6.S890: Topics in Multiagent Learning

Lecture 8 – Prof. Daskalakis

Fall 2023

Refresher

- Last few times we stated and proved the theorems of Nash, Sperner, and Brouwer.
- [Nash'1950]: Every finite game has a Nash equilibrium.
- [Brouwer'1911]: Every continuous function $f: D \to D$ from a convex compact set D to itself has a fixed point $x^* = f(x^*)$.
- [Sperner'1928]: Every legal 3-coloring of a 2-d triangulated square has a tri-chromatic triangle.
- We also saw that:

  Sperner Lemma ⇒ Brouwer Theorem ⇒ Nash Theorem

- which implies as a corollary that:

  Computing Nash Equilibria → Computing Brouwer Fixed Points → Finding Sperner Triangles

- But what is the complexity of these problems?
  - we remarked that these problems are in the complexity class TFNP of *total search problems in NP*
  - "total" : they always have a solution, unlike e.g. SAT
- So what is their complexity?

A Complexity Theory of Total Search Problems ?

100-feet overview of our methodology:

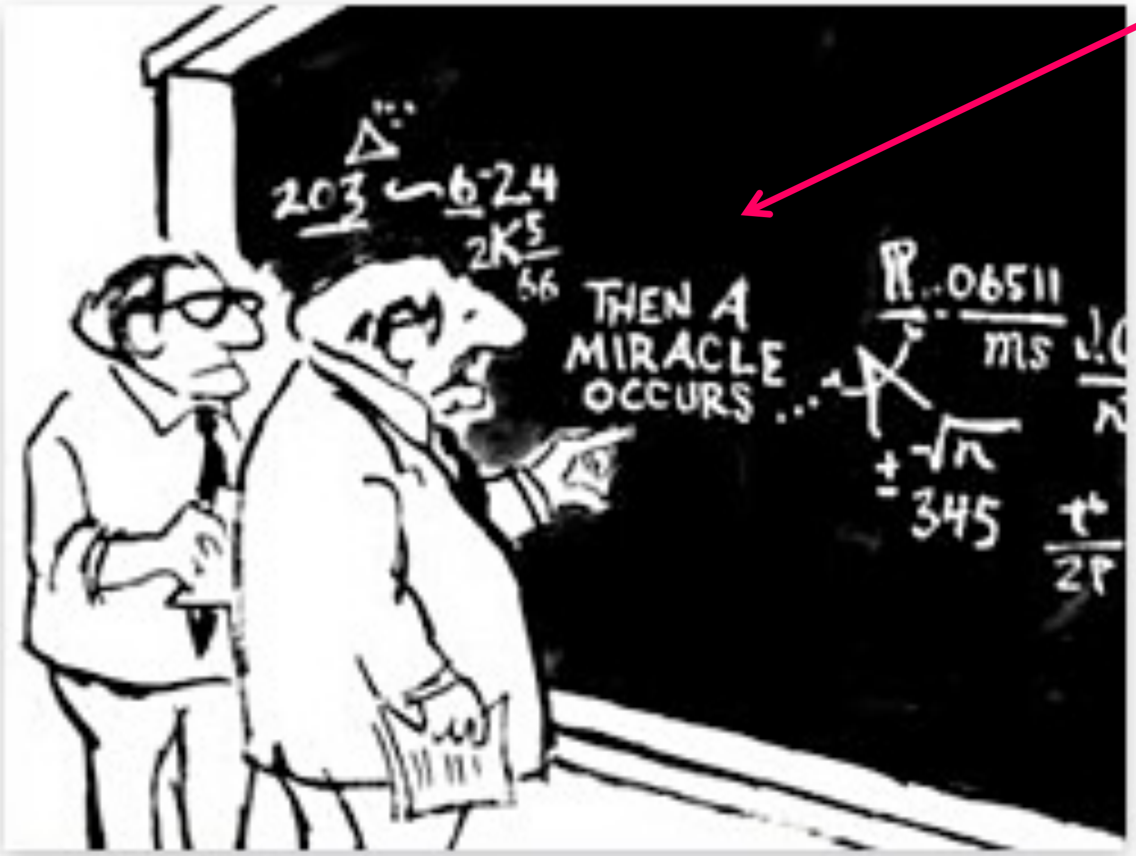
1. identify the combinatorial argument of existence, responsible for making these problems total;

2. define a complexity class inspired by the argument of existence;

3. make sure that the complexity of the problem was captured as tightly as possible (via completeness results).

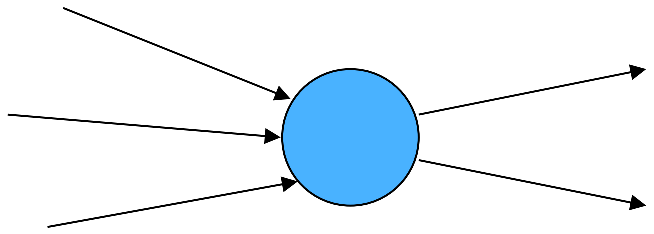
OK, so what is the combinatorial argument of existence underlying Sperner, Brouwer and Nash?

??



A parity lemma in directed graphs:

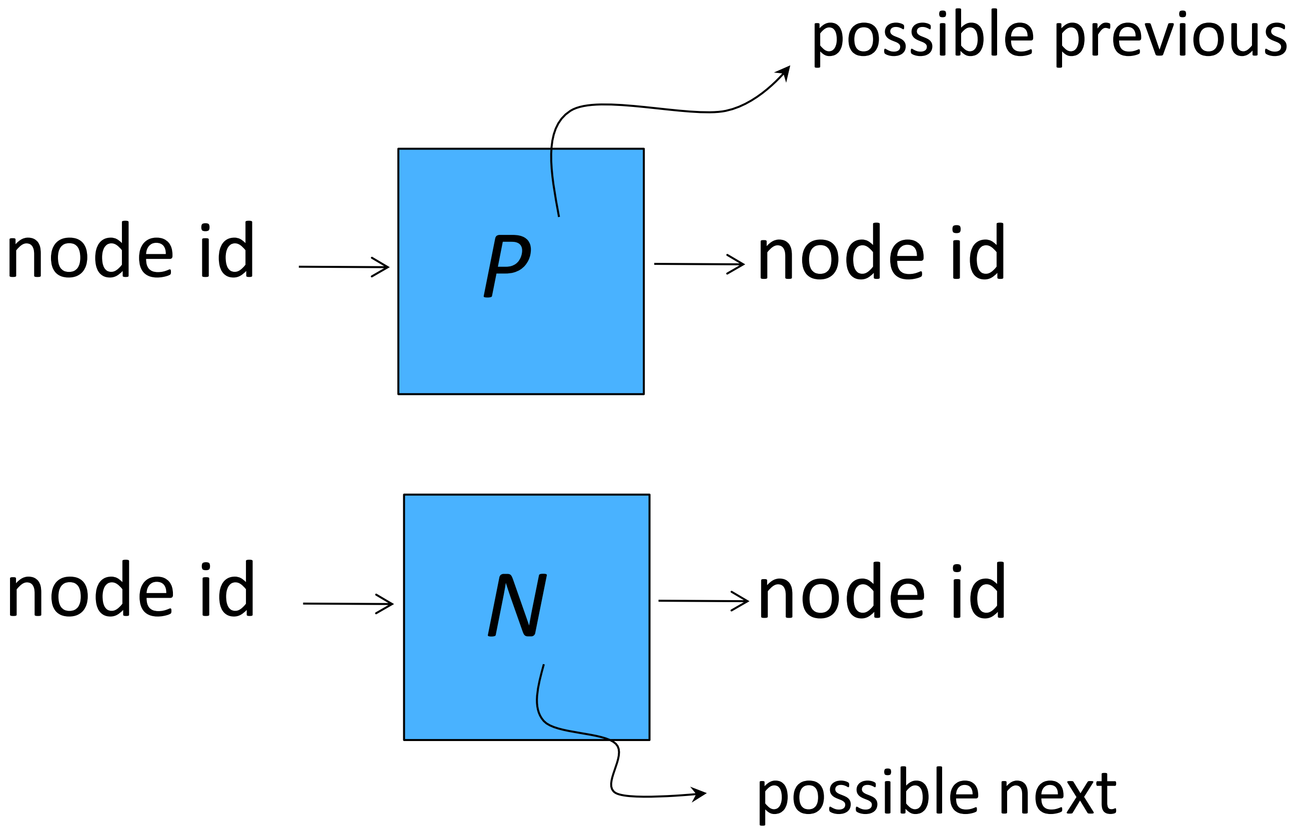
A directed graph with an unbalanced node (a node with indegree ≠ outdegree) must have another.



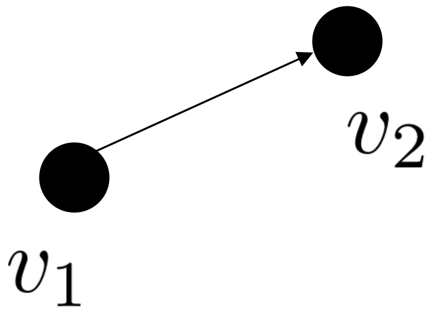
The PPAD Class [Papadimitriou '94]

a complexity class capturing TFNP problems whose totality is due to the directed parity argument

Suppose that an exponentially large graph with vertex set $\{0,1\}^n$ is defined by two circuits:



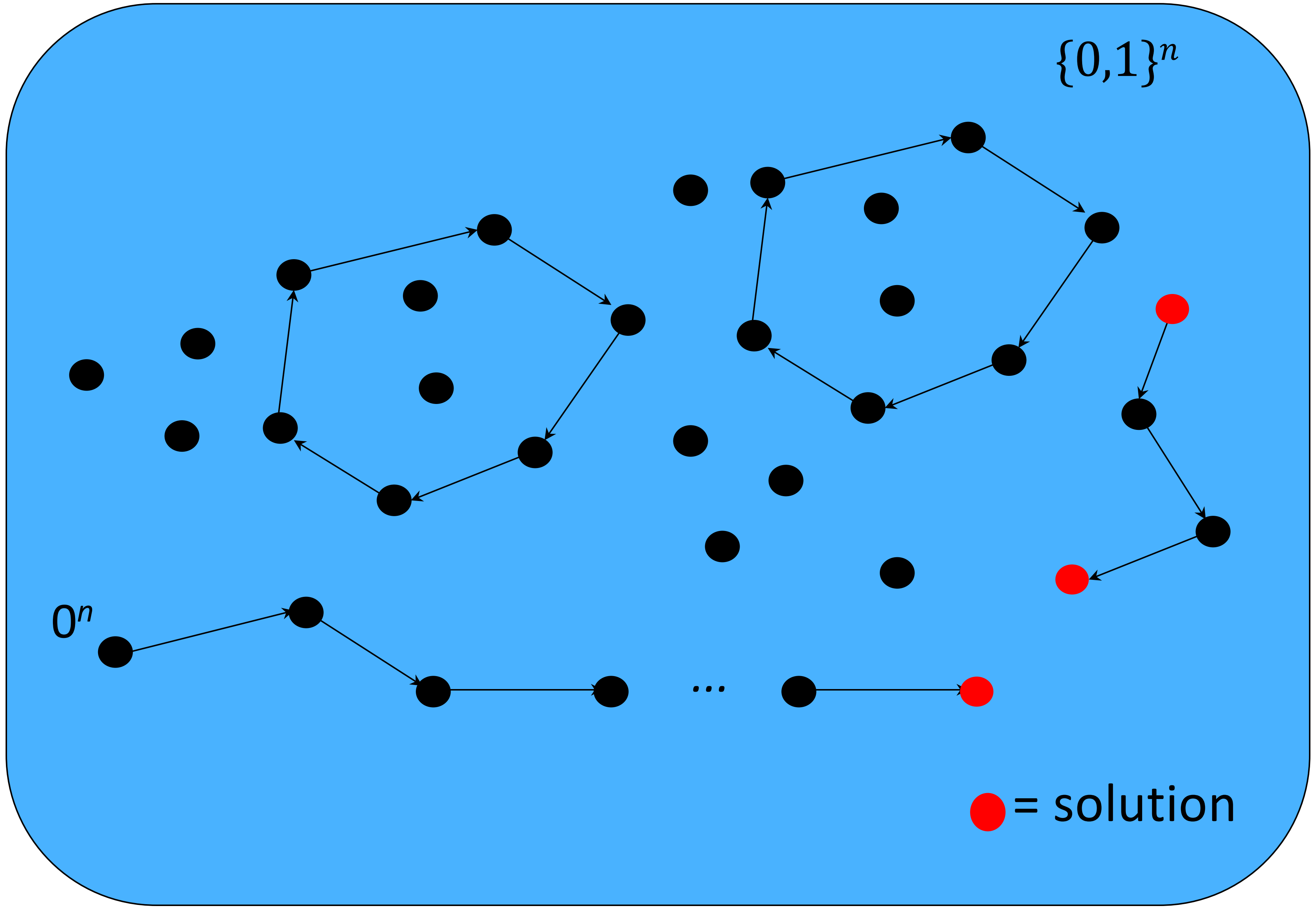
$$P(v_2) = v_1 \wedge N(v_1) = v_2$$

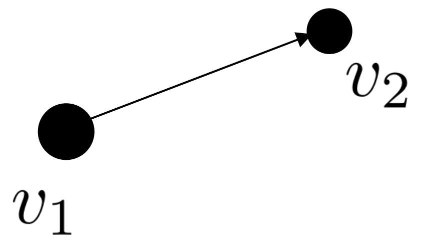


END OF THE LINE: Given P and N: If 0^n is an unbalanced node, find another unbalanced node. Otherwise output 0^n.

PPAD = { *Search problems in* FNP *reducible to* END OF THE LINE }

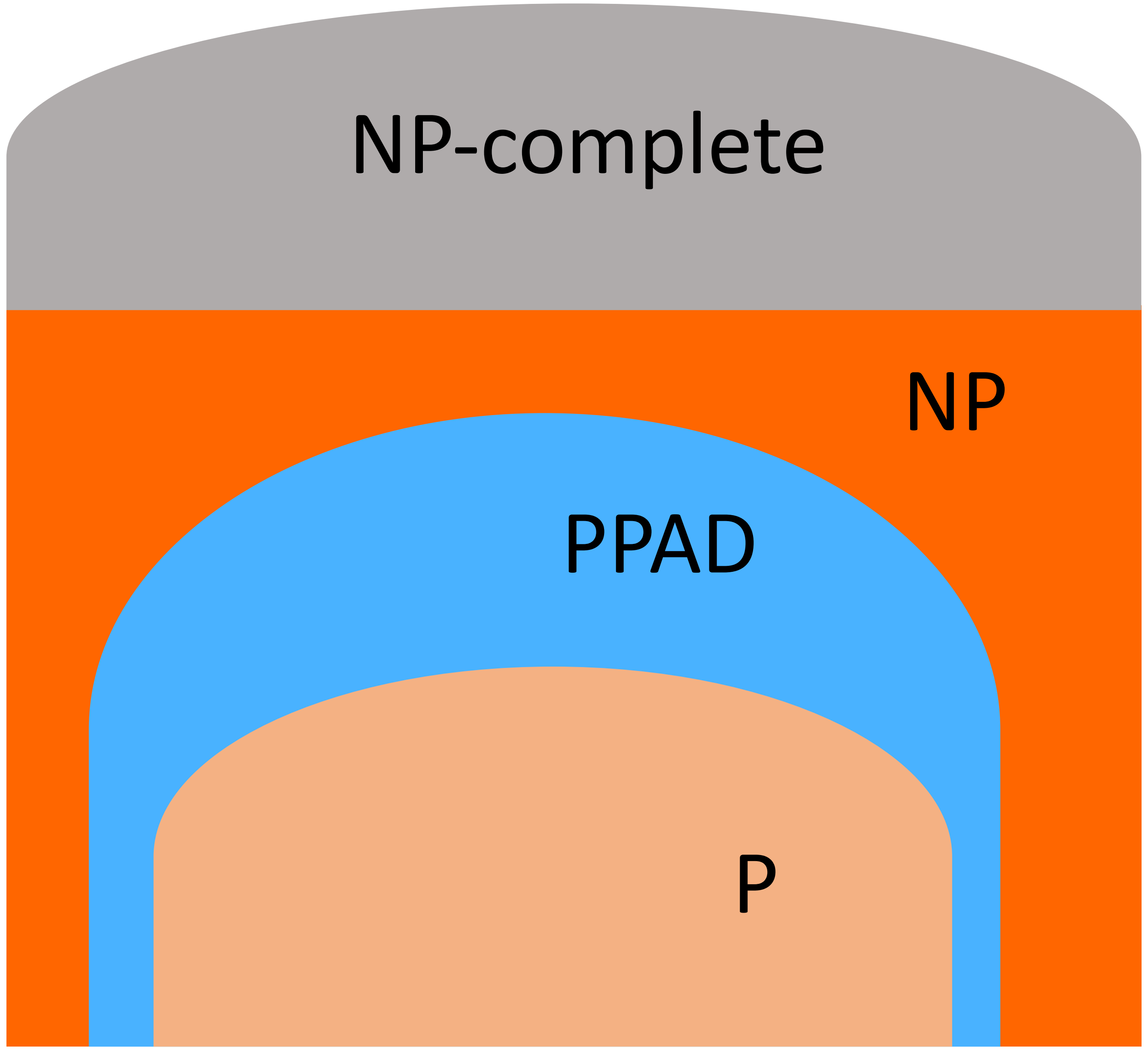
END OF THE LINE

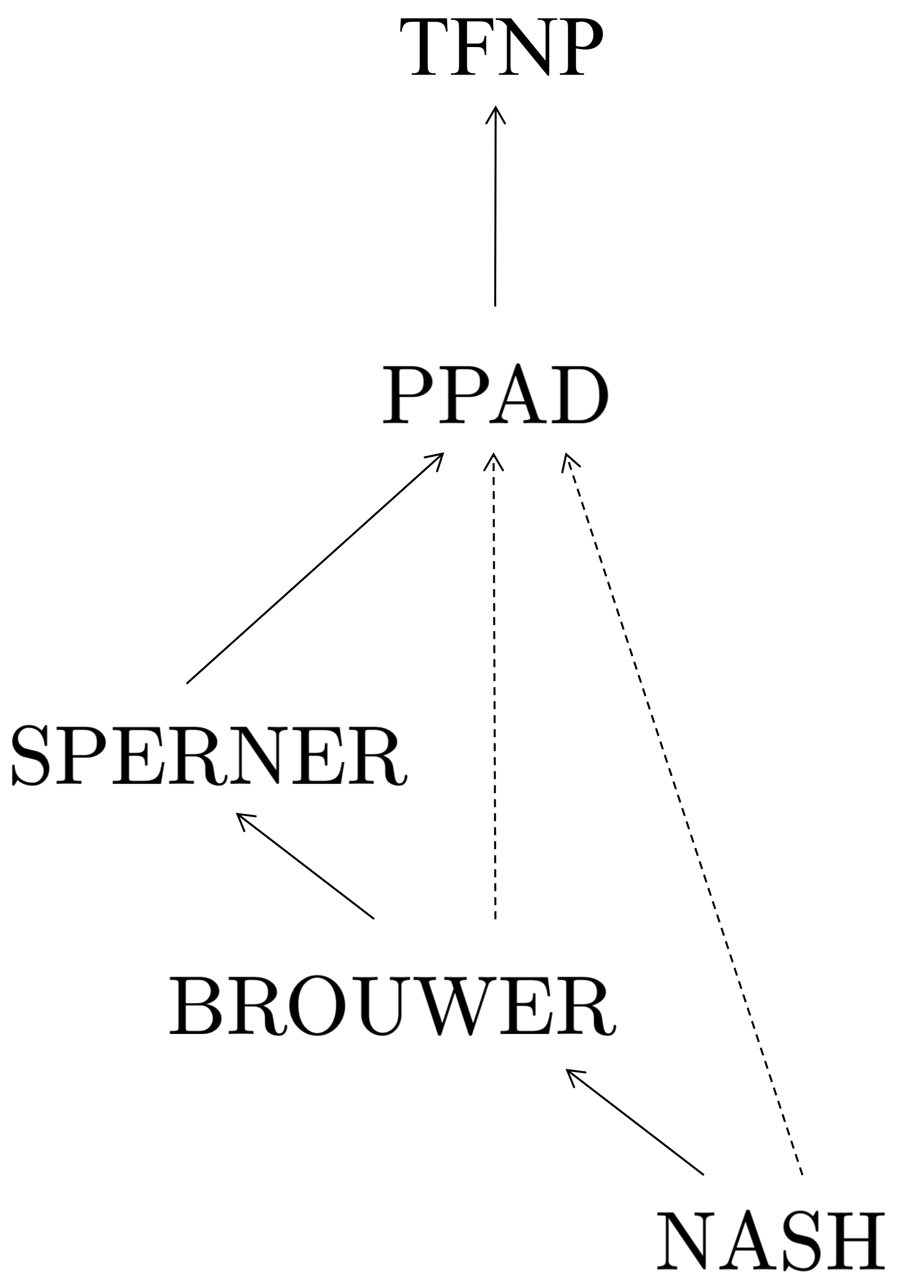




$$\Leftrightarrow \quad P(v_2) = v_1 \wedge N(v_1) = v_2$$

Believed Location of PPAD



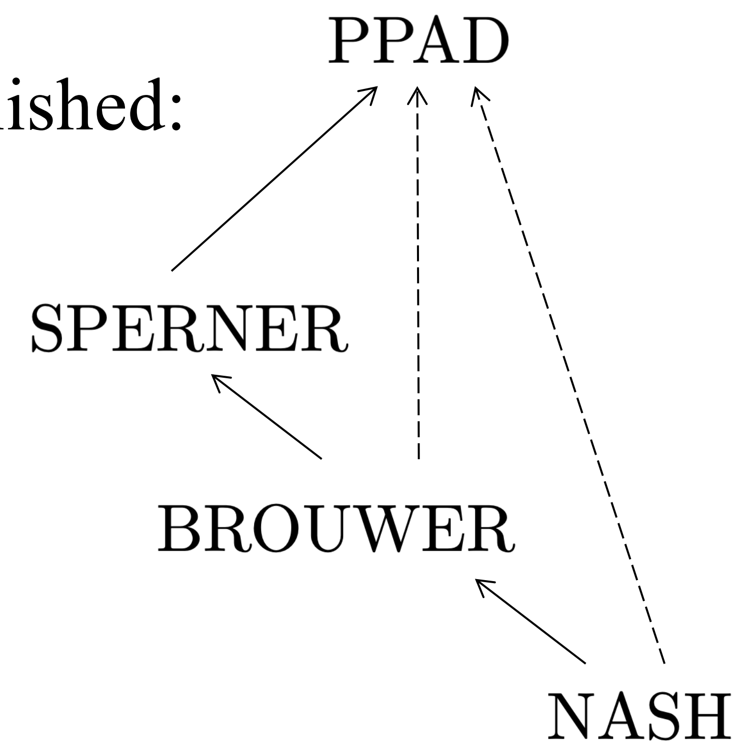


→ means poly-time reduction

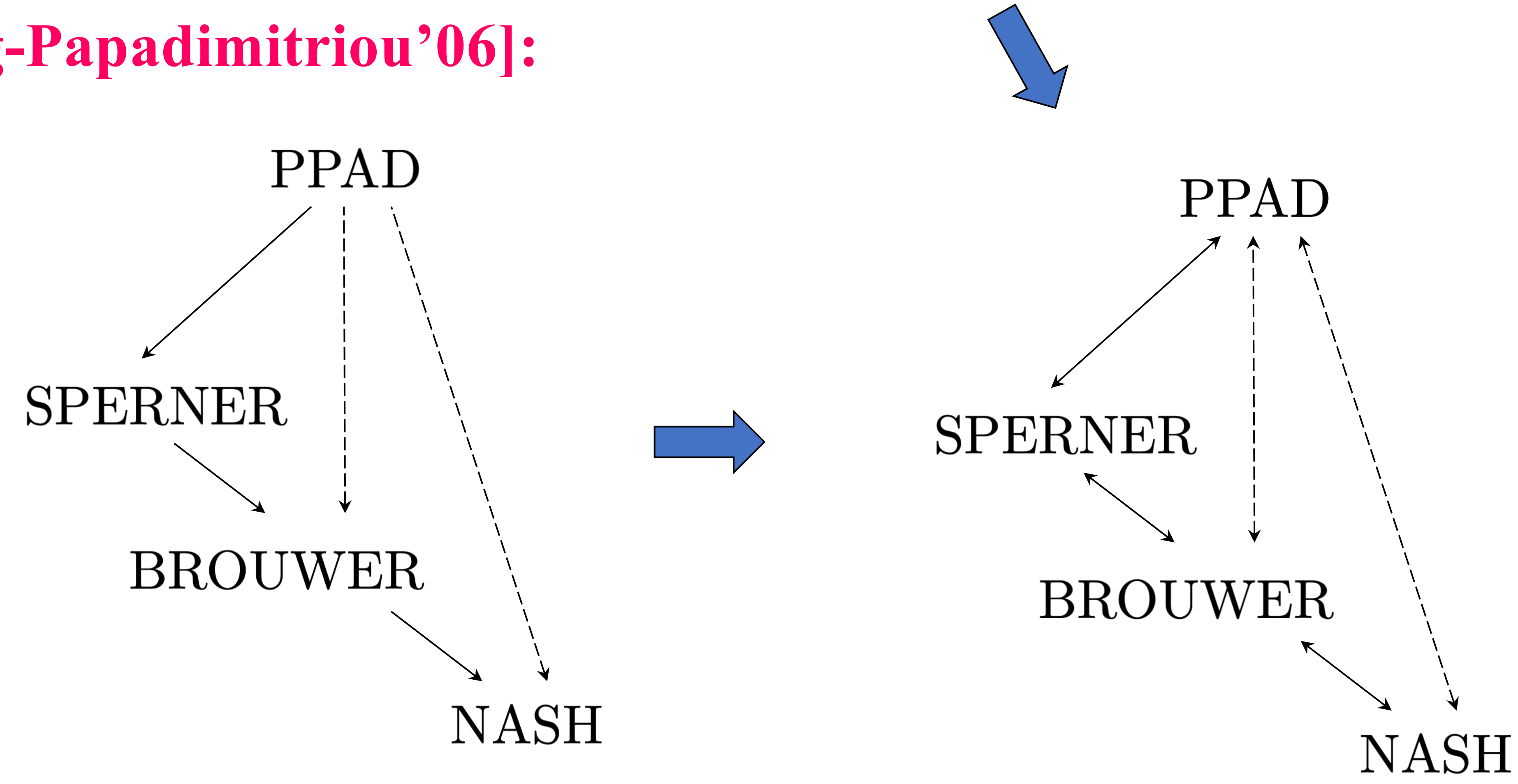
Partial Success: NASH, BROUWER, and SPERNER are in PPAD

Litmus Test: Are NASH, BROUWER, and SPERNER PPAD-complete?

Poly-time Reductions that we just established:



[Daskalakis-Goldberg-Papadimitriou’06]:



Menu

Refresher: Nash, Sperner, Brouwer, PPAD

Total Search Problems in NP

PPAD

PPAD-hardness of NASH

Menu

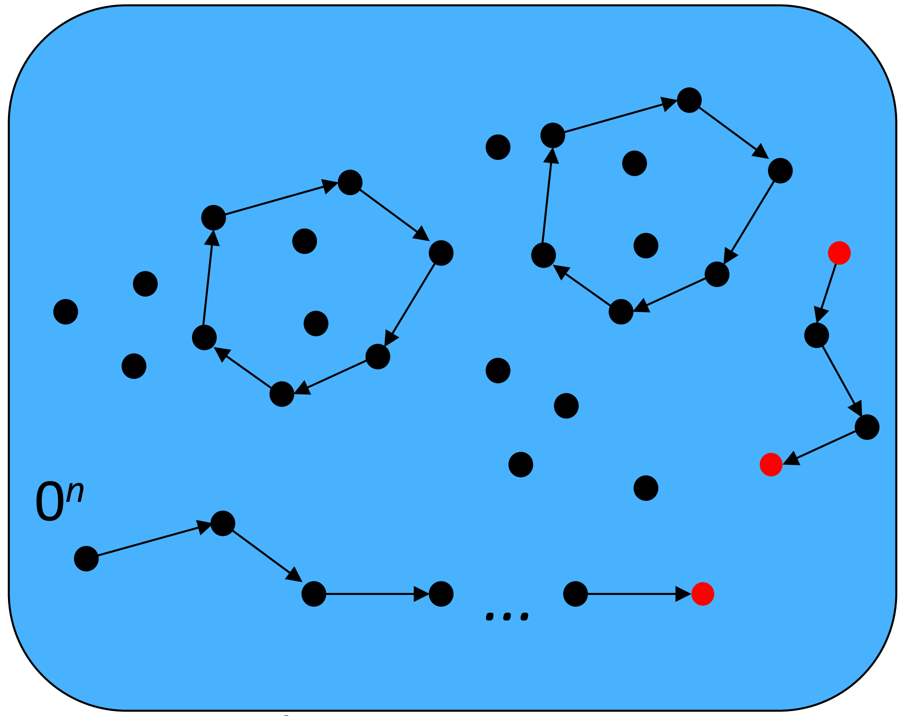
Refresher: Nash, Sperner, Brouwer, PPAD

Total Search Problems in NP

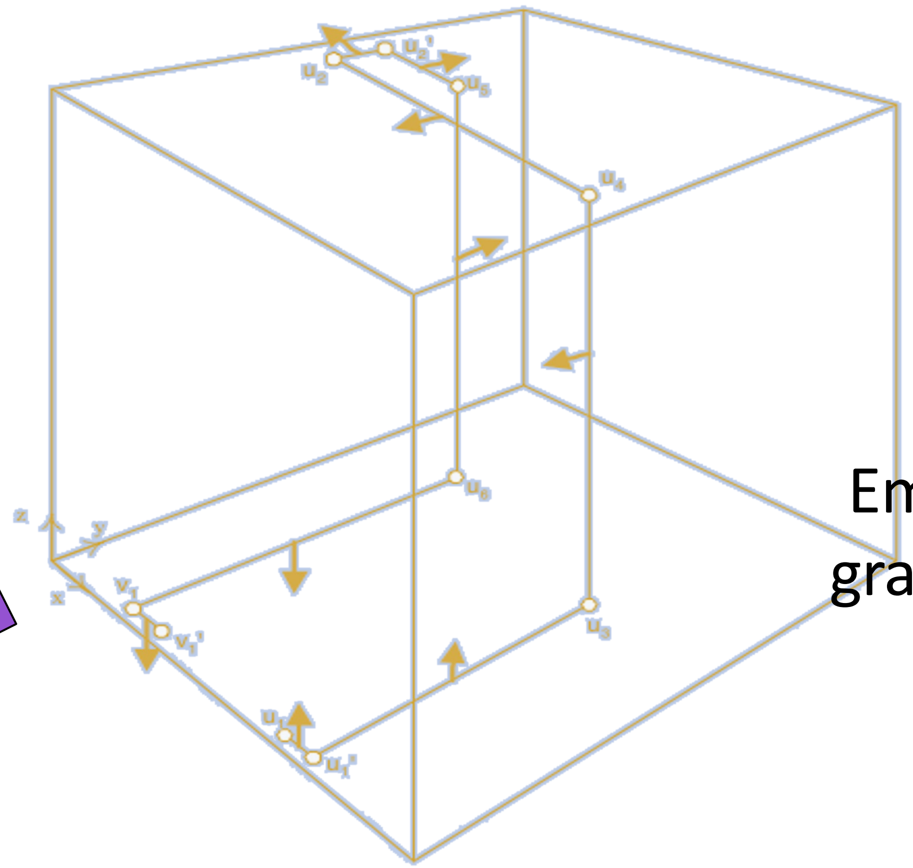
PPAD

PPAD-hardness of NASH

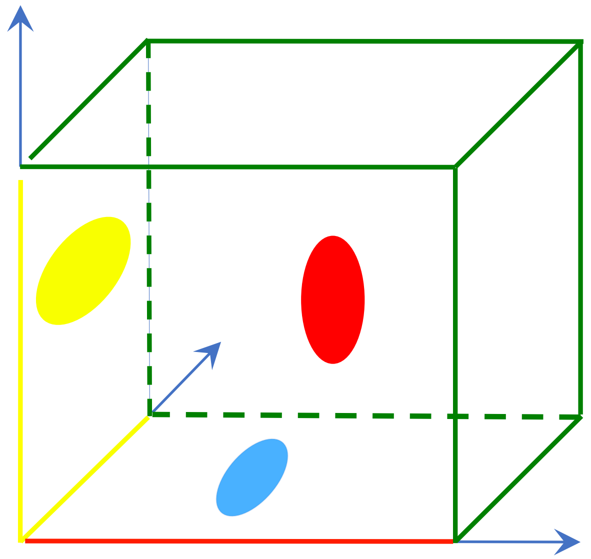
PPAD-Completeness of NASH [Daskalakis, Goldberg, Papadimitriou'06]



Generic PPAD

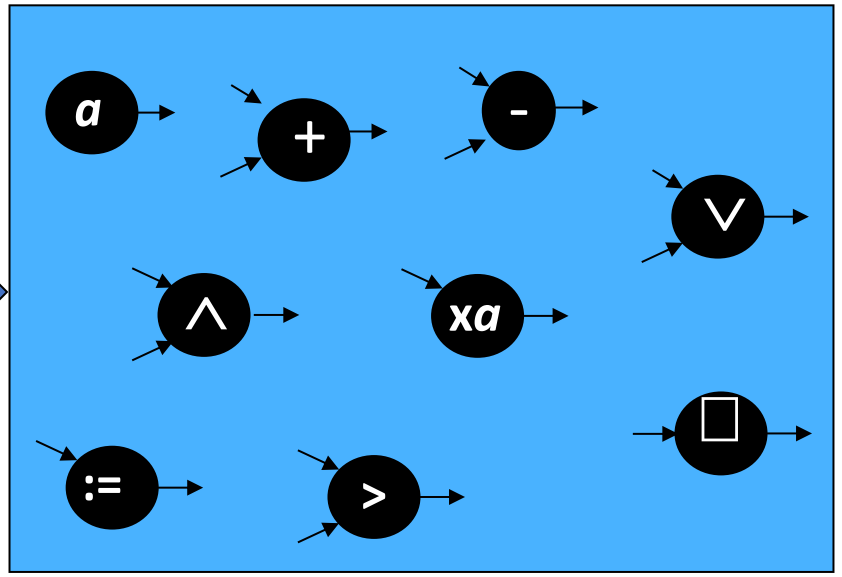


Embed PPAD graph in $[0,1]^3$

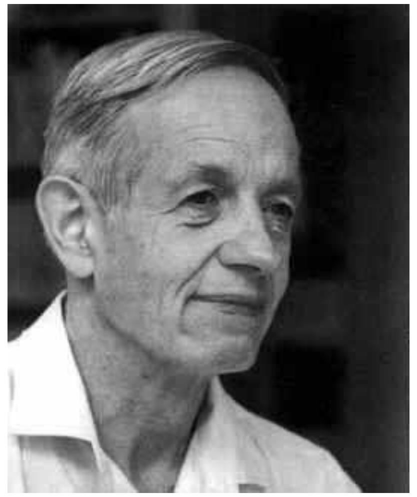


3D-SPERNER

3D-BROUWER

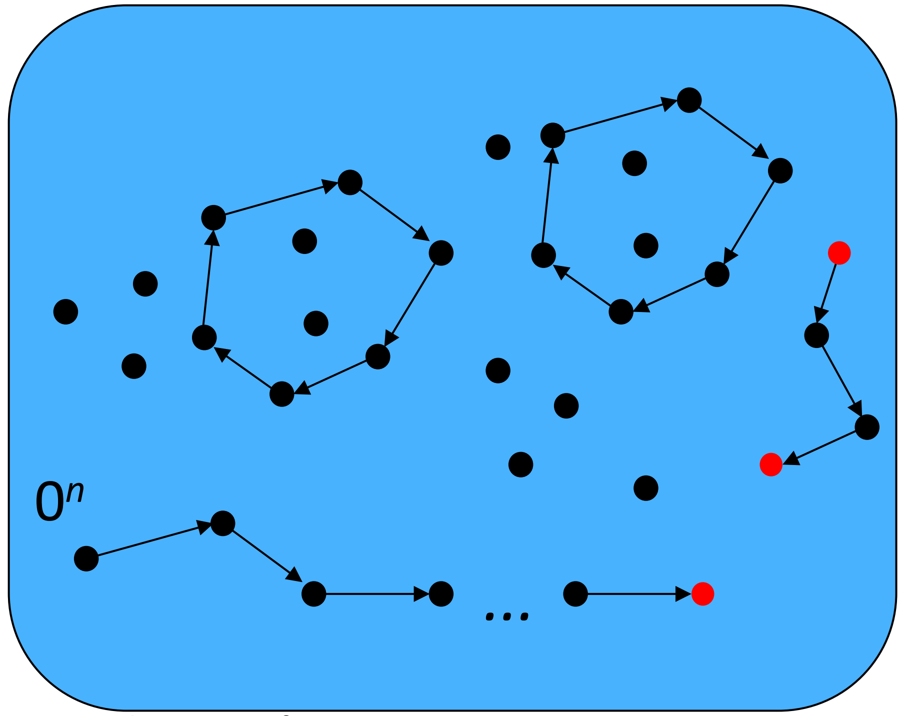


ARITHMCIRCUITSAT

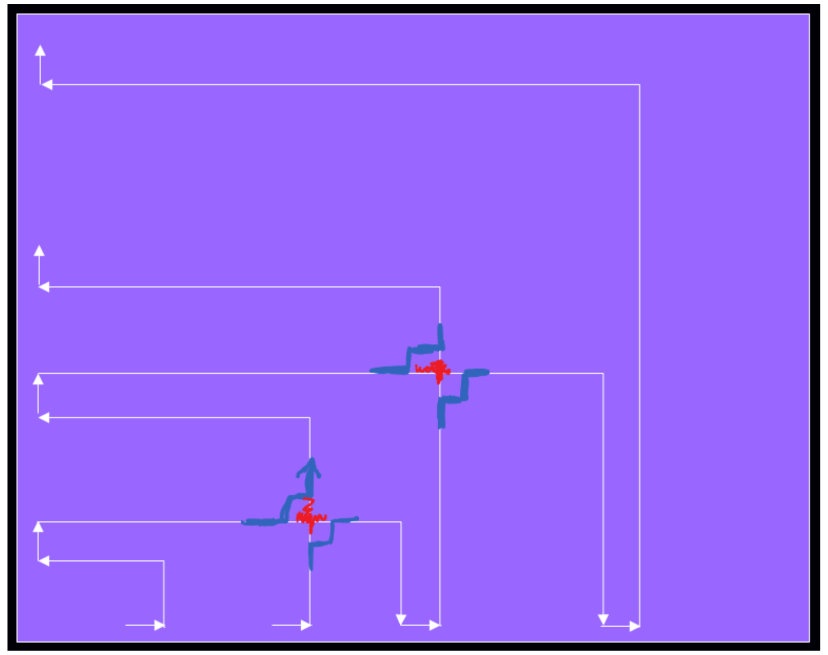


NASH

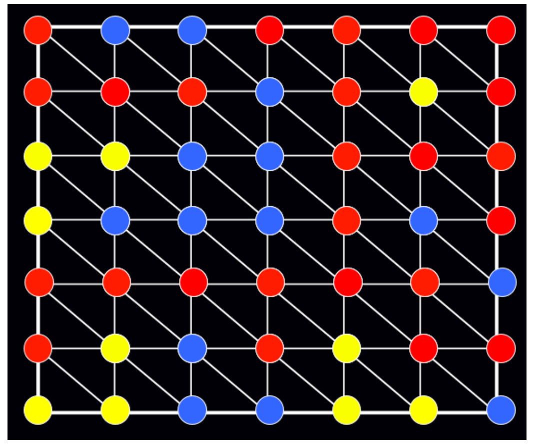
PPAD-Completeness of NASH [Daskalakis, Goldberg, Papadimitriou'06]



Generic PPAD

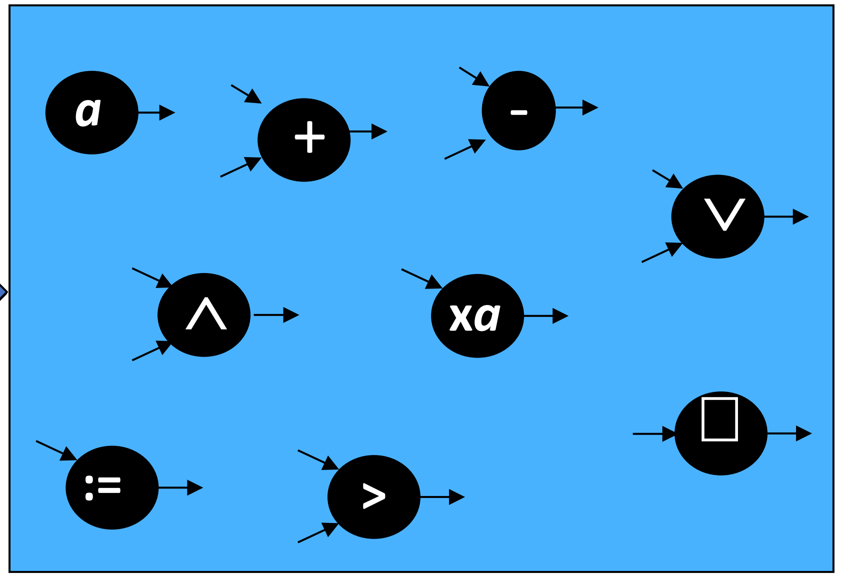


Embed PPAD graph in $[0,1]^2$

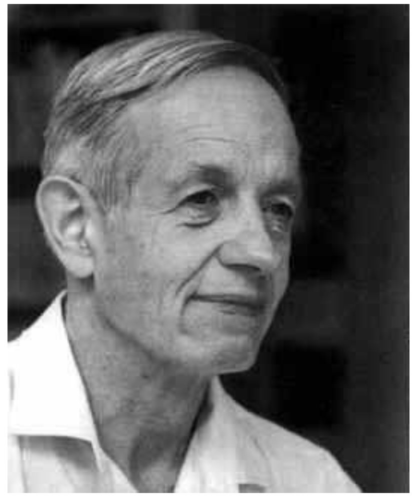


2D-SPERNER

2D-BROUWER

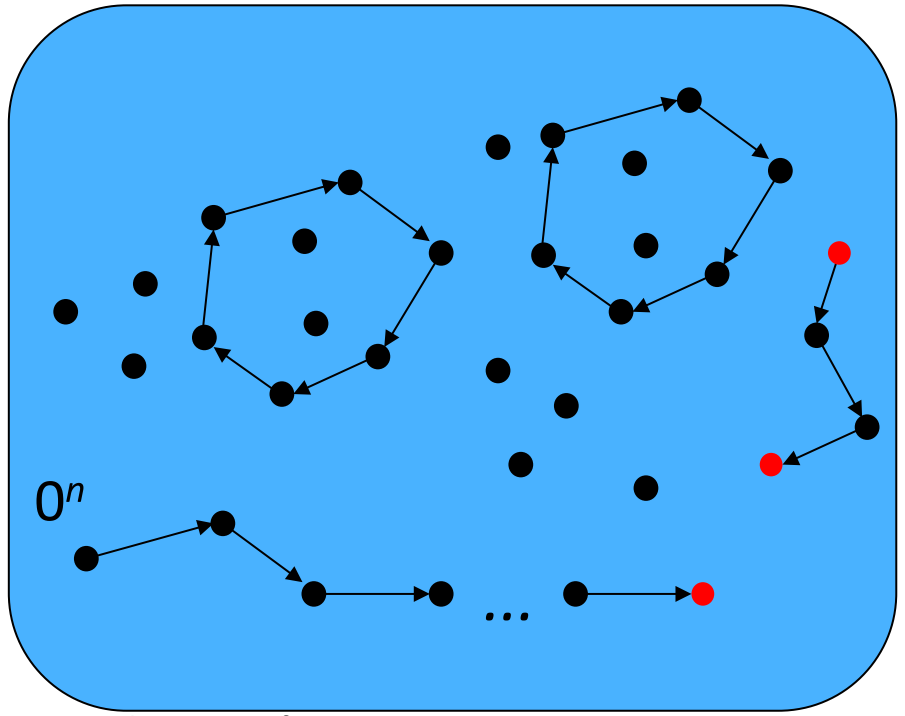


ARITHMCIRCUITSAT

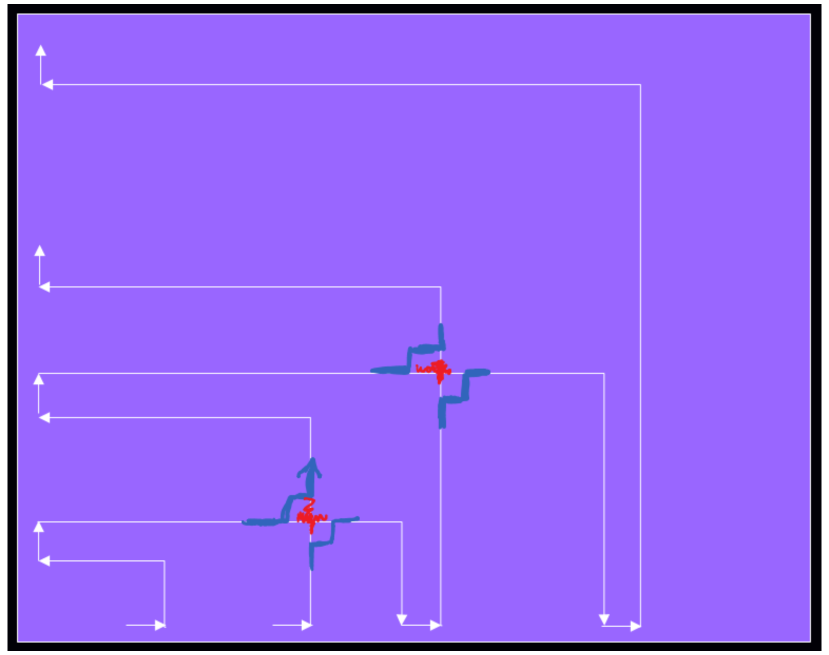


NASH

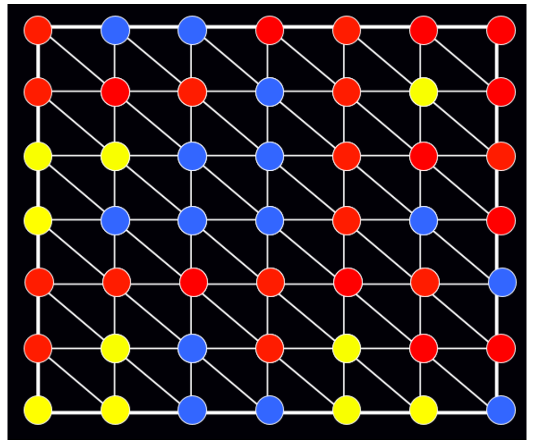
PPAD-Completeness of NASH [Daskalakis, Goldberg, Papadimitriou'06]



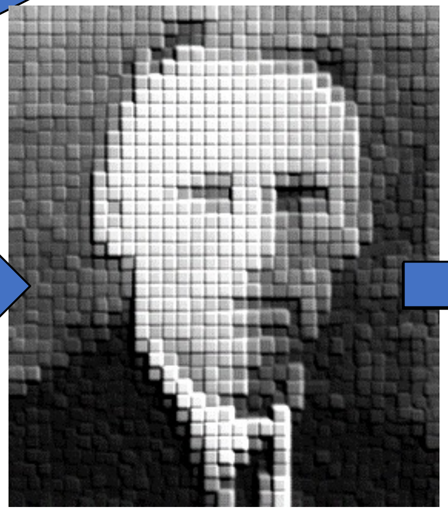
Generic PPAD



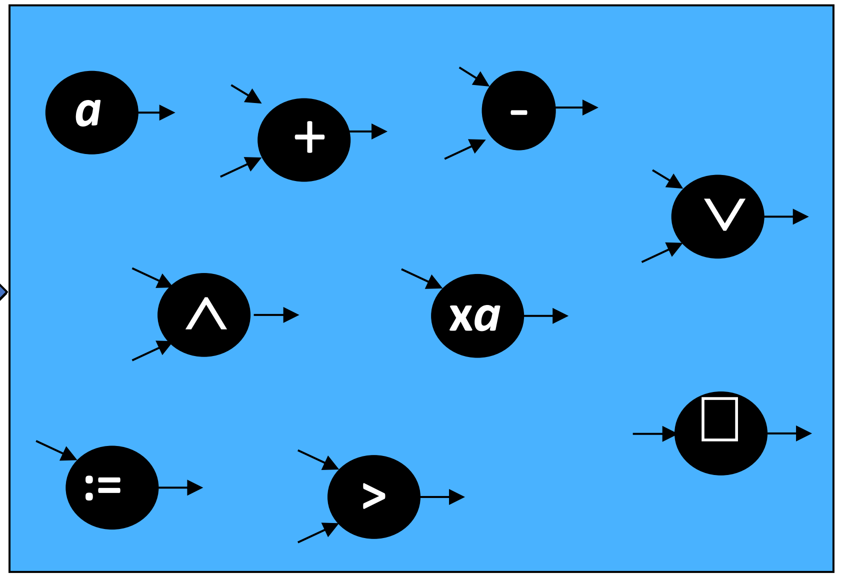
Embed PPAD graph in $[0,1]^2$



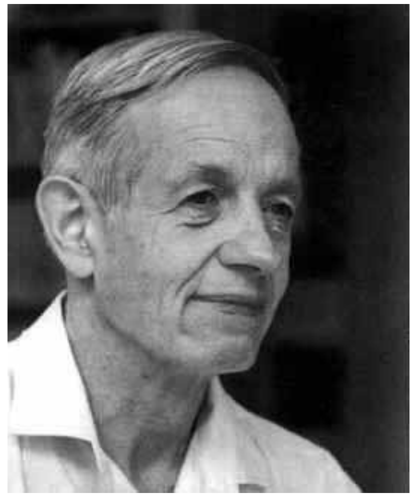
2D-SPERNER



2D-BROUWER



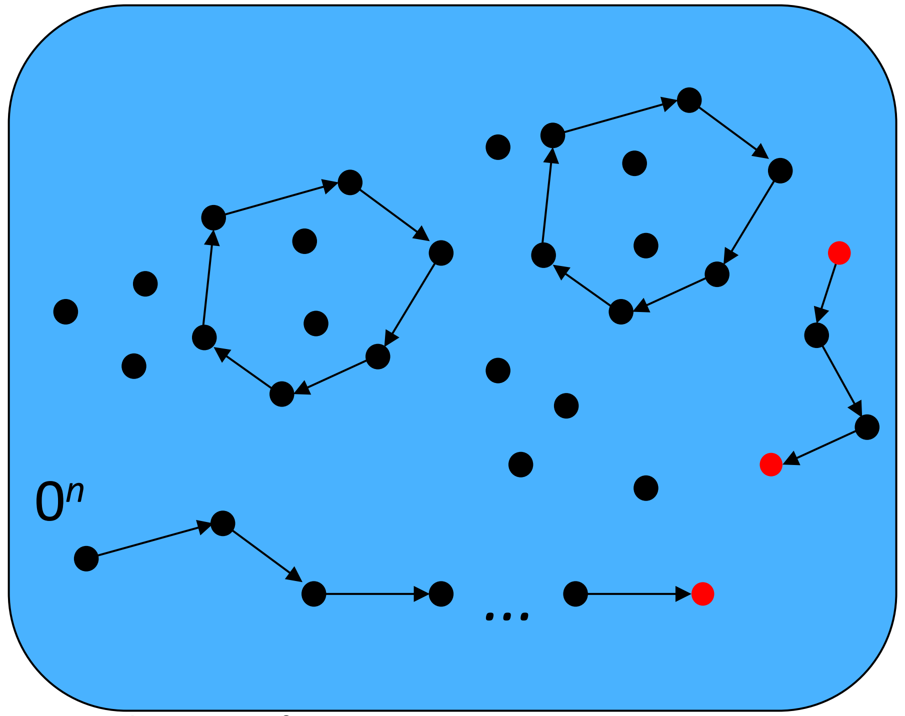
ARITHMCIRCUITSAT



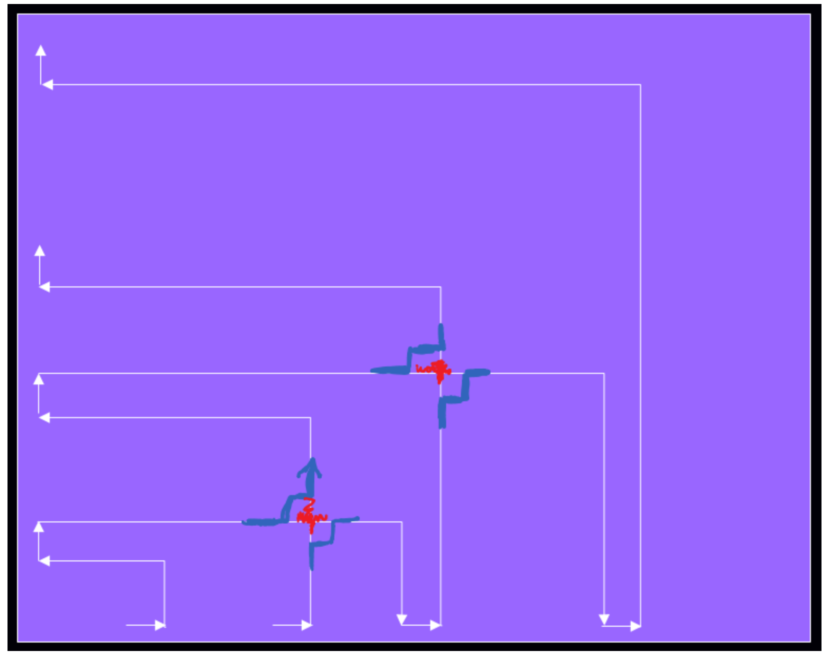
NASH

PPAD-Completeness of NASH

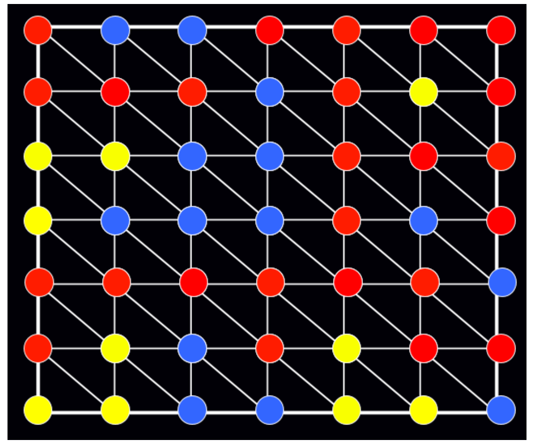
[Daskalakis, Goldberg, Papadimitriou'06]



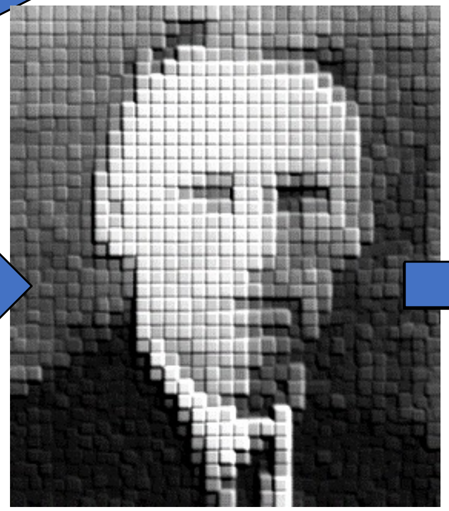
Generic PPAD



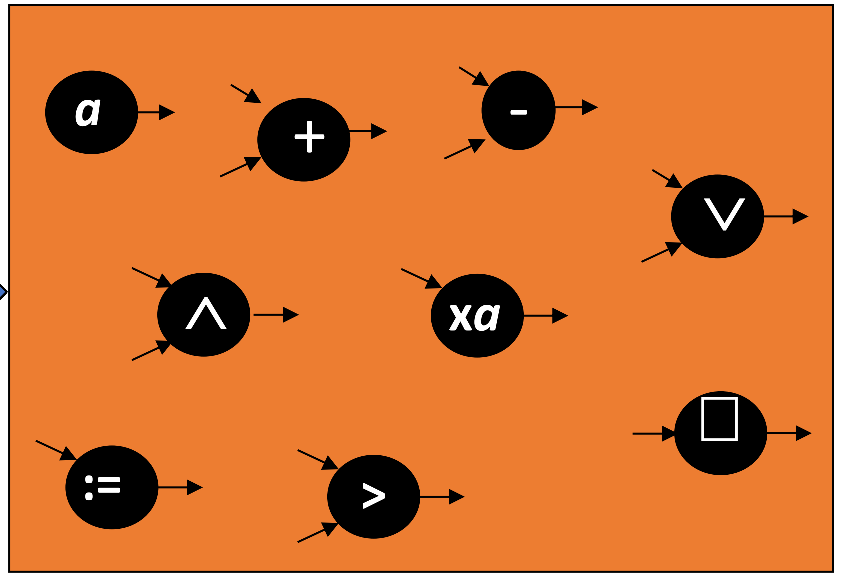
Embed PPAD graph in $[0,1]^2$



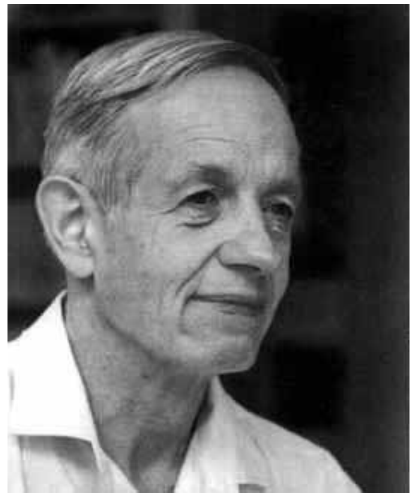
2D-SPERNER



2D-BROUWER



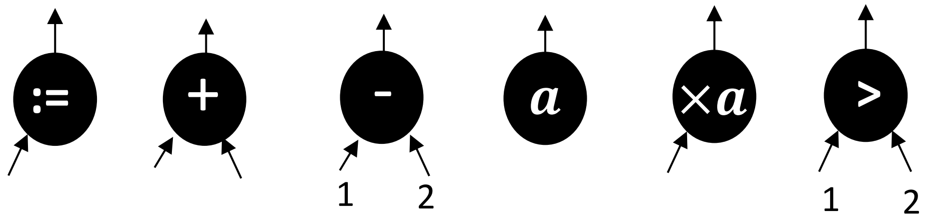
ARITHMCIRCUITSAT



NASH

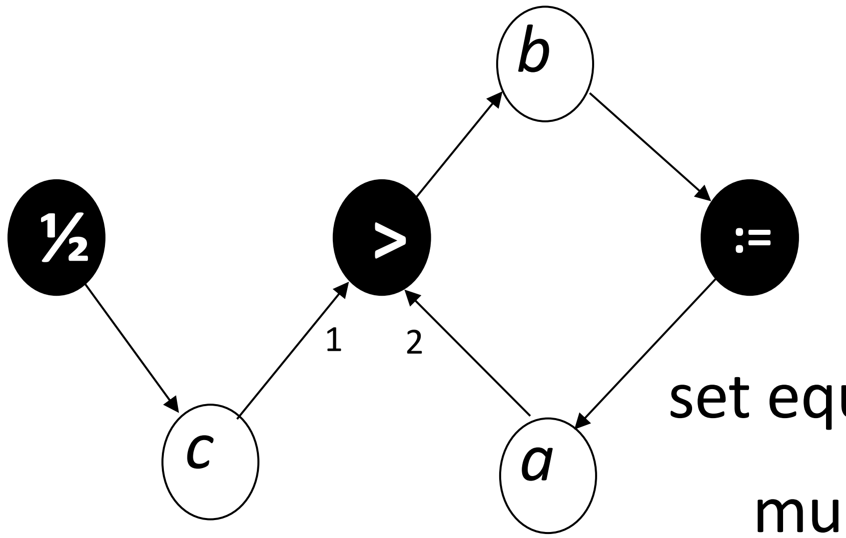
ARITHMCIRCUITSAT [Daskalakis, Goldberg, Papadimitriou'06]

INPUT: A circuit comprising:

- variable nodes $v_1,\ldots, v_n$
- gate nodes $g_1,\ldots, g_m$ of types: 

- directed edges connecting variables to gates and gates to variables (loops are allowed);
- variable nodes have in-degree 1; gates have 0, 1, or 2 inputs depending on type as above; gates & nodes have arbitrary fan-out

OUTPUT: Values $v_1,\ldots, v_n \in [0,1]$ satisfying the gate constraints:



assignment : $y == x_1$

addition : $y == \min\{1, x_1 + x_2\}$

subtraction : $y == \max\{0, x_1 - x_2\}$

set equal to a constant : $y == \max\{0, \min\{1, a\}\}$

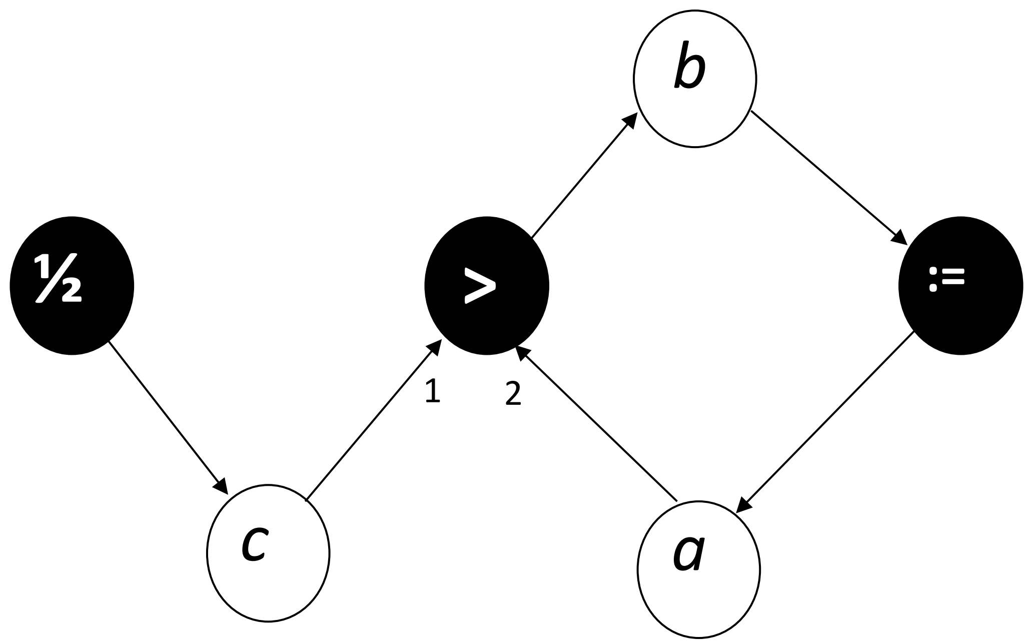
multiply by constant : $y == \max\{0, \min\{1, a \cdot x_1\}\}$

Comparator Gate Constraints

$$y == \begin{cases} 1, & \text{if } x_1 > x_2 \\ 0, & \text{if } x_1 < x_2 \\ *, & \text{if } x_1 = x_2 \end{cases}$$

any value is allowed

ARITHMCIRCUITSAT (example)



$$y == \begin{cases} 1, & \text{if } x_1 > x_2 \\ 0, & \text{if } x_1 < x_2 \\ *, & \text{if } x_1 = x_2 \end{cases}$$

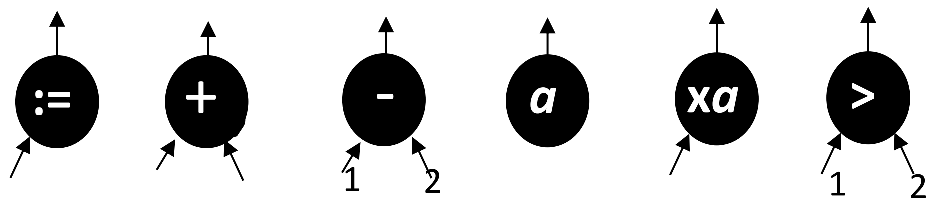
Satisfying assignment?

a = *b* = *c* = ½

ArithmCircuitSAT [Daskalakis, Goldberg, Papadimitriou'06]

INPUT: A circuit comprising:

- variable nodes $v_1,\ldots, v_n$
- gate nodes $g_1,\ldots, g_m$ of types:



- directed edges connecting variables to gates and gates to variables (loops are allowed);
- variable nodes have in-degree 1; gates have 0, 1, or 2 inputs depending on type as above; gates & nodes have arbitrary fanout

OUTPUT: An assignment of values $v_1,\ldots, v_n \in [0,1]$ satisfying:

[DGP'06]: Always exists satisfying assignment!

[DGP'06]: but is PPAD-complete to find

:= $y == x_1$

\+ $y == \min\{1, x_1 + x_2\}$

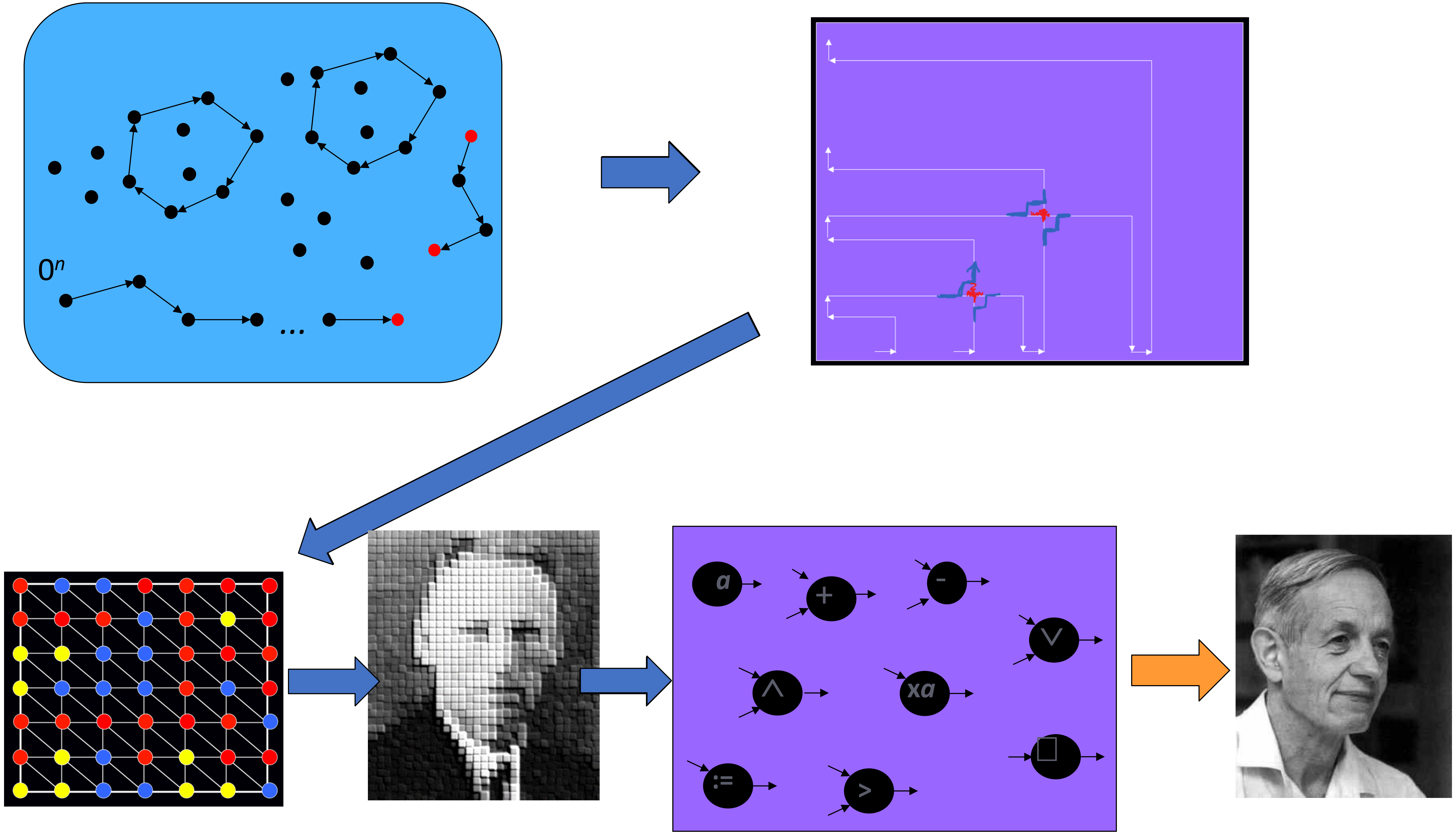
\- $y == \max\{0, x_1 - x_2\}$

a $y == \max\{0, \min\{1, a\}\}$

xa $y == \max\{0, \min\{1, a \cdot x_1\}\}$

\> $y == \begin{cases} 1, & \text{if } x_1 > x_2 \\ 0, & \text{if } x_1 < x_2 \\ *, & \text{if } x_1 = x_2 \end{cases}$

PPAD-Completeness of NASH [Daskalakis, Goldberg, Papadimitriou’06]



ArithmCircuitSAT

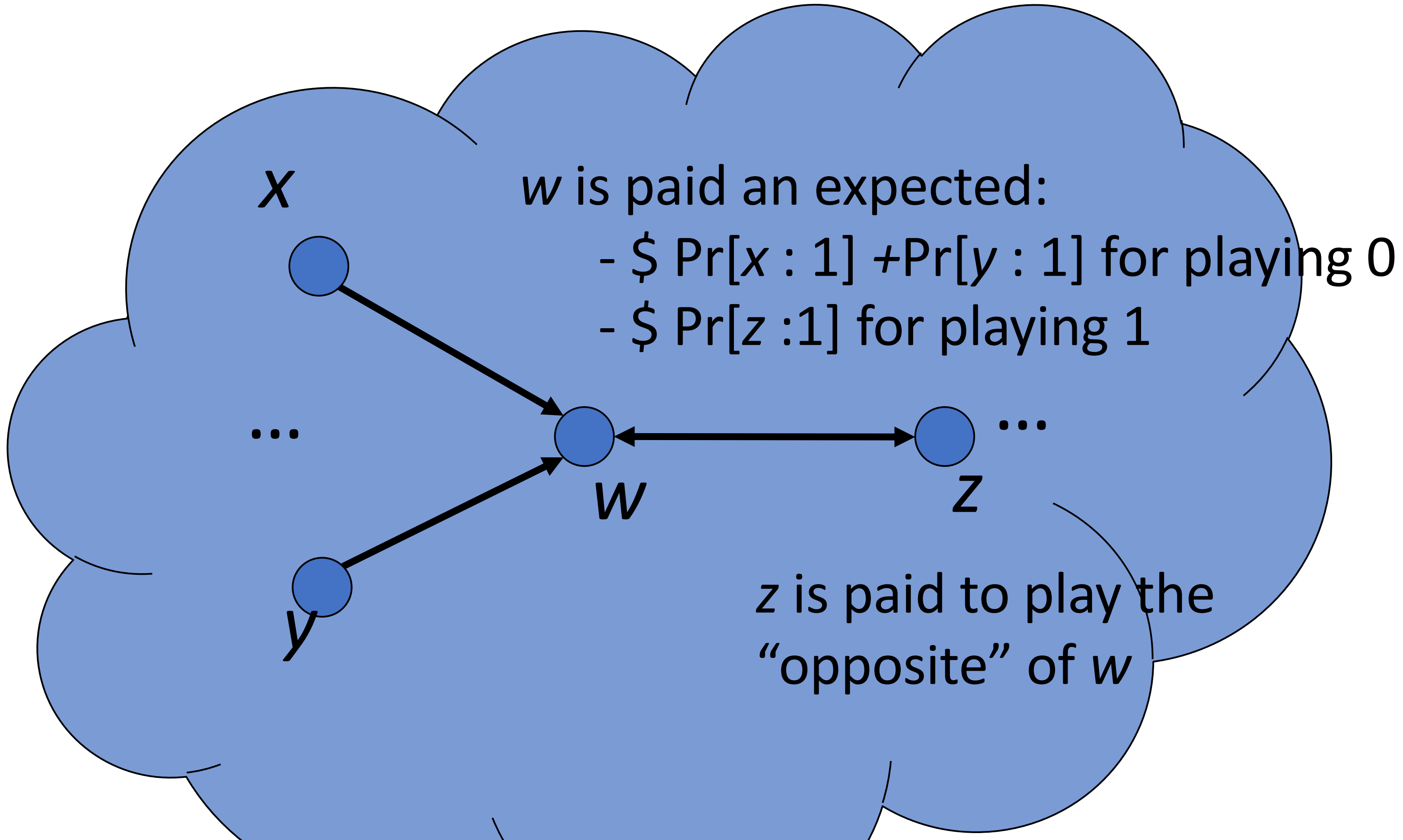
Game Gadgets: Small games performing real arithmetic at their Nash equilibrium.

Addition Gadget

Suppose two strategies per player: {0,1}

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

e.g. ***addition game***

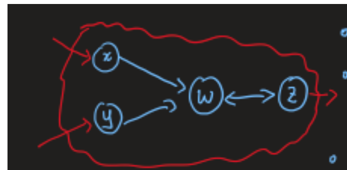


$$u(w:0) = \Pr[x:1] + \Pr[y:1]$$
$$u(w:1) = \Pr[z:1]$$

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

Claim: In any Nash equilibrium of a game containing the above gadget

$$\Pr[z:1] = \min\{\Pr[x:1] + \Pr[y:1], 1\}$$



- all players have strategy set $\{0,1\}$
- player x, y's payoffs depend on other players strategies (not shown here)
- Player w's payoff:
  - if w plays 0, her payoff doesn't depend on z's strategy and depends on x's & y's:

	y:0	y:1
x:0	0	1
x:1	1	2

  - If w plays 1, her payoff doesn't depend on x's or y's strategy and depends on z's as follows

z:0	z:1
0	1

- Player z's payoff:

	z:0	z:1
w:0	0.5	1
w:1	0.5	0

Claim: In all Nash Eq of bigger game containing this it must be that $\Pr[z \text{ plays } 1] = \min\{\Pr[x=1] + \Pr[y=1], 1\}$.

Proof: Suppose $\Pr[z=1] < \min\{\Pr[x=1]+\Pr[y=1], 1\} < \Pr[x=1]+\Pr[y=1]$

$\Rightarrow \Pr[w=0]=1 \Rightarrow \Pr[z=1]=1$ contradiction

Suppose $\Pr[z=1] > \min\{\Pr[x=1]+\Pr[y=1], 1\}$

well $\min\{\Pr[x=1]+\Pr[y=1], 1\}$ cannot be 1 as o.w. $\Pr[z=1] > 1$ impossible

thus $\Pr[z=1] > \Pr[x=1] + \Pr[y=1]$

$\Rightarrow \Pr[w=1]=1 \Rightarrow \Pr[z=1]=0$ contradiction

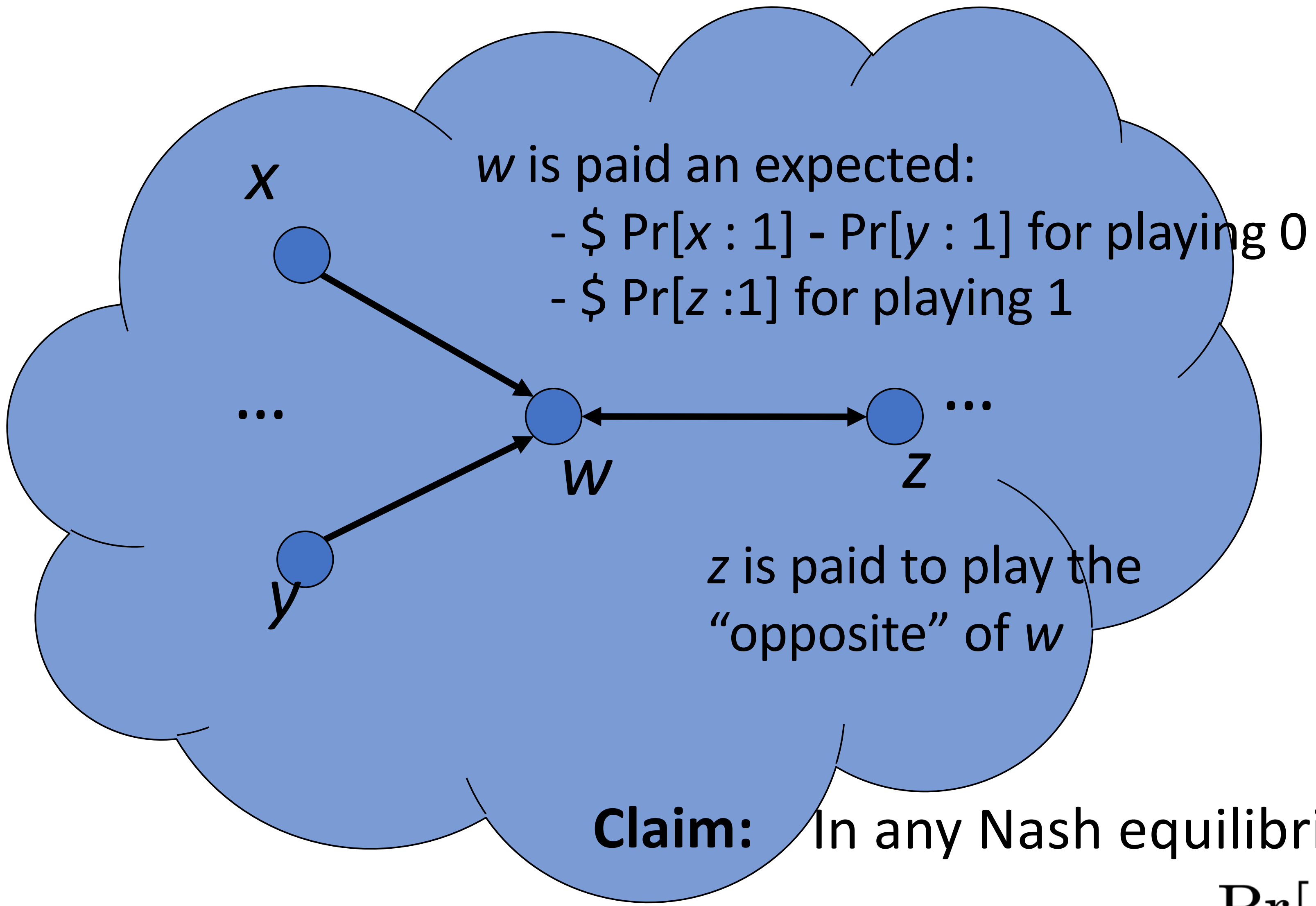
Thus only remaining possibility: $\Pr[z=1] = \min\{\Pr[x=1]+\Pr[y=1], 1\}$ ☺

Subtraction Gadget

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

$$\Pr[z:1] = \max\{0, \Pr[x:1] - \Pr[y:1]\}$$

Notational convention: Use the name of the player and the probability of that player playing 1 interchangeably.

$$x \quad \leftrightsquigarrow \quad \Pr[x : 1]$$

List of Game Gadgets

copy :	$z = x$
addition :	$z = \min\{1, x + y\}$
subtraction :	$z = \max\{0, x - y\}$
set equal to a constant :	$z = \max\{0, \min\{1, \alpha\}\}$
multiply by constant :	$z = \max\{0, \min\{1, \alpha \cdot x\}\}$
comparison :	$z = \begin{cases} 1, & \text{if } x > y \\ 0, & \text{if } x < y \\ *, & \text{if } x = y \end{cases}$

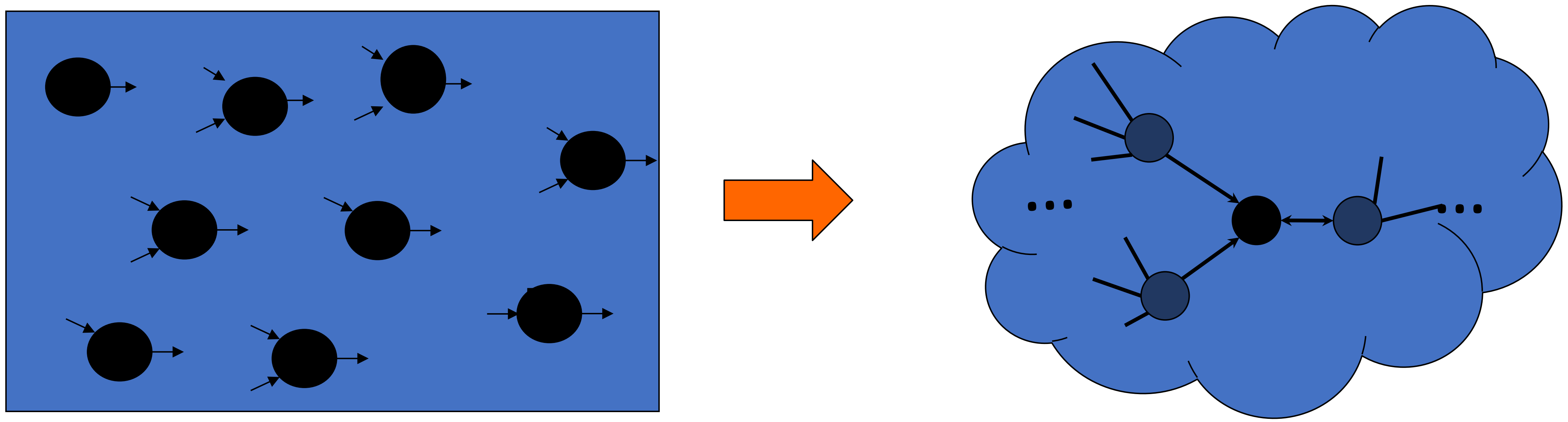
If any of these gadgets is contained in a bigger game, these conditions hold at ***any*** Nash eq. of that bigger game.

Bigger game can only have edges into the "input players" and out of the "output players."

z: "output player" of the gadget

x, ***y***: "input players" of the gadget

PPAD-Completeness of NASH [Daskalakis, Goldberg, Papadimitriou'06]



ARITHMCIRCUITSAT

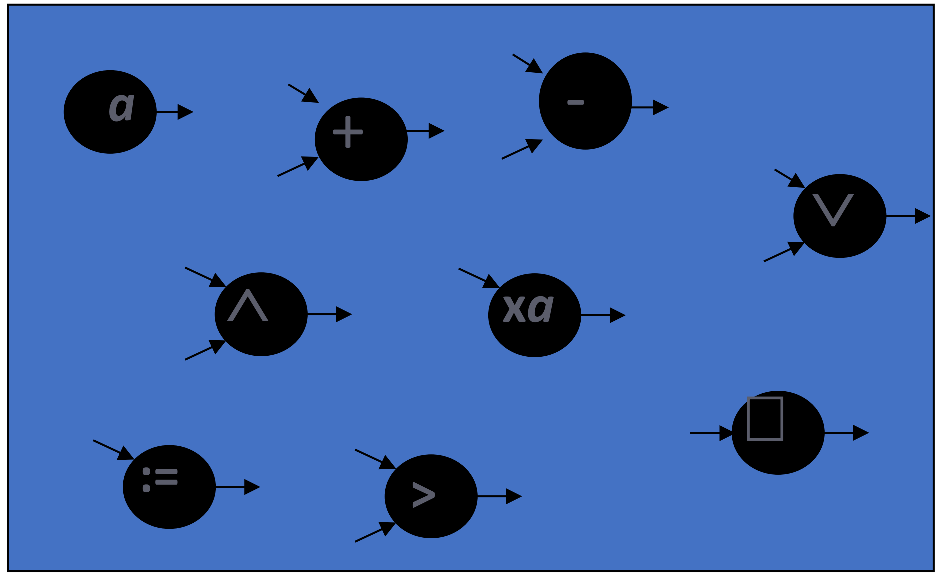
MULTIPLAYERNASH

Given arbitrary instance of ARITHMCIRCUITSAT can create multiplayer game by appropriately composing game gadgets corresponding to each of the gates.

At any Nash equilibrium of resulting game, the gate conditions are satisfied.

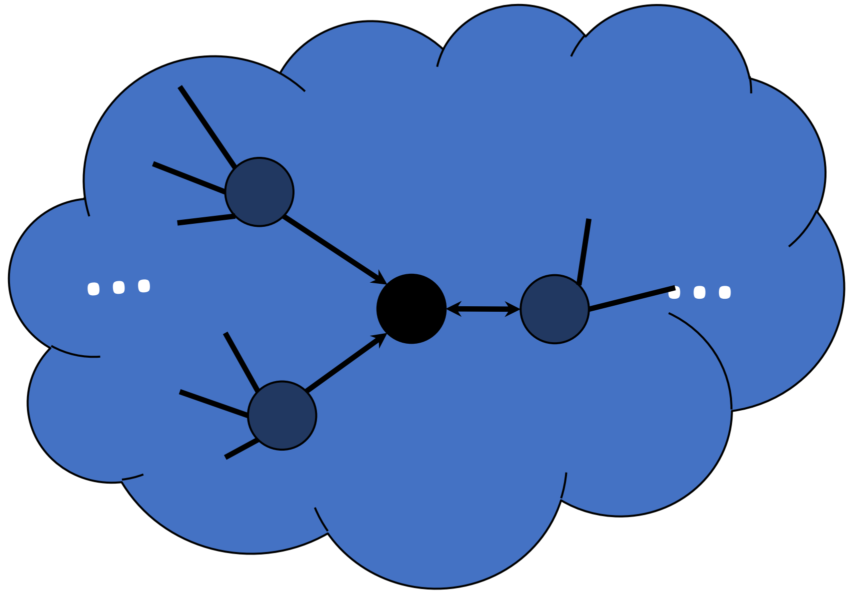
PPAD-Completeness of NASH [Daskalakis, Goldberg, Papadimitriou’06]

DGP=Daskalakis-Goldberg-Papadimitriou



ARITHMCIRCUITSAT

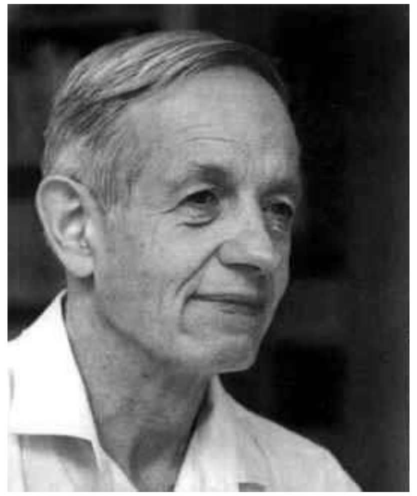
[DGP’06]



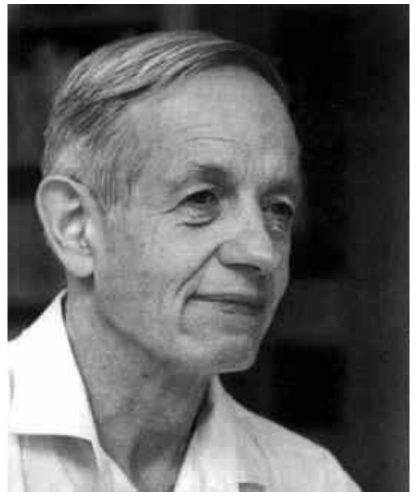
MULTIPLAYERNASH

[DGP’06]

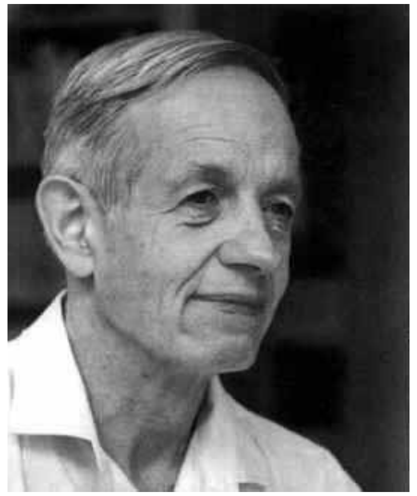
[Chen-Deng’06]



4-PLAYER NASH



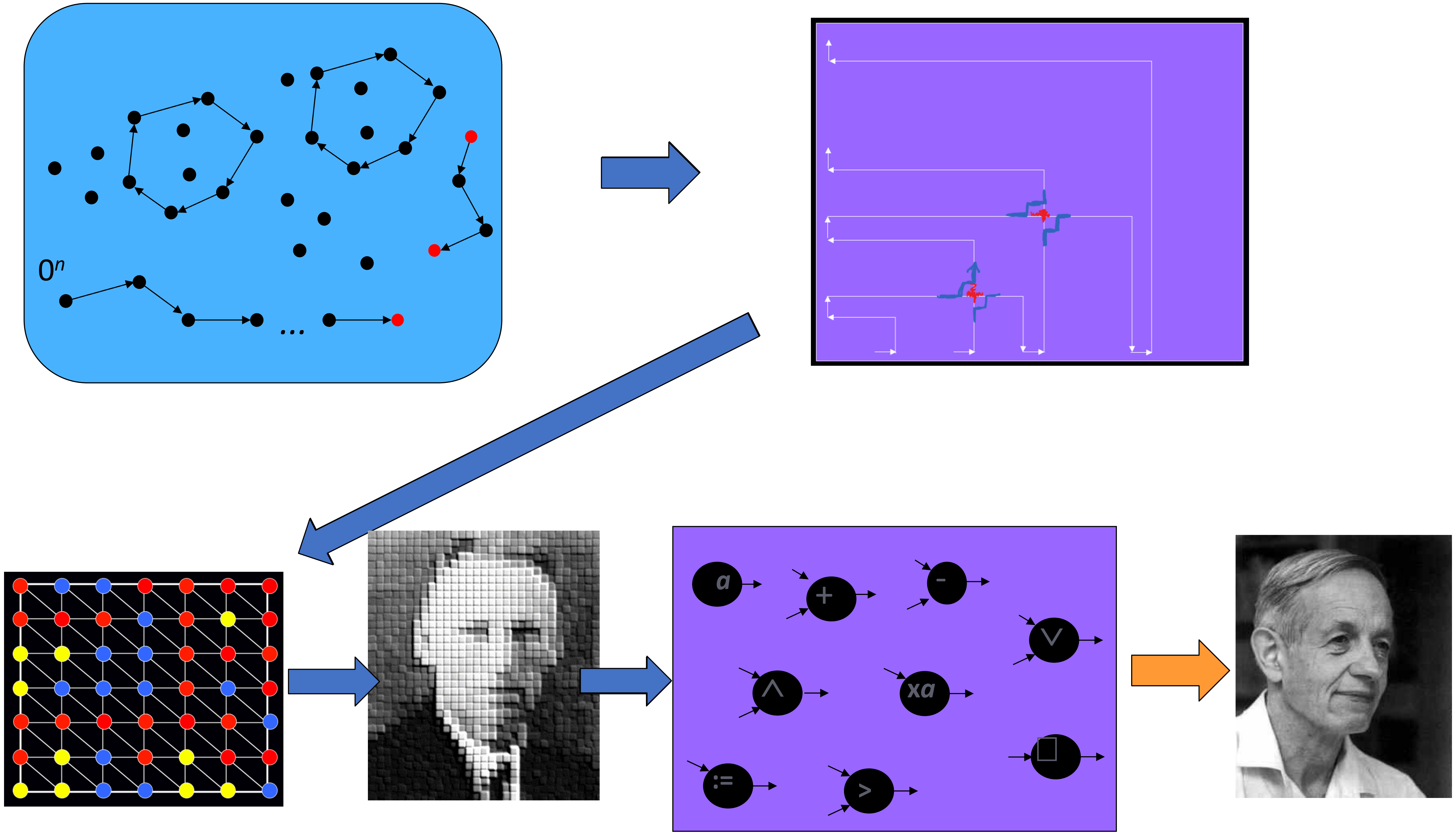
3-PLAYER NASH



2-PLAYER NASH

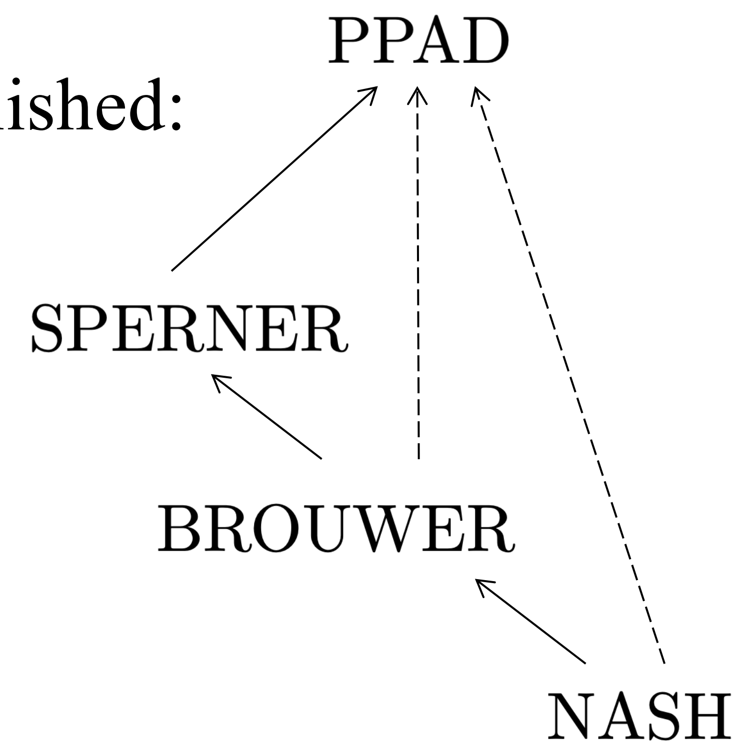
PPAD-Completeness of NASH

[Daskalakis, Goldberg, Papadimitriou'06]

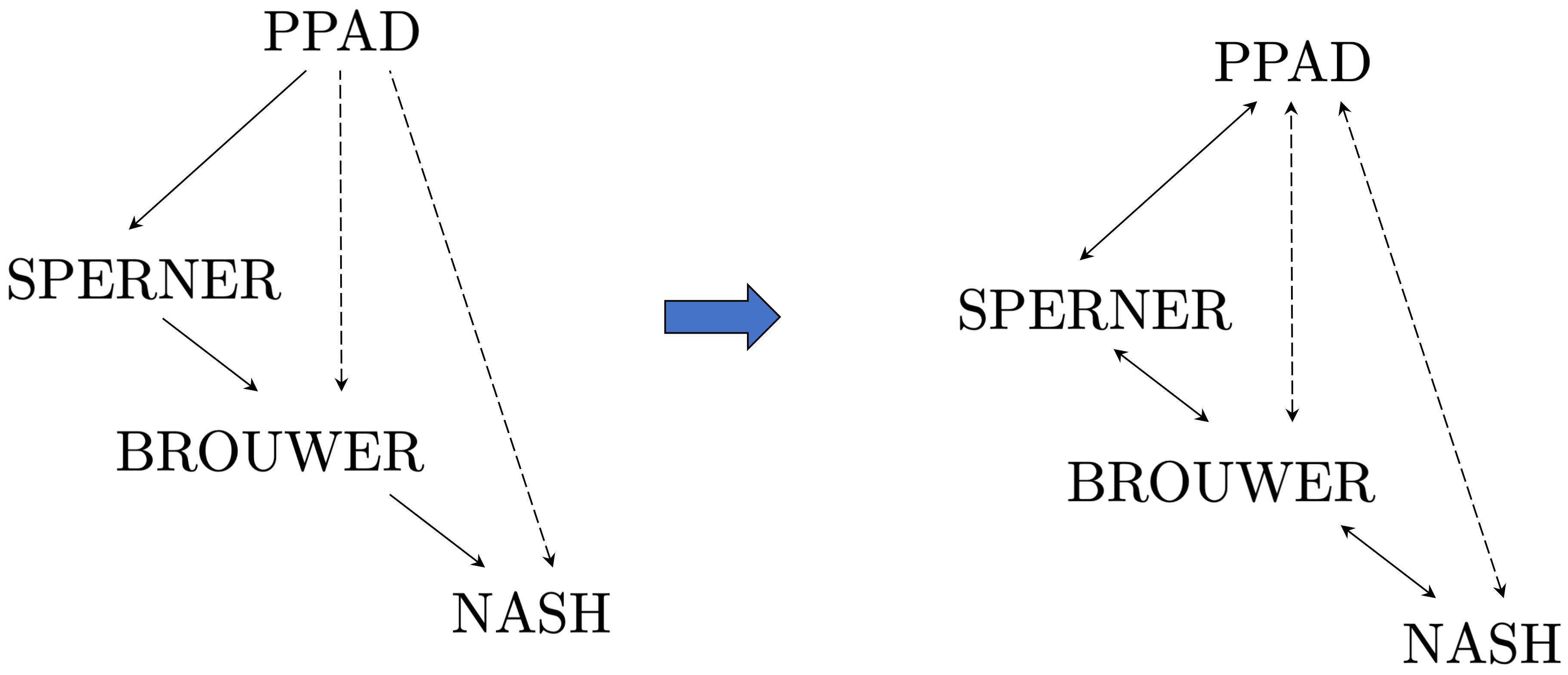


ArithmCircuitSAT

Poly-time Reductions that we just established:



[Daskalakis-Goldberg-Papadimitriou'06]:



Nash Equilibrium Complexity

[John Nash ’50]: A Nash equilibrium exists in every finite game.

Deep influence in Economics, enabling other existence results.

Proof non-constructive (uses Brouwer’s fixed point theorem)

No simpler proof has been discovered

[Daskalakis-Goldberg-Papadimitriou’06]: no simpler proof exists

i.e. **Nash Equilibrium** ⟺ **Brouwer’s Fixed Point Theorem**

Menu

Refresher: Nash, Sperner, Brouwer, PPAD

Total Search Problems in NP

PPAD

PPAD-hardness of NASH

Final Musings

Menu

Refresher: Nash, Sperner, Brouwer, PPAD

Total Search Problems in NP

PPAD

PPAD-hardness of NASH

Final Musings

Other arguments of existence, and resulting complexity classes

"If a graph has a node of odd degree, then it must have another."

PPA

"Every directed acyclic graph must have a sink."

PLS

"If a function maps n elements to n-1 elements, then there is a collision."

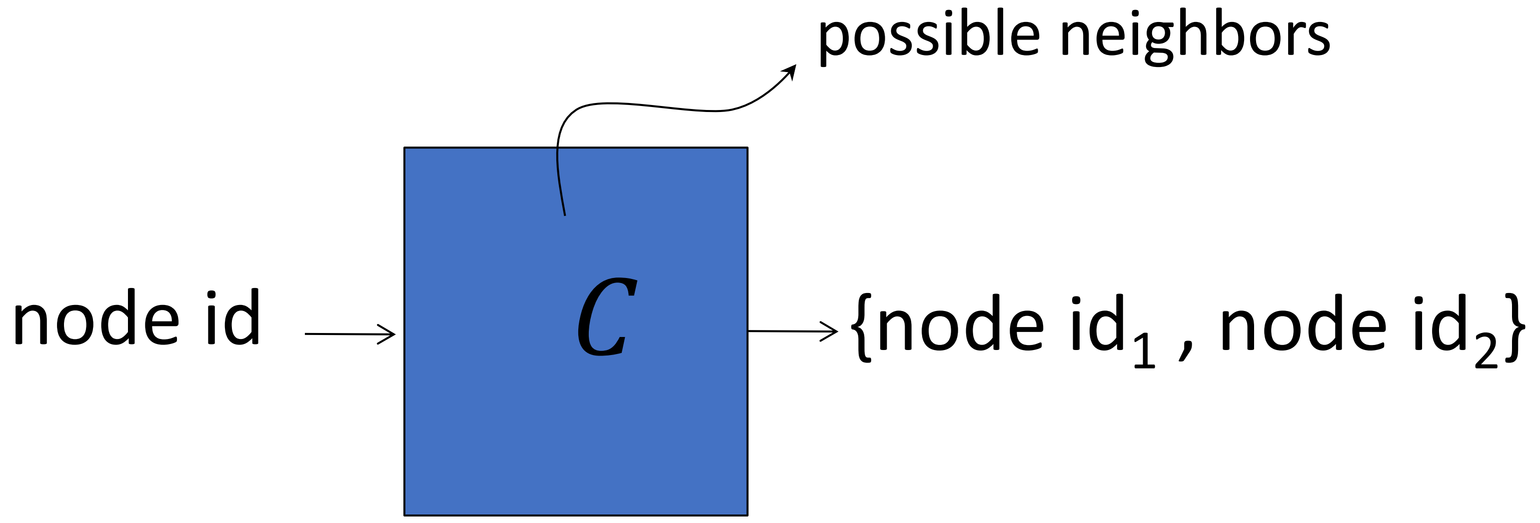
PPP

Formally?

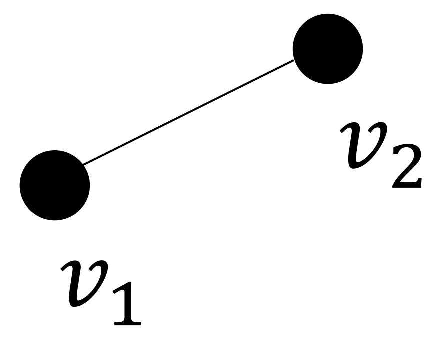
The Class PPA [Papadimitriou '94]

"If a graph has a node of odd degree, then it must have another."

Suppose that an exponentially large graph with vertex set $\{0,1\}^n$ is defined by one circuit:



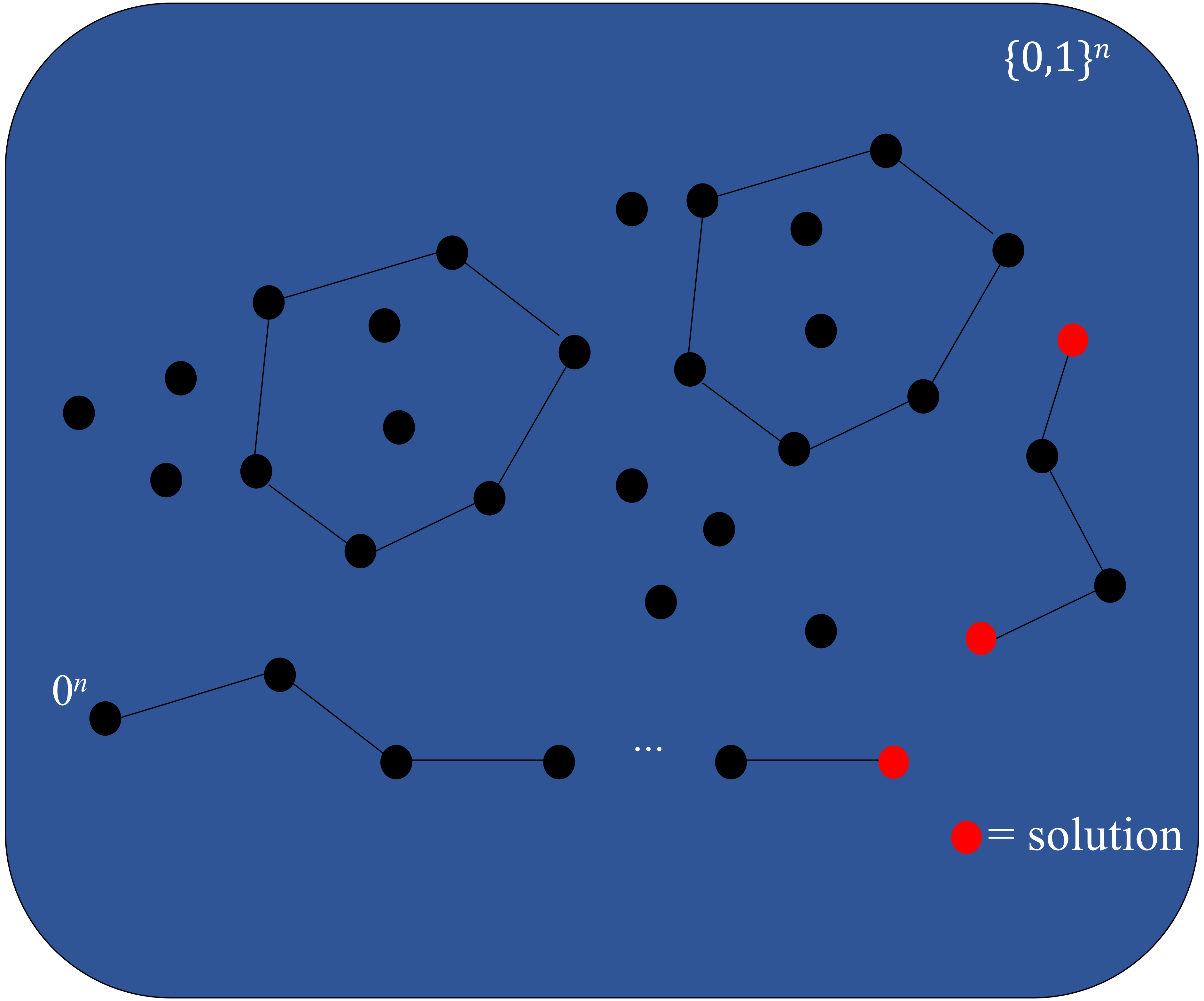
$$v_2 \in C(v_1) \wedge v_1 \in C(v_2)$$



ODDDEGREENODE: Given C: If 0^n has odd degree, find another node with odd degree. Otherwise output 0^n.

PPA = $\{$ *Search problems in* FNP *reducible to* ODDDEGREENODE $\}$

OddDegreeNode



SMITH ∈ PPA

SMITH: Given Hamiltonian cycle in 3-regular graph, find another one.

[Smith]: There must be another one.

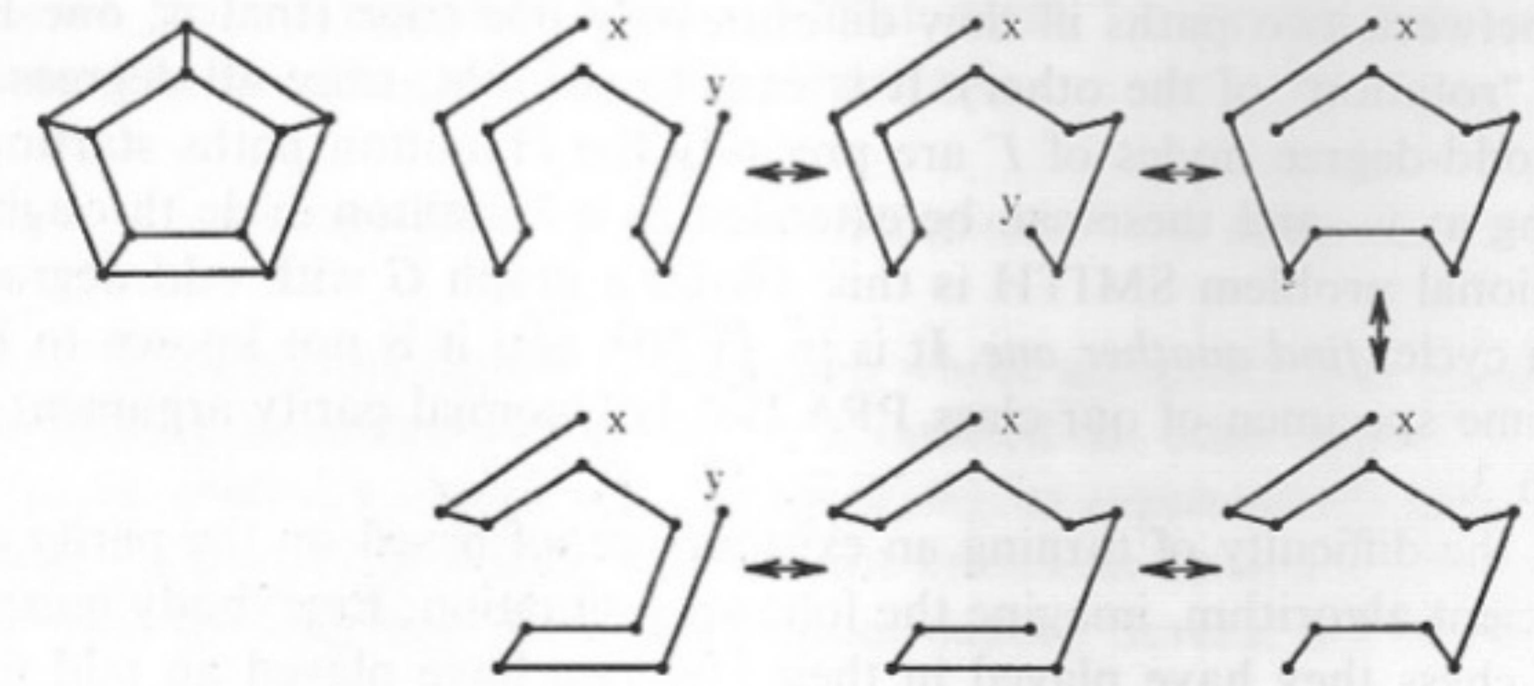
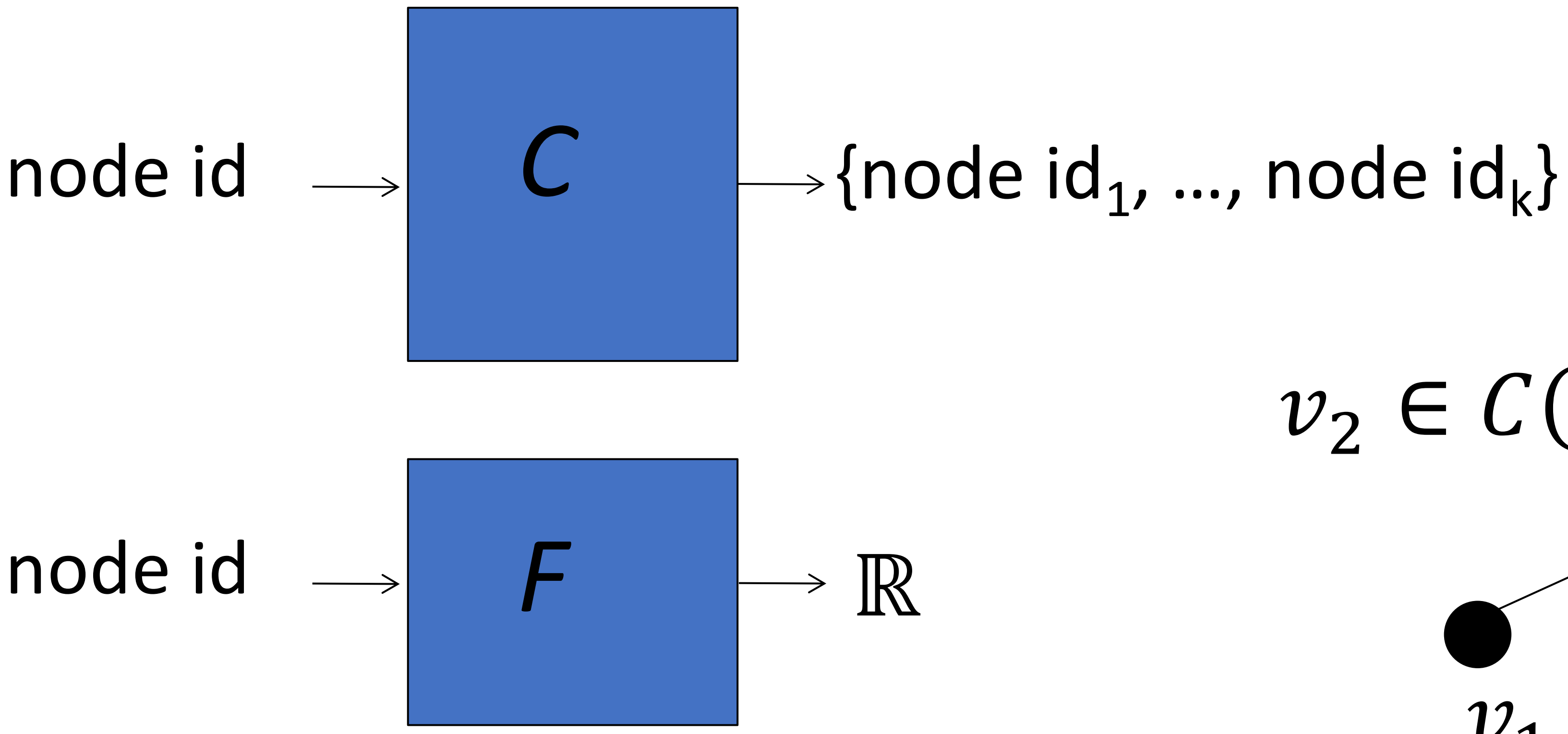


FIG. 1. Smiths theorem in the case of a cubic graph.

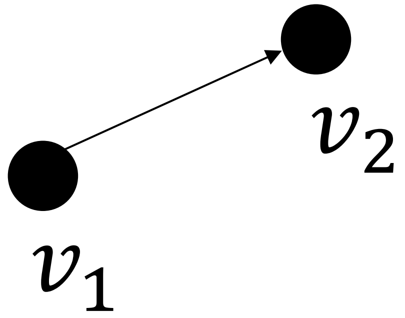
The Class PLS [Johnson-Papapdimitriou-Yannakakis ’89]

“Every DAG has a sink.”

Suppose that a DAG with vertex set $\{0,1\}^n$ is defined by two circuits:



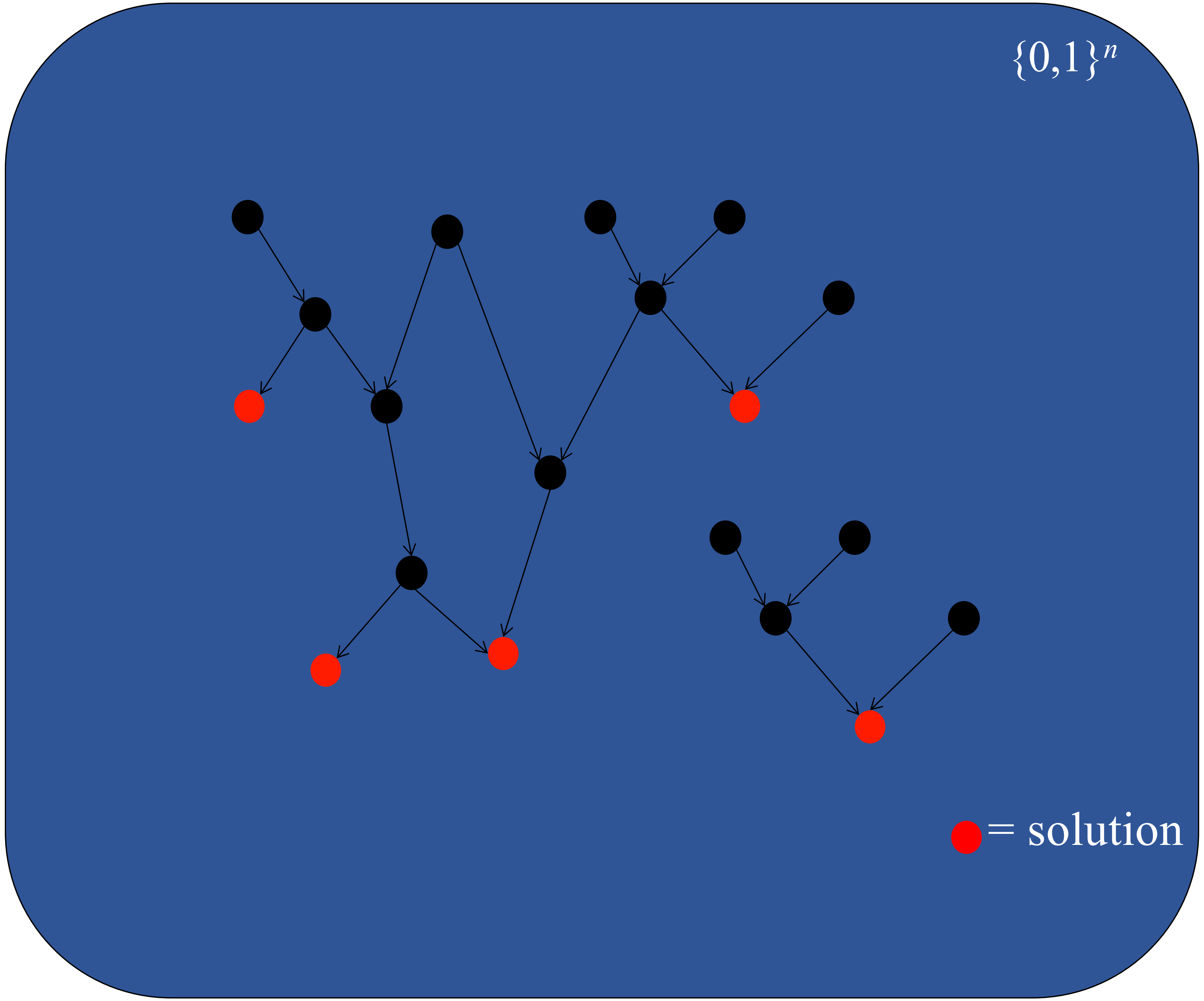
$$v_2 \in C(v_1) \wedge F(v_2) > F(v_1)$$



FINDSINK: Given C, F: Find x s.t. $F(x) \geq F(y)$, for all $y \in C(x)$.

PLS = $\{$ *Search problems in* FNP *reducible to* FINDSINK $\}$

FindSink



LocalMaxCut is PLS-complete

LocalMaxCut:

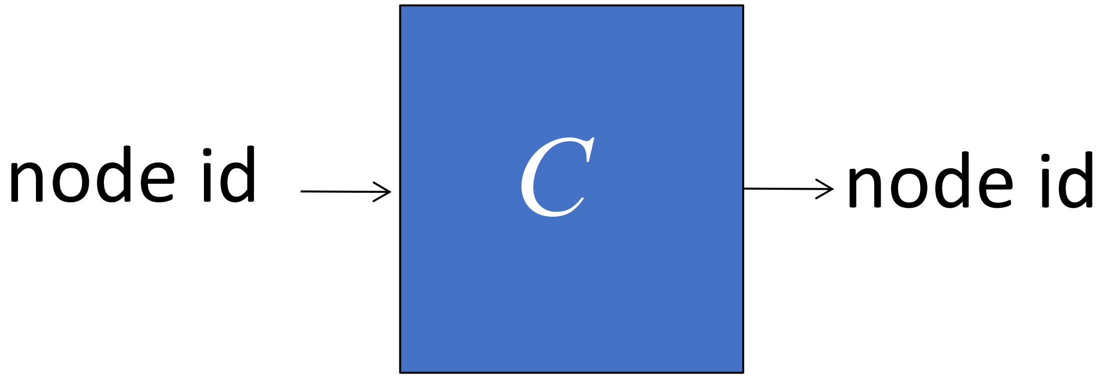
Given weighted graph $\mathrm{G} = (\mathrm{V}, \mathrm{E}, \mathrm{w})$, find a partition $\mathrm{V} = \mathrm{V}_1 \cup V_2$ that is locally optimal (i.e. can't move any single vertex to the other side to increase the cut size).

[Schaffer-Yannakakis'91]: LocalMaxCut is PLS-complete.

The Class PPP [Papadimitriou ’94]

“If a function maps n elements to $n-1$ elements, then there is a collision.”

Suppose that an exponentially large graph with vertex set $\{0,1\}^n$ is defined by one circuit:

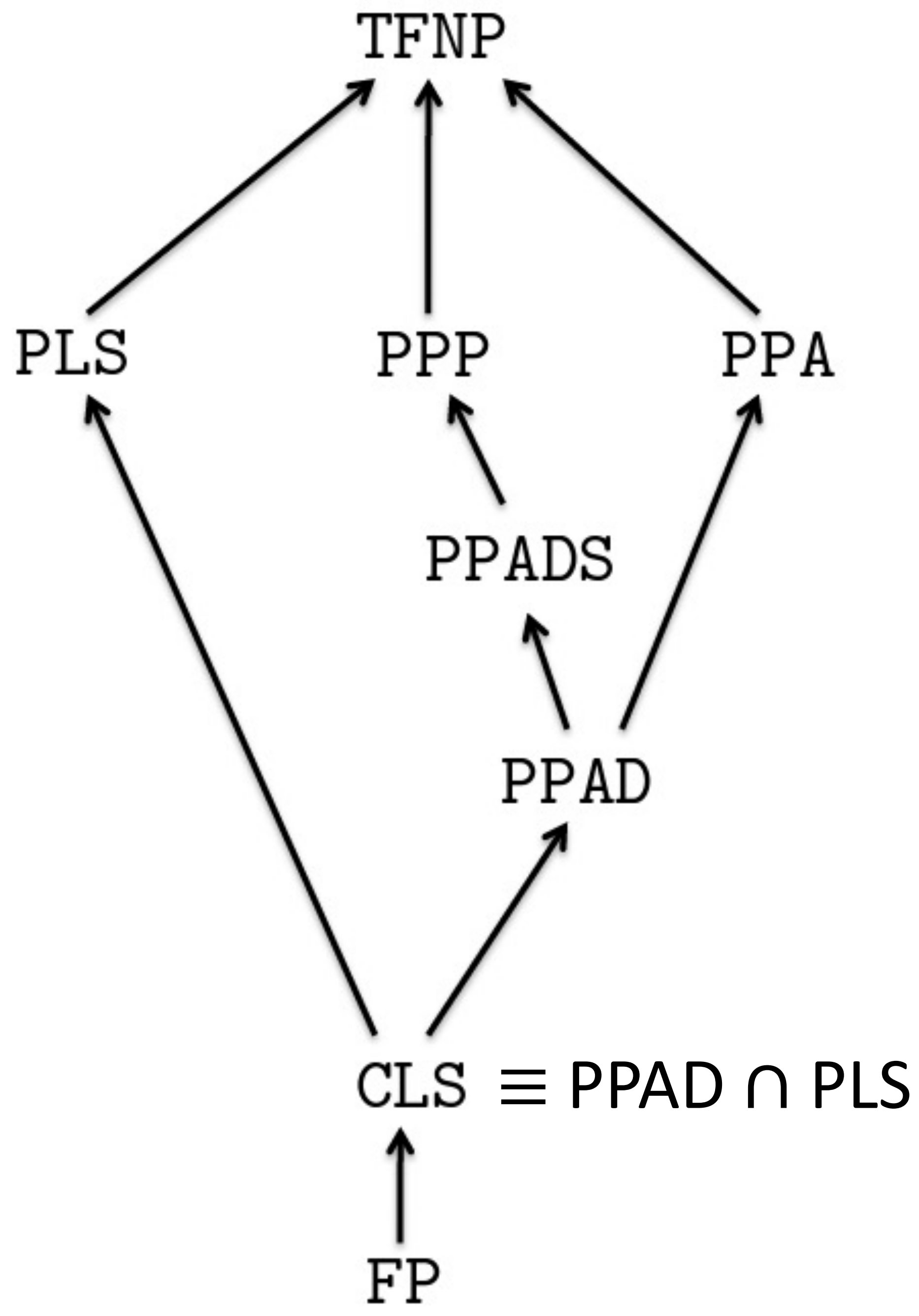


Collision Given C: Find x s.t. $C(x) = 0^n$; or find $x \neq y$ s.t. $C(x) = C(y)$.

PPP = $\{$ *Search problems in FNP reducible to* **Collision** $\}$

in PPP: Factoring
PPP-complete: constrained Short-Integer-Solution

in PPA: Smith, Factoring
PPA-complete: concensus halving, fixed points in unorientable spaces, combinatorial nullstellensatz, chevalley-warning, Necklace Splitting, Discrete Ham Sandwich



CLS: continuous local search, capturing e.g. fixed points of gradient descent [Daskalakis-Papadimitriou’11]

CLS ≡ PPAD ∩ PLS shown by [Fearnley-Goldber-Hollender-Savani’21]