#P
Complexity of the permanent
An interactive proof for $P^{\#P}$

Dmitry Itsykson

Saint-Petersburg State University

Plan

- #P, reductions, complete instances
- The Permanent
- An Interactive proof for $P^{\#P}$ with prover from $P^{\#P}$
- Permanent is #P-complete

#P: computation that counts

- #SAT: Given a boolean expression, compute the number of different assignments that satisfy it
- #Hamilton Path: compute the number of Hamilton paths in given graph
- #Clique: compute the number of cliques of size k or larger

#P: definition

$Q \subseteq X \times Y -$ binary relation

1) $\forall\ (x, y) \in Q\ \ |y| < |x|^k : Q$ is polynomially balanced

2) polynomial-time decidable

Counting problem associated with Q:

"Given x, how many $y: \ (x, y) \in Q$?"

#P: class of all counting problems associated with polynomially balanced polynomial-time decidable relations

Reductions between counting problems

- Reduction from A to B:

$R: A \to B$ polynomial-time computable function

$S: A \times \{0,1,2\ldots\} \to \{0,1,2,\ldots\}$ polynomial-time computable function

If x is instance of A and N is the answer for the instance R(x) of B, then S(x,N) is the answer for instance x of A.

#SAT is #P-complete

Theorem #SAT, #3-SAT are #P-complete

Proof (sketch) Reduction from Cook's theorem preserves the number of solutions. I.e. function R is from Cook's theorem, function S=Id.

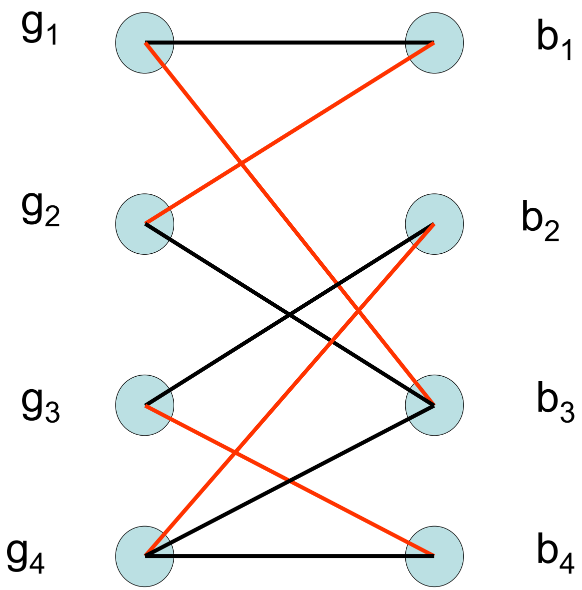
Bipartite graphs and perfect matching

$H = (V = (G, B), E)$ bipartite graph;

$G = \{g_1, g_2, ..., g_n\}$ girls

$B = \{b_1, b_2, ..., b_n\}$ boys

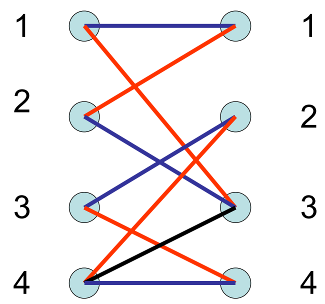
$E \subseteq G \times B$ love edges



Perfect matching: girl-boy love pairs: each boy has exactly one girl in the pair. Each girl has exactly one boy in the pair.

Matching vs. permanent

- Consider the counting problem: compute number of perfect matching in bipartite graph.



$$H=(V=(G,B),E); G=\{g_1,g_2,...,g_n\}; B=\{b_1,b_2,...,b_n\}; E\subseteq G\times B;$$

A is adjacency matrix: $A_{ij}=1 \Leftrightarrow (g_i,b_j)\in E$

$$\det A=\sum_{\pi\in S_n}(-1)^{\sigma(\pi)}\prod_{i=1}^{n}A_{i,\pi(i)} \text{ determinant}$$

$$\operatorname{perm} A=\sum_{\pi\in S_n}\prod_{i=1}^{n}A_{i,\pi(i)} \text{ permanent = number of matching}$$

$$A=\begin{bmatrix}1&0&1&0\\1&0&1&0\\0&1&0&1\\0&1&1&1\end{bmatrix}$$

Corollary: 0-1 Permanent is in #P

Problem to the audience

I will prove, that to compute the permanent is at least NP-hard, therefore to compute the number of perfect matchings is hard problem.

Problem: how to compute the parity of the number of perfect matching in polynomial time?

Solution: just to compute the determinant mod 2.

Motivation

- We prove that permanent is #P-complete later
- $P^{\#P} = P^{Permanent} \subseteq$ PSPACE. Shamir's theorem (IP=PSPACE) states, that every language from PSPACE has Interactive proof with prover from PSPACE.
- We will prove that every language from $P^{\#P}$ has Interactive proof with prover from $P^{\#P}$

Facts

- 0/1 Permanent is #P-complete
- Integer Permanent modulo N is in #P if N is bounded by polynomial on size of the matrix.

We prove this statements later.

An Interactive proof for $P^{\#P}$

Theorem. There exists interactive proof for language $\mathbf{P^{\#P}}$ (with permanent as an oracle) with prover from $\mathbf{P^{\#P}}$.

Proof. Consider language L from $\mathbf{P^{\#P}}$. M is polynomial time Turing Machine with permanent as an oracle, deciding L.

The verifier simulates M and uses Interactive Protocol for permanent computing.

Interactive protocol for Permanent

- *The Verifier asks to compute* **perm** *A of 0/1 matrix A* n×n , prover's *answer is b*
- $p_1, p_2, \ldots, p_n$ are large enough different primes.
- p_i<poly(n).

$$0 \le \text{perm}\, A, b < n! < p_1 p_2 \ldots p_n$$

The Verifier wants to verify:

$$\text{perm } A \equiv b \pmod{p_1}$$

$$\text{perm } A \equiv b \pmod{p_2}$$

$$\ldots$$

$$\text{perm } A \equiv b \pmod{p_n}$$

$$\text{perm } A - b \,\vdots\, p_1 p_2 \ldots p_n \Rightarrow \text{perm}\, A = b$$

Decomposition of the Permanent

$$perm\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} = a_{11} \times perm\begin{pmatrix} a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots \\ a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

$$+a_{12} \times perm\begin{pmatrix} a_{21} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} + \ldots + a_{1n} \times perm\begin{pmatrix} a_{21} & \cdots & a_{2(n-1)} \\ \cdots & \cdots & \cdots \\ a_{n1} & \cdots & a_{n(n-1)} \end{pmatrix}$$

$$perm\,A = a_{11} \times perm A_1 + a_{12} \times perm A_2 + \ldots + a_{1n} \times perm A_n$$

Interactive protocol for Permanent

- *p is enough big prime number.* ***F****=Z_p is the finite field. All evaluations are in* ***F.***
- *The Verifier asks to compute* **perm** *A_1,* **perm** *A_2 ,…,* **perm** *A_n; the Prover answer: $b_1, b_2, \ldots, b_n$.*
- *The Verifier verifies:*
  $b = a_{11}b_1 + a_{12}b_2 + \ldots + a_{1n}b_n$

If **perm** *$A \neq b$, then exists i:* **perm** *$A_i \neq b_i$*

Interactive protocol for Permanent

The Verifiers has to verify the following list S of pairs: $S=\{(A_1, b_1), (A_2, b_2)\dots, (A_n, b_n)\}$

- The Verifier takes (C,d) and (E,f) from S and asks Prover to compute polynomial: **perm** $(Cx+E(1-x))$ (this polynomial is of degree n and Prover from $P^{\#P}$ is able to compute its coefficients using interpolation);

  The Prover answers the polynomial $q(x)$.
- The Verifier verifies that $d=q(1)$ and $f=q(0)$, *(therefore incorrectness of pair* (C,d)*(or* (E,f)*) implies incorrectness* $q(x)$*)*

Interactive protocol for Permanent

- Take y from **F** at random
- Replace *(C,d)* and *(E,f)* by (Cy+E(1-y),q(y))
- If **perm** *(Cx+E(1-x)) is not q(x) then*

$$\Pr_y\{\text{perm }(Cy+E(1-y))=q(y)\}\leq\frac{n}{|\mathrm{F}|}$$

- *Repeat this (n-1) times and S will contain only one pair (A',b'). A' is (n-1) ×(n-1) and (if initial permanent is incorrect):*

$$\Pr\{\text{perm }A'=b'\,|\,\text{perm }A\neq b\}\leq\frac{n^2}{|\mathrm{F}|}$$

Interactive protocol for Permanent

- Repeat this procedure (n-1) times:

A' is matrix (n-1)×(n-1)

A'' is matrix (n-2)×(n-2)

…

$A^{(n-1)}$ is matrix 1×1

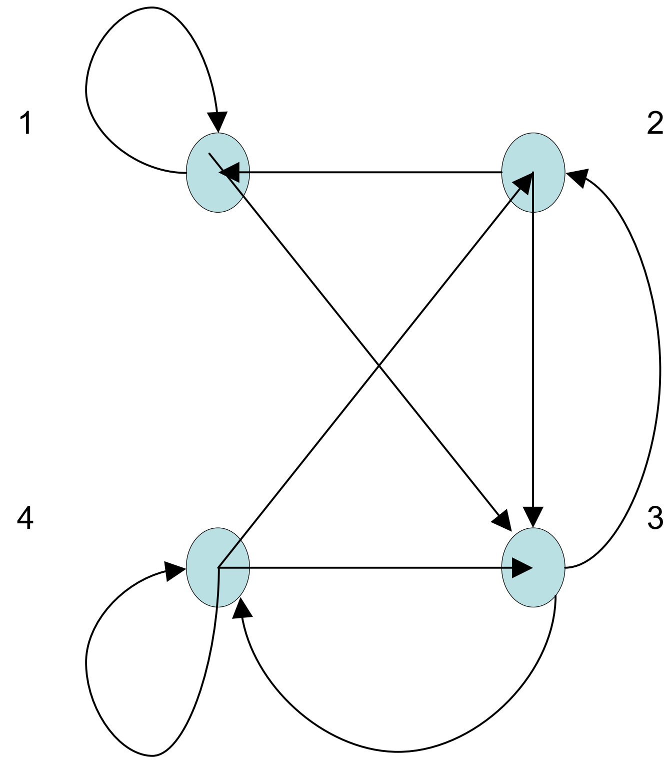
$$\Pr\{\text{perm } A^{(n-1)} = b^{(n-1)} \mid \text{perm } A \neq b\} \leq \frac{n^3}{|\mathrm{F}|}$$

So we are to choose $p=|F|>n^4$

Last part of the talk: 0/1 Permanent is #P-complete

Matrix-graph corespondence

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$



(i,j) is edge iff A_{ij}=1

Cycle form of Permutations

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 4 & 7 & 2 & 1 & 6 & 5 & 8 \end{pmatrix}$$

$$1 \xrightarrow[\pi]{} 3 \xrightarrow[\pi]{} 7 \xrightarrow[\pi]{} 5 \xrightarrow[\pi]{} 1$$

$$2 \xrightarrow[\pi]{} 4 \xrightarrow[\pi]{} 2$$

$$6 \xrightarrow[\pi]{} 6$$

$$8 \xrightarrow[\pi]{} 8$$

$$\pi = (1,3,7,5)(2,4)(6)(8)$$

Cycle covering vs. permanent

Consider 0-1 $n \times n$ matrix A. Define the directed graph

$G(V = [1..n], E)$ based on A: $(i, j) \in E \Leftrightarrow A_{ij} = 1$

Cycle covering: $\{C_1, C_2, \ldots, C_k\}$ – set of disjoint cycles

$\forall v \in V \; \exists i : v \in C_i$

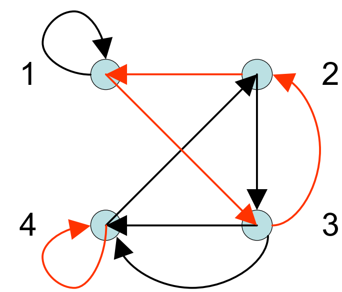
$$\text{perm } A = \sum_{\pi \in S_n} \prod_{i=1}^{n} A_{i,\pi(i)}$$ – number of cycle coverings

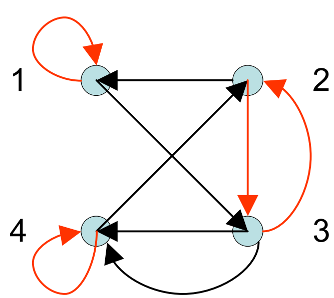
$\pi \in S_n, \pi = (i_1, i_2, \ldots, i_{k_1})(i_{k_1+1}, i_{k_1+2}, \ldots, i_{k_2})\ldots(i_{k_{l-1}+1}, i_{k_{l-1}+2}, \ldots, i_n).$

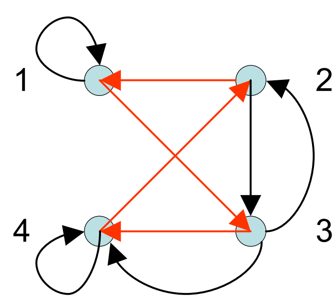
$i_1 \to i_2 \to \ldots \to i_{k_1} \to i_1$ – a cycle

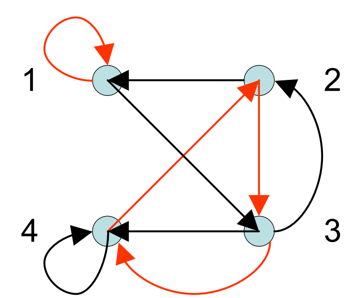
$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

$perm\, A = 4$





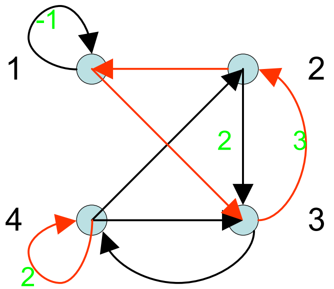




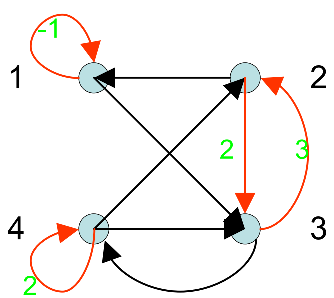
Weighted cycle covering

$$A = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 1 & 0 & 2 & 0 \\ 0 & 3 & 0 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix}$$

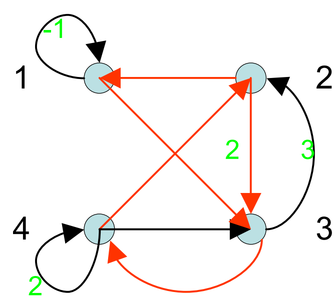
$$perm\, A = -6$$



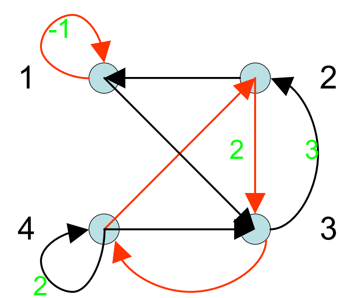
weight=6



weight=-12



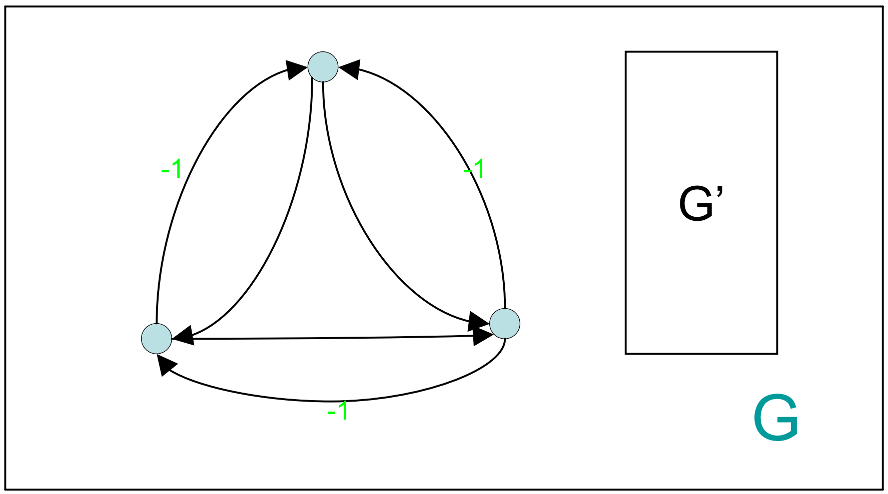
weight=2

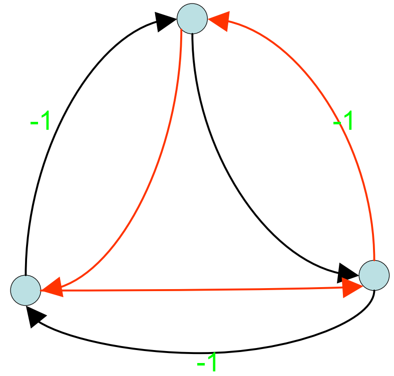


weight=-2

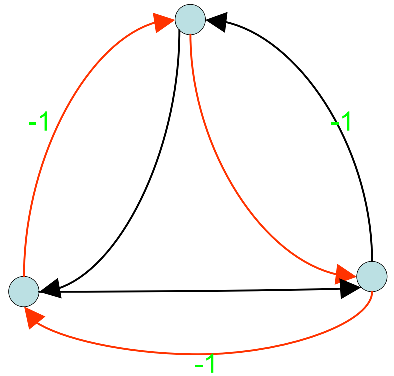
Unlabeled edges have weight 1

Warm-up (example)





weight=-1



weight=1

Total weight equals 0

Permanent of graph G equals 0.

Permanent is #P-complete

Theorem (Valiant's Theorem) 0/1 Permanent is #P-complete

Plan of the proof:

1) **Reduction from #3-SAT to Weighted Cycle Covering (Permanent under integers)**
2) **Reduction from Weighted Cycle Covering to Cycle Covering (0/1 Permanent)**

Part 1: #3-SAT to Weighted Cycle Covering

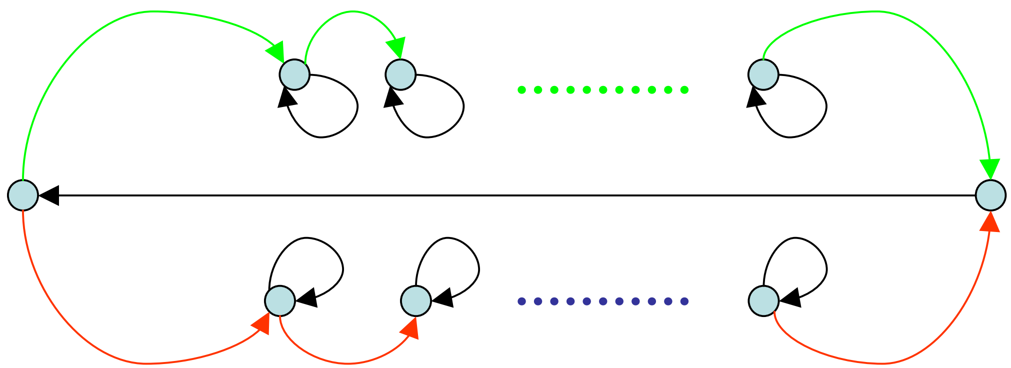
Proof: Given a boolean formula *φ* in 3-CNF witn *n* variables and *m* clauses we construct a graph *G* with weighted cycle covering (or integer matrix *A* with permanent) $4^{3m}(\# \varphi)$. *# φ stands for the number of satisfying assignments of φ.*

To construct G from *φ, we use three kinds of gadgets: two syntax (variable-gadget and clause-gadget) and one semantic (xor-gadget).*

The Variable-gadget

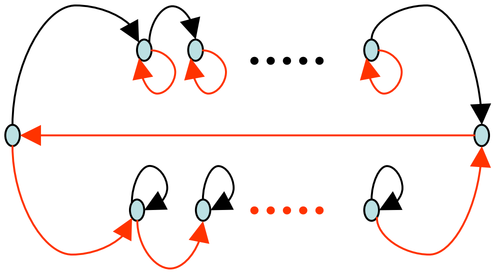
Variable x:

False edges: one per clause, containing ¬x.

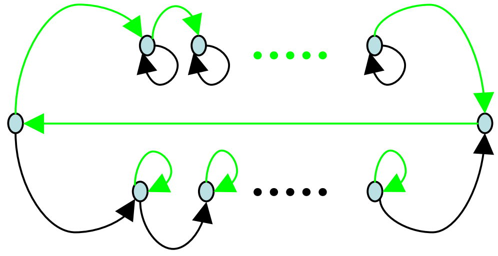


True edges: one per clause, containing x.

True-value cycle covering:

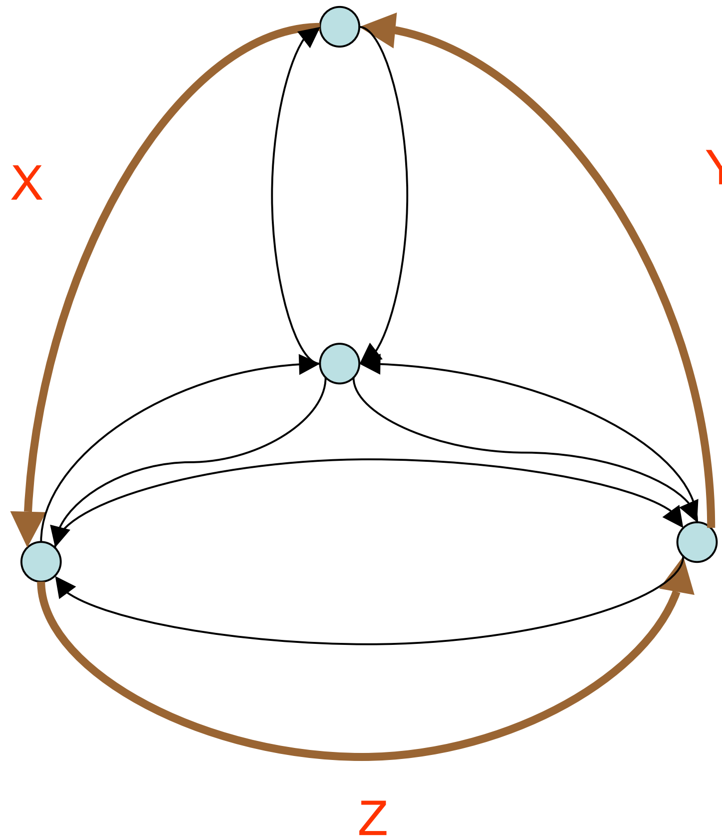


False-value cycle covering:



The Clause-gadget

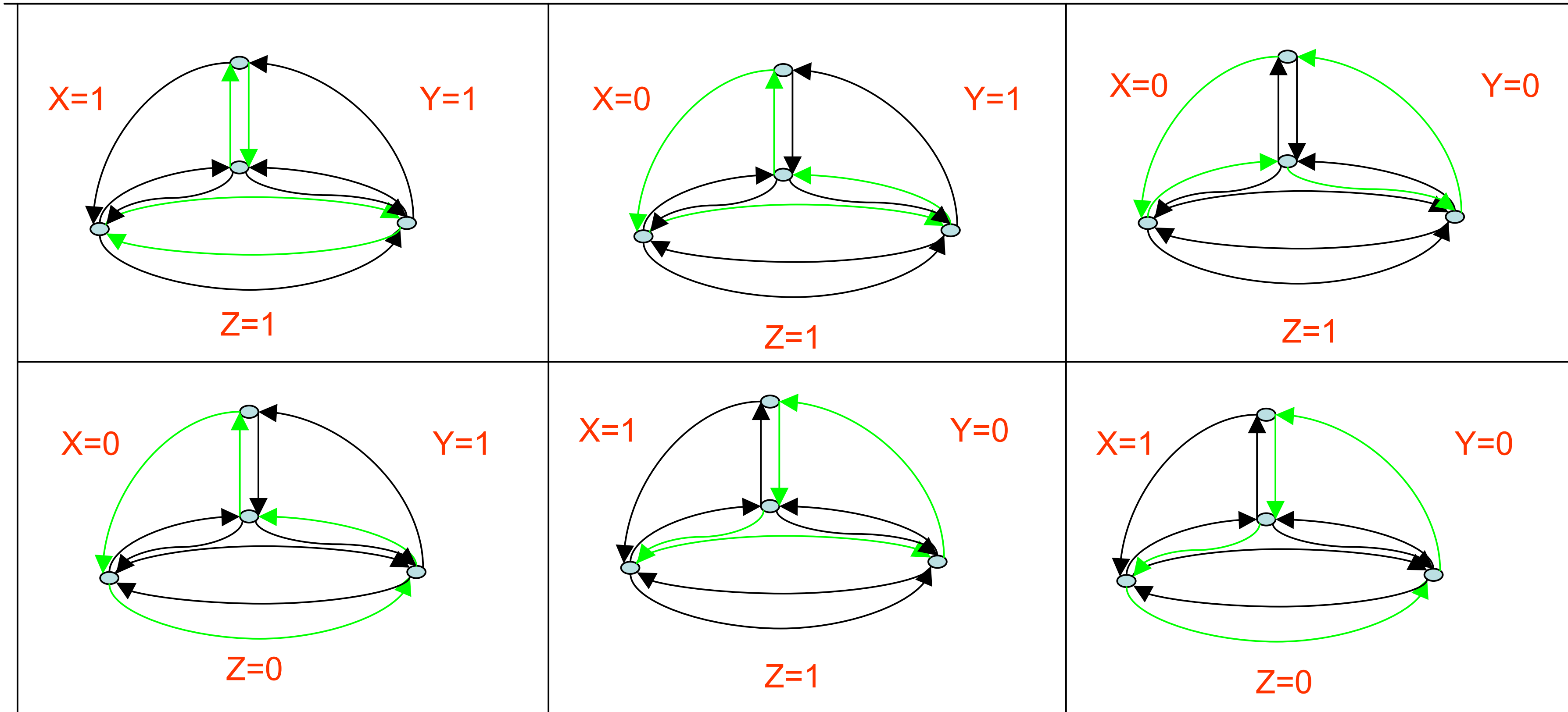
Clause
$(X \vee Y \vee Z)$

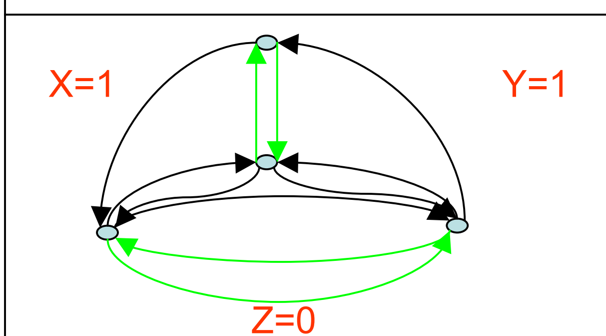


Clause-gadget has no cycle covering traversing all external (brown) edges.

Brown edges: external edges

Clause-gadget cycle covering

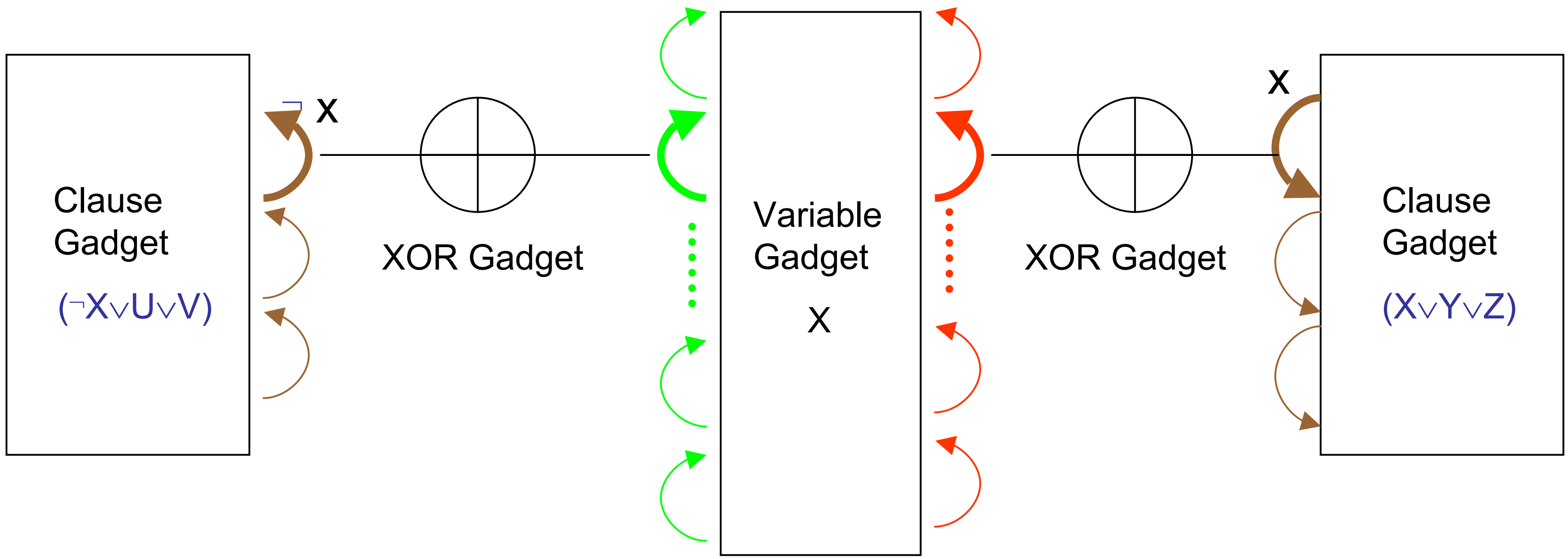




Cycle covering corresponds to satisfying assignment of the clause.

The value of variable: “cycle covering doesn’t traverse my external edge”

General construction



XOR-gadget: exact one of two edges is included in cycle covering

The XOR-gadget

Fact (can be easily checked):

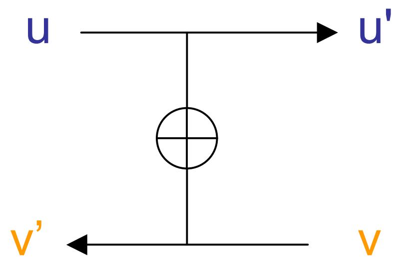
The following cycle covers have total weight of 0:

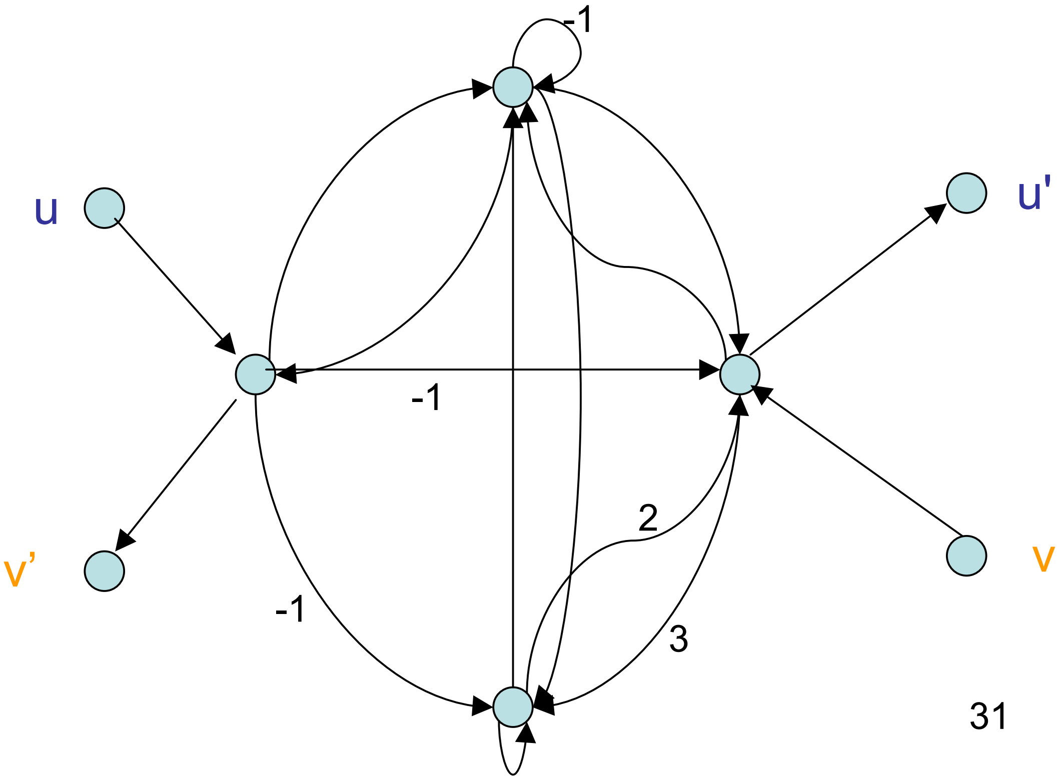
1) Those that do not enter or leave the gadget
2) Those that enter at u and leave at v'
3) Those that enter at v and leave at u'

Only cycle cover that have nonzero (weight=4) contribution:

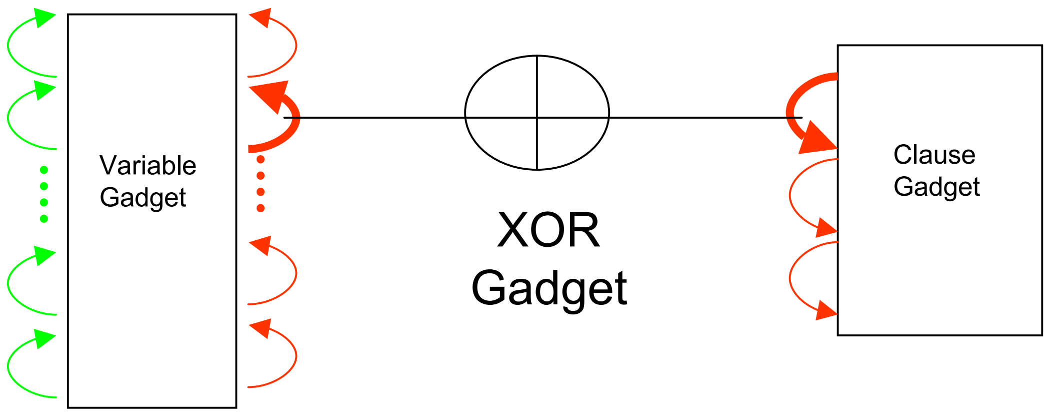
a) enter at u and leave at u'

b) enter at v and leave at v'





Total:



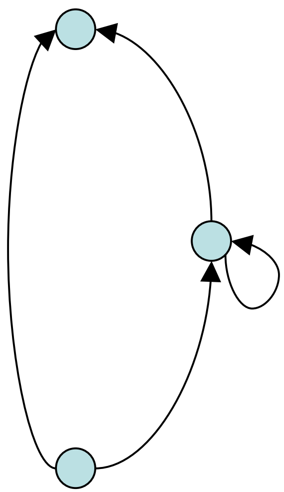
We have some correspondence between truth assignments and nonzero cycle coverings.

Each nonzero cycle covering has the weight 4^{3m}: *each XOR-gadget give weight 4 and we have 3m XOR gadgets (3 for each clause).*

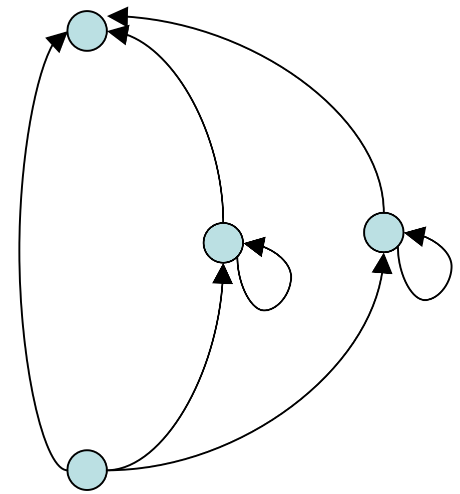
Part 2: from weighted cycle covering to unweighted

- Positive weights simulating
- MOD *N* Permanent
- Weight -1 simulating

Positive weights simulating



Weight 2 simulating



Weight 3 simulating

Corollary: permanent mod N is in #P if N<poly("size of matrix")

MOD N Permanent

- All evaluations modulo N
- If N>**perm** A, *then*

  ((**perm** A) *mod* N) = **perm** A

Weight -1 simulating

Consider k: *perm* $A<2^k$

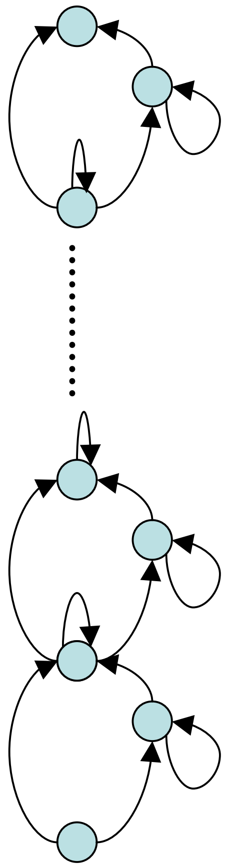
$(k=6m+n+1$:

$2^{6m+n+1}>4^{3m}2^n)$.

$N=2^k+1$

Evaluations modulo N.

$-1 \bmod N = 2^k$



Weight 2^k simulating

Conclusion

- We prove that the 0/1 permanent is #P-complete
- We give Interactive protocol for the language from $P^{\#P}$ with prover from $P^{\#P}$

Any questions?

References

- C. Papadimitriou, Computational Complexity, *Addison Wesley, 1994, chapter 18*
- S. Arora, Computational Complexity: Modern Approach, Chapter 8
- E.A. Hirsch, Lecture notes.