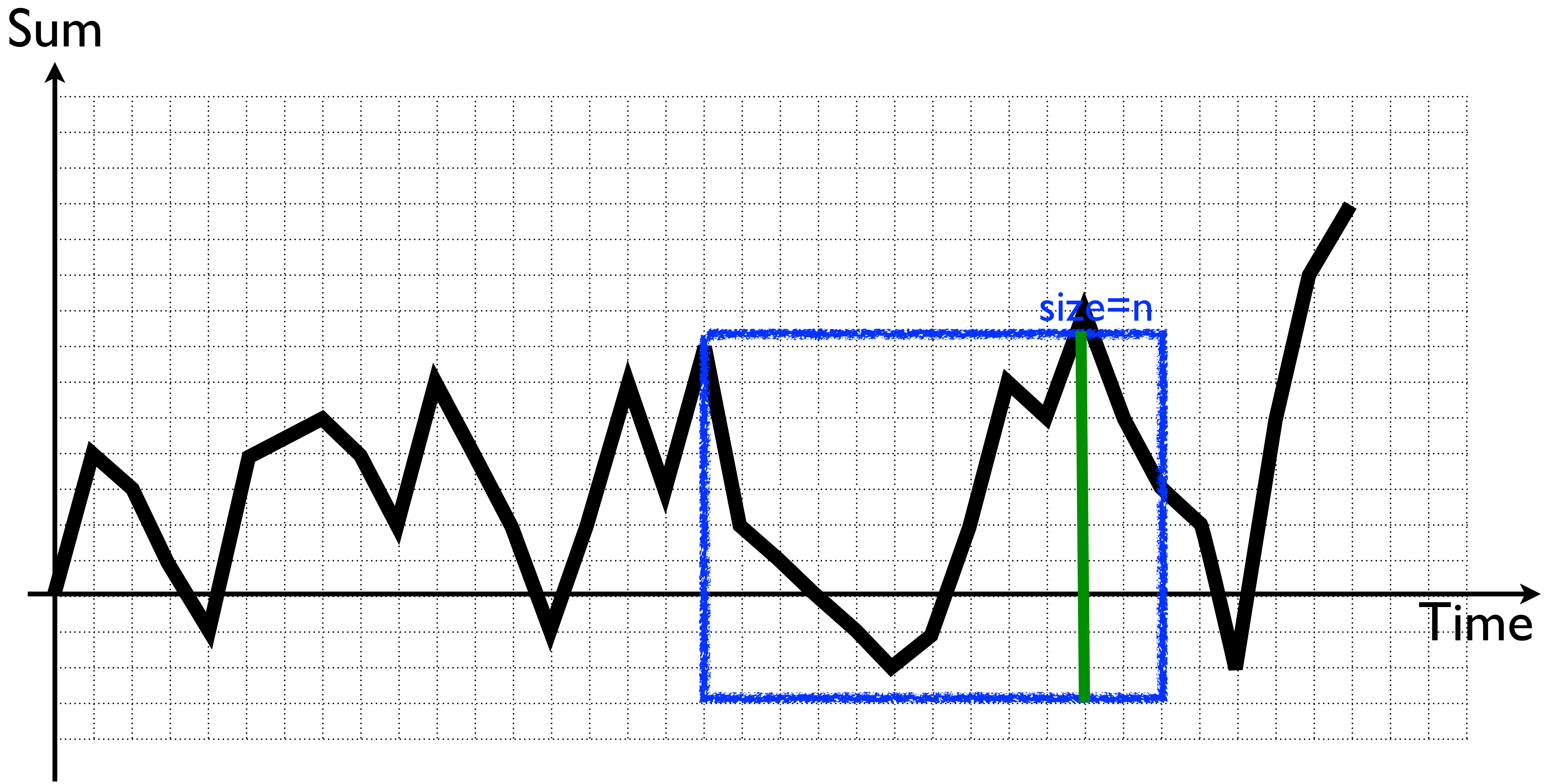
Idea: look at payoffs through a local finite window

$\Longrightarrow$ mean should be above zero **within** window size



Window Objectives - Definitions

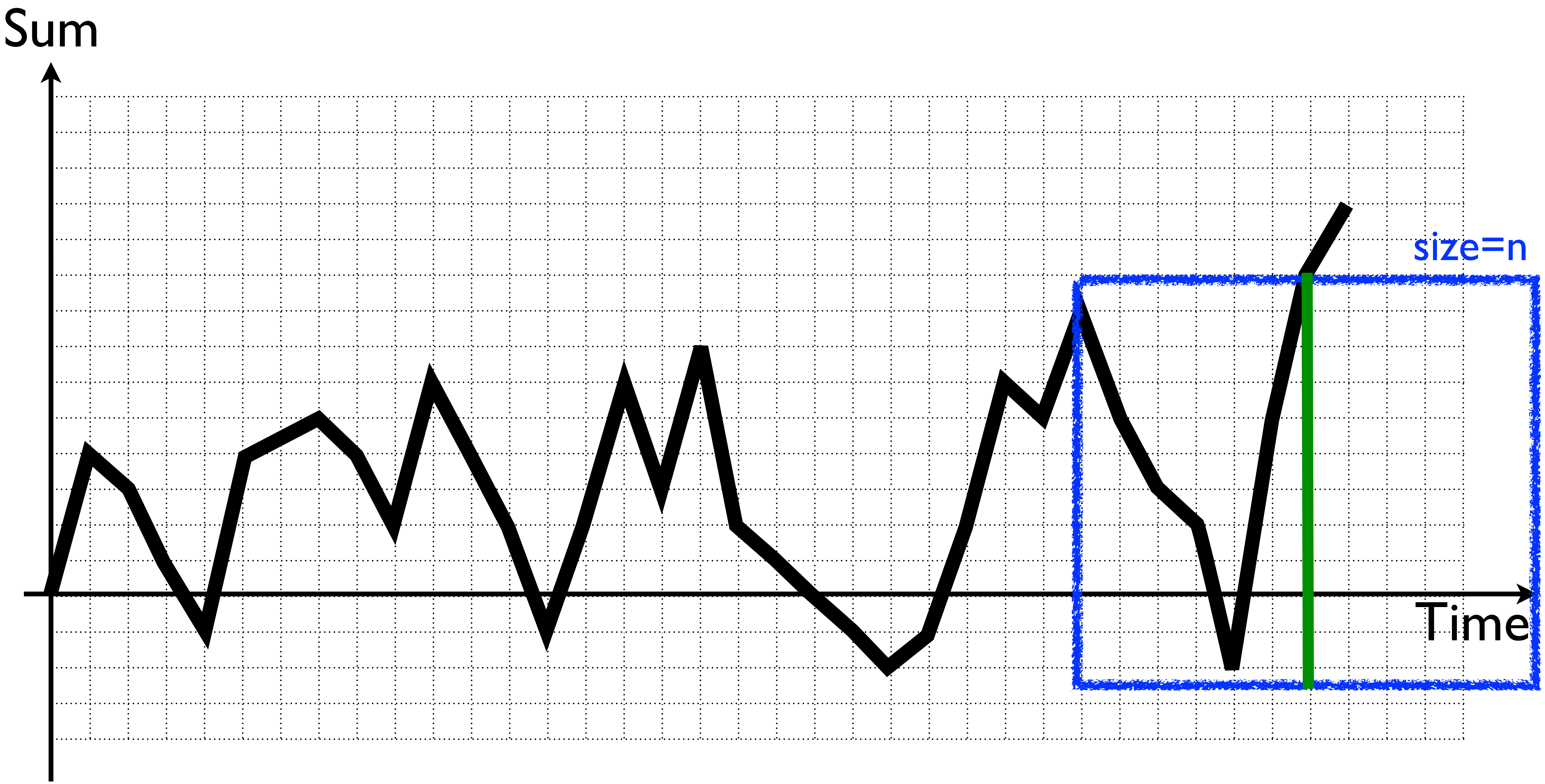
Idea: look at payoffs through a local finite window
$\Longrightarrow$ mean should be above zero **within** window size



Window Objectives - Definitions

Idea: look at payoffs through a local finite window

$\Longrightarrow$ mean should be above zero **within** window size

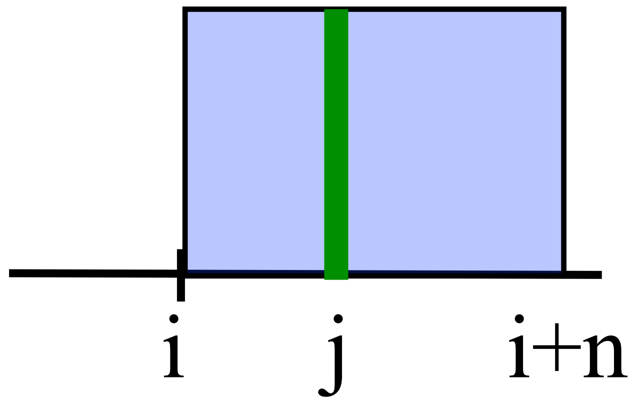


Formal Definitions

Formal Definitions

Good Window of size n

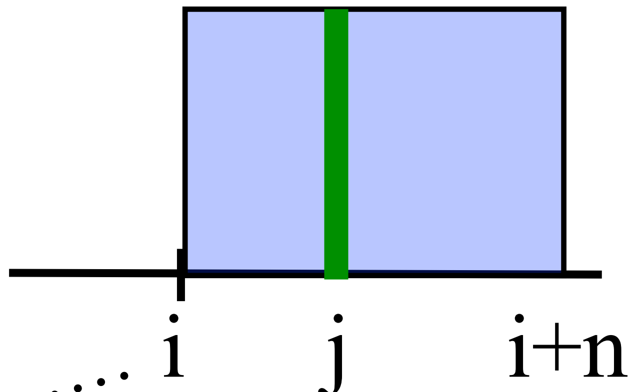
play ρ satisfies $\mathbf{GW}(n)$ at position $i \geq 0$
iff $\exists j$: $i \leq j \leq i+n$ s.t. $\mathrm{Sum}(\rho(i..j)) \geq 0$,
noted $\rho(i..\infty) \vDash \mathbf{GW}(n)$



Formal Definitions

Good Window of size n

play ρ satisfies **GW**(n) at position i≥0
iff ∃j: i ≤ j ≤ i+n s.t. Sum(ρ(i..j))≥0,
noted ρ(i..∞) ⊨ **GW**(n)



Window opens at i

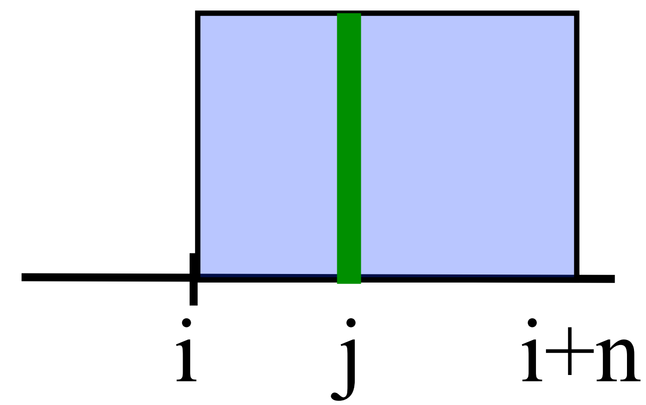
successfully closes at j

Otherwise, it unsuccessfully closes at i+n

Formal Definitions

Good Window of size n

play ρ satisfies $\mathbf{GW}(n)$ at position $i \geq 0$
iff $\exists j$: $i \leq j \leq i+n$ s.t. $\mathrm{Sum}(\rho(i..j)) \geq 0$,
noted $\rho(i..\infty) \vDash \mathbf{GW}(n)$



Direct Fixed Window of size n:

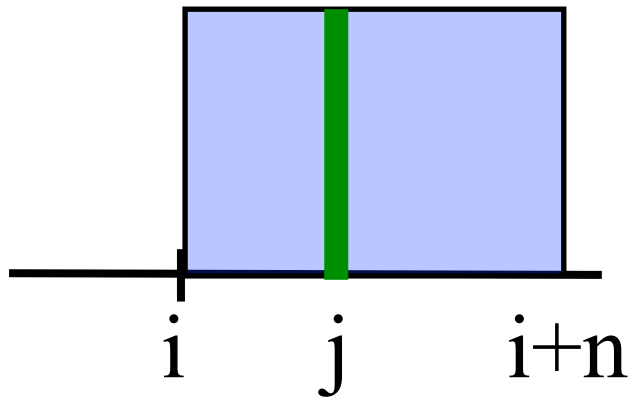
play ρ satisfies $\mathbf{DFW}(n)$ iff for all $i \geq 0$, $\rho(i..\infty) \vDash \mathbf{GW}(n)$

$$\mathbf{DFW}(n) \equiv \Box\, \mathbf{GW}(n)$$

Formal Definitions

Good Window of size n

play ρ satisfies **GW**(n) at position $i \geq 0$
iff $\exists j$: $i \leq j \leq i+n$ s.t. $Sum(\rho(i..j)) \geq 0$,
noted $\rho(i..\infty) \vDash$ **GW**(n)



Direct Fixed Window of size n:

play ρ satisfies **DFW**(n) iff for all $i \geq 0$, $\rho(i..\infty) \vDash$ **GW**(n)

DFW(n)
$\equiv \Box$ **GW**(n)

Fixed Window of size n

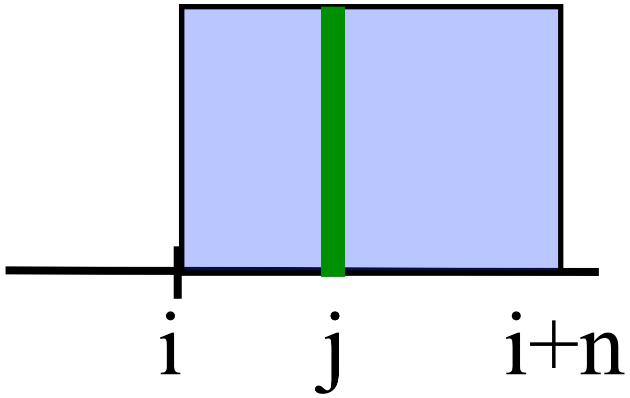
play ρ satisfies **FixedW**(n) iff
there exists $i \geq 0$, $\rho(i..\infty) \vDash$ **DFW**(n)

FixedW(n)
$\equiv \Diamond \Box$ **GW**(n)

Formal Definitions

Good Window of size n

play ρ satisfies $\mathbf{GW}(n)$ at position $i \geq 0$
iff $\exists j$: $i \leq j \leq i+n$ s.t. $Sum(\rho(i..j)) \geq 0$,
noted $\rho(i..\infty) \vDash \mathbf{GW}(n)$



Direct Fixed Window of size n:

play ρ satisfies $\mathbf{DFW}(n)$ iff for all $i \geq 0$, $\rho(i..\infty) \vDash \mathbf{GW}(n)$

$$\mathbf{DFW}(n) \equiv \Box\, \mathbf{GW}(n)$$

Fixed Window of size n

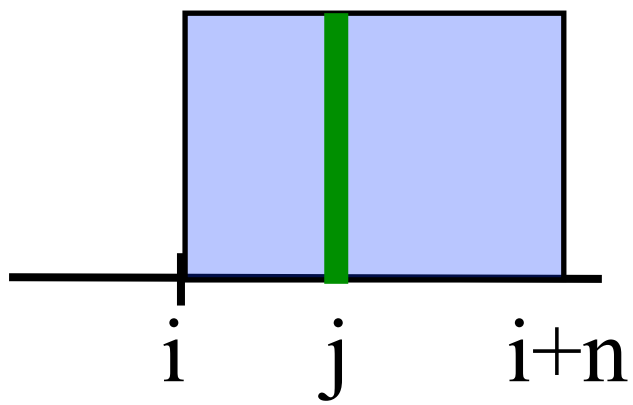
play ρ satisfies $\mathbf{FixedW}(n)$ iff
there exists $i \geq 0$, $\rho(i..\infty) \vDash \mathbf{DFW}(n)$

$$\mathbf{FixedW}(n) \equiv \Diamond \Box\, \mathbf{GW}(n)$$

Formal Definitions

Good Window of size n

play ρ satisfies $\mathbf{GW}(n)$ at position $i \geq 0$
iff $\exists j$: $i \leq j \leq i+n$ s.t. $\mathrm{Sum}(\rho(i..j)) \geq 0$,
noted $\rho(i..\infty) \models \mathbf{GW}(n)$



Direct Fixed Window of size n:

play ρ satisfies $\mathbf{DFW}(n)$ iff for all $i \geq 0$, $\rho(i..\infty) \models \mathbf{GW}(n)$

$$\mathbf{DFW}(n) \equiv \Box\, \mathbf{GW}(n)$$

Fixed Window of size n

play ρ satisfies $\mathbf{FixedW}(n)$ iff
there exists $i \geq 0$, $\rho(i..\infty) \models \mathbf{DFW}(n)$

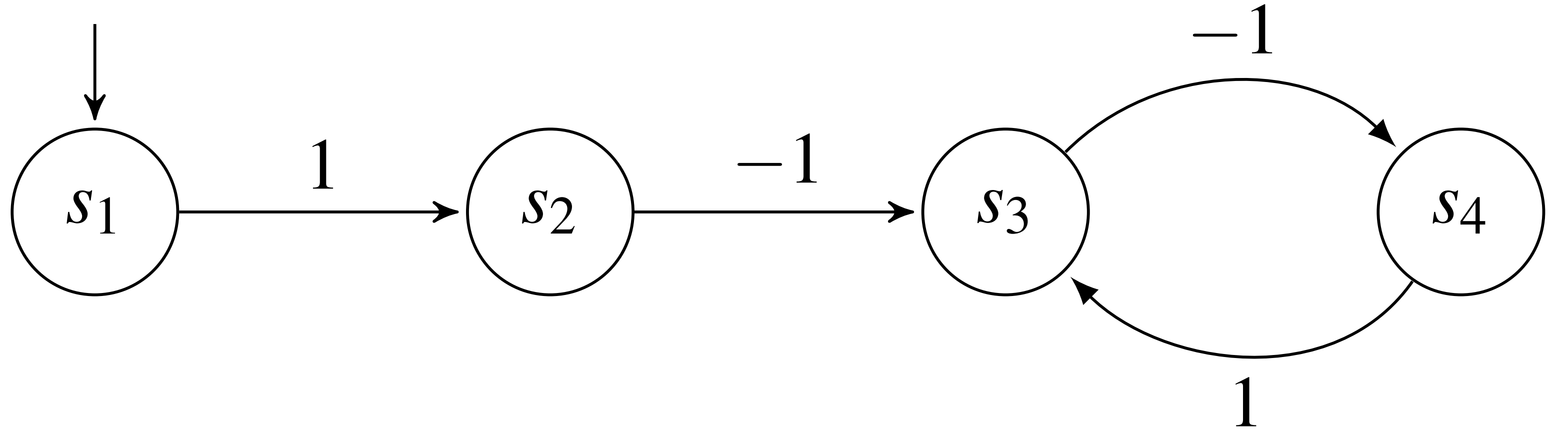
$$\mathbf{FixedW}(n) \equiv \Diamond \Box\, \mathbf{GW}(n)$$

Bounded Window

play ρ satisfies $\mathbf{BW}$ iff
there exists $n \geq 1$, $\rho \models \mathbf{FixedW}(n)$

$$\mathbf{BW} \equiv \exists n \cdot \mathbf{FixedW}(n)$$

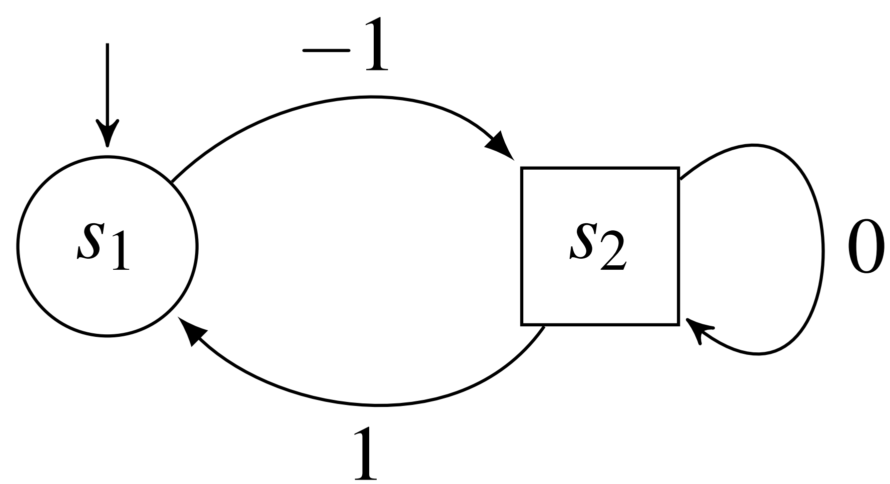
Examples



➡ Fixed window is satisfied for size≥2

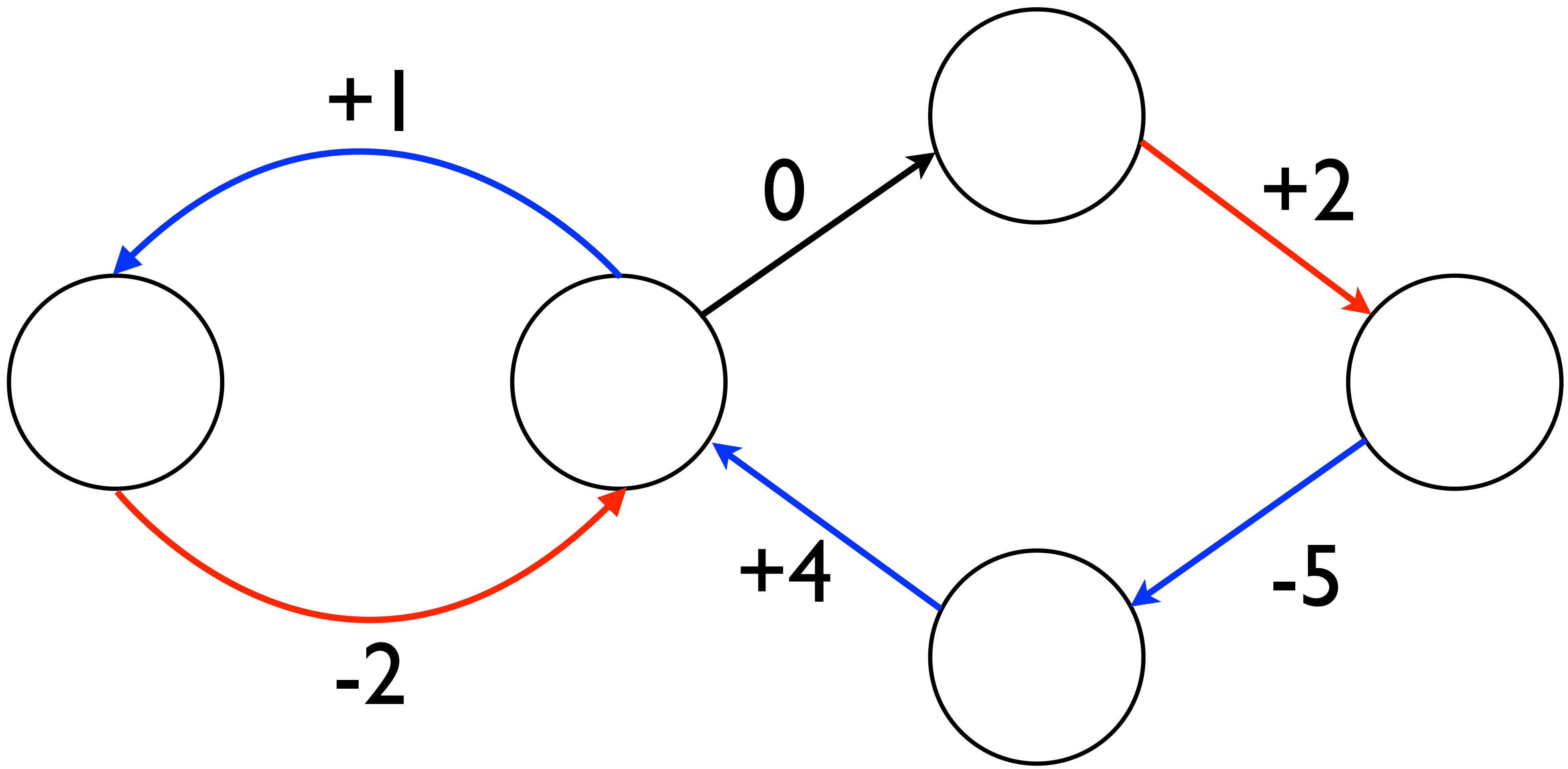
➡ Direct fixed window is not satisfied for any size !

Examples



Player 1 wins mean-payoff
... but he looses for every window objectives

Examples



For n=3: memory needed

For n=4 : no memory needed

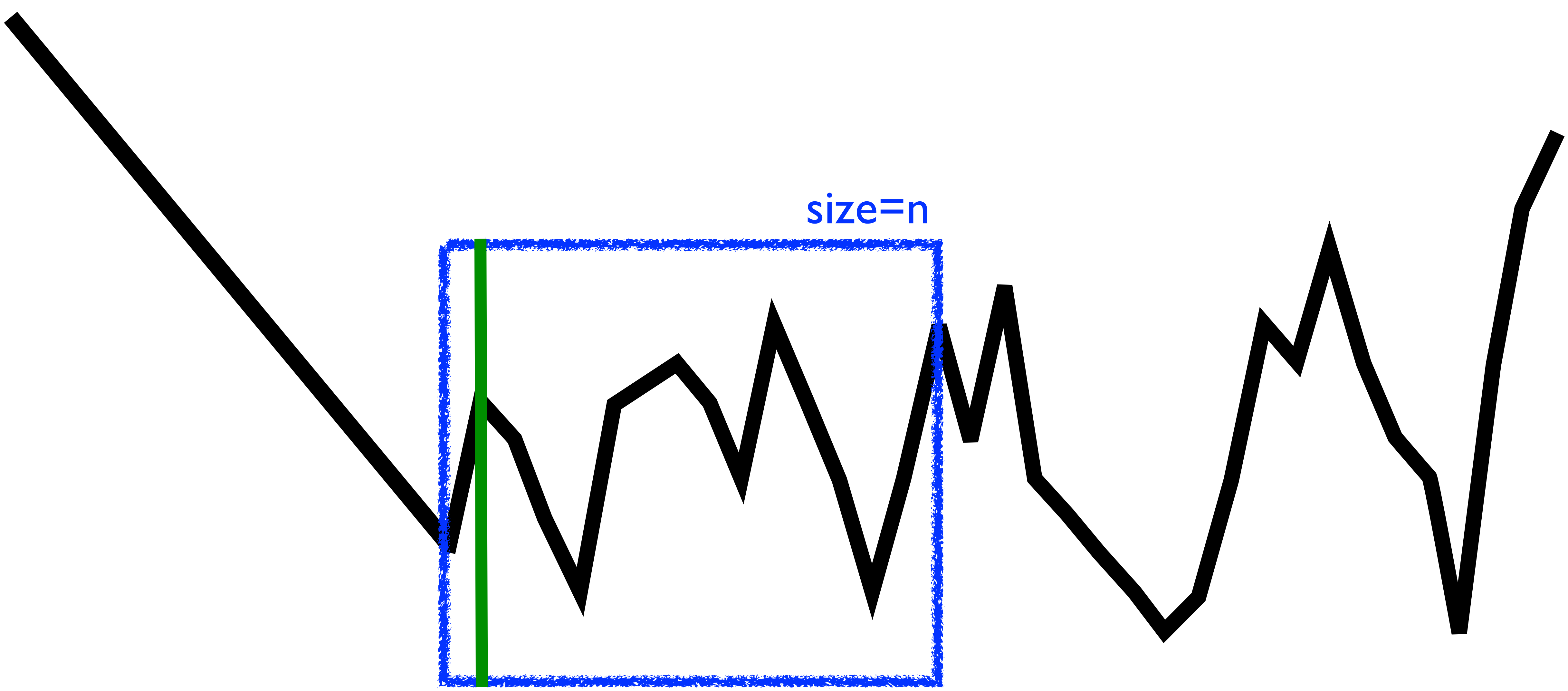
Relation with “classical” objectives

Relation between MP/TP and W.O.

If the answer to one of window mean-payoff problems is YES, *then the answer to the mean-payoff threshold problem for threshold zero is also* YES.

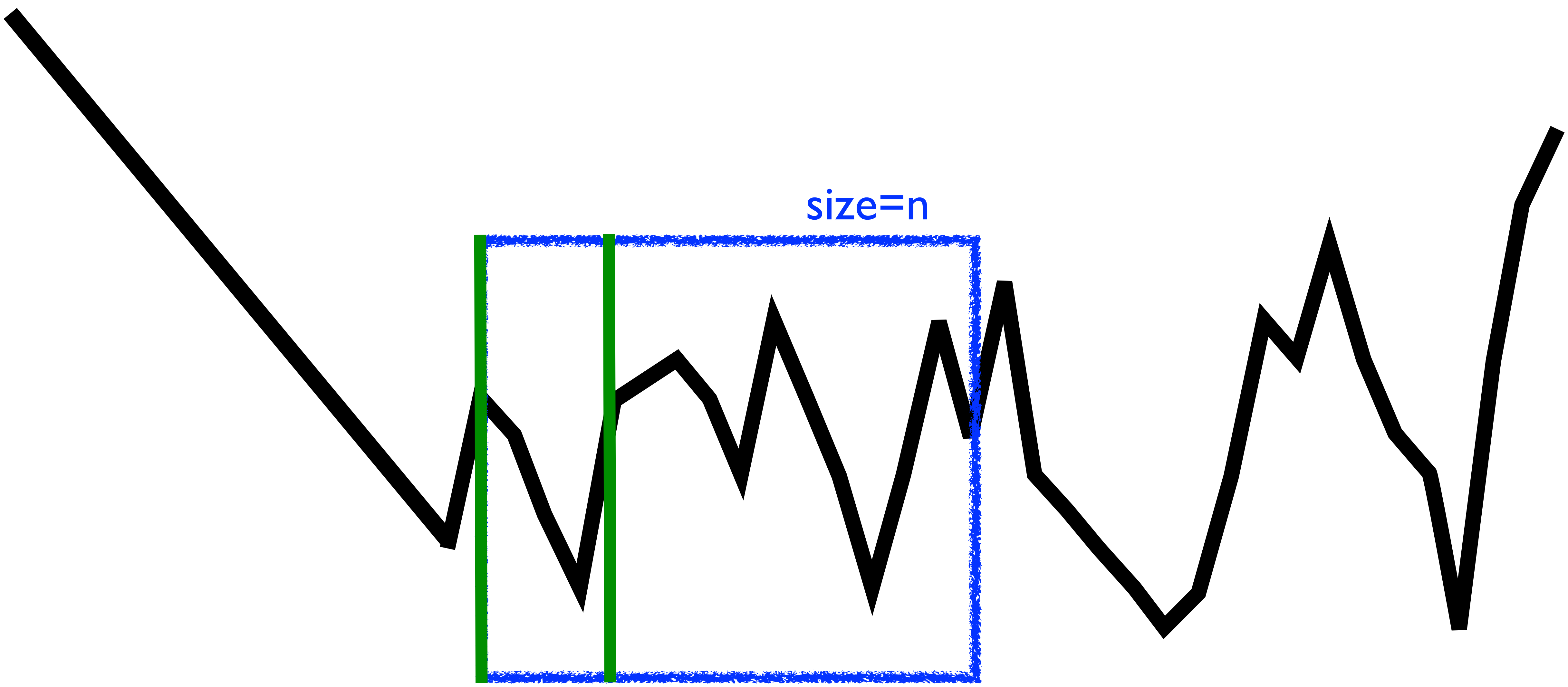
Relation between MP/TP and W.O.

If the answer to one of window mean-payoff problems is YES*, then the answer to the mean-payoff threshold problem for threshold zero is also* YES.



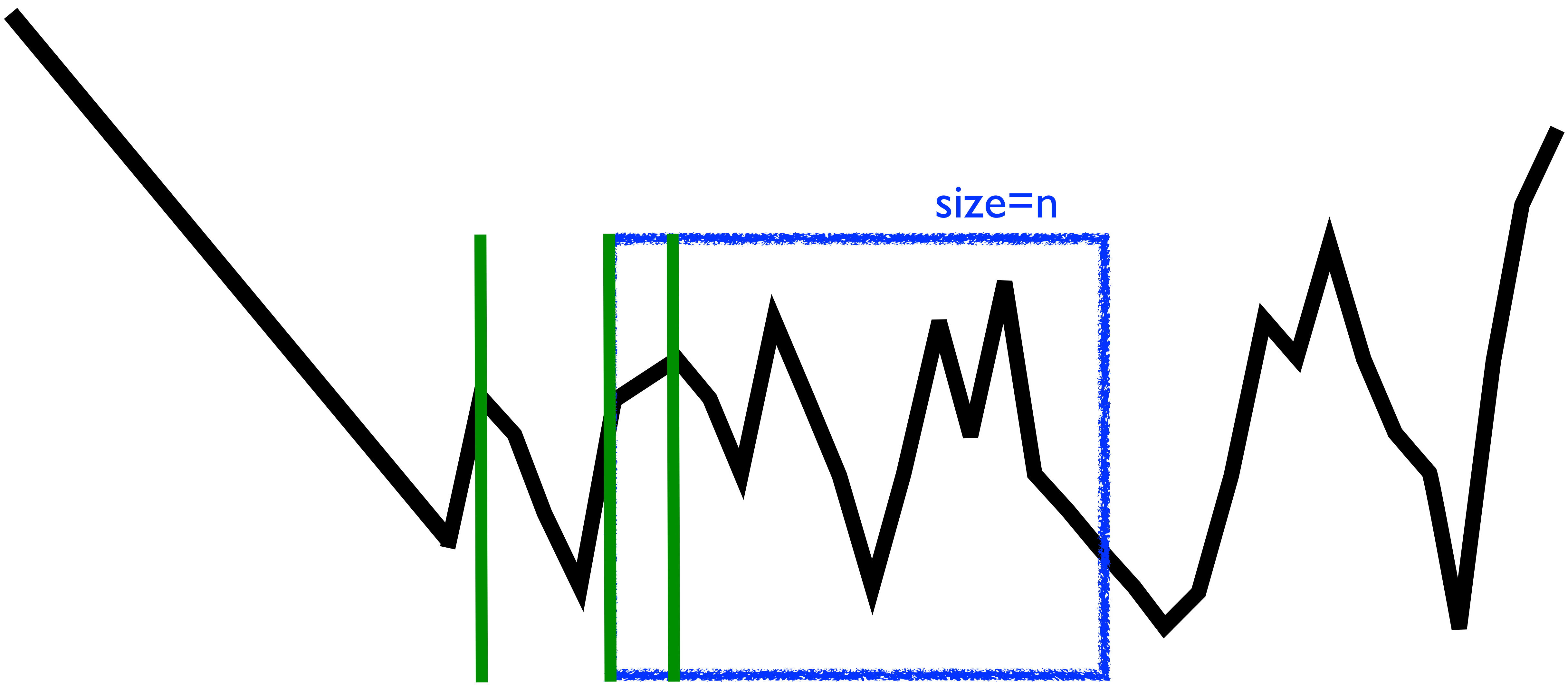
Relation between MP/TP and W.O.

If the answer to one of window mean-payoff problems is YES, *then the answer to the mean-payoff threshold problem for threshold zero is also* YES.



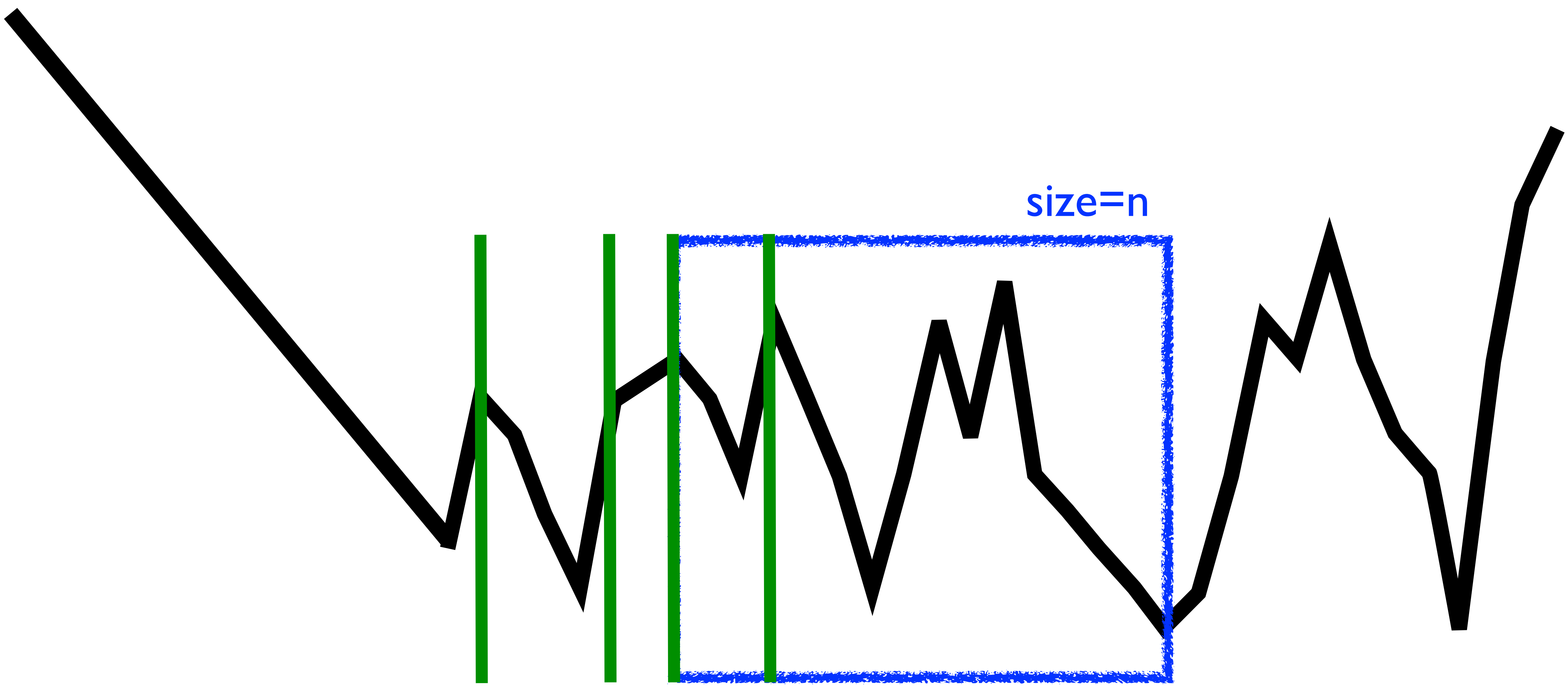
Relation between MP/TP and W.O.

If the answer to one of window mean-payoff problems is YES, *then the answer to the mean-payoff threshold problem for threshold zero is also* YES.



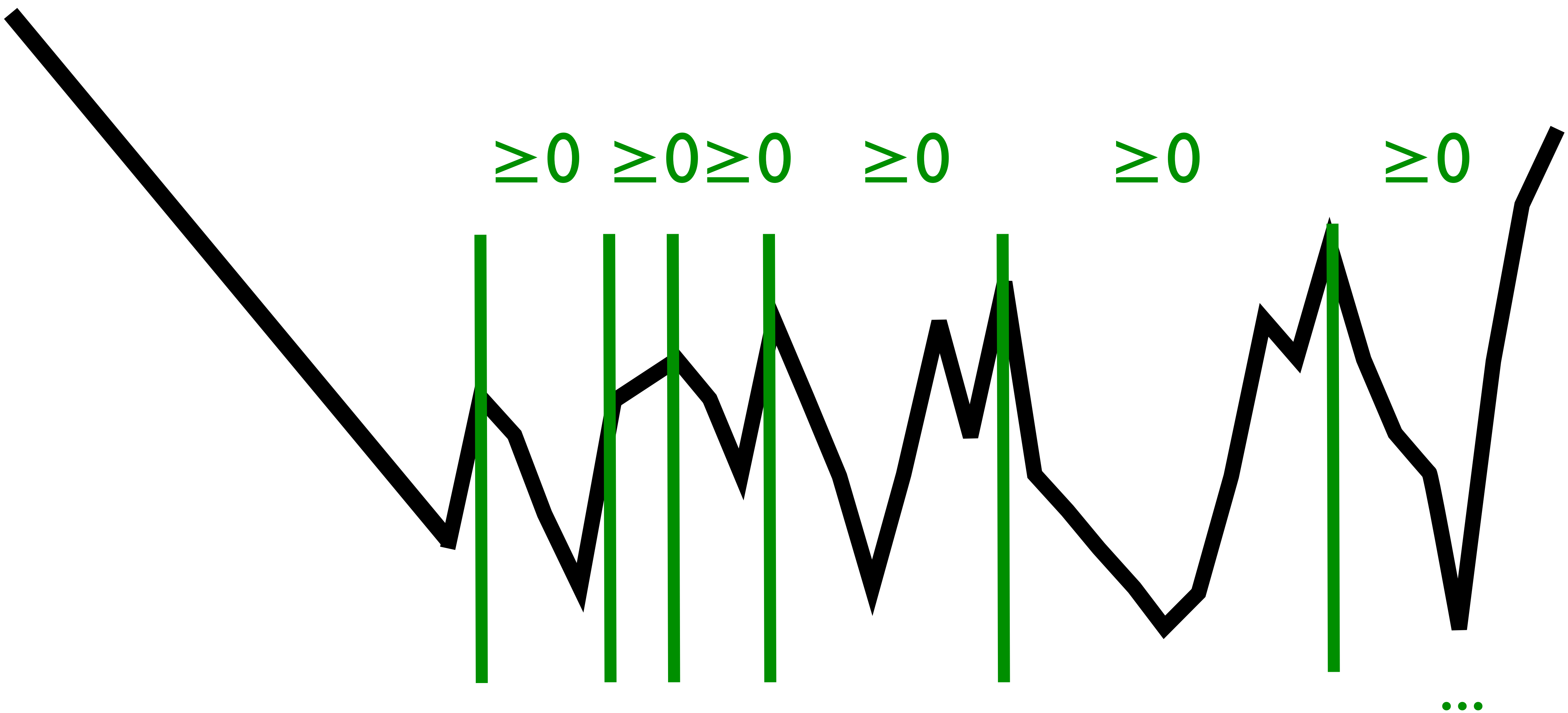
Relation between MP/TP and W.O.

If the answer to one of window mean-payoff problems is YES, *then the answer to the mean-payoff threshold problem for threshold zero is also* YES.



Relation between MP/TP and W.O.

If the answer to one of window mean-payoff problems is YES*,*
then the answer to the mean-payoff threshold problem for
threshold zero is also YES.

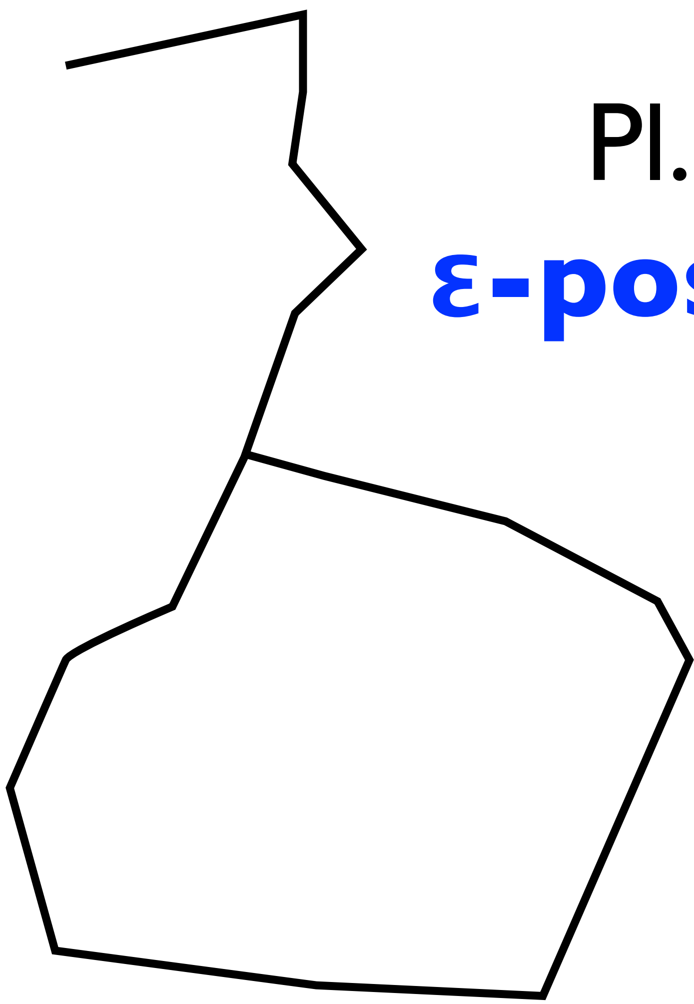


Relation between MP/TP and W.O.

If there exists $\varepsilon > 0$ *s.t. the answer to the mean-payoff threshold problem for threshold* ε *is* YES, *then the answer to the* **BW** *problem is also* YES.

Relation between MP/TP and W.O.

If there exists $\varepsilon > 0$ *s.t. the answer to the mean-payoff threshold problem for threshold* ε *is* YES, *then the answer to the* **BW** *problem is also* YES.



Pl.1 can force

ε-positive cycles

$\Rightarrow$

Pl.1 can win the **DFW**(n) objective

for n **large enough**

i.e. $n=(|S|-1)\cdot(1+|S|\cdot W)$

Proof uses cycle decomposition of outcomes

Relation between MP/TP and W.O.

Theorem

1. *If the answer to the one of window mean-payoff problems is* YES, *then the answer to the mean-payoff threshold problem for threshold zero is also* YES.

2. *If there exists* $\varepsilon > 0$ *s.t. the answer to the mean-payoff threshold problem for threshold* ε *is* YES, *then the answer to the* **BW** *problem is also* YES.

$\Longrightarrow$ **Window objectives** can be seen as **ε-approximations** of mean-payoff for large enough windows

<u>Solving</u> **FixedW**(n) for 1 dim.

Algorithm for **FixedW**(n)

Idea:

Follow the structure of definition

+

recurse on subgames

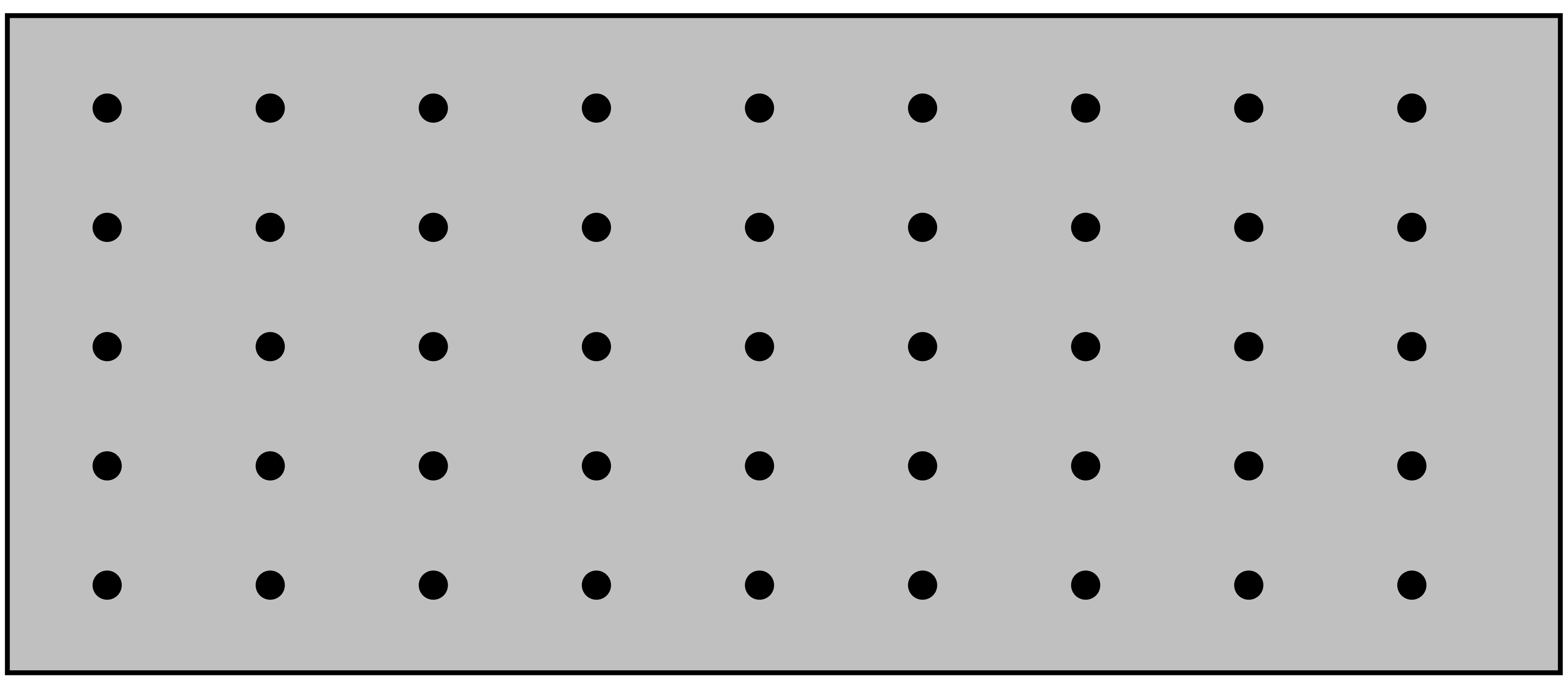
Algorithm for **FixedW**(n)

1) compute $W_1 = \{ s \mid s \models \langle\langle 1 \rangle\rangle \, \mathbf{GW}(n) \}$

i.e. the set of states from which Pl. 1 can force a positive sum within n steps (dynamic programming) $\mathbf{O}(n \cdot |G| \cdot \log(W))$

Algorithm for **FixedW**(n)

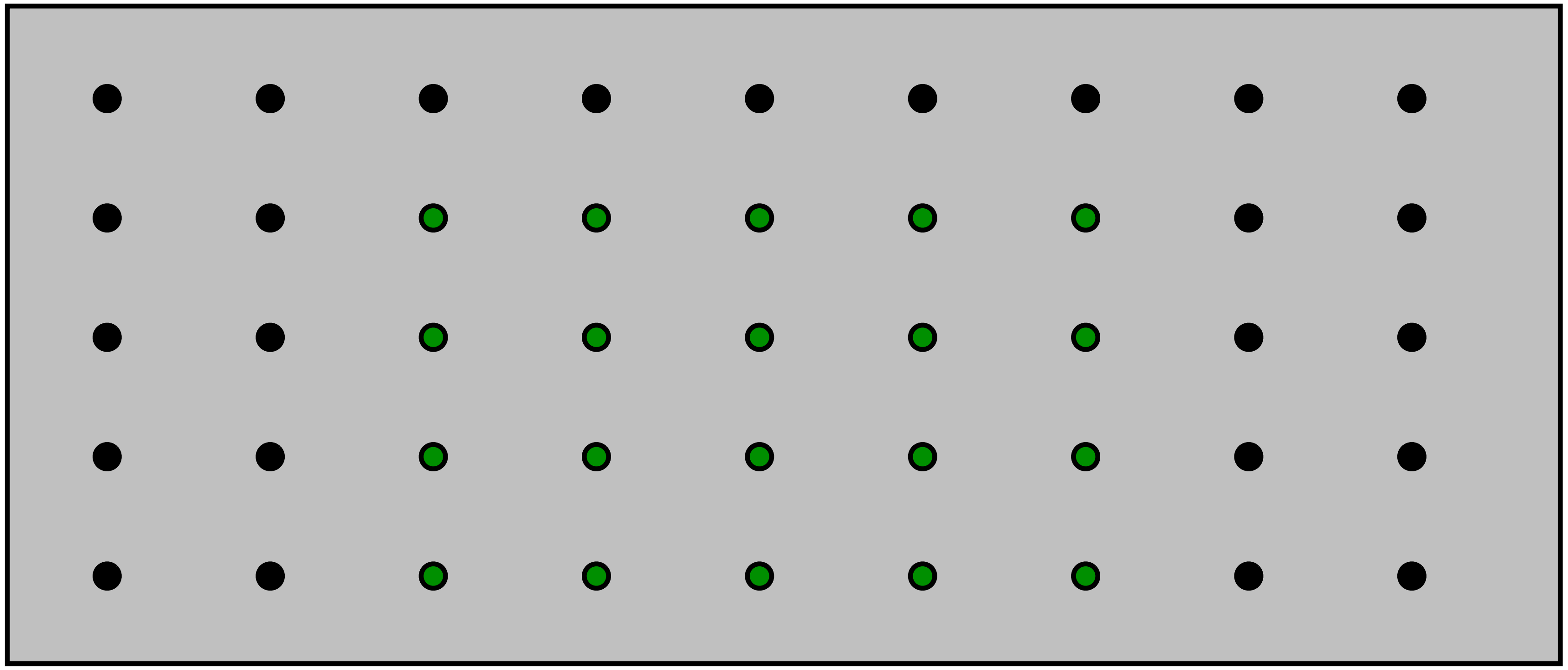
1) compute $W_1 = \{ s \mid s \models \langle\langle 1 \rangle\rangle \mathbf{GW}(n) \}$
i.e. the set of states from which Pl. 1 can force a positive sum within n steps (dynamic programming) $\mathbf{O}(n \cdot |G| \cdot \log(W))$



Algorithm for **FixedW**(n)

1) compute $W_1 = \{ s \mid s \models \langle\langle 1 \rangle\rangle \mathbf{GW}(n) \}$

i.e. the set of states from which Pl. 1 can force a positive sum within n steps (dynamic programming) $\mathbf{O}(n \cdot |G| \cdot \log(W))$



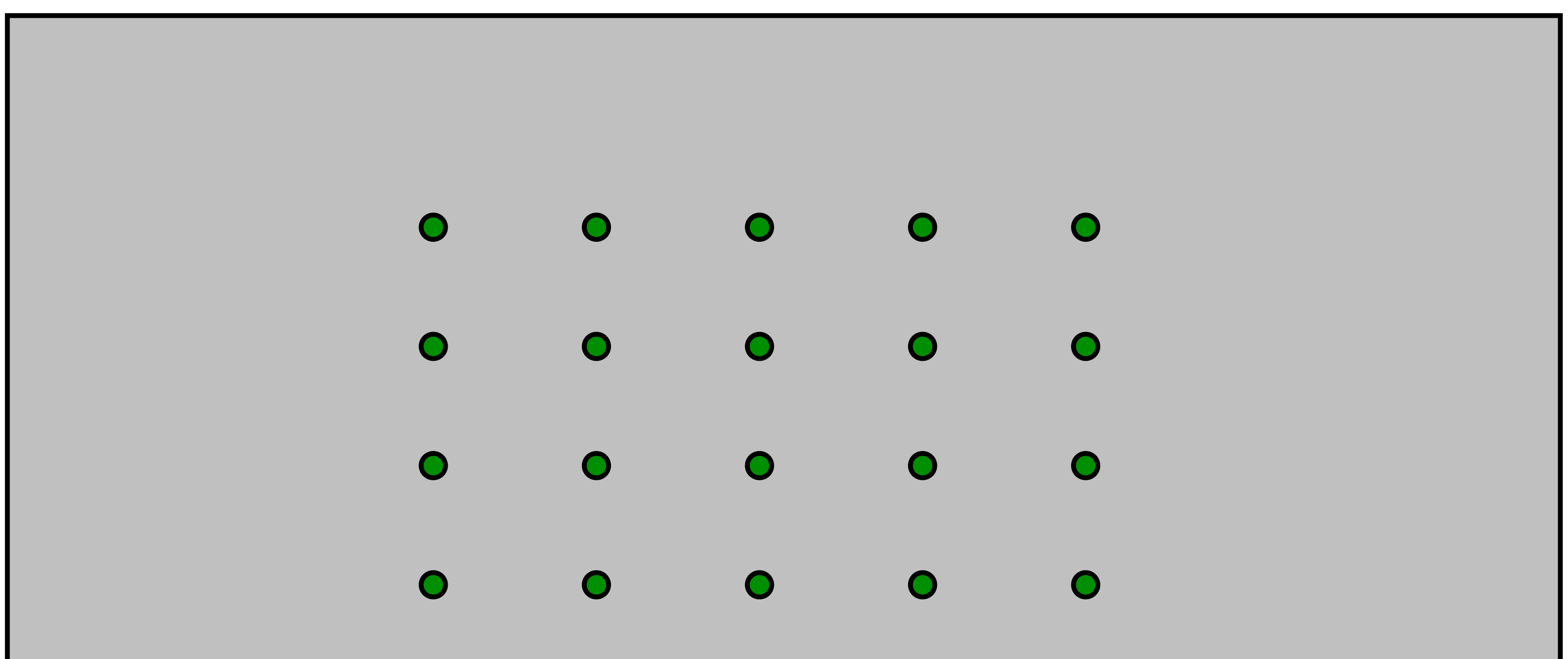
Algorithm for **FixedW**(n)

1) compute $W_1 = \{ s \mid s \models \langle\langle 1 \rangle\rangle\ \mathbf{GW}(n) \}$

i.e. the set of states from which Pl. 1 can force a positive sum within n steps (dynamic programming) $\mathbf{O}(n \cdot |G| \cdot \log(W))$

2) compute $W_2 = \{ s \mid s \models \langle\langle 1 \rangle\rangle \Box W_1 \}$ winning for **DFW**(n)

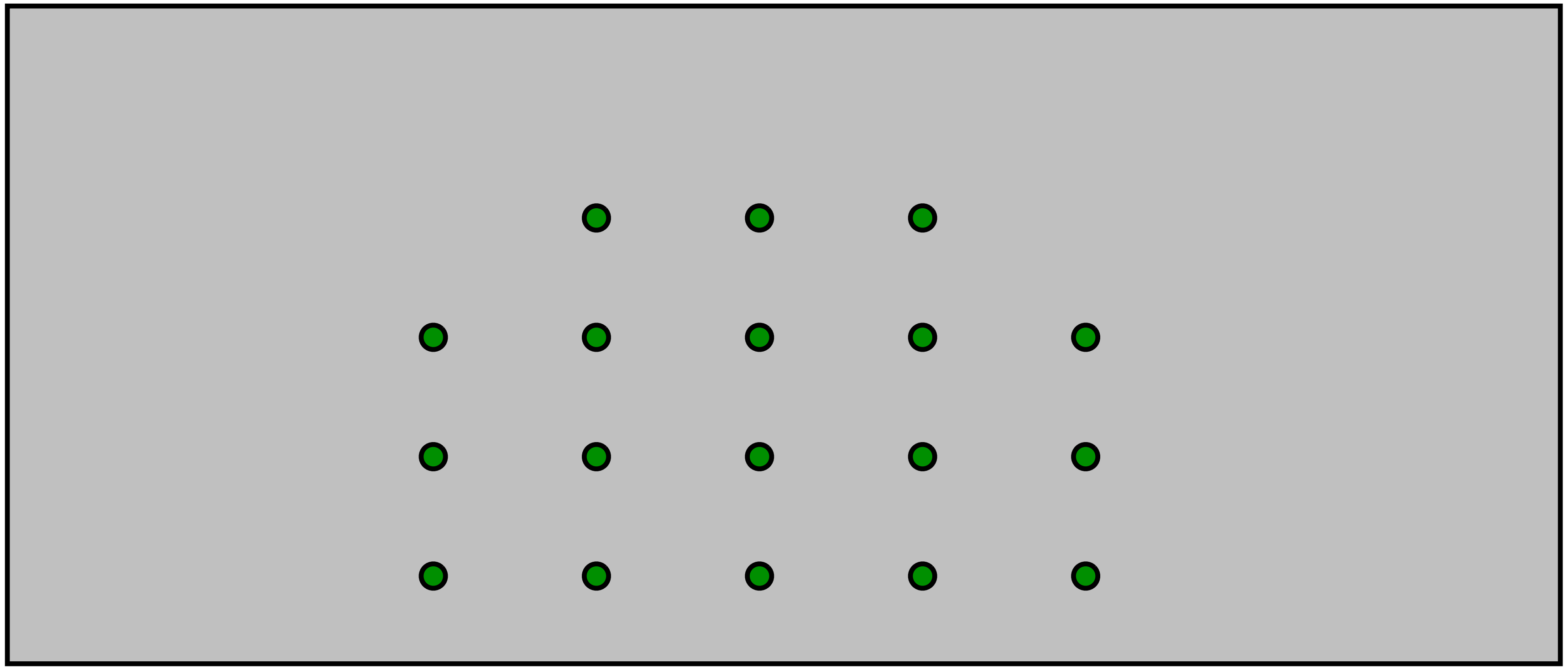
*i.e. the set of states from which Pl. 1 can win the **direct window objective** for size n. Complexity:* $\mathbf{O}(|G|)$



Algorithm for **FixedW**(n)

1) compute $W_1 = \{ s \mid s \models \langle\langle 1 \rangle\rangle \mathbf{GW}(n) \}$
i.e. the set of states from which Pl. 1 can force a positive sum within n steps (dynamic programming) $\mathbf{O}(n \cdot |G| \cdot \log(W))$

2) compute $W_2 = \{ s \mid s \models \langle\langle 1 \rangle\rangle \Box W_1 \}$ winning for **DFW**(n)
i.e. the set of states from which Pl. 1 can win the direct window objective for size n. Complexity: $\mathbf{O}(|G|)$



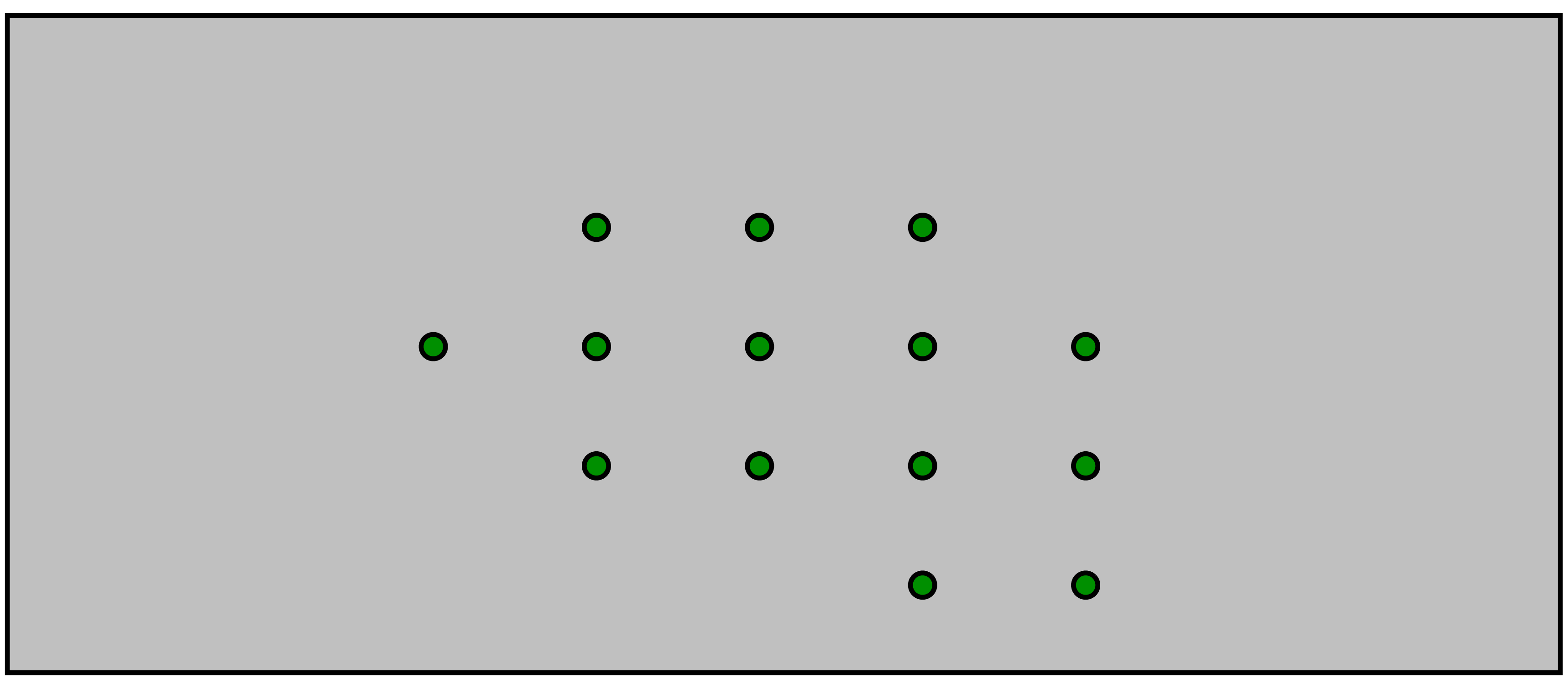
Algorithm for **FixedW**(n)

1) compute $W_1 = \{ s \mid s \vDash \langle\langle 1 \rangle\rangle\ \mathbf{GW}(n) \}$

i.e. the set of states from which Pl. 1 can force a positive sum within n steps (dynamic programming) $\mathbf{O}(n \cdot |G| \cdot \log(W))$

2) compute $W_2 = \{ s \mid s \vDash \langle\langle 1 \rangle\rangle \Box W_1 \}$ winning for **DFW**(n)

i.e. the set of states from which Pl. 1 can win the direct window objective for size n. Complexity: $\mathbf{O}(|G|)$



Algorithm for **FixedW**(n)

1) compute $W_1 = \{ s \mid s \models \langle\langle 1 \rangle\rangle\ \mathbf{GW}(n) \}$

i.e. the set of states from which Pl. 1 can force a positive sum within n steps (dynamic programming) $\mathbf{O}(n \cdot |G| \cdot \log(W))$

2) compute $W_2 = \{ s \mid s \models \langle\langle 1 \rangle\rangle \Box W_1 \}$

i.e. the set of states from which Pl. 1 can win the direct window objective for size n. Complexity: $\mathbf{O}(|G|)$

3) compute $W_3 = \{ s \mid s \models \langle\langle 1 \rangle\rangle \Diamond W_2 \}$ winning for **FixedW**(n)

*i.e. the **attractor** of W_2 is clearly winning for Pl. 1. Complexity:* $\mathbf{O}(|G|)$

Those states are **winning for Player 1**

So they should be **avoided at all cost by player 2**

Algorithm for **FixedW**(n)

1) compute $W_1 = \{ s \mid s \models \langle\langle 1 \rangle\rangle\ \mathbf{GW}(n) \}$

i.e. the set of states from which Pl. 1 can force a positive sum within n steps (dynamic programming) $\mathbf{O}(n{\cdot}|G|{\cdot}\log(W))$

2) compute $W_2 = \{ s \mid s \models \langle\langle 1 \rangle\rangle \Box W_1 \}$

i.e. the set of states from which Pl. 1 can win the direct window objective for size n. Complexity: $\mathbf{O}(|G|)$

3) compute $W_3 = \{ s \mid s \models \langle\langle 1 \rangle\rangle \Diamond W_2 \}$

i.e. the attractor of W_2 is clearly winning for Pl. 1. Complexity: $\mathbf{O}(|G|)$

4) Pl.2 should avoid W_3 at all cost: **recurse** on $S \backslash W_3$

The number of recursive calls is bounded by $\mathbf{O}(|G|)$

Overall complexity: *bounded by* $\mathbf{O}(|G|^3{\cdot}n{\cdot}\log(W))$

Solving **FixedW**(n)

Theorem

In two-player one-dimension games:

*(a) the fixed arbitrary window mean-payoff problem is decidable in time **polynomial** in the size of the game **and** the window size*

*(b) the fixed polynomial-size window mean-payoff problem is **P-complete***

*(c) Both players **require memory**, and memory of size **linear** in the size of the game and the size of the window is sufficient*

Main Results

<table>
<tr><th></th><th colspan="3">one-dimension</th><th colspan="3">k-dimension</th></tr>
<tr><th></th><th>complexity</th><th>$\mathcal{P}_1$ mem.</th><th>$\mathcal{P}_2$ mem.</th><th>complexity</th><th>$\mathcal{P}_1$ mem.</th><th>$\mathcal{P}_2$ mem.</th></tr>
<tr><td>$\underline{\text{MP}}$ / $\overline{\text{MP}}$</td><td>NP$\cap$coNP</td><td colspan="2">mem-less</td><td>coNP-c. / NP$\cap$coNP</td><td>infinite</td><td>mem-less</td></tr>
<tr><td>$\underline{\text{TP}}$ / $\overline{\text{TP}}$</td><td>NP$\cap$coNP</td><td colspan="2">mem-less</td><td>undec. (Thm. 1)</td><td>-</td><td>-</td></tr>
<tr><td>WMP: fixed polynomial window</td><td>P-c. (Thm. 2)</td><td colspan="2" rowspan="2">mem. req. $\leq$ linear$(\lvert S\rvert \cdot l_{\max})$ (Thm. 2)</td><td>PSPACE-h. (Thm. 4) EXP-easy (Thm. 4)</td><td colspan="2" rowspan="2">exponential (Thm. 4)</td></tr>
<tr><td>WMP: fixed arbitrary window</td><td>P$(\lvert G\rvert, l_{\max})$ (Thm. 2)</td><td>EXP-c. (Thm. 4)</td></tr>
<tr><td>WMP: bounded window problem</td><td>NP$\cap$coNP (Thm. 3)</td><td>mem-less (Thm. 3)</td><td>infinite (Thm. 3)</td><td>NPR-h. (Thm. 5)</td><td>-</td><td>-</td></tr>
</table>

More details in
http://arxiv.org/abs/1209.3234

The Complexity of Multi-Mean-Payoff and Multi-Energy Games⋆,⋆⋆

Yaron Velner[1], Krishnendu Chatterjee[2], Laurent Doyen[3], Thomas A. Henzinger[2], Alexander Rabinovich[1], and Jean-François Raskin[4]

[1] The Blavatnik School of Computer Science, Tel Aviv University, Israel
[2] IST Austria (Institute of Science and Technology Austria)
[3] LSV, ENS Cachan & CNRS, France
[4] Département d'Informatique, Université Libre de Bruxelles (U.L.B.)

Abstract. In mean-payoff games, the objective of the protagonist is to ensure that the limit average of an infinite sequence of numeric weights is nonnegative. In energy games, the objective is to ensure that the running sum of weights is always nonnegative. Multi-mean-payoff and multi-energy games replace individual weights by tuples, and the limit average (resp. running sum) of each coordinate must be (resp. remain) nonnegative. These games have applications in the synthesis of resource-bounded processes with multiple resources.
We prove the finite-memory determinacy of multi-energy games and show the inter-reducibility of multi-mean-payoff and multi-energy games for finite-memory strategies. We also improve the computational complexity for solving both classes of games with finite-memory strategies: while the previously best known upper bound was EXPSPACE, and no lower bound was known, we give an optimal coNP-complete bound. For memoryless strategies, we show that the problem of deciding the existence of a winning strategy for the protagonist is NP-complete. Finally we present the first solution of multi-mean-payoff games with infinite-memory strategies. We show that multi-mean-payoff games with mean-payoff-sup objectives can be decided in NP ∩ coNP, whereas multi-mean-payoff games with mean-payoff-inf objectives are coNP-complete.

Keywords: *Games on graphs; mean-payoff objectives; energy objectives; multi-dimensional objectives.*

1 Introduction

Graph games and multi-objectives. Two-player games on graphs are central in many appli-
computer science. For example, in the synthesis problem, implementatio
obtained from winning strategies in games with a qualit
specification [22, 21, 1]. In these applicatio
determines which player wi
natural mod

and in

Looking at Mean-Payoff and Total-Payoff through Windows

Krishnendu Chatterjee[1,⋆], Laurent Doyen[2], Mickael Randour[3,†], and Jean-François Raskin[4,‡]

[1] IST Austria (Institute of Science and Technology Austria)
[2] LSV - ENS Cachan, France
[3] Computer Science Department, Université de Mons (UMONS), Belgium
[4] Département d'Informatique, Université Libre de Bruxelles (U.L.B.), Belgium

Abstract. We consider two-player games played on weighted directed graphs with mean-payoff and total-payoff objectives, which are two classical quantitative objectives. While for single dimensional objectives all results for mean-payoff and total-payoff coincide, we show that in contrast to multi-dimensional mean-payoff games that are known to be coNP-complete, multi-dimensional total-payoff games are undecidable. We introduce conservative approximations of these objectives, where the payoff is considered over a local finite window sliding along a play, instead of the whole play. For single dimension, we show that (*i*) if the window size is polynomial, then the problem can be solved in polynomial time, and (*ii*) the existence of a bounded window can be decided in NP ∩ coNP, and is at least as hard as solving mean-payoff games. For multiple dimensions, we show that (*i*) the problem with fixed window size is EXPTIME-complete, and (*ii*) there is no primitive-recursive algorithm to decide the existence of a bounded window.

1 Introduction

Mean-payoff and total-payoff games. Two-player mean-payoff and total-payoff games are played on finite weighted directed graphs (in which every edge has an integer weight) with two types of vertices: in player-1 vertices, player 1 chooses the successor vertex from the set of outgoing edges; in player-2 vertices, player 2 does likewise. The game results in an infinite path (called a *play*) through the graph. The mean-payoff (resp. total-payoff) value of a play is the long-run average (resp. sum) of the edge-weights along the

see on arXiv:
http://arxiv.org/abs/1302.4248

Conclusion

- **Quantitative games**: energy games (+generalizations by Larsen, Bouyer,...), mean-payoff games, total-payoff games, ... is an active research area, part of a larger program...
- **From quality to quantity**: broad effort in order to lift boolean verification/ synthesis to quantitative verification/synthesis, e.g.:
    - quantitative languages (Henzinger et al.) def. by **weighted automata**
- In this talk, we have shown:
    - MP games can be extended to multi-dim.
    - Space for alternative objectives: e.g. **window objectives**
        - "approximations" of MP/TP
        - natural definitions and interesting algorithms