15-453

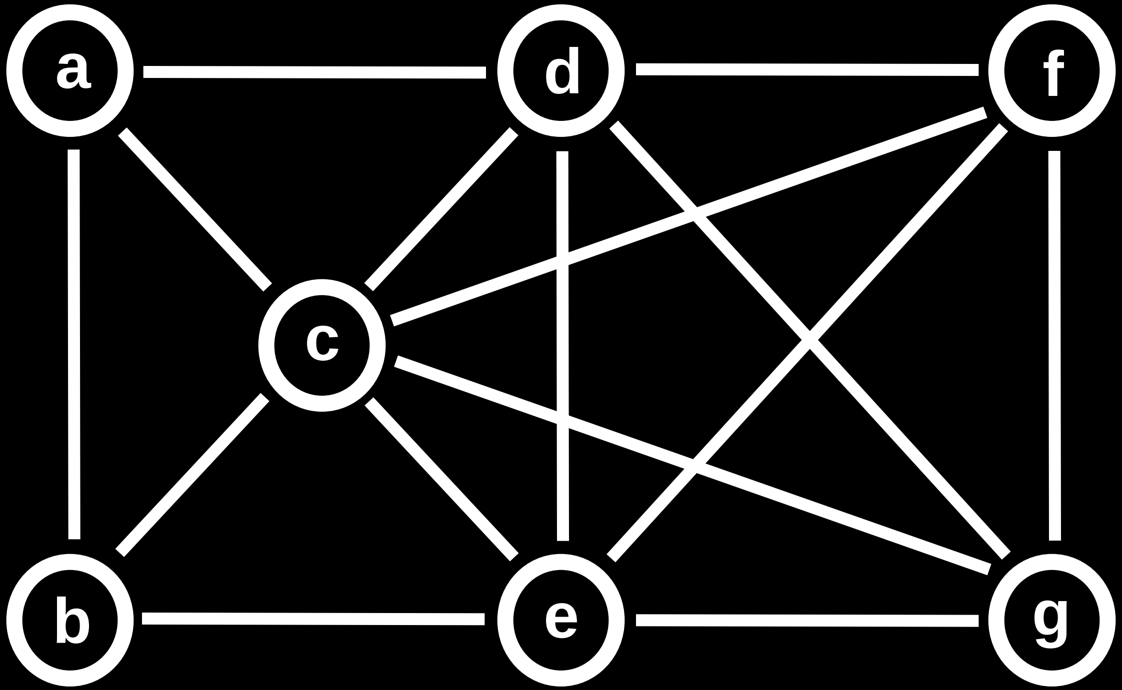
FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY

NP-COMPLETENESS II

Tuesday April 1

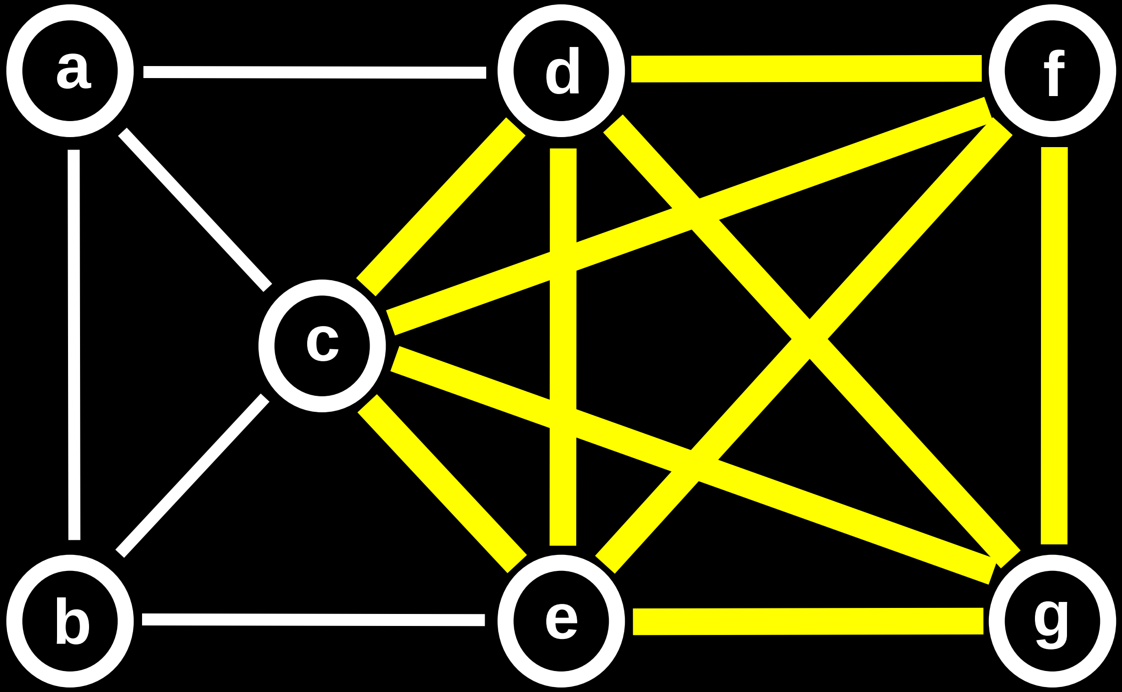
There are googols of NP-complete languages

K-CLIQUE



k-clique = complete subgraph of k nodes

K-CLIQUE



k-clique = complete subgraph of k nodes

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

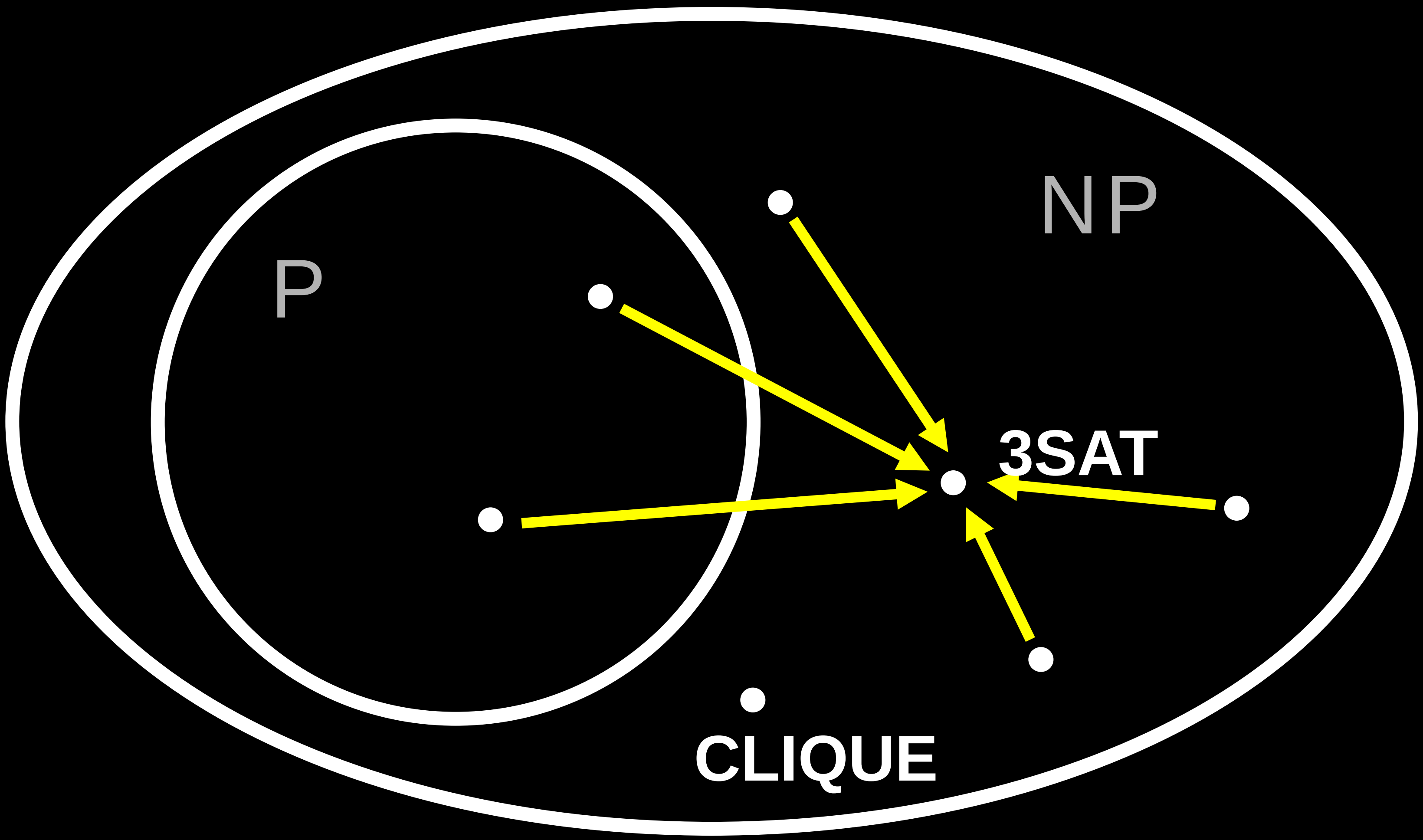
CLIQUE = { (G,k) | G is an undirected graph with a k-clique }

Theorem: CLIQUE is NP-Complete

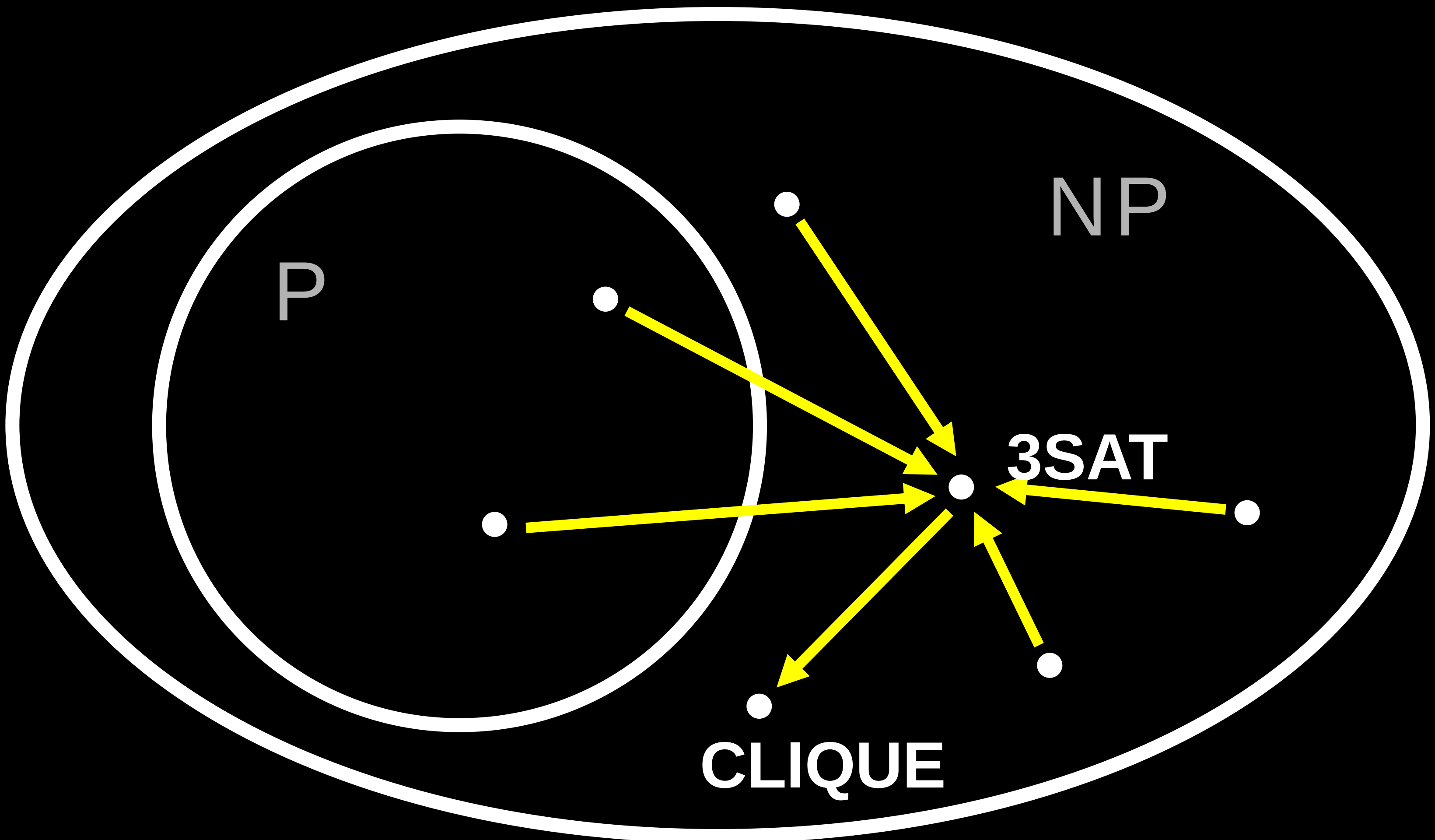
(1) CLIQUE $\in$ NP

(2) 3SAT $\leq_P$ CLIQUE

CLIQUE is NP-Complete



CLIQUE is NP-Complete



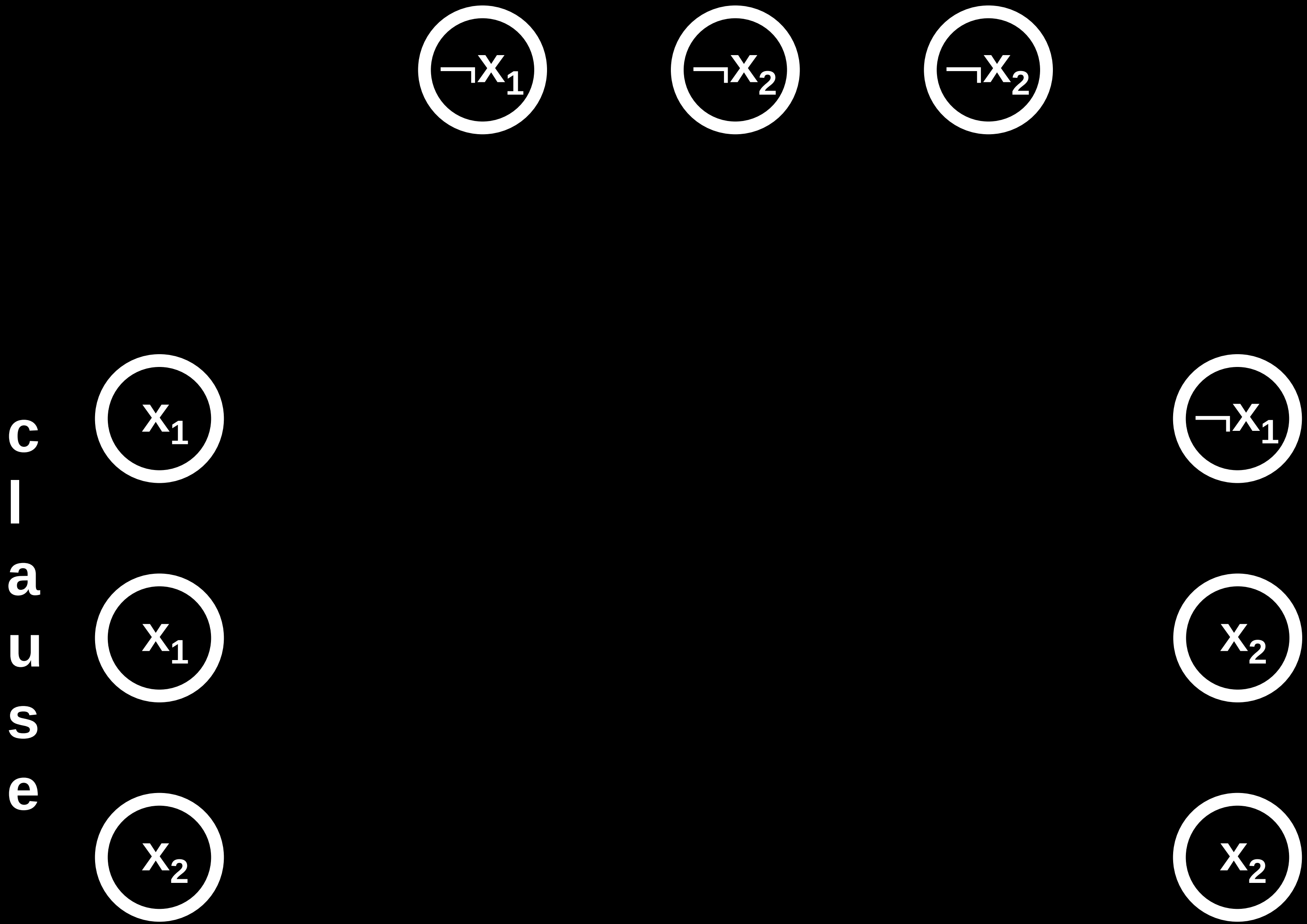
$3SAT \leq_P CLIQUE$

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

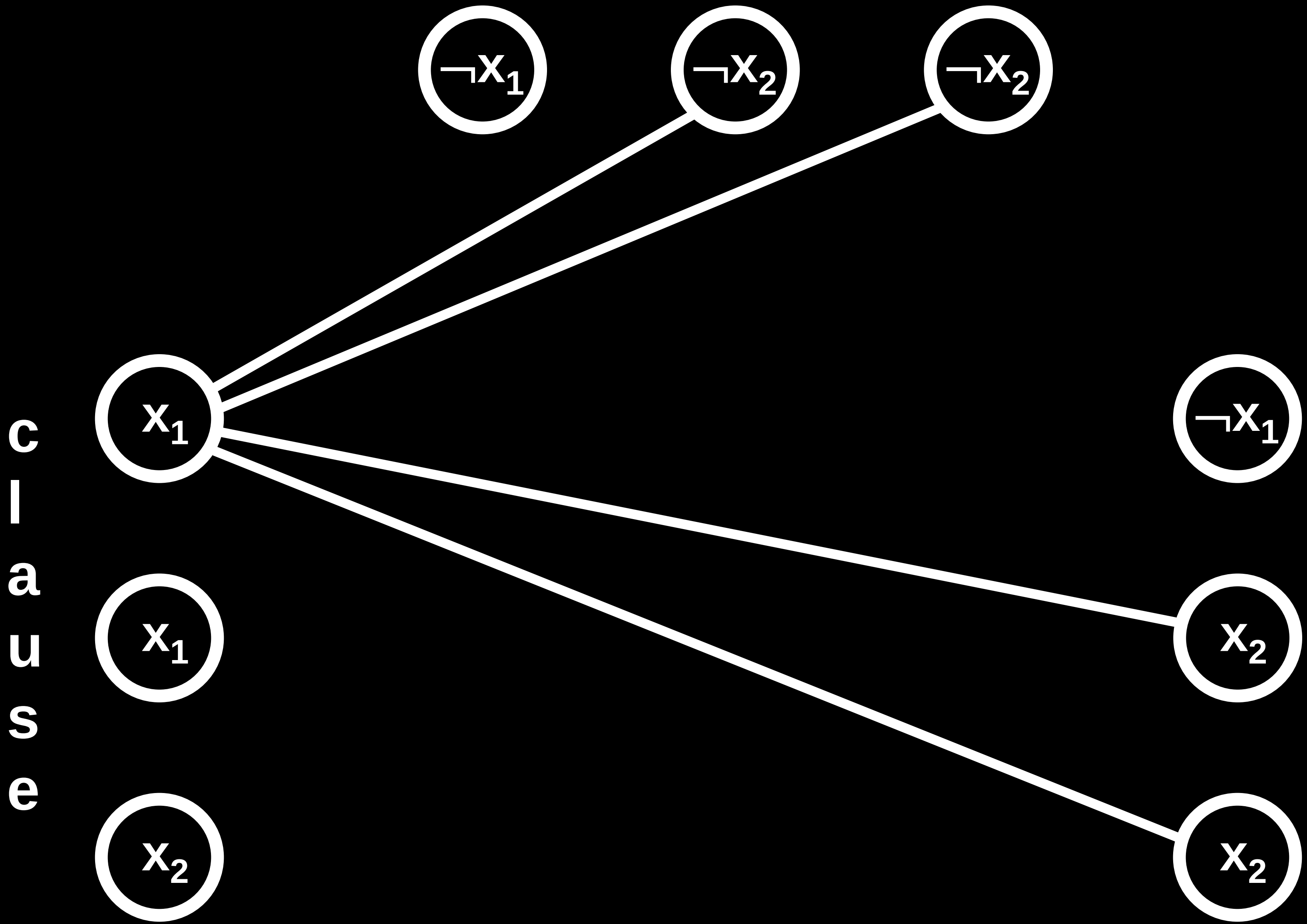
$(x_1 \lor x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2 \lor \neg x_2) \land (\neg x_1 \lor x_2 \lor x_2)$



#nodes = 3(# clauses)

k = #clauses

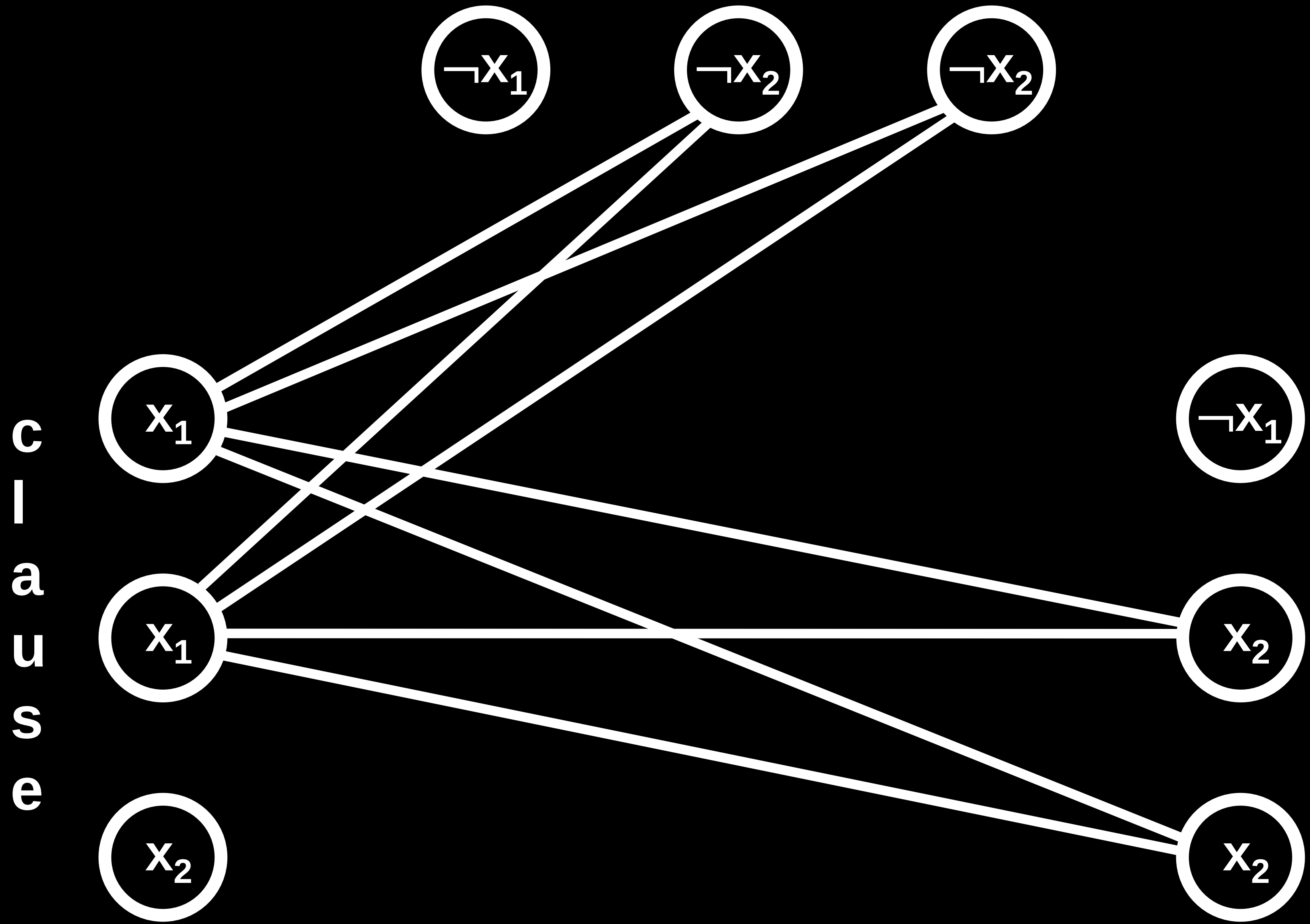
$$(x_1 \lor x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2 \lor \neg x_2) \land (\neg x_1 \lor x_2 \lor x_2)$$



#nodes = 3(# clauses)

k = #clauses

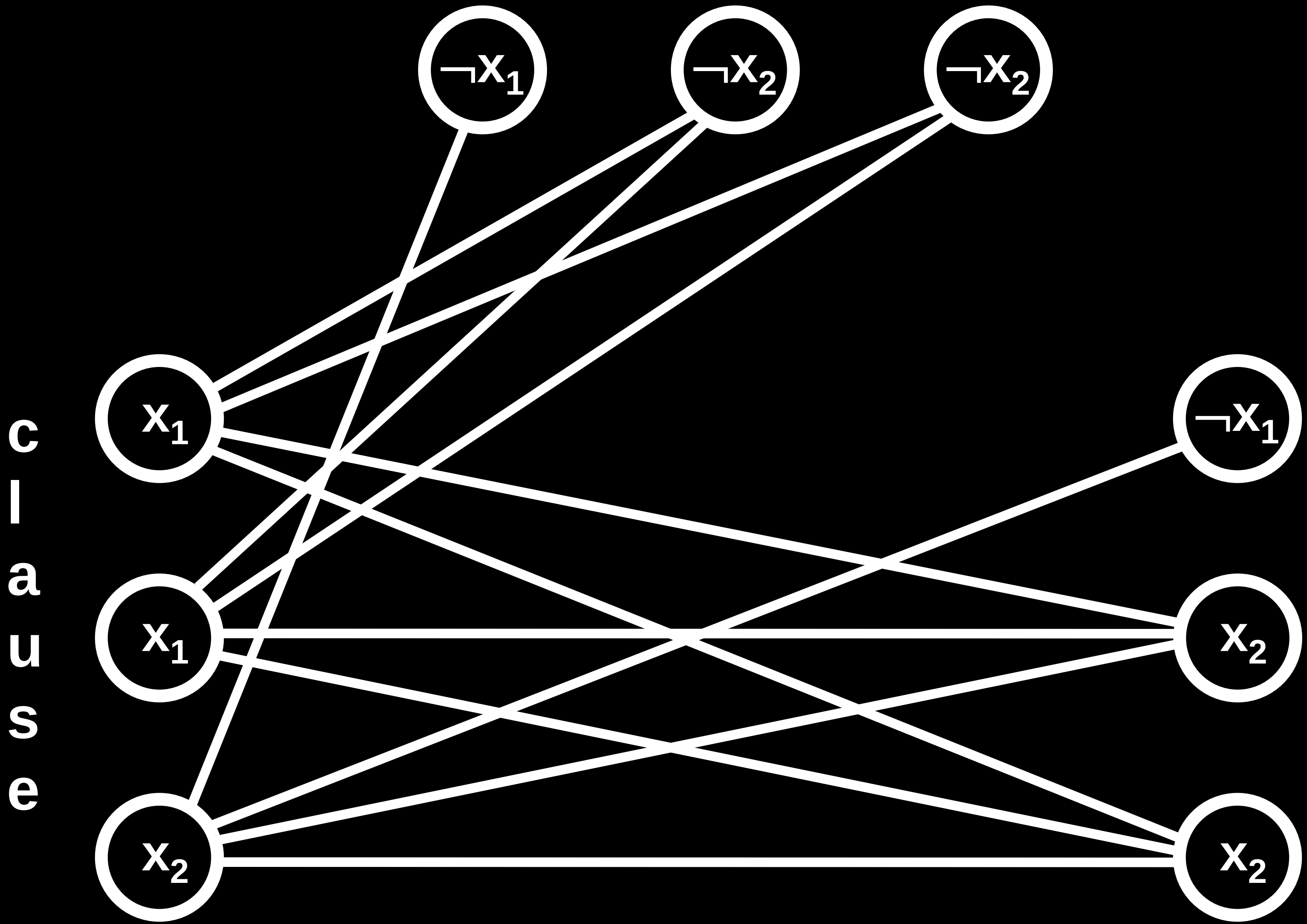
$$(x_1 \lor x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2 \lor \neg x_2) \land (\neg x_1 \lor x_2 \lor x_2)$$



#nodes = 3(# clauses)

k = #clauses

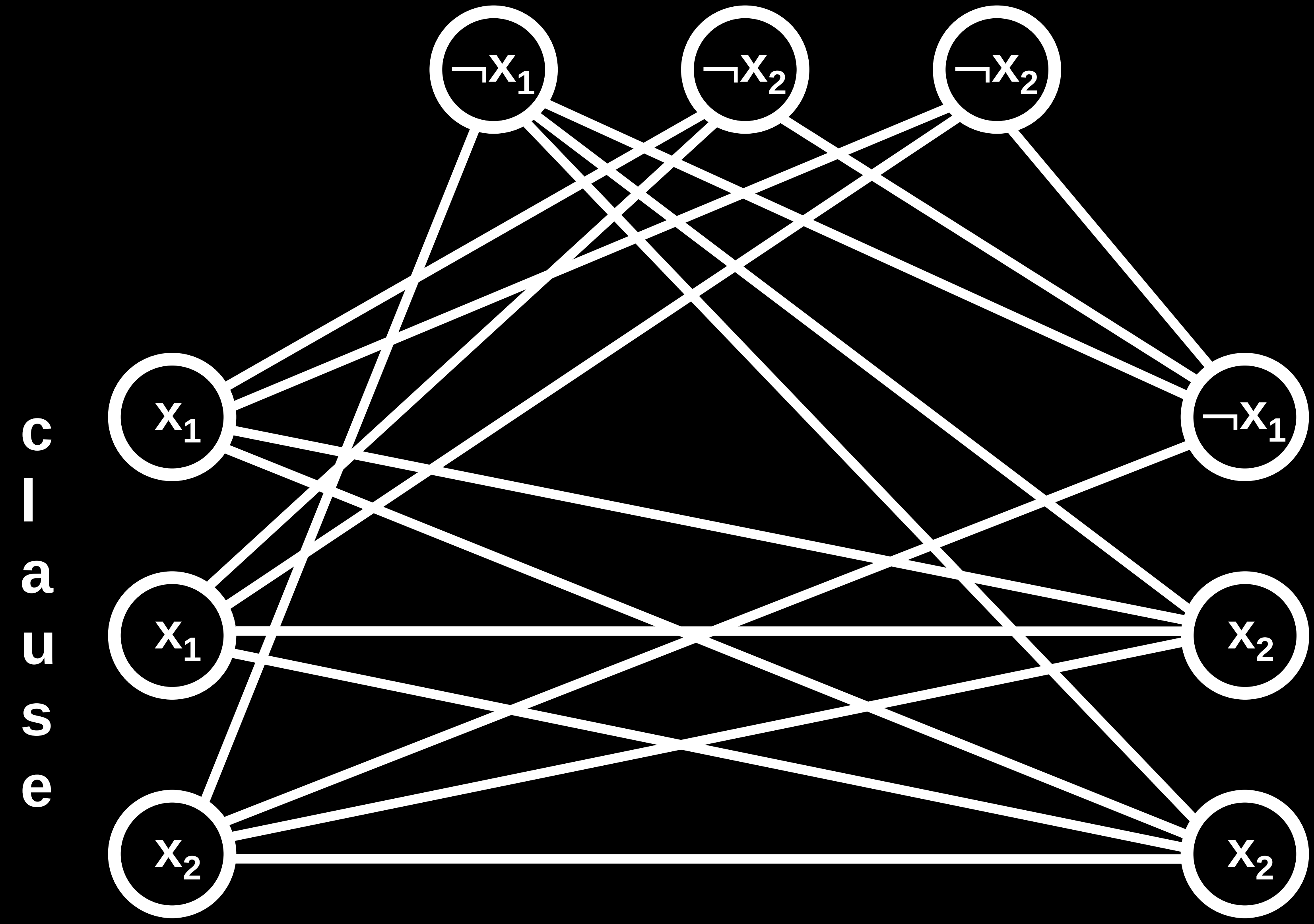
$(x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_2)$



#nodes = 3(# clauses)

k = #clauses

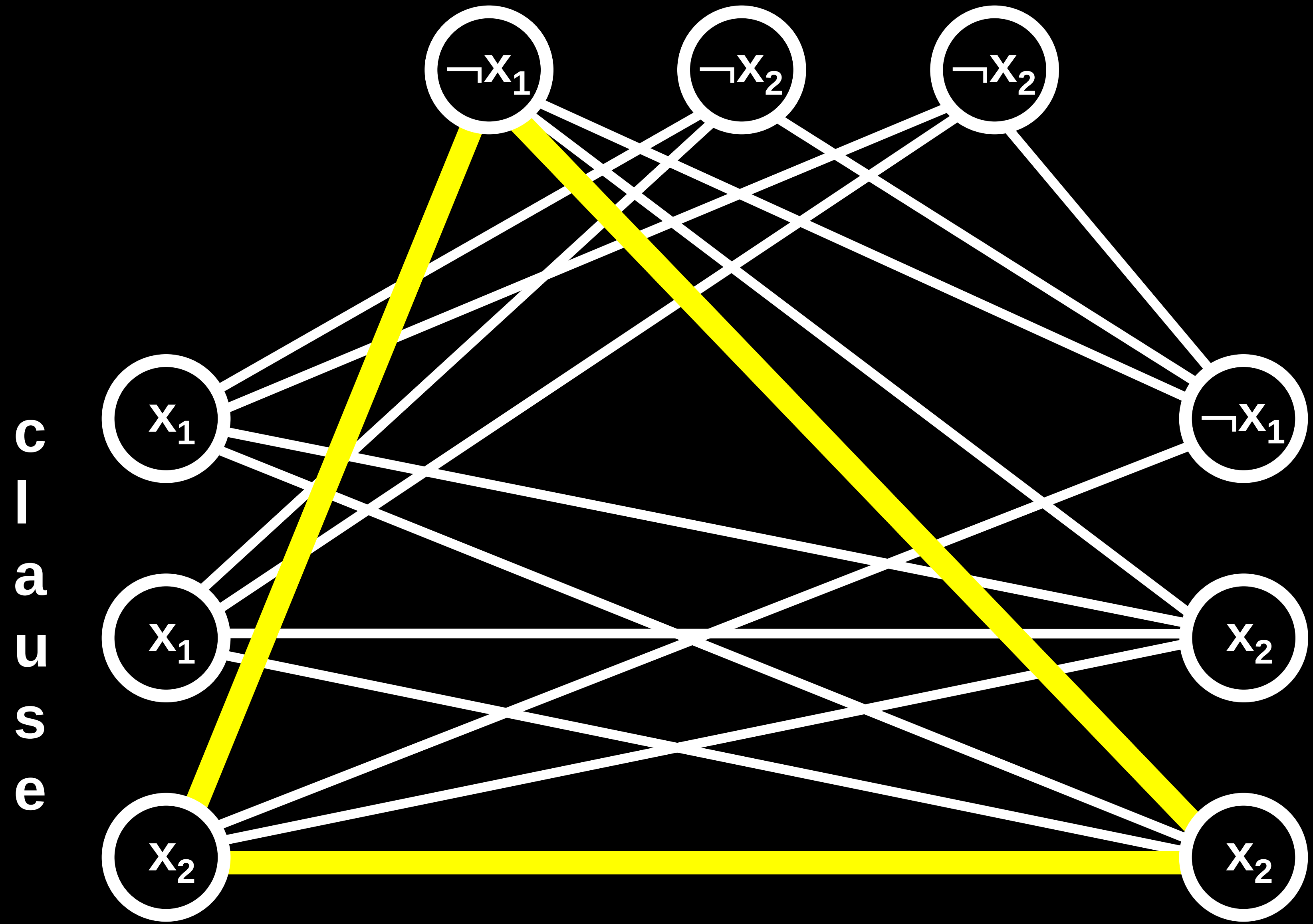
$$(x_1 \lor x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2 \lor \neg x_2) \land (\neg x_1 \lor x_2 \lor x_2)$$



#nodes = 3(# clauses)

k = #clauses

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



#nodes = 3(# clauses)

k = #clauses

3SAT $\leq_P$ CLIQUE

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

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

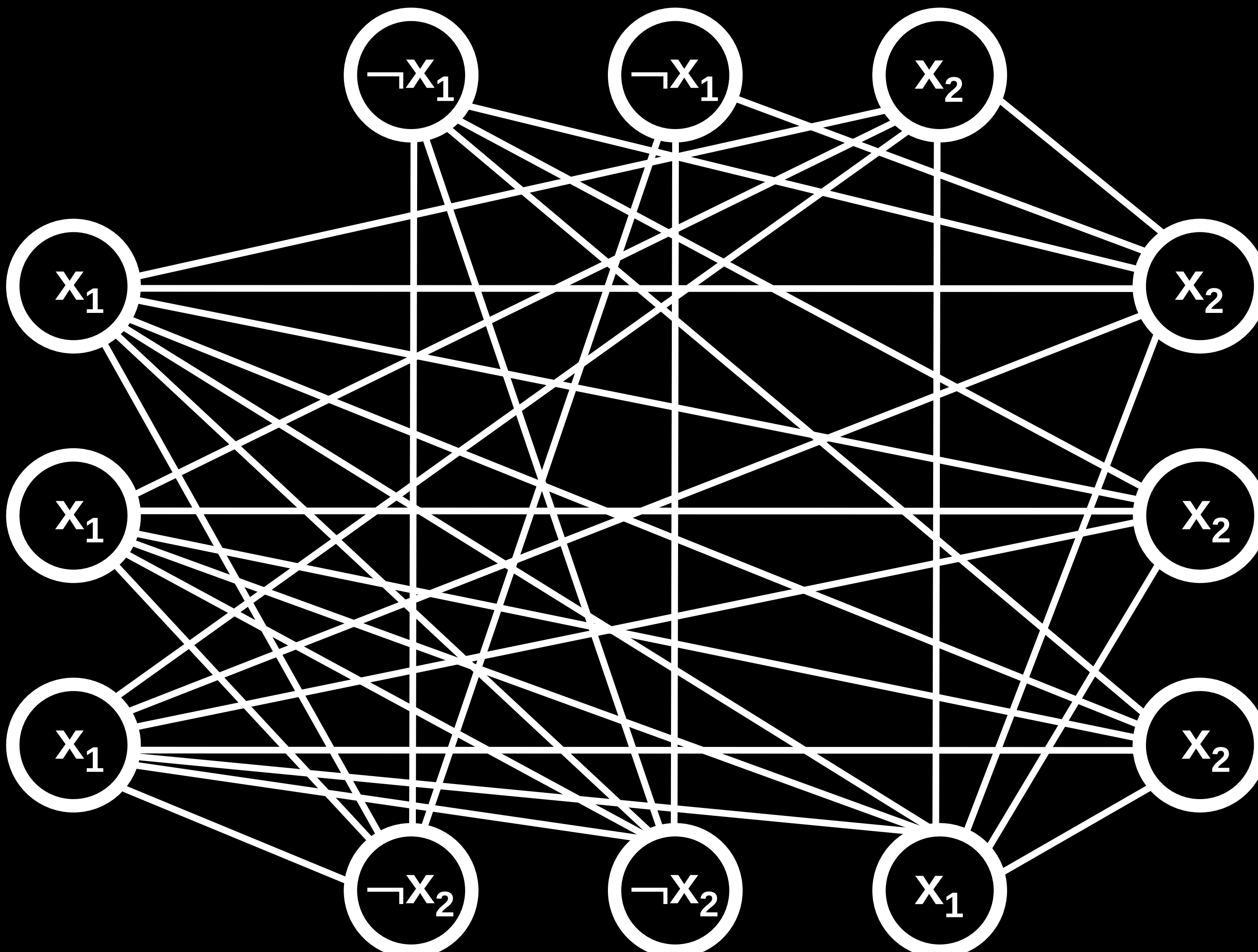
- **If ϕ has m clauses, we create a graph with m clusters of 3 nodes each, and set k=m**
- **Each cluster corresponds to a clause.**
- **Each node in a cluster is labeled with a literal from the clause.**

- **We do not connect any nodes in the same cluster**

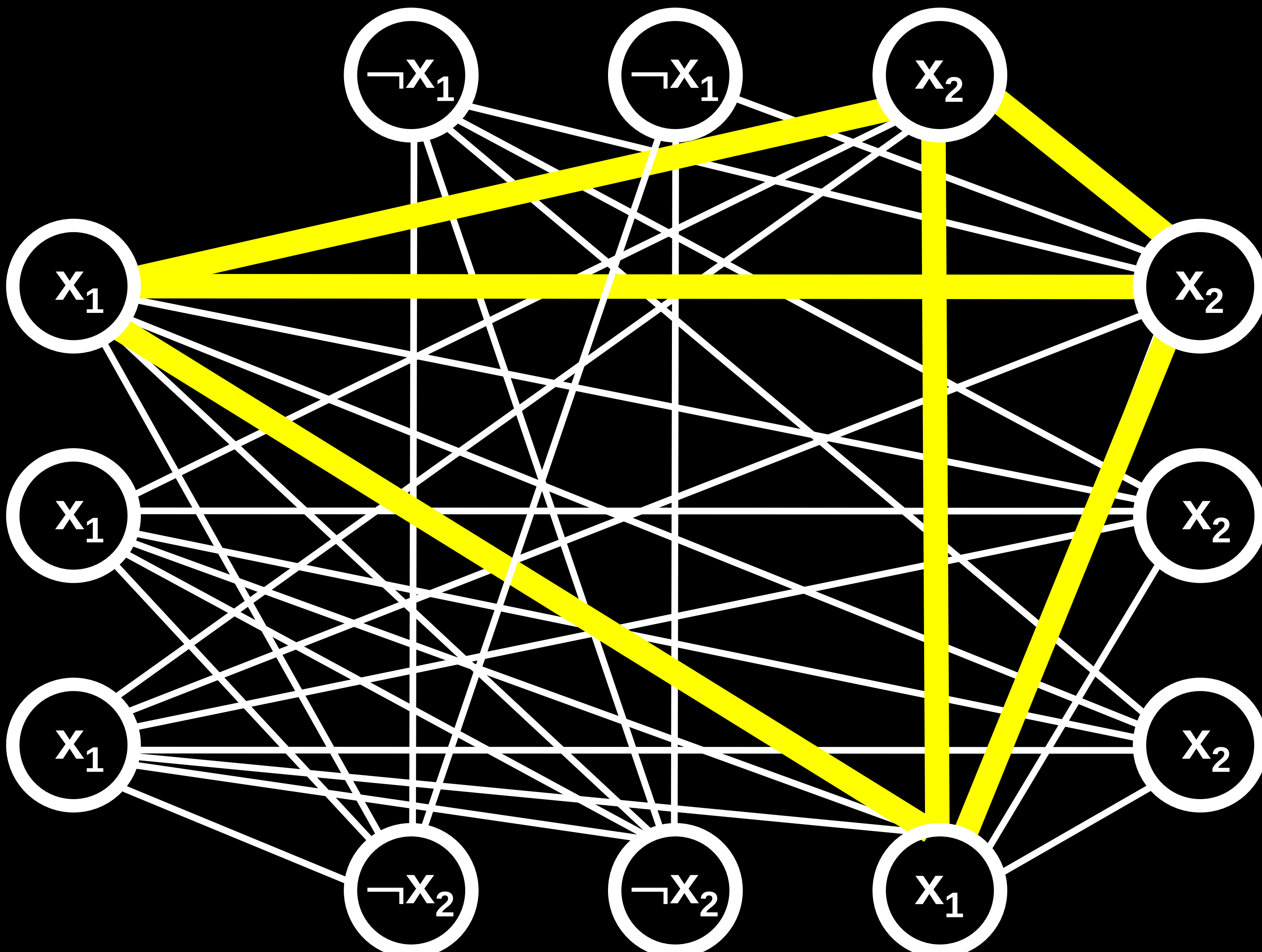
- **We connect nodes in different clusters whenever they are not contradictory**

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

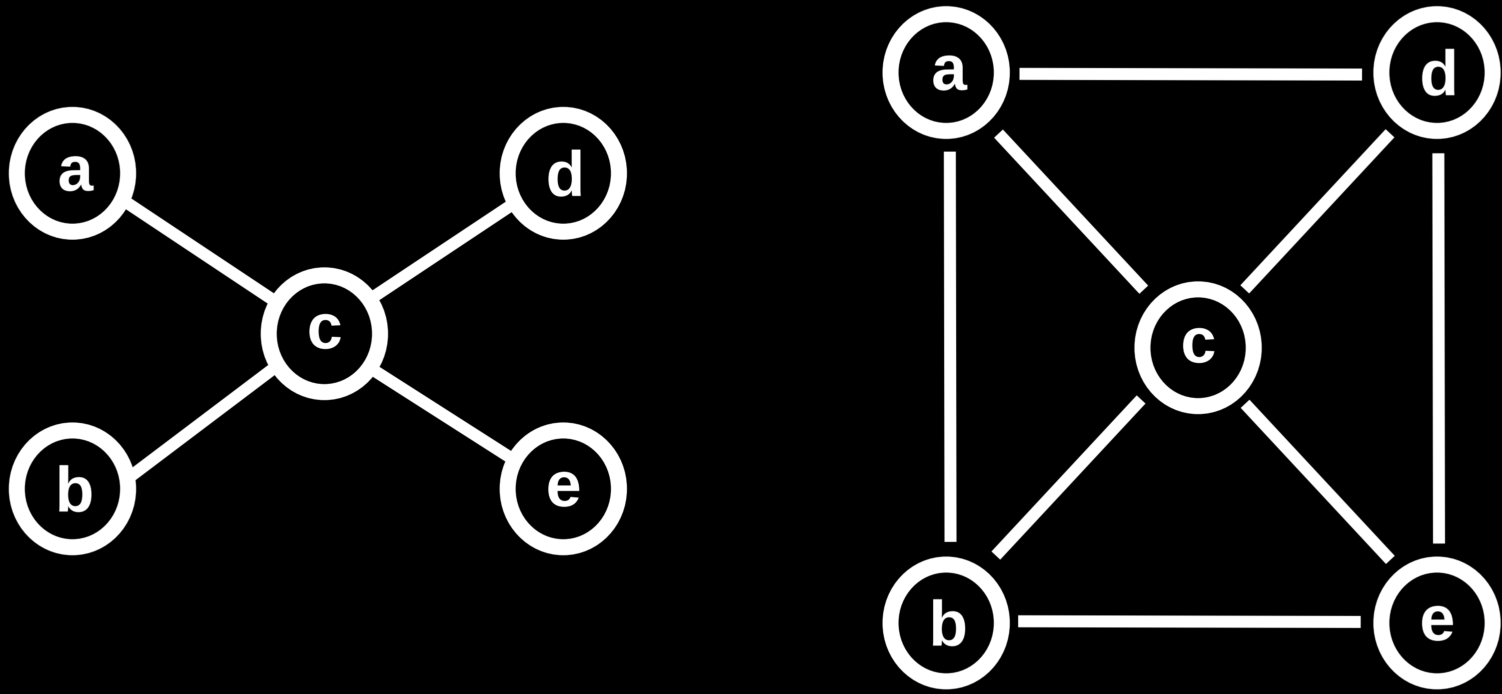
$$(x_1 \lor x_1 \lor x_1) \land (\neg x_1 \lor \neg x_1 \lor x_2) \land$$
$$(x_2 \lor x_2 \lor x_2) \land (\neg x_2 \lor \neg x_2 \lor x_1)$$



$$(x_1 \lor x_1 \lor x_1) \land (\neg x_1 \lor \neg x_1 \lor x_2) \land$$
$$(x_2 \lor x_2 \lor x_2) \land (\neg x_2 \lor \neg x_2 \lor x_1)$$

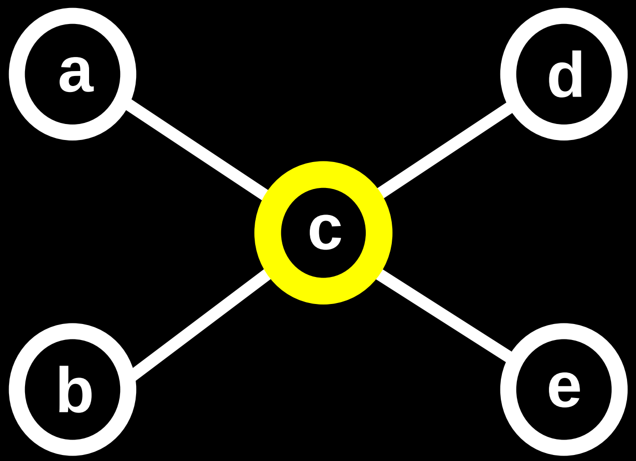


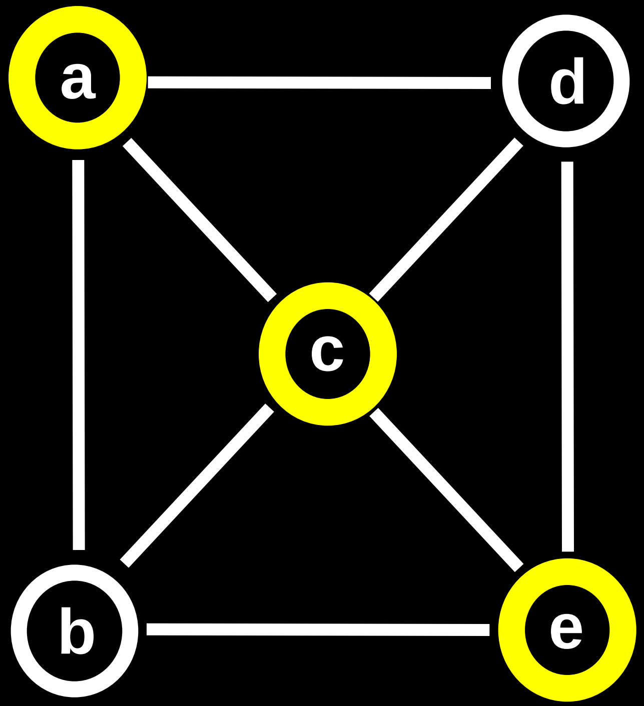
VERTEX COVER



vertex cover = set of nodes that cover all edges

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

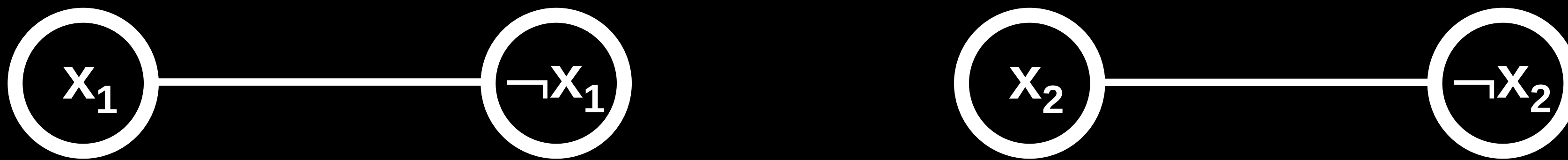
We transform a 3-cnf formula ϕ into (G,k) such that

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

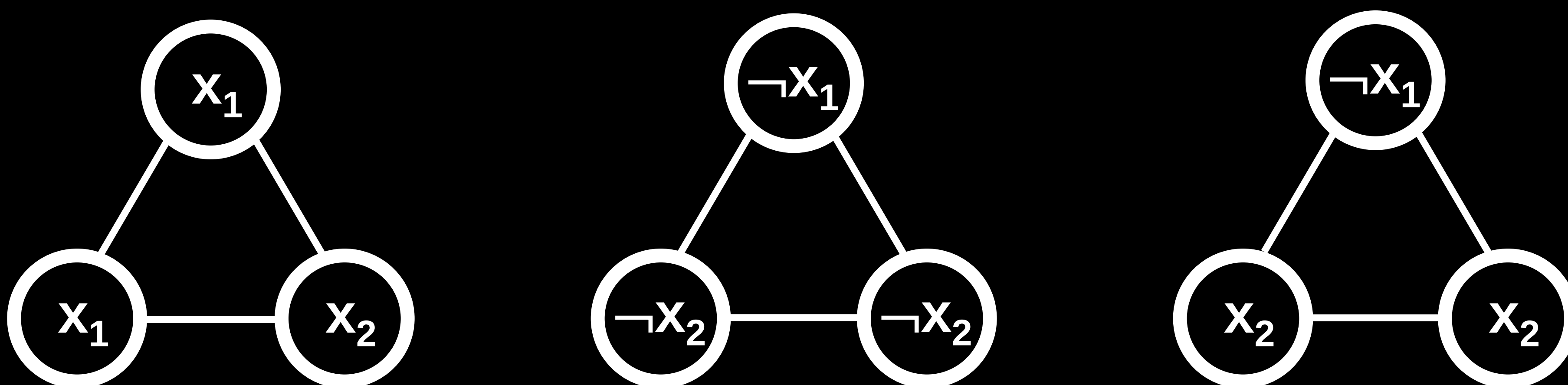
The transformation can be done in time polynomial in the length of ϕ

$$(x_1 \lor x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2 \lor \neg x_2) \land (\neg x_1 \lor x_2 \lor x_2)$$

Variables and negations of variables



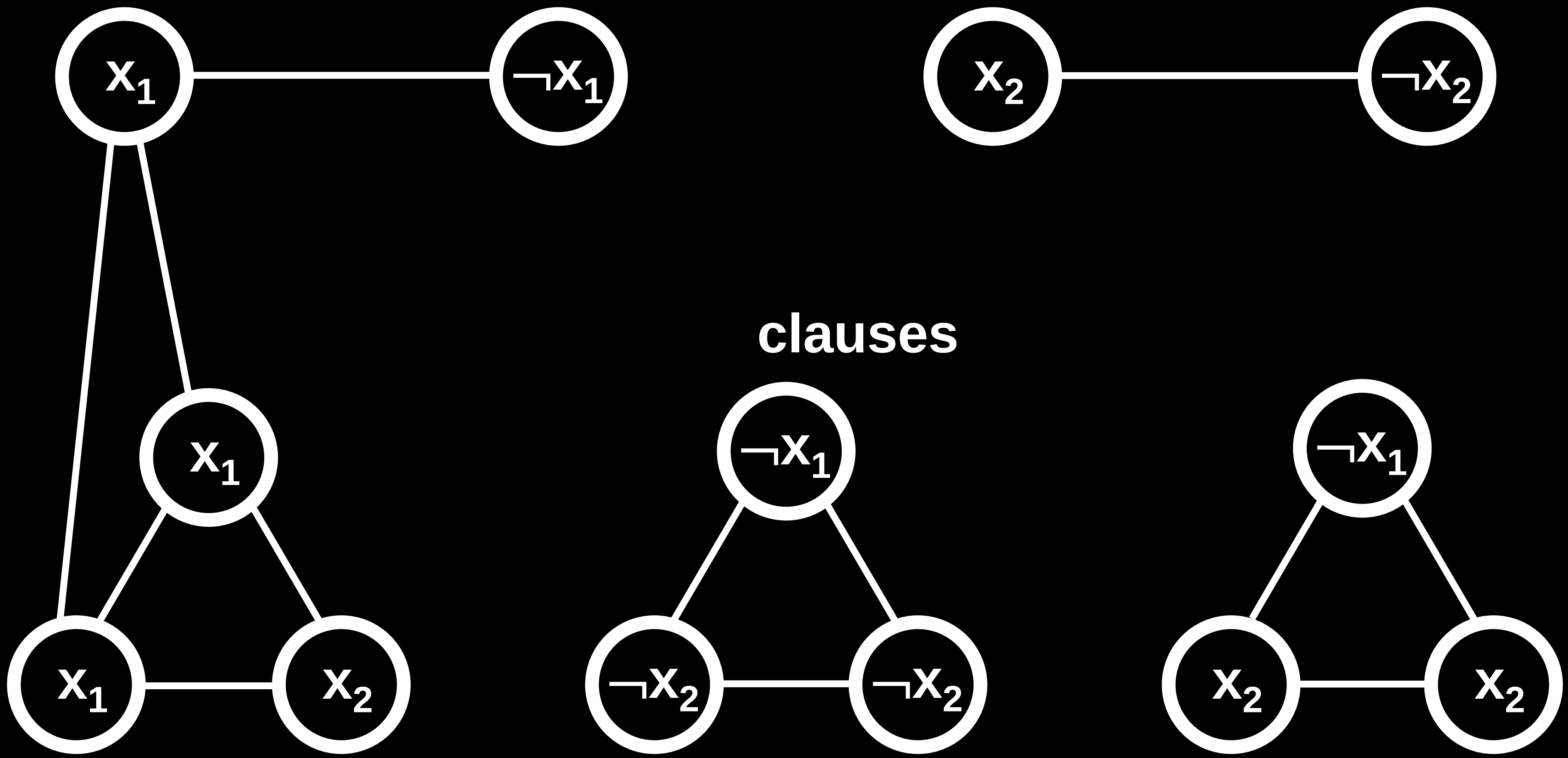
clauses



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

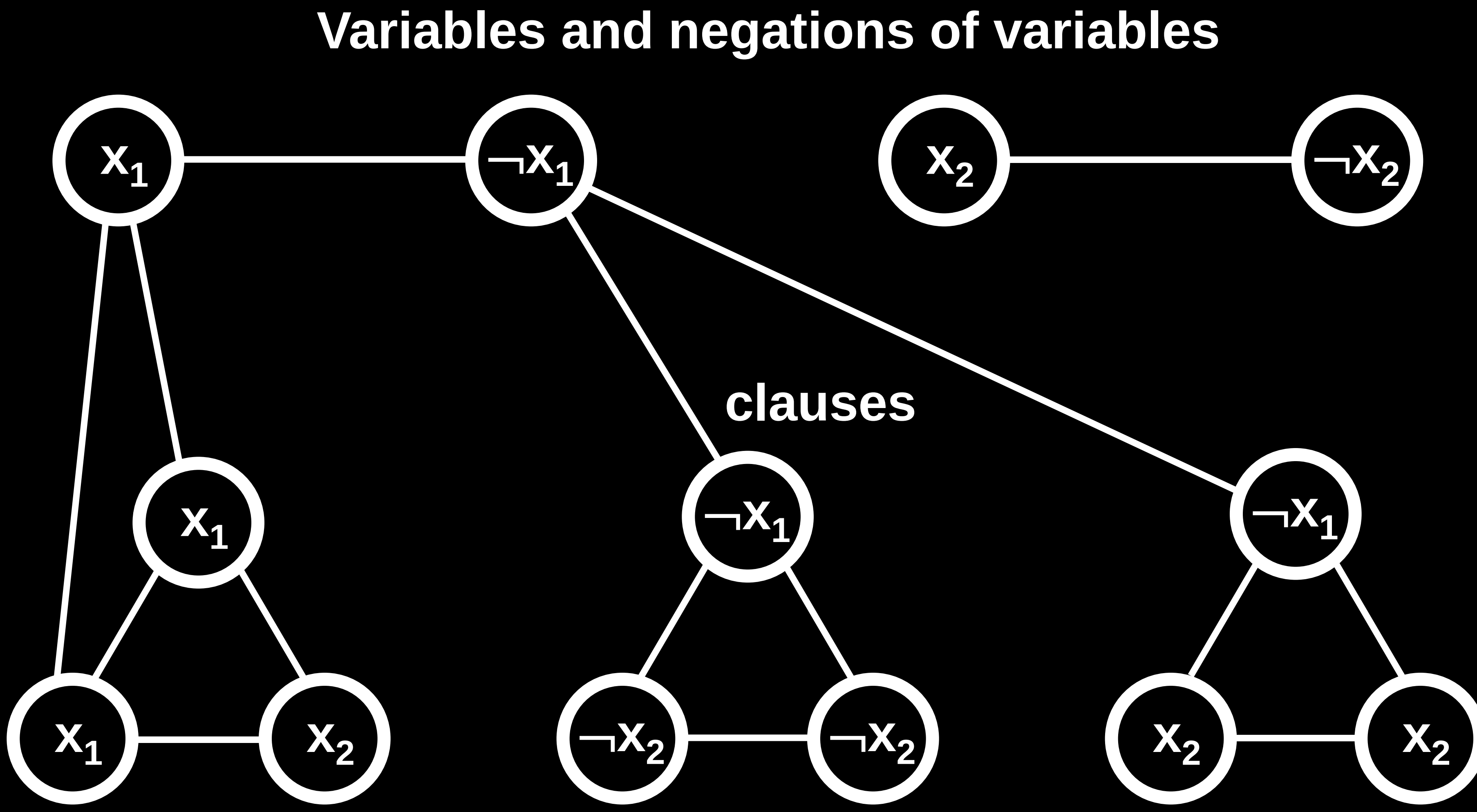
$$(x_1 \lor x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2 \lor \neg x_2) \land (\neg x_1 \lor x_2 \lor x_2)$$

Variables and negations of variables



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

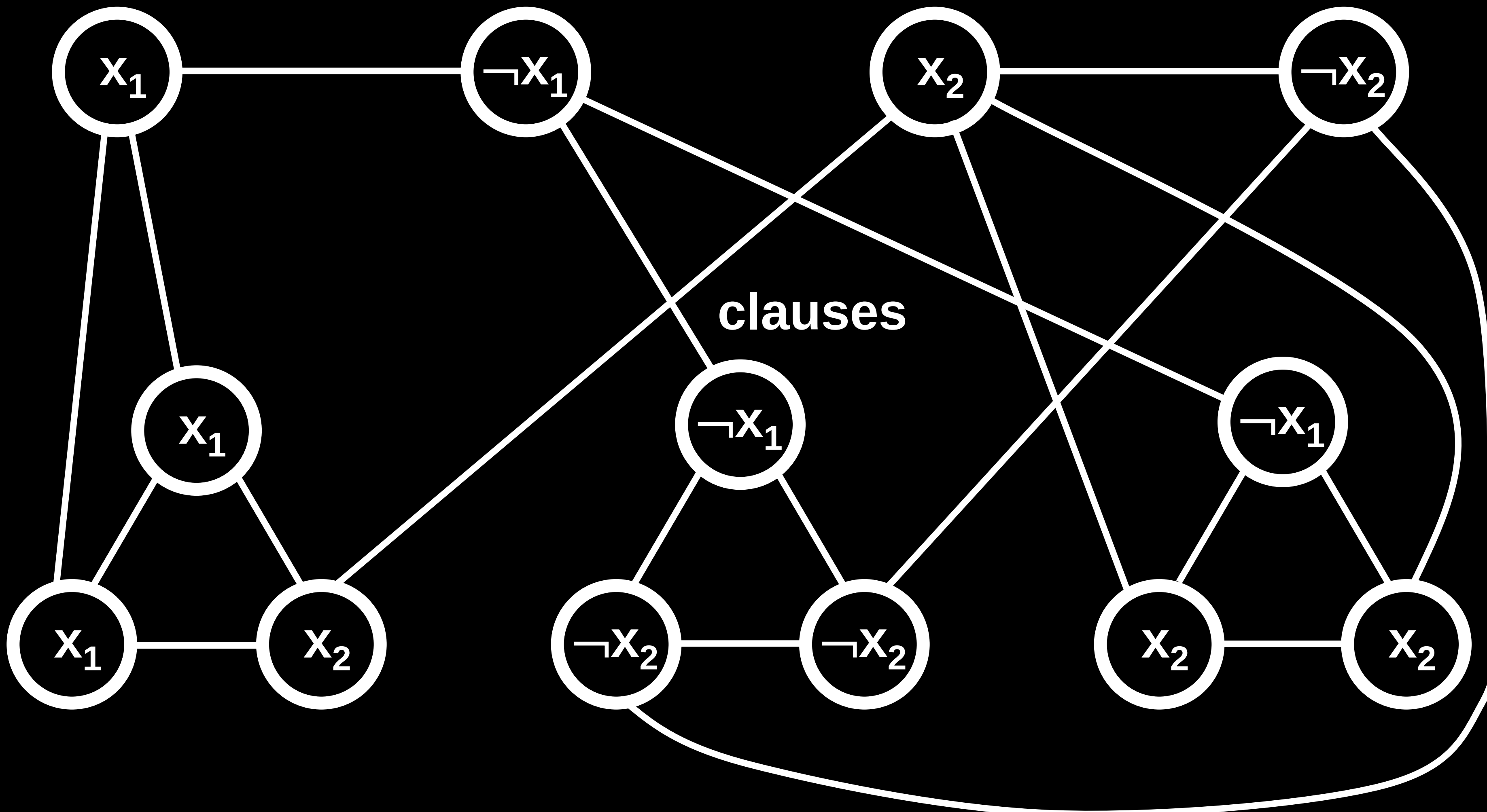
$(x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_2)$



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

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

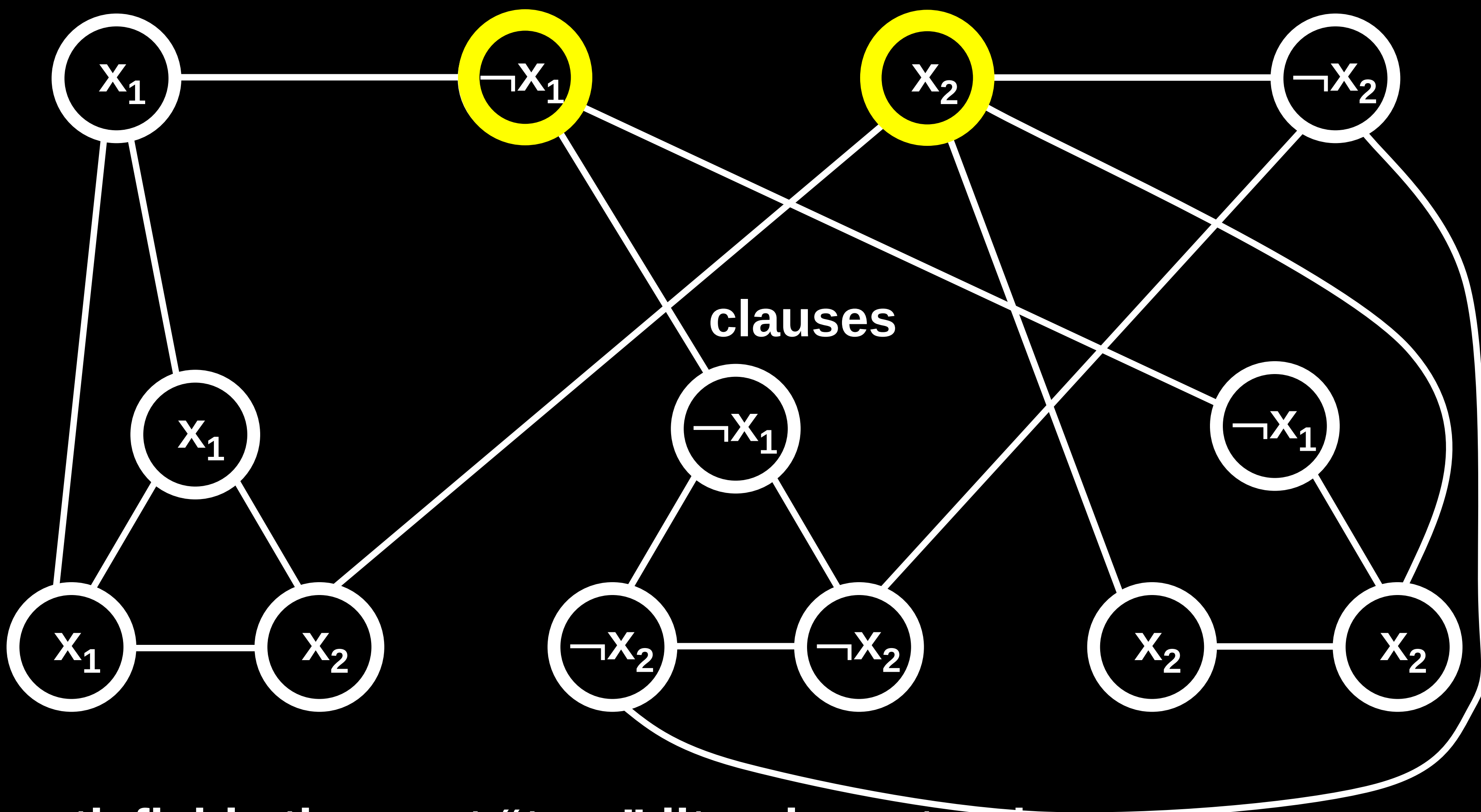
Variables and negations of variables



k = 2(#clauses) + (#variables)

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

Variables and negations of variables

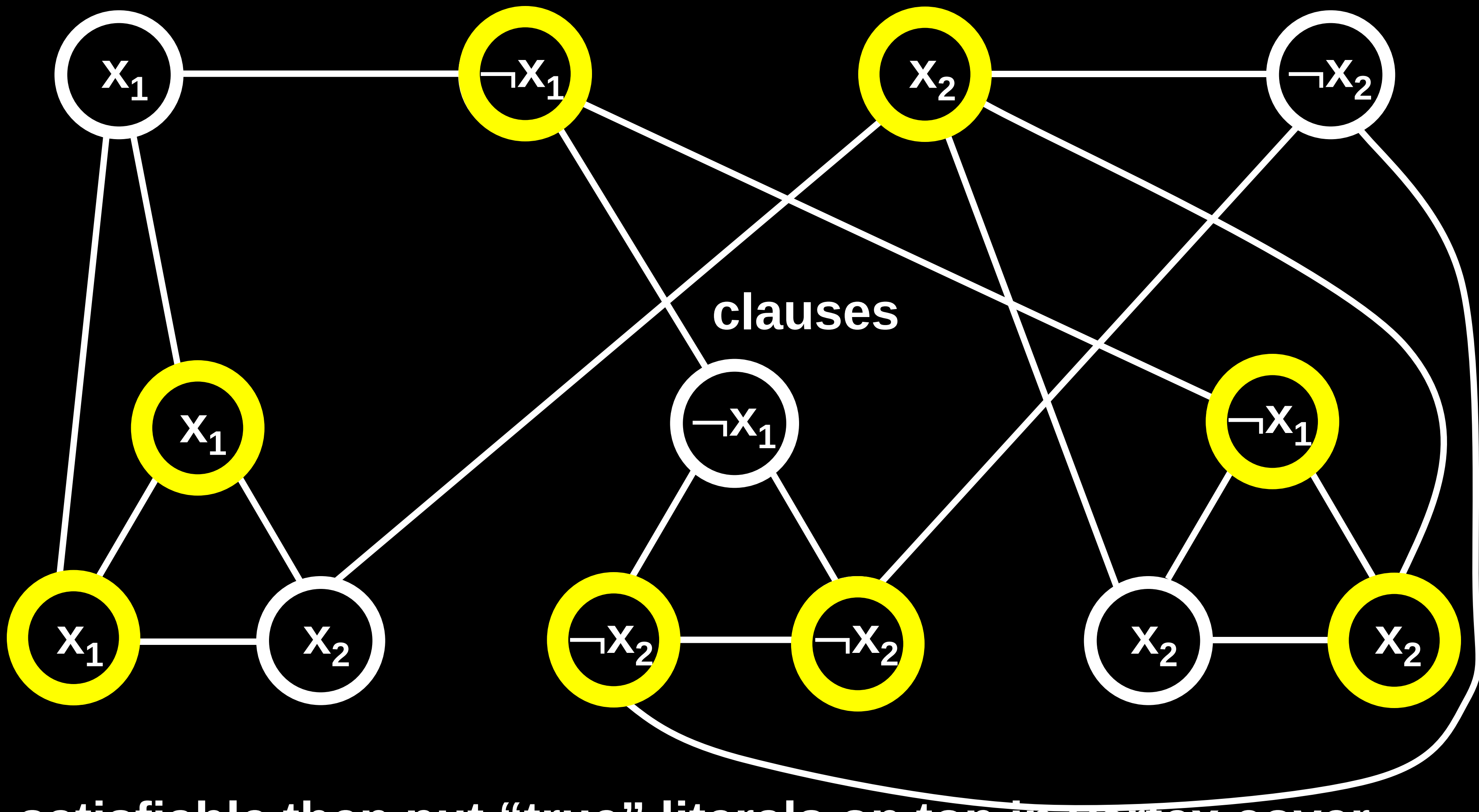


ϕ satisfiable then put "true" literals on top in vertex cover

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

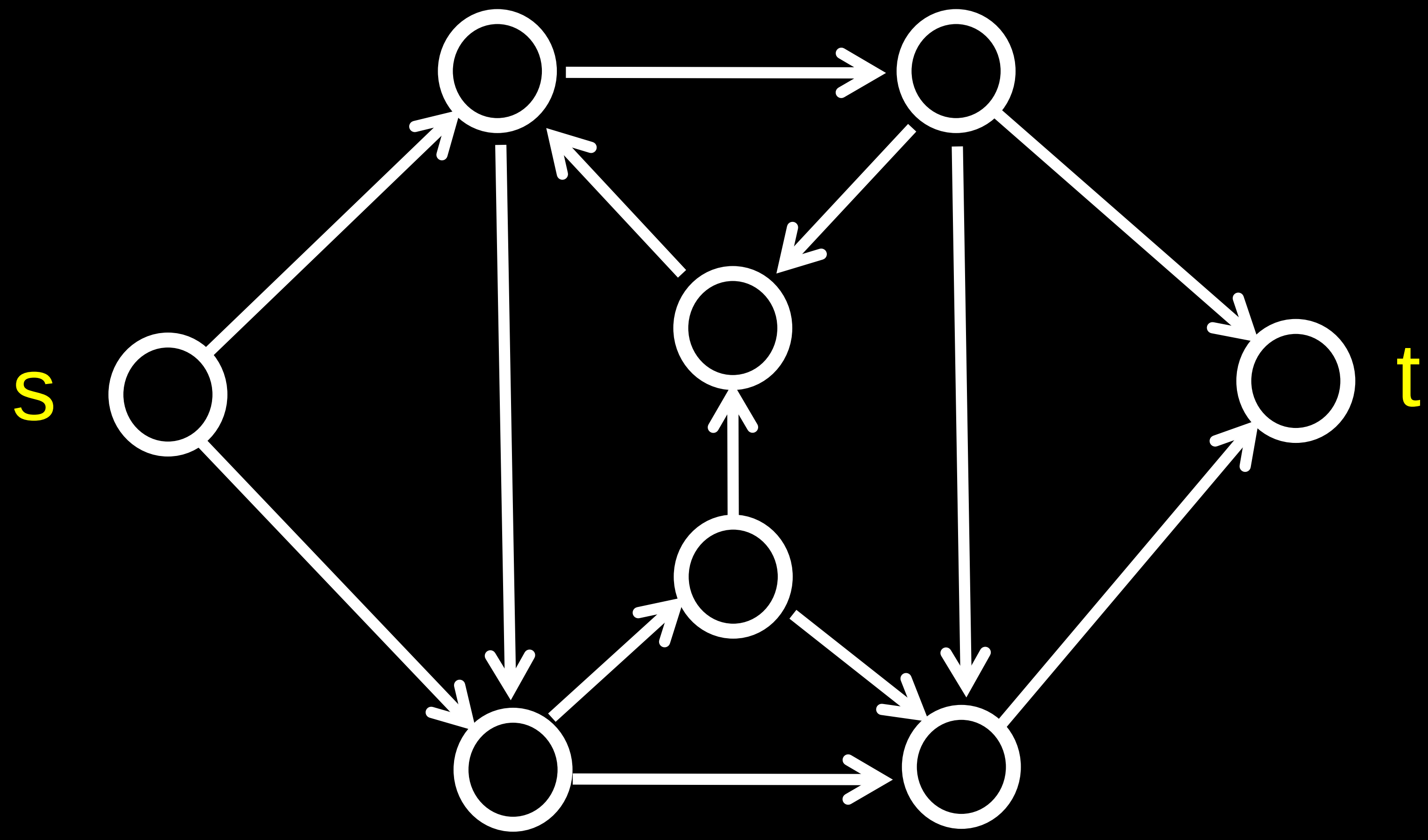


ϕ satisfiable then put "true" literals on top in vertex cover

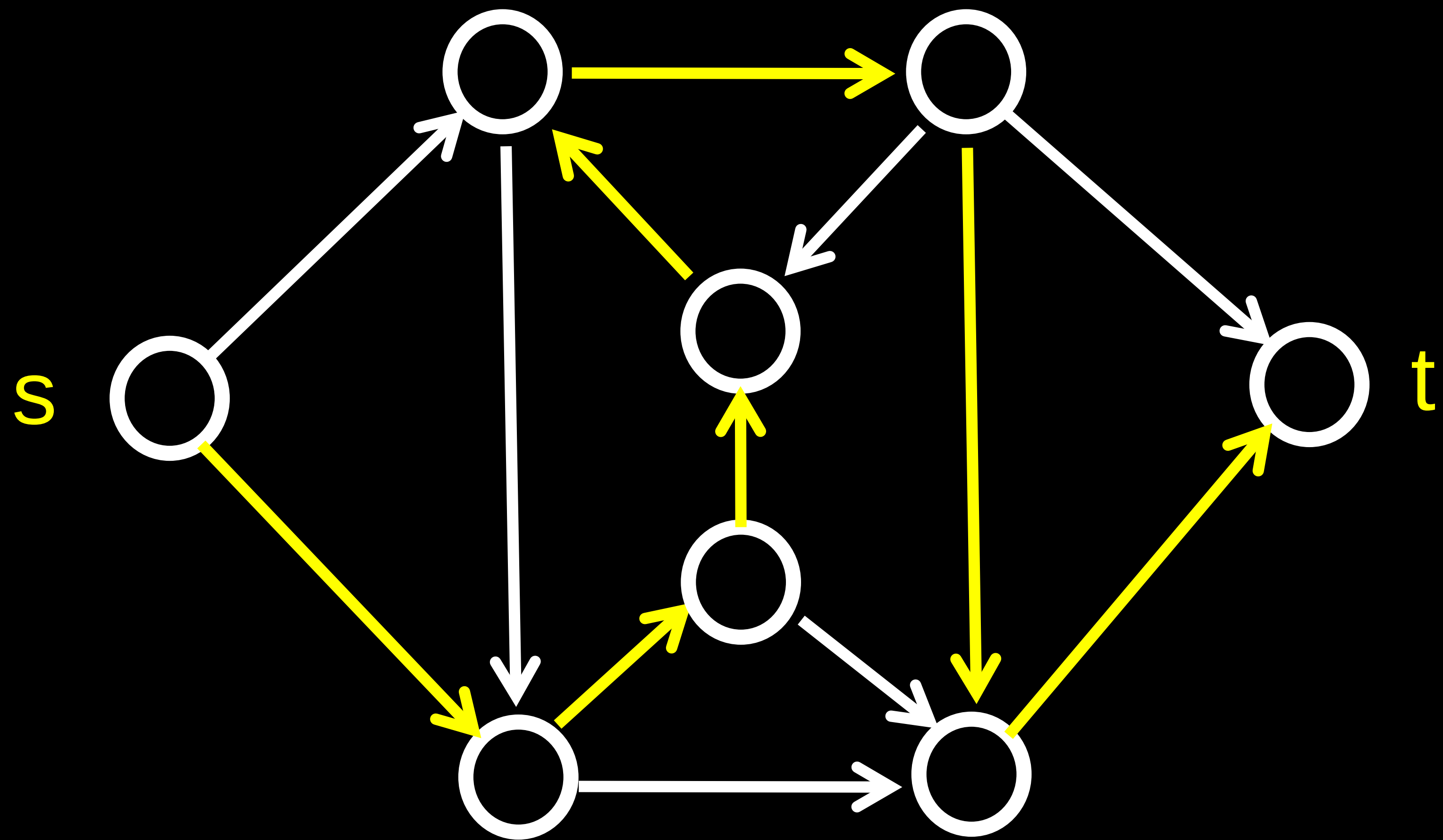
For each clause, pick a true literal and put other 2 in vertex cover

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

HAMILTON PATH



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

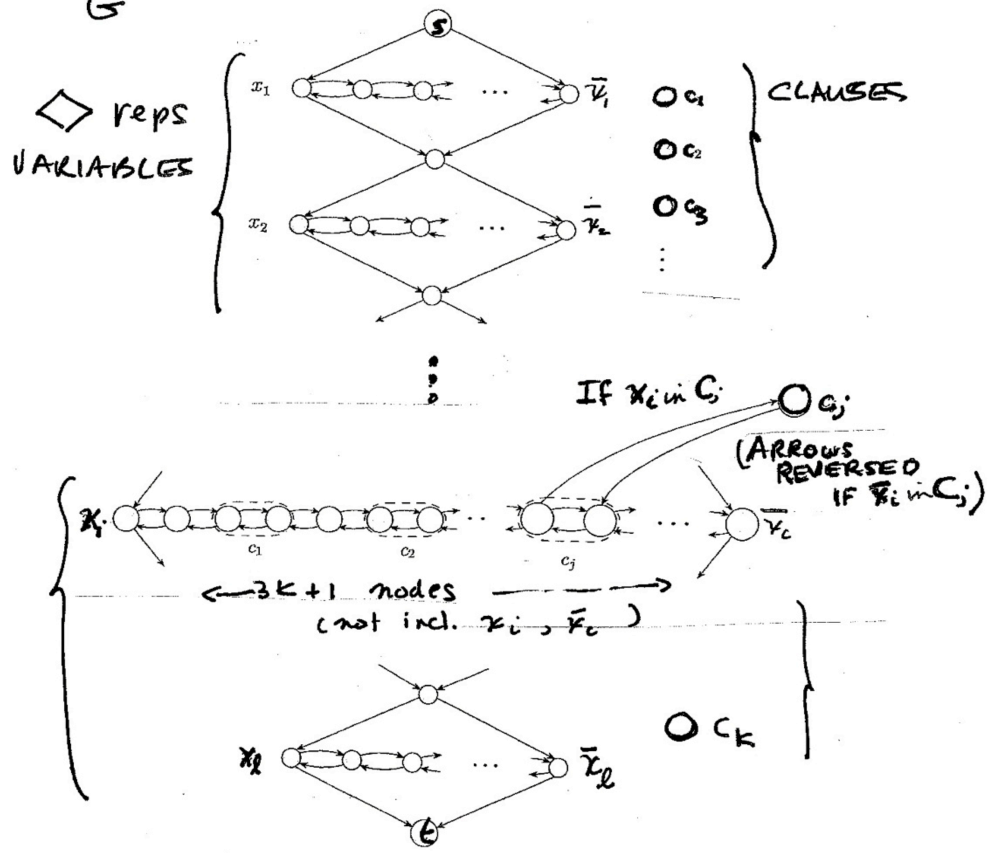
$3SAT \leq_p HAMPATH$

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

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

$\downarrow$

G

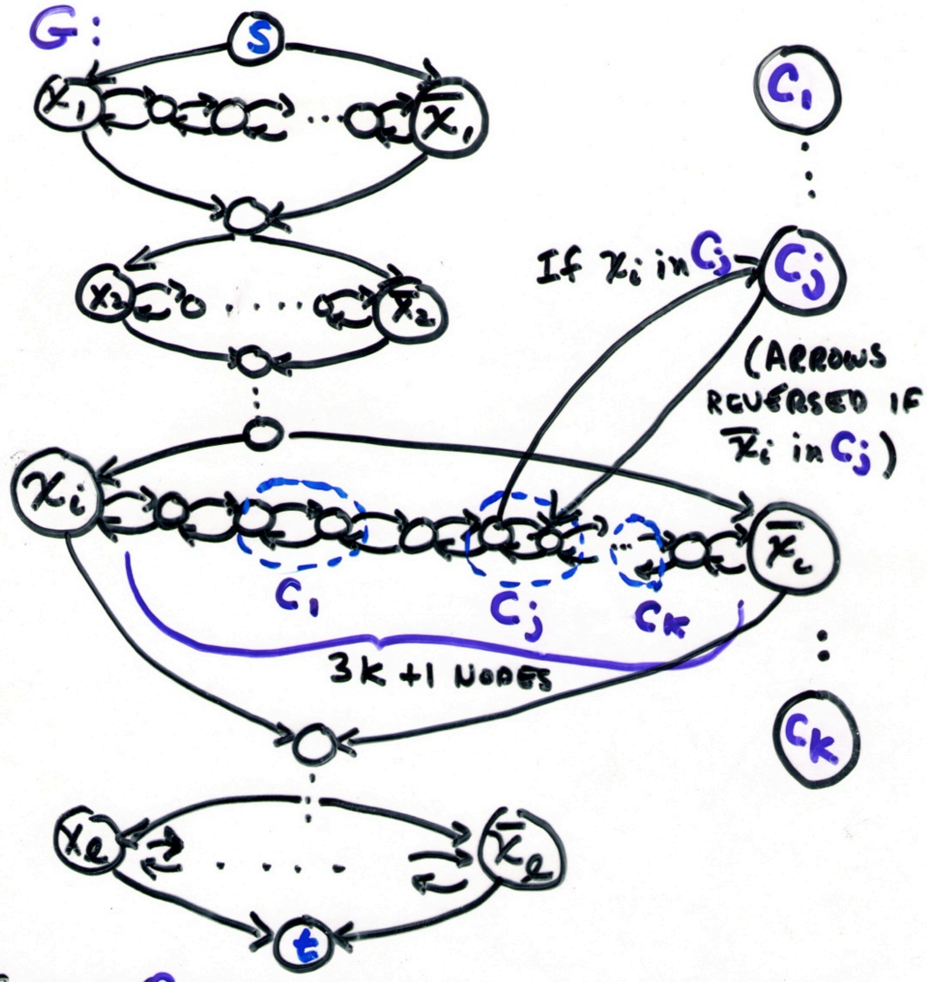


- SUPPOSE ϕ SATISFIABLE WITH SOME TRUTH ASSIGNMENT.
- ZIG-ZAG IF x_i is TRUE(1); ZAG-ZIG if $\bar{x}_i$ is TRUE(1).

• 3SAT $\leq_p$ HAM PATH

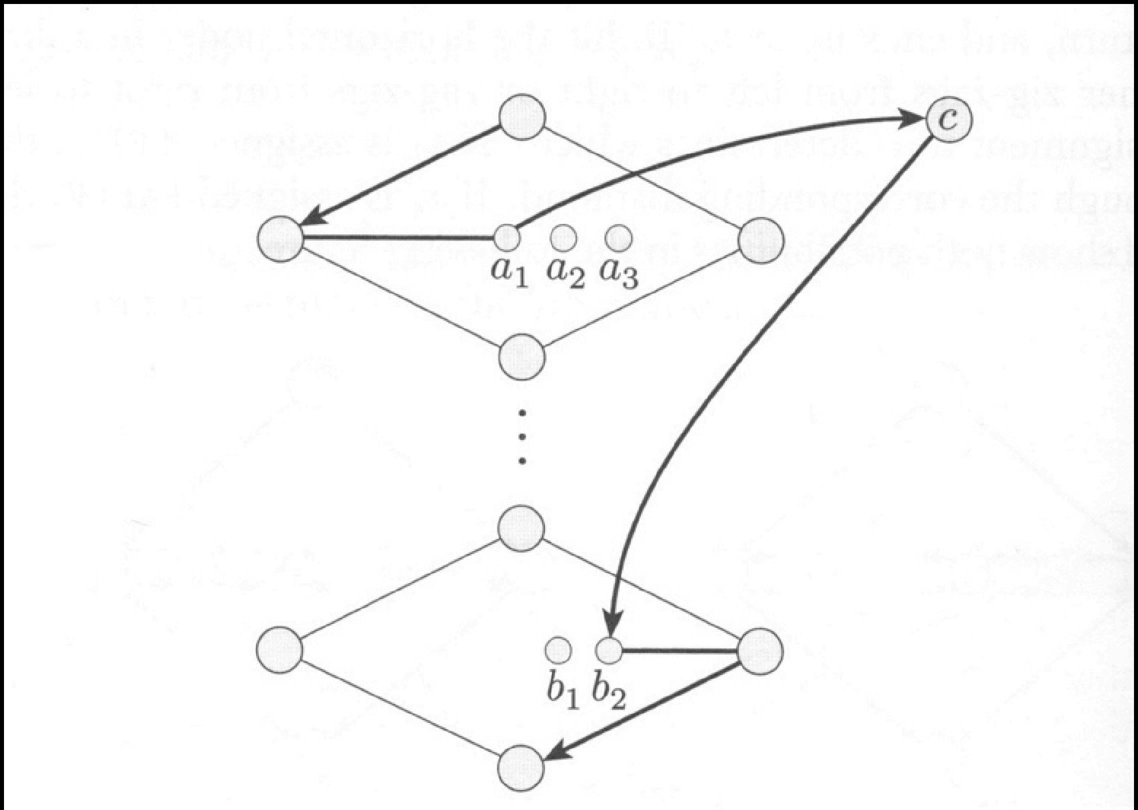
$\varphi = C_1 \wedge C_2 \wedge \cdot \wedge C_j \wedge \cdot\cdot \wedge C_k$ C_j, CLAUSE

$x_1, \ldots, x_\ell$ VARIABLES



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

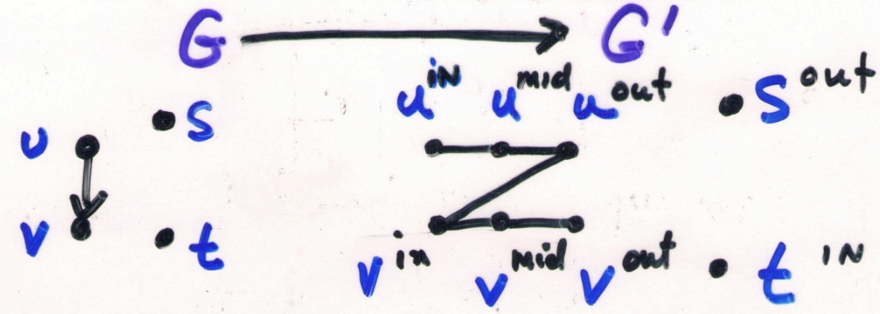
UHAMPATH = { (G,s,t) | G is an undirected graph with a Hamilton path from s to t}

Theorem: UHAMPATH is NP-Complete

(1) UHAMPATH $\in$ NP

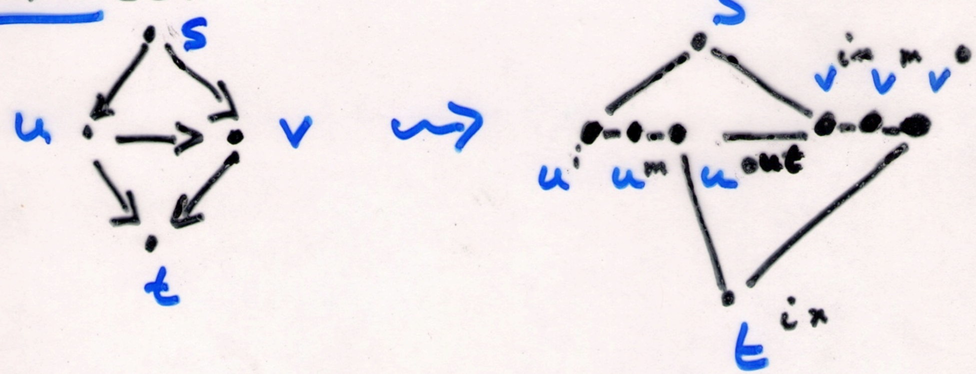
(2) HAMPATH $\leq_P$ UHAMPATH

- **HAMPATH $\leq_p$ UHAMPATH**



Rule: $u \rightarrow v$ 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

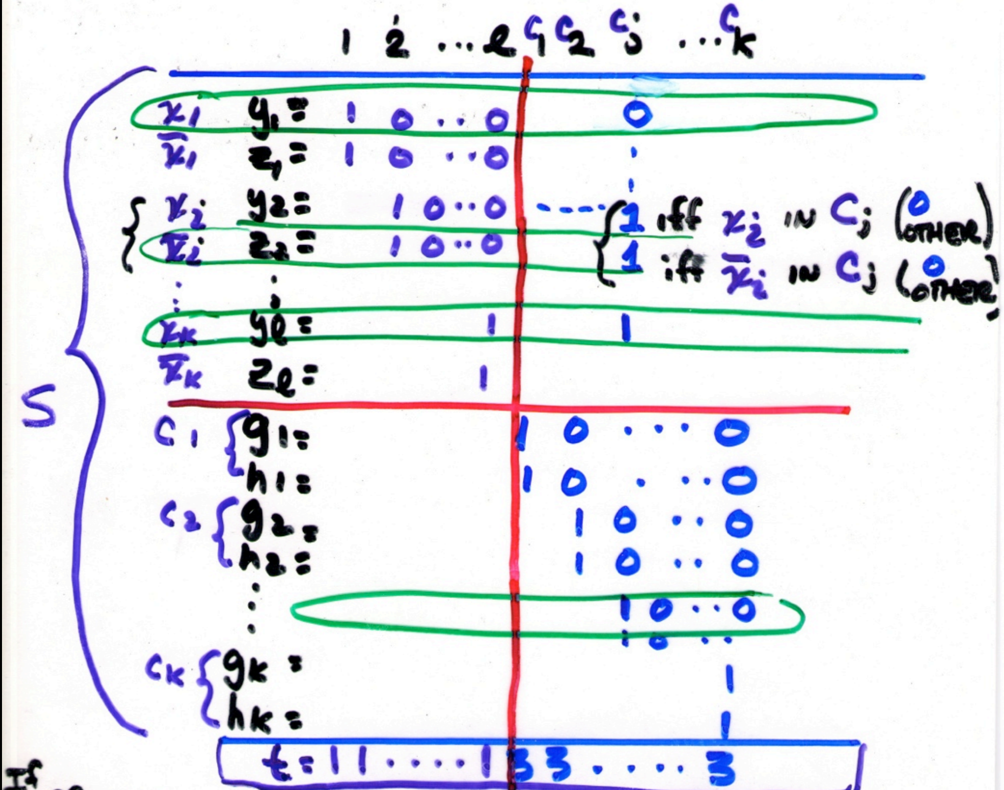
$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$ C_j, CLAUSE

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

$\downarrow$

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

$t = \underbrace{11 \cdots 1}_{\ell}\underbrace{33 \cdots 3}_{k}$



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