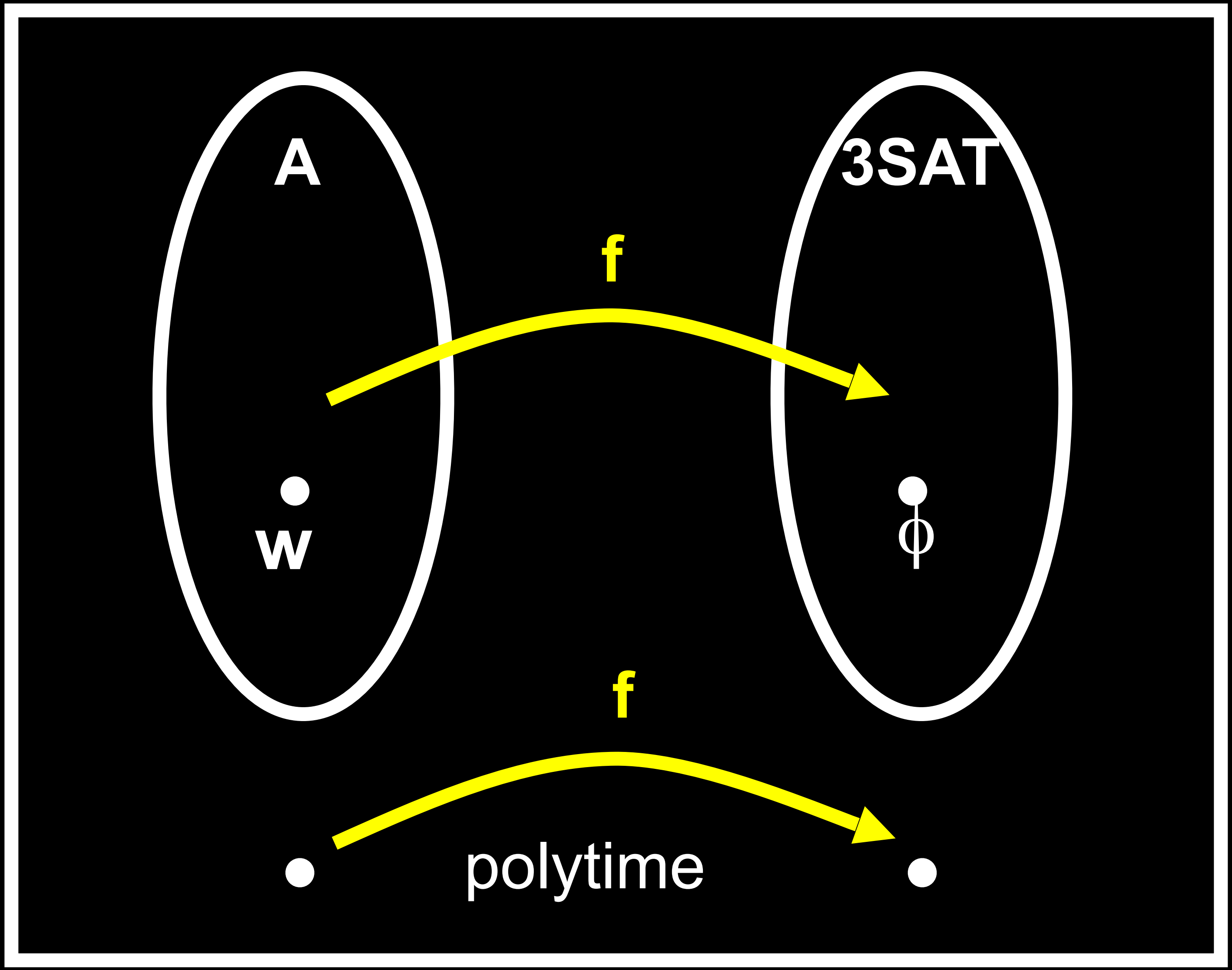
Theorem (Cook-Levin): 3SAT is NP-complete

Corollary: 3SAT $\in$ P if and only if P = NP

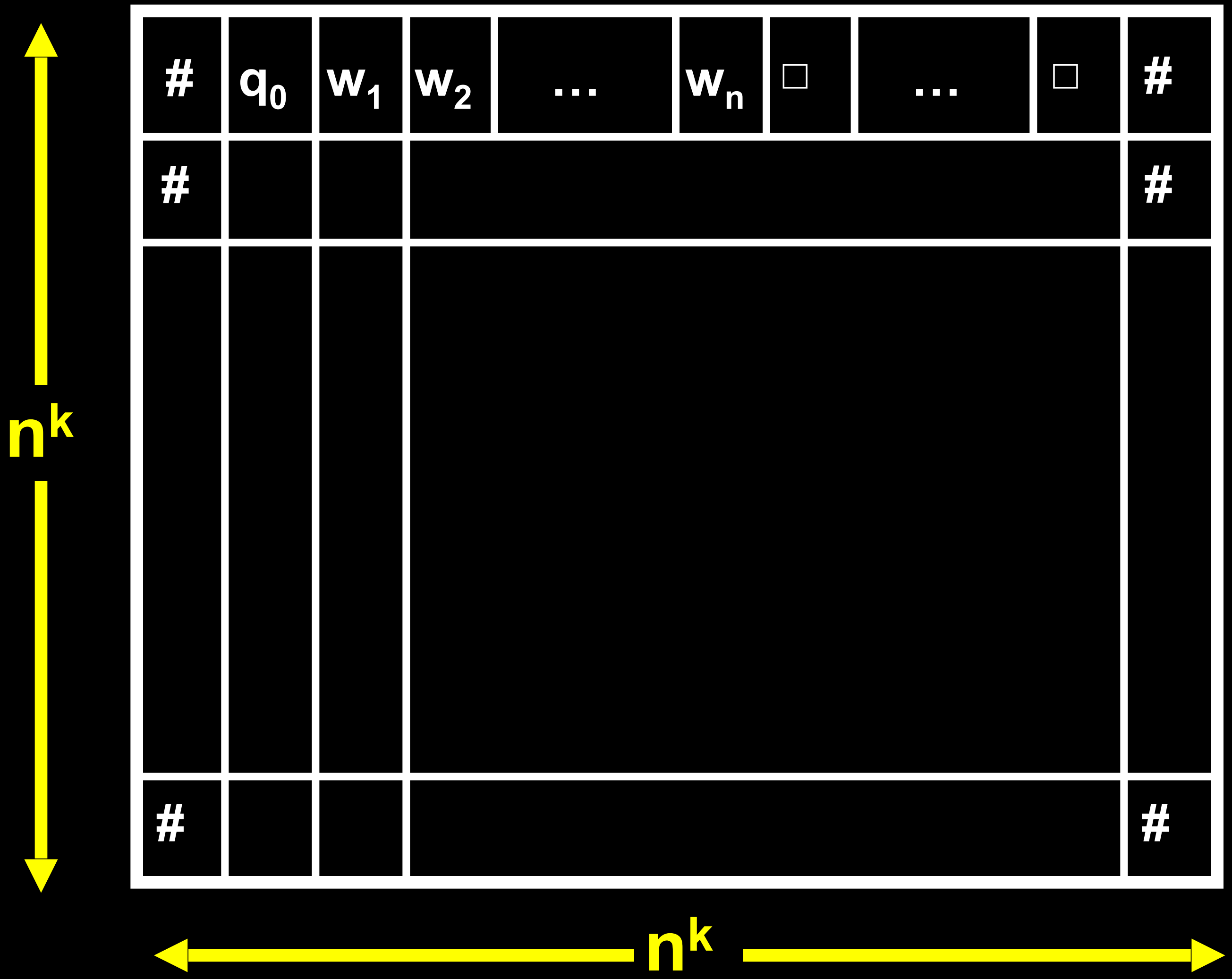


The reduction f turns a string w into a 3-cnf formula ϕ such that: w $\in$ A $\Leftrightarrow$ $\phi \in$ 3SAT.

ϕ will simulate the NP machine N for A on w.

Suppose $A \in NTIME(n^k)$ and let N be an NP machine for A.

A **tableau for N on w** is an $n^k \times n^k$ table whose rows are the configurations of *some* possible computation of N on input w.



$$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$$

$$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$$

$$\phi_{cell} = \bigwedge_{1 \leq i, j \leq n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{\substack{s,t \in C \\ s \neq t}} (\neg x_{i,j,s} \vee \neg x_{i,j,t}) \right) \right]$$

$O(n^{2k})$ clauses

$\text{Length}(\phi_{cell}) = O(n^{2k})\ O(\log n^k) = O(n^{2k} \log n)$

$O(\log n^k)$: length(indices)

$$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$$

$$\phi_{start} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge$$
$$x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \cdots \wedge x_{1,n+2,w_n} \wedge$$
$$x_{1,n+3,\square} \wedge \cdots \wedge x_{1,n^k-1,\square} \wedge x_{1,n^k,\#}$$

$$O(n^k)$$

$$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$$

$$\phi_{accept} = \bigvee_{1 \leq i, j \leq n^k} x_{i,j,q_{accept}}$$

$$O(n^{2k})$$

$$\phi_{move} = \bigwedge_{1 \le i,\, j \le n^k} (\text{ the (i, j) window is legal })$$

the (i, j) window is legal =

$$\bigwedge_{\substack{a_1, \ldots, a_6 \\ \text{ISN'T a legal window}}} \left(\overline{x}_{i,j-1,a_1} \vee \overline{x}_{i,j,a_2} \vee \overline{x}_{i,j,+1,a_3} \vee \overline{x}_{i+1,j-1,a_4} \vee \overline{x}_{i+1,j,a_5} \vee \overline{x}_{i+1,j+1,a_6} \right)$$

This is a conjunct over all ($\le |C|^6$) illegal sequences ($a_1, \ldots, a_6$).

$$O(n^{2k})$$

3-SAT?

How do we convert the whole thing into a 3-cnf formula?

Everything was an AND of ORs
We just need to make those ORs with 3 literals

If a clause has less than three variables:

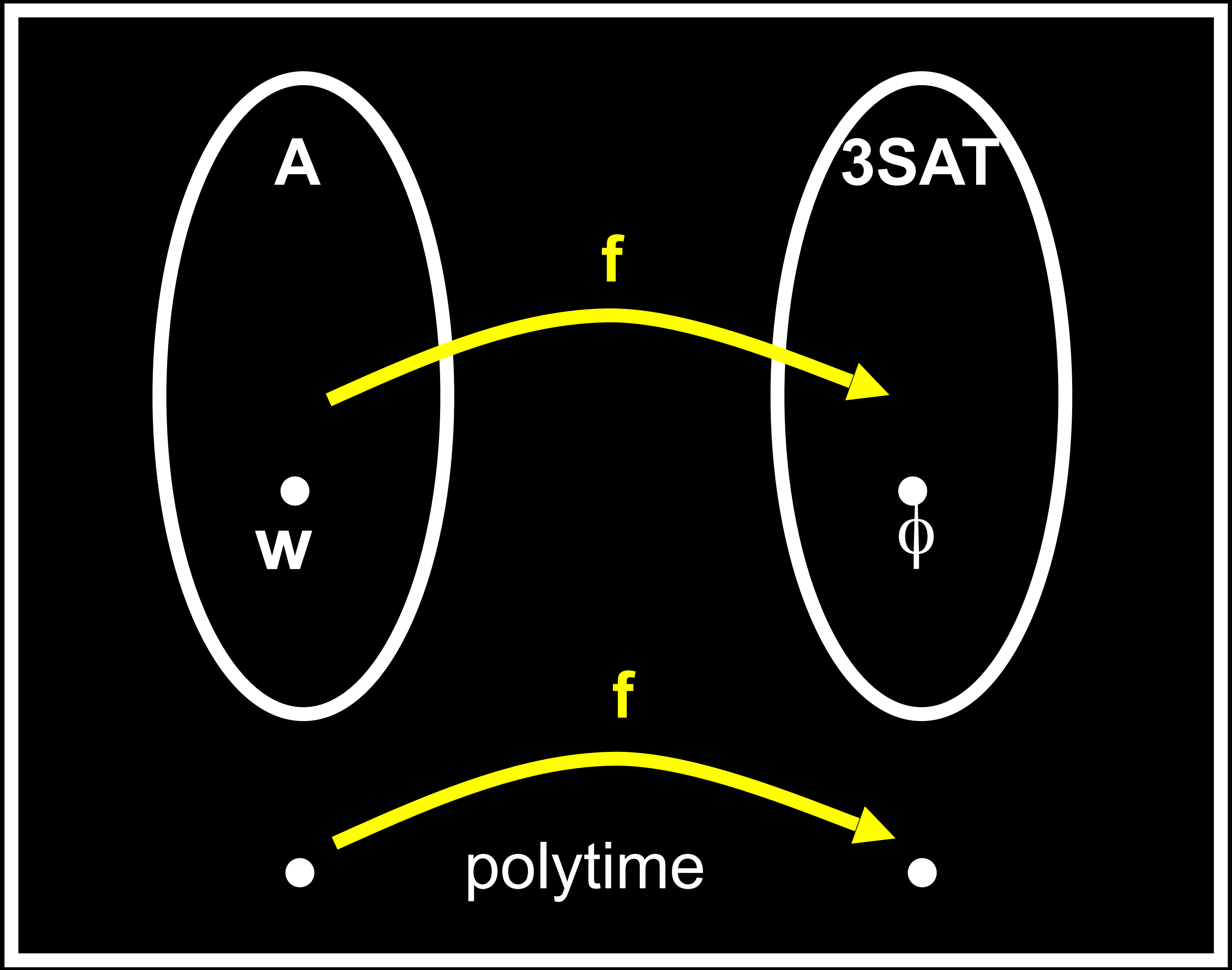
$a \equiv (a \vee a \vee a)$, $(a \vee b) \equiv (a \vee b \vee b)$

If a clause has more than three variables:

$(a \vee b \vee c \vee d) \equiv (a \vee b \vee z) \wedge (\neg z \vee c \vee d)$

$(a_1 \vee a_2 \vee \ldots \vee a_t) \equiv$

$(a_1 \vee a_2 \vee z_1) \wedge (\neg z_1 \vee a_3 \vee z_2) \wedge \ldots (\neg z_{t-3} \vee a_{t-1} \vee z_t)$



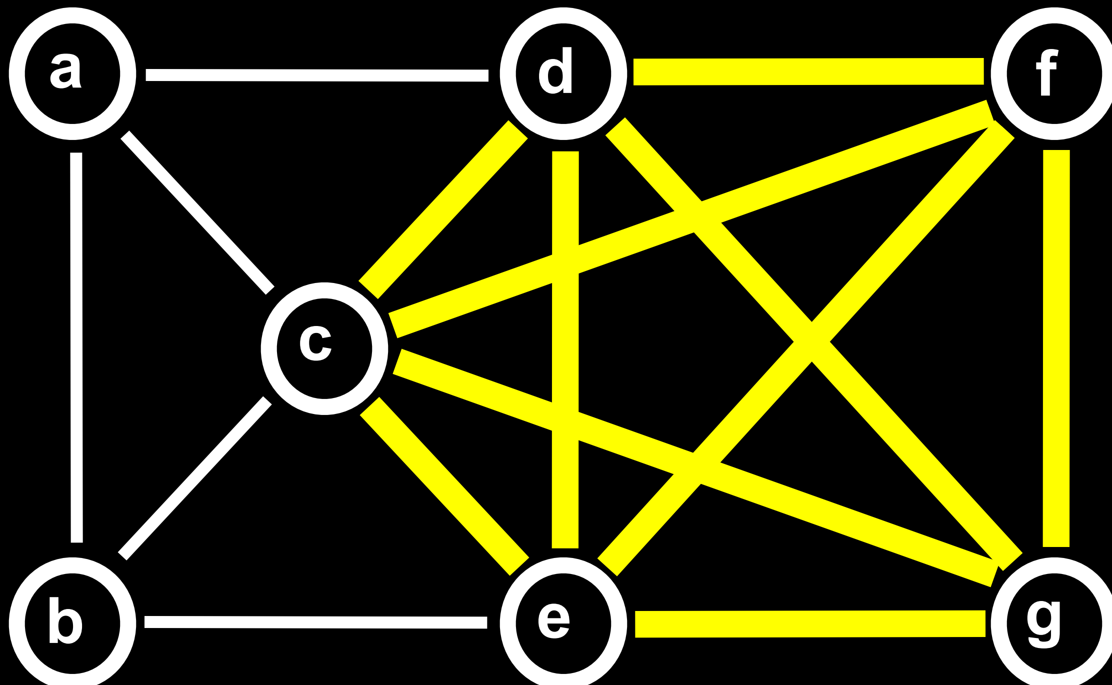
Given A in NP. The reduction f turned a string w into a 3-cnf formula ϕ such that: $w \in A \Leftrightarrow \phi \in 3SAT$.

NP-COMPLETENESS II

Tuesday April 1

There are googols of NP-complete languages

K-**CLIQUE**



k-clique = complete subgraph of k nodes

Assume a reasonable encoding of graphs (example: the adjacency matrix is reasonable)

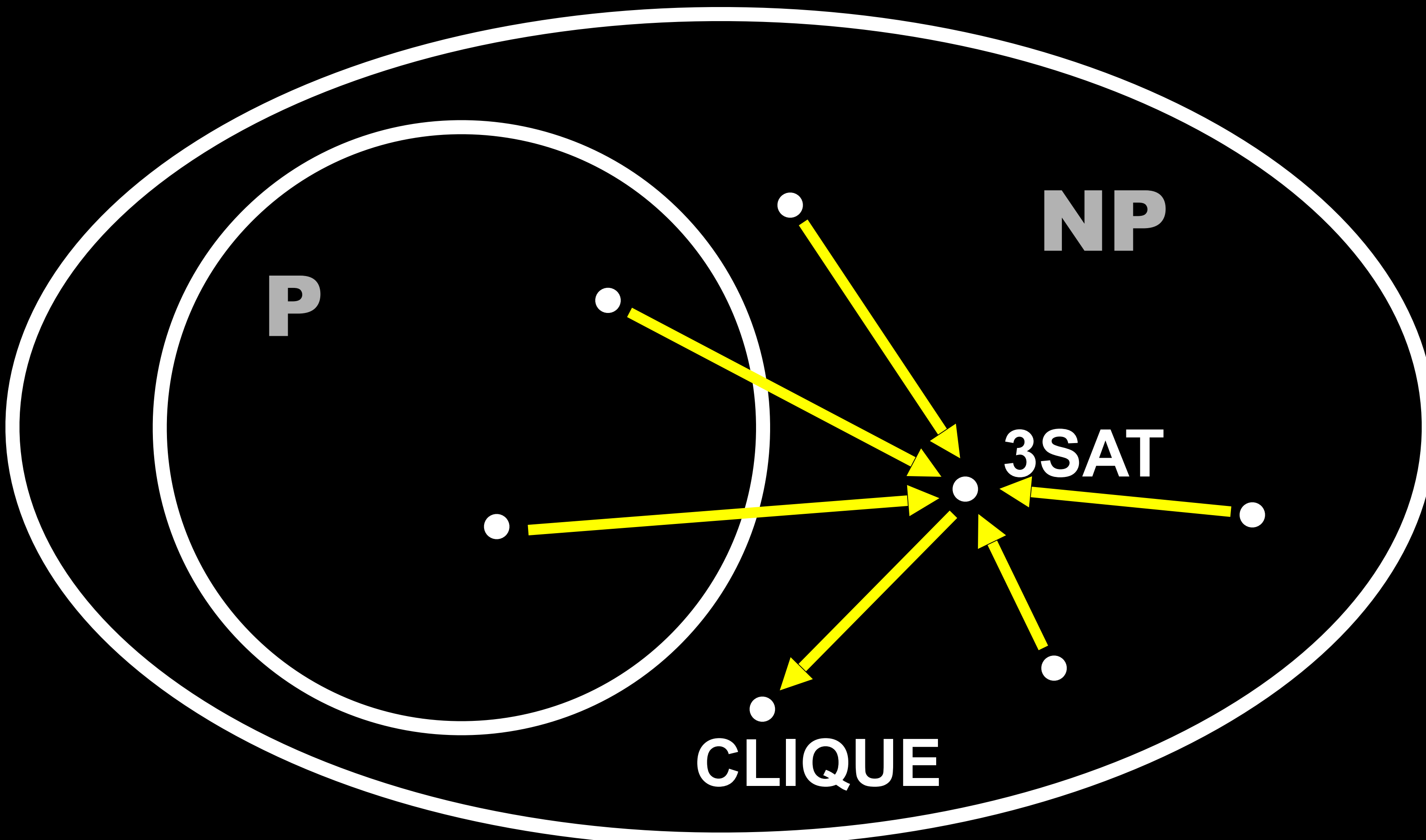
CLIQUE = { (G,k) | G is an undirected graph with a k-clique }

Theorem: CLIQUE is NP-Complete

(1) CLIQUE $\in$ NP

(2) 3SAT $\leq_P$ CLIQUE

CLIQUE is NP-Complete

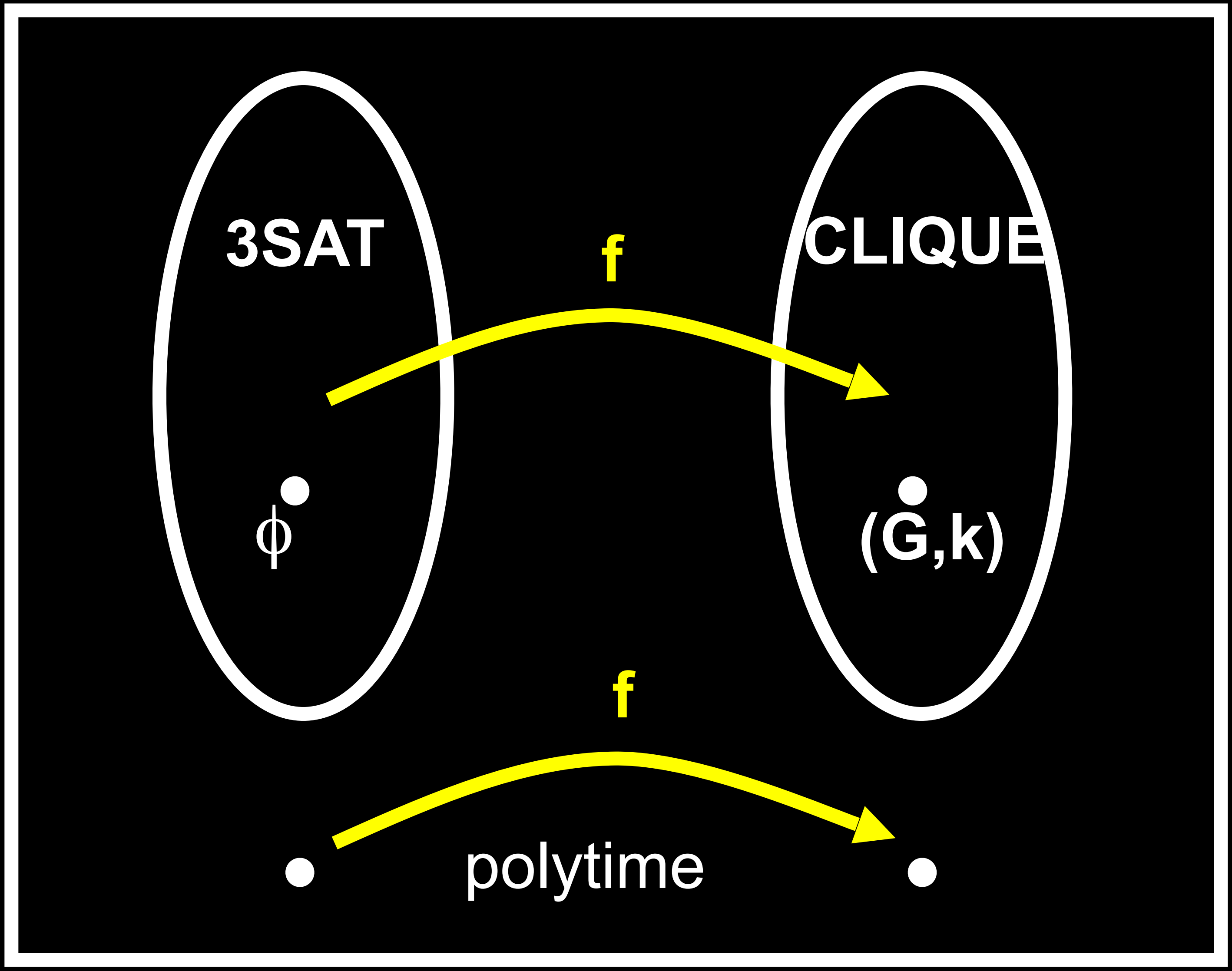


3SAT $\leq_P$ CLIQUE

We transform a 3-cnf formula ϕ into (G,k) such that

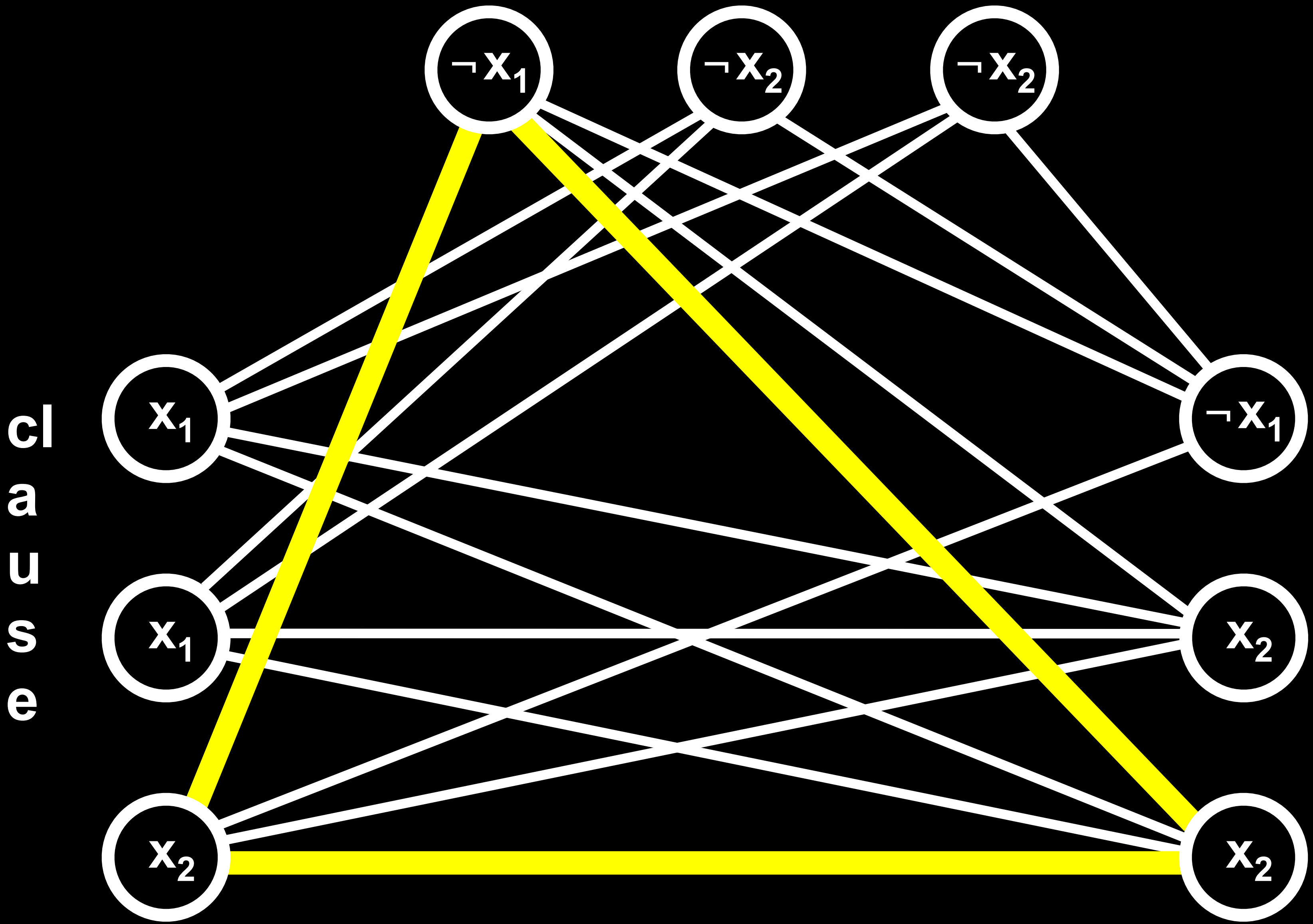
$$\phi \in \text{3SAT} \Leftrightarrow (G,k) \in \text{CLIQUE}$$

The transformation can be done in time that is polynomial in the length of ϕ



The reduction f will turn a 3-cnf formula ϕ into a graph (G,k) such that $\phi \in$ 3SAT $\Leftrightarrow$ (G,k) $\in$ CLIQUE

$(x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_2)$



#nodes = 3(# clauses)

k = #clauses

3SAT $\leq_P$ CLIQUE

We transform a 3-cnf formula ϕ into (G,k) such that

$$\phi \in 3SAT \Leftrightarrow (G,k) \in CLIQUE$$

If ϕ has k clauses, we create a graph with k clusters of 3 nodes each.

Each cluster corresponds to a clause.

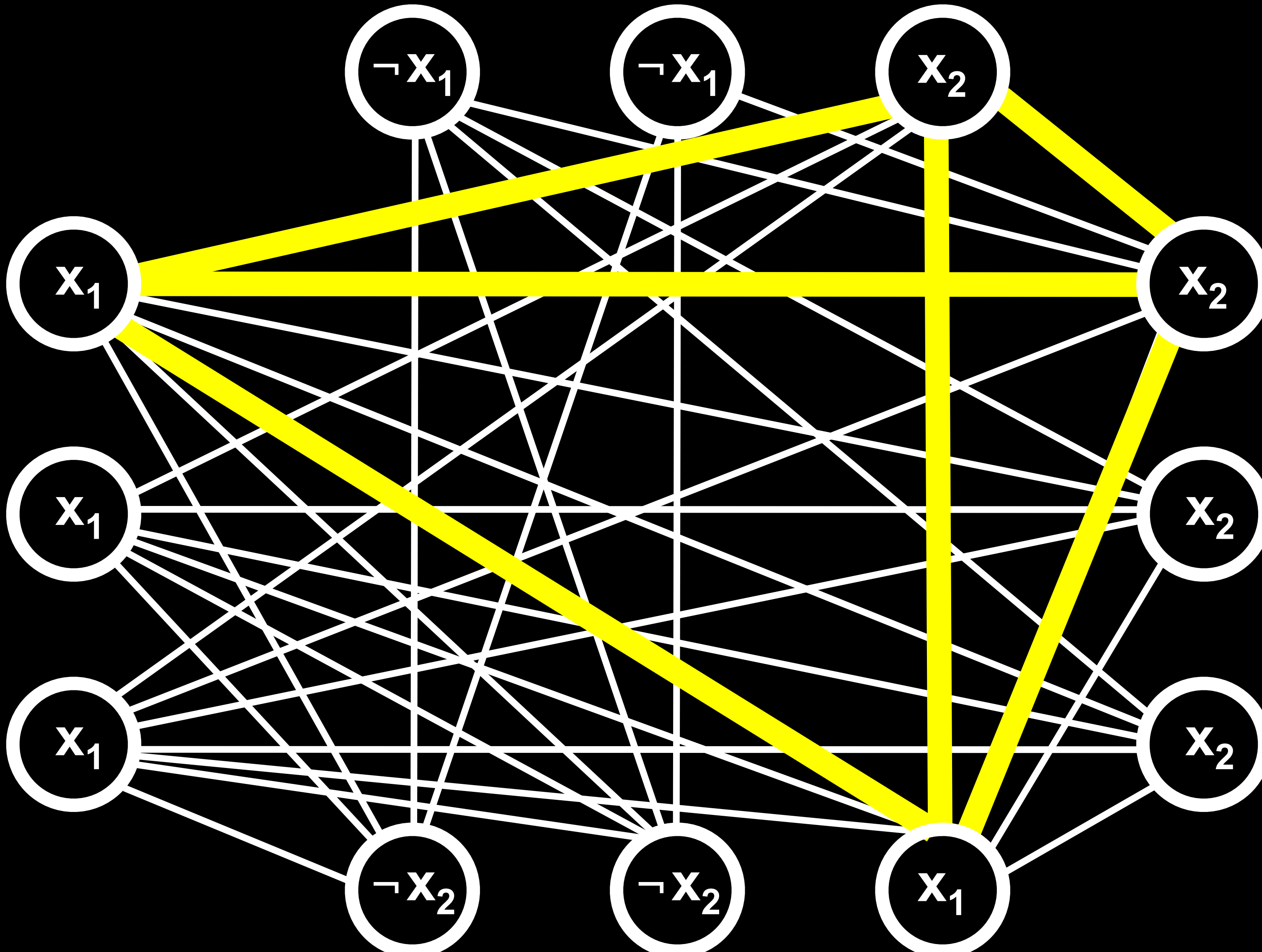
Each node in a cluster is labeled with a literal from the clause.

We do not connect any nodes in the same cluster

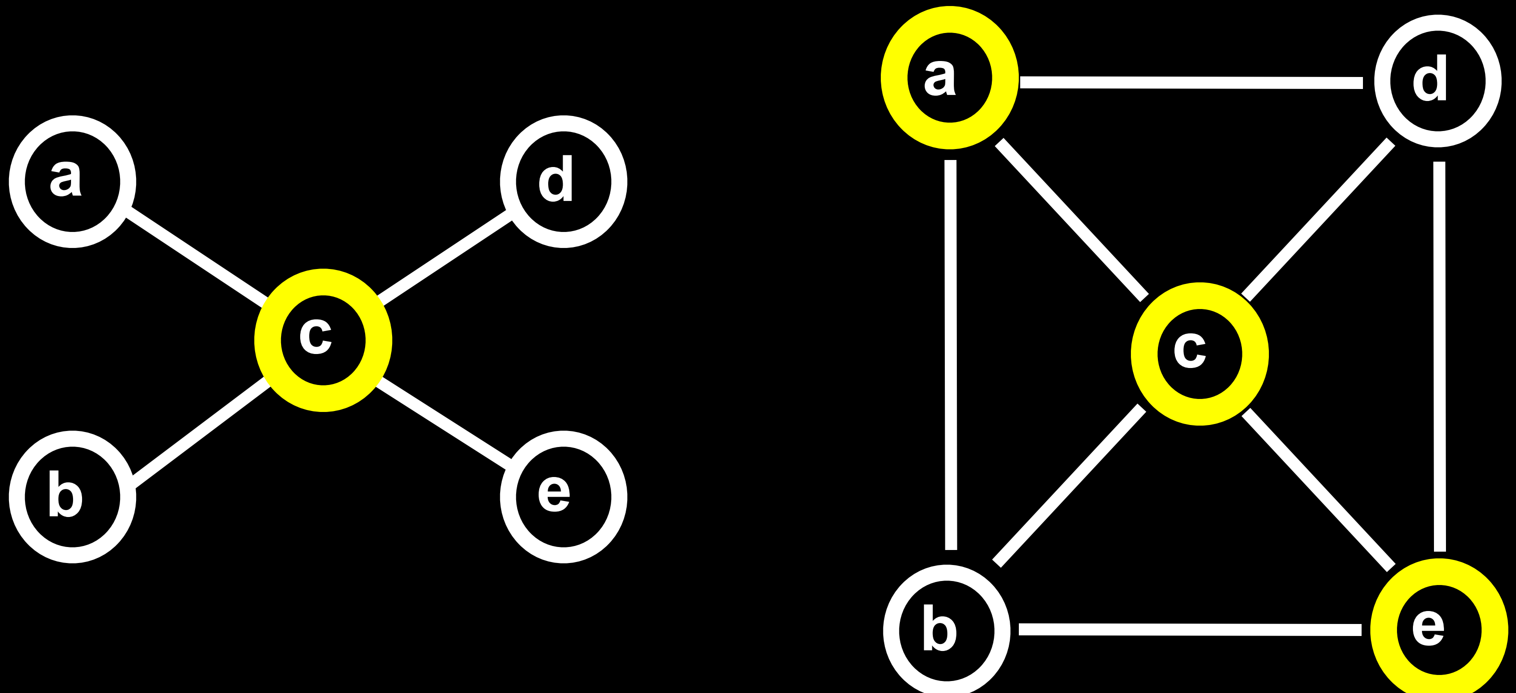
We connect nodes in different clusters whenever they are not contradictory

The transformation can be done in time that is polynomial in the length of ϕ

$$(x_1 \vee x_1 \vee x_1) \wedge (\neg x_1 \vee \neg x_1 \vee x_2) \wedge$$
$$(x_2 \vee x_2 \vee x_2) \wedge (\neg x_2 \vee \neg x_2 \vee x_1)$$



VERTEX-COVER



vertex cover = set of nodes that cover all edges

VERTEX-COVER = { (G,k) | G is an undirected graph with a k-node vertex cover }

Theorem: VERTEX-COVER is NP-Complete

(1) VERTEX-COVER $\in$ NP

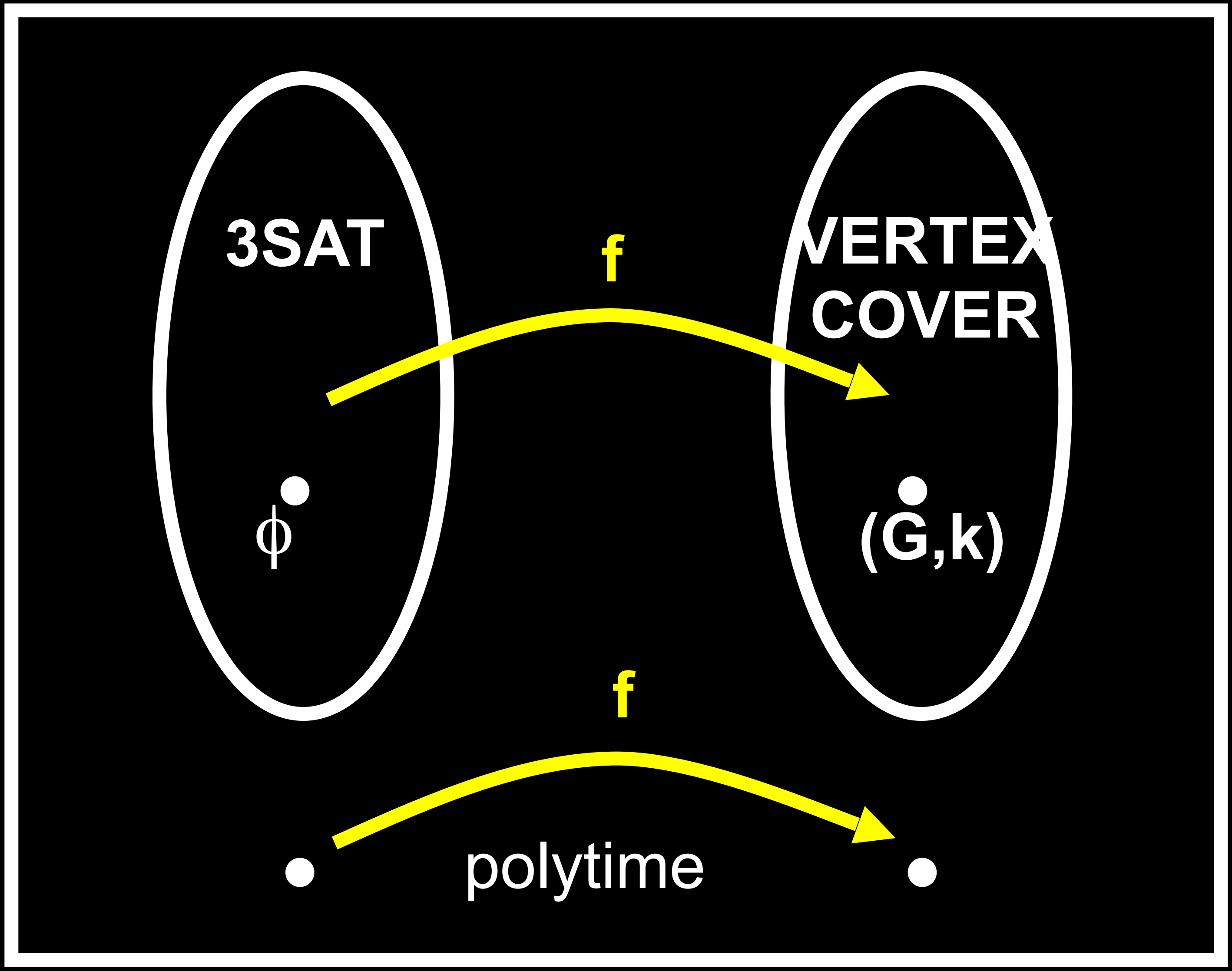
(2) 3SAT $\leq_P$ VERTEX-COVER

3SAT $\leq_P$ VERTEX-COVER

We transform a 3-cnf formula ϕ into (G,k) such that

$$\phi \in \text{3SAT} \Leftrightarrow (G,k) \in \text{VERTEX-COVER}$$

The transformation can be done in time polynomial in the length of ϕ

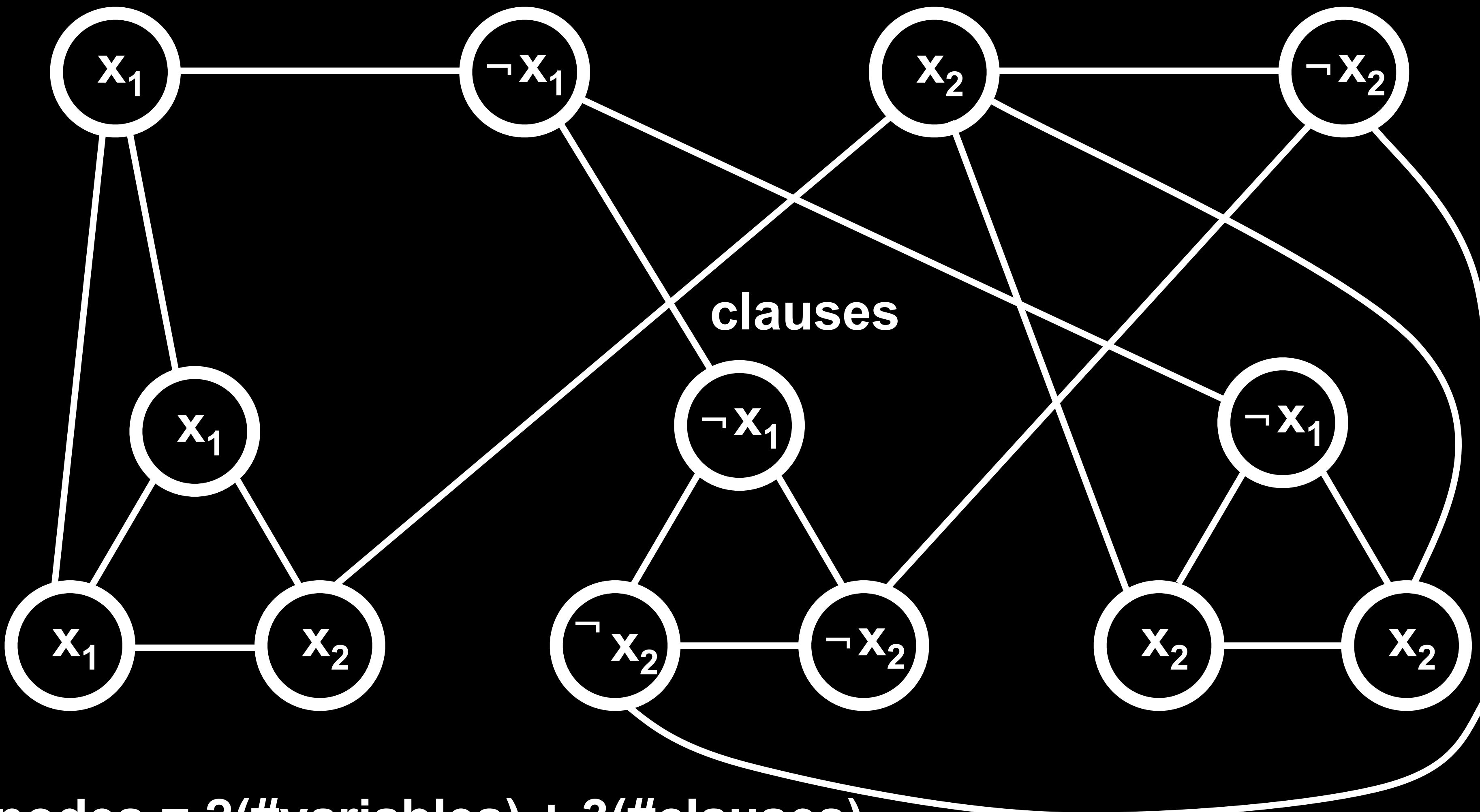


The reduction f will turn a 3-cnf formula ϕ into a graph (G,k)

such that $\phi \in$ 3SAT $\Leftrightarrow$ (G,k) $\in$ VERTEX-COVER

$(x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_2)$

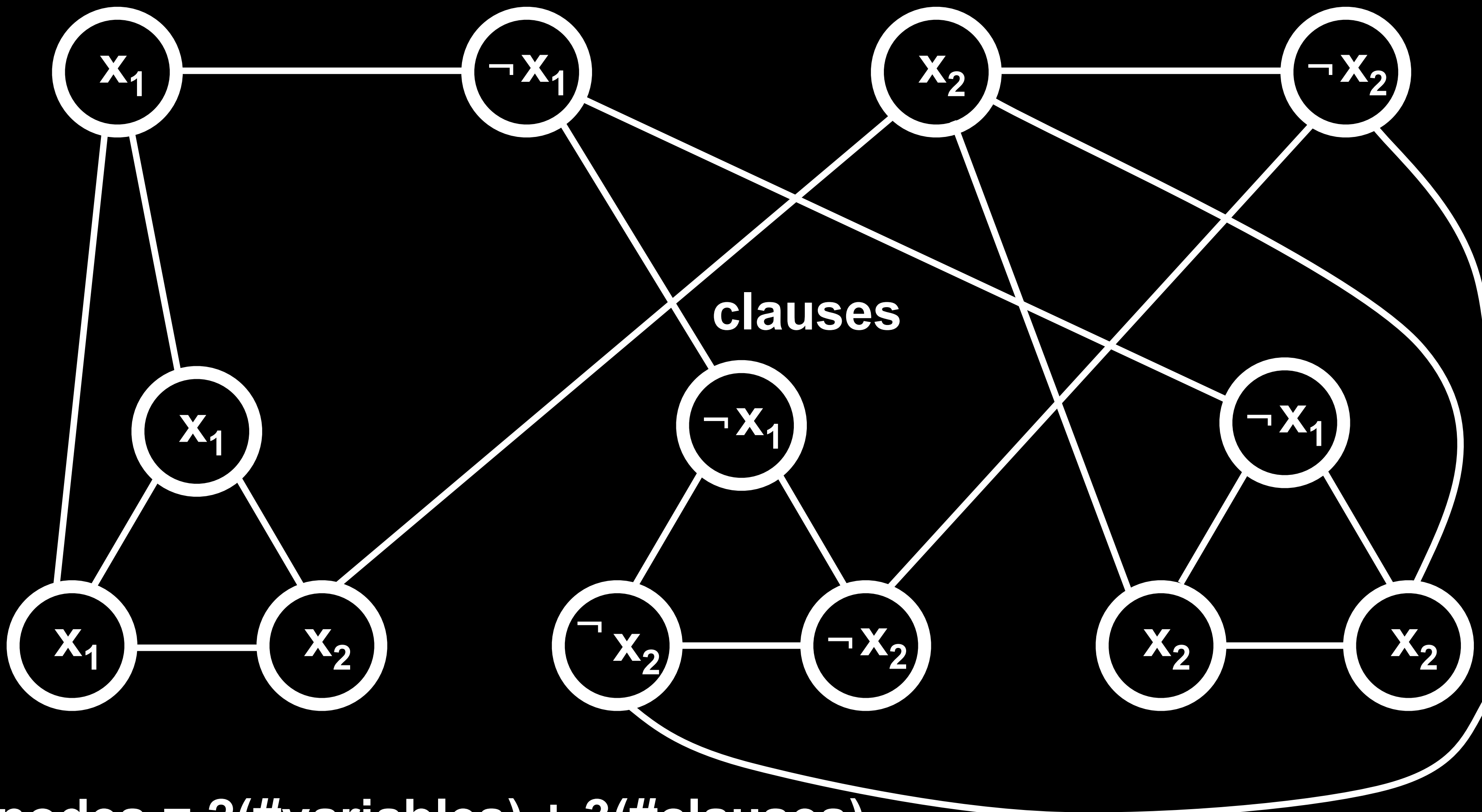
Variables and negations of variables



#nodes = 2(#variables) + 3(#clauses)

$$(x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_2)$$

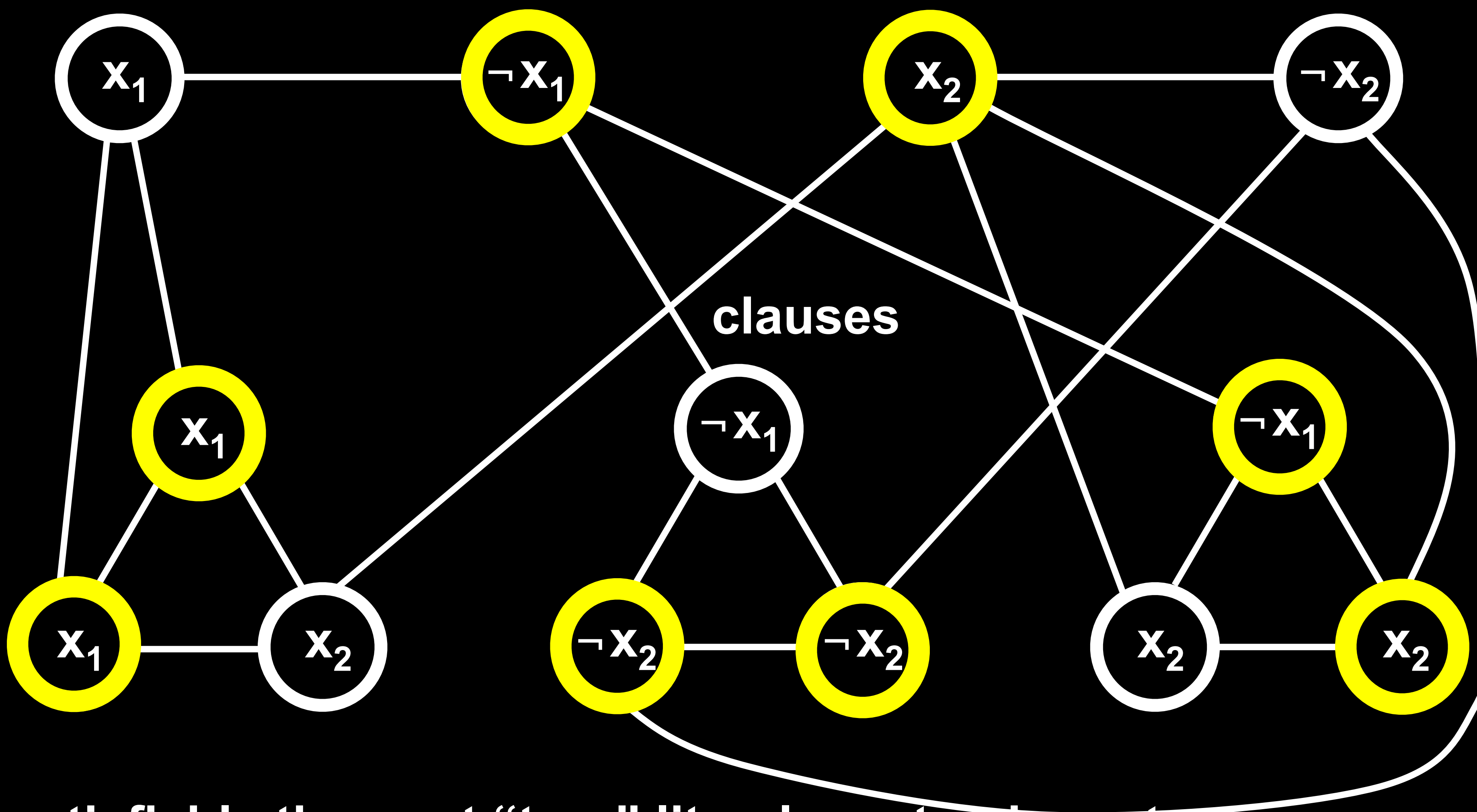
Variables and negations of variables



#nodes = 2(#variables) + 3(#clauses)

$(x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_2)$

Variables and negations of variables

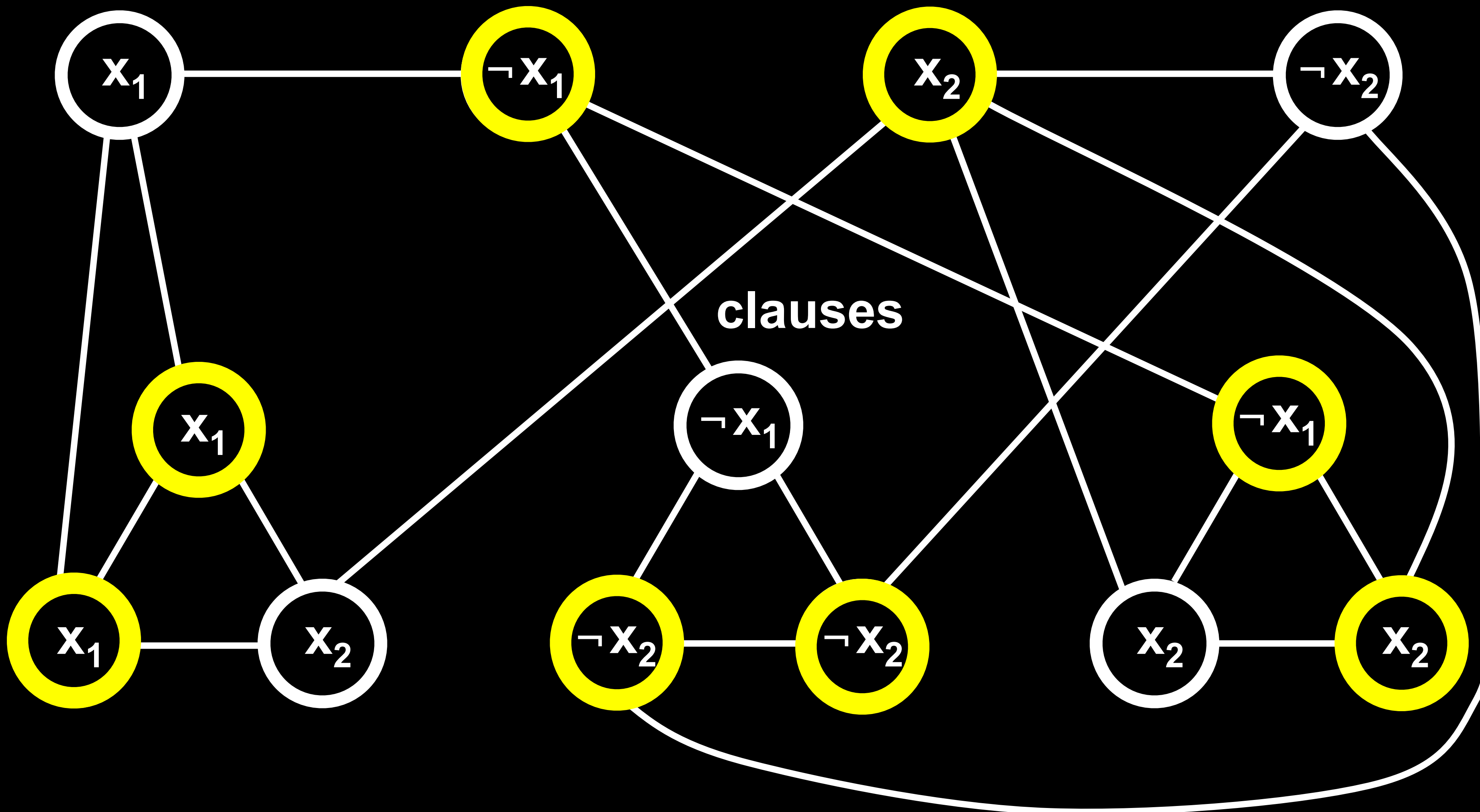


ϕ satisfiable then put "true" literals on top in vertex cover

For each clause, pick a true literal and put other 2 in vertex cover

$$(x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_2)$$

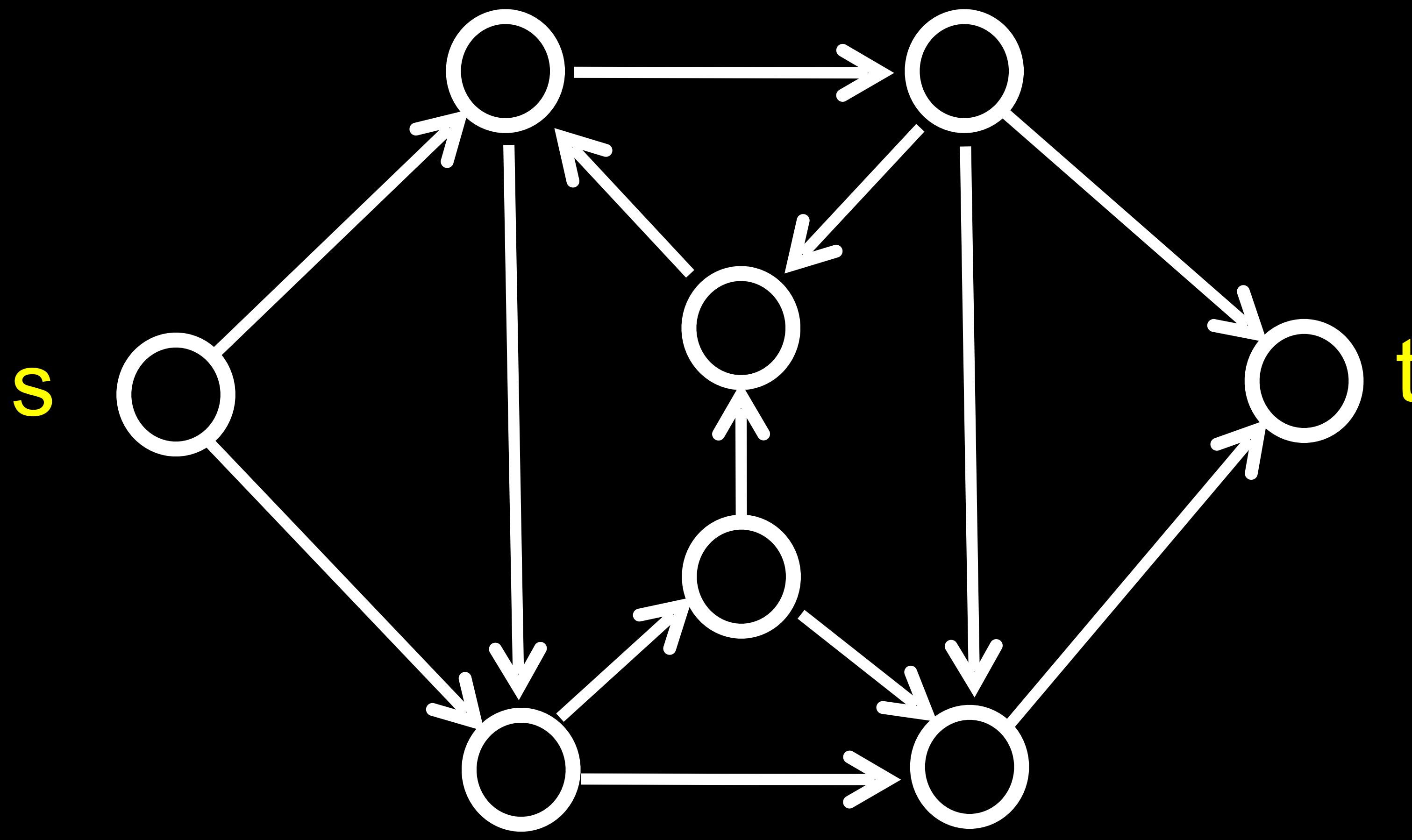
Variables and negations of variables



$k = 2(\#\text{clauses}) + (\#\text{variables})$

$$(x_1 \vee x_1 \vee x_1) \wedge (\neg x_1 \vee \neg x_1 \vee x_2) \wedge$$
$$(x_2 \vee x_2 \vee x_2) \wedge (\neg x_2 \vee \neg x_2 \vee x_1)$$

HAMILTON PATH



HAMPATH = { (G,s,t) | G is an directed graph
with a Hamilton path from s to t}

Theorem: HAMPATH is NP-Complete

(1) HAMPATH $\in$ NP

(2) 3SAT $\leq_P$ HAMPATH

Proof is in Sipser, Chapter 7.5

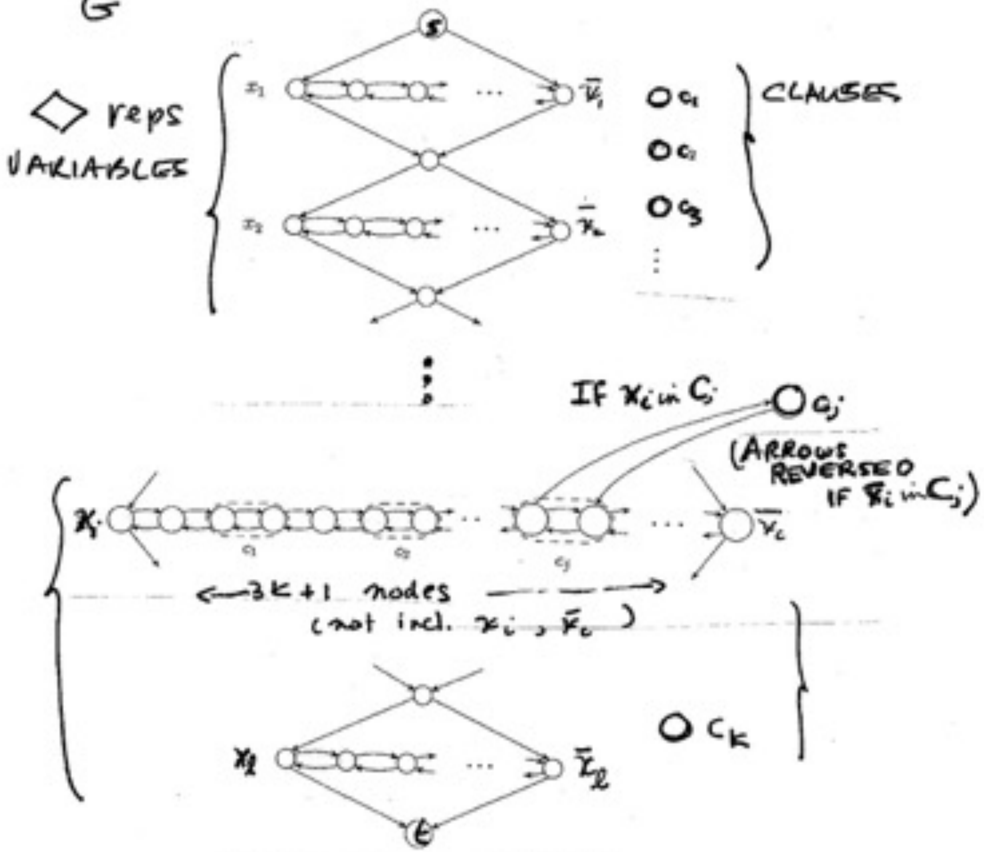
$3SAT \leq_P HAMPATH$

$\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_j \wedge \cdots \wedge C_k$ C_j, CLAUSE

$x_1, x_2, \ldots, x_\ell$ VARIABLES

$\downarrow$

G



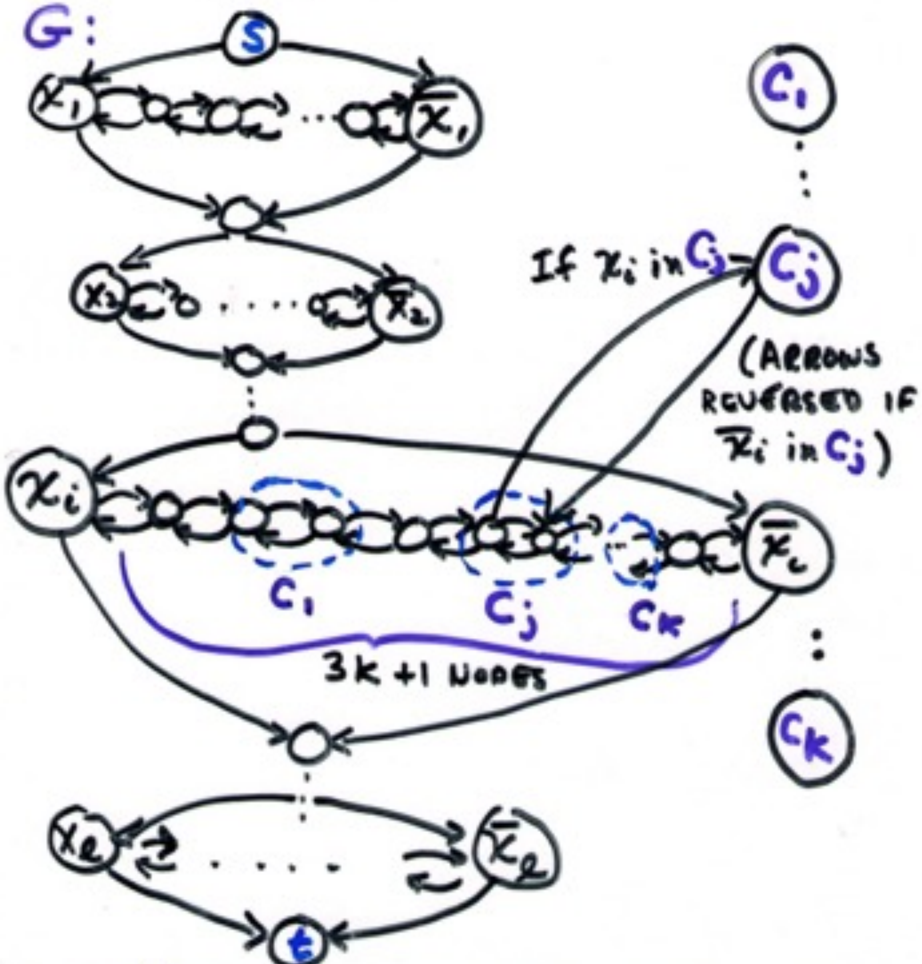
- SUPPOSE ϕ SATISFIABLE WITH SOME TRUTH ASSIGNMENT.
- ZIG-ZAG IF x_i is TRUE(1); ZAG-ZIG if $\bar{x}_i$ is TRUE(1).

3SAT $\leq_p$ HAM PATH

$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_j \wedge \cdots \wedge C_k$ C_j, CLAUSE

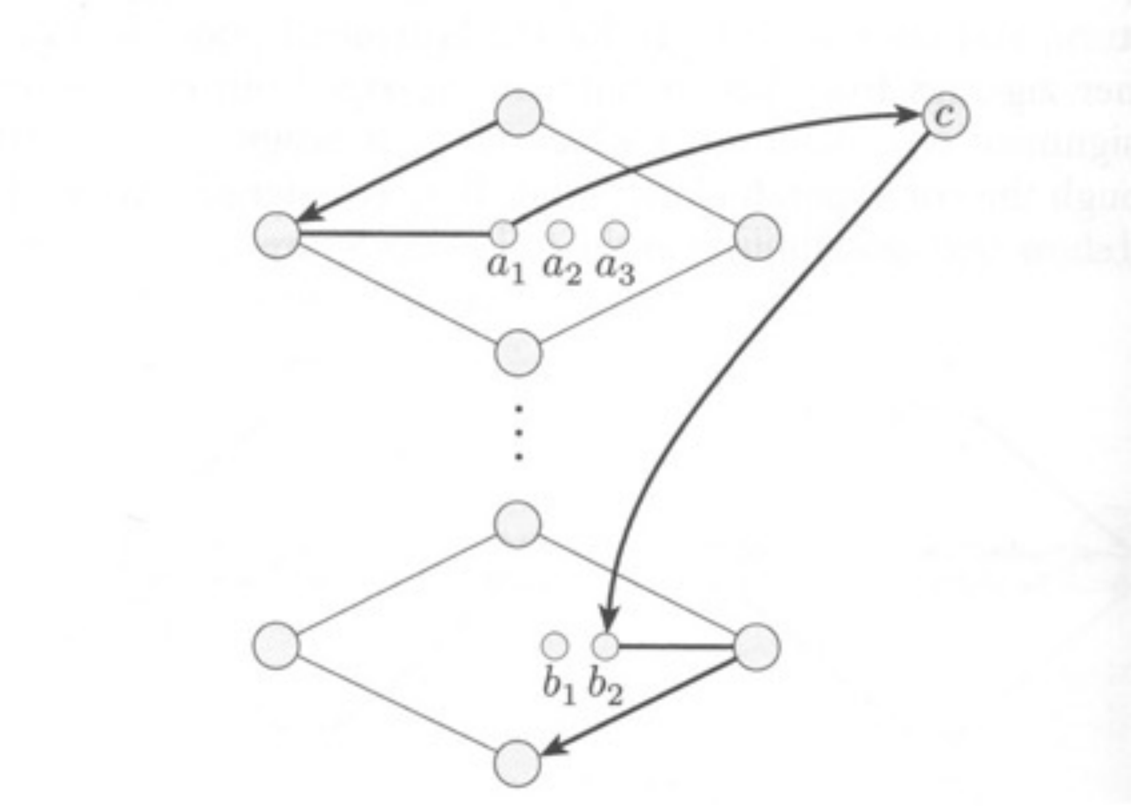
$\downarrow$ $x_1, \ldots, x_\ell$ VARIABLES

G:



SUPPOSE φ SATISFIABLE WITH SOME TRUTH ASSIGNMENT.
ZIG ZAG if x_i is TRUE, ZAG-ZIG IF $\overline{x}_i$ TRUE.
DETOUR ON CLAUSES NOT ALREADY COVERED.

If hamiltonian path were not normal:



Case: a_2 separator node

Only edges entering a_2 would be a_1 and a_3

Case: a_3 separator node. Then a_1, a_2 in same clause pair

Only edges entering a_2 would be a_1, a_3, c

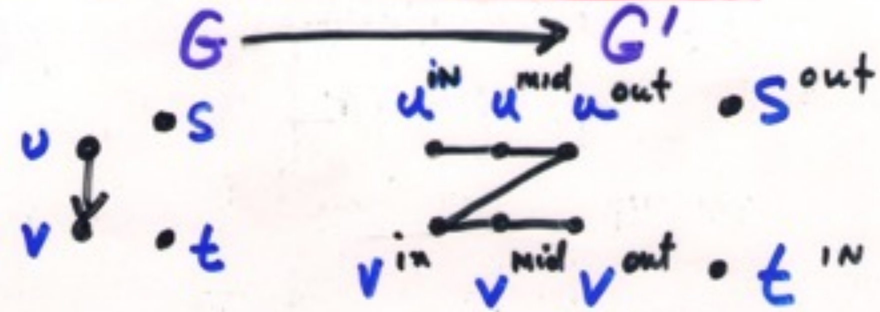
UHAMPATH = { (G,s,t) | G is an undirected graph with a Hamilton path from s to t}

Theorem: UHAMPATH is NP-Complete

(1) UHAMPATH $\in$ NP

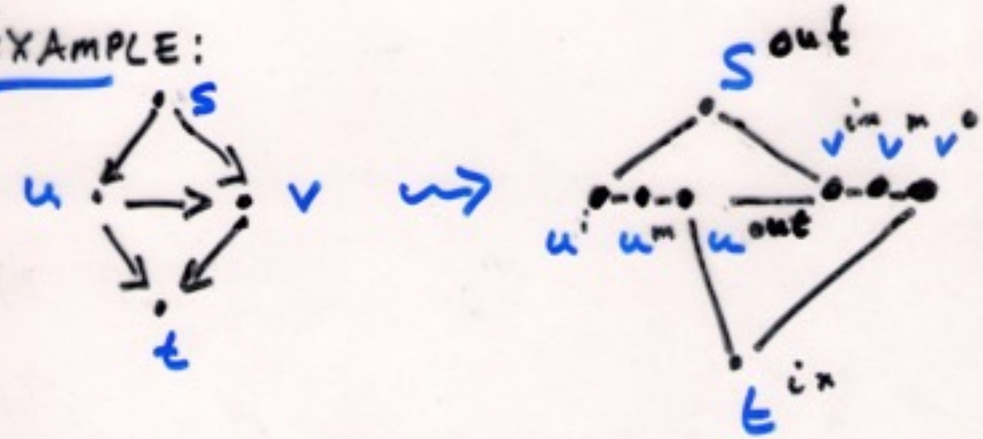
(2) HAMPATH $\leq_P$ UHAMPATH

HAMPATH $\leq_P$ UHAMPATH



Rule: $u \to v$ then $u^{out} — v^{in}$

Example:



- Why do we need mid?

SUBSETSUM = { (S, t) | S is multiset of integers and for some $Y \subseteq S$, we have $\sum_{y \in Y} y = t$ }

Theorem: SUBSETSUM is NP-Complete

(1) SUBSETSUM $\in$ NP

(2) 3SAT $\leq_P$ SUBSETSUM

3SAT $\leq_P$ SUBSET SUM

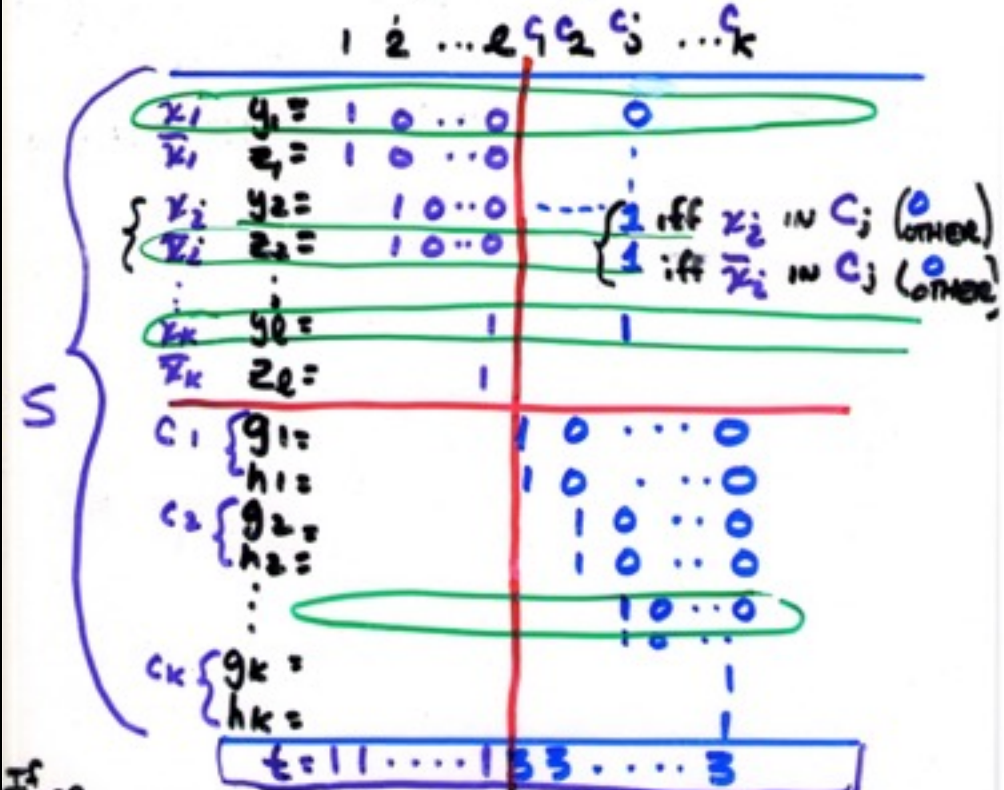
$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$ C_j, CLAUSE

VARIABLES: $x_1, \ldots, x_\ell$

$\downarrow$

(S, t) $S = \{ y_i, z_i, g_j, h_j \mid i = 1, \ldots, \ell;\ j = 1, \ldots, k \}$

$t = \underbrace{11\cdots1}_{\ell}\underbrace{33\cdots3}_{k}$



If φ SATISFIABLE WITH some truth assignment FOR SUBSET CHOOSE ROWS WITH LITERALS TRUE & g_j's & h_j's AS NECESSARY TO ADD UP.

HW

Let G denote a graph, and s and t denote nodes.

SHORTEST PATH
= {(G, s, t, k) |
G has a simple path of length < k from s to t }

LONGEST PATH
= {(G, s, t, k) |
G has a simple path of length > k from s to t }

WHICH IS EASY? WHICH IS HARD? Justify
(see Sipser 7.21)

$$(x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2)$$