# 6.853: Topics in Algorithmic Game Theory

## Lecture 10

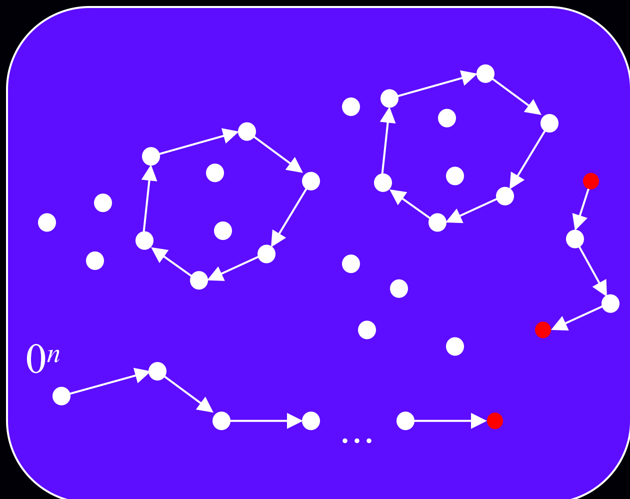
**Fall 2011**

*Constantinos Daskalakis*

*Towards PPAD-hardness of SPERNER, BROUWER, NASH…*

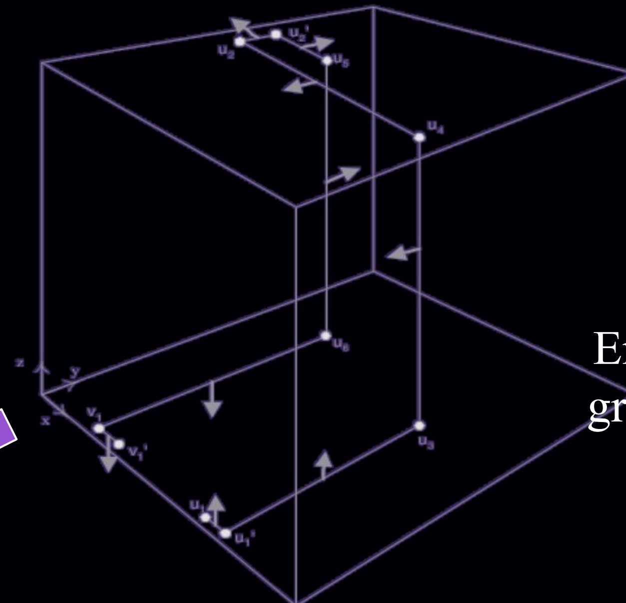
# The PLAN

DGP = Daskalakis, Goldberg, Papadimitriou
DP = Daskalakis, Papadimitriou
CD = Chen, Deng



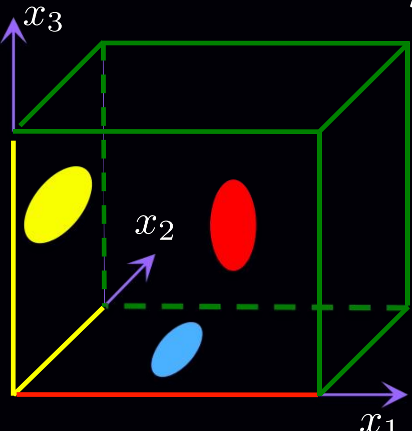

Generic PPAD

[Pap '94]
[DGP '05]



Embed PPAD graph in $[0,1]^3$

[DGP '05]




3D-SPERNER

[DGP '05]

p.w. linear BROUWER

[DGP '05]

multi-player NASH

[DGP '05]

4-player NASH

[DP '05]
[CD'05]

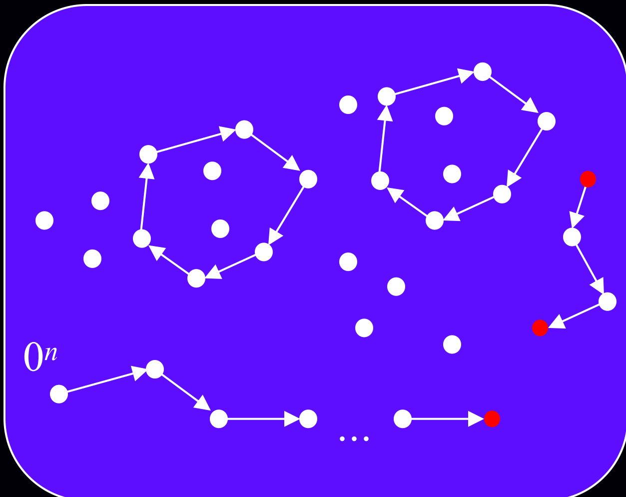
3-player NASH

[CD'06]

2-player NASH

# *Last Lecture*

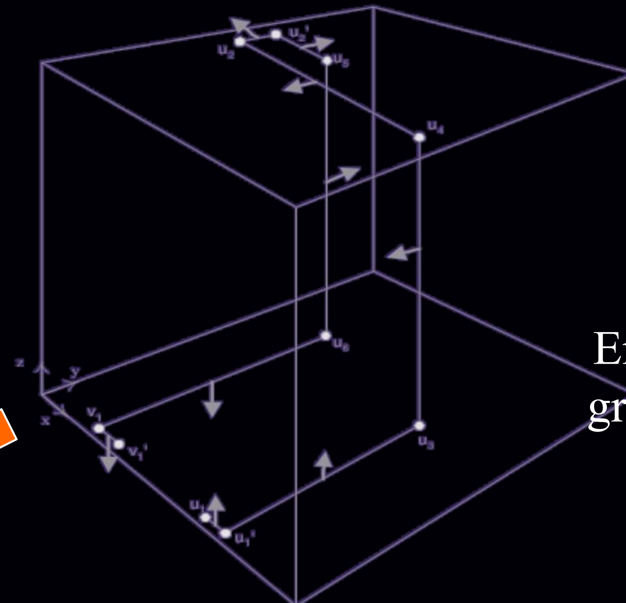
DGP = Daskalakis, Goldberg, Papadimitriou
DP = Daskalakis, Papadimitriou
CD = Chen, Deng




Generic PPAD

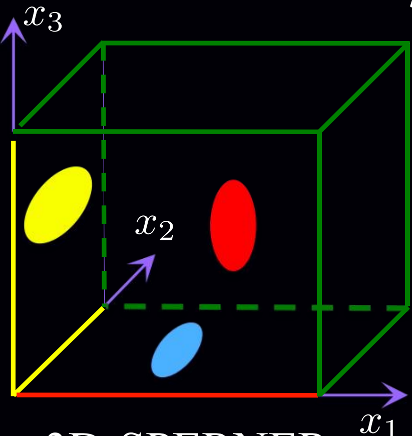
[Pap '94]

[DGP '05]



Embed PPAD graph in $[0,1]^3$

[DGP '05]




3D-SPERNER

[DGP '05]

p.w. linear BROUWER

[DGP '05]

multi-player NASH

[DGP '05]

4-player NASH

[DP '05]
[CD'05]

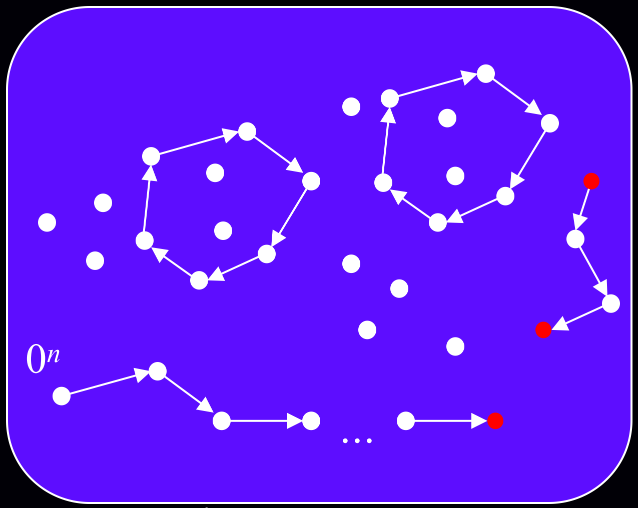
3-player NASH

[CD'06]

2-player NASH

# *This Lecture*

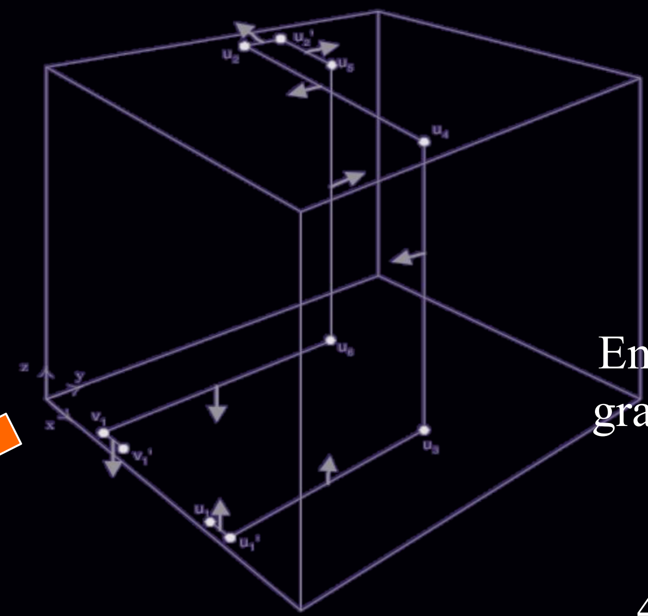
DGP = Daskalakis, Goldberg, Papadimitriou
DP = Daskalakis, Papadimitriou
CD = Chen, Deng




Generic PPAD

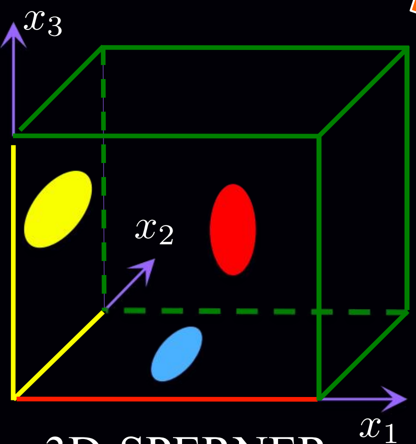
[Pap '94]

[DGP '05]



Embed PPAD graph in $[0,1]^3$

[DGP '05]




3D-SPERNER

[DGP '05]

p.w. linear BROUWER

[DGP '05]

multi-player NASH

[DGP '05]

4-player NASH

[DP '05]
[CD'05]

3-player NASH

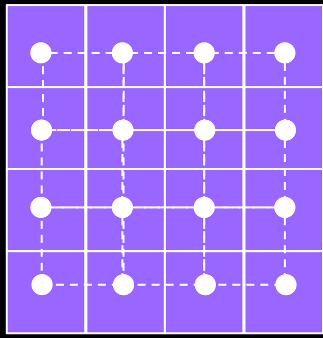
[CD'06]

2-player NASH

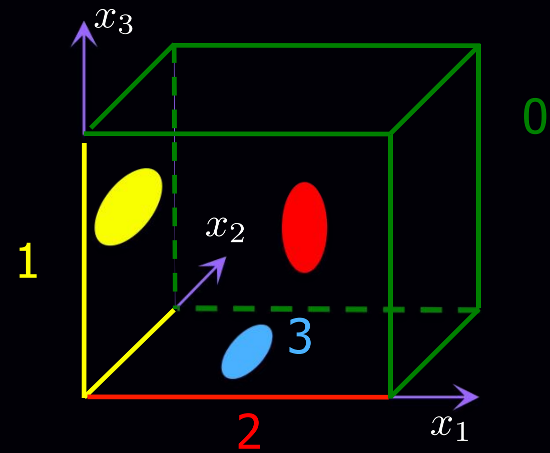
# *(dual) 3-d Sperner*

For convenience we shall work with the dual simplicization:

*a. Instead of coloring vertices of the standard cubeletes (the points of the cube whose coordinates are integer multiples of $2^{-m}$), we color the centers of these cubelets; i.e. we work with the dual graph.*



*b. Boundary Coloring:*

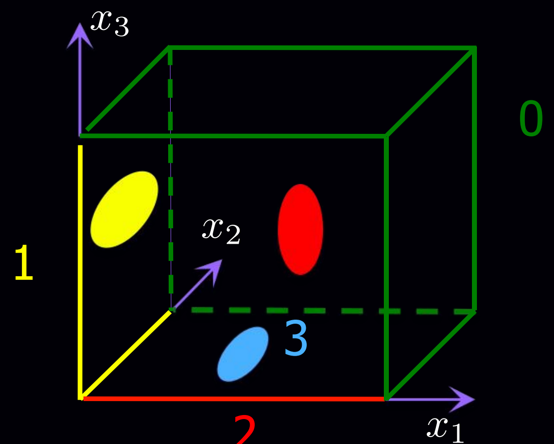



*c. Solution to dual-SPERNER: a vertex of the standard subdivision such that all colors are present among the centers of the cubelets using this vertex as a corner. Such vertex is called panchromatic.*

Lemma: If the canonical simplicization of the dual graph has a panchromatic simplex, then this simplex contains a vertex of the subdivision of the primal graph that is panchromatic.

PPAD

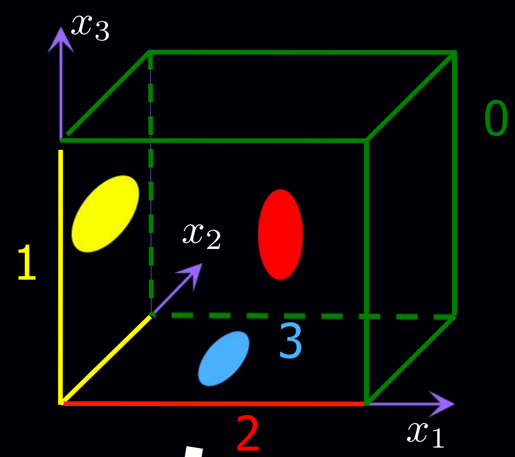
SPERNER



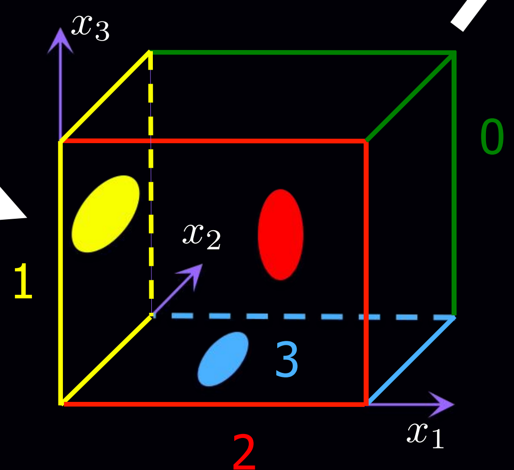

*N.B.: In the resulting PPAD-hard SPERNER instance, most of the cube is colored **0**, except for the cubelets around the single dimensional subset L of the cube, where the PPAD graph was embedded.*

## *PPAD-completeness of BROUWER*

# *(special) SPERNER* ⟹ *BROUWER*



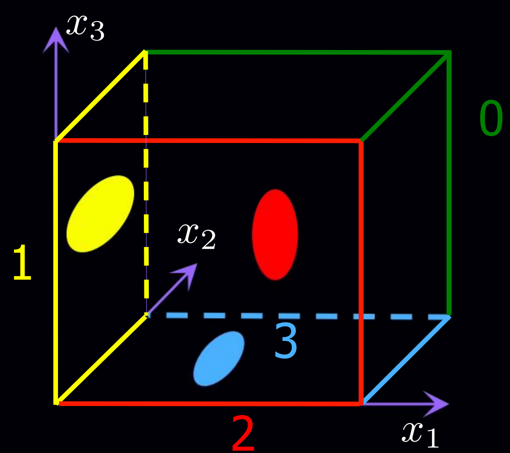

*boundary coloring tweaking*




**Claim:** Boundary coloring is not a legal Sperner coloring anymore, but no new panchromatic points were introduced by the modification.

**Proof:** The points that (were not but) could potentially become panchromatic after the modification are those with: $x_1$, $x_2$, or $x_3$=1-$2^{-m}$. But since the ambient space is colored green and the line L is far from the boundary, this won't happen.

# *(special) SPERNER* → *BROUWER*




- Define BROUWER instance on the (slightly smaller) cube defined by the convex hull of the centers of the cubelets. This is thinner by $2^{-m}$ in each dimension.

- Colors correspond to direction of the displacement vector $f(x) - x$:

color 0 (ambient color) $\rightsquigarrow$ $(-1, -1, -1) \times \alpha$

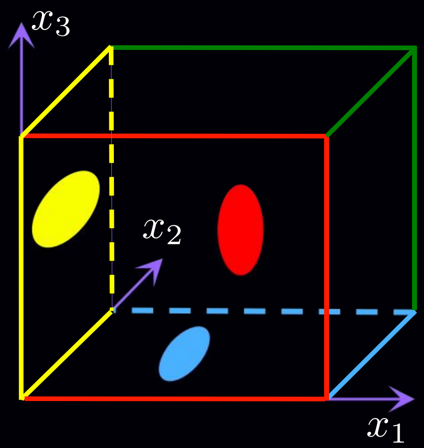
color 1 $\rightsquigarrow$ $(1, 0, 0) \times \alpha$

color 2 $\rightsquigarrow$ $(0, 1, 0) \times \alpha$

color 3 $\rightsquigarrow$ $(0, 0, 1) \times \alpha$

$$\boxed{\alpha = 2^{-2m}}$$

# (Special) SPERNER ⟶ BROUWER

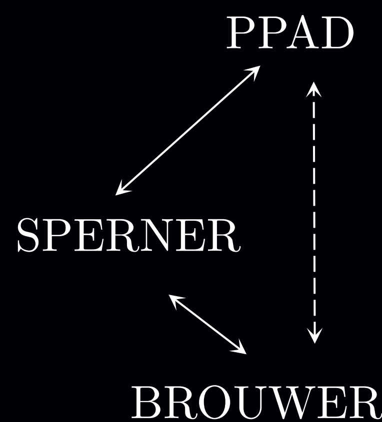



$f$ is extended on the remaining cube by interpolation: The cube is triangulated in the canonical way. To compute the displacement of $f$ at some point $x$, we find the simplex $S$ to which $x$ belongs. Then

if $x = \sum_{i=1}^{4} w_i \cdot x_i$ , where $x_i$ are the corners of $S$, we define :

$$f(x) - x := \sum_{i=1}^{4} w_i \cdot (f(x_i) - x_i)$$

**Claim:** Let $x$ be a $2^{-3m}$-approximate Brouwer Fixed Point of $f$. Then the corners of the simplex $S$ containing $x$ must have all colors/displacements.


PPAD
SPERNER
BROUWER

*PPAD-completeness of NASH*

# *(Special) BROUWER* ⇒ *NASH*

**Initial thoughts:** *BROUWER, SPERNER as well as END OF THE LINE are defined in terms of explicit circuits (for computing the function value, coloring, or candidate next/previous nodes) specified in the description of the instance.*

*In usual NP reductions, the computations performed by the gates in the circuits of the source problem need to somehow be simulated in the target problem.*

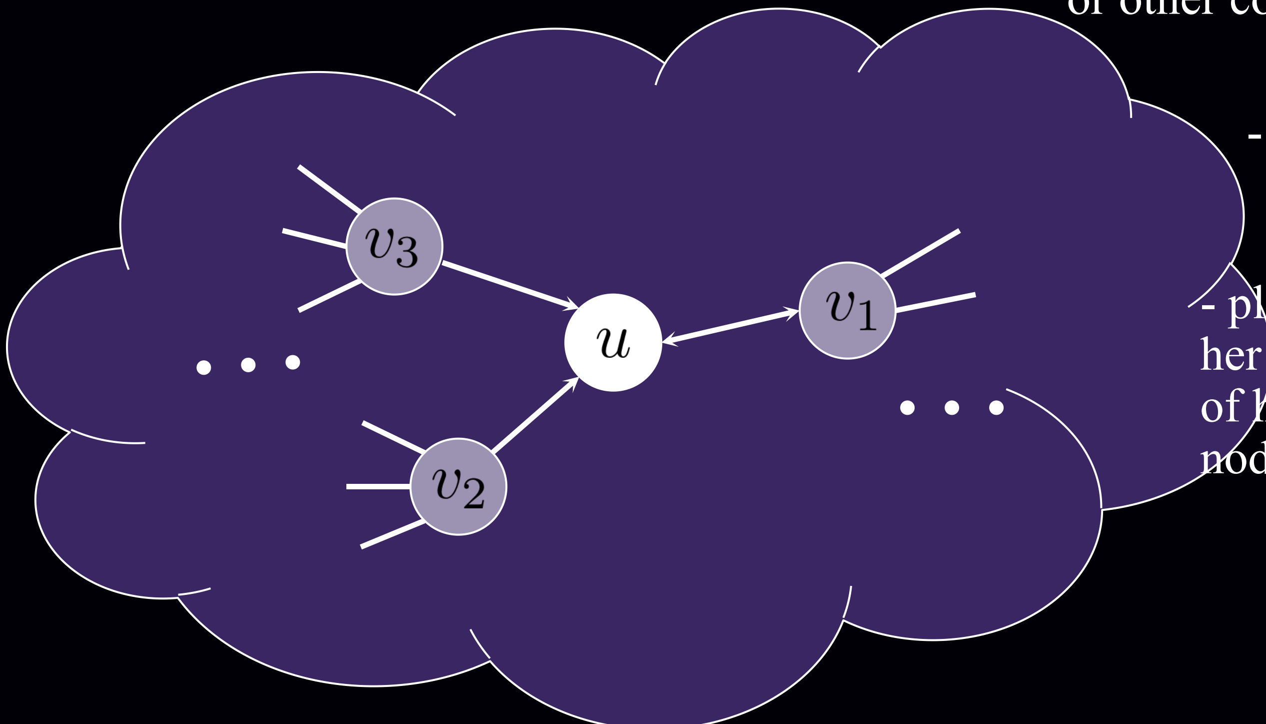
*The trouble with NASH is that no circuit is explicitly given in the description of a game.*

*On the other hand, in many FNP-complete problems, e.g. Vertex Cover, we do not have a circuit in the definition of the instance either. But at least we have a combinatorial object to work with, such as a graph, which isn't the case here either…*

# *(Special) BROUWER* ➡ *NASH*

Introducing a graph structure, via *graphical games*.

defined to capture sparse player interactions, such as those arising under geographical, communication or other constraints.

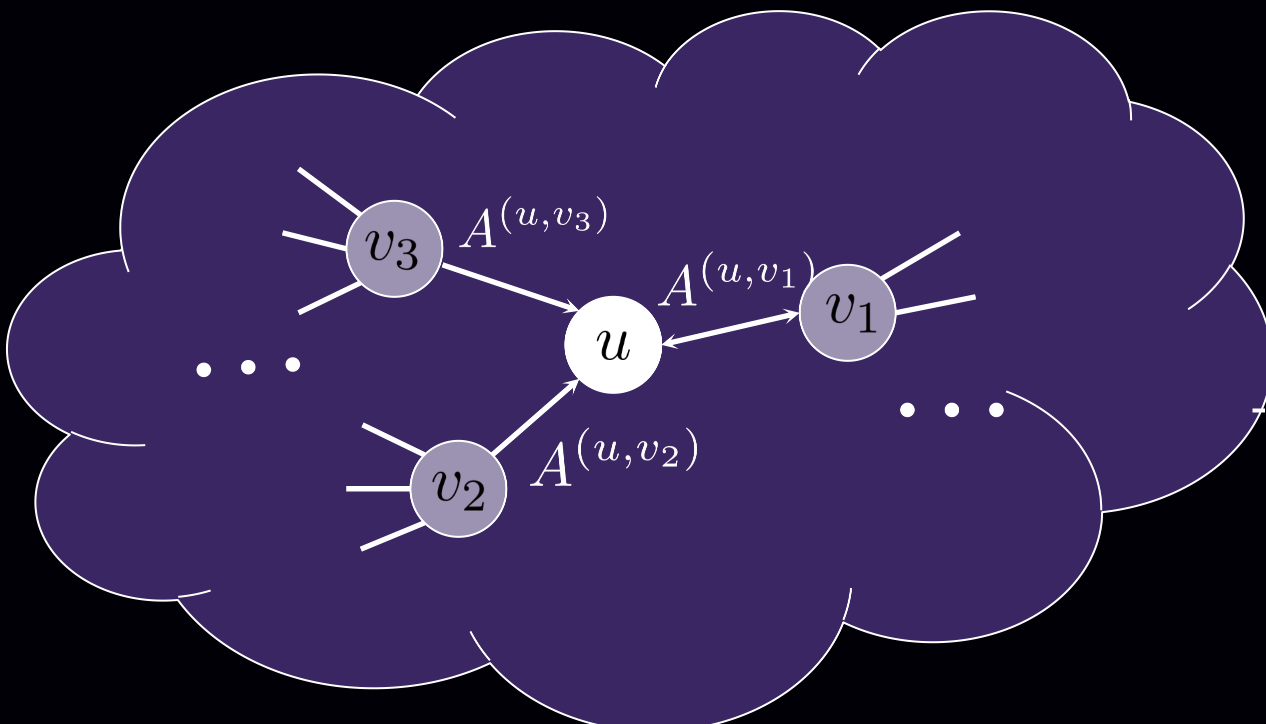



- players are nodes in a graph

- player's payoff is only affected by her own strategy and the strategies of her in-neighbors in the graph (i.e. nodes pointing to her)

# *(Special) BROUWER* ⟶ *NASH*

In particular, we restrict ourselves to the special class of separable multiplayer games, aka polymatrix games, which we saw earlier in the course. These are just graphical games with edge-wise separable utility functions.




- edges are 2-player games

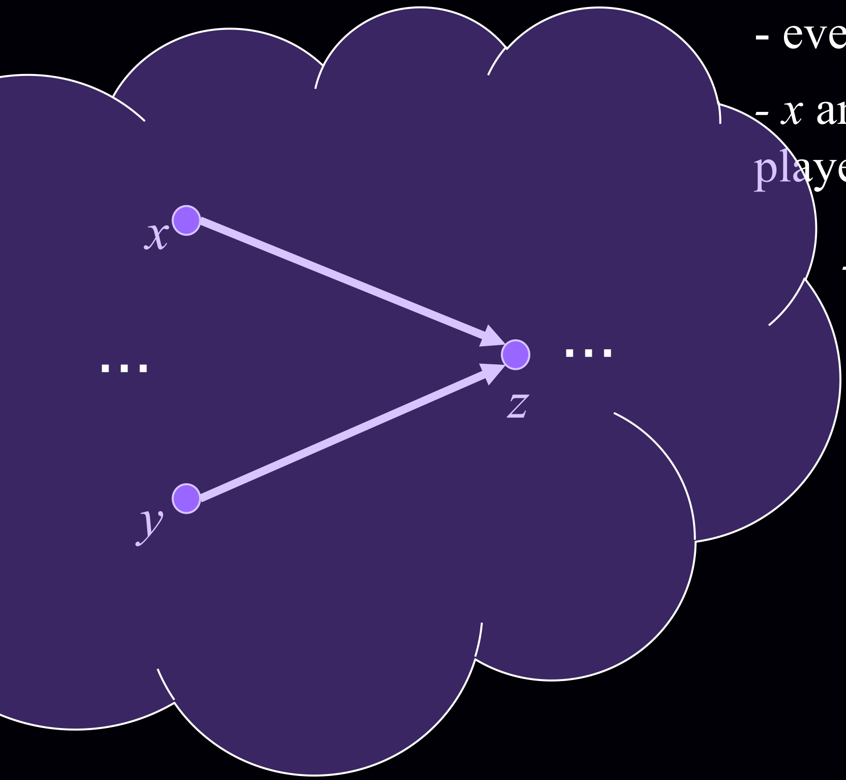
- player's payoff is the sum of payoffs from all adjacent edges

$$\sum_{i=1}^{3} x_u^{\mathrm{T}} A^{(u,v_i)} x_{v_i}$$

*Can games perform conventional binary computation?*

# *Binary* *Computation with Games*

- 3 players: $x$, $y$, $z$
  *(may or may not be part of a larger graphical game)*
- every player has strategy set {0, 1}
- $x$ and $y$ do not care about $z$ (but potentially care about other players), while $z$ cares about $x$ and $y$ (but no other player)




- $z$'s payoff table:

**separable**

$z : 0$

| | $y : 0$ | $y : 1$ |
|---|---|---|
| $x : 0$ | 1 | 0.5 |
| $x : 1$ | 0.5 | 0 |

$z : 1$

| | $y : 0$ | $y : 1$ |
|---|---|---|
| $x : 0$ | 0 | 1 |
| $x : 1$ | 1 | 2 |

**Claim**: In any Nash equilibrium where $\Pr[x{:}1], \Pr[y{:}1] \in \{0,1\}$, we have:

$$\Pr[z : 1] = \Pr[x : 1] \vee \Pr[y : 1].$$

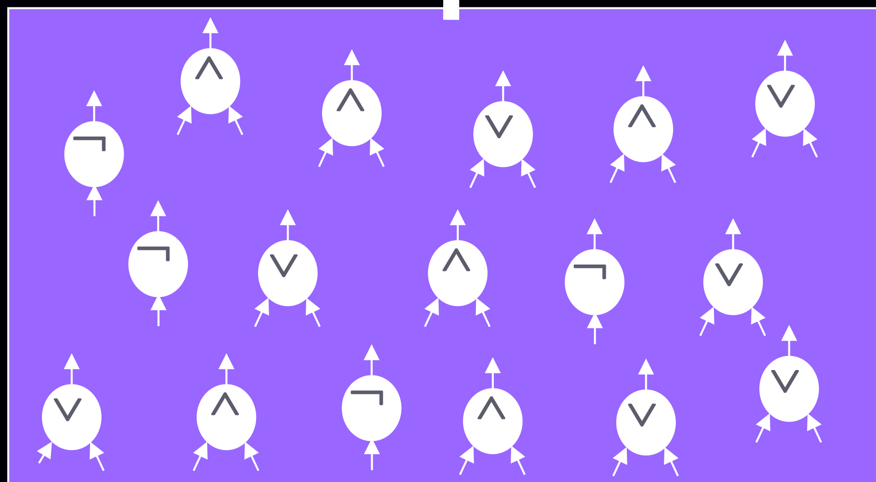
*So we obtained an OR gate, and we can similarly obtain AND and NOT gates.*

# *A possible PPAD-hardness reduction*

**does not exist** unconditionally

*if input is 0 enter a mode with no Nash eq.*

"output" 1 if it is, and 0 if it is not



**exists**

check if the point $(i, j, k) \cdot 2^{-m}$ is panchromatic; all this is done in pure strategies, since the "input" to this part is in pure strategies

$x_1$ … $x_m$ $y_1$ … $y_m$ $z_1$ … $z_m$

interpret these pure strategies as the coordinates $i, j, k$ of a point in the subdivision of the hypercube

**exists**

game gadget whose purpose is to have players $x_1, \ldots, z_m$ play **pure strategies** in any Nash equilibrium *bottom line:*

- *need feedback in the circuit*
- *a reduction restricted to pure strategy equilibria is likely to fail*
- *real numbers seem to play a fundamental role in the reduction*

*Can games do* ***real*** *arithmetic?*

*What in a Nash equilibrium is capable of storing reals?*

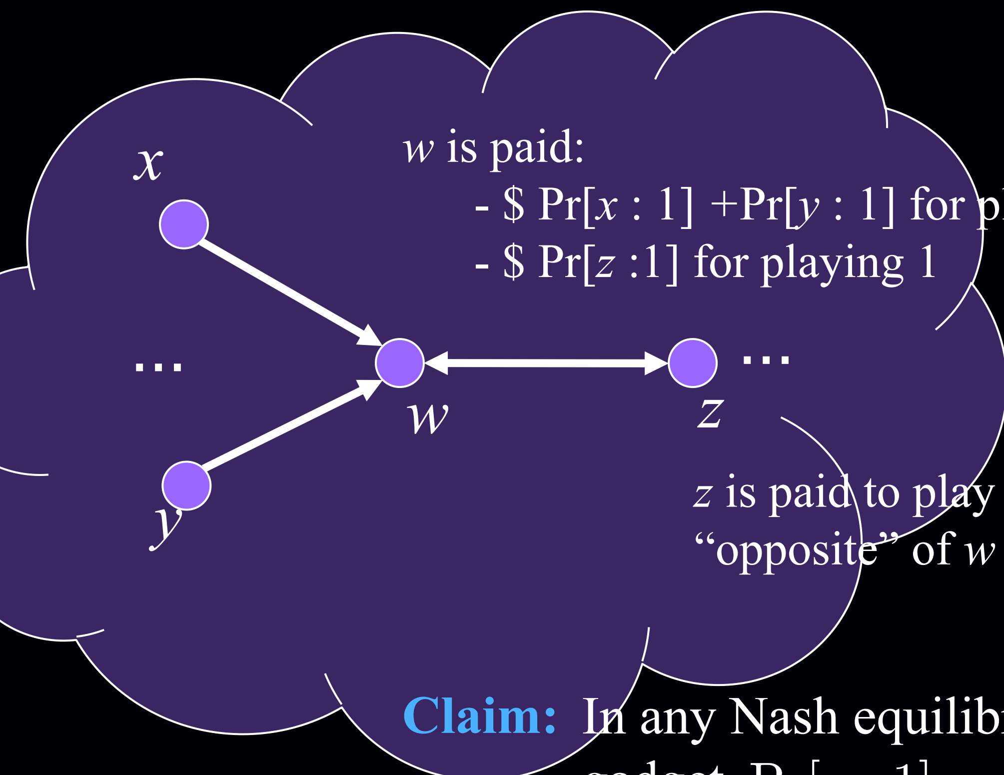
# *Games that do **real** arithmetic*

**separable**

Suppose two strategies per player: {0,1}

then mixed strategy ≡ a number in [0,1] (the probability of playing 1)

e.g. *addition game*




*w* is paid:

- \$ Pr[*x* : 1] +Pr[*y* : 1] for playing 0
- \$ Pr[*z* :1] for playing 1

*z* is paid to play the "opposite" of *w*

$$\boxed{\begin{aligned} u(w:0) &= \Pr[x:1] + \Pr[y:1] \\ u(w:1) &= \Pr[z:1] \end{aligned}}$$

$$\boxed{\begin{aligned} u(z:0) &= 0.5 \\ u(z:1) &= 1 - Pr[w:1] \end{aligned}}$$

**Claim:** In any Nash equilibrium of a game containing the above gadget $\Pr[z:1] = \min\{\Pr[x:1] + \Pr[y:1], 1\}$.

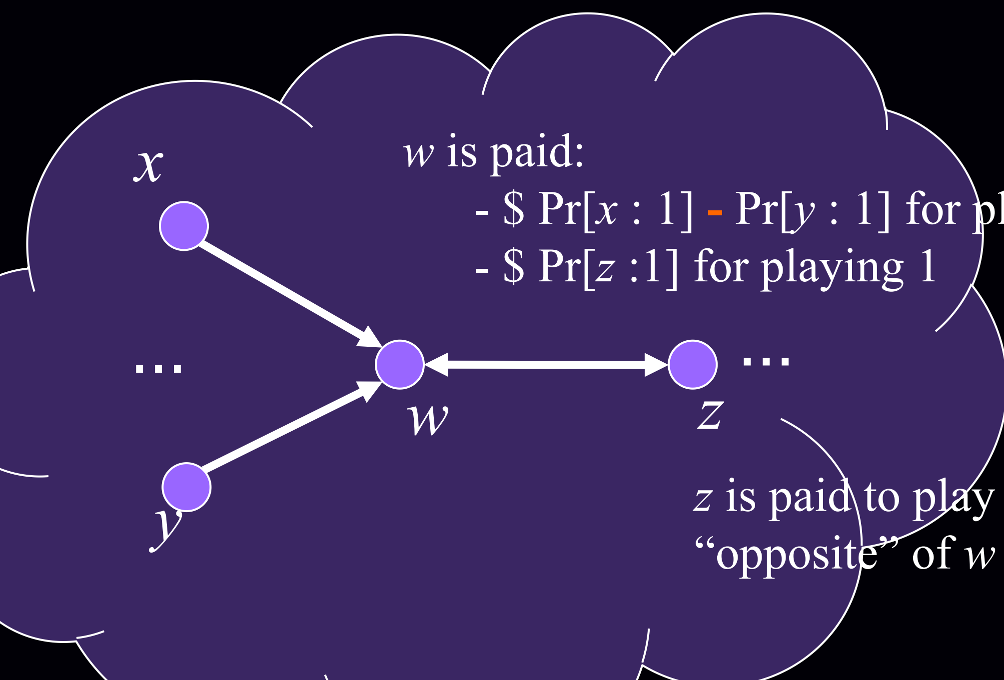
# *Games that do **real** arithmetic*

**separable**

Suppose two strategies per player: {0,1}

then mixed strategy ≡ a number in [0,1] (the probability of playing 1)

e.g. *subtraction*




$$u(w:0) = \Pr[x:1] - \Pr[y:1]$$
$$u(w:1) = \Pr[z:1]$$

$$u(z:0) = 0.5$$
$$u(z:1) = 1 - Pr[w:1]$$

**Claim:** In any Nash equilibrium of a game containing the above gadget $\Pr[z:1] = \max\{0, \Pr[x:1] - \Pr[y:1]\}$.

*From now on, use the name of the node and the probability of that node playing 1 interchangeably.*

$$x \longleftrightarrow \Pr[x:1]$$

# *Games that do **real** arithmetic*

copy : $z = x$

addition : $z = \min\{1, x + y\}$

subtraction : $z = \max\{0, x - y\}$

set equal to a constant : $z = \alpha, \text{ for any } \alpha \in [0,1]$

multiply by constant : $z = \min\{1, \alpha \cdot x\}$

**separable**

---

can also do multiplication $z = x \cdot y$

**non separable!**

*won't be used in our reduction*

# *Comparison Gadget*

$$z = \begin{cases} 1, & \text{if } x > y \\ 0, & \text{if } x < y \\ *, & \text{if } x = y \end{cases}$$

brittleness

Unfortunately, it has to be brittle! (**Exercise**)

# *Our Gates*

Constants:

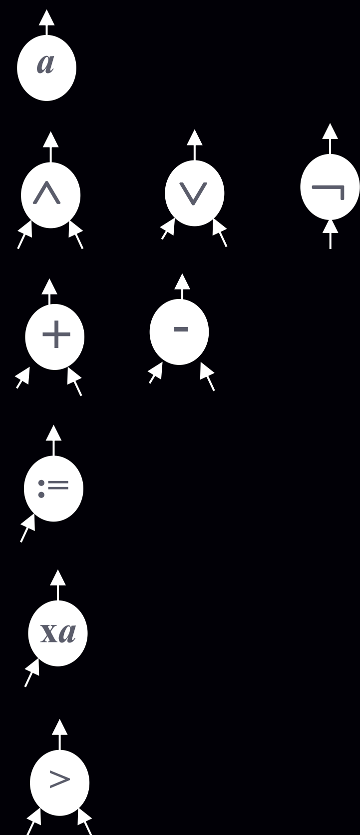
Binary gates:

Linear gates:

Copy gate:

Scale:

Brittle Comparison:




with truncation at 0, 1

*any circuit using these gates can be implemented with a separable game*

*need not be a DAG circuit, i.e. feedback is allowed*

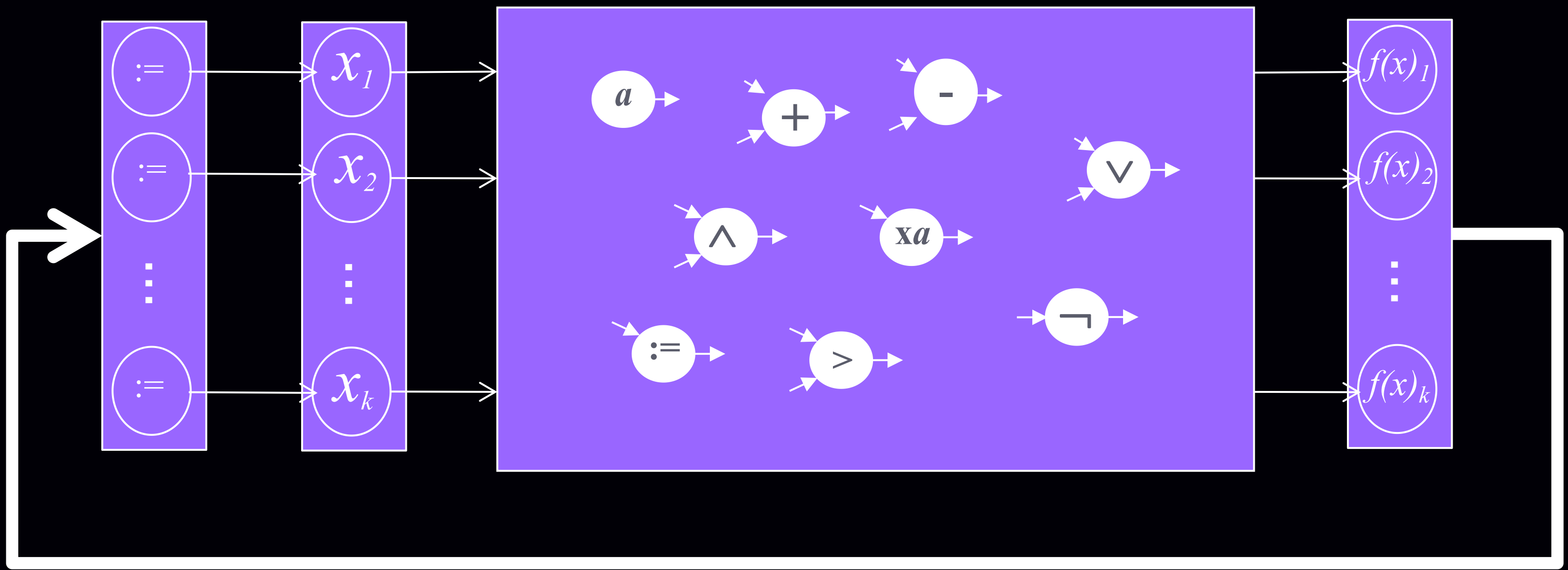
let's call any such circuit a *game-inspired straight-line program*

# *Fixed Point Computation*

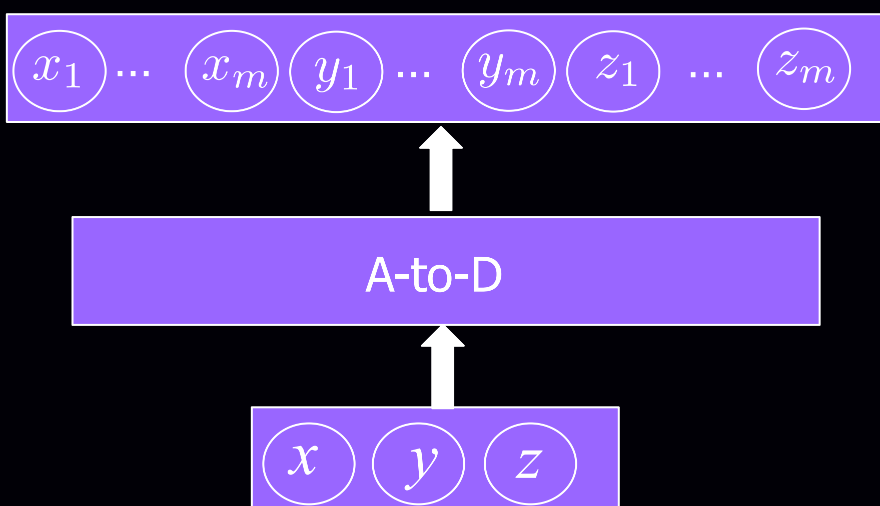
Suppose function $f : [0,1]^k \rightarrow [0,1]^k$ is computed by a game-inspired straight-line program.

➔ Can construct a polymatrix-game whose Nash equilibria are in many-to-one and onto correspondence with the fixed points of *f*.

➔ Can forget about games, and try to reduce PPAD to finding a fixed point of a game-inspired straight-line program.



$\begin{pmatrix}\text{4-displacement}\\ \text{p.w. linear}\end{pmatrix}$ BROUWER → *fixed point of game-inspired straight-line program*




extract $m$ bits from each of $x$, $y$, $z$

three variables (players) whose mixed strategies represent a point in $[0,1]^3$

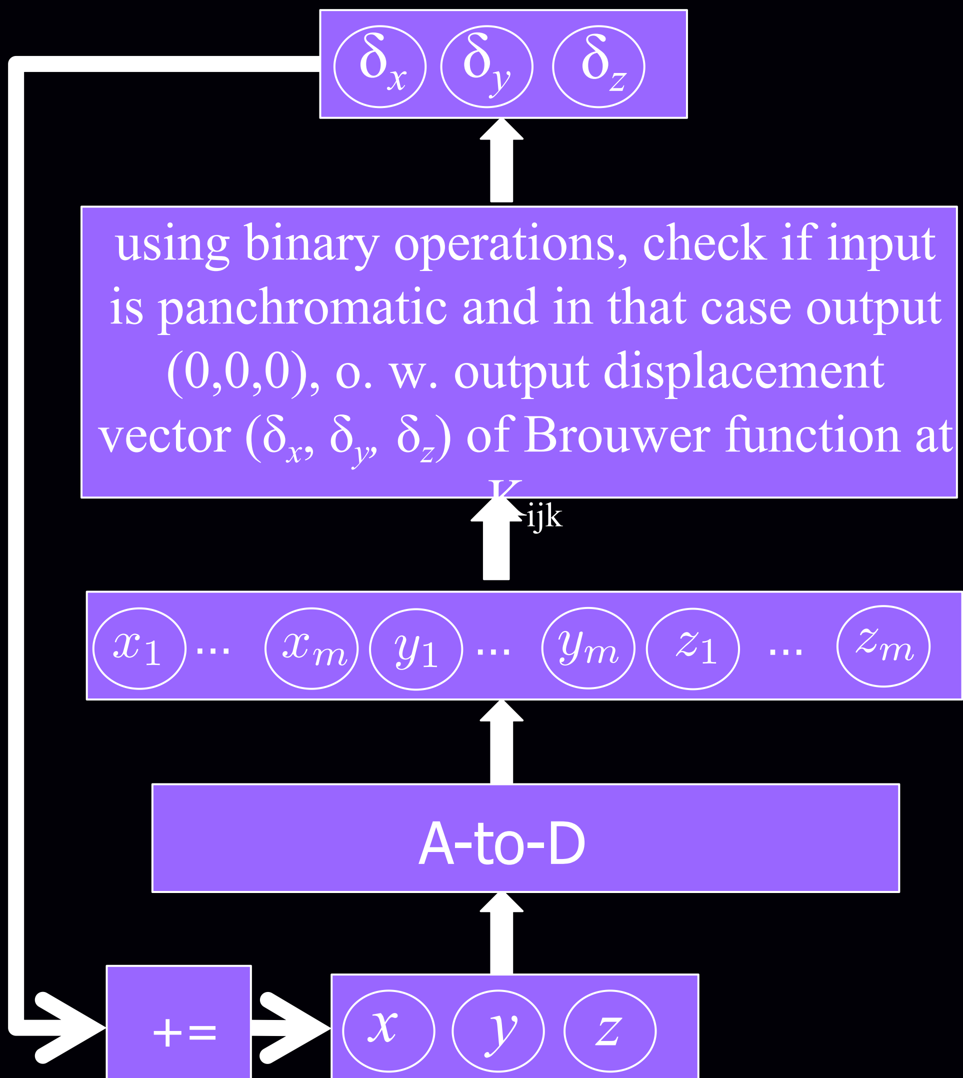
# Analog-to-Digital

$$v_1 = x;$$
for $i = 1, \ldots, m$ do:
$$x_i := (2^{-i} < v_i);\ v_{i+1} := v_i - x_i \cdot 2^{-i};$$
similarly for $y$ and $z$;

Can implement the above computation via a game-inspired straight-line program.

The output of the program is always 0/1, except if $x$, $y$ or $z$ is an integer multiple of $2^{-m}$.

(4-displacement p.w. linear) BROUWER → *fixed point of game-inspired straight-line program*




the displacement vector satisfies

$$(\delta_x, \delta_y, \delta_z) + (x, y, z) \in [0,1]^3$$

(because Brouwer function maps $[0,1]^3$ to $[0,1]^3$

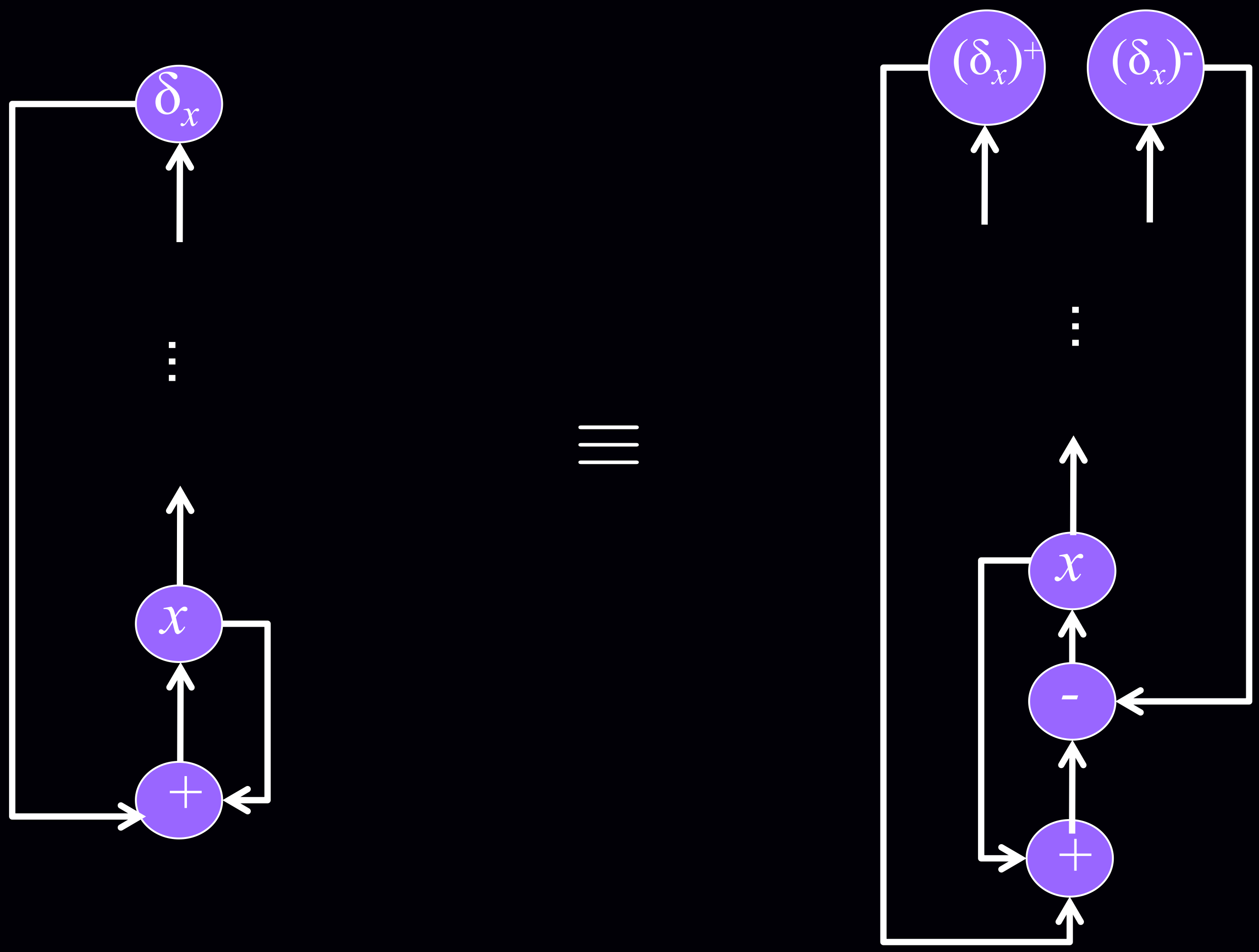
(hopefully) represents a point of the subdivision

extract $m$ bits from each of $x$, $y$, $z$

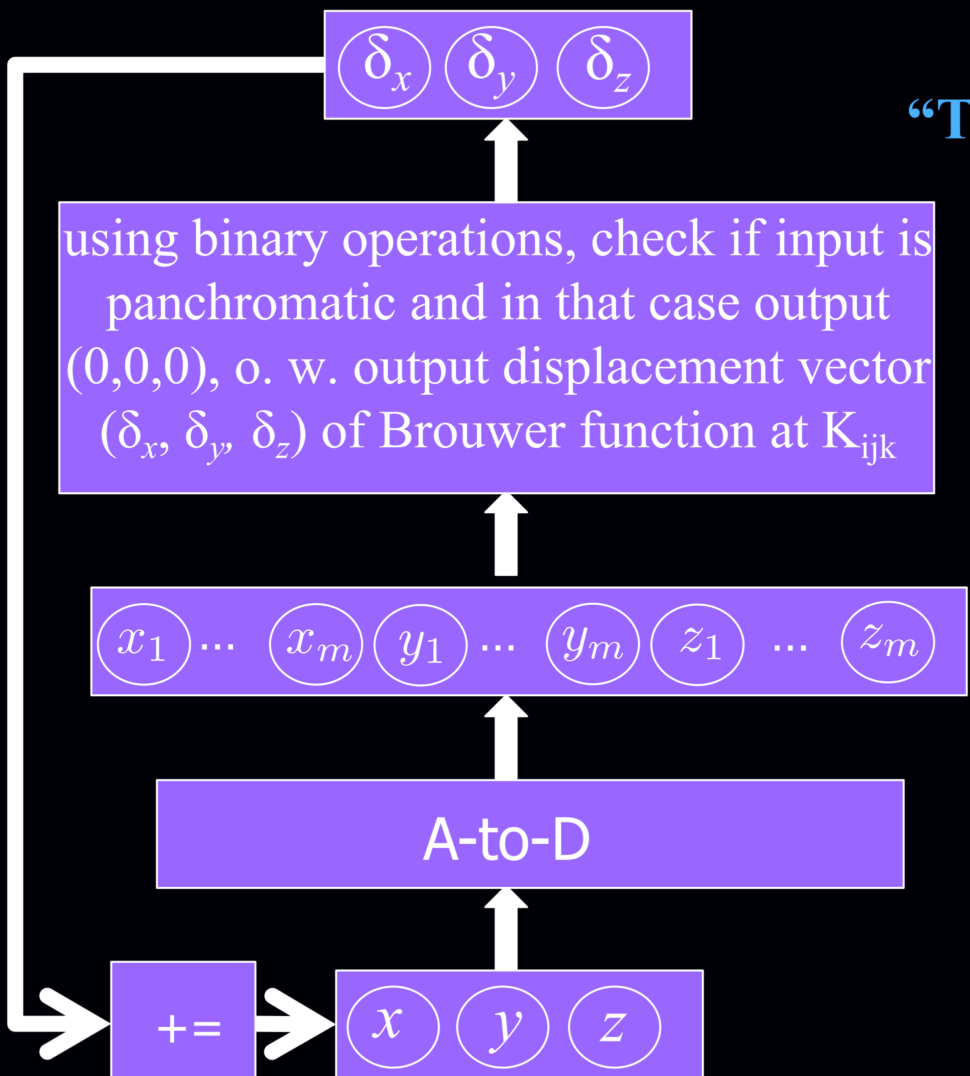
three players whose mixed strategies represent a point in $[0,1]^3$

# Add it up

since negative numbers are not allowed



$\begin{pmatrix}\text{4-displacement}\\ \text{p.w. linear}\end{pmatrix}$ BROUWER ➔ *fixed point of game-inspired straight-line program*




**"Theorem":**

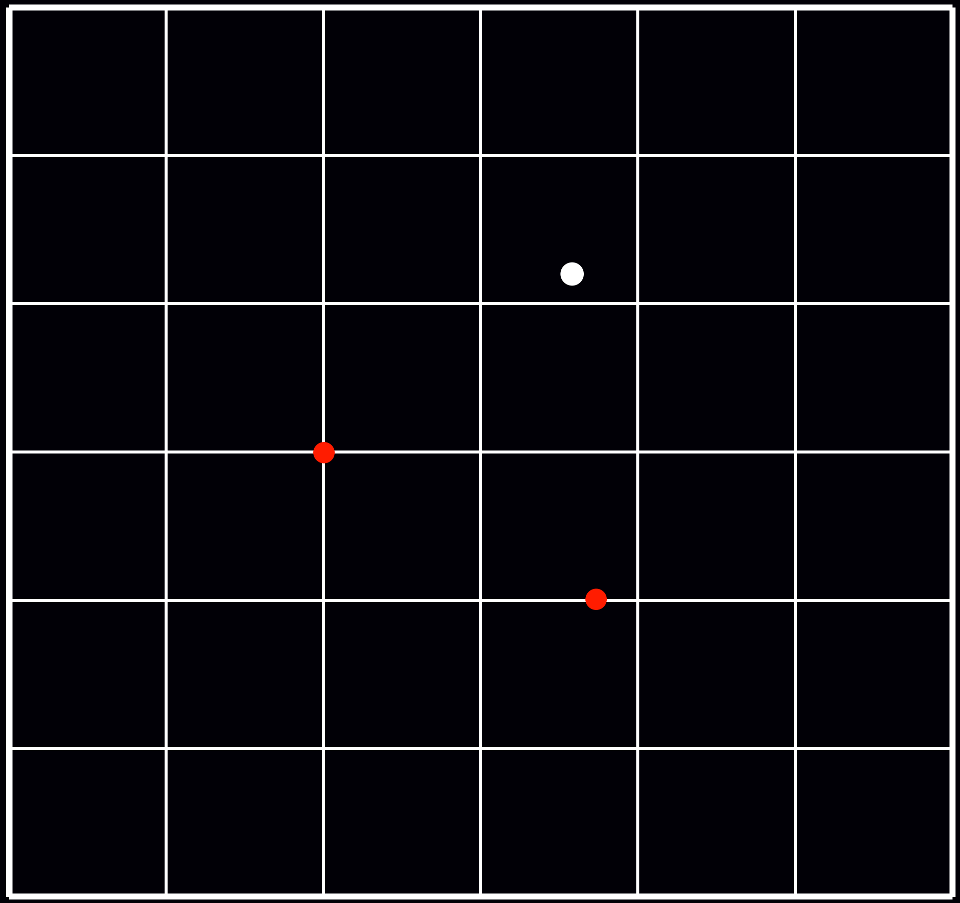
In any fixed point of the circuit shown on the left, the binary description of the point ($x$, $y$, $z$) is panchromatic.

**BUT:** Brittle comparators don't think so!

this is not necessarily binary

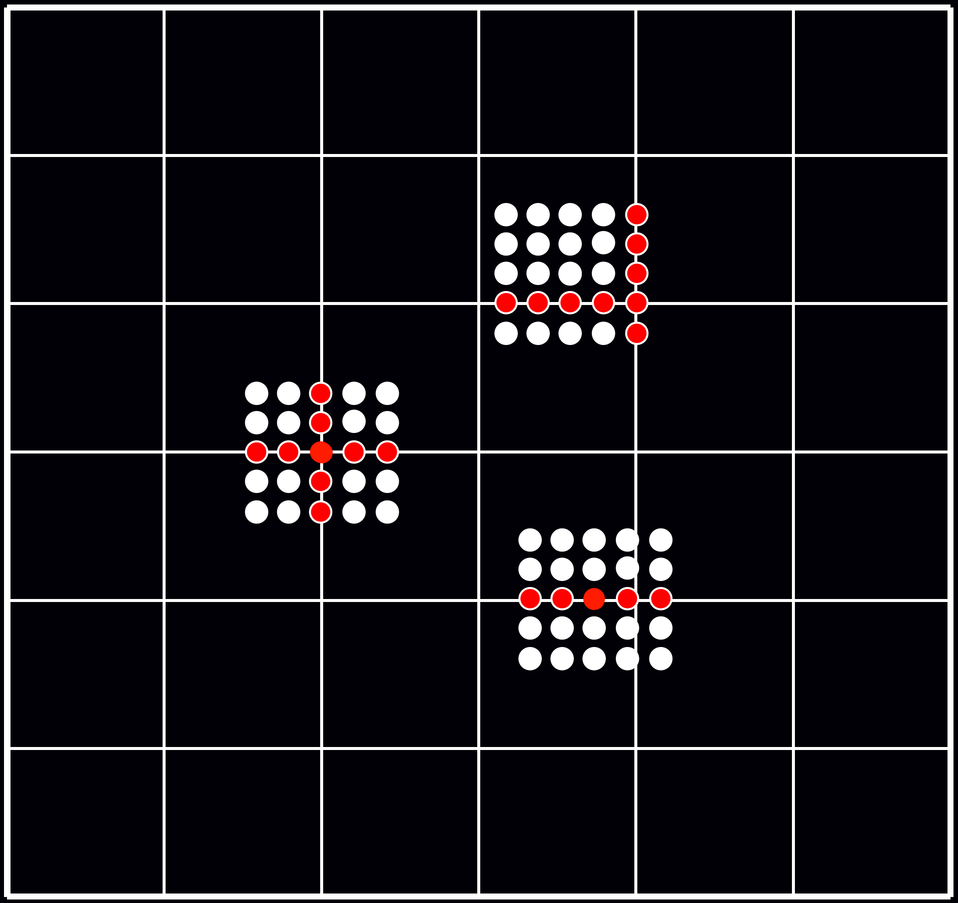
# The Final Blow

When did measure-zero sets scare us?



# The Final Blow

When did measure-zero sets scare us?



- Create a micro-lattice of copies around the original point (*x*, *y*, *z*):

$$(x + p \cdot 2^{-2m},\ y + q \cdot 2^{-2m},\ z + s \cdot 2^{-2m}),$$
$$-\ell \le p, q, s \le \ell$$

- For each copy, extract bits, and compute the displacement of the Brouwer function at the corresponding cubelet, indexed by these bits.

- Compute the average of the displacements found, and add the average to *(x*, *y*, *z*).

# Logistics

- There are $M := (2\ell + 1)^3$ copies of the point $(x, y, z)$.

- Out of these copies, at most $3(2\ell + 1)^2$ are broken, i.e. have a coordinate be an integer multiple of $2^{-m}$. We cannot control what displacement vectors will result from broken computations. — bad set $\mathcal{B}$

- On the positive side, the displacement vectors computed by at least $(2\ell - 2)(2\ell + 1)^2$ copies correspond to actual displacement vectors of Brouwer's function in the proximity of point (x,y,z). — good set $\mathcal{G}$

- At a fixed point of our circuit, it must be that the (0, 0, 0) displacement vector is added to $(x, y, z)$.

- So the average displacement vector computed by our copies must be (0,0,0).

**Theorem:** For the appropriate choice of the constant $\ell$, even if the set $\mathcal{B}$ "conspires" to output any collection of displacement vectors they want, in order for the average displacement vector to be (0, 0, 0) it must be that among the displacement vectors output by the set $\mathcal{G}$ we encounter all of (1,0,0), (0,1,0), (0,0,1), (-1,-1,-1).

# Finishing the Reduction

**Theorem:** For the appropriate choice of the constant $\ell$, even if the set $\mathcal{B}$ "conspires" to output any collection of displacement vectors they want, in order for the average displacement vector to be (0, 0, 0) it must be that among the displacement vectors output by the set $\mathcal{G}$ we encounter all of (1,0,0), (0,1,0), (0,0,1), (-1,-1,-1).

➔ In any fixed point of our circuit, $(x, y, z)$ is in the proximity of a point $(x^*, y^*, z^*)$ of the subdivision surrounded by all four displacements. This point can be recovered in polynomial time given $(x, y, z)$.

➔ in any Nash equilibrium of the polymatrix game corresponding to our circuit the mixed strategies of the players $x$, $y$, $z$ define a point located in the proximity of a point $(x^*, y^*, z^*)$ of the subdivision surrounded by all four displacements. This point can be recovered in polynomial time given $(x, y, z)$.

$\Longrightarrow$ (exact) POLYMATRIX NASH is PPAD-complete

# Finishing the Reduction

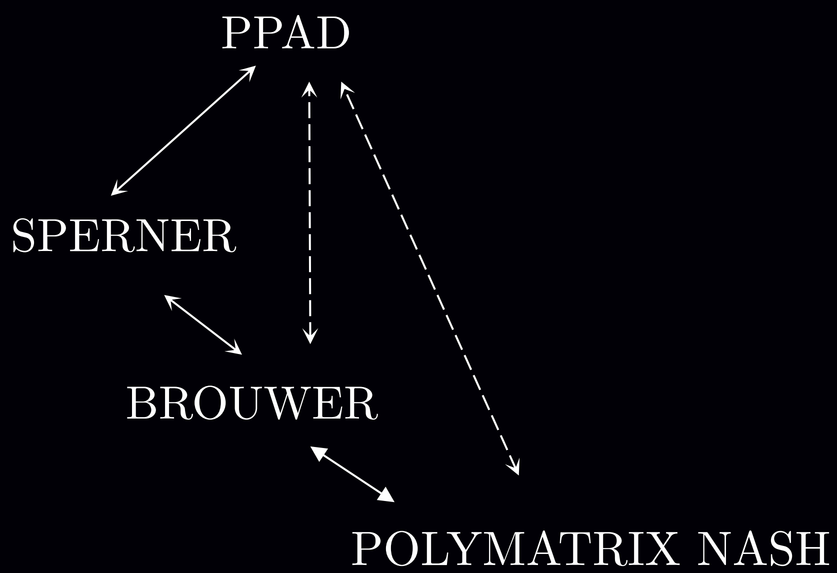
**Theorem:** Given a polymatrix game $\mathcal{G}$ there exists $\epsilon^*$ such that:

1. $|\epsilon^*| = \mathrm{poly}(|\mathcal{G}|$
2. given a $\epsilon^*$-Nash equilibrium of $\mathcal{G}$ we can find in polynomial time an exact Nash equilibrium of $\mathcal{G}$ .

**Proof: exercise**

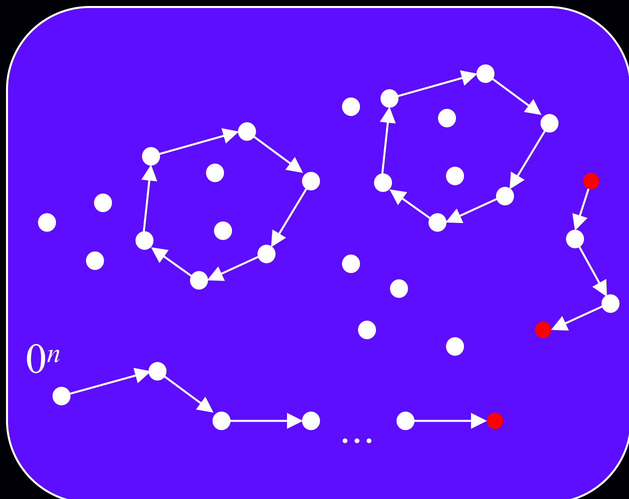
$\Longrightarrow$ (exact) POLYMATRIX NASH $\equiv$ POLYMATRIX NASH

$\Longrightarrow$ POLYMATRIX NASH is PPAD-complete


PPAD
SPERNER
BROUWER
POLYMATRIX NASH

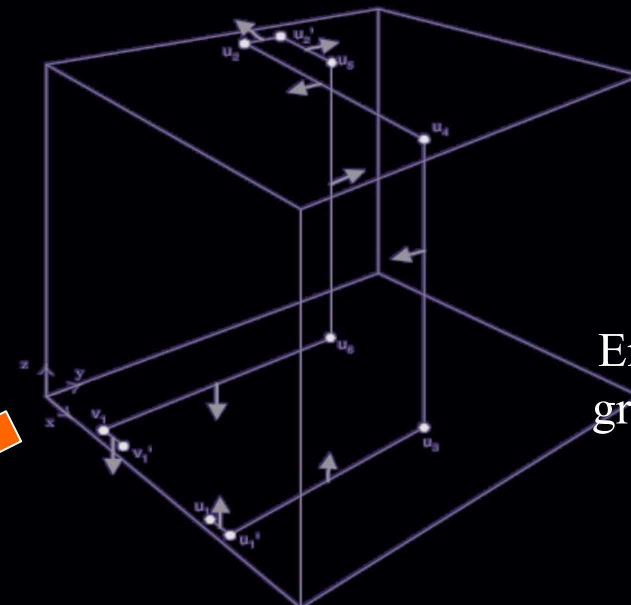
# *Next Lecture*

DGP = Daskalakis, Goldberg, Papadimitriou
DP = Daskalakis, Papadimitriou
CD = Chen, Deng



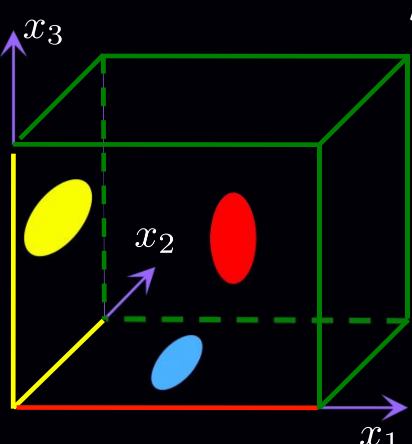

Generic PPAD

[Pap '94]
[DGP '05]



Embed PPAD graph in $[0,1]^3$

[DGP '05]

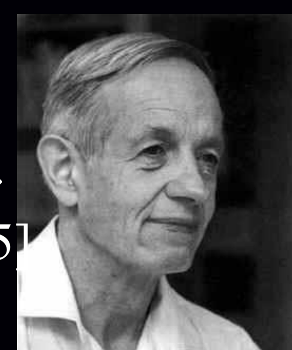



3D-SPERNER

[DGP '05]

p.w. linear BROUWER

[DGP '05]



multi-player NASH

[DGP '05]
4-player NASH

[DP '05]
[CD'05]
3-player NASH

[CD'06]
2-player NASH