Total Search Problems in Game Theory and Economics

Aris Filos-Ratsikas, University of Edinburgh

Alexandros Hollender, University of Oxford, EPFL

Tutorial SAGT 2022

Focus on PPAD

Some PPAD-complete Problems

Some PPAD-complete Problems

- NASH
  - $k \geq 2$ players [Daskalakis-Goldberg-Papadimitriou 09, Chen-Deng-Teng 09]

Some PPAD-complete Problems

- NASH
  - $k \geq 2$ players [Daskalakis-Goldberg-Papadimitriou 09, Chen-Deng-Teng 09]
  - Constant rank [Mehta 18], Anonymous [Chen-Durfee-Orfanou 15], Sparse [Chen-Deng-Teng 09]

Some PPAD-complete Problems

- NASH
  - $k \geq 2$ players [Daskalakis-Goldberg-Papadimitriou 09, Chen-Deng-Teng 09]
  - Constant rank [Mehta 18], Anonymous [Chen-Durfee-Orfanou 15], Sparse [Chen-Deng-Teng 09]
  - Constant inapproximability for n players: polymatrix, graphical games [Rubinstein 18]

Some PPAD-complete Problems

- NASH
  - $k \geq 2$ players [Daskalakis-Goldberg-Papadimitriou 09, Chen-Deng-Teng 09]
  - Constant rank [Mehta 18], Anonymous [Chen-Durfee-Orfanou 15], Sparse [Chen-Deng-Teng 09]
  - Constant inapproximability for n players: polymatrix, graphical games [Rubinstein 18]
- Market Equilibrium
  - Arrow-Debreu [Codenotti-Saberi-Varadarajan-Ye 08, Chen-Dai-Du-Teng 09]

Some PPAD-complete Problems

- NASH
  - $k \geq 2$ players [Daskalakis-Goldberg-Papadimitriou 09, Chen-Deng-Teng 09]
  - Constant rank [Mehta 18], Anonymous [Chen-Durfee-Orfanou 15], Sparse [Chen-Deng-Teng 09]
  - Constant inapproximability for n players: polymatrix, graphical games [Rubinstein 18]
- Market Equilibrium
  - Arrow-Debreu [Codenotti-Saberi-Varadarajan-Ye 08, Chen-Dai-Du-Teng 09]
  - Fisher [Vazirani-Yannakakis 11]

Some PPAD-complete Problems

- NASH
  - $k \geq 2$ players [Daskalakis-Goldberg-Papadimitriou 09, Chen-Deng-Teng 09]
  - Constant rank [Mehta 18], Anonymous [Chen-Durfee-Orfanou 15], Sparse [Chen-Deng-Teng 09]
  - Constant inapproximability for n players: polymatrix, graphical games [Rubinstein 18]
- Market Equilibrium
  - Arrow-Debreu [Codenotti-Saberi-Varadarajan-Ye 08, Chen-Dai-Du-Teng 09]
  - Fisher [Vazirani-Yannakakis 11]
- Cake Cutting
  - with general preferences [Deng-Qi-Saberi 12]

Why do we think PPAD is hard?

What reasons are there to believe that PPAD ≠ P?

Why do we think PPAD is hard?

What reasons are there to believe that PPAD ≠ P?

- many seemingly hard problems lie in PPAD…

Why do we think PPAD is hard?

What reasons are there to believe that PPAD ≠ P?

- many seemingly hard problems lie in PPAD…
- oracle separations

Why do we think PPAD is hard?

What reasons are there to believe that PPAD ≠ P?

- many seemingly hard problems lie in PPAD…
- oracle separations
- hard under cryptographic assumptions

Focus on PPAD

I. How do we prove PPAD-membership?

II. How do we prove PPAD-hardness?

Focus on PPAD

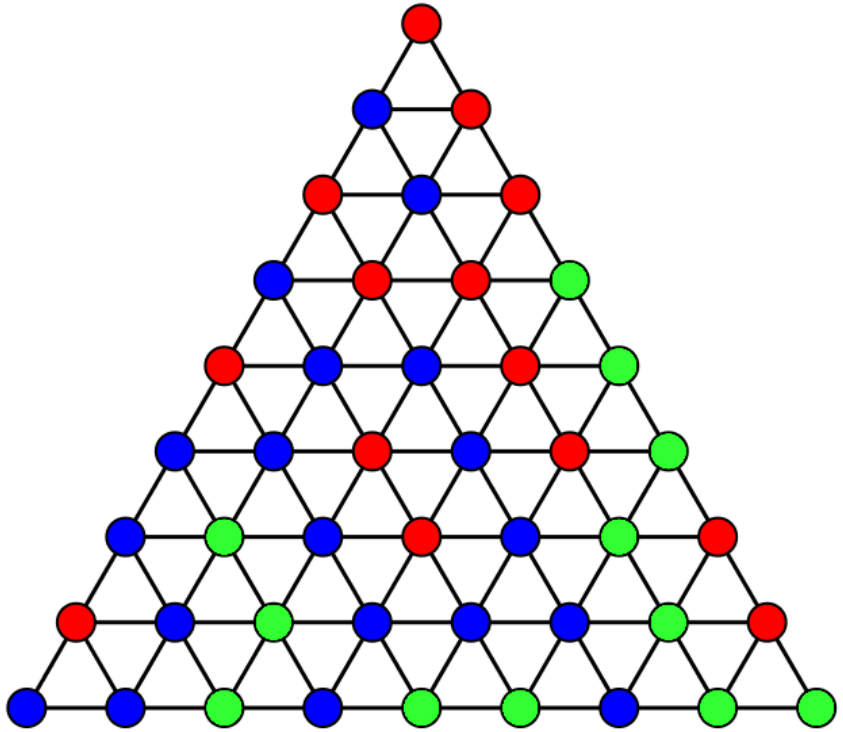
I. How do we prove PPAD-membership?
   1. **Sperner's Lemma**

II. How do we prove PPAD-hardness?

Sperner's Lemma

Sperner's Lemma (Sperner 1928):

Triangulation of a big triangle.



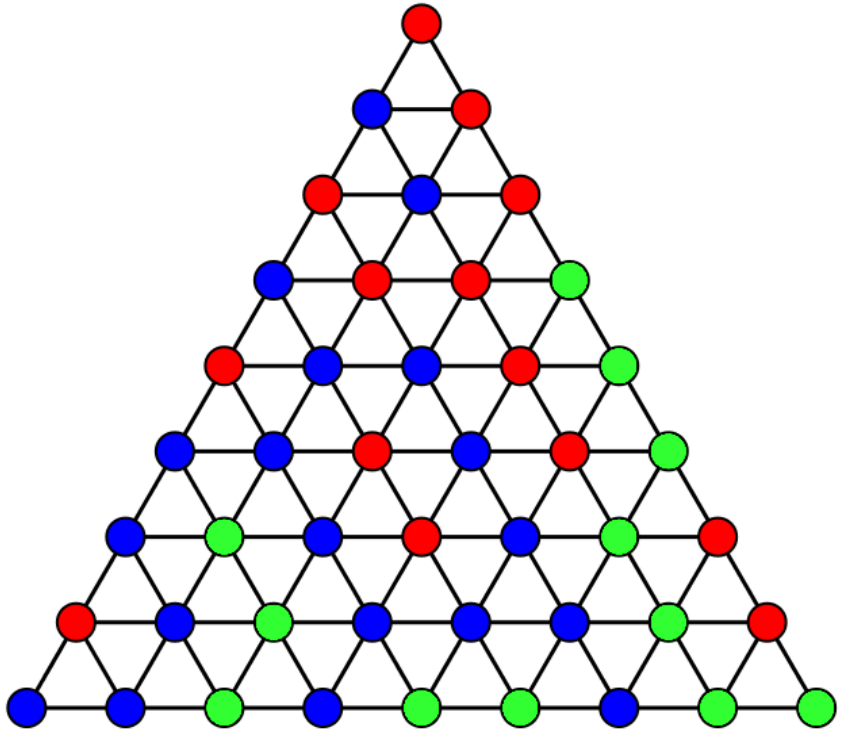
Sperner's Lemma

Sperner's Lemma (Sperner 1928):

Triangulation of a big triangle.

Sperner coloring (3 colors):

vertices of facet j do not receive the color j

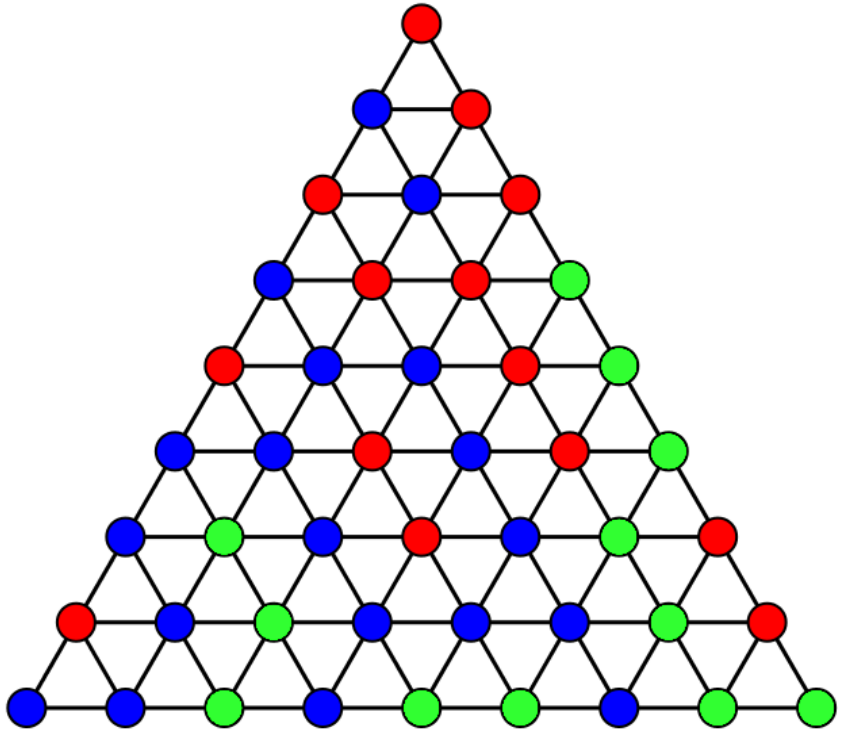


Sperner's Lemma

Sperner's Lemma (Sperner 1928):

Triangulation of a big triangle.

Sperner coloring (3 colors):
vertices of facet j do not receive the color j



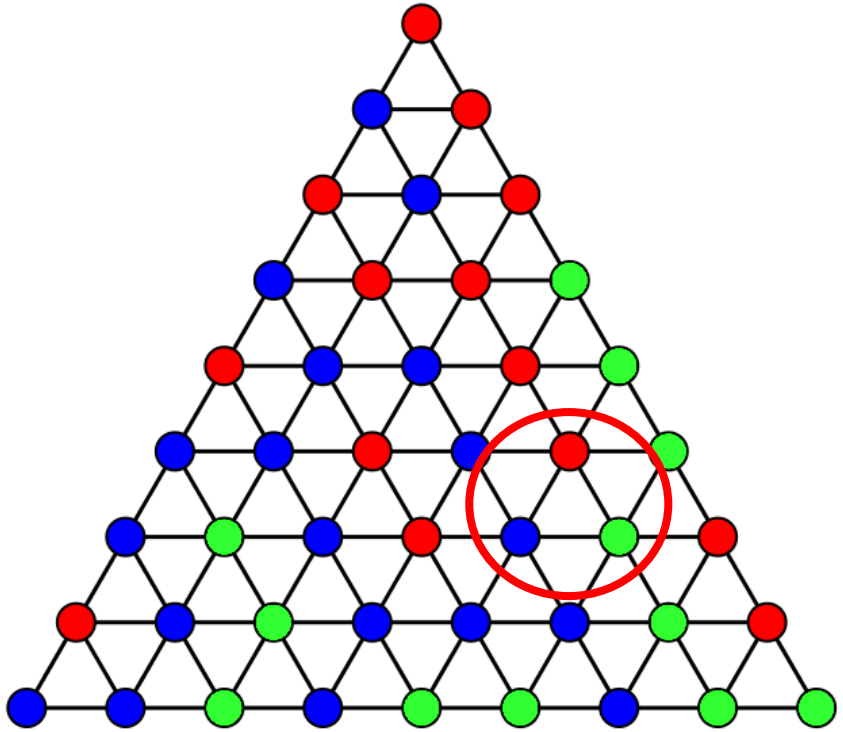
→ There always exists a trichromatic small triangle.

Sperner's Lemma

Sperner's Lemma (Sperner 1928):

Triangulation of a big triangle.

Sperner coloring (3 colors):
vertices of facet j do not receive the color j



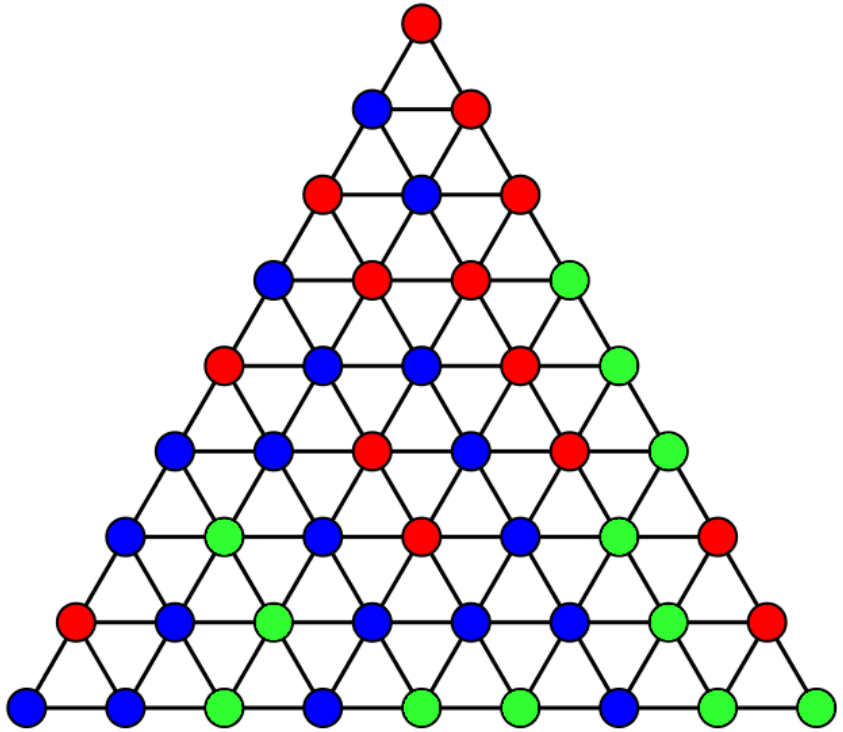
→ There always exists a trichromatic small triangle.

Sperner's Lemma

Sperner's Lemma (Sperner 1928):

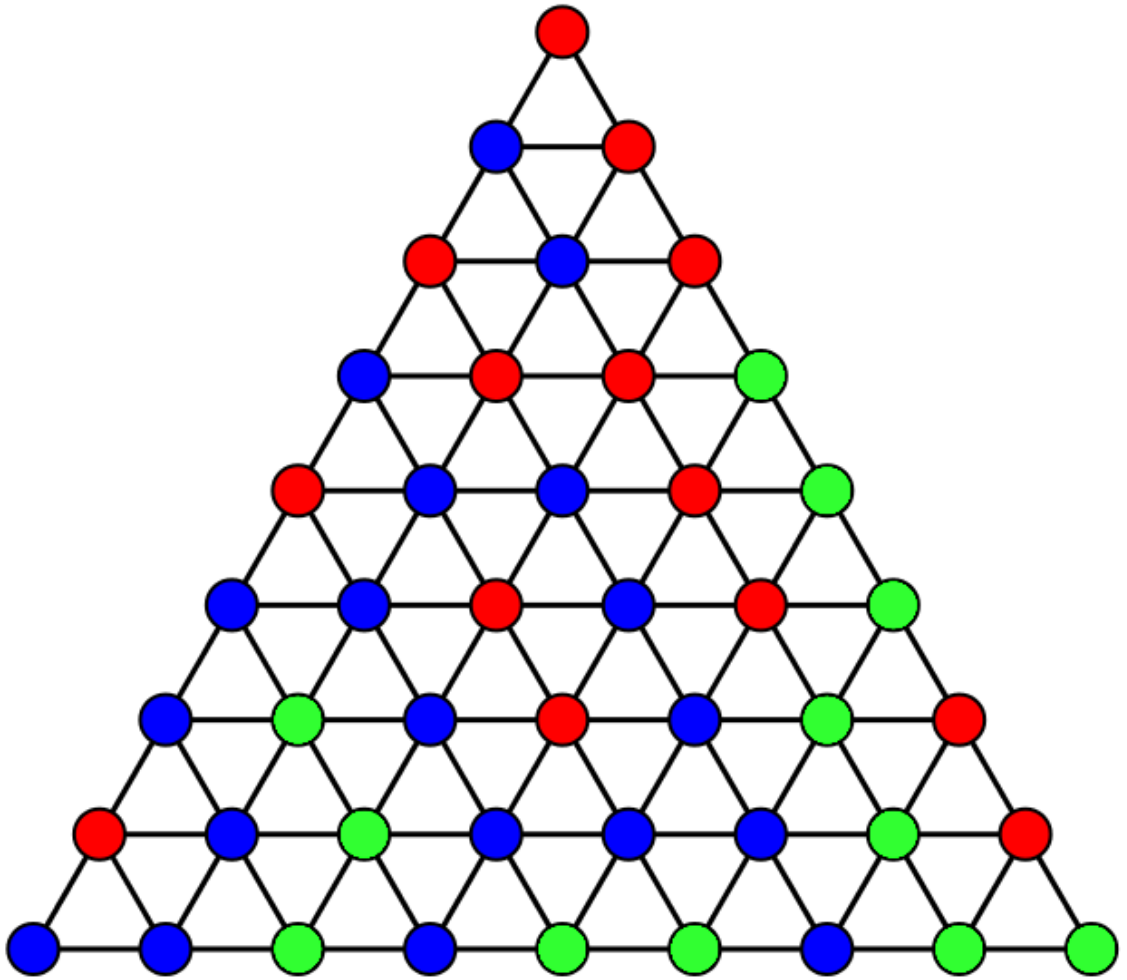
Triangulation of a d-dimensional simplex.

Sperner coloring ($d + 1$ colors):
vertices of facet j do not receive the color j

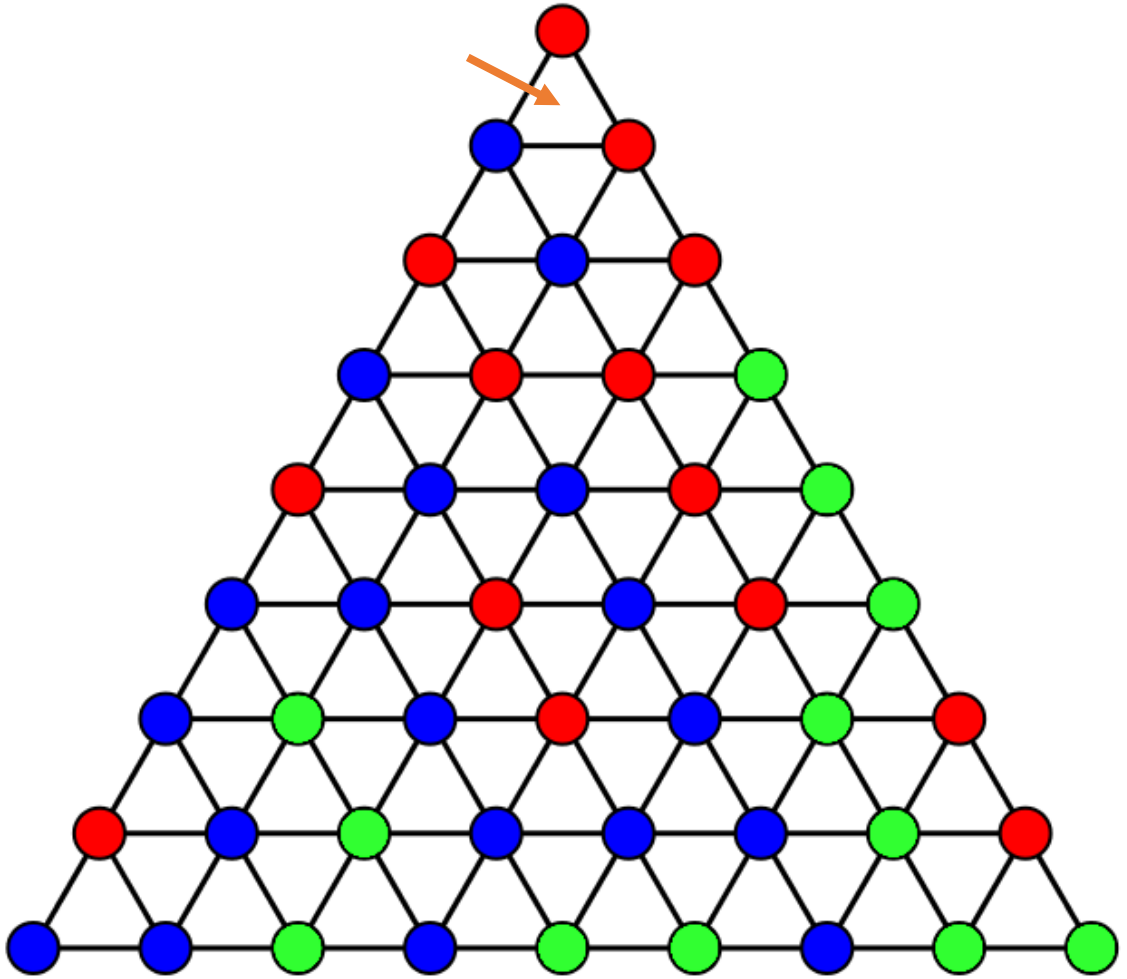


→ There always exists a panchromatic small simplex.

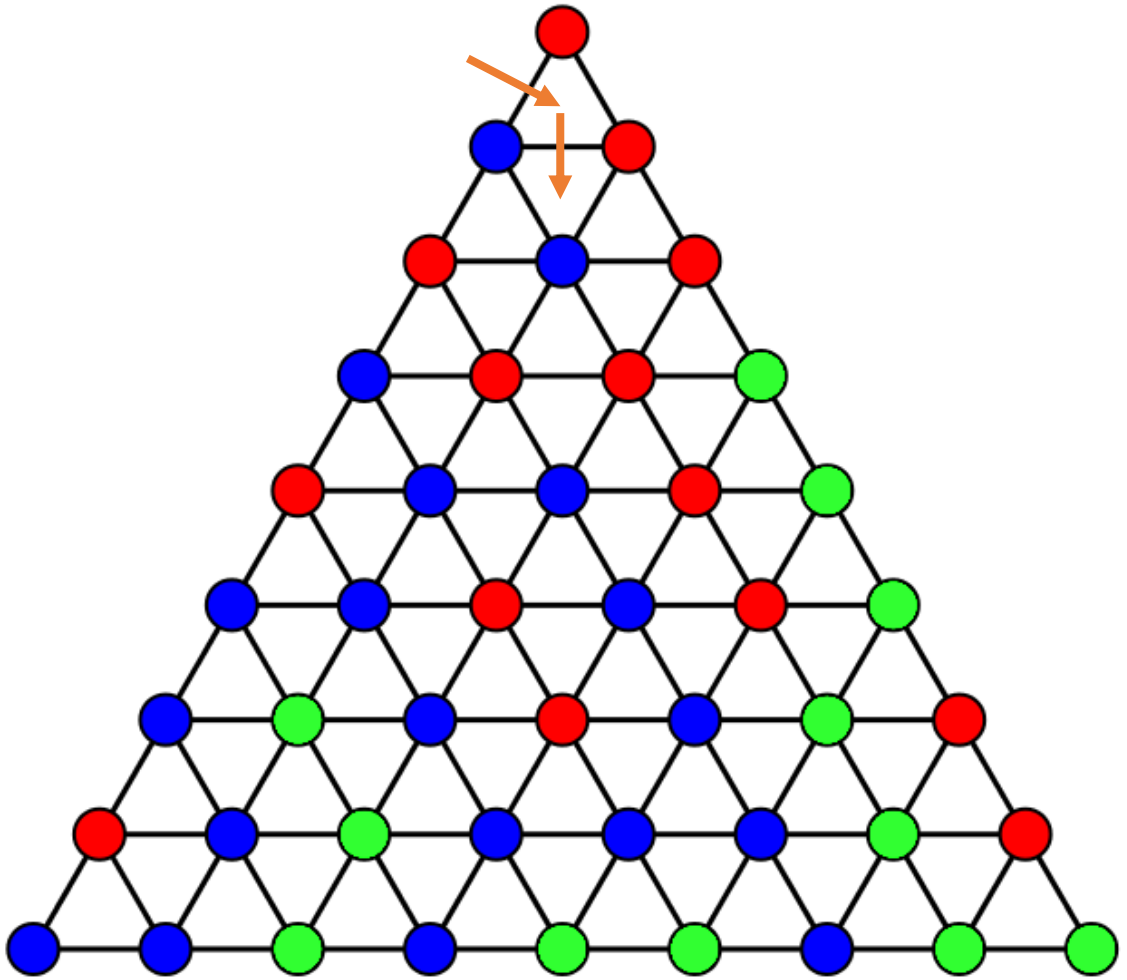
A Proof of Sperner's Lemma



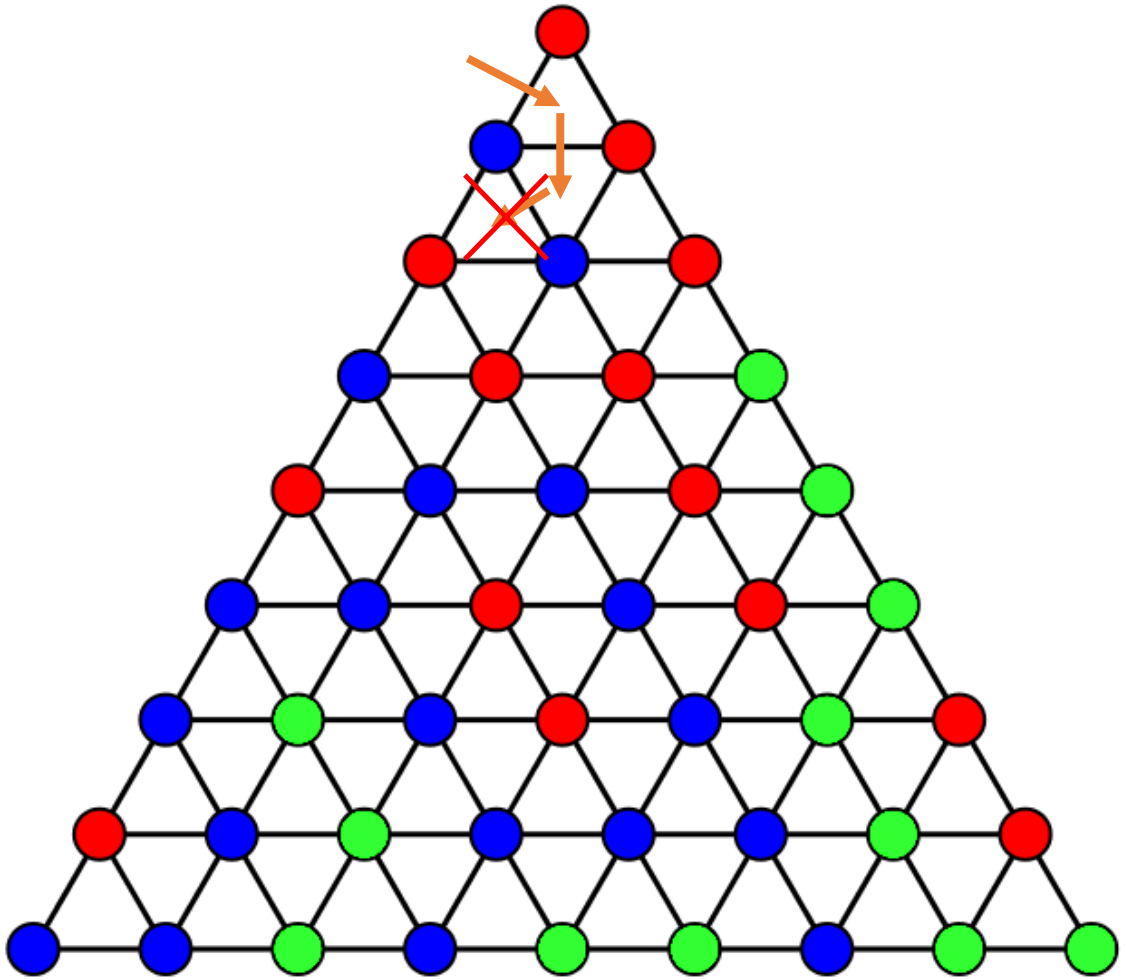
A Proof of Sperner's Lemma



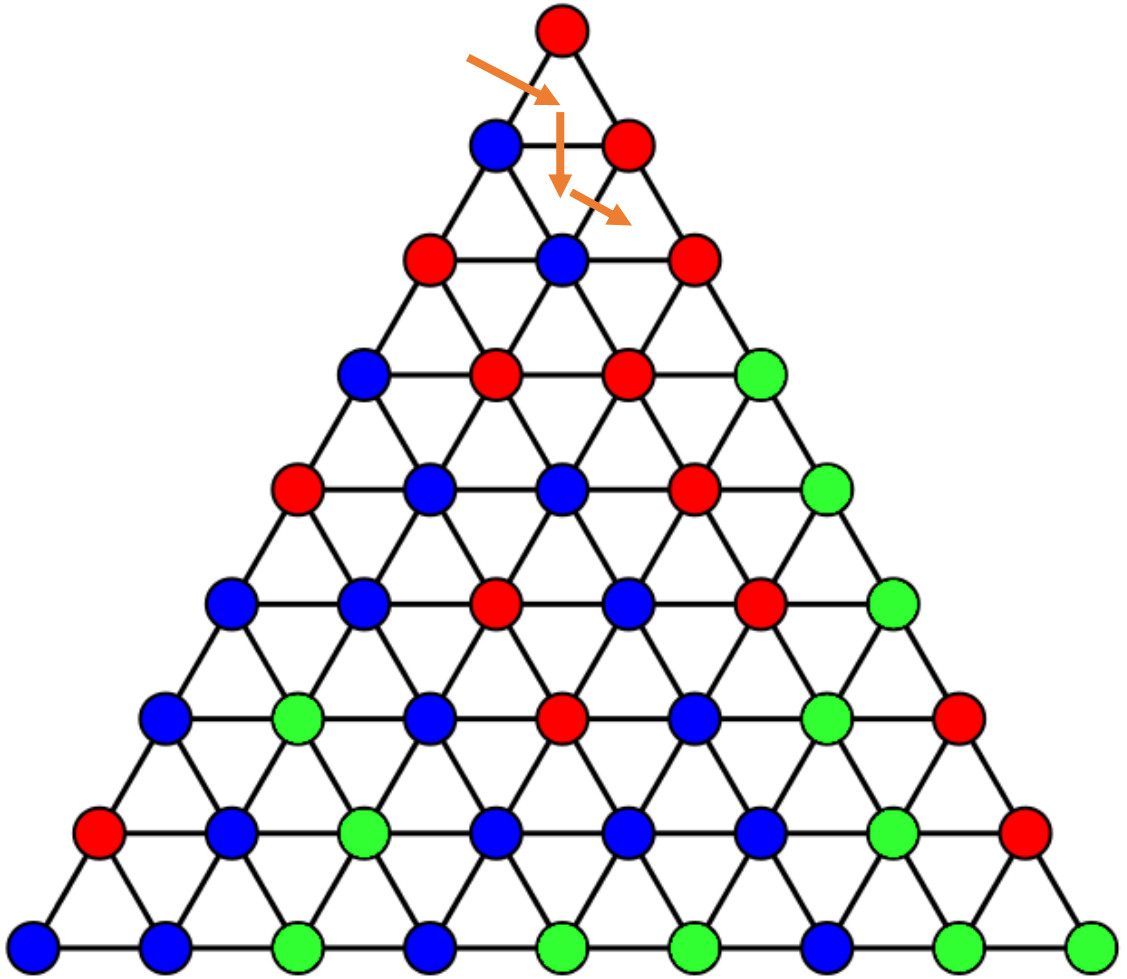
A Proof of Sperner's Lemma



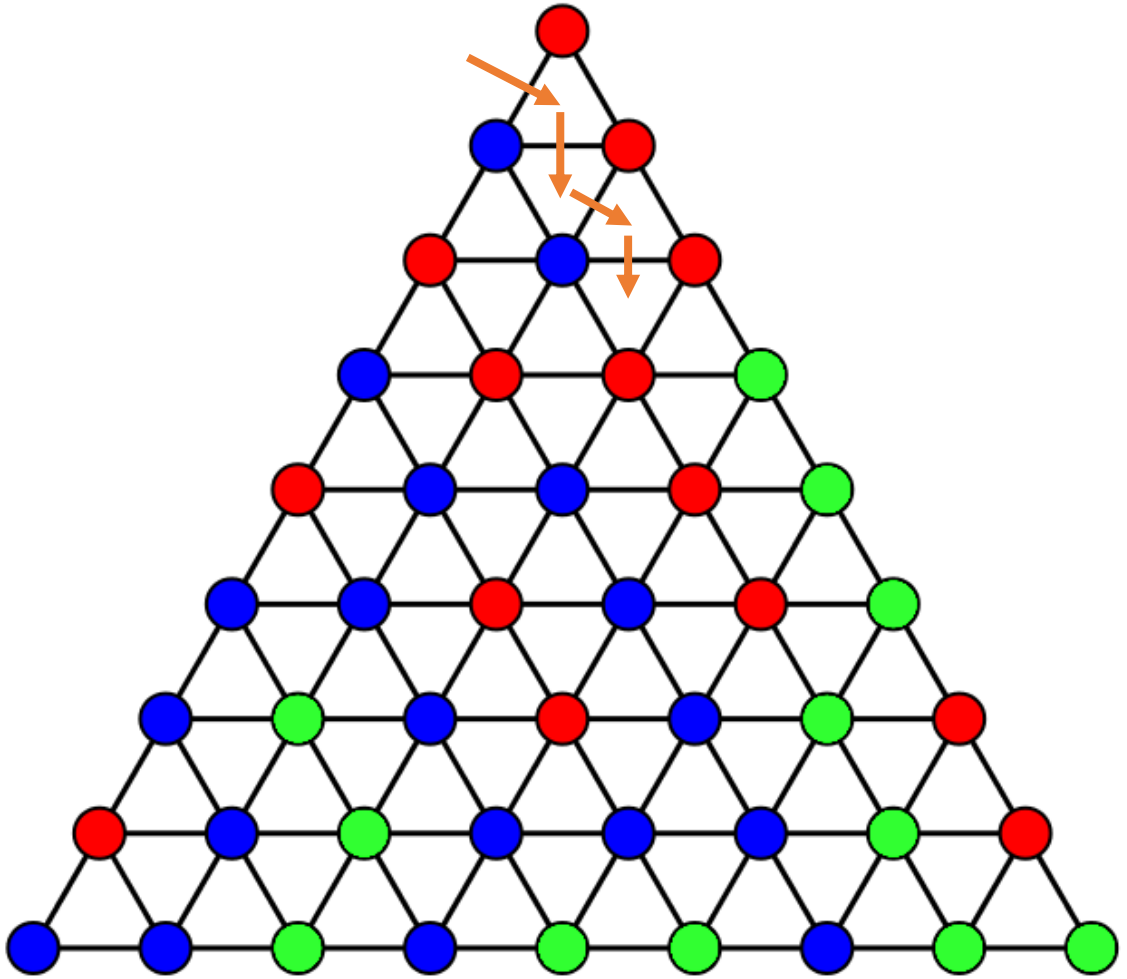
A Proof of Sperner's Lemma



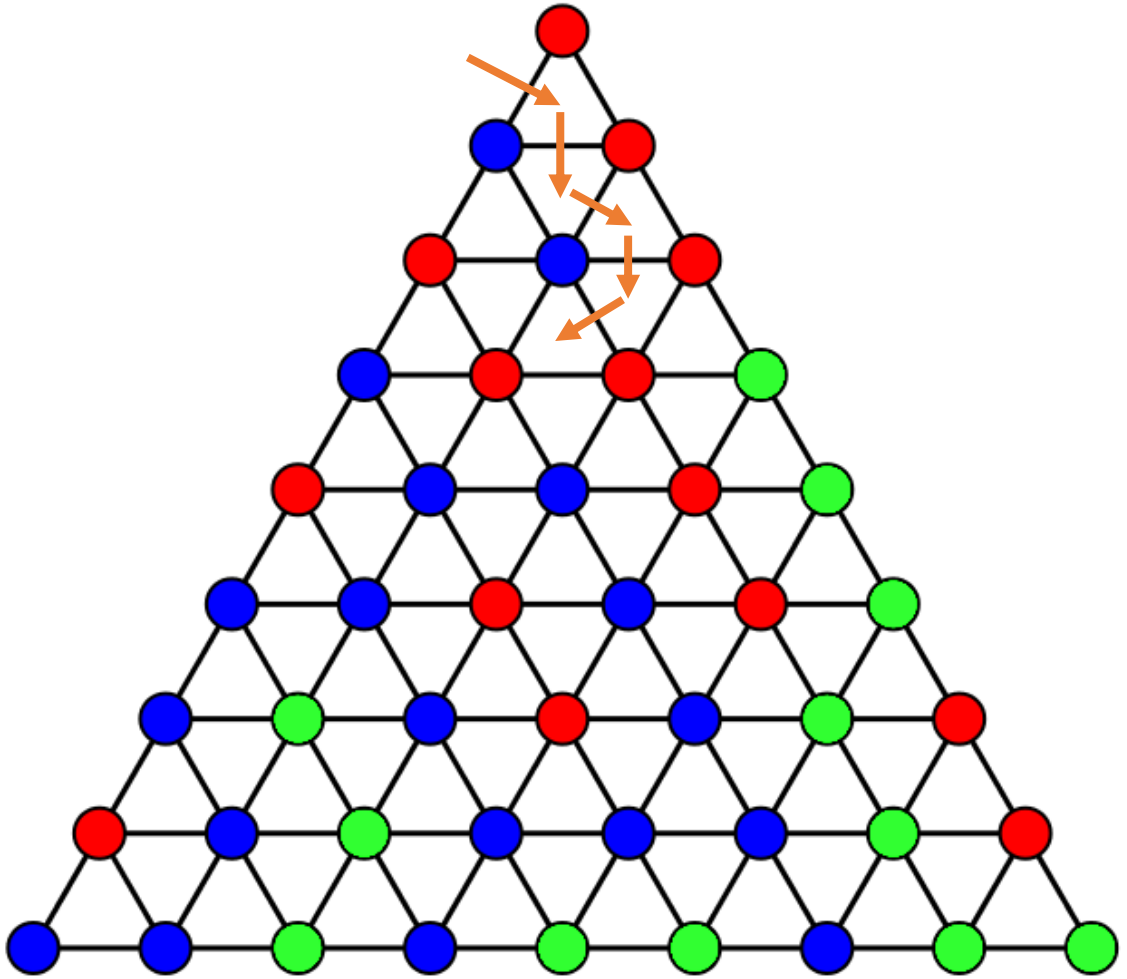
A Proof of Sperner's Lemma



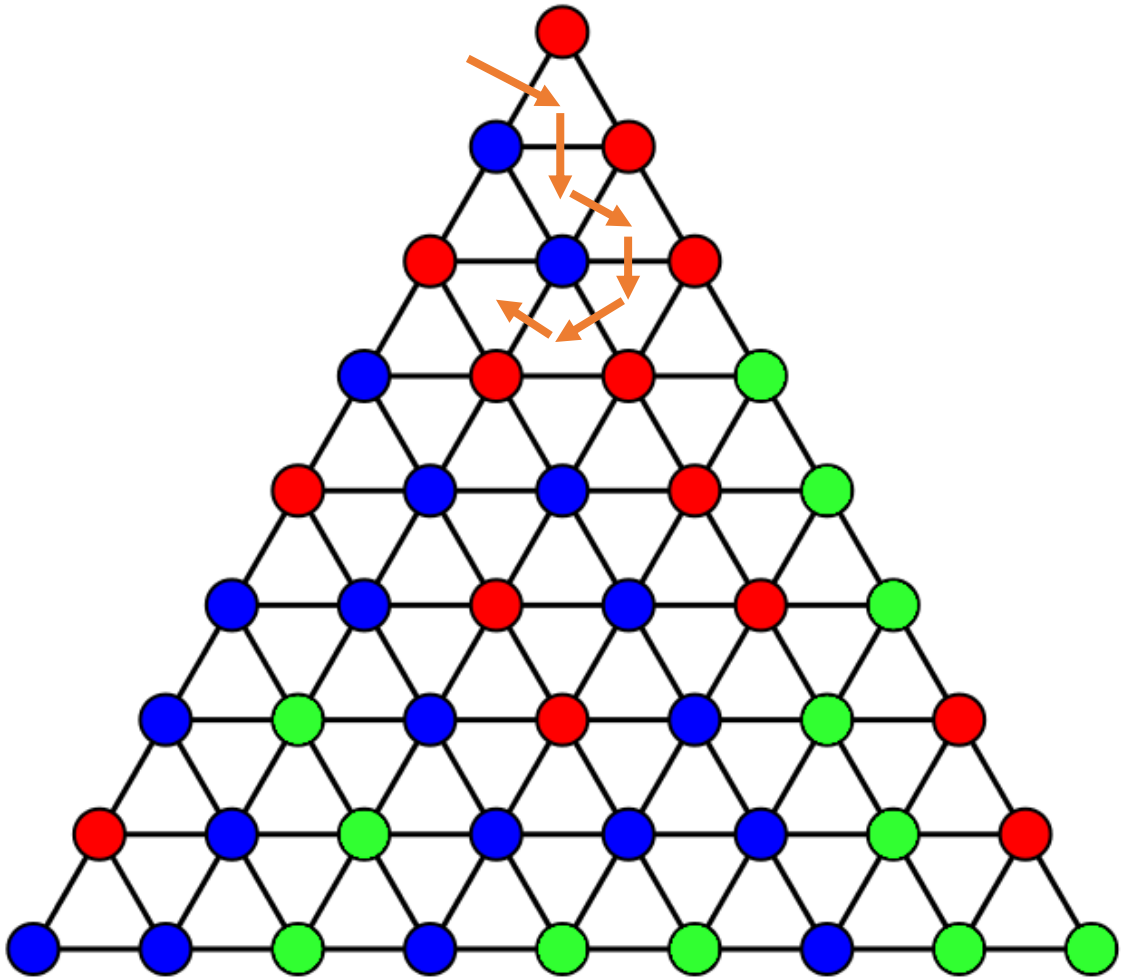
A Proof of Sperner's Lemma



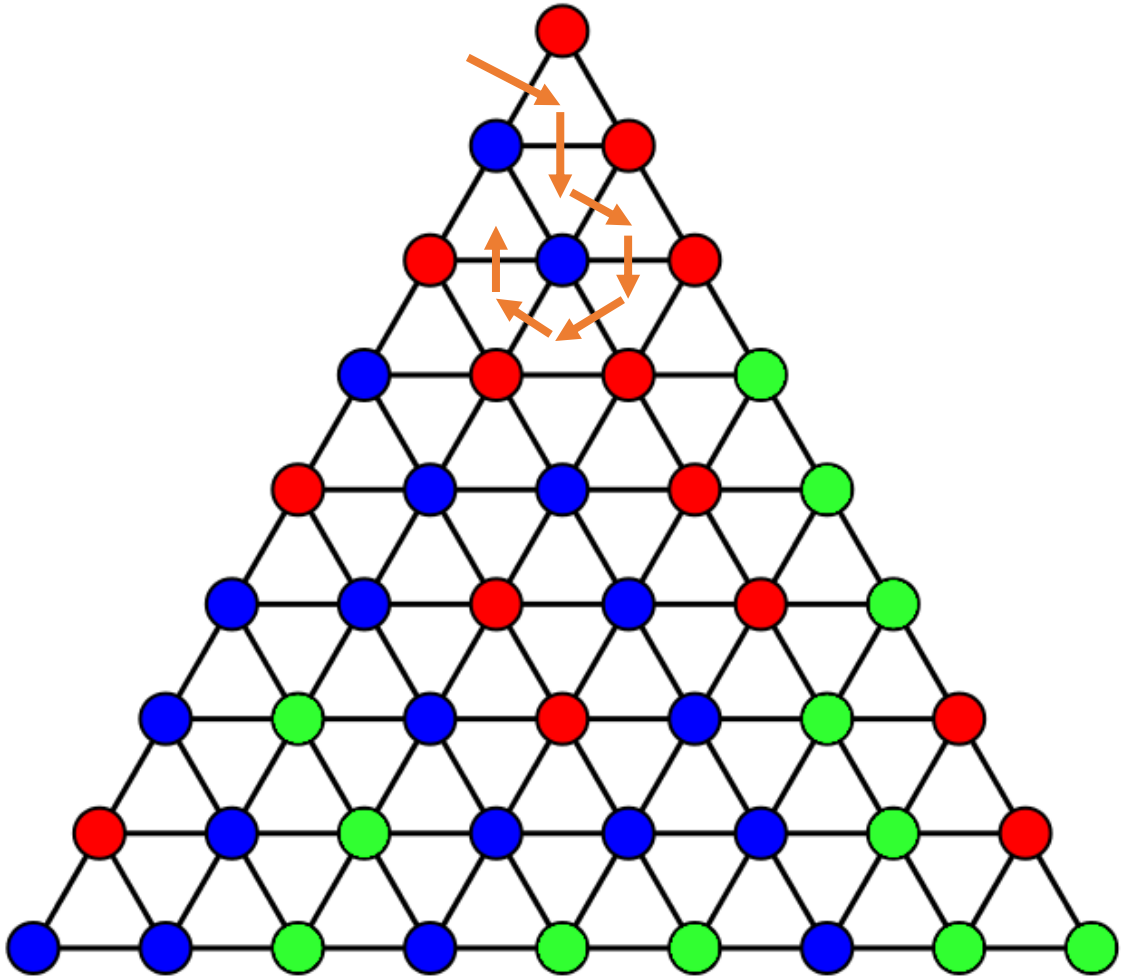
A Proof of Sperner's Lemma



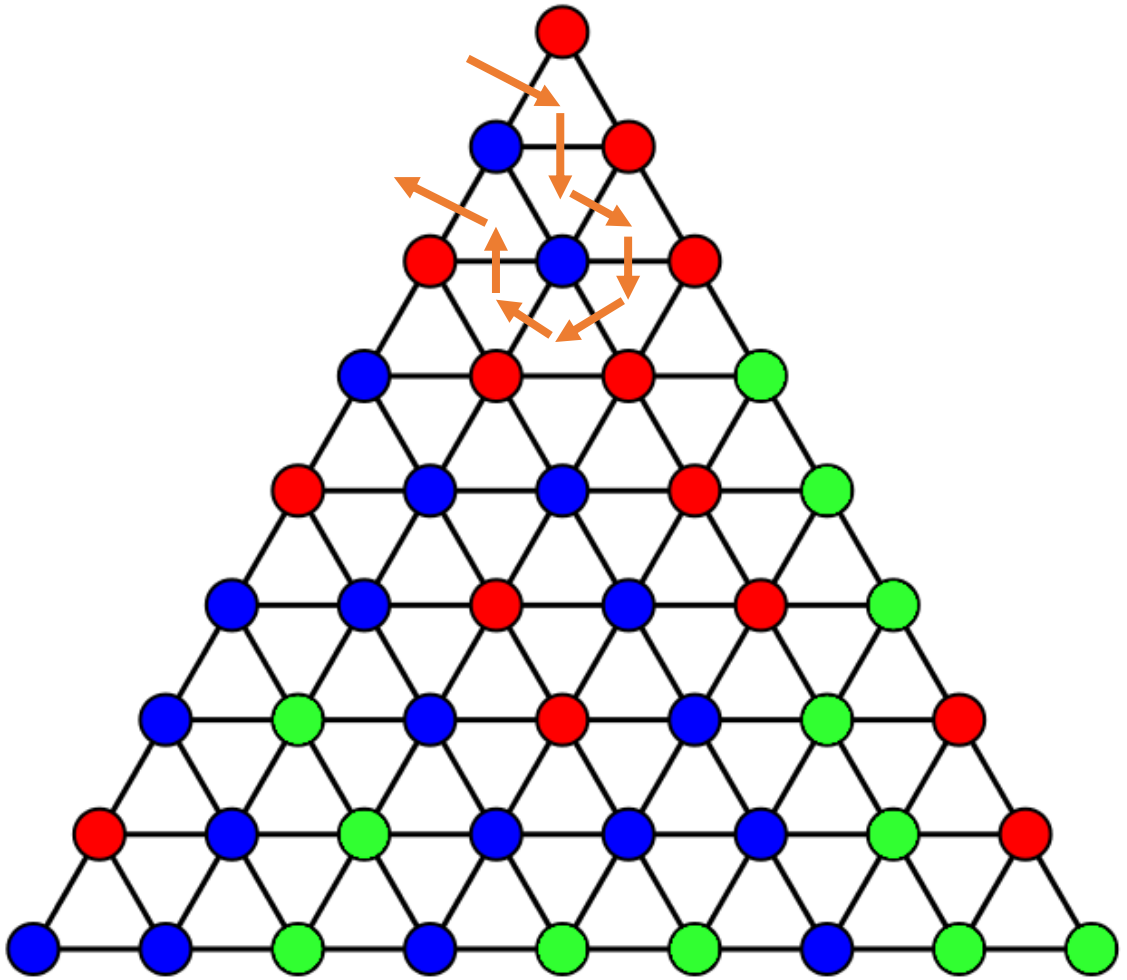
A Proof of Sperner's Lemma



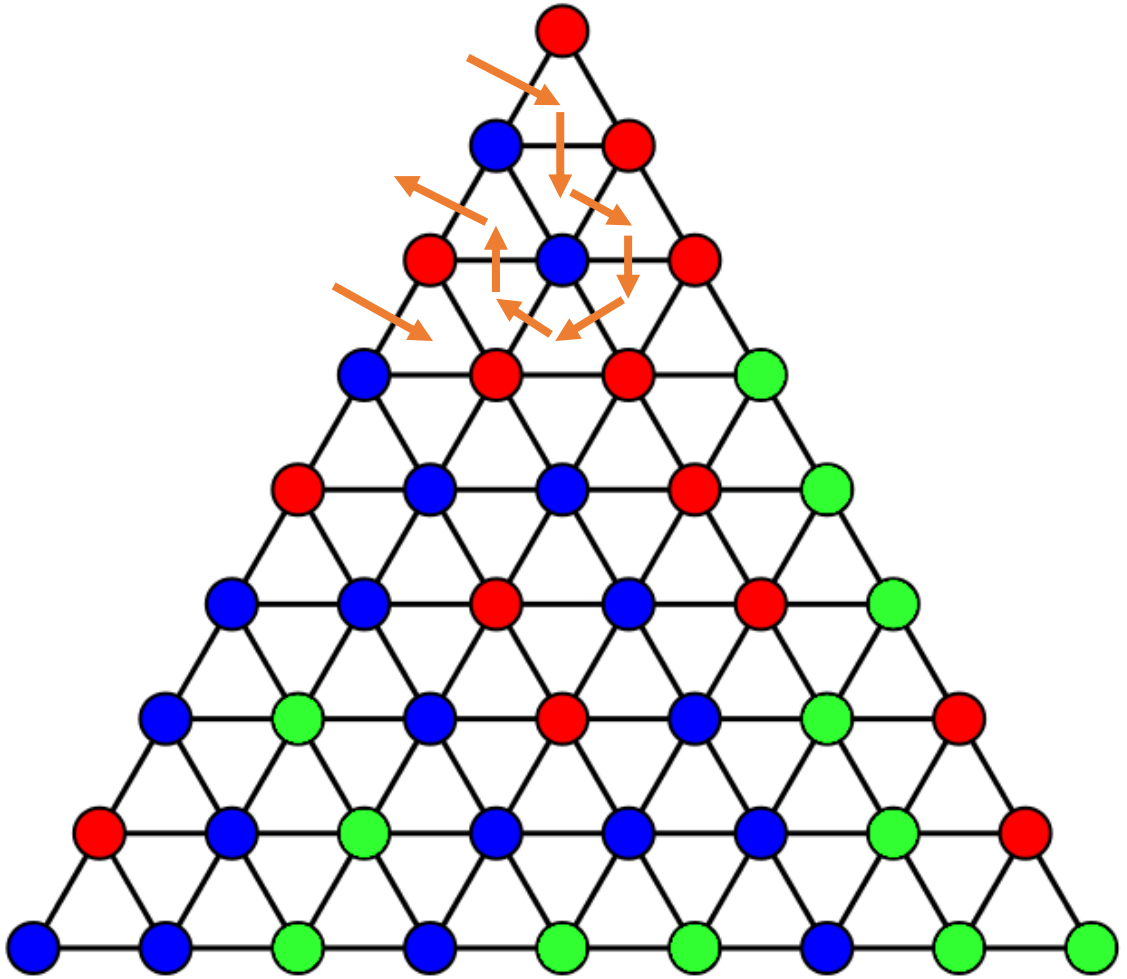
A Proof of Sperner's Lemma



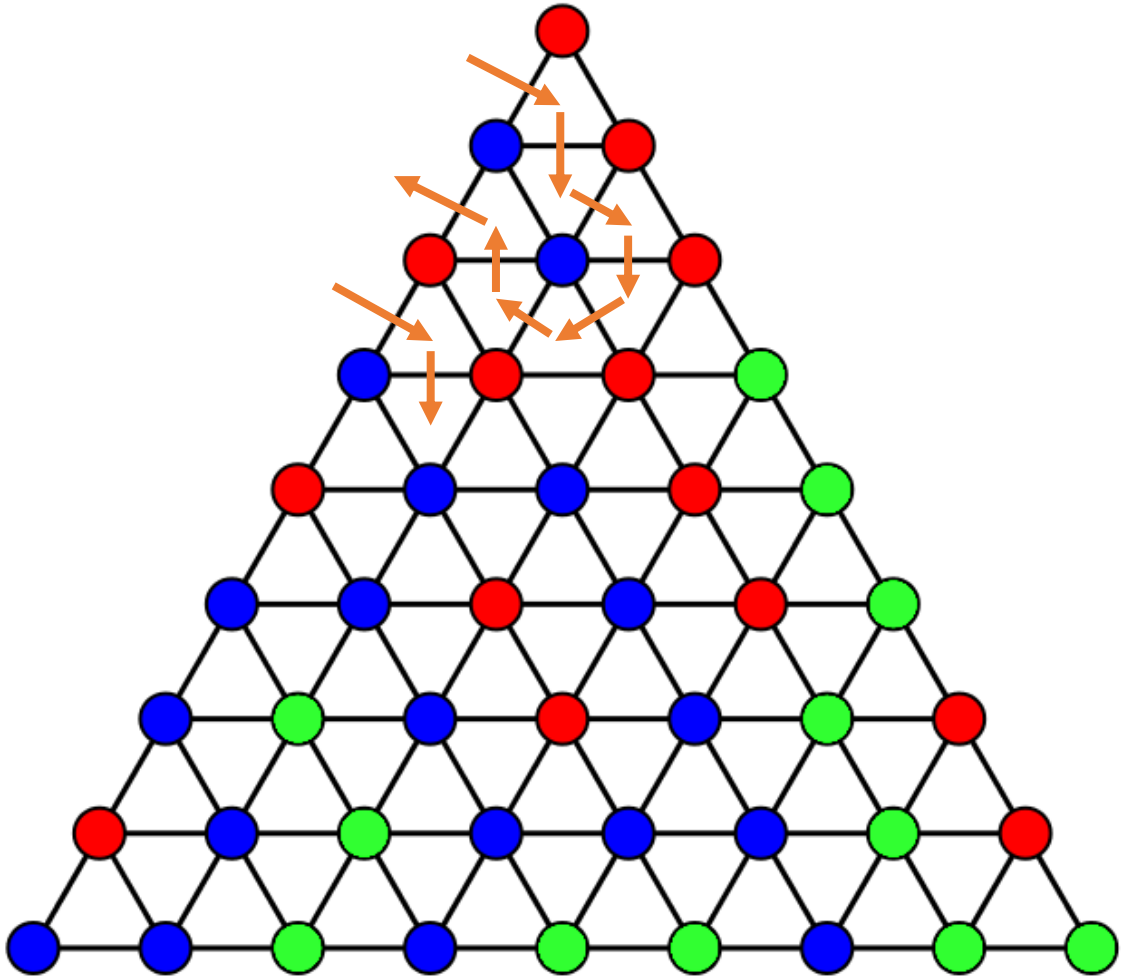
A Proof of Sperner's Lemma



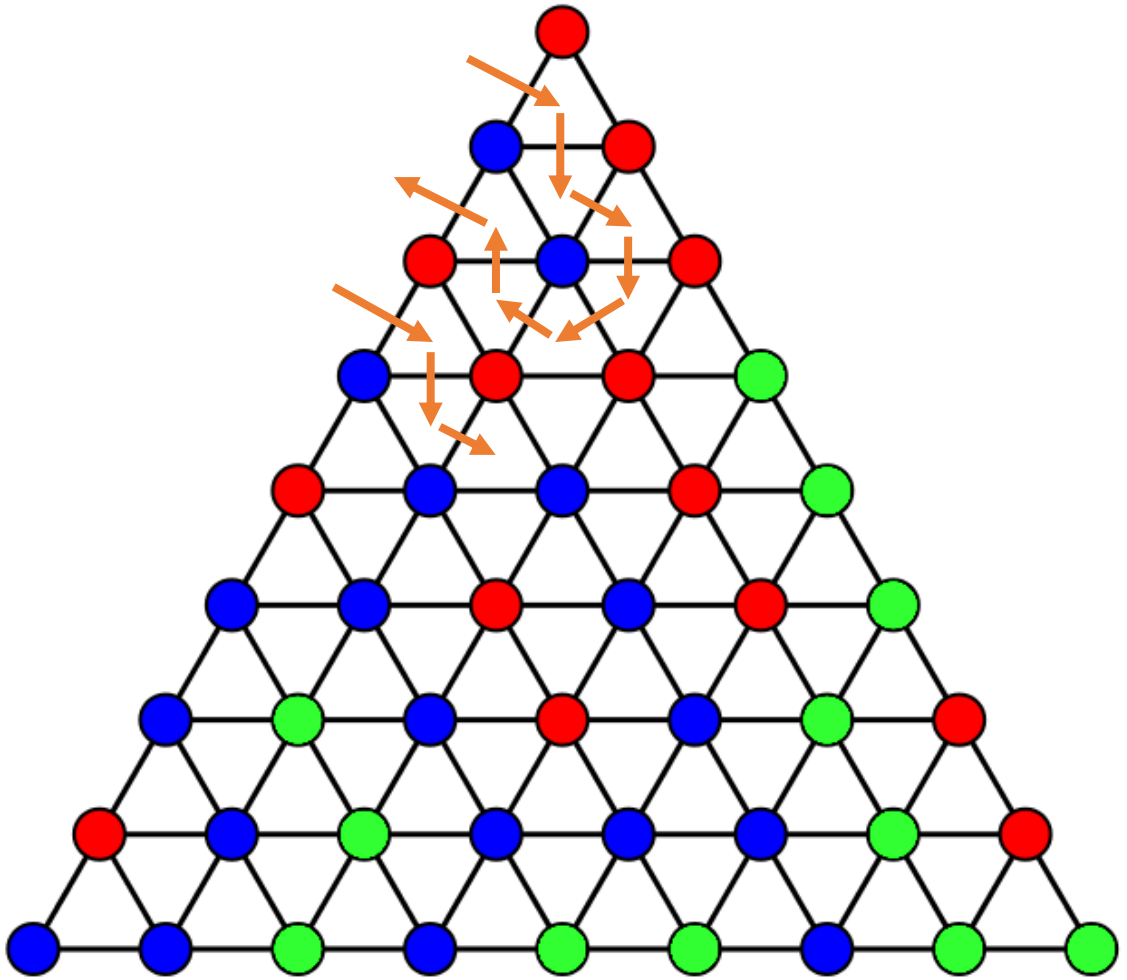
A Proof of Sperner's Lemma



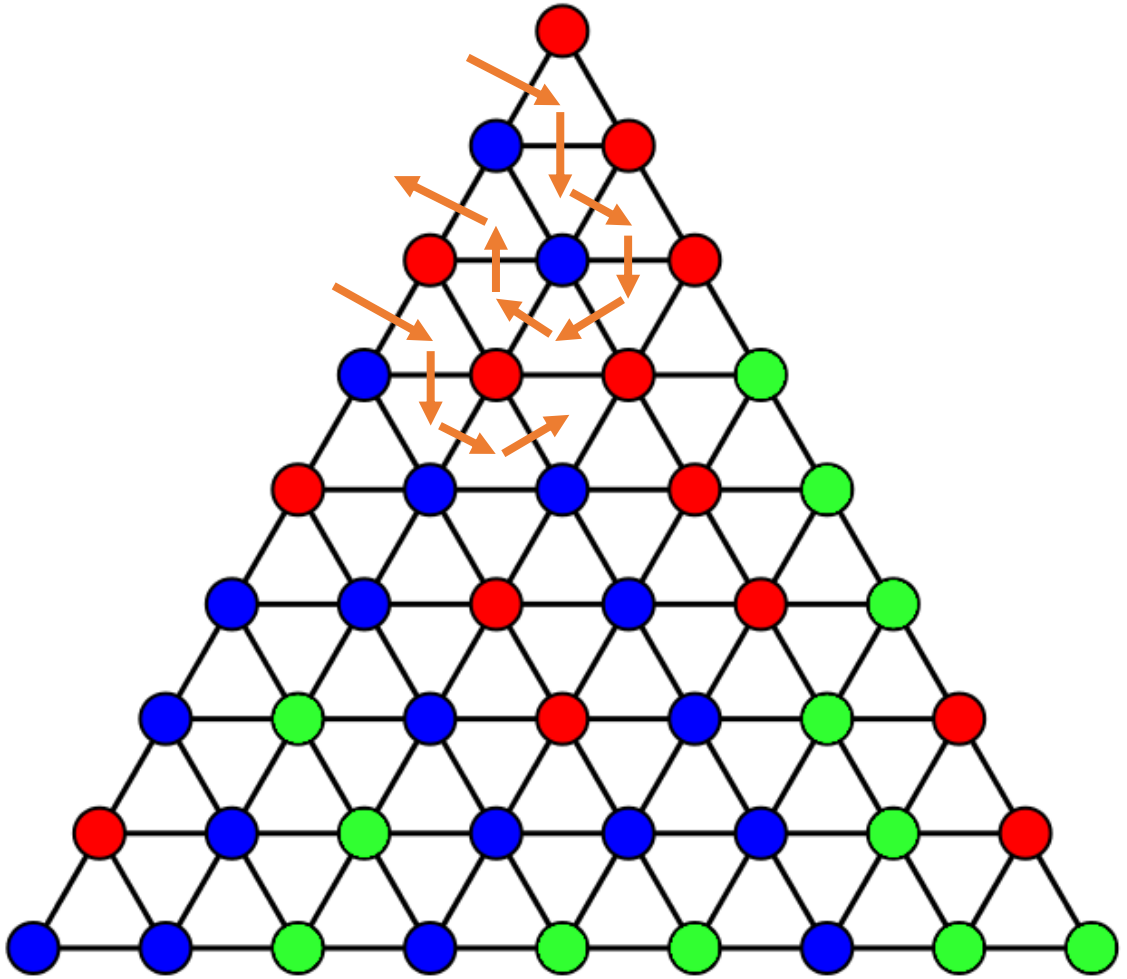
A Proof of Sperner's Lemma



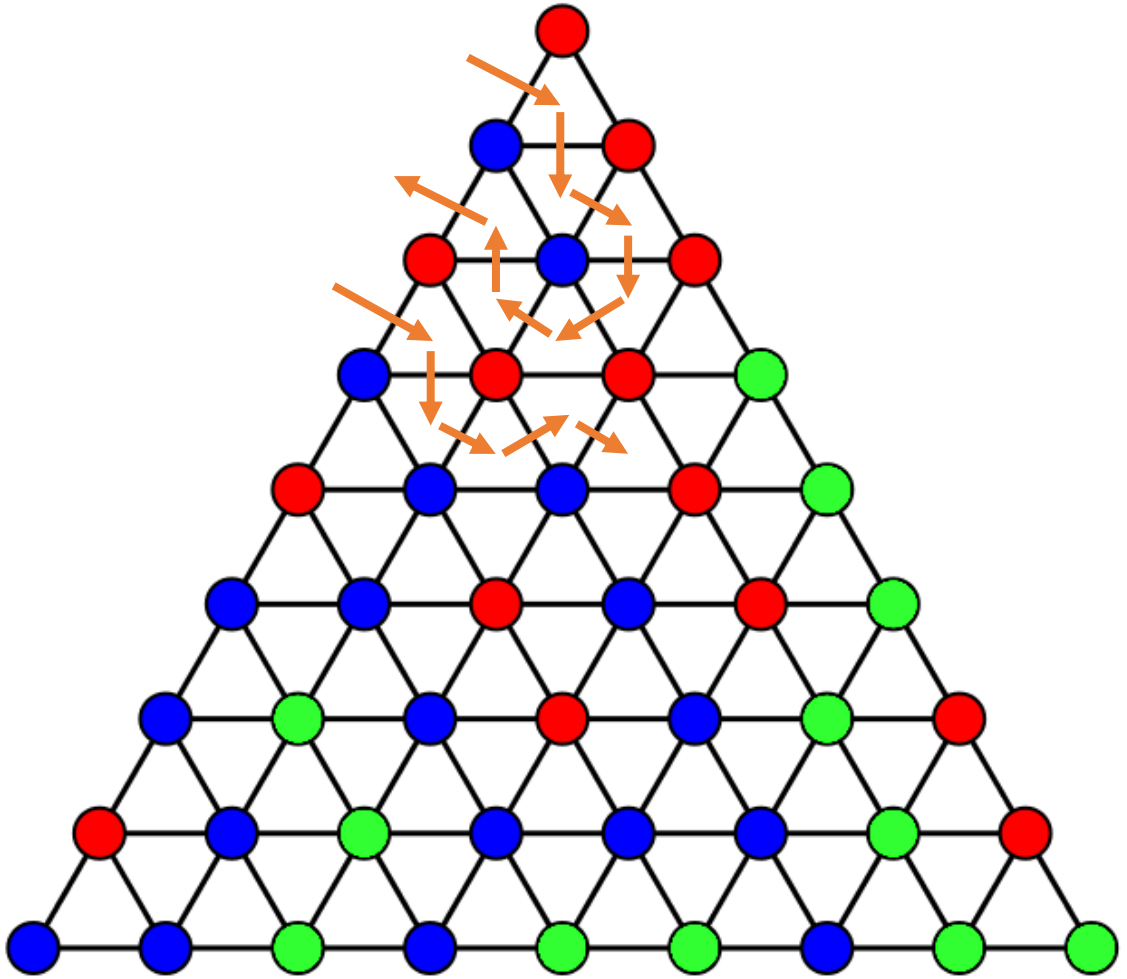
A Proof of Sperner's Lemma



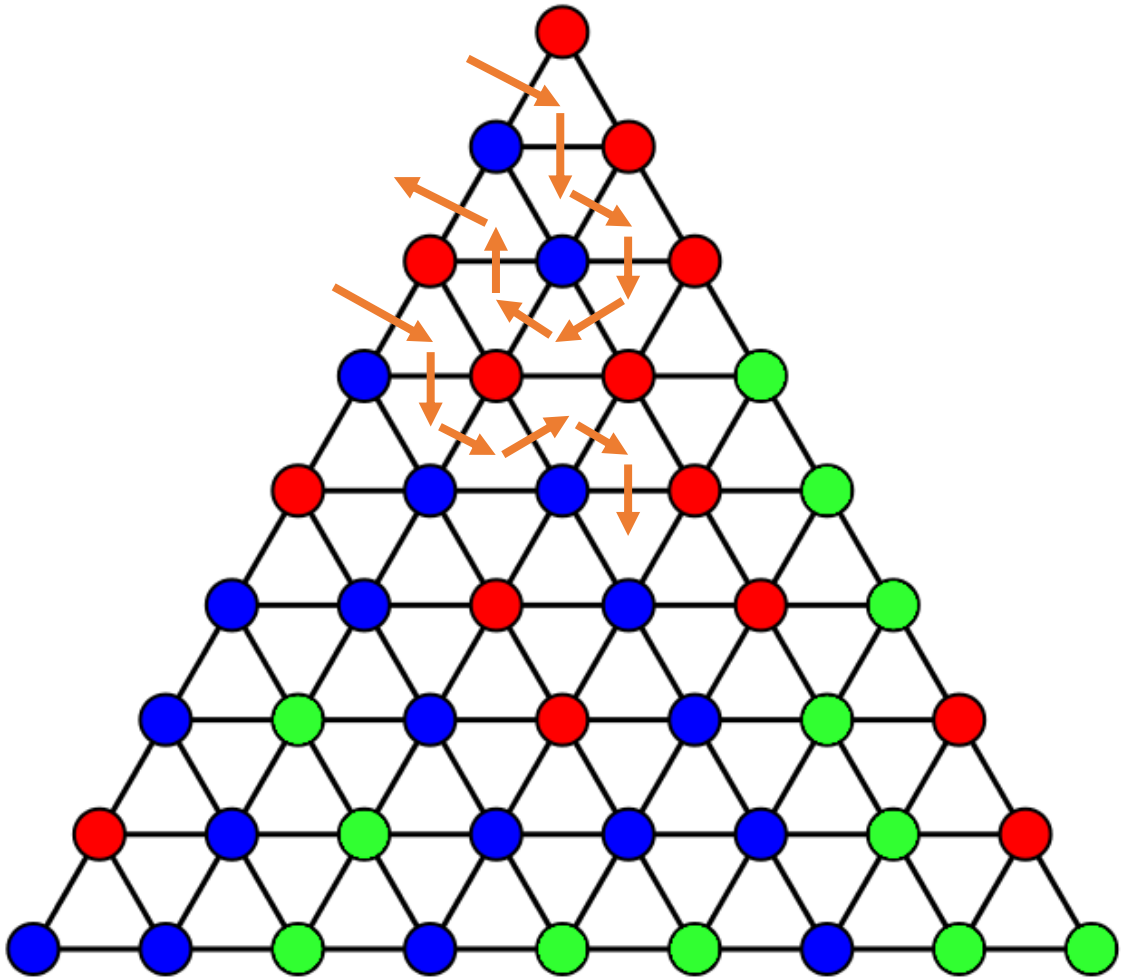
A Proof of Sperner's Lemma



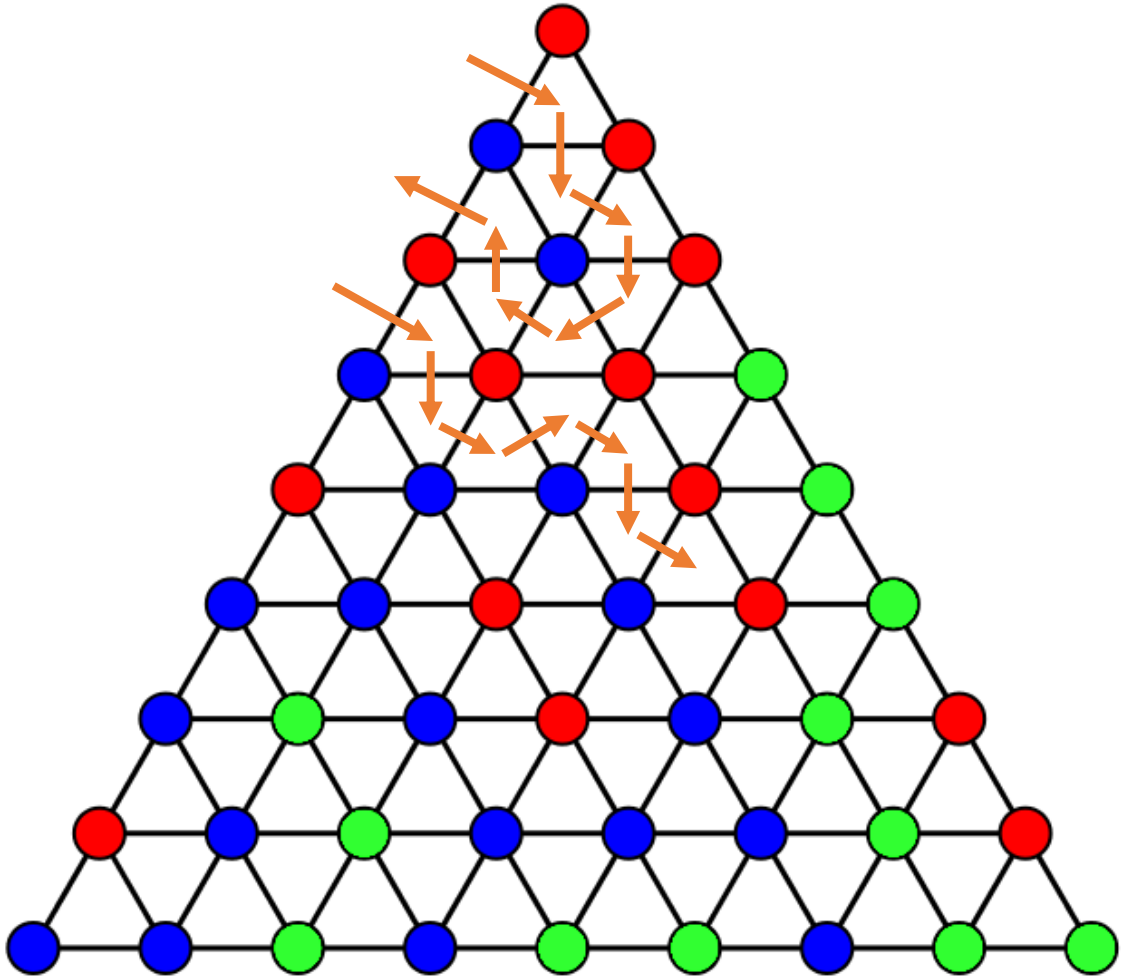
A Proof of Sperner's Lemma



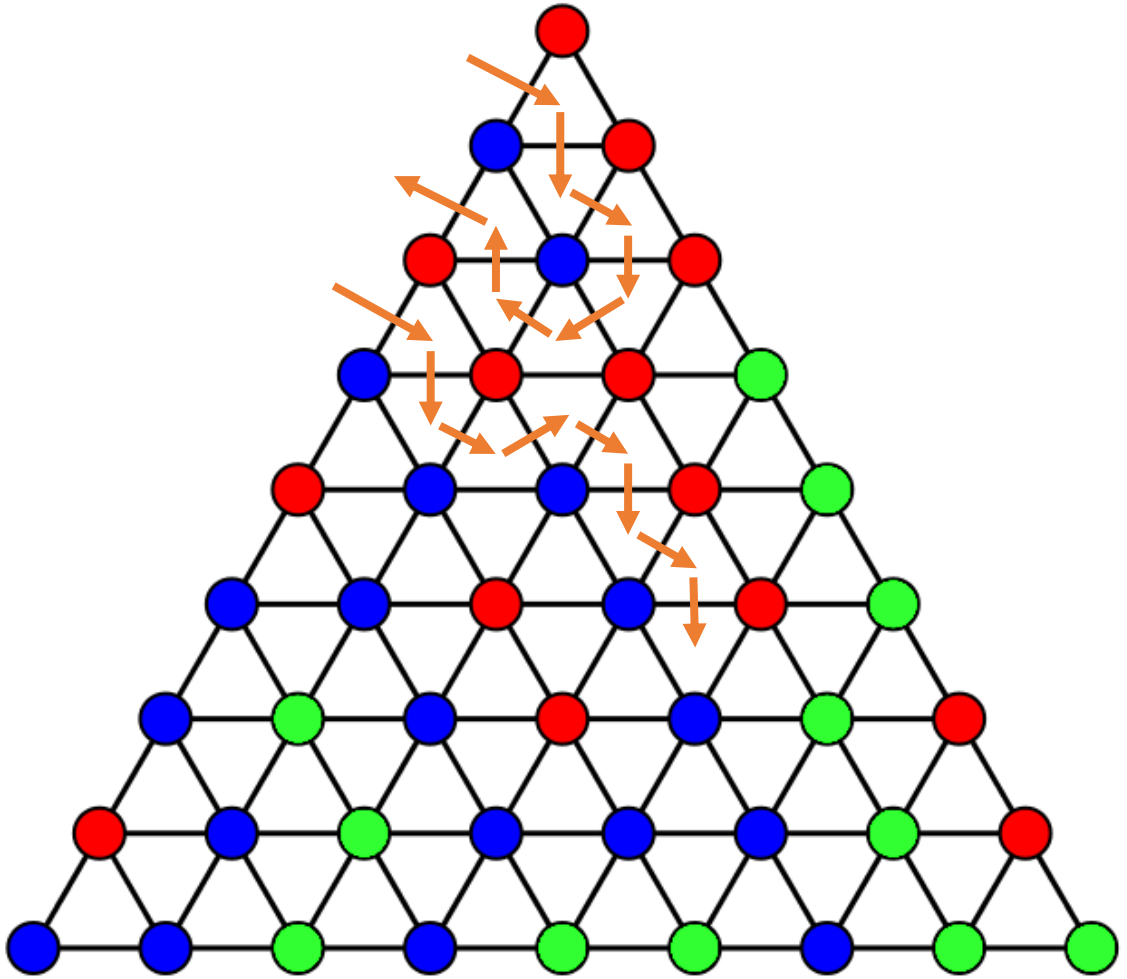
A Proof of Sperner's Lemma



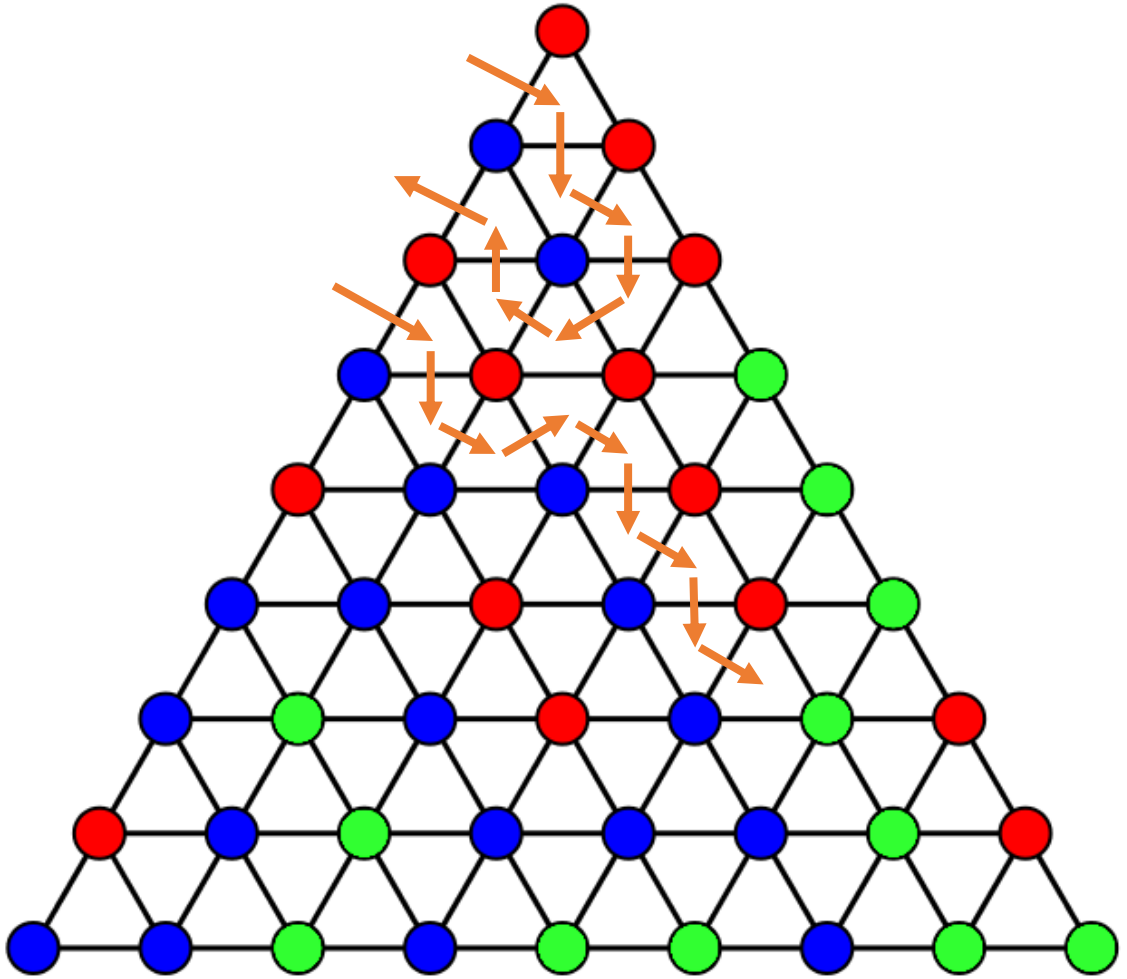
A Proof of Sperner's Lemma



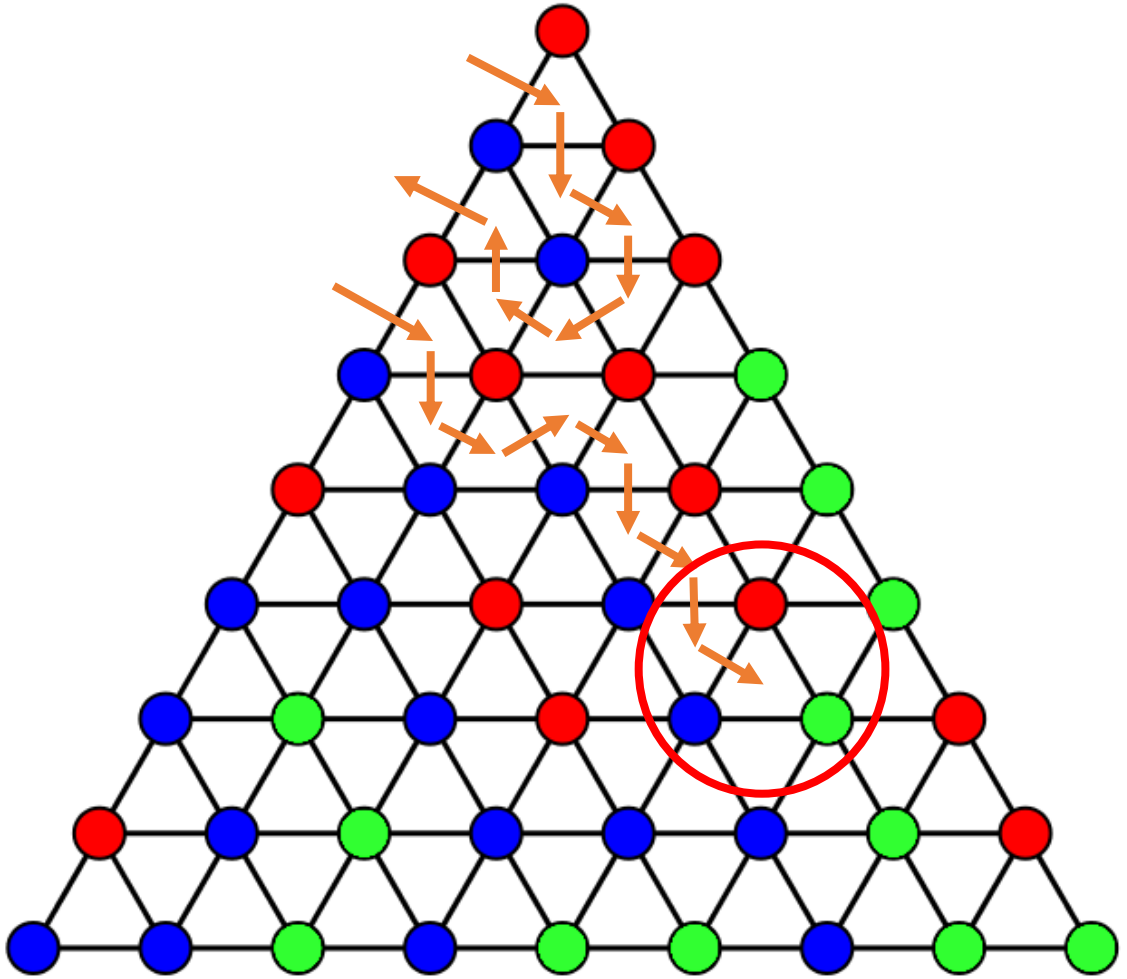
A Proof of Sperner's Lemma



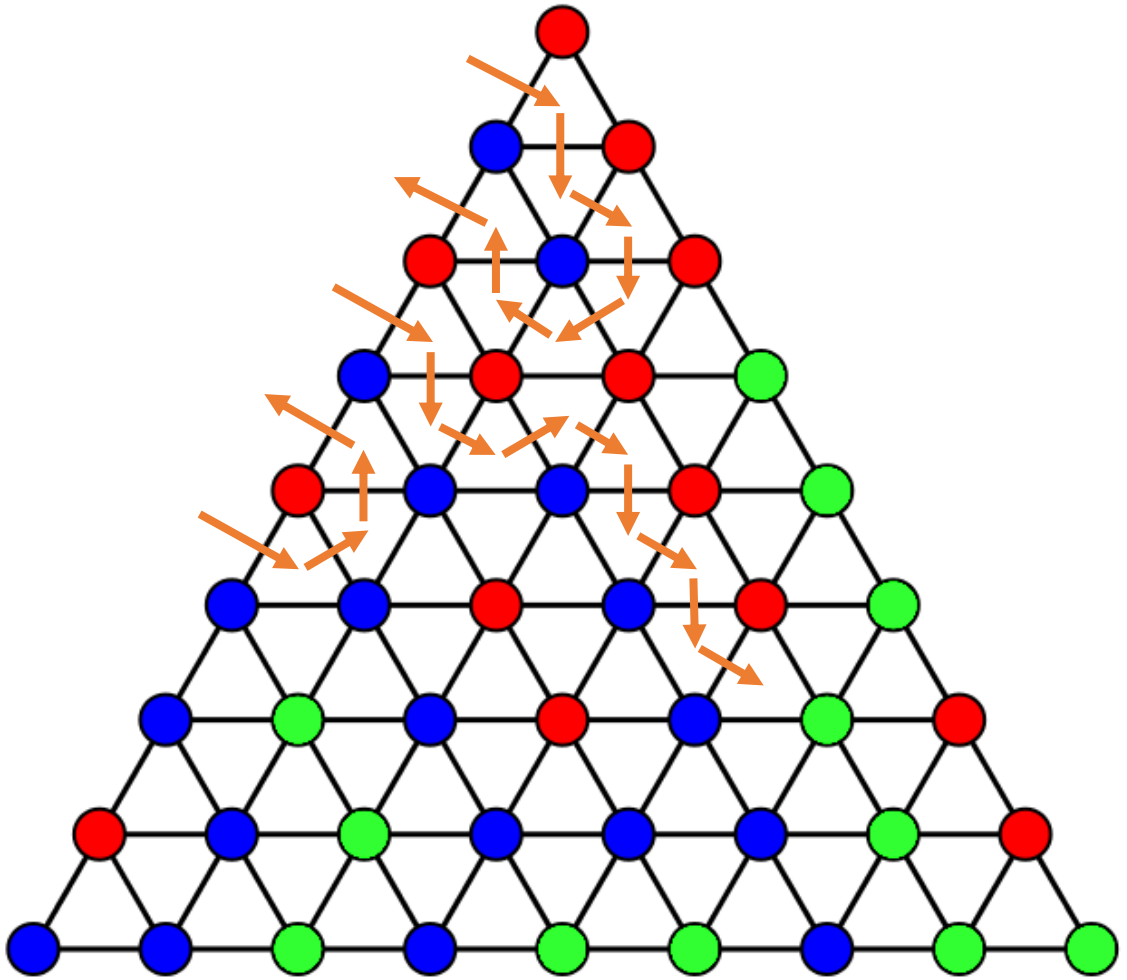
A Proof of Sperner's Lemma



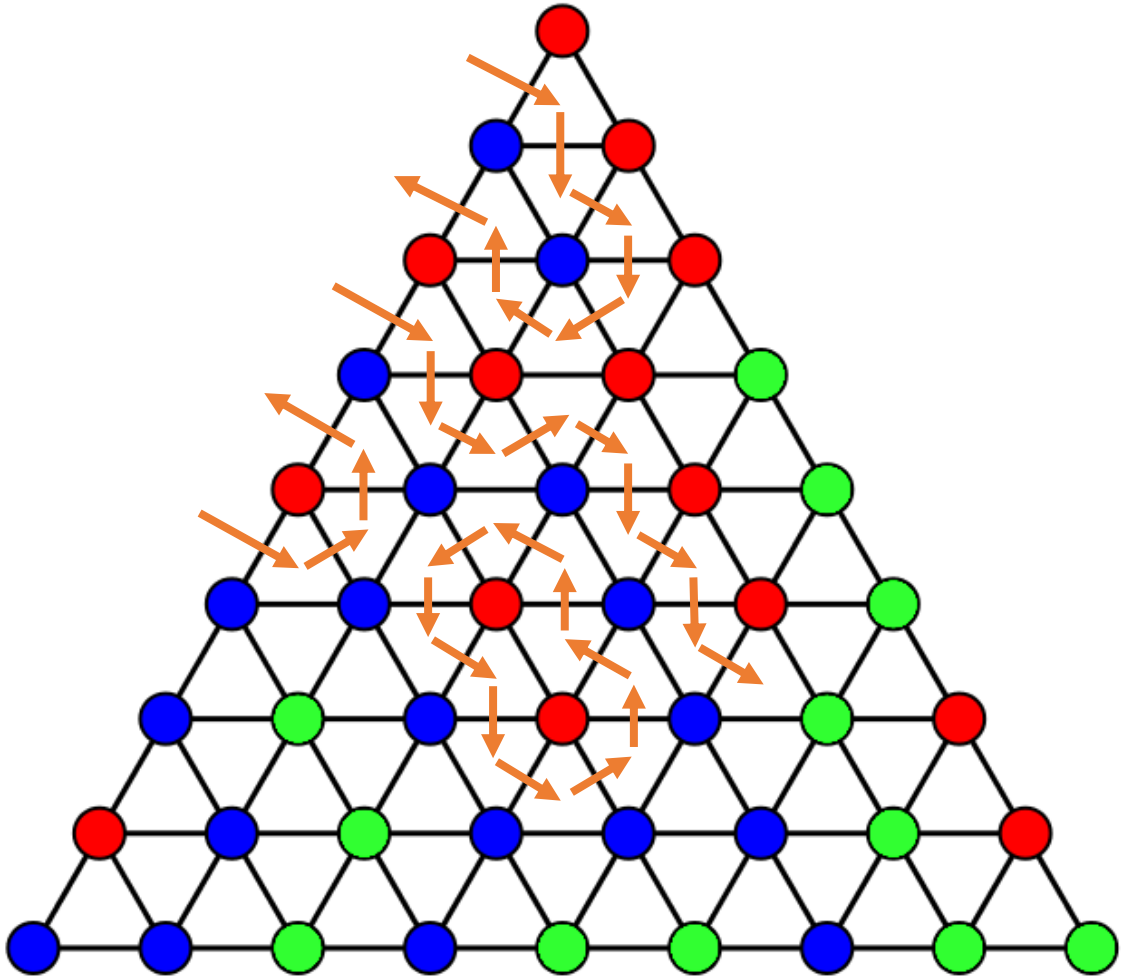
A Proof of Sperner's Lemma



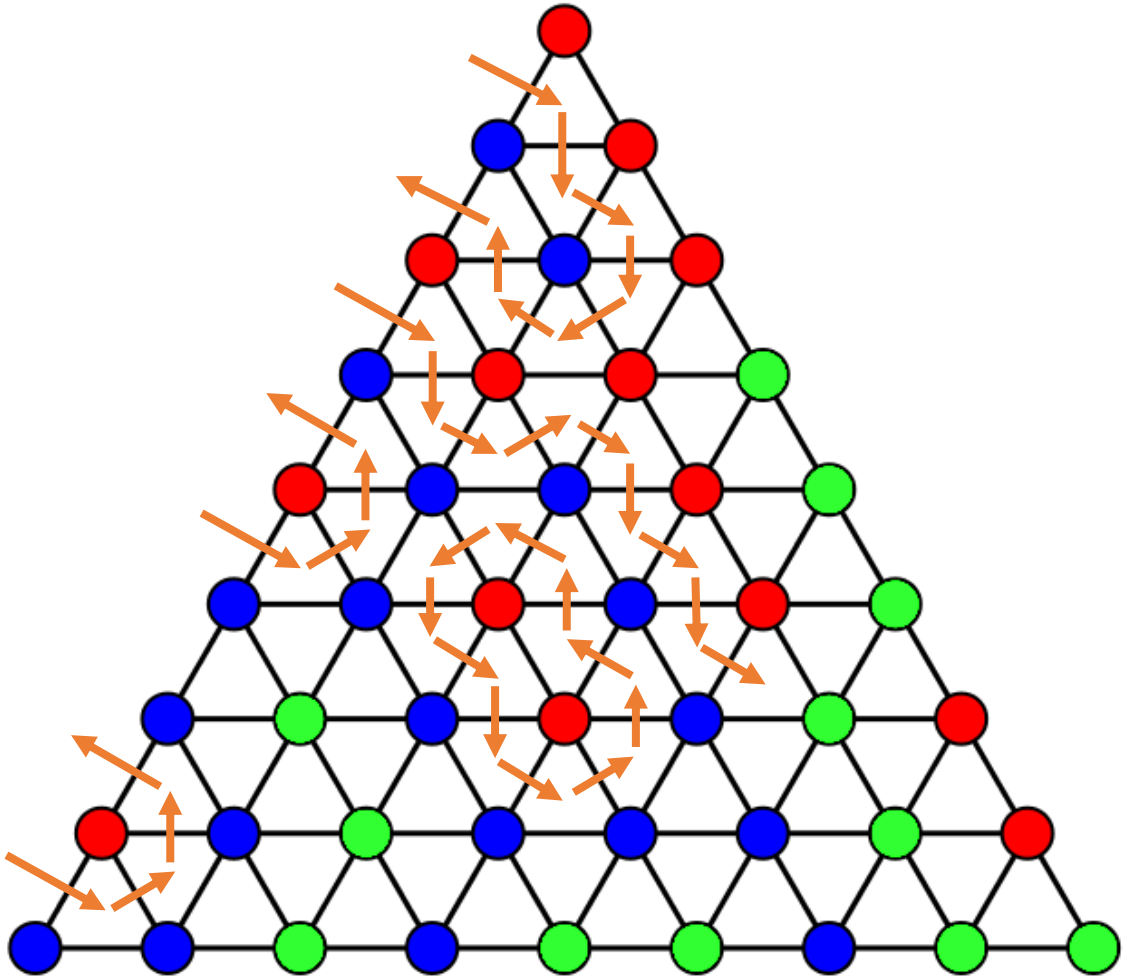
A Proof of Sperner's Lemma



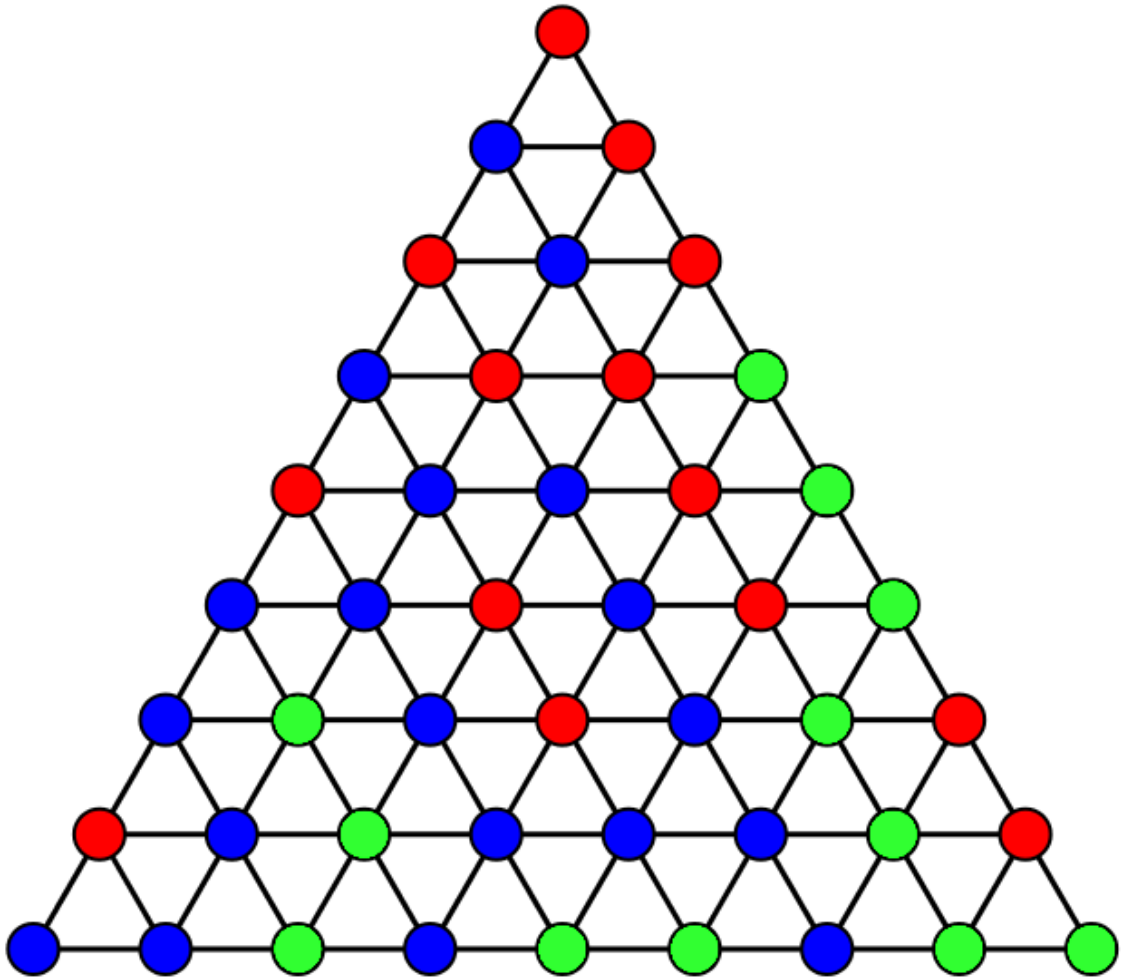
A Proof of Sperner's Lemma



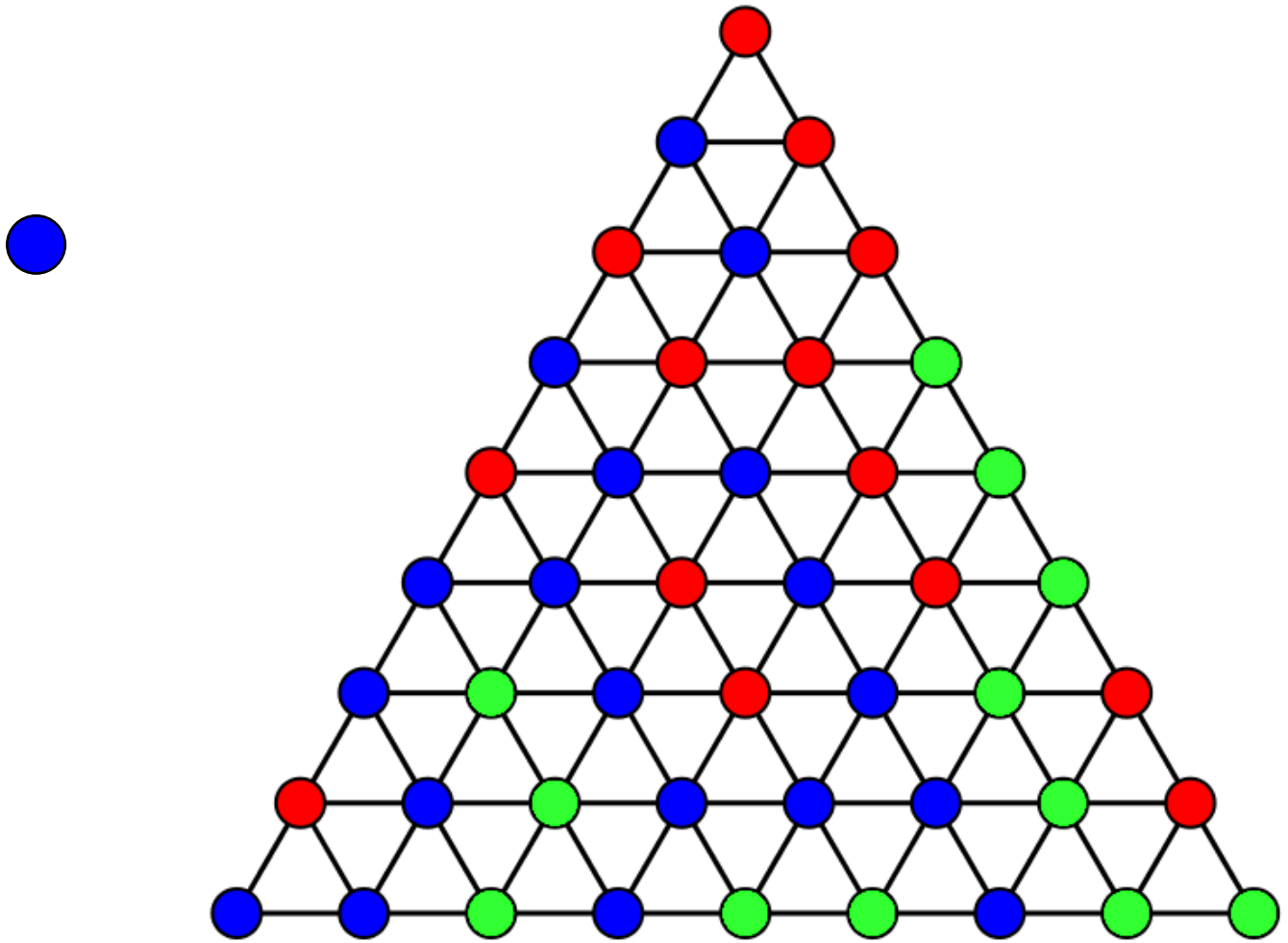
A Proof of Sperner's Lemma



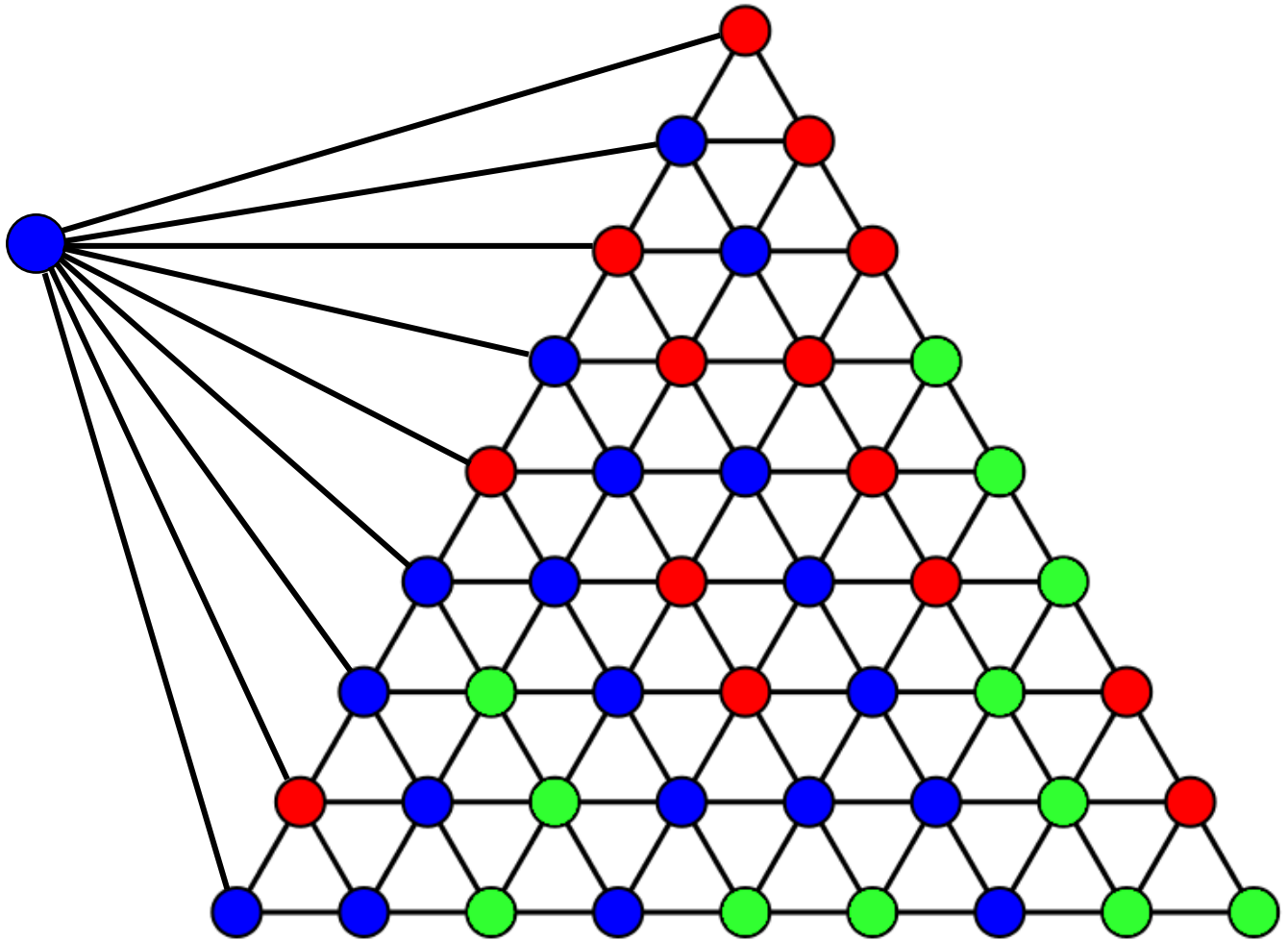
A Proof of Sperner's Lemma



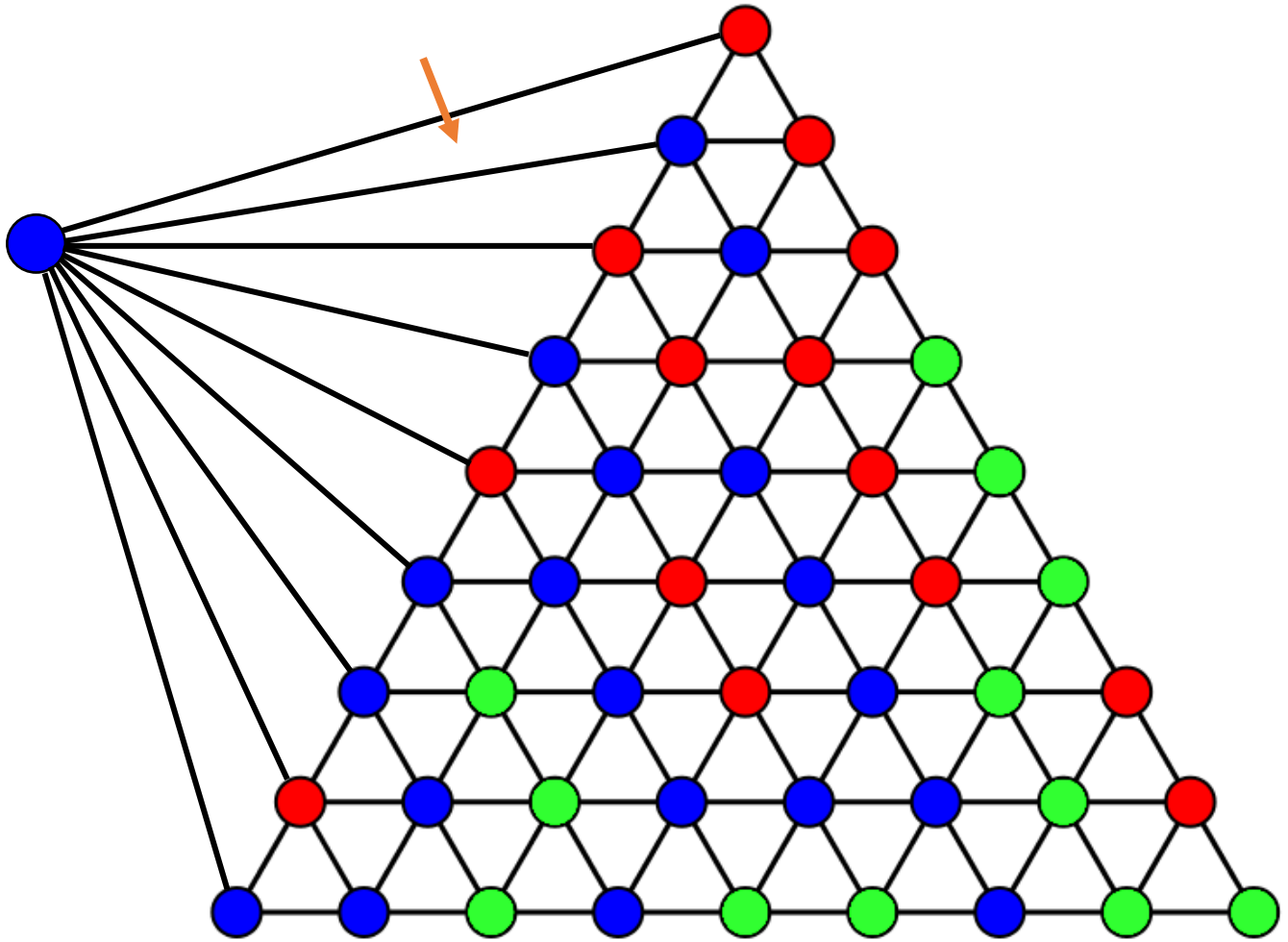
A Proof of Sperner’s Lemma



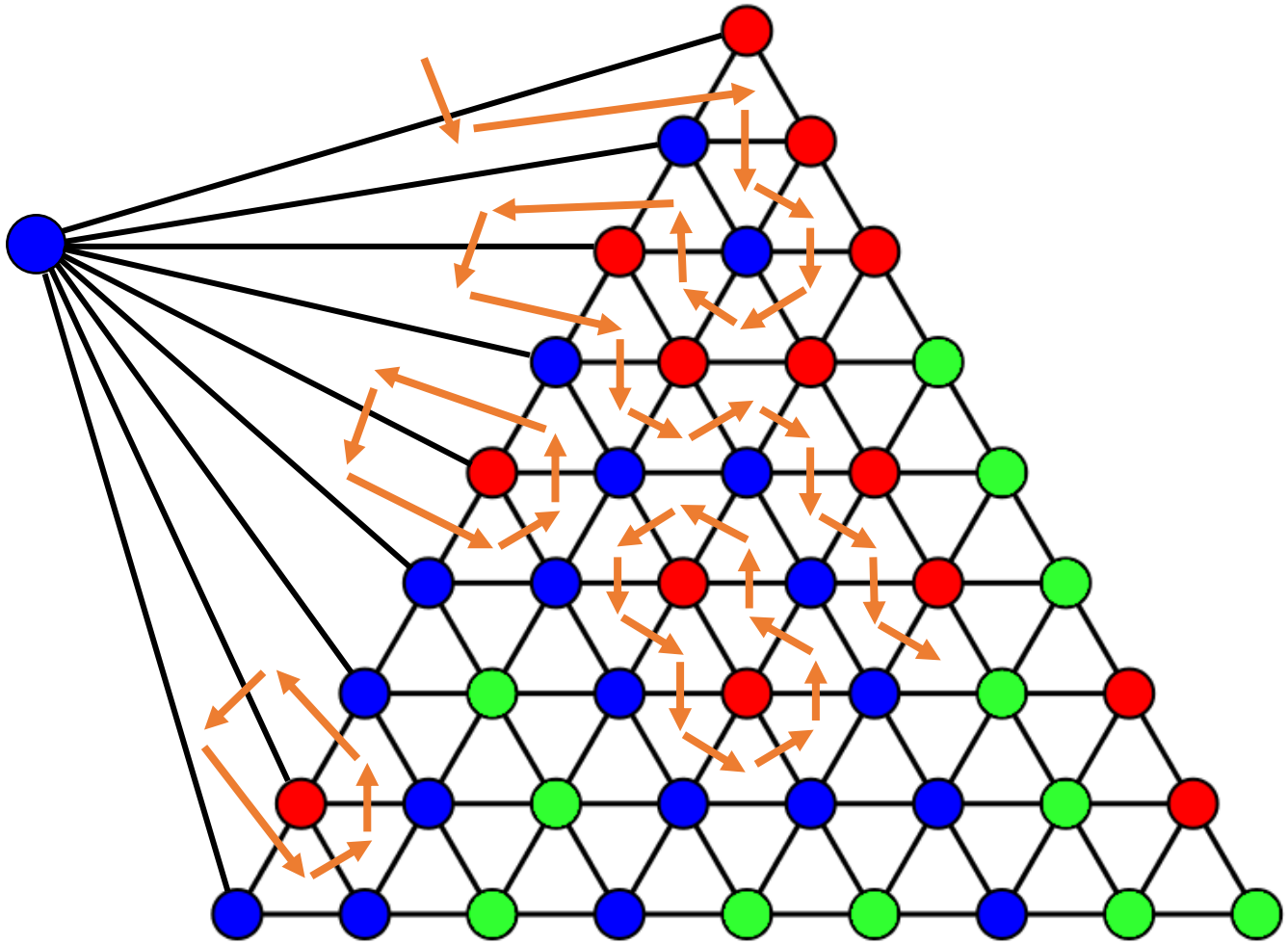
A Proof of Sperner's Lemma



A Proof of Sperner's Lemma



A Proof of Sperner's Lemma



Focus on PPAD

I. How do we prove PPAD-membership?
   1. Sperner's Lemma
   2. **The END-OF-LINE Problem**

II. How do we prove PPAD-hardness?

Formal Definition of PPAD

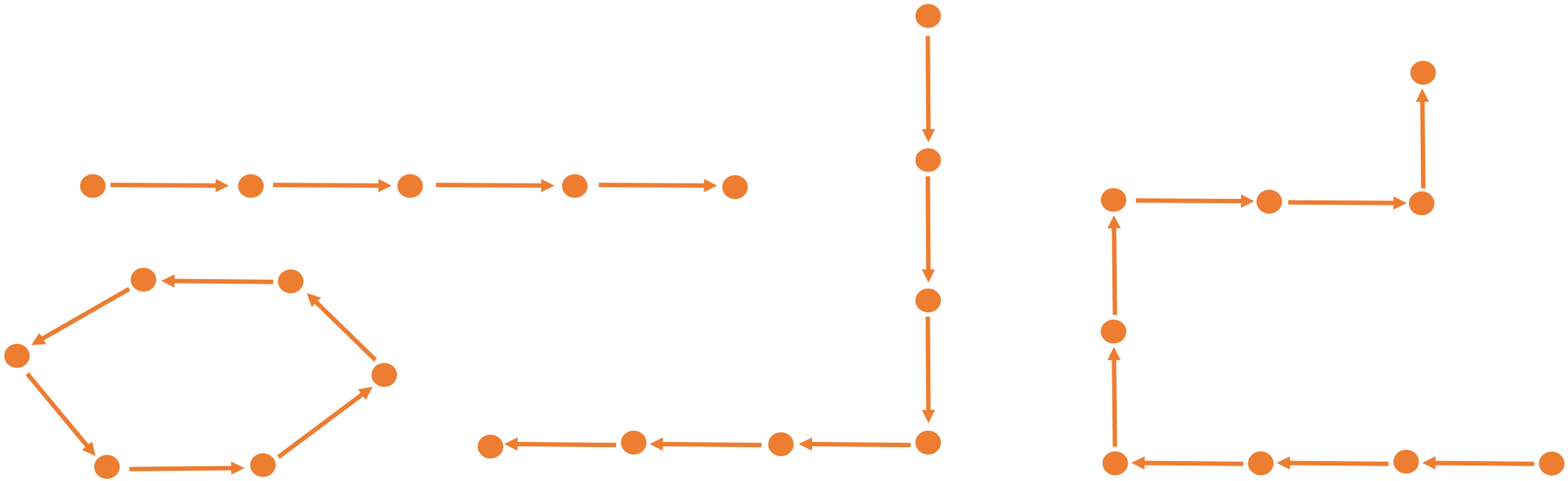
- Canonical complete problem: **END-OF-LINE**

Formal Definition of PPAD

- Canonical complete problem: **END-OF-LINE**
- Input: directed graph of paths and cycles, and a source

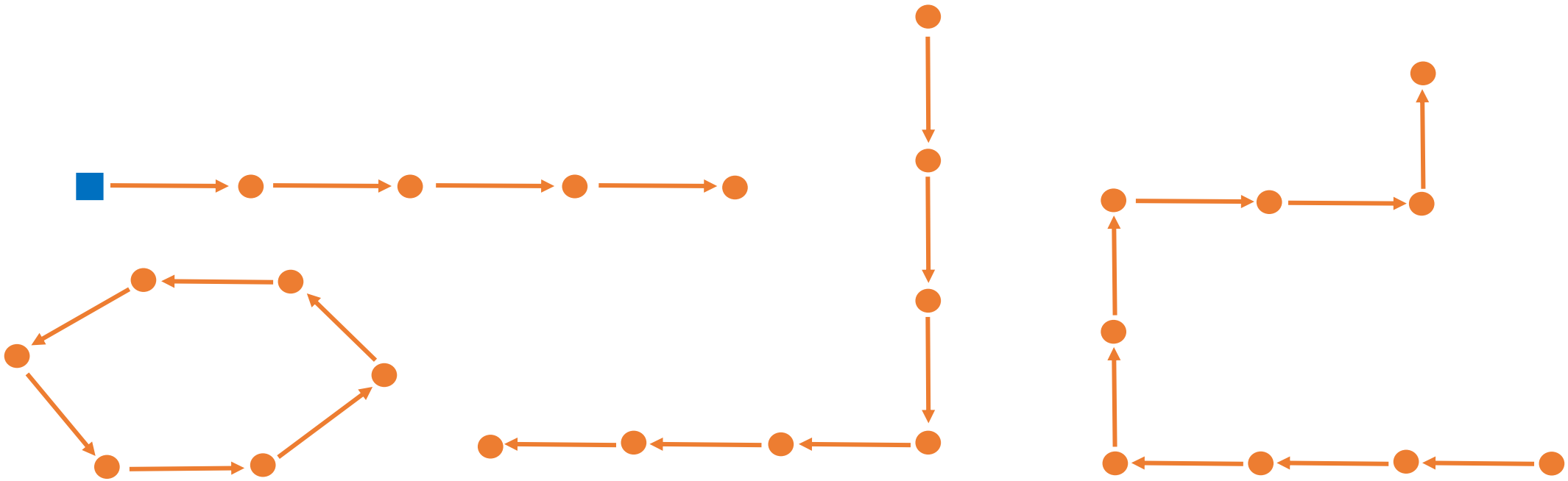
Formal Definition of PPAD

- Canonical complete problem: **END-OF-LINE**
- Input: directed graph of paths and cycles, and a source



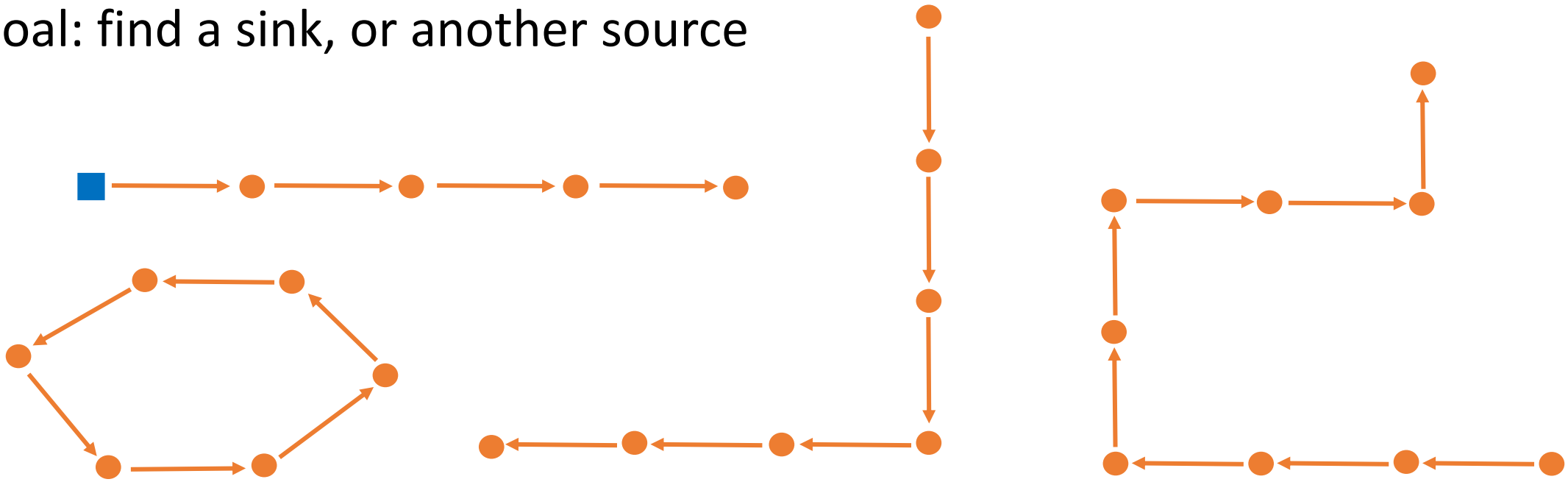
Formal Definition of PPAD

- Canonical complete problem: **END-OF-LINE**
- Input: directed graph of paths and cycles, and a source



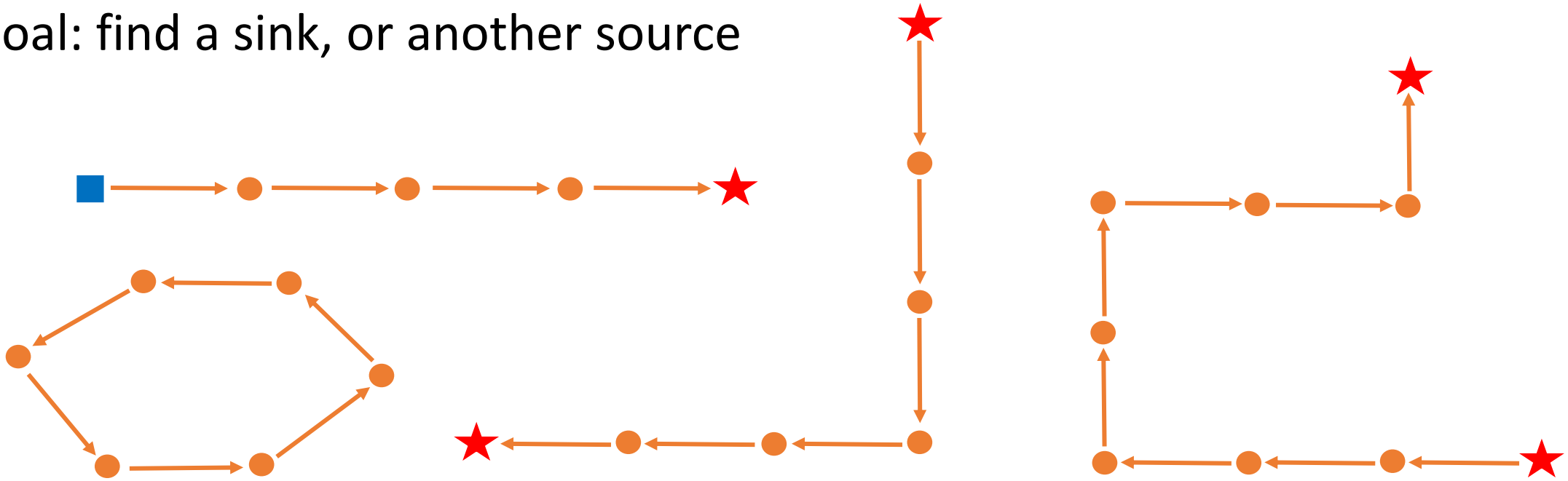
Formal Definition of PPAD

- Canonical complete problem: **END-OF-LINE**
- Input: directed graph of paths and cycles, and a source
- Goal: find a sink, or another source



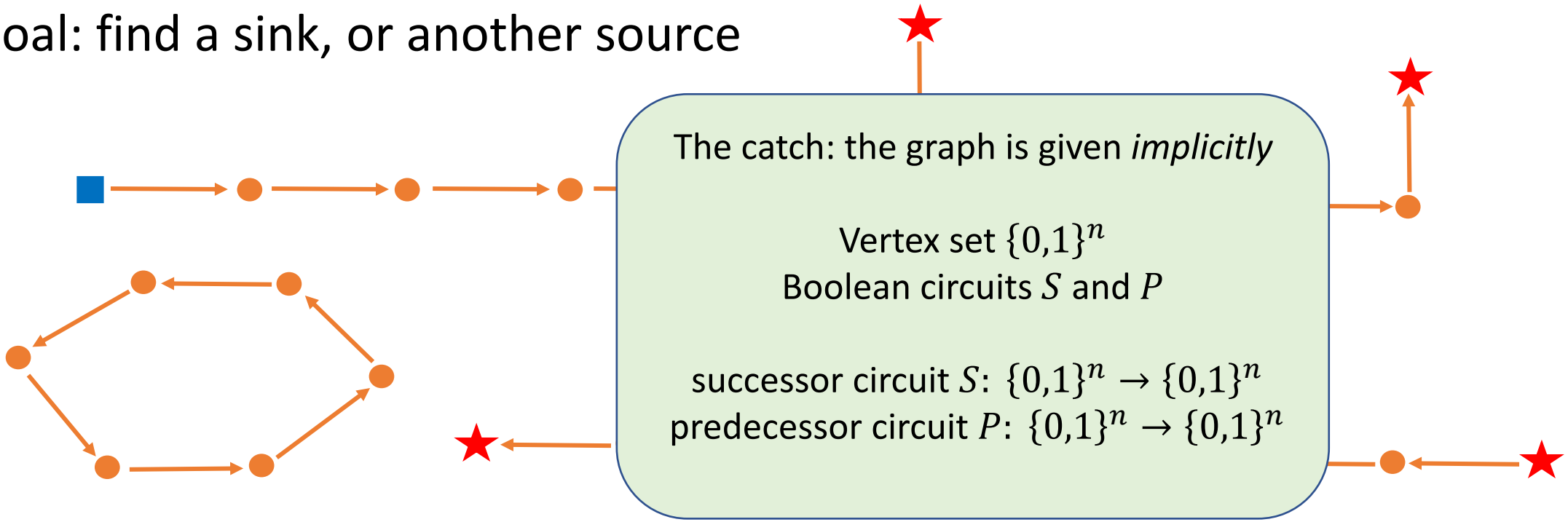
Formal Definition of PPAD

- Canonical complete problem: **END-OF-LINE**
- Input: directed graph of paths and cycles, and a source
- Goal: find a sink, or another source

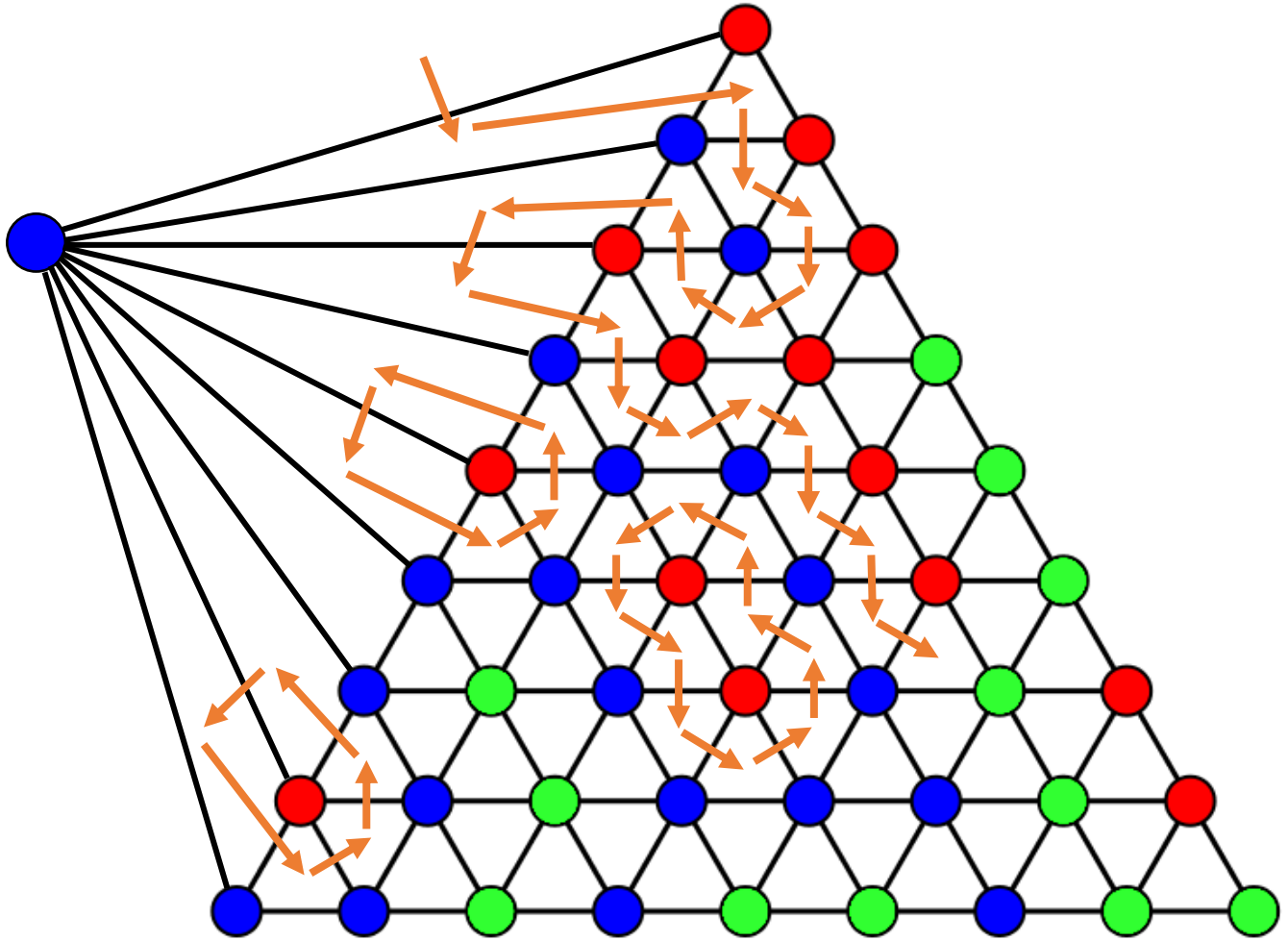


Formal Definition of PPAD

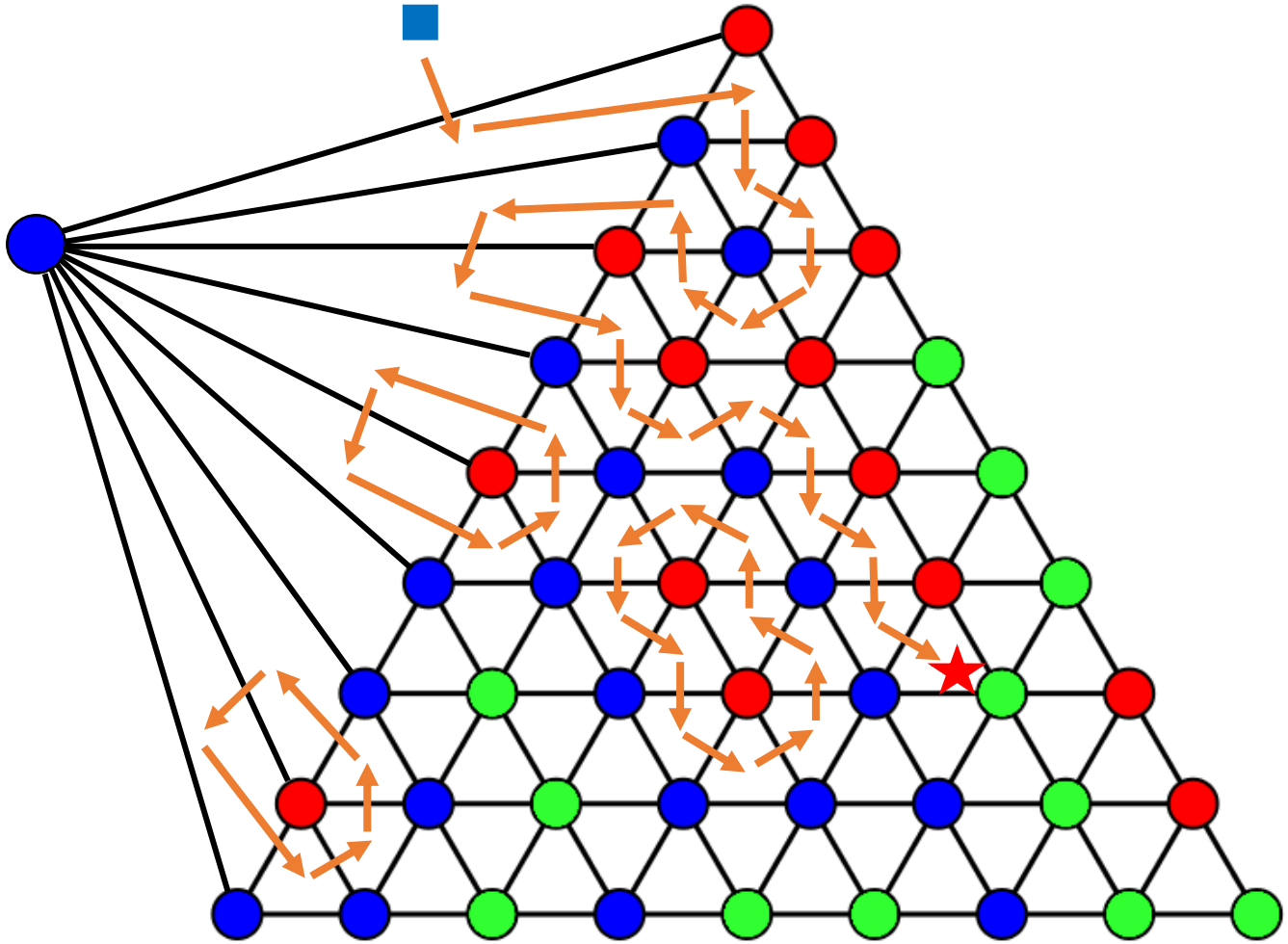
- Canonical complete problem: **END-OF-LINE**
- Input: directed graph of paths and cycles, and a source
- Goal: find a sink, or another source



SPERNER lies in PPAD



SPERNER lies in PPAD



Focus on PPAD

I. How do we prove PPAD-membership?
   1. Sperner's Lemma
   2. The END-OF-LINE Problem
   3. **Example: NASH**

II. How do we prove PPAD-hardness?

The NASH Problem

A set of players with actions (pure strategies) $a_i, i = 1, \dots, n$. Given a vector of strategies $a = (a_1, a_2, \dots, a_n)$ (strategy profile), every agent receives a payoff $u_i(a)$.

Agents choose mixed strategies, i.e., probability mixtures over actions.

The NASH Problem

A set of players with actions (pure strategies) $a_i, i = 1, \dots, n$. Given a vector of strategies $a = (a_1, a_2, \dots, a_n)$ (strategy profile), every agent receives a payoff $u_i(a)$.

Agents choose mixed strategies, i.e., probability mixtures over actions.

Nash equilibrium: Given the strategies of the others, every agent chooses a strategy that maximizes their payoff.

The NASH Problem

A set of players with actions (pure strategies) $a_i, i = 1, \dots, n$. Given a vector of strategies $a = (a_1, a_2, \dots, a_n)$ (strategy profile), every agent receives a payoff $u_i(a)$.

Agents choose mixed strategies, i.e., probability mixtures over actions.

Nash equilibrium: Given the strategies of the others, every agent chooses a strategy that maximizes their payoff.

Theorem (Nash 1951): Every finite strategic game has a Nash equilibrium (in mixed strategies)

Proof of Nash’s Theorem

Brouwer’s Fixed Point Theorem:

Proof of Nash's Theorem

Brouwer's Fixed Point Theorem:

Let D be convex and compact. Then, every continuous function $F: D \to D$ admits a fixed point, i.e., $x \in D$ such that $F(x) = x$.

Proof of Nash's Theorem

Brouwer's Fixed Point Theorem:

Let D be convex and compact. Then, every continuous function $F: D \to D$ admits a fixed point, i.e., $x \in D$ such that $F(x) = x$.

Proof of Nash's Theorem:

Proof of Nash's Theorem

Brouwer's Fixed Point Theorem:

Let D be convex and compact. Then, every continuous function $F: D \to D$ admits a fixed point, i.e., $x \in D$ such that $F(x) = x$.

Proof of Nash's Theorem:

Let $D = \Delta_{m_1} \times \cdots \times \Delta_{m_n}$. Consider $F: D \to D$ defined by $F_{i,j}(x) := \frac{x_{i,j} + g_{i,j}(x)}{1 + \sum_{k=1}^{m_i} g_{i,k}(x)}$

where $g_{i,j}(x) := \max\{0,\ u_i(a_{i,j}, x_{-i}) - u_i(x)\}$.

Proof of Nash's Theorem

Brouwer's Fixed Point Theorem:

Let D be convex and compact. Then, every continuous function $F: D \to D$ admits a fixed point, i.e., $x \in D$ such that $F(x) = x$.

Proof of Nash's Theorem:

Let $D = \Delta_{m_1} \times \cdots \times \Delta_{m_n}$. Consider $F: D \to D$ defined by $F_{i,j}(x) := \frac{x_{i,j} + g_{i,j}(x)}{1 + \sum_{k=1}^{m_i} g_{i,k}(x)}$

where $g_{i,j}(x) := \max\{0,\ u_i(a_{i,j}, x_{-i}) - u_i(x)\}$.

→ any fixed point x of F is a Nash equilibrium!

PPAD-membership of NASH

Replace Brouwer by Sperner:

PPAD-membership of NASH

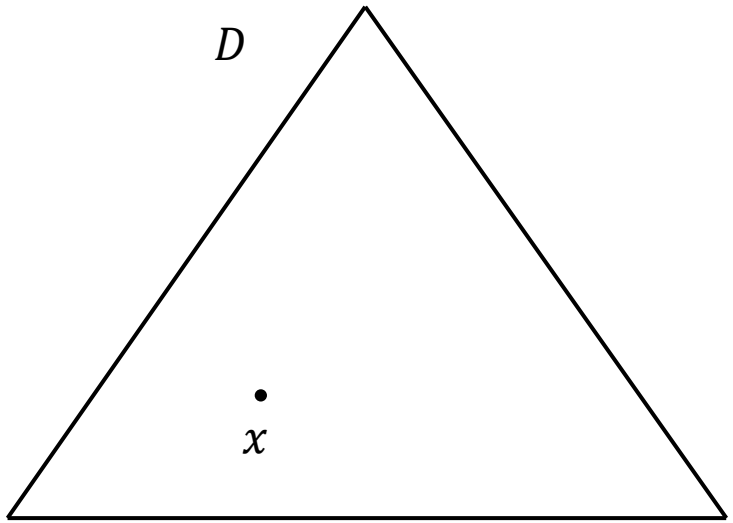
Replace Brouwer by Sperner:

Brouwer function $F: D \to D$

PPAD-membership of NASH

Replace Brouwer by Sperner:

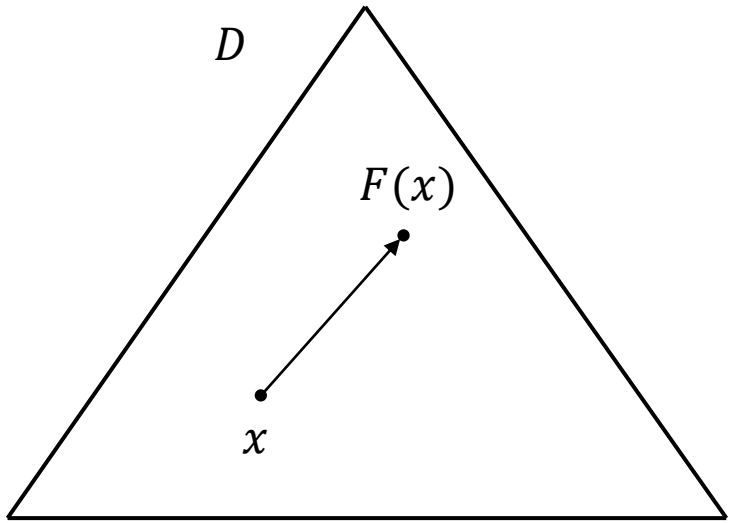
Brouwer function $F: D \to D$



PPAD-membership of NASH

Replace Brouwer by Sperner:

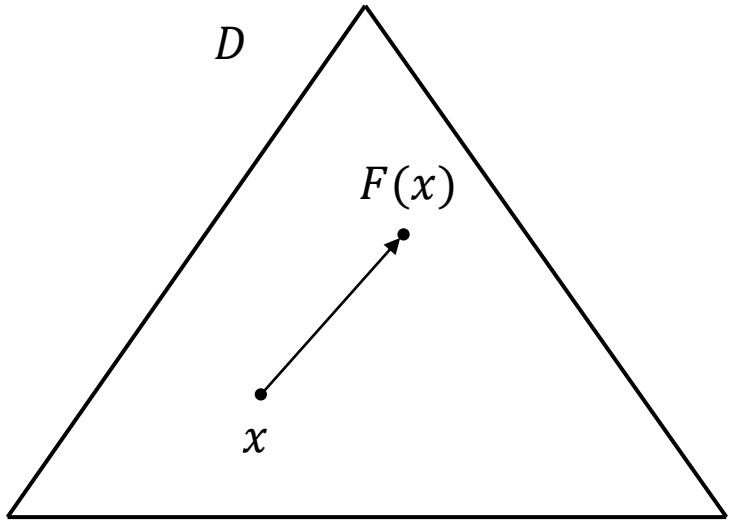
Brouwer function $F: D \to D$



PPAD-membership of NASH

Replace Brouwer by Sperner:

Brouwer function $F: D \to D$



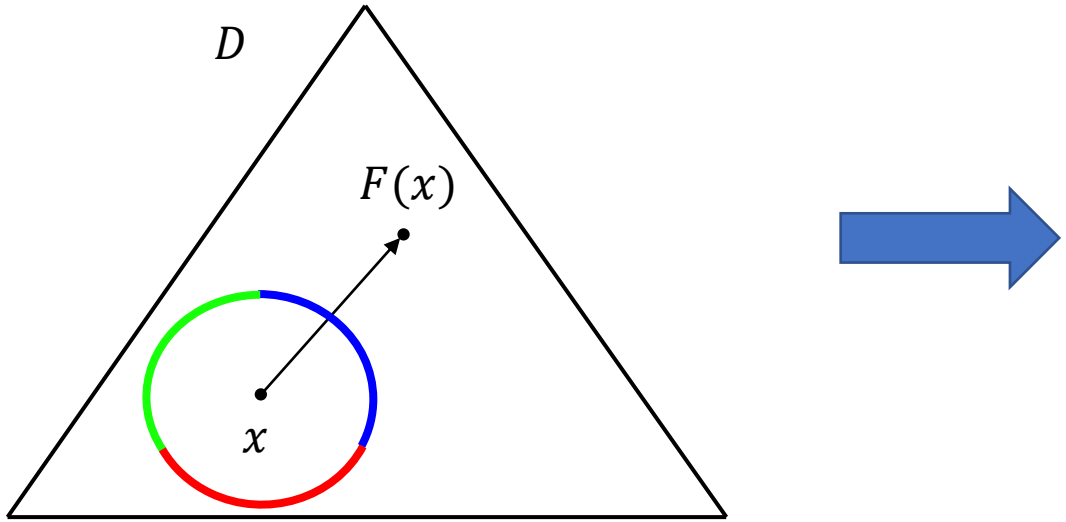
Sperner coloring:

PPAD-membership of NASH

Replace Brouwer by Sperner:

Brouwer function $F: D \to D$

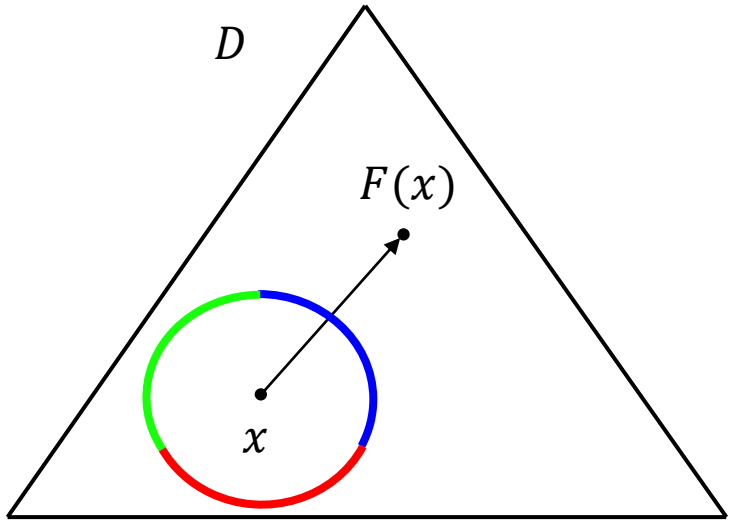
Sperner coloring:



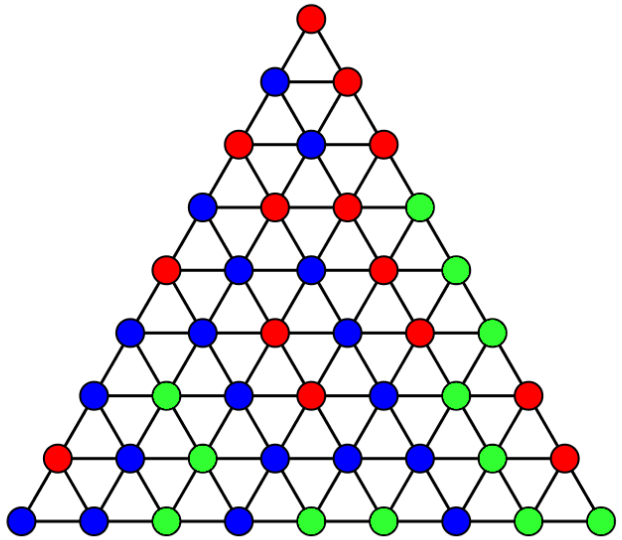
PPAD-membership of NASH

Replace Brouwer by Sperner:

Brouwer function $F: D \to D$



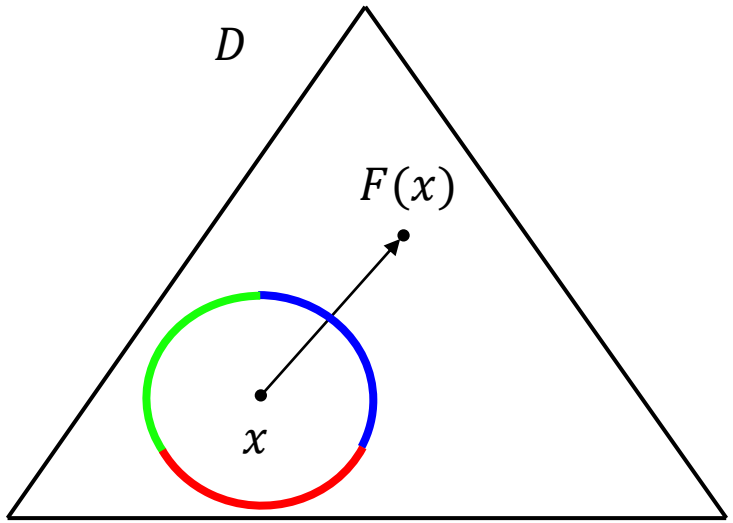
Sperner coloring:



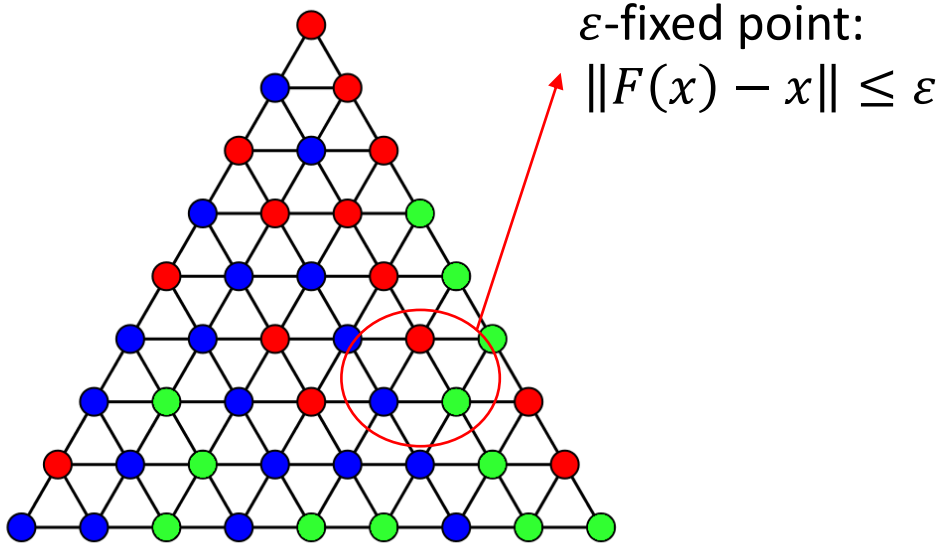
PPAD-membership of NASH

Replace Brouwer by Sperner:

Brouwer function $F: D \to D$



Sperner coloring:



Focus on PPAD

I. How do we prove PPAD-membership?
   1. Sperner's Lemma
   2. The END-OF-LINE Problem
   3. Example: NASH

II. How do we prove PPAD-hardness?
   1. **Refresher: Reductions**

How to prove PPAD-hardness?

How to prove PPAD-hardness?

Refresher: How to prove NP-hardness of some problem A?

How to prove PPAD-hardness?

Refresher: How to prove NP-hardness of some problem A?

→ reduce from an NP-complete problem, e.g., 3-SAT

How to prove PPAD-hardness?

Refresher: How to prove NP-hardness of some problem A?

→ reduce from an NP-complete problem, e.g., 3-SAT

3-SAT $\leq_p$ A :

How to prove PPAD-hardness?

Refresher: How to prove NP-hardness of some problem A?

→ reduce from an NP-complete problem, e.g., 3-SAT

3-SAT $\leq_p$ A : there exists a poly-time function f that maps instances of 3-SAT to instances of A, and such that for any instance x of 3-SAT:

How to prove PPAD-hardness?

Refresher: How to prove NP-hardness of some problem A?

→ reduce from an NP-complete problem, e.g., 3-SAT

3-SAT $\leq_p$ A : there exists a poly-time function f that maps instances of 3-SAT to instances of A, and such that for any instance x of 3-SAT:

x is YES-instance for 3-SAT iff $f(x)$ is YES-instance for A

How to prove PPAD-hardness?

Refresher: How to prove NP-hardness of some problem A?

→ reduce from an NP-complete problem, e.g., 3-SAT

3-SAT $\leq_p$ A : there exists a poly-time function f that maps instances of 3-SAT to instances of A, and such that for any instance x of 3-SAT:

x is YES-instance for 3-SAT iff $f(x)$ is YES-instance for A

Intuition: “if we can solve A, then we can solve 3-SAT”

How to prove PPAD-hardness?

Refresher: How to prove PPAD-hardness of NASH?

→ reduce from an PPAD-complete problem, e.g., END-OF-LINE

END-OF-LINE $\leq_p$ **NASH** : there exists a poly-time function f that maps instances of END-OF-LINE to instances of NASH, and such that for any instance x of END-OF-LINE:

x is YES-instance for END-OF-LINE iff $f(x)$ is YES-instance for NASH

Intuition: "if we can solve NASH, then we can solve END-OF-LINE"

How to prove PPAD-hardness?

Refresher: How to prove PPAD-hardness of NASH?

→ reduce from an PPAD-complete problem, e.g., END-OF-LINE

END-OF-LINE $\leq_p$ **NASH** : there exists a poly-time function f that maps instances of END-OF-LINE to instances of NASH, and such that for any instance x of END-OF-LINE:

~~x is YES-instance for END-OF-LINE iff $f(x)$ is YES-instance for NASH~~

Intuition: "if we can solve NASH, then we can solve END-OF-LINE"

How to prove PPAD-hardness?

Refresher: How to prove PPAD-hardness of NASH?

→ reduce from an PPAD-complete problem, e.g., END-OF-LINE

END-OF-LINE $\leq_p$ **NASH** : there exists a poly-time function f that maps instances of END-OF-LINE to instances of NASH, and such that for any instance x of END-OF-LINE:

~~x is YES-instance for END-OF-LINE iff $f(x)$ is YES-instance for NASH~~

Given any solution y to instance $f(x)$ of NASH, we can efficiently obtain a solution to instance x of END-OF-LINE

Intuition: "if we can solve NASH, then we can solve END-OF-LINE"

How to prove PPAD-hardness for NASH?

How to prove PPAD-hardness for NASH?

In essence:

$$\text{END-OF-LINE} \leq_p \text{NASH}$$

How to prove PPAD-hardness for NASH?

In essence:

$$\text{END-OF-LINE} \leq_p \text{NASH}$$

But actually:

$$\text{END-OF-LINE} \leq_p \text{DISCRETE-BROUWER} \leq_p \text{GENERALIZED-CIRCUIT} \leq_p \text{NASH}$$

How to prove PPAD-hardness for NASH?

In essence:

$$\text{END-OF-LINE} \leq_p \text{NASH}$$

But actually:

$$\text{END-OF-LINE} \leq_p \text{DISCRETE-BROUWER} \leq_p \text{GENERALIZED-CIRCUIT} \leq_p \text{NASH}$$

All subsequent work reduces from GENERALIZED-CIRCUIT or some version of NASH

Generalized Circuit

DEFINITION 3.4. *Given a generalized circuit $\mathcal{S} = (V, \mathcal{T})$, we say that an assignment* $\mathbf{x}\colon V \to [0,1]$ ϵ-approximately satisfies $\mathcal{S}$ *if for each of the following gates,* $\mathbf{x}$ *satisfies the corresponding constraints:*

Gate	*Constraint*
$G_\zeta\,(\alpha \mid\mid a)$	$\mathbf{x}[a] = \alpha \pm \epsilon$
$G_{\times\zeta}\,(\alpha \mid a \mid b)$	$\mathbf{x}[b] = \alpha \cdot \mathbf{x}[a] \pm \epsilon$
$G_=\,(\mid a \mid b)$	$\mathbf{x}[b] = \mathbf{x}[a] \pm \epsilon$
$G_+\,(\mid a, b \mid c)$	$\mathbf{x}[c] = \min(\mathbf{x}[a] + \mathbf{x}[b], 1) \pm \epsilon$
$G_-\,(\mid a, b \mid c)$	$\mathbf{x}[c] = \max(\mathbf{x}[a] - \mathbf{x}[b], 0) \pm \epsilon$
$G_<\,(\mid a, b \mid c)$	$\mathbf{x}[c] = \begin{cases} 1 \pm \epsilon & \mathbf{x}[a] < \mathbf{x}[b] - \epsilon \\ 0 \pm \epsilon & \mathbf{x}[a] > \mathbf{x}[b] + \epsilon \end{cases}$
$G_\vee\,(\mid a, b \mid c)$	$\mathbf{x}[c] = \begin{cases} 1 \pm \epsilon & \mathbf{x}[a] = 1 \pm \epsilon \text{ or } \mathbf{x}[b] = 1 \pm \epsilon \\ 0 \pm \epsilon & \mathbf{x}[a] = 0 \pm \epsilon \text{ and } \mathbf{x}[b] = 0 \pm \epsilon \end{cases}$
$G_\wedge\,(\mid a, b \mid c)$	$\mathbf{x}[c] = \begin{cases} 1 \pm \epsilon & \mathbf{x}[a] = 1 \pm \epsilon \text{ and } \mathbf{x}[b] = 1 \pm \epsilon \\ 0 \pm \epsilon & \mathbf{x}[a] = 0 \pm \epsilon \text{ or } \mathbf{x}[b] = 0 \pm \epsilon \end{cases}$
$G_\neg\,(\mid a \mid b)$	$\mathbf{x}[b] = \begin{cases} 1 \pm \epsilon & \mathbf{x}[a] = 0 \pm \epsilon \\ 0 \pm \epsilon & \mathbf{x}[a] = 1 \pm \epsilon \end{cases}$

(where G_ζ and $G_{\times\zeta}$ also take a parameter $\alpha \in [0,1]$).

*Given a generalized circuit $\mathcal{S} = (V, \mathcal{T})$, ϵ-*GCIRCUIT *is the problem of finding an assignment that ϵ-approximately satisfies it.*

Focus on PPAD

I. How do we prove PPAD-membership?
   1. Sperner's Lemma
   2. The END-OF-LINE Problem
   3. Example: NASH

II. How do we prove PPAD-hardness?
   1. Refresher: Reductions
   2. **The PURE-CIRCUIT Problem**

The Pure-Circuit Problem

The Pure-Circuit Problem

Input: A Boolean circuit, with a twist:

The Pure-Circuit Problem

Input: A Boolean circuit, with a twist:

1. The circuit can have cycles

The Pure-Circuit Problem

Input: A Boolean circuit, with a twist:

1. The circuit can have cycles
2. Nodes take values in $\{0,1,\perp\}$, instead of just $\{0,1\}$

The Pure-Circuit Problem

Input: A Boolean circuit, with a twist:

1. The circuit can have cycles
2. Nodes take values in $\{0,1,\perp\}$, instead of just $\{0,1\}$
3. In addition to the standard logical gates (NOT, OR, AND), the circuit can also have “PURIFY” gates

The Pure-Circuit Problem

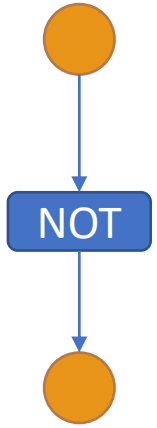
Input: A Boolean circuit, with a twist:

1. The circuit can have cycles
2. Nodes take values in $\{0,1,\perp\}$, instead of just $\{0,1\}$
3. In addition to the standard logical gates (NOT, OR, AND), the circuit can also have “PURIFY” gates

Goal: Assign a value (in $\{0,1,\perp\}$) to each node, such that all gates are “satisfied”

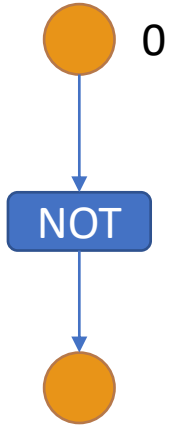
The Pure-Circuit Problem

NOT gate:



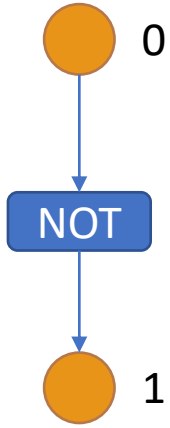
The Pure-Circuit Problem

NOT gate:



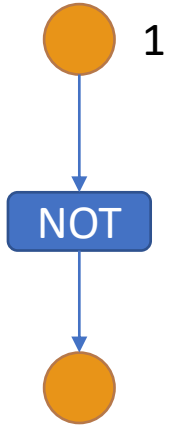
The Pure-Circuit Problem

NOT gate:



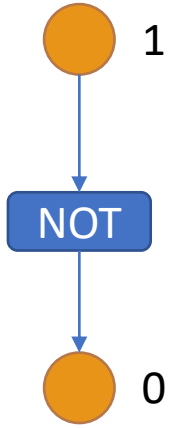
The Pure-Circuit Problem

NOT gate:



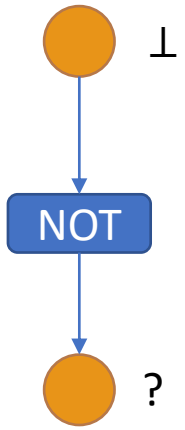
The Pure-Circuit Problem

NOT gate:



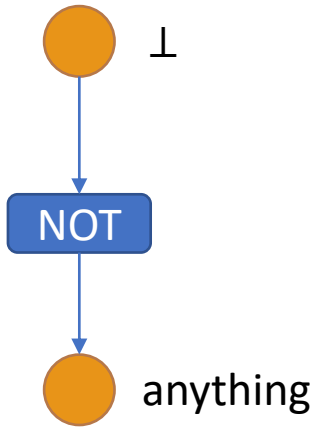
The Pure-Circuit Problem

NOT gate:



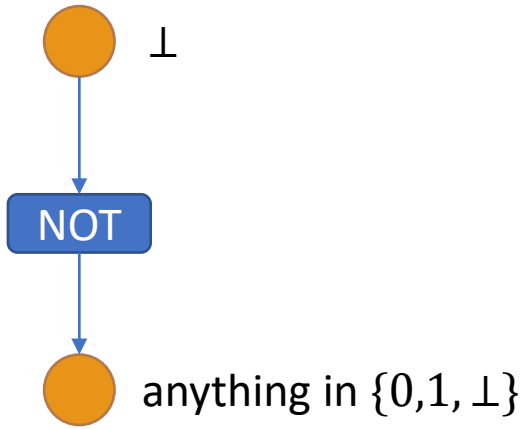
The Pure-Circuit Problem

NOT gate:



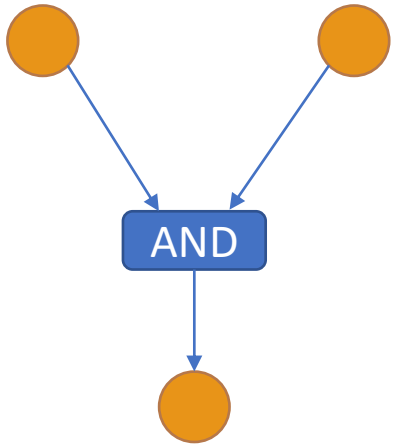
The Pure-Circuit Problem

NOT gate:



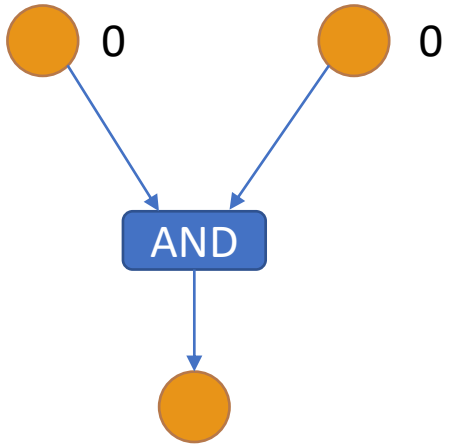
The Pure-Circuit Problem

AND gate:



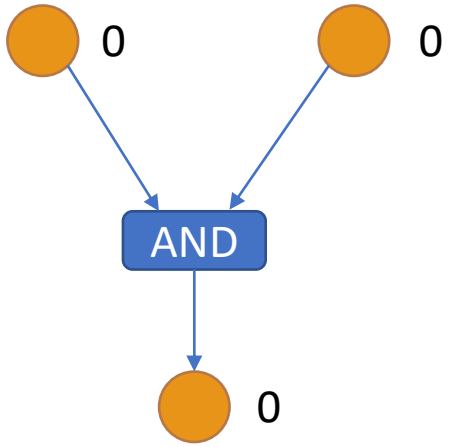
The Pure-Circuit Problem

AND gate:



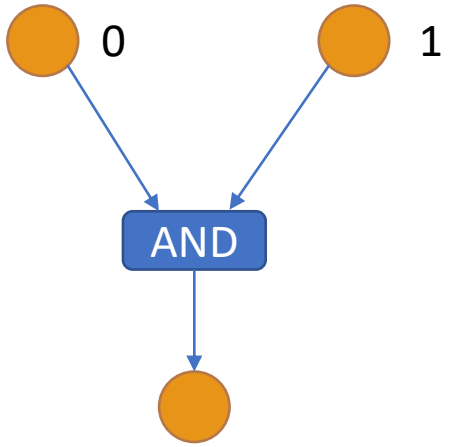
The Pure-Circuit Problem

AND gate:



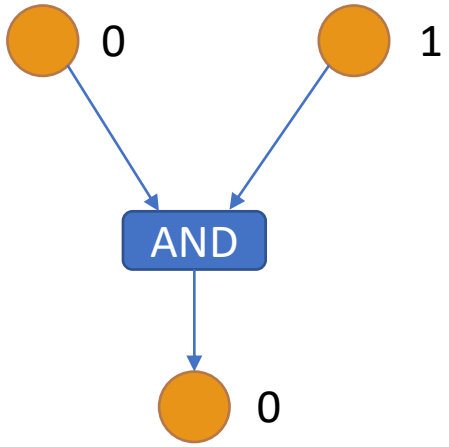
The Pure-Circuit Problem

AND gate:



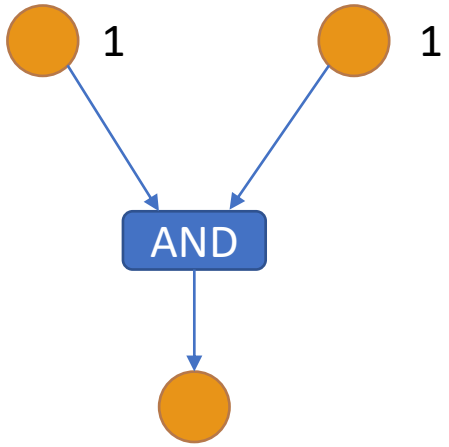
The Pure-Circuit Problem

AND gate:



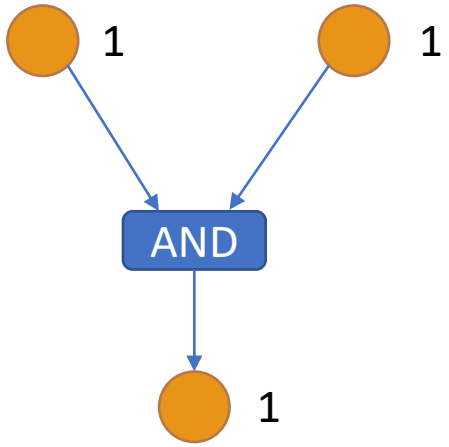
The Pure-Circuit Problem

AND gate:



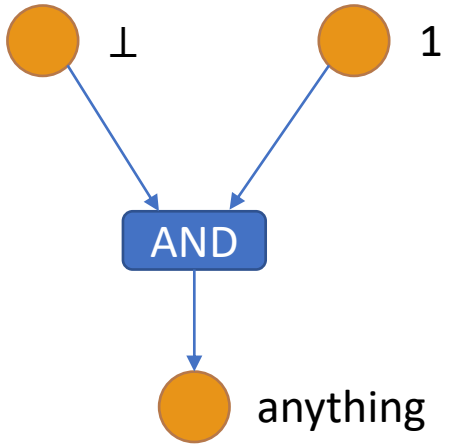
The Pure-Circuit Problem

AND gate:



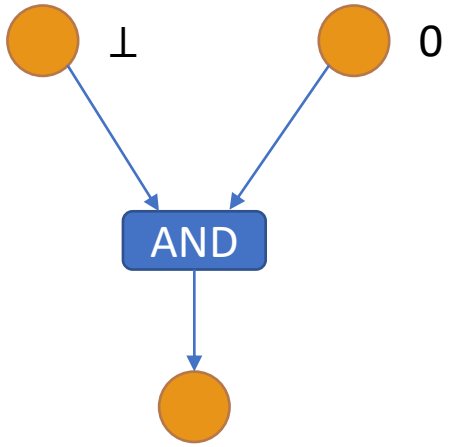
The Pure-Circuit Problem

AND gate:



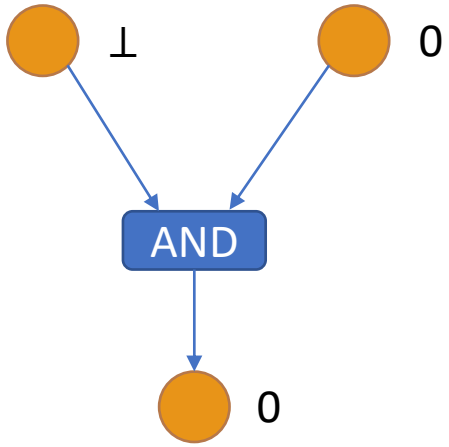
The Pure-Circuit Problem

AND gate:



The Pure-Circuit Problem

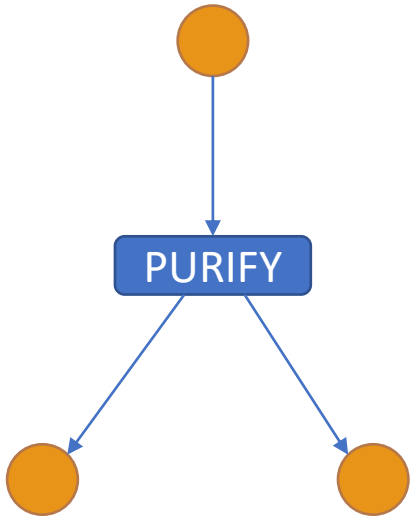
AND gate:



(with robustness!)

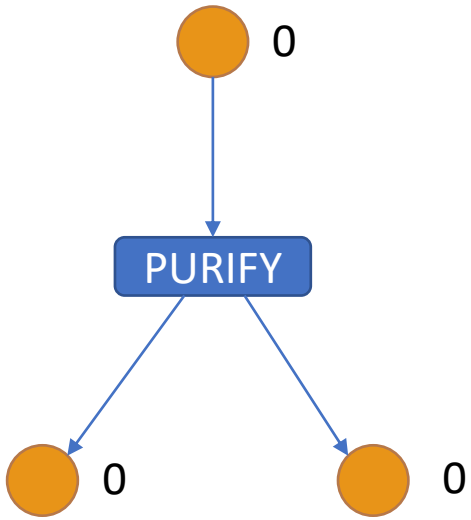
The Pure-Circuit Problem

PURIFY gate:



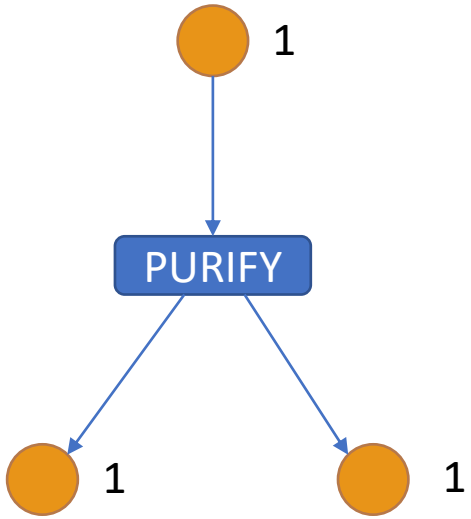
The Pure-Circuit Problem

PURIFY gate:



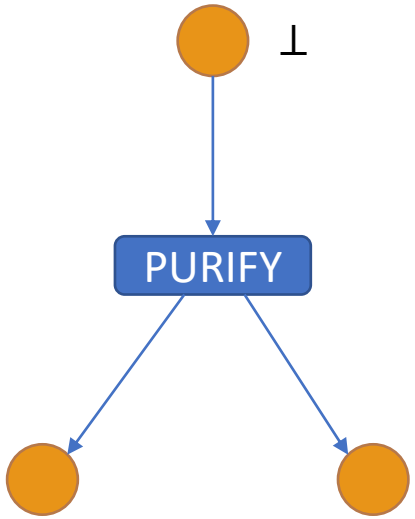
The Pure-Circuit Problem

PURIFY gate:



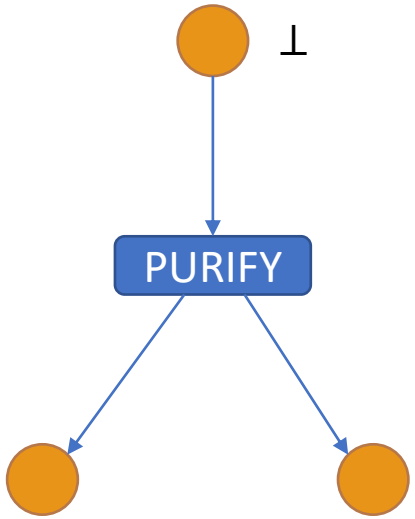
The Pure-Circuit Problem

PURIFY gate:



The Pure-Circuit Problem

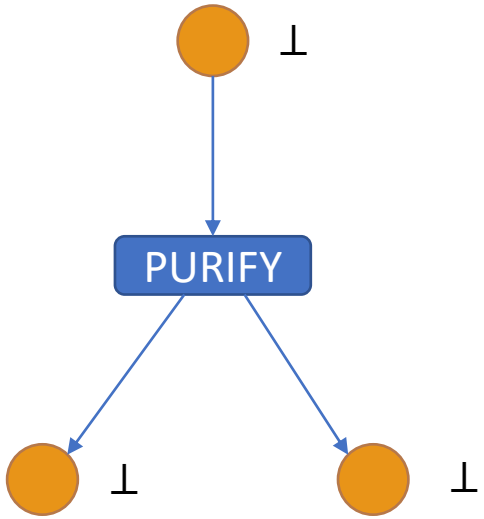
PURIFY gate:



at least one in {0,1}

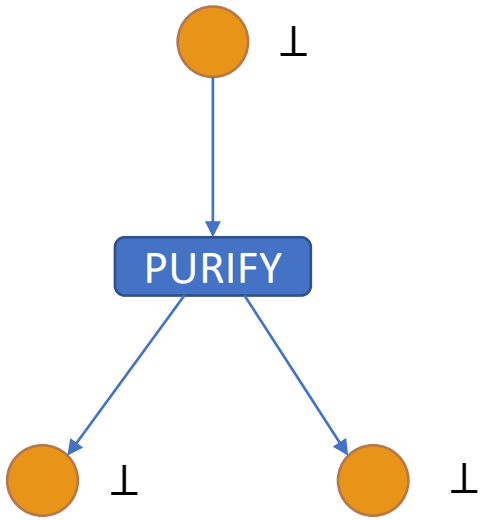
The Pure-Circuit Problem

PURIFY gate:



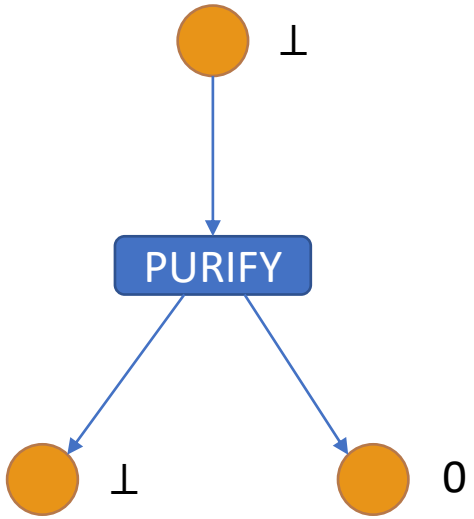
The Pure-Circuit Problem

PURIFY gate:



The Pure-Circuit Problem

PURIFY gate:

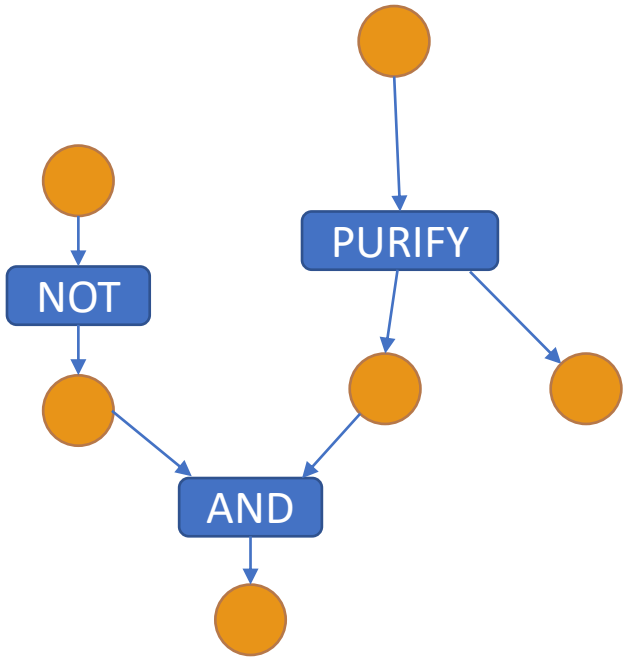


The Pure-Circuit Problem: Examples

1. The circuit can have cycles

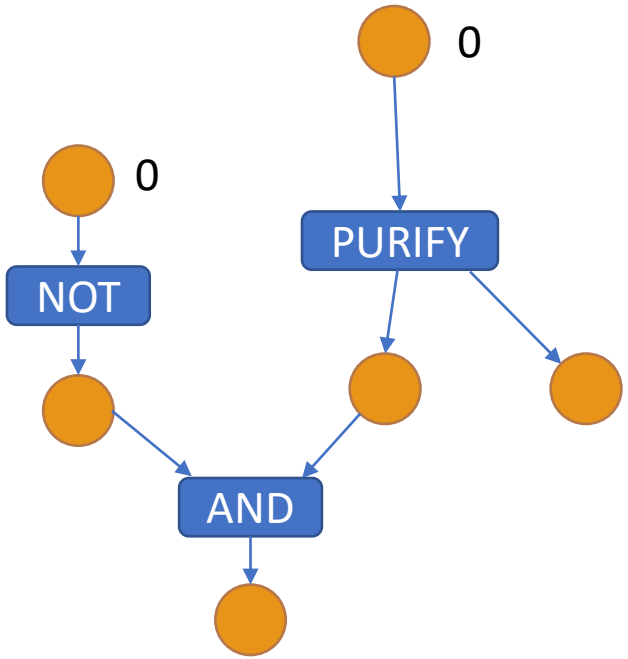
The Pure-Circuit Problem: Examples

1. The circuit can have cycles



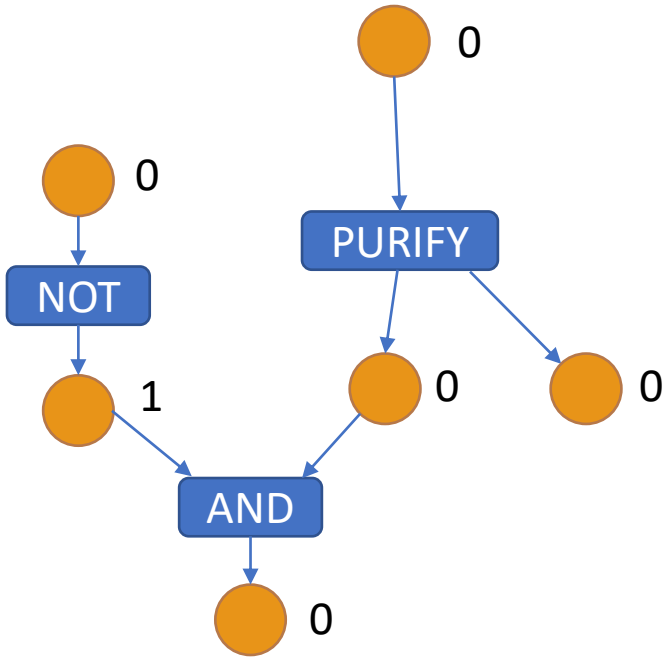
The Pure-Circuit Problem: Examples

1. The circuit can have cycles



The Pure-Circuit Problem: Examples

1. The circuit can have cycles

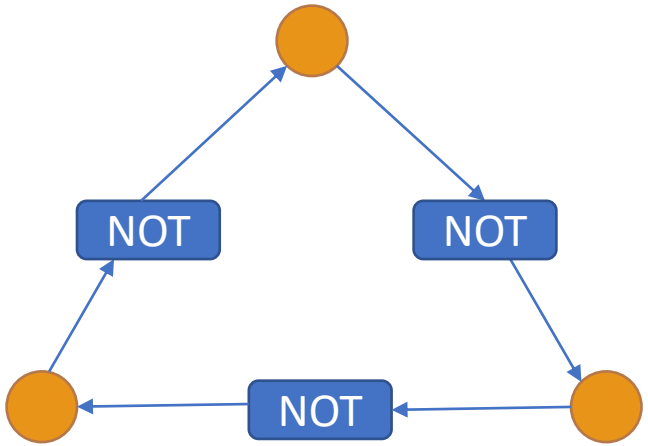


The Pure-Circuit Problem: Examples

2. Nodes take values in $\{0,1,\perp\}$, instead of just $\{0,1\}$

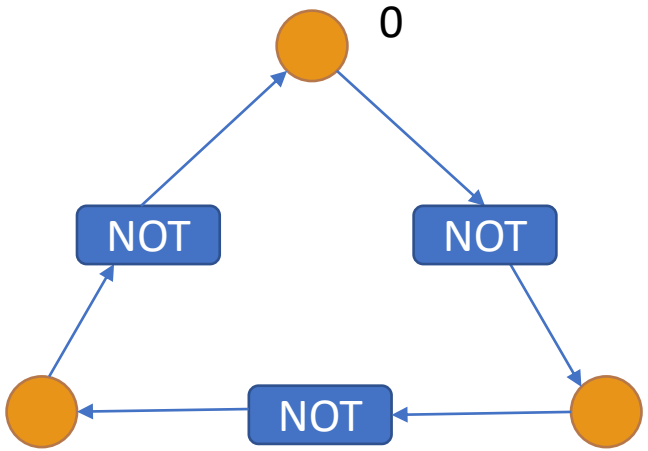
The Pure-Circuit Problem: Examples

2. Nodes take values in $\{0,1,\perp\}$, instead of just $\{0,1\}$



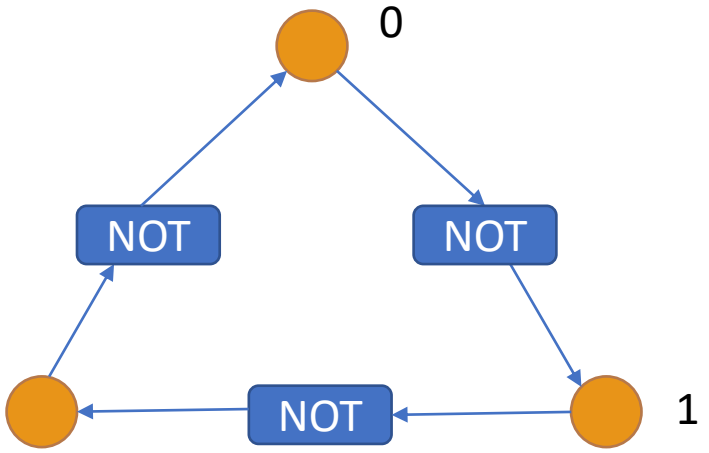
The Pure-Circuit Problem: Examples

2. Nodes take values in $\{0,1,\perp\}$, instead of just $\{0,1\}$



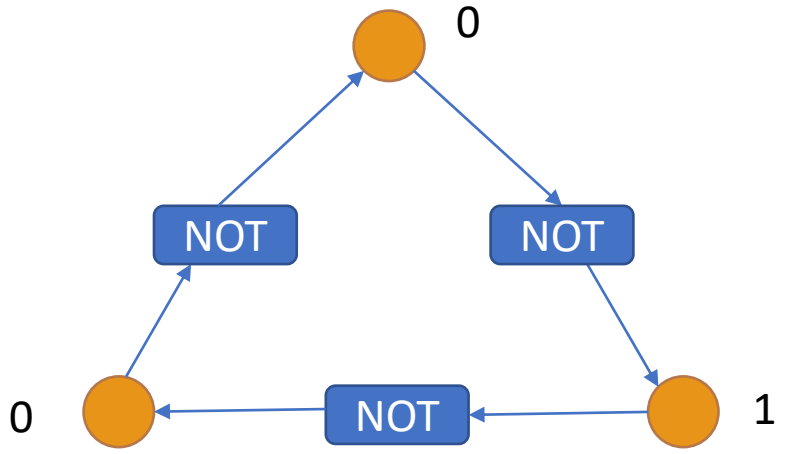
The Pure-Circuit Problem: Examples

2. Nodes take values in $\{0,1,\perp\}$, instead of just $\{0,1\}$



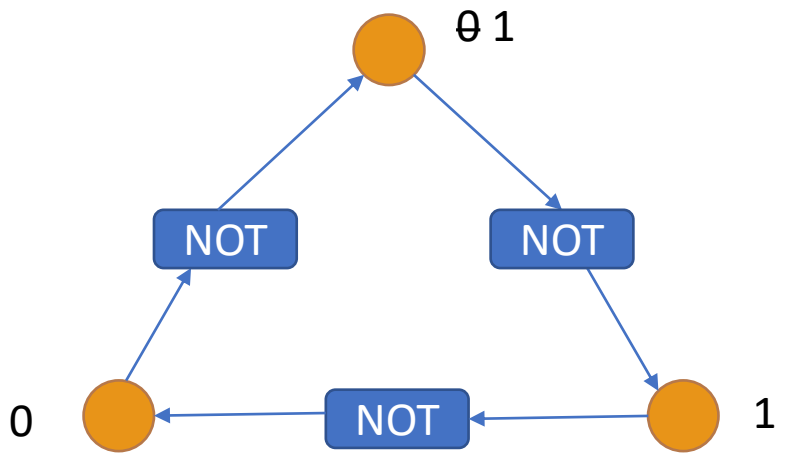
The Pure-Circuit Problem: Examples

2. Nodes take values in $\{0,1,\perp\}$, instead of just $\{0,1\}$



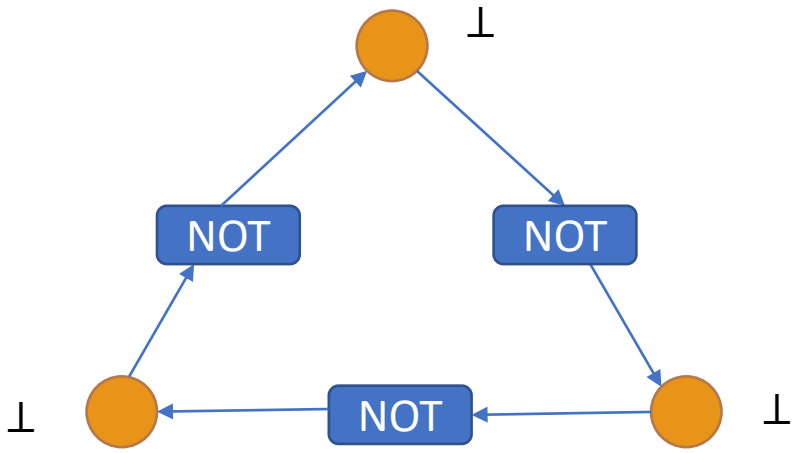
The Pure-Circuit Problem: Examples

2. Nodes take values in $\{0,1,\perp\}$, instead of just $\{0,1\}$



The Pure-Circuit Problem: Examples

2. Nodes take values in $\{0,1,\perp\}$, instead of just $\{0,1\}$

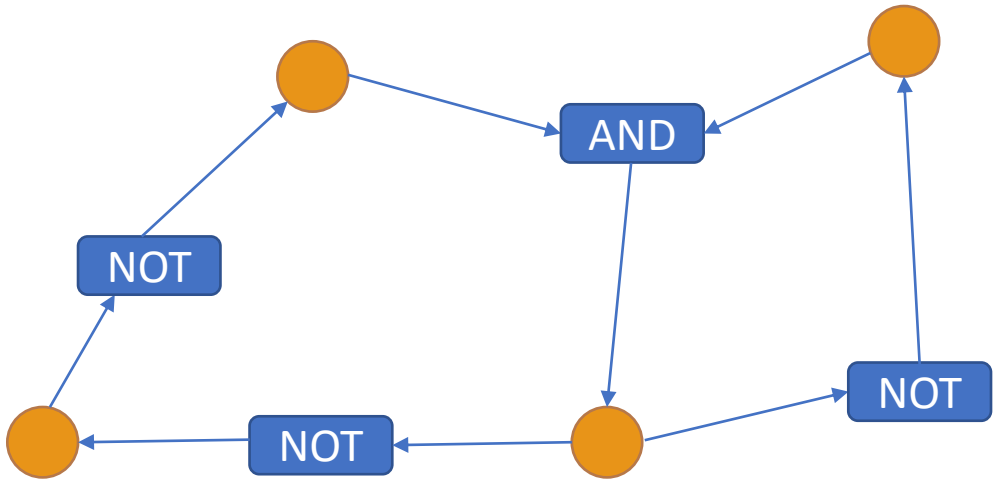


The Pure-Circuit Problem: Examples

3. In addition to the standard logical gates (NOT, OR, AND), the circuit can also have “PURIFY” gates

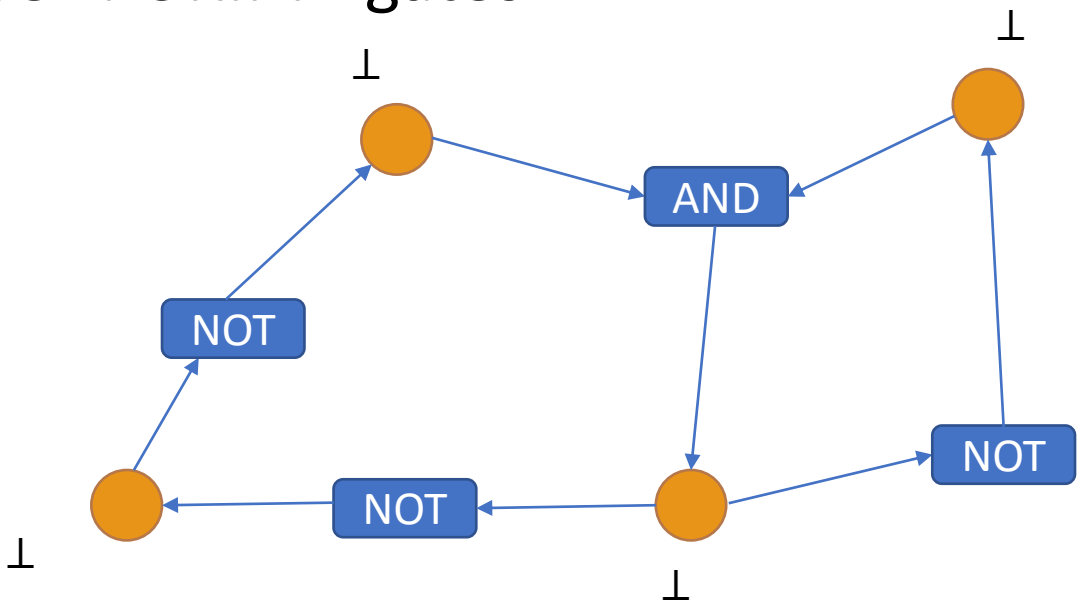
The Pure-Circuit Problem: Examples

3. In addition to the standard logical gates (NOT, OR, AND), the circuit can also have “PURIFY” gates

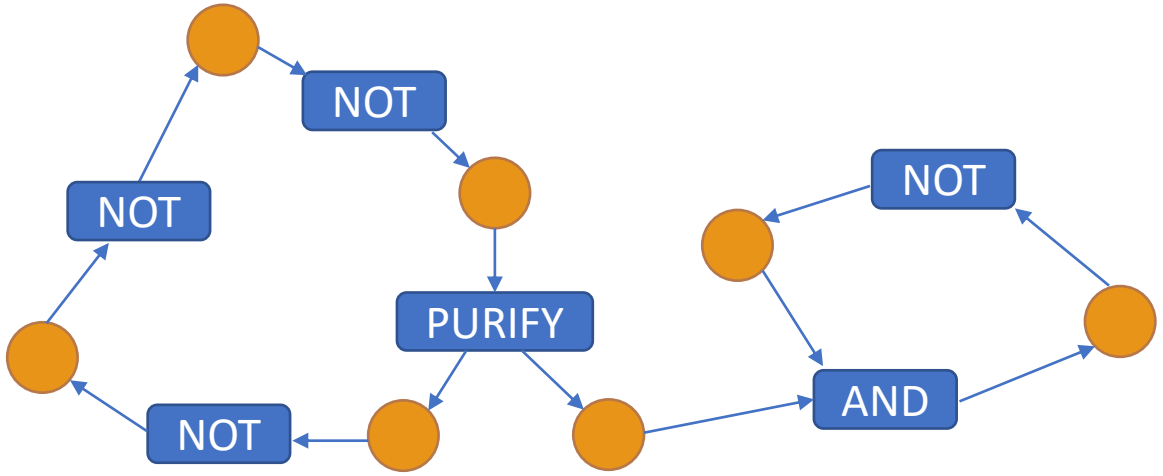


The Pure-Circuit Problem: Examples

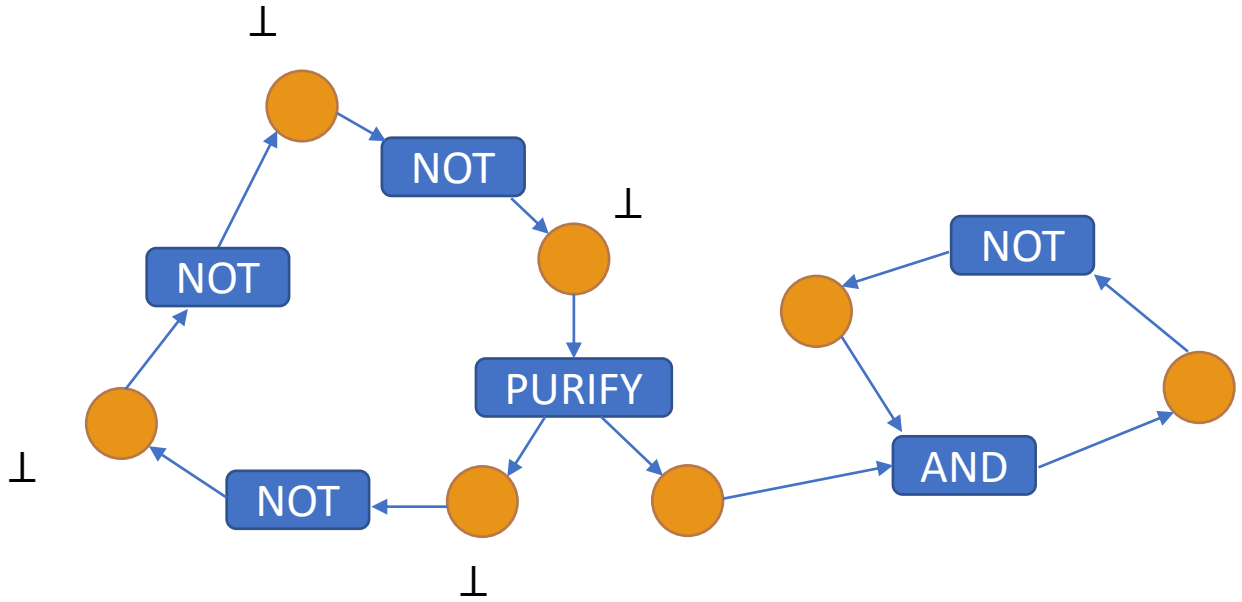
3. In addition to the standard logical gates (NOT, OR, AND), the circuit can also have “PURIFY” gates



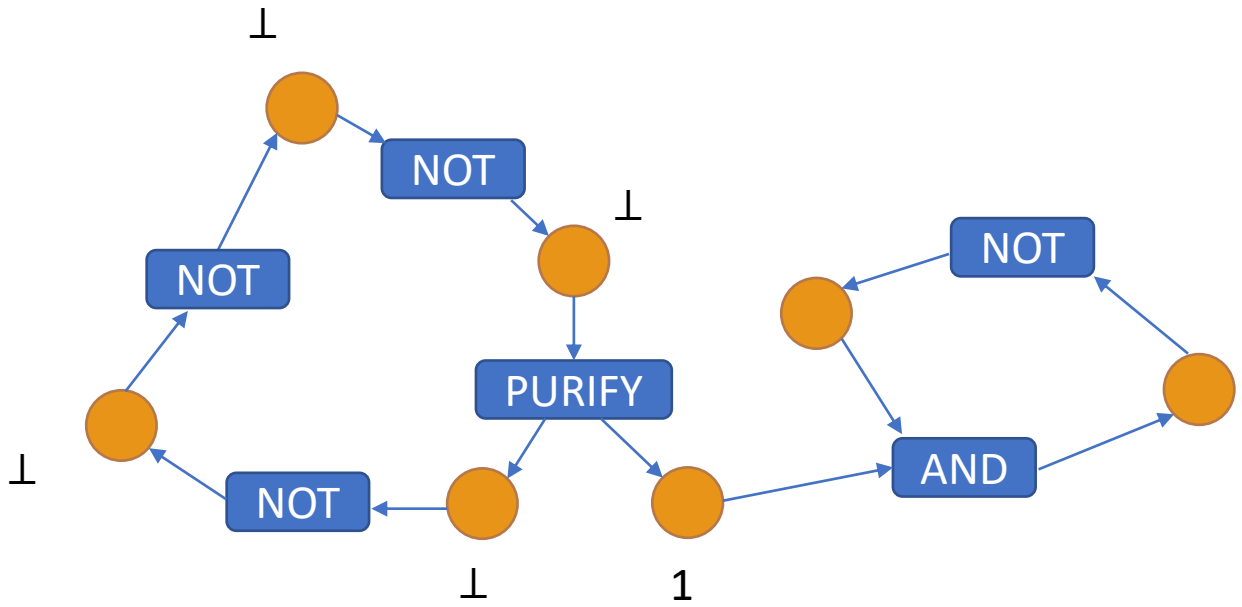
The Pure-Circuit Problem: Examples



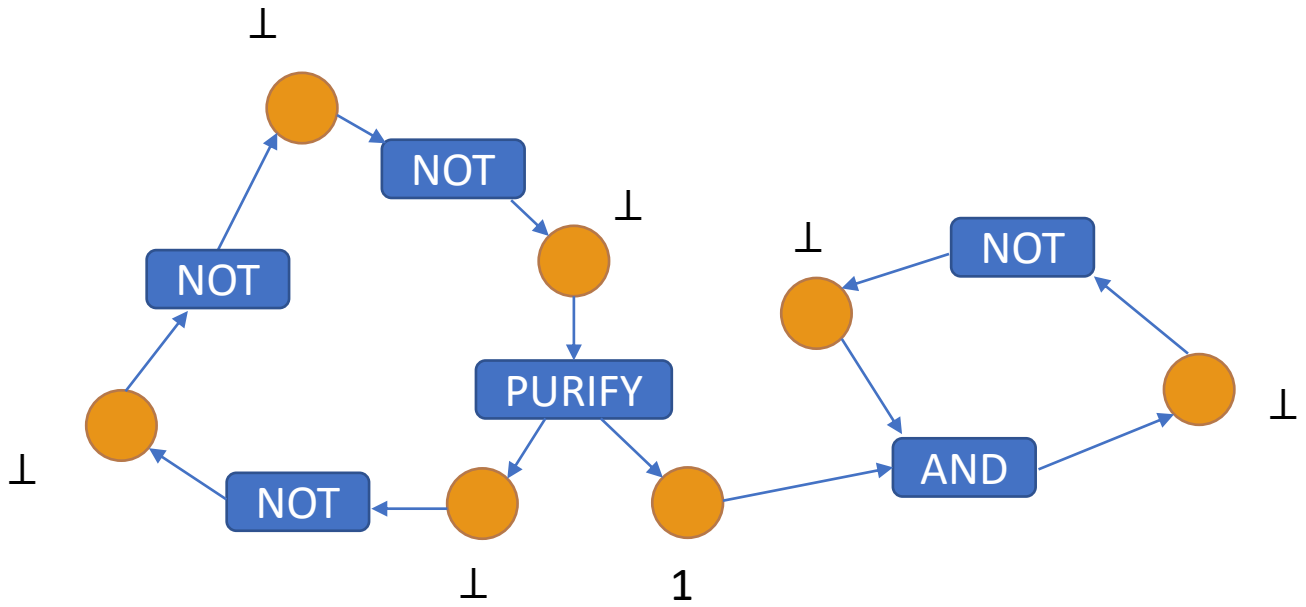
The Pure-Circuit Problem: Examples



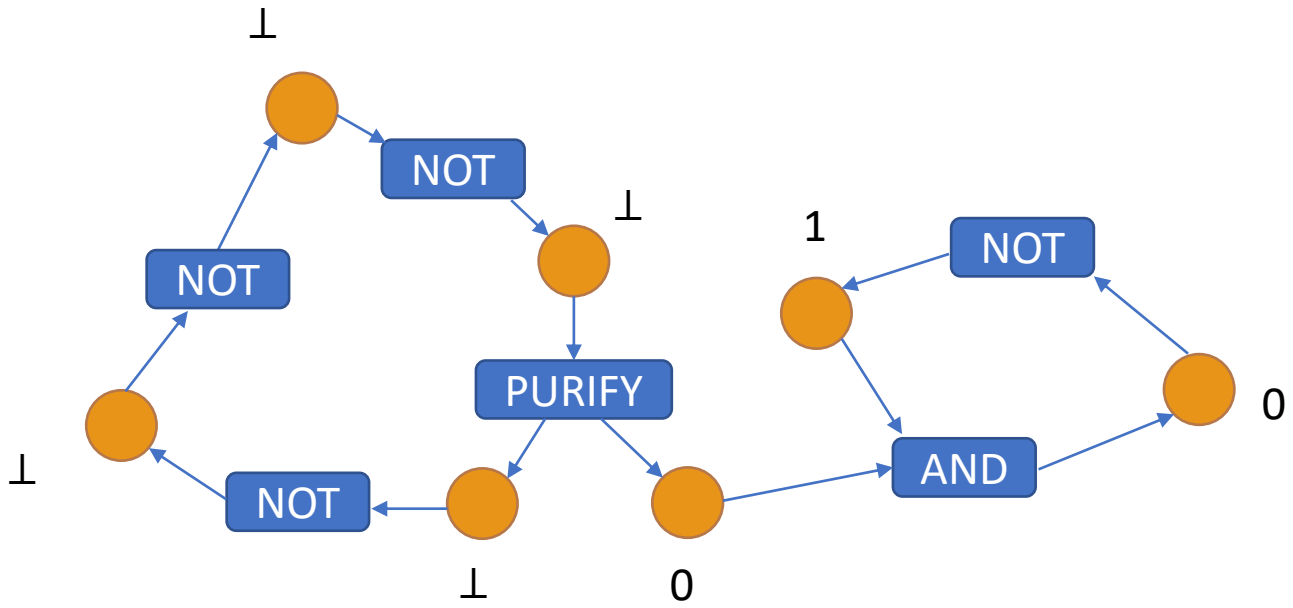
The Pure-Circuit Problem: Examples



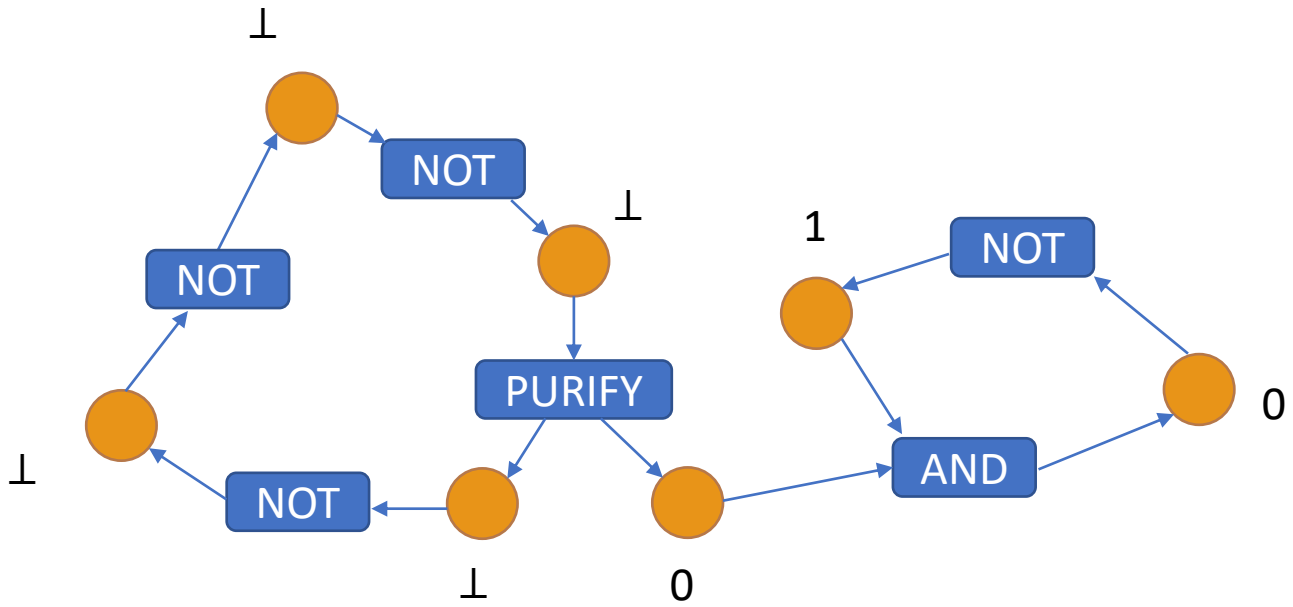
The Pure-Circuit Problem: Examples



The Pure-Circuit Problem: Examples



The Pure-Circuit Problem: Examples



Theorem (with Deligkas, Fearnley, Melissourgos ‘22):
Pure-Circuit is PPAD-complete with gates NOT, AND, PURIFY.

Focus on PPAD

I. How do we prove PPAD-membership?
   1. Sperner's Lemma
   2. The END-OF-LINE Problem
   3. Example: NASH

II. How do we prove PPAD-hardness?
   1. Refresher: Reductions
   2. The PURE-CIRCUIT Problem
   3. **Example: Polymatrix Games**

(Binary-action) Polymatrix Games

(Binary-action) Polymatrix Games

- n players, each with two possible actions ($zero$ and one)

(Binary-action) Polymatrix Games

- n players, each with two possible actions ($zero$ and one)
- the number of pure strategy profiles $(a_1, a_2, \ldots, a_n) \in \{zero, one\}^n$ is 2^n

(Binary-action) Polymatrix Games

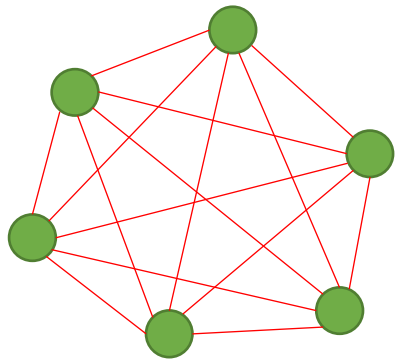
- n players, each with two possible actions ($zero$ and one)
- the number of pure strategy profiles $(a_1, a_2, \dots, a_n) \in \{zero, one\}^n$ is 2^n
- polymatrix game: additional structure that allows succinct representation of utilities

$$u_i(a_1, a_2, \dots, a_n) = U_{i1}(a_i, a_1) + U_{i2}(a_i, a_2) + \cdots + U_{in}(a_i, a_n)$$

(Binary-action) Polymatrix Games

- n players, each with two possible actions ($zero$ and one)
- the number of pure strategy profiles $(a_1, a_2, \dots, a_n) \in \{zero, one\}^n$ is 2^n
- polymatrix game: additional structure that allows succinct representation of utilities

$$u_i(a_1, a_2, \dots, a_n) = U_{i1}(a_i, a_1) + U_{i2}(a_i, a_2) + \cdots + U_{in}(a_i, a_n)$$



Polymatrix Games: Nash Equilibria

Nash equilibrium: Every player plays a best-response mixed strategy

Polymatrix Games: Nash Equilibria

Nash equilibrium: Every player plays a best-response mixed strategy

Nash's theorem: There exists a mixed Nash equilibrium.

Polymatrix Games: Nash Equilibria

Nash equilibrium: Every player plays a best-response mixed strategy

Nash's theorem: There exists a mixed Nash equilibrium.

Question: Can we compute it in polynomial time? Or at least an approximate Nash equilibrium?

Polymatrix Games: Nash Equilibria

Nash equilibrium: Every player plays a best-response mixed strategy

Nash's theorem: There exists a mixed Nash equilibrium.

Question: Can we compute it in polynomial time? Or at least an approximate Nash equilibrium?

ε-Well-Supported-Nash-Equilibrium (**$\boldsymbol{\varepsilon}$-WSNE**): Every player plays a mixed strategy that only puts weight on ε-best-response actions

Polymatrix Games: Nash Equilibria

Nash equilibrium: Every player plays a best-response mixed strategy

Nash's theorem: There exists a mixed Nash equilibrium.

Question: Can we compute it in polynomial time? Or at least an approximate Nash equilibrium?

ε-Well-Supported-Nash-Equilibrium (**$\boldsymbol{\varepsilon}$-WSNE**): Every player plays a mixed strategy that only puts weight on ε-best-response actions

(Note: additive approximation, smaller ε is better)

A simple algorithm

- A simple algorithm for computing a $1/3$-WSNE in a binary-action polymatrix game:

A simple algorithm

- A simple algorithm for computing a $1/3$-WSNE in a binary-action polymatrix game:

1. Find a player such that one of its two actions is always a $1/3$-best-response (no matter what the other players play). Fix the player's strategy to that action, and remove the player from the game.

A simple algorithm

- A simple algorithm for computing a $1/3$-WSNE in a binary-action polymatrix game:

1. Find a player such that one of its two actions is always a $1/3$-best-response (no matter what the other players play). Fix the player's strategy to that action, and remove the player from the game.
2. Repeat Step 1 until no such player exists anymore.

A simple algorithm

- A simple algorithm for computing a $1/3$-WSNE in a binary-action polymatrix game:

1. Find a player such that one of its two actions is always a $1/3$-best-response (no matter what the other players play). Fix the player's strategy to that action, and remove the player from the game.
2. Repeat Step 1 until no such player exists anymore.
3. For the remaining players, have them mix uniformly between their two actions, i.e., have them play $\left(\frac{1}{2}, \frac{1}{2}\right)$.

A simple algorithm

- A simple algorithm for computing a $1/3$-WSNE in a binary-action polymatrix game:

1. Find a player such that one of its two actions is always a $1/3$-best-response (no matter what the other players play). Fix the player's strategy to that action, and remove the player from the game.
2. Repeat Step 1 until no such player exists anymore.
3. For the remaining players, have them mix uniformly between their two actions, i.e., have them play $\left(\frac{1}{2}, \frac{1}{2}\right)$.

→ Can show that this always yields a $1/3$-WSNE (by a simple direct computation)

This is a pretty naive algorithm…

This is a pretty naive algorithm…
Can we do better?

This is a pretty naive algorithm…
Can we do better?
Can we provide evidence that we *cannot* do better?

PPAD-hardness

PPAD-hardness

Theorem [Daskalakis, Goldberg, Papadimitriou 09]:

Computing an ε-WSNE is PPAD-complete, for *inverse-exponential* ε.

PPAD-hardness

Theorem [Daskalakis, Goldberg, Papadimitriou 09]:

Computing an ε-WSNE is PPAD-complete, for *inverse-exponential* ε.

Theorem [Chen, Deng, Teng 09]:

Computing an ε-WSNE is PPAD-complete, for *inverse-polynomial* ε.

PPAD-hardness

Theorem [Daskalakis, Goldberg, Papadimitriou 09]:

Computing an ε-WSNE is PPAD-complete, for *inverse-exponential* ε.

Theorem [Chen, Deng, Teng 09]:

Computing an ε-WSNE is PPAD-complete, for *inverse-polynomial* ε.

Theorem [Rubinstein 18]:

There exists a constant $\varepsilon \in (0,1)$ such that computing an ε-WSNE is PPAD-complete.

PPAD-hardness

Theorem [Rubinstein 18]:

There exists a constant $\varepsilon \in (0,1)$ such that computing an ε-WSNE is PPAD-complete.

PPAD-hardness

Theorem [Rubinstein 18]:

There exists a constant $\varepsilon \in (0,1)$ such that computing an ε-WSNE is PPAD-complete.

Theorem:

For any constant $\varepsilon < 1/3$, computing an ε-WSNE is PPAD-complete.

PPAD-hardness

Theorem [Rubinstein 18]:

There exists a constant $\varepsilon \in (0,1)$ such that computing an ε-WSNE is PPAD-complete.

Theorem:

For any constant $\varepsilon < 1/3$, computing an ε-WSNE is PPAD-complete.

Proof: For any $\varepsilon < 1/3$, provide a reduction:

$$\text{Pure-Circuit} \leq_p \varepsilon\text{-WSNE-Polymatrix}.$$

Reduction

Given: An instance of Pure-Circuit

Goal: Construct a polymatrix game

Reduction

Given: An instance of Pure-Circuit

Goal: Construct a polymatrix game

Idea: Create one player for each node in the Pure-Circuit instance.
Every player has two actions: ‘zero’ and ‘one’.

Reduction

Given: An instance of Pure-Circuit

Goal: Construct a polymatrix game

Idea: Create one player for each node in the Pure-Circuit instance.
Every player has two actions: ‘zero’ and ‘one’.

Interpretation:

- If a player plays action ‘zero’ → node value = 0

Reduction

Given: An instance of Pure-Circuit

Goal: Construct a polymatrix game

Idea: Create one player for each node in the Pure-Circuit instance.
Every player has two actions: ‘zero’ and ‘one’.

Interpretation:

- If a player plays action ‘zero’ → node value = 0
- If a player plays action ‘one’ → node value = 1

Reduction

Given: An instance of Pure-Circuit

Goal: Construct a polymatrix game

Idea: Create one player for each node in the Pure-Circuit instance.
Every player has two actions: ‘zero’ and ‘one’.

Interpretation:

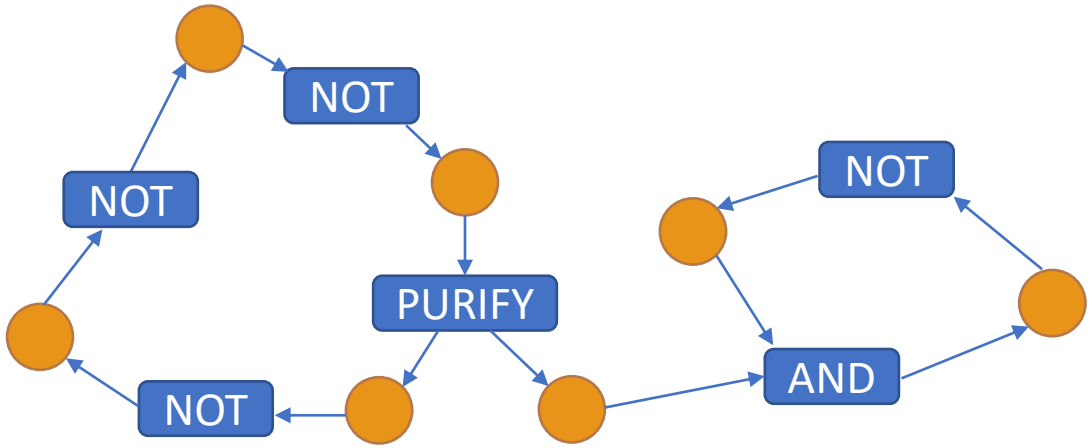
- If a player plays action ‘zero’ → node value = 0
- If a player plays action ‘one’ → node value = 1
- If a player mixes between ‘zero’ and ‘one’ → node value = $\perp$

Reduction

Given: An instance of Pure-Circuit

Goal: Construct a polymatrix game

Idea: Create one player for each node in the Pure-Circuit instance.
Every player has two actions: ‘zero’ and ‘one’.

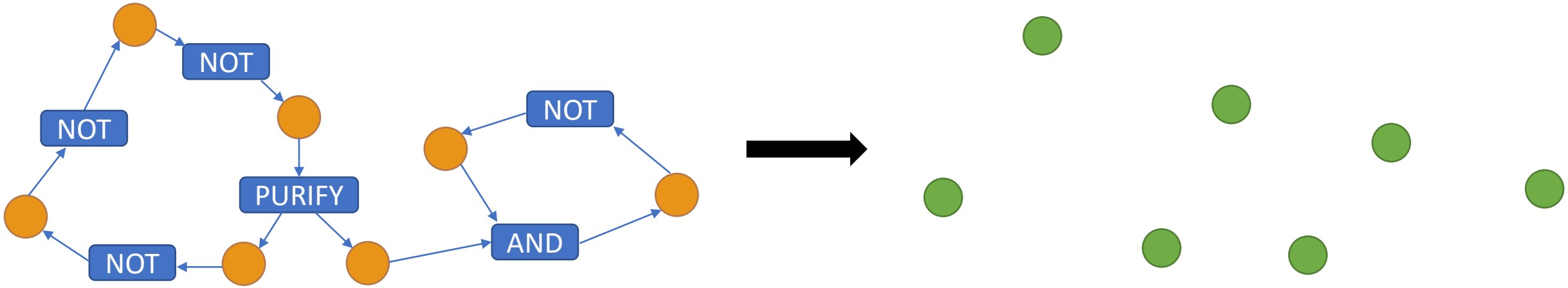


Reduction

Given: An instance of Pure-Circuit

Goal: Construct a polymatrix game

Idea: Create one player for each node in the Pure-Circuit instance.
Every player has two actions: ‘zero’ and ‘one’.

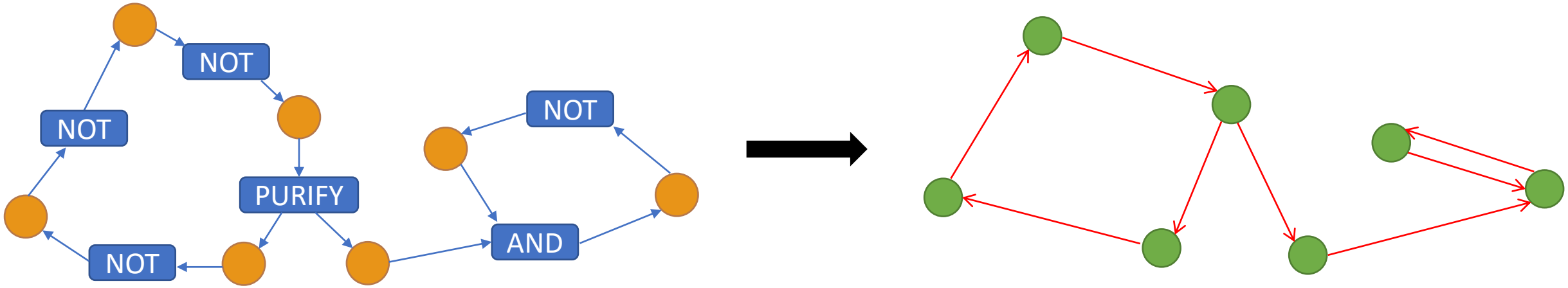


Reduction

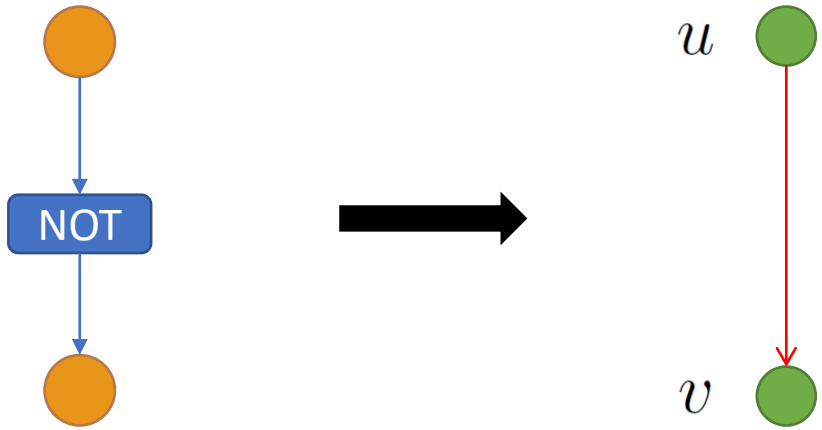
Given: An instance of Pure-Circuit

Goal: Construct a polymatrix game

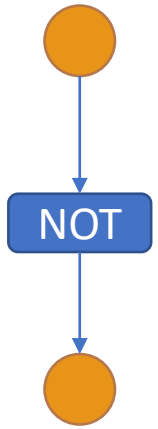
Idea: Create one player for each node in the Pure-Circuit instance.
Every player has two actions: ‘zero’ and ‘one’.

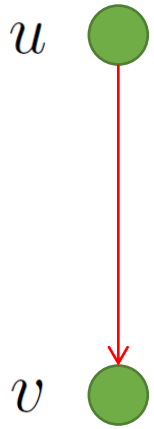


Reduction: NOT-gates



Reduction: NOT-gates

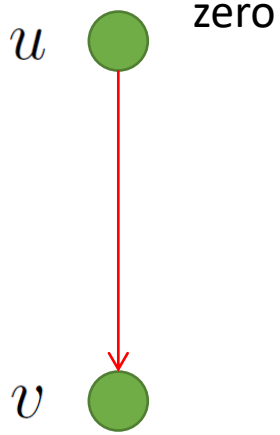




v \ u	zero	one
zero	0 0 0	0 1 0
one	0 1 0	0 0 0

(works for any $\varepsilon < 1$)

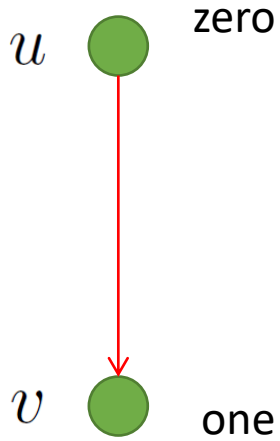
Reduction: NOT-gates



v \ u	zero	one
zero	0 0	0 1
one	0 1	0 0

(works for any $\varepsilon < 1$)

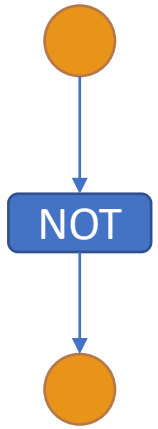
Reduction: NOT-gates

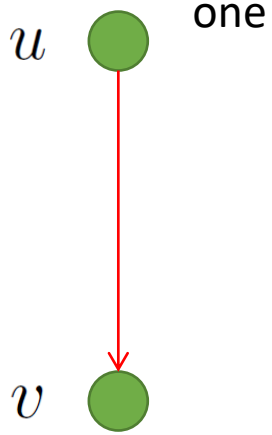


v \ u	zero	one
zero	0 0	0 1
one	0 1	0 0

(works for any $\varepsilon < 1$)

Reduction: NOT-gates

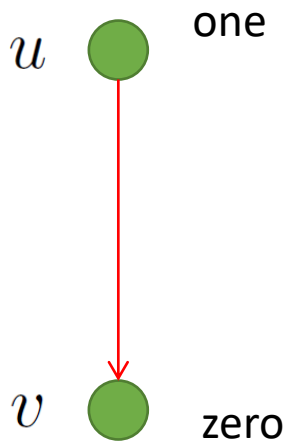




v \ u	zero	one
zero	0 0	0 1
one	0 1	0 0

(works for any $\varepsilon < 1$)

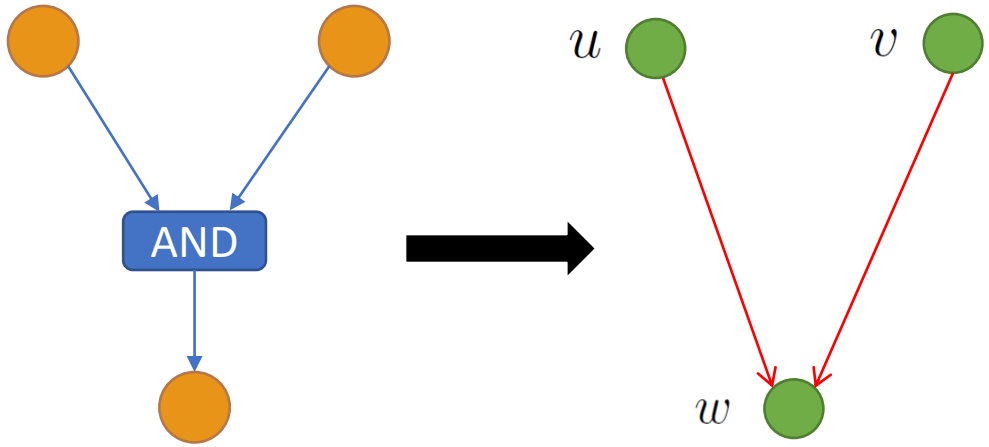
Reduction: NOT-gates



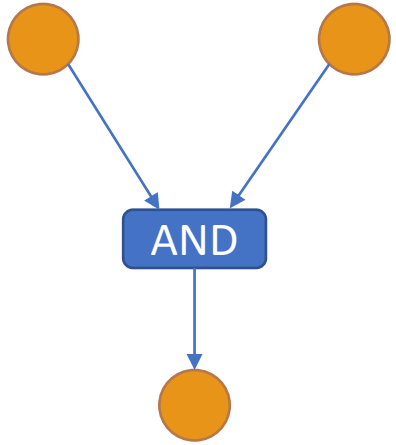
v \ u	zero	one
zero	0 0	0 1
one	0 1	0 0

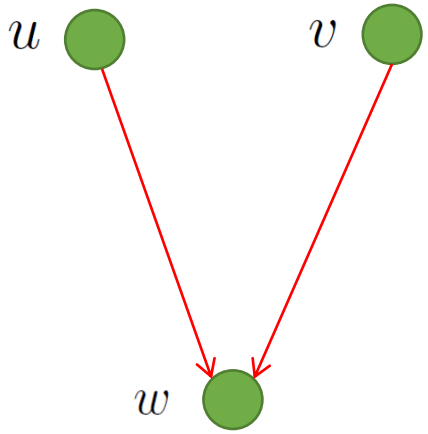
(works for any $\varepsilon < 1$)

Reduction: AND-gates



Reduction: AND-gates



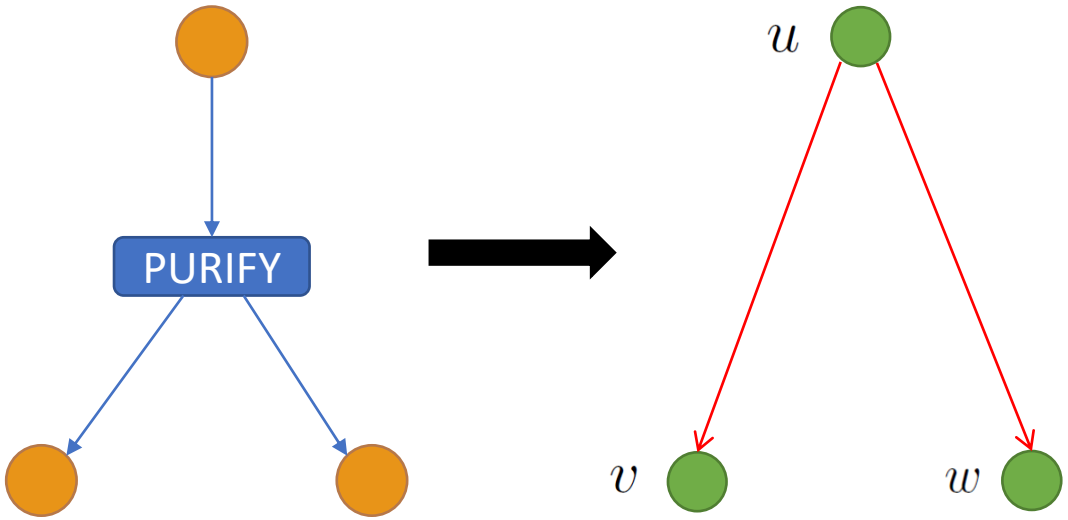


w \ u	zero	one
zero	0, $\frac{1}{2}$	0, 0
one	0, 0	0, $\frac{1}{6}$

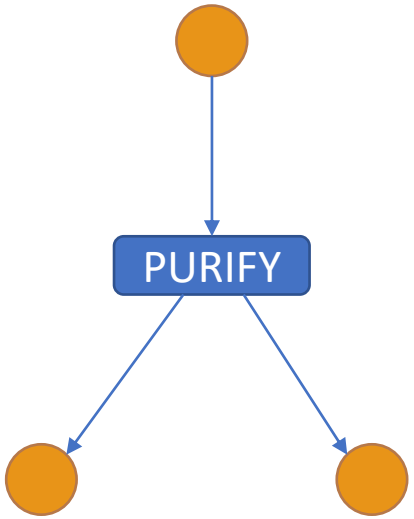
w \ v	zero	one
zero	0, $\frac{1}{2}$	0, 0
one	0, 0	0, $\frac{1}{6}$

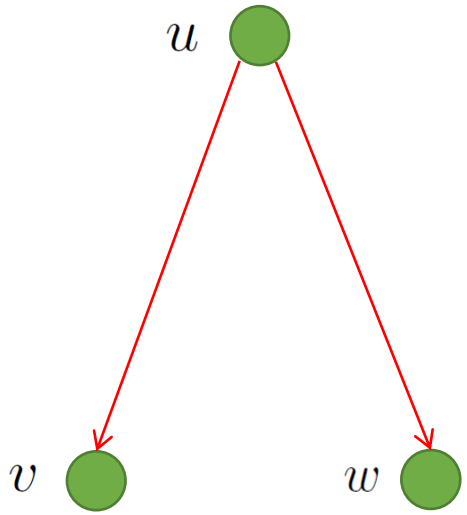
(works for any $\varepsilon < 1/3$)

Reduction: PURIFY-gates



Reduction: PURIFY-gates





v \ u	zero	one
zero	0, $\frac{1}{3}$	0, 0
one	0, 0	0, 1

w \ u	zero	one
zero	0, 1	0, 0
one	0, 0	0, $\frac{1}{3}$

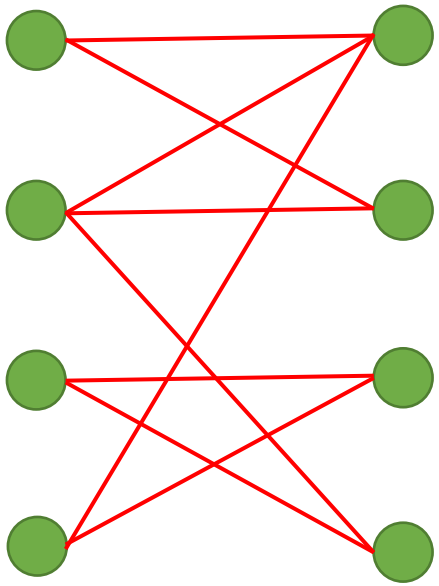
Focus on PPAD

I. How do we prove PPAD-membership?
   1. Sperner's Lemma
   2. The END-OF-LINE Problem
   3. Example: NASH

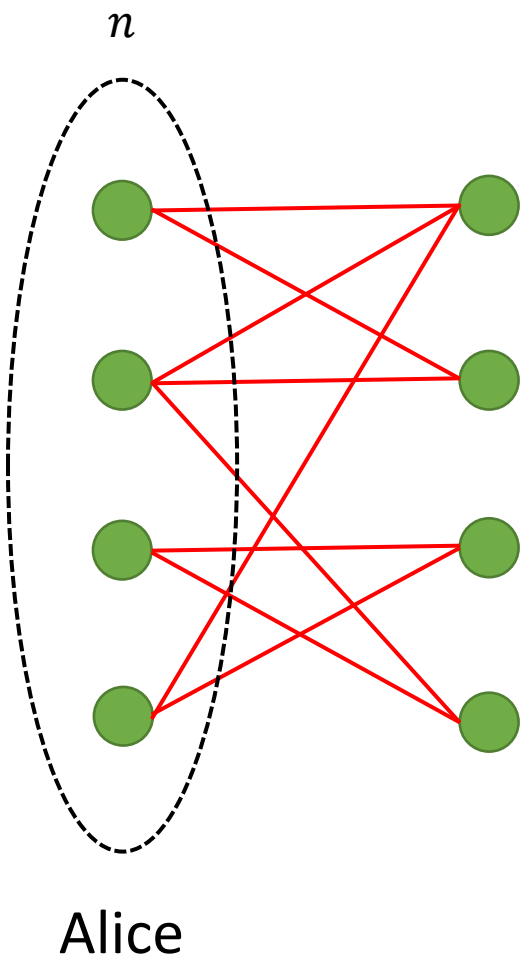
II. How do we prove PPAD-hardness?
   1. Refresher: Reductions
   2. The PURE-CIRCUIT Problem
   3. Example: Polymatrix Games
   4. **Bonus: From Polymatrix Games to Bimatrix Games**

From Polymatrix to Bimatrix Games

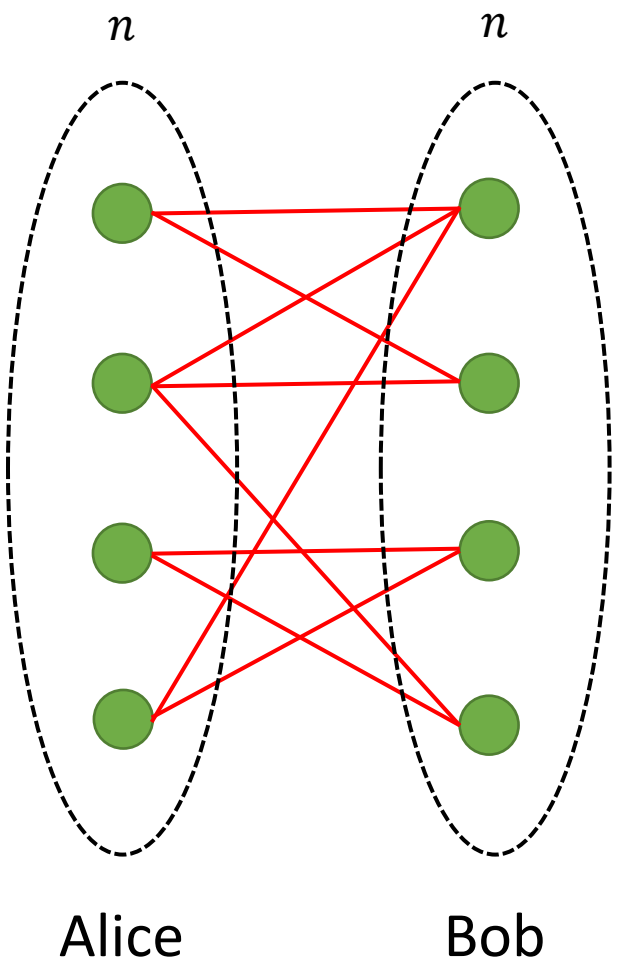
From Polymatrix to Bimatrix Games



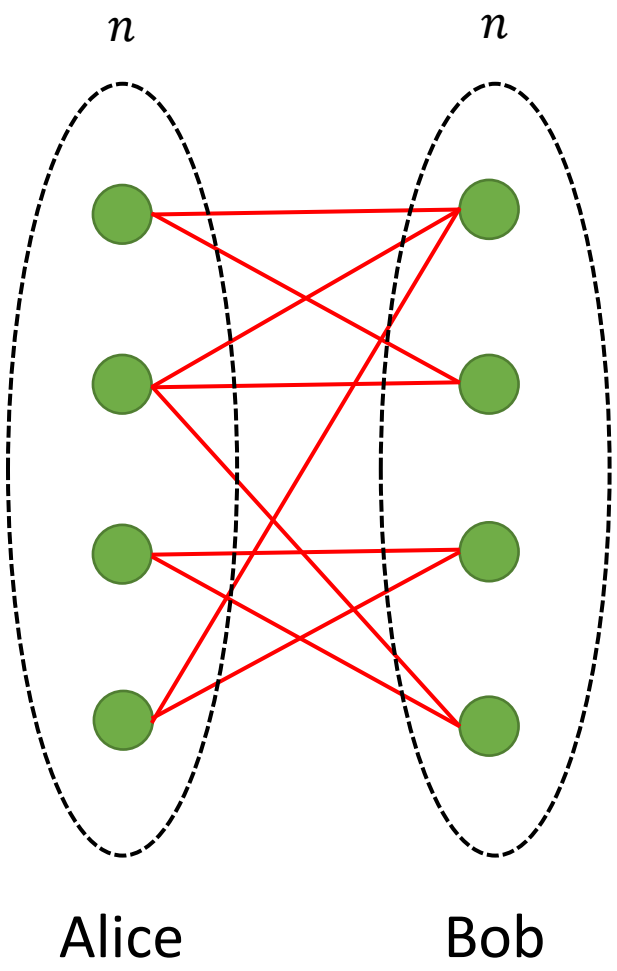
From Polymatrix to Bimatrix Games



From Polymatrix to Bimatrix Games

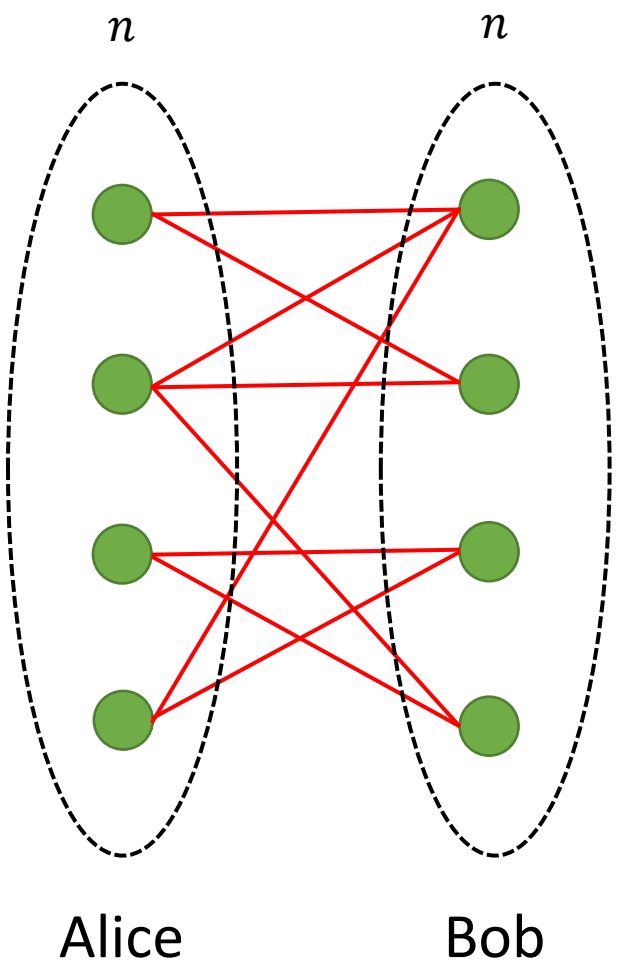


From Polymatrix to Bimatrix Games



Bimatrix game:

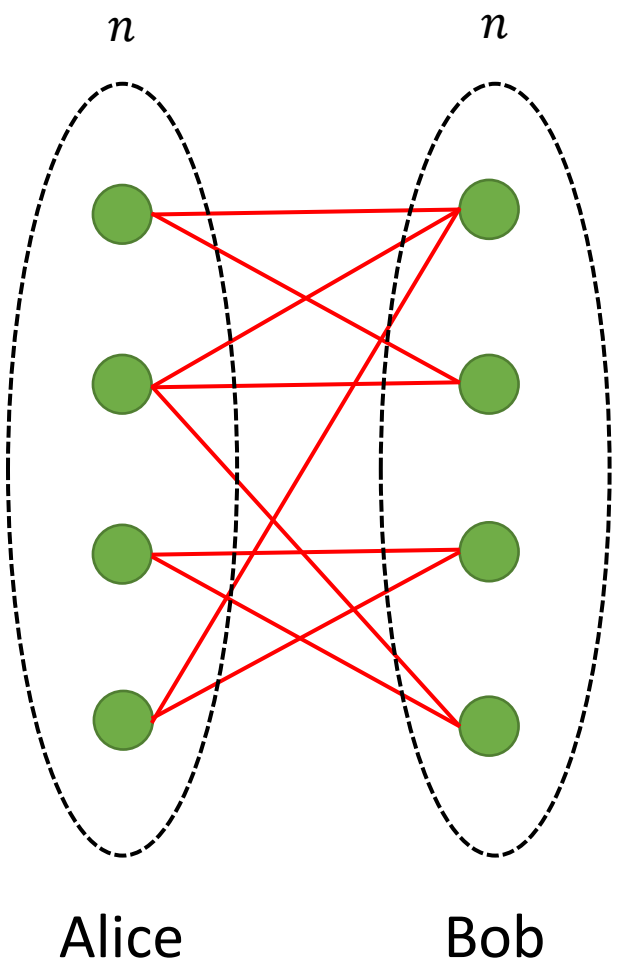
From Polymatrix to Bimatrix Games



Bimatrix game:

- 2 players (Alice and Bob)

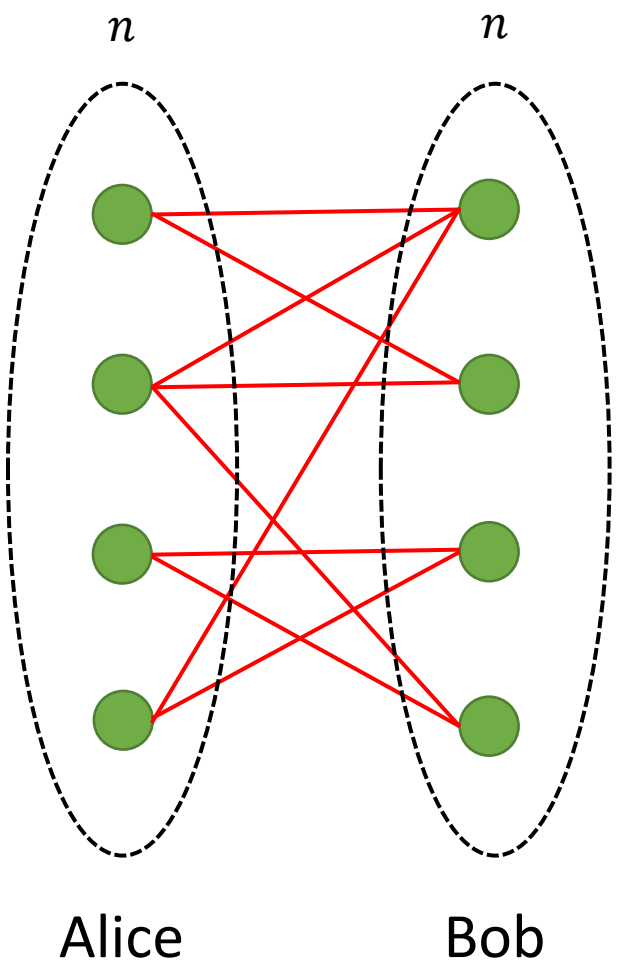
From Polymatrix to Bimatrix Games



Bimatrix game:

- 2 players (Alice and Bob)
- $2n$ actions each

From Polymatrix to Bimatrix Games



Bimatrix game:

- 2 players (Alice and Bob)
- $2n$ actions each

$\varepsilon/p(n)$-equilibrium of bimatrix game
yields ε-equilibrium of polymatrix game

Thank you!