The Proof: AND

- The approximate AND of crude circuits $\mathrm{CC}(\mathcal{X})$ and $\mathrm{CC}(\mathcal{Y})$ is

$$\mathrm{CC}(\mathrm{pluck}(\{X_i \cup Y_j : X_i \in \mathcal{X}, Y_j \in \mathcal{Y}, |X_i \cup Y_j| \leq \ell\})).$$

  – Note that if $\mathrm{CC}(\mathcal{Z})$ is true, then $\mathrm{CC}(\mathrm{pluck}(\mathcal{Z}))$ must be true.

- We now count the number of errors this approximate AND makes on the positive and negative examples.

The Proof: AND (concluded)

- The approximate AND *introduces* a **false positive** if a negative example makes either CC($\mathcal{X}$) or CC($\mathcal{Y}$) return false but makes the approximate AND return true.
- The approximate AND *introduces* a **false negative** if a positive example makes both CC($\mathcal{X}$) and CC($\mathcal{Y}$) return true but makes the approximate AND return false.
- How many false positives and false negatives are introduced by the approximate AND?

The Number of False Positives

Lemma 89 *The approximate* AND *introduces at most* $M^2 2^{-p}(k-1)^n$ *false positives.*

- $\text{CC}(\{X_i \cup Y_j : X_i \in \mathcal{X}, Y_j \in \mathcal{Y}\})$ introduces no false positives.
  - If $X_i \cup Y_j$ is a clique, both X_i and Y_j must be cliques, making both $\text{CC}(\mathcal{X})$ and $\text{CC}(\mathcal{Y})$ return true.
- $\text{CC}(\{X_i \cup Y_j : X_i \in \mathcal{X}, Y_j \in \mathcal{Y}, |X_i \cup Y_j| \leq \ell\})$ introduces no false positives as we are testing fewer sets for cliques.

Proof of Lemma 89 (concluded)

- $|\{X_i \cup Y_j : X_i \in \mathcal{X}, Y_j \in \mathcal{Y}, |X_i \cup Y_j| \leq \ell\}| \leq M^2$.
- Each plucking reduces the number of sets by $p - 1$.
- So $\text{pluck}(\{X_i \cup Y_j : X_i \in \mathcal{X}, Y_j \in \mathcal{Y}, |X_i \cup Y_j| \leq \ell\})$ involves $\leq M^2/(p-1)$ pluckings.
- Each plucking introduces at most $2^{-p}(k-1)^n$ false positives by the proof of Lemma 87 (p. 703).
- The desired upper bound is

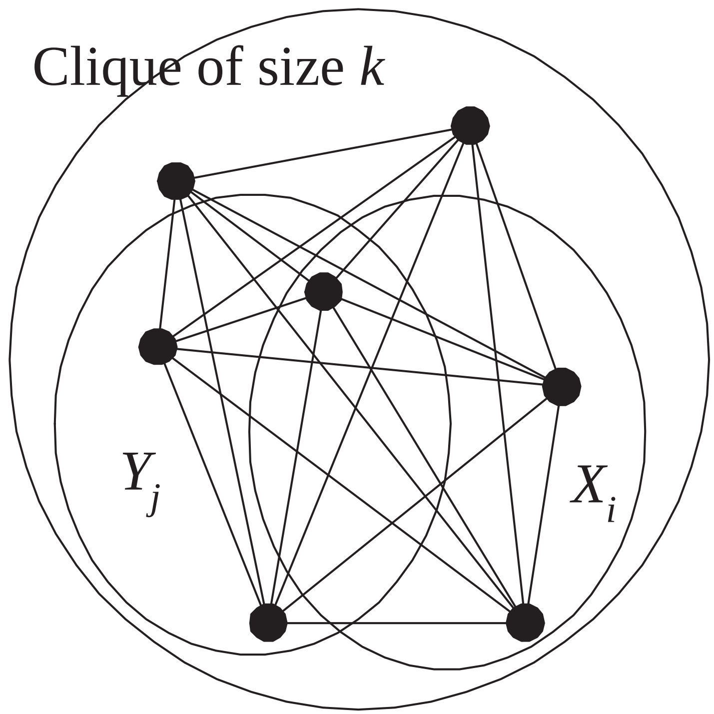
$$[\, M^2/(p-1)\,]\, 2^{-p}(k-1)^n \leq M^2 2^{-p}(k-1)^n.$$

The Number of False Negatives

Lemma 90 *The approximate* AND *introduces at most* $M^2\binom{n-\ell-1}{k-\ell-1}$ *false negatives.*

- We follow the same three-step proof as before.
- $\text{CC}(\{X_i \cup Y_j : X_i \in \mathcal{X}, Y_j \in \mathcal{Y}\})$ introduces no false negatives.
  - Suppose both $\text{CC}(\mathcal{X})$ and $\text{CC}(\mathcal{Y})$ accept a positive example with a clique of size k.
  - This clique must contain an $X_i \in \mathcal{X}$ and a $Y_j \in \mathcal{Y}$.
    - This is why both $\text{CC}(\mathcal{X})$ and $\text{CC}(\mathcal{Y})$ return true.
  - As the clique contains $X_i \cup Y_j$, the new circuit returns true.

Proof of Lemma 90 (continued)



Proof of Lemma 90 (concluded)

- $\mathrm{CC}(\{X_i \cup Y_j : X_i \in \mathcal{X}, Y_j \in \mathcal{Y}, |X_i \cup Y_j| \leq \ell\})$ introduces $\leq M^2 \binom{n-\ell-1}{k-\ell-1}$ false negatives.
  - Deletion of set $Z = X_i \cup Y_j$ larger than ℓ introduces false negatives only if the clique contains Z.
  - There are $\binom{n-|Z|}{k-|Z|}$ such cliques.
    * It is the number of positive examples whose clique contains Z.
  - $\binom{n-|Z|}{k-|Z|} \leq \binom{n-\ell-1}{k-\ell-1}$ as $|Z| > \ell$.
  - There are at most M^2 such Zs.
- Plucking introduces no false negatives.

Two Summarizing Lemmas

From Lemmas 87 (p. 703) and 89 (p. 712), we have:

Lemma 91 *Each approximation step introduces at most* $M^2 2^{-p}(k-1)^n$ *false positives.*

From Lemmas 88 (p. 708) and 90 (p. 714), we have:

Lemma 92 *Each approximation step introduces at most* $M^2 \binom{n-\ell-1}{k-\ell-1}$ *false negatives.*

The Proof (continued)

- The above two lemmas show that each approximation step introduce “few” false positives and false negatives.
- We next show that the resulting crude circuit has “a lot” of false positives or false negatives.

The Final Crude Circuit

Lemma 93 *Every final crude circuit either is identically false—thus wrong on all positive examples—or outputs true on at least half of the negative examples.*

- Suppose it is not identically false.
- By construction, it accepts at least those graphs that have a clique on some set X of nodes, with $|\, X \,| \leq \ell$, which at $n^{1/8}$ is less than $k = n^{1/4}$.
- The proof of Lemma 87 (p. 703ff) shows that at least half of the colorings assign different colors to nodes in X.
- So half of the negative examples have a clique in X and are accepted.

The Proof (continued)

- Recall the constants on p. 695: $k = n^{1/4}$, $\ell = n^{1/8}$, $p = n^{1/8} \log n$, $M = (p-1)^{\ell} \ell! < n^{(1/3)n^{1/8}}$ for large n.
- Suppose the final crude circuit is identically false.
  - By Lemma 92 (p. 717), each approximation step introduces at most $M^2 \binom{n-\ell-1}{k-\ell-1}$ false negatives.
  - There are $\binom{n}{k}$ positive examples.
  - The original crude circuit for $\text{CLIQUE}_{n,k}$ has at least

$$\frac{\binom{n}{k}}{M^2 \binom{n-\ell-1}{k-\ell-1}} \geq \frac{1}{M^2} \left(\frac{n-\ell}{k} \right)^{\ell} \geq n^{(1/12)n^{1/8}}$$

gates for large n.

The Proof (concluded)

- Suppose the final crude circuit is not identically false.
  - Lemma 93 (p. 719) says that there are at least $(k-1)^n/2$ false positives.
  - By Lemma 91 (p. 717), each approximation step introduces at most $M^2 2^{-p}(k-1)^n$ false positives.
  - The original crude circuit for $\text{CLIQUE}_{n,k}$ has at least

$$\frac{(k-1)^n/2}{M^2 2^{-p}(k-1)^n} = \frac{2^{p-1}}{M^2} \geq n^{(1/3)n^{1/8}}$$

    gates.

P $\neq$ NP Proved?

- Razborov's theorem says that there is a monotone language in NP that has no polynomial monotone circuits.
- If we can prove that all monotone languages in P have polynomial monotone circuits, then P $\neq$ NP.
- But Razborov proved in 1985 that some monotone languages in P have no polynomial monotone circuits!

Computation That Counts

Counting Problems

- Counting problems are concerned with the number of solutions.
  - #SAT: the number of satisfying truth assignments to a boolean formula.
  - #HAMILTONIAN PATH: the number of Hamiltonian paths in a graph.
- They cannot be easier than their decision versions.
  - The decision problem has a solution if and only if the solution count is larger than 0.
- But they can be harder than their decision versions.

Decision and Counting Problems

- FP is the set of polynomial-time computable functions $f : \{0,1\}^* \to \mathbb{Z}$.
  - GCD, LCM, matrix-matrix multiplication, etc.
- If $\#\textsc{sat} \in \text{FP}$, then $\text{P} = \text{NP}$.
  - Given boolean formula ϕ, calculate its number of satisfying truth assignments, k, in polynomial time.
  - Declare "$\phi \in \textsc{sat}$" if and only if $k \geq 1$.
- The validity of the reverse direction is open.

A Counting Problem Harder than Its Decision Version

- Some counting problems are harder than their decision versions.
- CYCLE asks if a directed graph contains a cycle.
- #CYCLE counts the number of cycles in a directed graph.
- CYCLE is in P by a simple greedy algorithm.
- But #CYCLE is hard unless P = NP.

Counting Class #P

A function f is in #P (or $f \in$ #P) if

- There exists a polynomial-time NTM M.
- $M(x)$ has $f(x)$ accepting paths for all inputs x.
- $f(x) =$ number of accepting paths of $M(x)$.

Some #P Problems

- $f(\phi) =$ number of satisfying truth assignments to ϕ.
  - The desired NTM guesses a truth assignment T and accepts ϕ if and only if $T \models \phi$.
  - Hence $f \in$ #P.
  - f is also called #SAT.
- #HAMILTONIAN PATH.
- #3-COLORING.

#P Completeness

- Function f is #P-complete if
  - $f \in$ #P.
  - #P $\subseteq \text{FP}^f$.
    * Every function in #P can be computed in polynomial time with access to a black box or **oracle** for f.
  - Of course, oracle f will be accessed only a polynomial number of times.
  - #P is said to be **polynomial-time Turing-reducible to** f.

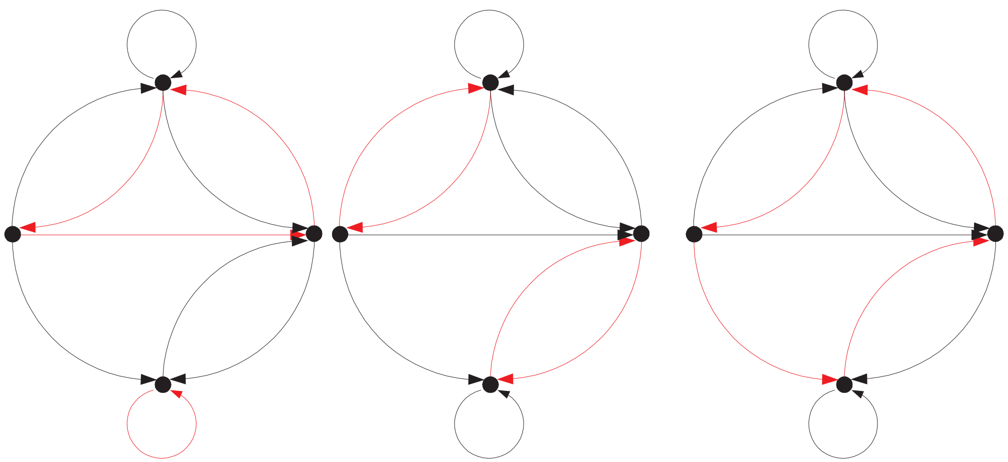
#SAT Is #P-Complete

- First, it is in #P (p. 728).
- Let $f \in$ #P compute the number of accepting paths of M.
- Cook's theorem uses a *parsimonious* reduction from M on input x to an instance ϕ of SAT (p. 277).
  - Hence the number of accepting paths of $M(x)$ equals the number of satisfying truth assignments to ϕ.
- Call the oracle #SAT with ϕ to obtain the desired answer regarding $f(x)$.

Leslie G Valiant (1949–)

CYCLE COVER

- A set of node-disjoint cycles that cover all nodes in a directed graph is called a **cycle cover**.



- There are 3 cycle covers (in red) above.

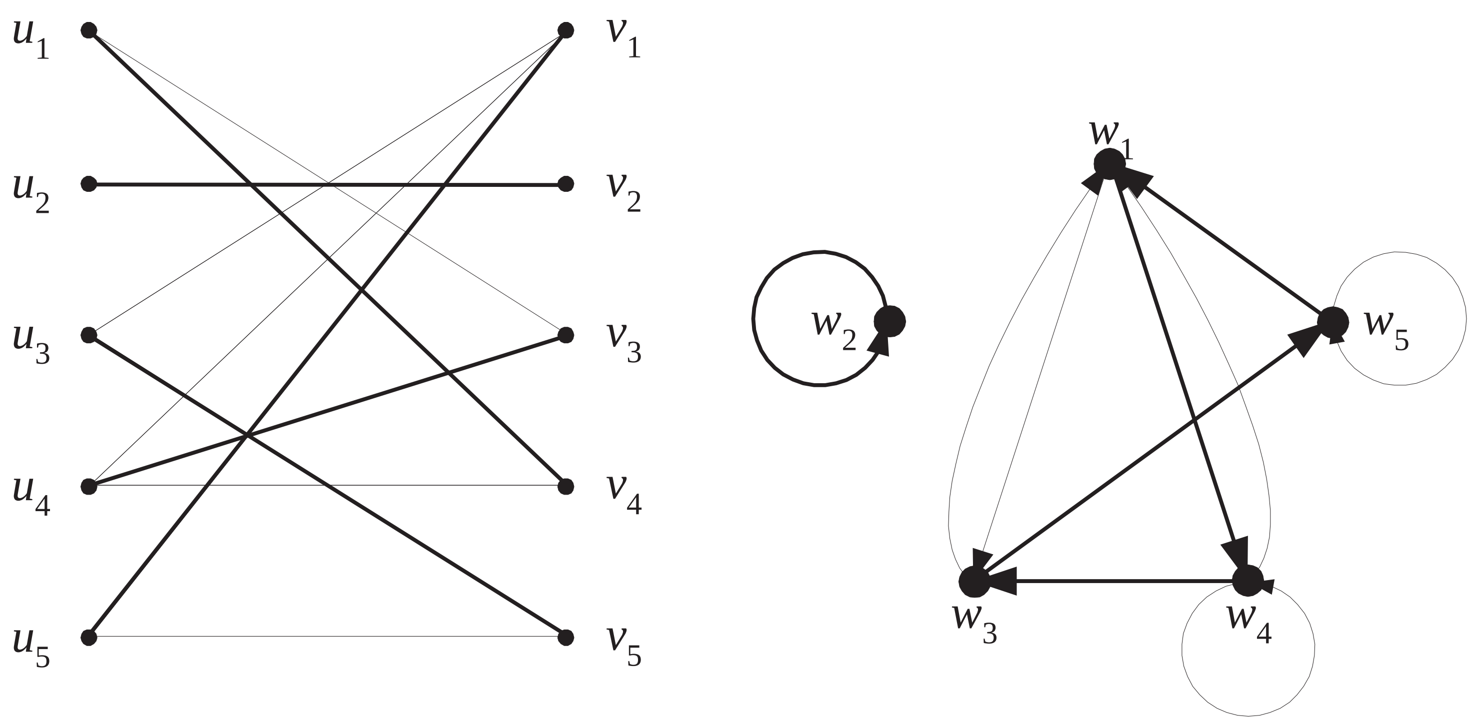
CYCLE COVER and BIPARTITE PERFECT MATCHING

Proposition 94 CYCLE COVER *and* BIPARTITE PERFECT MATCHING *(p. 424) are parsimoniously reducible to each other.*

- A polynomial-time algorithm creates a bipartite graph G' from any directed graph G.
- Moreover, the number cycle covers for G equals the number of bipartite perfect matchings for G'.
- And vice versa.

Corollary 95 CYCLE COVER $\in P$.

Illustration of the Proof



Permanent

- The **permanent** of an $n \times n$ integer matrix A is

$$\operatorname{perm}(A) = \sum_{\pi} \prod_{i=1}^{n} A_{i,\pi(i)}.$$

  - π ranges over all permutations of n elements.
- 0/1 PERMANENT computes the permanent of a 0/1 (binary) matrix.
  - The permanent of a binary matrix is at most $n!$.
- Simpler than determinant (5) on p. 426: no signs.
- But, surprisingly, much harder to compute than determinant!

Permanent and Counting Perfect Matchings

- BIPARTITE PERFECT MATCHING is related to determinant (p. 427).
- #BIPARTITE PERFECT MATCHING is related to permanent.

Proposition 96 0/1 PERMANENT *and* BIPARTITE PERFECT MATCHING *are parsimoniously reducible to each other.*

The Proof

- Given a bipartite graph G, construct an $n \times n$ binary matrix A.
  - The (i, j)th entry A_{ij} is 1 if $(i, j) \in E$ and 0 otherwise.
- Then $\text{perm}(A) =$ number of perfect matchings in G.

Illustration of the Proof Based on p. 734 (Left)

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

- $\text{perm}(A) = 4$.
- The permutation corresponding to the perfect matching on p. 734 is marked.

Permanent and Counting Cycle Covers

Proposition 97 *0/1* PERMANENT *and* CYCLE COVER *are parsimoniously reducible to each other.*

- Let A be the adjacency matrix of the graph on p. 734 (right).
- Then $\text{perm}(A) =$ number of cycle covers.

Three Parsimoniously Equivalent Problems

We summarize Propositions 94 (p. 733) and 96 (p. 736) in the following.

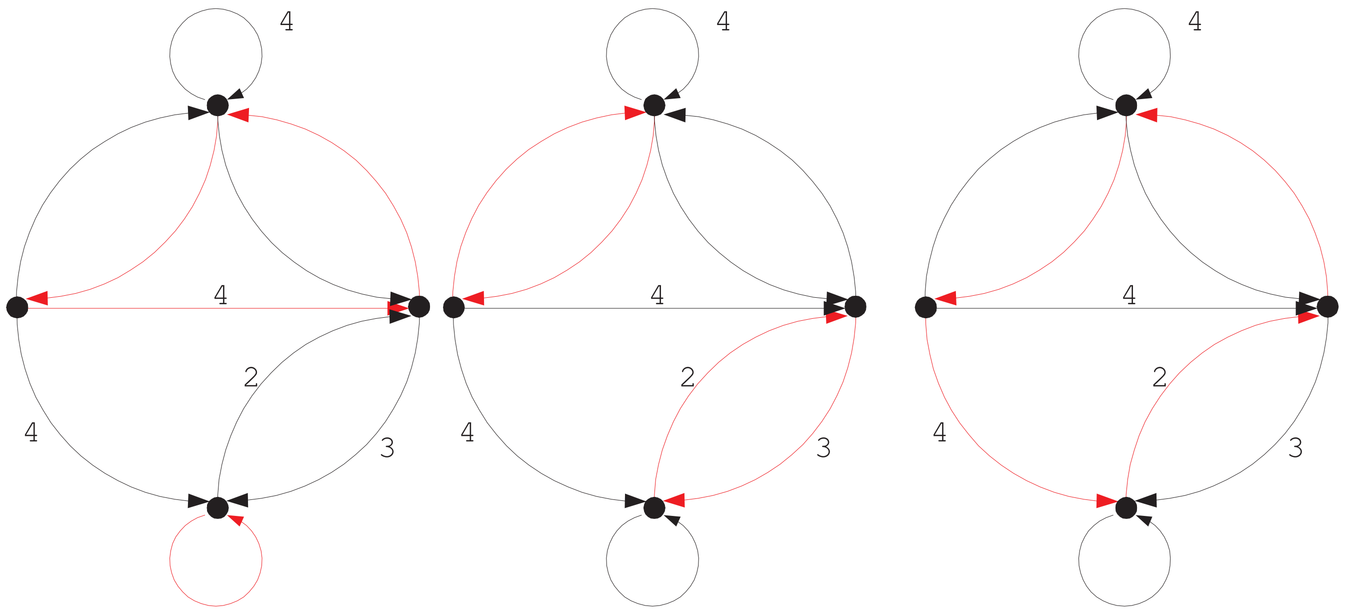
Lemma 98 0/1 PERMANENT, BIPARTITE PERFECT MATCHING, *and* CYCLE COVER *are* **parsimoniously equivalent**.

We will show that the counting versions of all three problems are in fact #P-complete.

WEIGHTED CYCLE COVER

- Consider a directed graph G with integer weights on the edges.
- The weight of a cycle cover is the product of its edge weights.
- The **cycle count** of G is sum of the weights of all cycle covers.
  - Let A be G's adjacency matrix but $A_{ij} = w_i$ if the edge (i, j) has weight w_i.
  - Then $\text{perm}(A) = G$'s cycle count (same proof as Proposition 97 on p. 739).
- #CYCLE COVER is a special case: All weights are 1.

An Example[a]



There are 3 cycle covers, and the cycle count is

$$(4 \cdot 1 \cdot 1) \cdot (1) + (1 \cdot 1) \cdot (2 \cdot 3) + (4 \cdot 2 \cdot 1 \cdot 1) = 18.$$

[a]Each edge has weight 1 unless stated otherwise.

Three #P-Complete Counting Problems

Theorem 99 (Valiant (1979)) 0/1 PERMANENT, #BIPARTITE PERFECT MATCHING, *and* #CYCLE COVER *are #P-complete.*

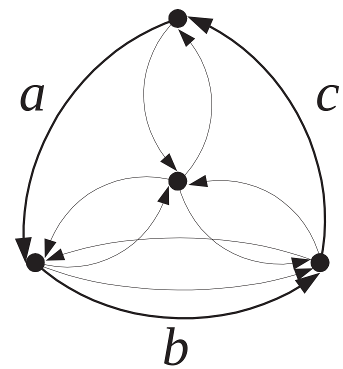
- By Lemma 98 (p. 740), it suffices to prove that #CYCLE COVER is #P-complete.
- #SAT is #P-complete (p. 730).
- #3SAT is #P-complete because it and #SAT are parsimoniously equivalent (p. 280).
- We shall prove that #3SAT is polynomial-time Turing-reducible to #CYCLE COVER.

The Proof (continued)

- Let ϕ be the given 3SAT formula.
  - It contains n variables and m clauses (hence $3m$ literals).
  - It has $\#\phi$ satisfying truth assignments.
- First we construct a *weighted* directed graph H with cycle count
$$\#H = 4^{3m} \times \#\phi.$$
- Then we construct an unweighted directed graph G.
- We make sure $\#H$ (hence $\#\phi$) is polynomial-time Turing-reducible to G's number of cycle covers (denoted $\#G$).

The Proof: the Clause Gadget (continued)

- Each clause is associated with a **clause gadget**.



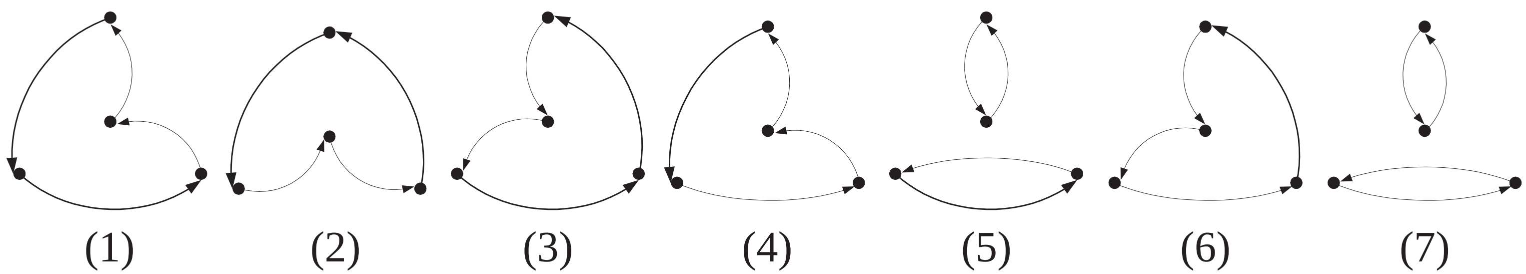
- Each edge has weight 1 unless stated otherwise.
- Each bold edge corresponds to one literal in the clause.
- There are not *parallel* lines as bold edges are schematic only (preview p. 758).

The Proof: the Clause Gadget (continued)

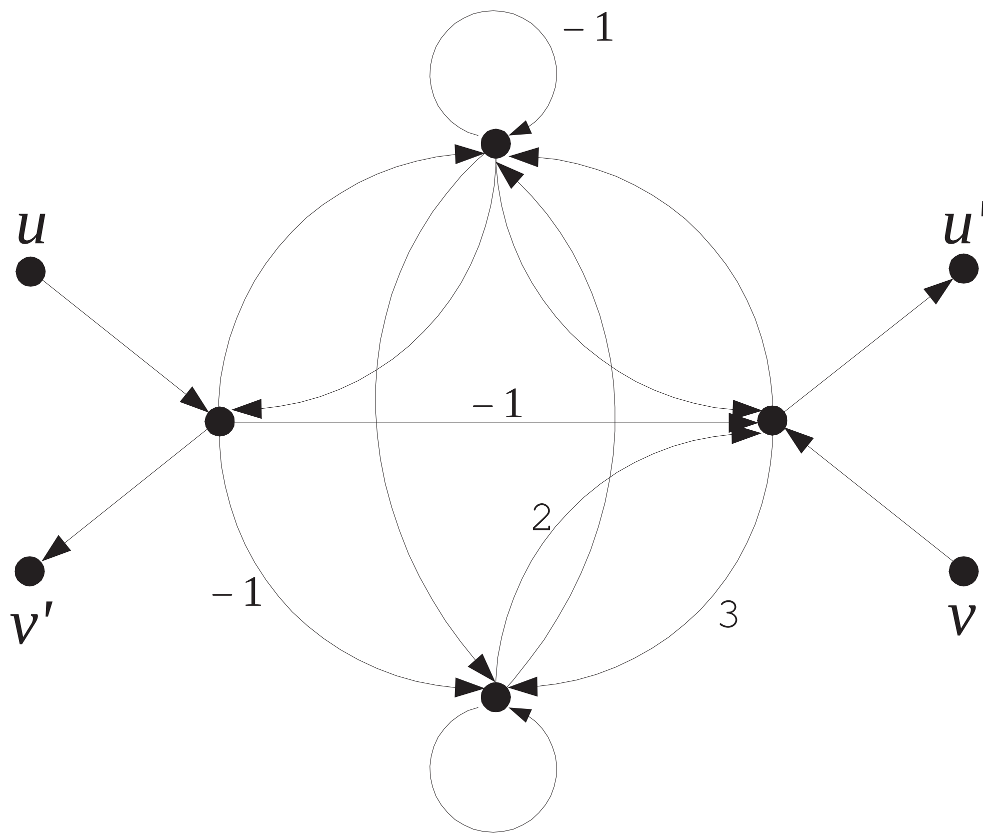
- Following a bold edge means making the literal false (0).
- A cycle cover cannot select *all* 3 bold edges.
  - The interior node would be missing.
- Every proper nonempty subset of bold edges corresponds to a unique cycle cover of weight 1 (see next page).

The Proof: the Clause Gadget (continued)

7 possible cycle covers, one for each satisfying assignment: (1) $a = 0, b = 0, c = 1$, (2) $a = 0, b = 1, c = 0$, etc.

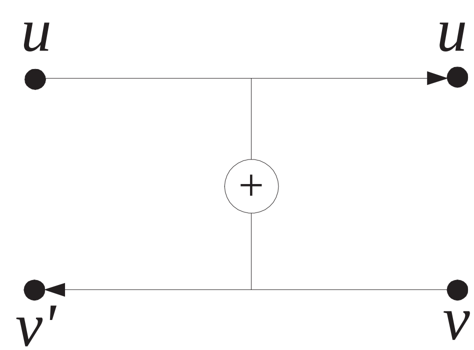


The Proof: the XOR Gadget (continued)



The Proof: Properties of the XOR Gadget (continued)

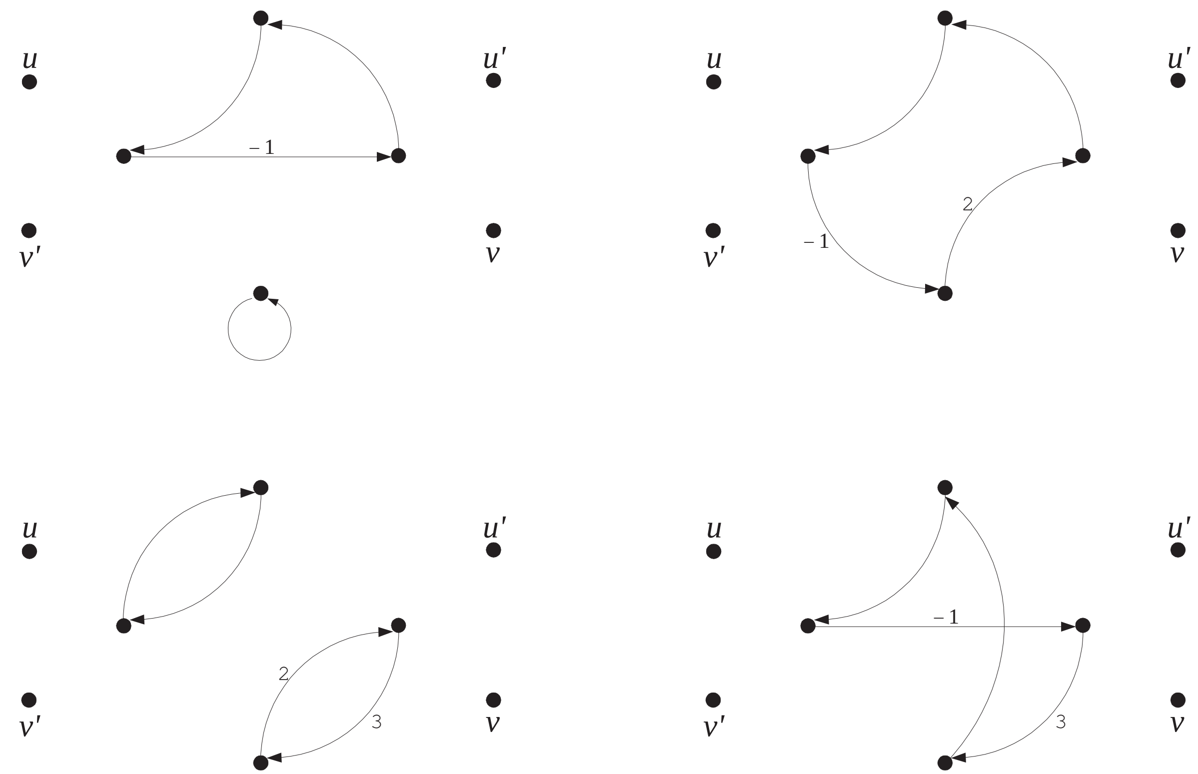
- The XOR gadget schema:



- *At most one* of the 2 schematic edges will be included in a cycle cover.
- There will be $3m$ XOR gadgets, one for each literal.

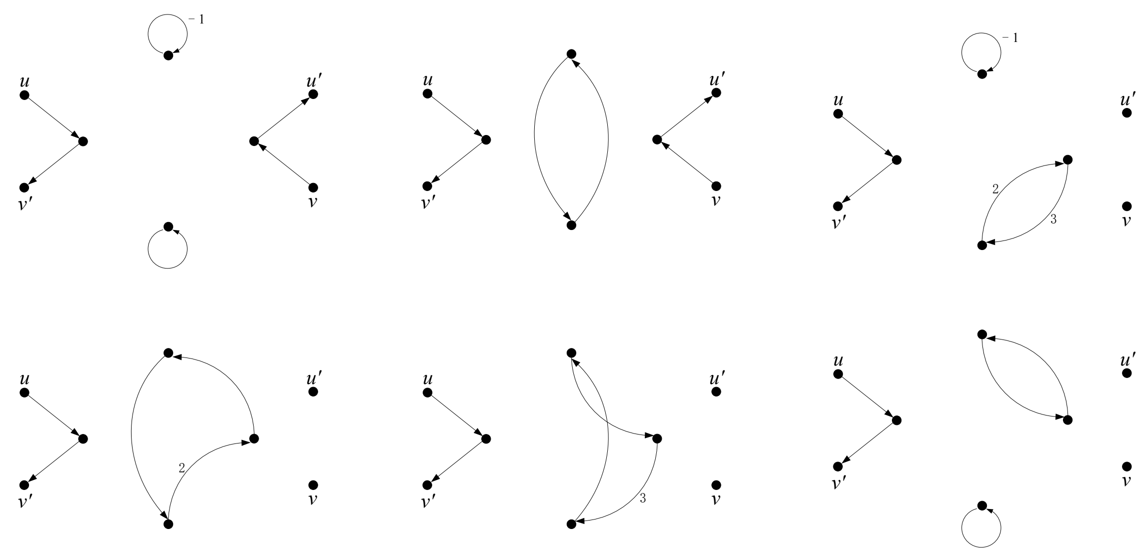
The Proof: Properties of the XOR Gadget (continued)

Total weight of $-1 - 2 + 6 - 3 = 0$ for cycle covers not entering or leaving it.



The Proof: Properties of the XOR Gadget (continued)

- Total weight of $-1+1-6+2+3+1=0$ for cycle covers entering at u and leaving at v'.[a]

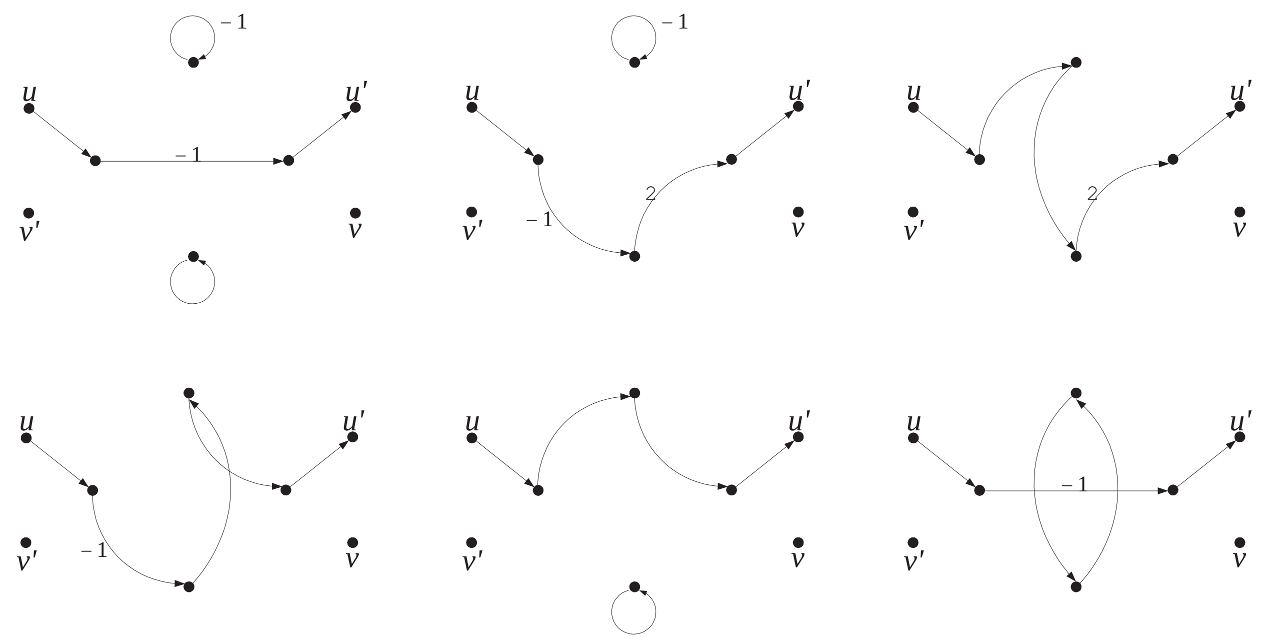


- Same for cycle covers entering at v and leaving at u'.

[a]Corrected by Mr. Yu-Tsung Dai (B91201046) and Mr. Che-Wei Chang (R95922093) on December 27, 2006.

The Proof: Properties of the XOR Gadget (continued)

- Total weight of $1 + 2 + 2 - 1 + 1 - 1 = 4$ for cycle covers entering at u and leaving at u'.



- Same for cycle covers entering at v and leaving at v'.

The Proof: Summary (continued)

- Cycle covers not entering *all* of the XOR gadgets contribute 0 to the cycle count.
  - Let x denote an XOR gadget not entered for a cycle cover c.
  - Now, the said cycle covers' total contribution is

$$
\begin{aligned}
&= \sum_{\text{cycle cover } c \text{ for } H} \text{weight}(c) \\
&= \sum_{\text{cycle cover } c \text{ for } H - x} \text{weight}(c) \sum_{\text{cycle cover } c \text{ for } x} \text{weight}(x) \\
&= \sum_{\text{cycle cover } c \text{ for } H - x} \text{weight}(c) \cdot 0 \\
&= 0.
\end{aligned}
$$

The Proof: Summary (continued)

- Cycle covers entering *any* of the XOR gadgets and leaving illegally contribute 0 to the cycle count.
- For every XOR gadget entered and exited legally, the total weight of a cycle cover is multiplied by 4.
  - With an XOR gadget x entered and exited legally fixed,

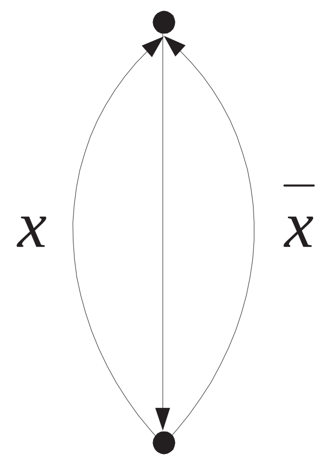
$$
\begin{aligned}
& \text{contributions of such cycle covers to the cycle count} \\
& \sum_{\text{cycle cover } c \text{ for } H} \text{weight}(c) \\
= \; & \sum_{\text{cycle cover } c \text{ for } H-x} \text{weight}(c) \sum_{\text{cycle cover } c \text{ for } x} \text{weight}(x) \\
= \; & \sum_{\text{cycle cover } c \text{ for } H-x} \text{weight}(c) \cdot 4.
\end{aligned}
$$

The Proof: Summary (continued)

- Hereafter we consider only cycle covers which enter every XOR gadget and leaves it legally.
  - Only these cycle covers contribute nonzero weights to the cycle count.
- They are said to **respect** the XOR gadgets.

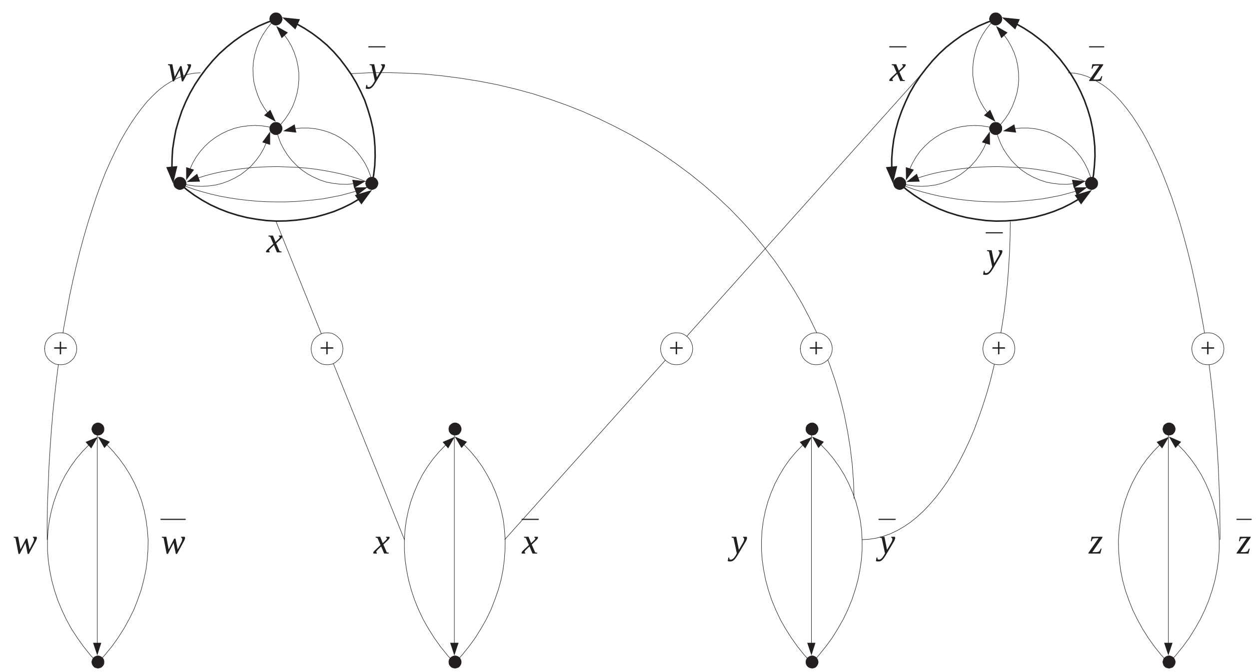
The Proof: the Choice Gadget (continued)

- One choice gadget (a schema) for each variable.

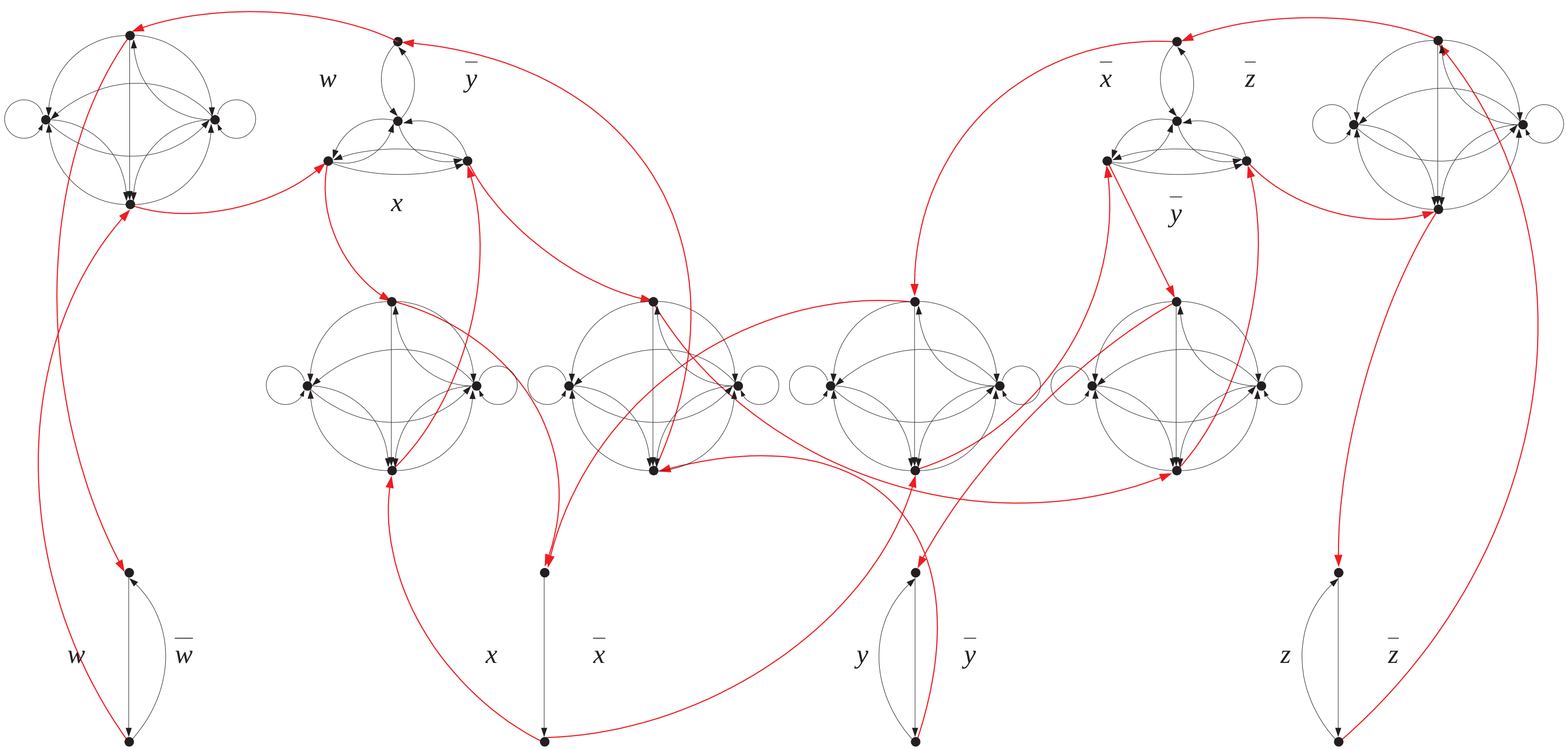


- It gives the truth assignment for the variable.
- Use it with the XOR gadget to enforce consistency.

Schema for $(w \vee x \vee \bar{y}) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$



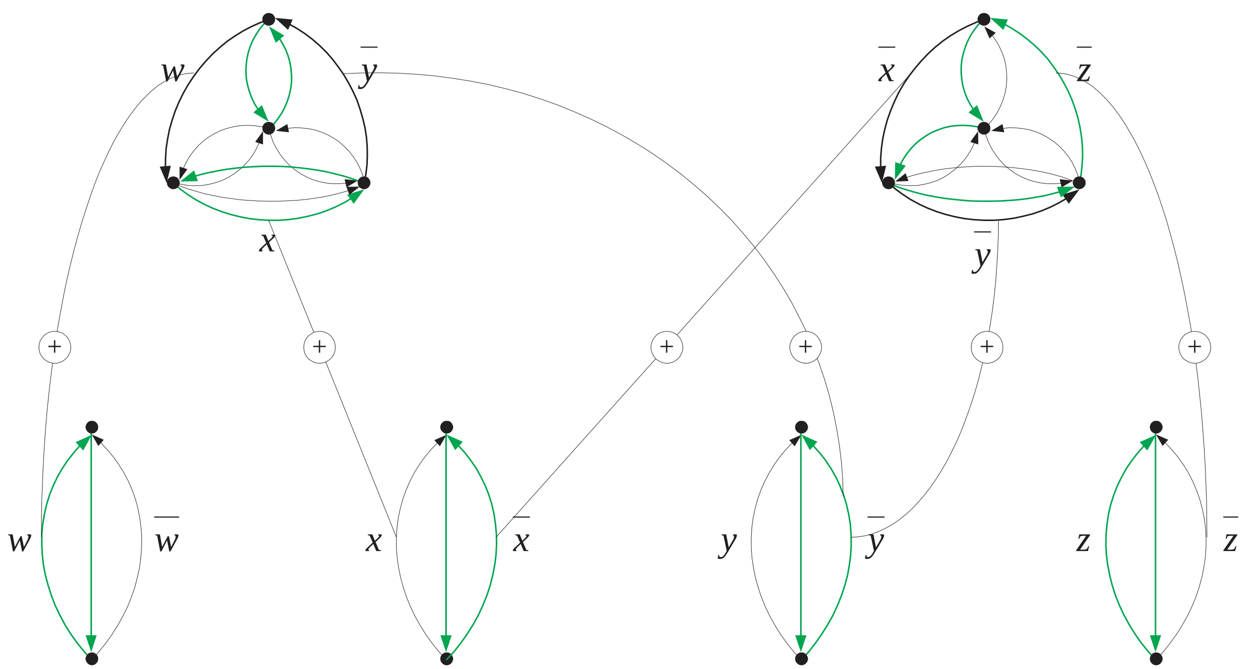
Full Graph $(w \vee x \vee \bar{y}) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$



The Proof: a Key Observation (continued)

Each satisfying truth assignment to ϕ corresponds to a schematic cycle cover that respects the XOR gadgets.

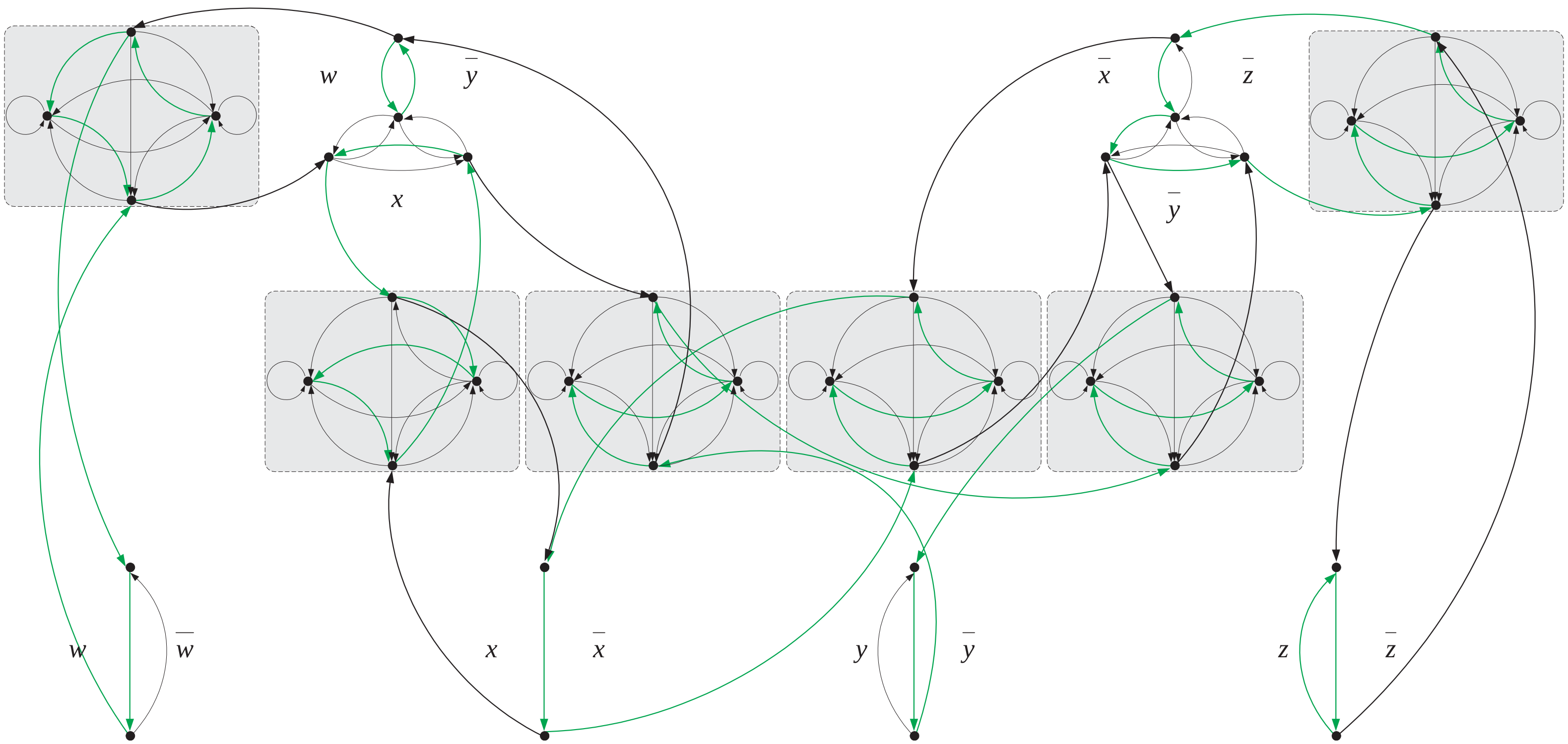
$w = 1, x = 0, y = 0, z = 1 \Leftrightarrow$ One Cycle Cover



The Proof: a Key Corollary (continued)

- Recall that there are $3m$ XOR gadgets.
- Each satisfying truth assignment to ϕ contributes 4^{3m} to the cycle count $\#H$.
- Hence
$$\#H = 4^{3m} \times \#\phi,$$
as desired.

"$w = 1, x = 0, y = 0, z = 1$" Adds 4^6 to Cycle Count



The Proof (continued)

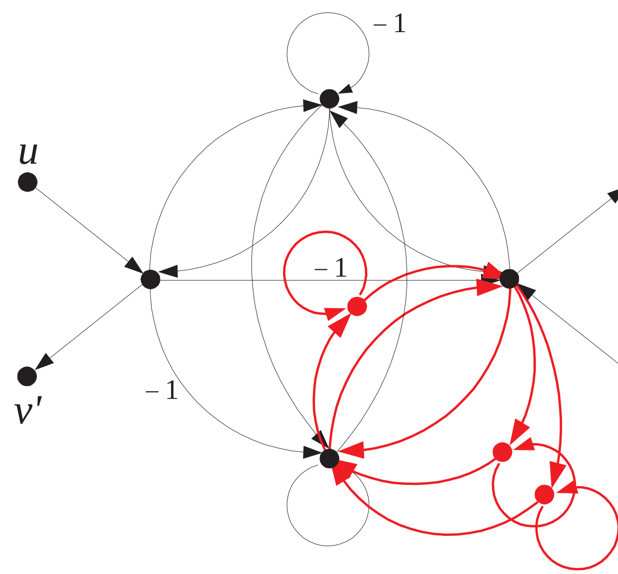
- We are almost done.
- The weighted directed graph H needs to be *efficiently* replaced by some unweighted graph G.
- Furthermore, knowing $\#G$ should enable us to calculate $\#H$ *efficiently*.
  - This done, $\#\phi$ will have been Turing-reducible to $\#G$.[a]
- We proceed to construct this graph G.

---

[a]By way of $\#H$ of course.

The Proof: Construction of G (continued)

- Replace edges with weights 2 and 3 as follows (note that the graph cannot have parallel edges):



- The cycle count $\#H$ remains *unchanged*.

The Proof: Construction of G (continued)

- We move on to edges with weight -1.
- First, we count the number of nodes, M.
- Each clause gadget contains 4 nodes (p. 745), and there are m of them (one per clause).
- Each revised XOR gadget contains 7 nodes (p. 764), and there are $3m$ of them (one per literal).
- Each choice gadget contains 2 nodes (p. 756), and there are $n \leq 3m$ of them (one per variable).
- So

$$M \leq 4m + 21m + 6m = 31m.$$

The Proof: Construction of G (continued)

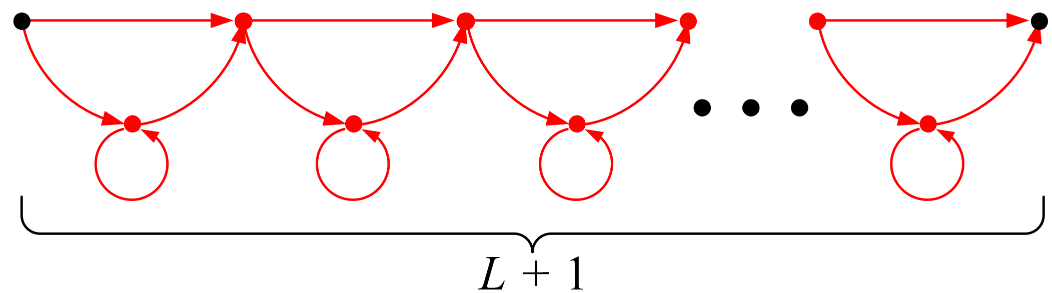
- $\#H \leq 2^L$ for some $L = O(m \log m)$.
  - The maximum absolute value of the edge weight is 1.
  - Hence each term in the permanent is at most 1.
  - There are $M! \leq (31m)!$ terms.
  - Hence

$$\begin{aligned} \#H &\leq \sqrt{2\pi(31m)} \left(\frac{31m}{e}\right)^{31m} e^{\frac{1}{12\times(31m)}} \\ &= 2^{O(m \log m)} \end{aligned} \tag{12}$$

    by a refined Stirling's formula.

The Proof: Construction of G (continued)

- Replace each edge with weight -1 with the following:



- Each increases the number of cycle covers 2^{L+1}-fold.
- The desired unweighted G has been obtained.

The Proof (continued)

- $\#G$ equals $\#H$ after replacing each appearance -1 in $\#H$ with 2^{L+1}:

$$\begin{aligned}
\#H &= \cdots + \overbrace{(-1) \cdot 1 \cdots\cdots 1}^{\text{a cycle cover}} + \cdots\,, \\
\#G &= \cdots + \overbrace{2^{L+1} \cdot 1 \cdots\cdots 1}^{\text{a cycle cover}} + \cdots\,.
\end{aligned}$$

- Let $\#G = \sum_{i=0}^{n} a_i \times (2^{L+1})^i$, where $0 \leq a_i < 2^{L+1}$.
- As $\#H \leq 2^L$ even if we replace -1 by 1 (p. 766), each a_i equals the number of cycle covers with i edges of weight -1.

The Proof (concluded)

- We conclude that

$$\#H = a_0 - a_1 + a_2 - \cdots + (-1)^n a_n,$$

indeed easily computable from $\#G$.

- We know $\#H = 4^{3m} \times \#\phi$ (p. 761).
- So

$$\#\phi = \frac{a_0 - a_1 + a_2 - \cdots + (-1)^n a_n}{4^{3m}}.$$

  – More succinctly,

$$\#\phi = \frac{\#G \bmod (2^{L+1} + 1)}{4^{3m}}.$$

Finis