CMSC 28100

Introduction to Complexity Theory

Spring 2025
Instructor: William Hoza

k-CNF formulas

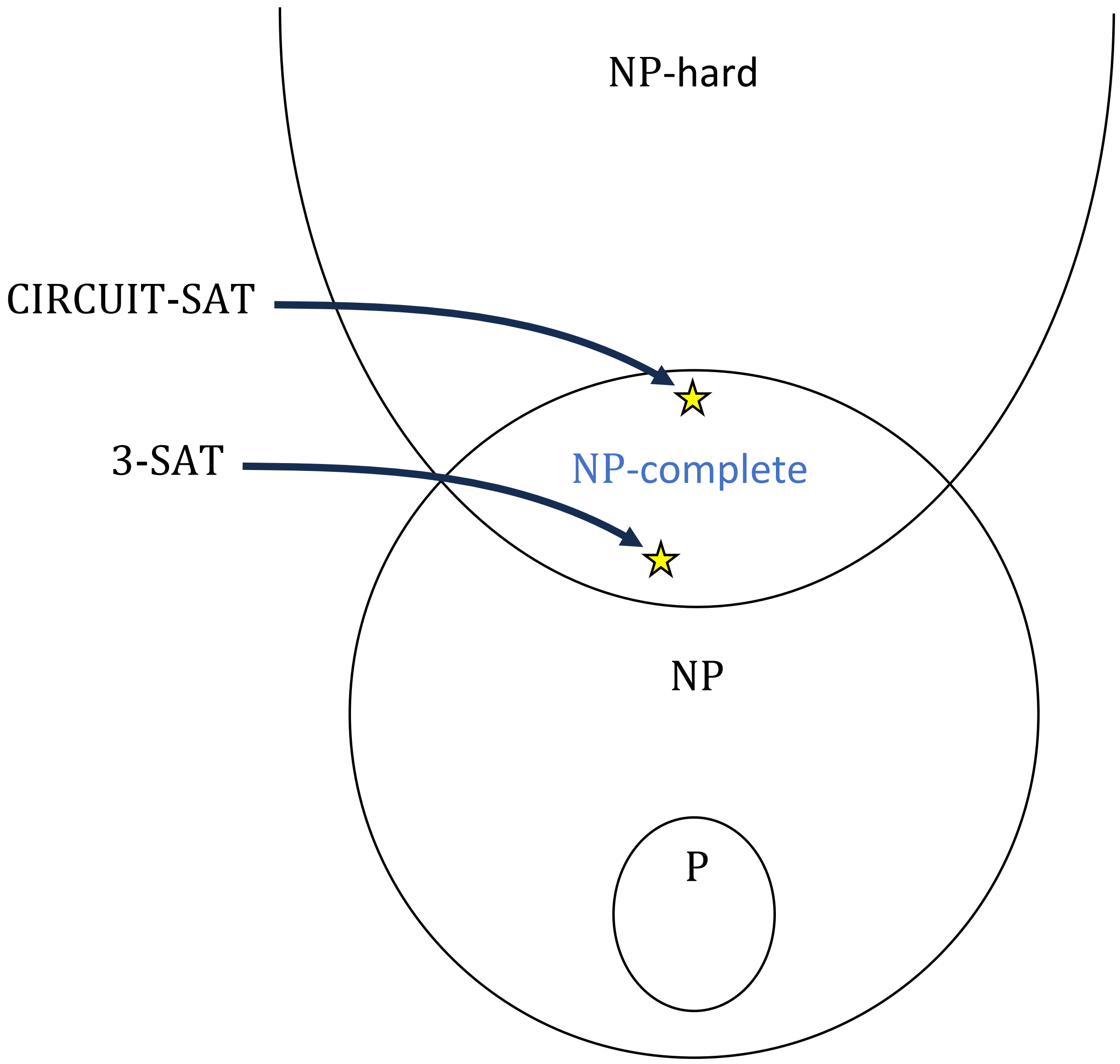
- Recall: A CNF formula is an "AND of ORs of literals"
- **Definition:** A k-CNF formula is a CNF formula in which every clause has at most k literals
- Example of a 3-CNF formula with two clauses:

$$\phi = (x_1 \vee \bar{x}_2 \vee \bar{x}_6) \wedge (x_5 \vee x_1 \vee x_2)$$

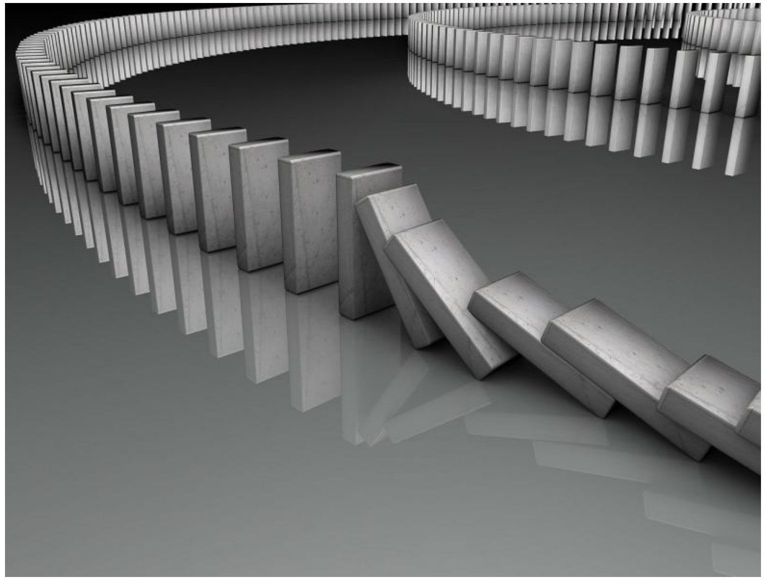
The Cook-Levin Theorem

- Define $k\text{-SAT} = \{\langle \phi \rangle : \phi \text{ is a satisfiable } k\text{-CNF formula}\}$

The Cook-Levin Theorem: 3-SAT is NP-complete


NP-hard
CIRCUIT-SAT
3-SAT
NP-complete
NP
P

Chaining reductions together



- 3-SAT is the starting point for many NP-hardness proofs
- We are finally ready to prove that CLIQUE is NP-complete

CLIQUE is NP-complete

- Recall $\mathrm{CLIQUE} = \{\langle G, k\rangle : G \text{ contains a } k\text{-clique}\}$

Theorem: CLIQUE is NP-complete

- **Proof:** We showed $\mathrm{CLIQUE} \in \mathrm{NP}$ in a previous class
- To prove that CLIQUE is NP-hard, we will do a reduction from 3-SAT

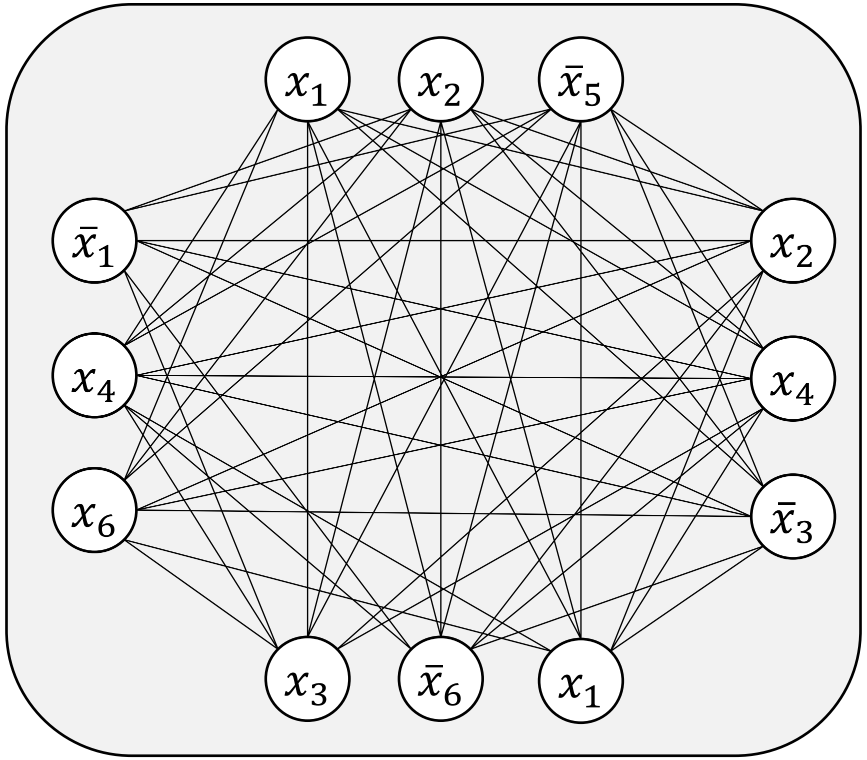
Proof that $3\text{-SAT} \leq_{\mathrm{P}} \mathrm{CLIQUE}$

- Let ϕ be a 3-CNF formula (an instance of 3-SAT)
- Reduction: Given $\langle \phi \rangle$, produce $\langle G, k \rangle$
  - k is the number of clauses in ϕ
  - G is a graph on $\leq 3k$ vertices defined as follows

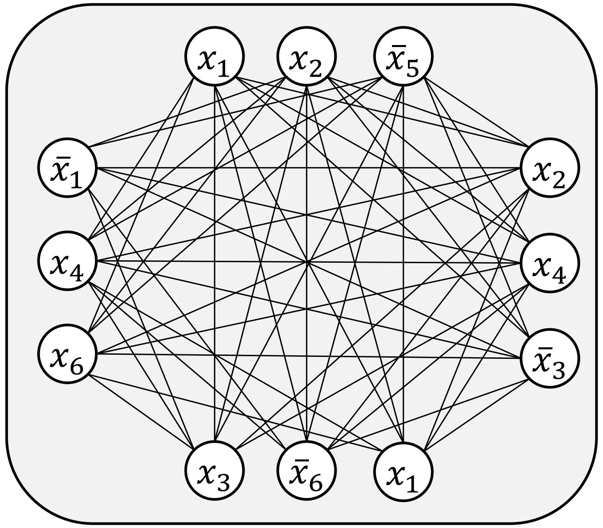
Reduction from **3-SAT** to **CLIQUE**

- For each clause $(\ell_1 \vee \ell_2 \vee \ell_3)$, create a "group" of three vertices labeled ℓ_1, ℓ_2, ℓ_3
  - (If the clause only has one or two literals, then only use one or two vertices)
- Put an edge $\{u, v\}$ if u and v are in different groups and u and v do not have contradictory labels (x_i and $\bar{x}_i$)

- E.g., $\phi = (x_1 \vee x_2 \vee \bar{x}_5) \wedge (\bar{x}_1 \vee x_4 \vee x_6) \wedge (x_2 \vee x_4 \vee \bar{x}_3) \wedge (x_3 \vee \bar{x}_6 \vee x_1)$

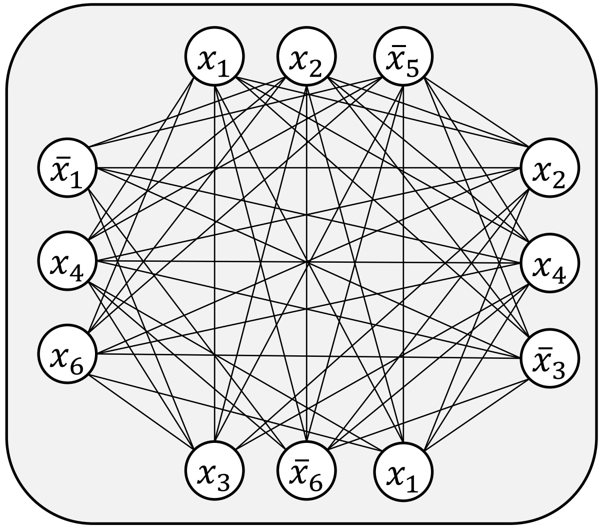


YES maps to YES



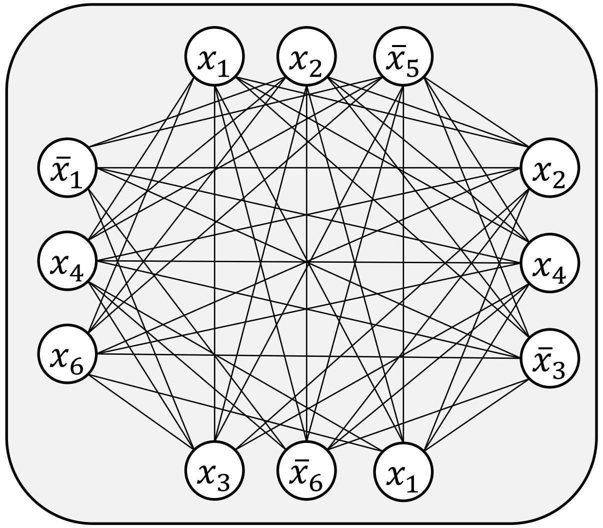
- Suppose there exists x such that $\phi(x) = 1$
- In each clause, at least one literal is satisfied by x
- Therefore, in each group, at least one vertex is "satisfied by x," i.e., it is labeled by a literal that is satisfied by x
- Let S be a set consisting of one satisfied vertex from each group
- Then S is a k-clique (vertices in S cannot have contradictory labels)

NO maps to NO

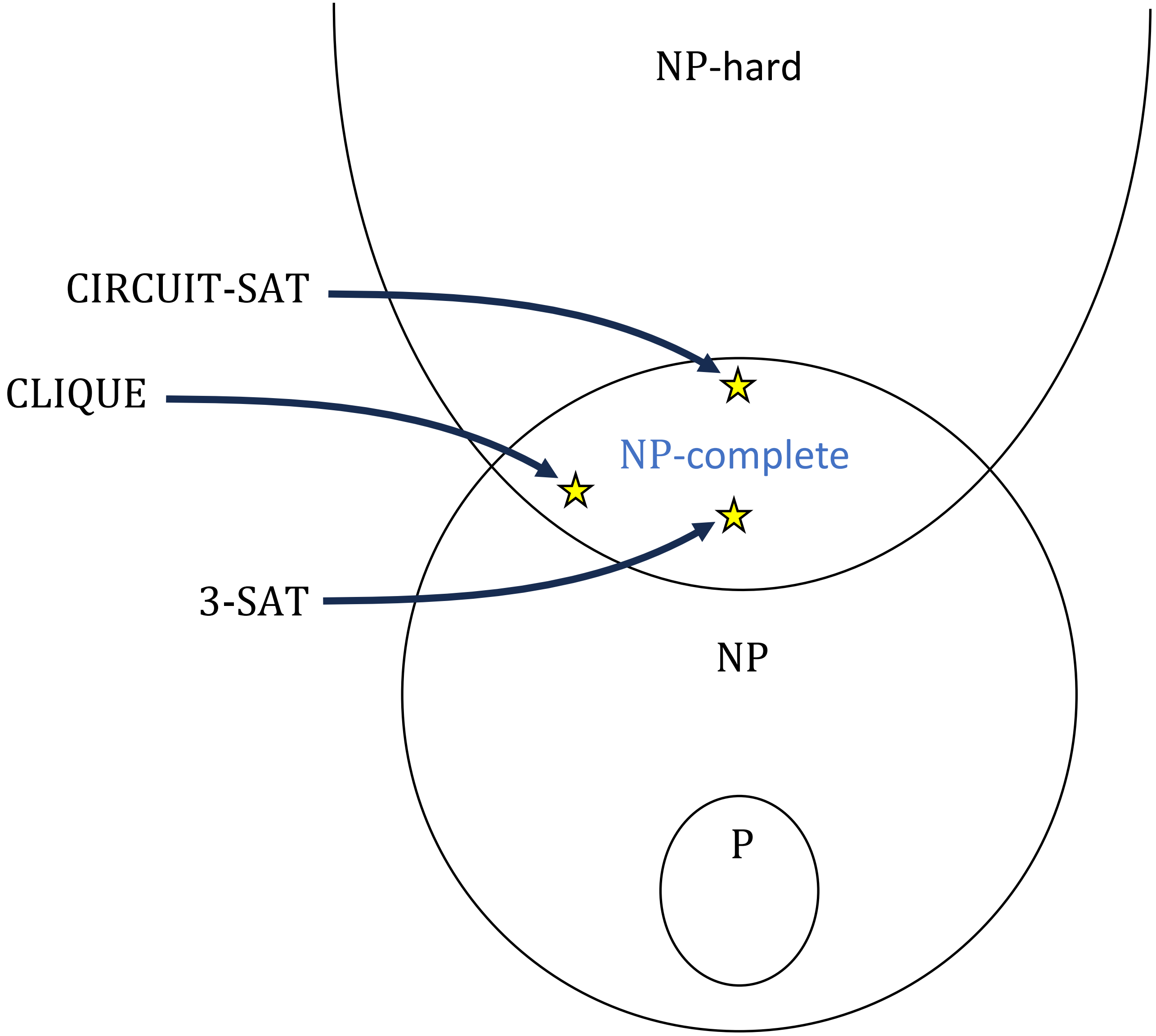


- Suppose G has a k-clique S
- Let x be an assignment that satisfies each vertex in S
  - This exists because no two vertices in S have contradictory labels
- S cannot contain two vertices from a single group, and $|S| = k$, so S must contain one vertex from each group
- Therefore, x satisfies at least one literal in each clause, so $\phi(x) = 1$

Poly-time computable



- Hopefully it is clear that given $\langle \phi \rangle$, one can construct $\langle G, k \rangle$ in polynomial time


NP-hard
CIRCUIT-SAT
CLIQUE
NP-complete
3-SAT
NP
P

The subset sum problem

$$\text{SUBSET-SUM} = \left\{ \langle a_1, \ldots, a_k, T \rangle : \begin{array}{l} a_1, \ldots, a_k, T \in \mathbb{N} \text{ and there exists} \\ I \subseteq \{1, \ldots, k\} \text{ such that } \sum_{i \in I} a_i = T \end{array} \right\}$$

Theorem: SUBSET-SUM is NP-complete

- **Proof:** SUBSET-SUM $\in$ NP. (Why?)
- We will prove it is NP-hard by reduction from 3-SAT

Proof that 3-SAT $\leq_P$ SUBSET-SUM

Given $\langle\phi\rangle$ with variables $x_1, \dots, x_n$ and clauses $c_1, \dots, c_m$, the reduction produces:

Integers represented in base 8

	x_1	x_2	$\cdots$	x_n	c_1	c_2	$\cdots$	c_m
$a_{x_1} =$	1	0	$\cdots$	0	1	0	$\cdots$	0
$a_{\bar{x}_1} =$	1	0	$\cdots$	0	0	0	$\cdots$	0
$a_{x_2} =$		1	$\cdots$	0	0	1	$\cdots$	0
$a_{\bar{x}_2} =$		1	$\cdots$	0	1	0	$\cdots$	0
$\vdots$			$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$a_{x_n} =$				1	1	0	$\cdots$	1
$a_{\bar{x}_n} =$				1	0	1	$\cdots$	1
$a_{c_1} =$					1	0	$\cdots$	0
$a'_{c_1} =$					1	0	$\cdots$	0
$a_{c_2} =$						1	$\cdots$	0
$a'_{c_2} =$						1	$\cdots$	0
$\vdots$							$\ddots$	$\vdots$
$a_{c_m} =$								1
$a'_{c_m} =$								1
$T =$	1	1	$\cdots$	1	3	3	3	3

Does x_2 appear in c_2?

Does $\bar{x}_n$ appear in c_m?

- Suppose $\phi(x) = 1$
  - If $x_i = 1$, select a_{x_i}
  - If $x_i = 0$, select $a_{\bar{x}_i}$
  - If only two literals in c_j are satisfied, select a_{c_j}
  - If only one literal in c_j is satisfied, select a_{c_j} and a'_{c_j}
  - Selected numbers sum to T ✔

Proof that 3-SAT $\leq_P$ SUBSET-SUM

Given $\langle \phi \rangle$ with variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$, the reduction produces:

	x_1	x_2	$\cdots$	x_n	c_1	c_2	$\cdots$	c_m
$a_{x_1} =$	1	0	$\cdots$	0	1	0	$\cdots$	0
$a_{\bar{x}_1} =$	1	0	$\cdots$	0	0	0	$\cdots$	0
$a_{x_2} =$		1	$\cdots$	0	0	1	$\cdots$	0
$a_{\bar{x}_2} =$		1	$\cdots$	0	1	0	$\cdots$	0
$\vdots$			$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$a_{x_n} =$				1	1	0	$\cdots$	1
$a_{\bar{x}_n} =$				1	0	1	$\cdots$	1
$a_{c_1} =$					1	0	$\cdots$	0
$a'_{c_1} =$					1	0	$\cdots$	0
$a_{c_2} =$						1	$\cdots$	0
$a'_{c_2} =$						1	$\cdots$	0
$\vdots$							$\ddots$	$\vdots$
$a_{c_m} =$								1
$a'_{c_m} =$								1
$T =$	1	1	$\cdots$	1	3	3	3	3

Integers represented in base 8

Does x_2 appear in c_2?

Does $\bar{x}_n$ appear in c_m?

- Suppose a subset of the numbers sum to T
  - There are no "carries," because each column has at most five ones
  - If a_{x_i} is selected, set $x_i = 1$
  - If $a_{\bar{x}_i}$ is selected, set $x_i = 0$
  - Each clause must have at least one satisfied literal ✓

Proof that 3-SAT $\leq_P$ SUBSET-SUM

Given $\langle \phi \rangle$ with variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$, the reduction produces:

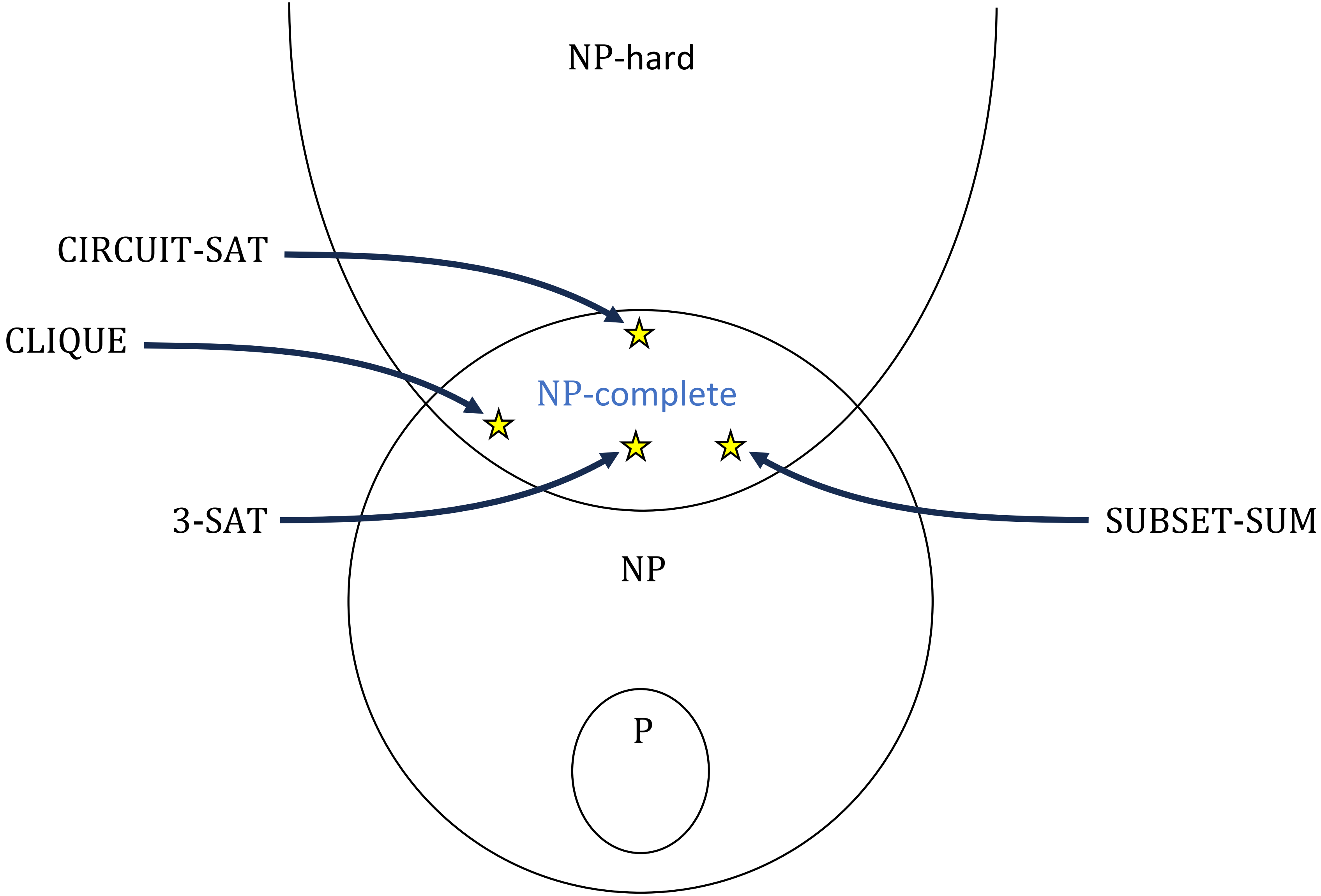
Integers represented in base 8

	x_1	x_2	$\cdots$	x_n	c_1	c_2	$\cdots$	c_m
$a_{x_1} =$	1	0	$\cdots$	0	1	0	$\cdots$	0
$a_{\bar{x}_1} =$	1	0	$\cdots$	0	0	0	$\cdots$	0
$a_{x_2} =$		1	$\cdots$	0	0	1	$\cdots$	0
$a_{\bar{x}_2} =$		1	$\cdots$	0	1	0	$\cdots$	0
$\vdots$			$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$a_{x_n} =$				1	1	0	$\cdots$	1
$a_{\bar{x}_n} =$				1	0	1	$\cdots$	1
$a_{c_1} =$					1	0	$\cdots$	0
$a'_{c_1} =$					1	0	$\cdots$	0
$a_{c_2} =$						1	$\cdots$	0
$a'_{c_2} =$						1	$\cdots$	0
$\vdots$							$\ddots$	$\vdots$
$a_{c_m} =$								1
$a'_{c_m} =$								1
$T =$	1	1	$\cdots$	1	3	3	3	3

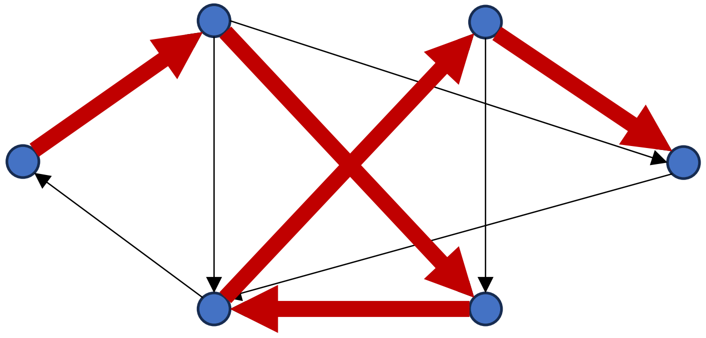
Does x_2 appear in c_2?

Does $\bar{x}_n$ appear in c_m?

- Reduction can be performed in polynomial time ✔


NP-hard
CIRCUIT-SAT
CLIQUE
NP-complete
3-SAT
SUBSET-SUM
NP
P

Hamiltonian paths



- Let G be a directed graph
- **Definition:** A Hamiltonian path is a directed path that visits every vertex exactly once

DIRECTED-HAM-PATH is NP-complete

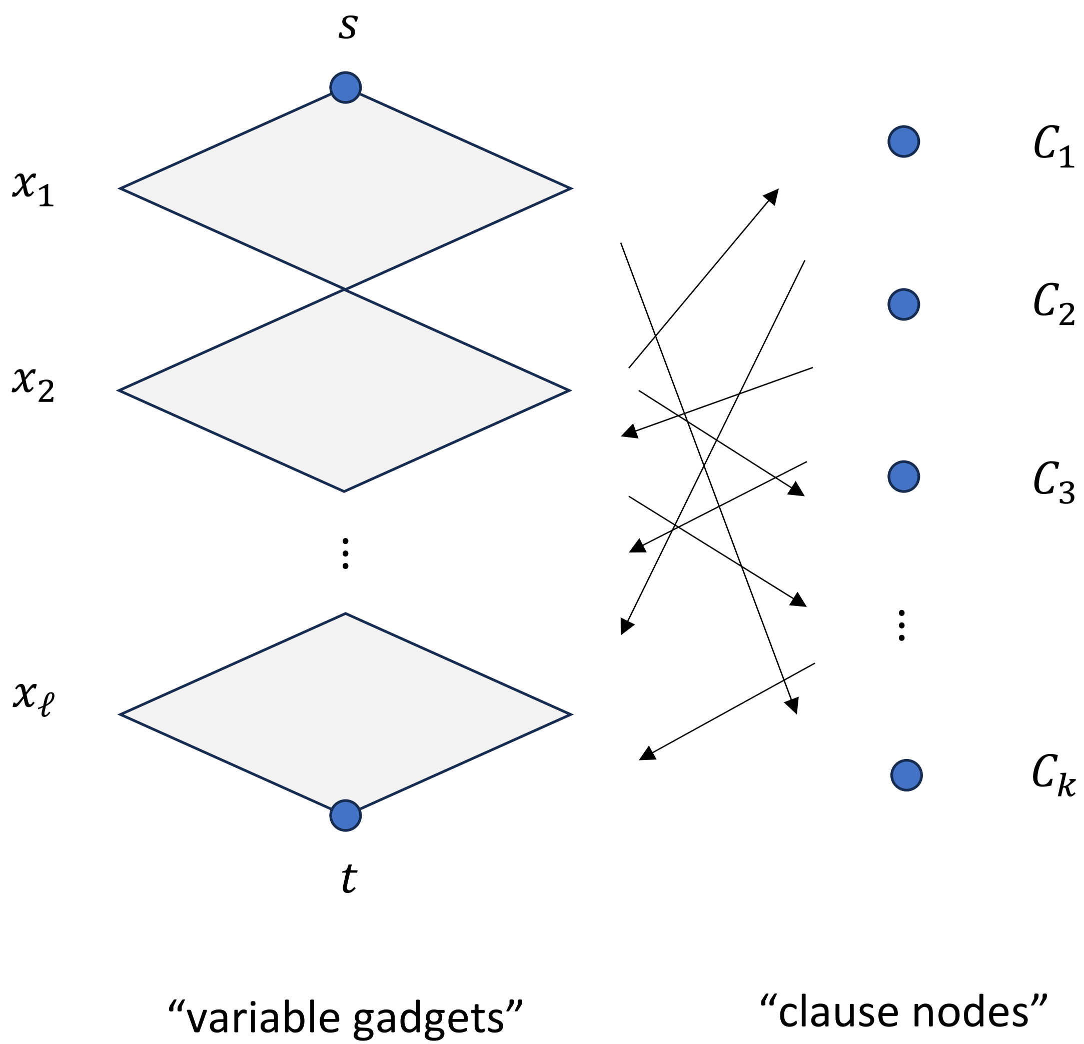
- Let $\text{DIRECTED-HAM-PATH} = \{\langle G, s, t\rangle : G$ is a digraph, s and t are vertices, and there exists a Hamiltonian path from s to $t\}$

Theorem: DIRECTED-HAM-PATH is NP-complete

- **Proof:** First, note DIRECTED-HAM-PATH $\in$ NP. (Why?)
- To show NP-hardness, we will do a reduction from 3-SAT

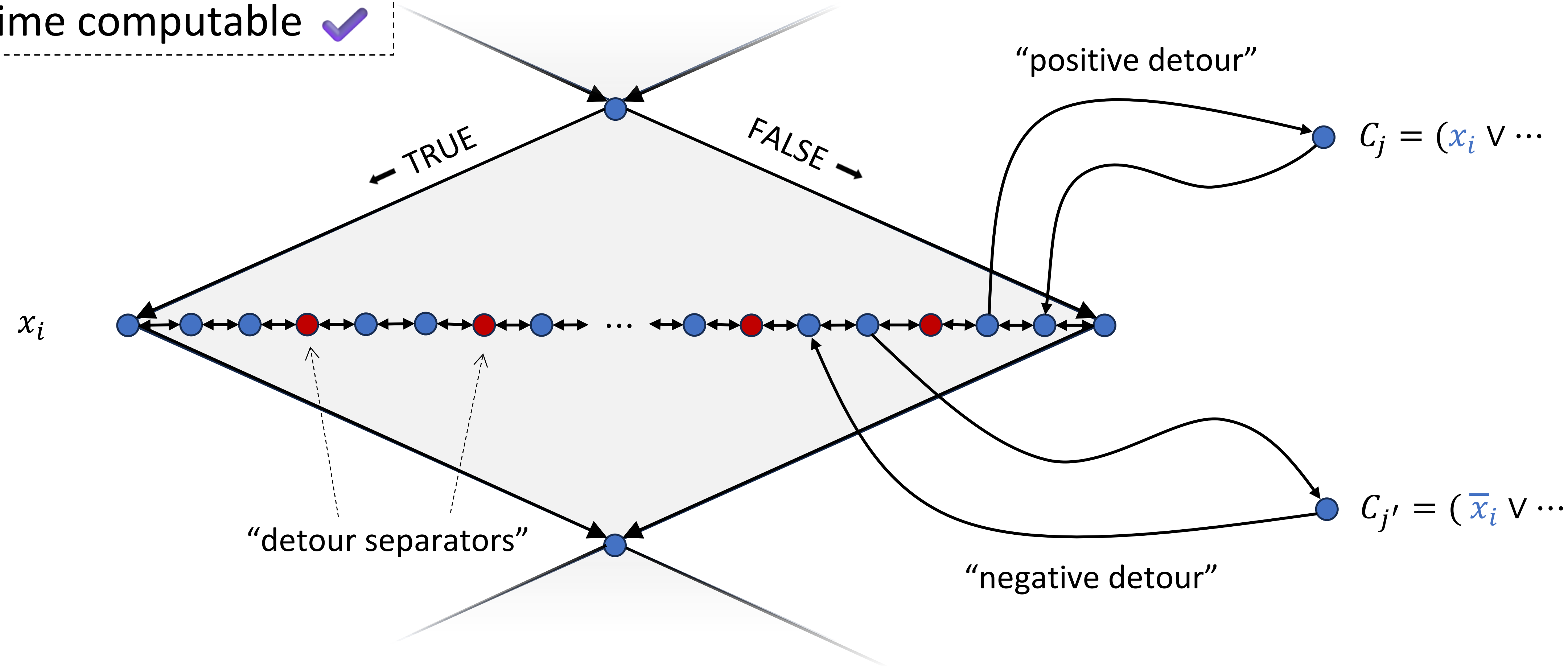
Proof that 3-SAT $\leq_{\mathrm{P}}$ DIRECTED-HAM-PATH

- Let $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$ be a 3-CNF formula on variables $x_1, \ldots, x_\ell$
- Reduction: Given $\langle \phi \rangle$, produce $\langle G, s, t \rangle$ defined on this and upcoming slides



Proof that 3-SAT $\leq_P$ DIRECTED-HAM-PATH

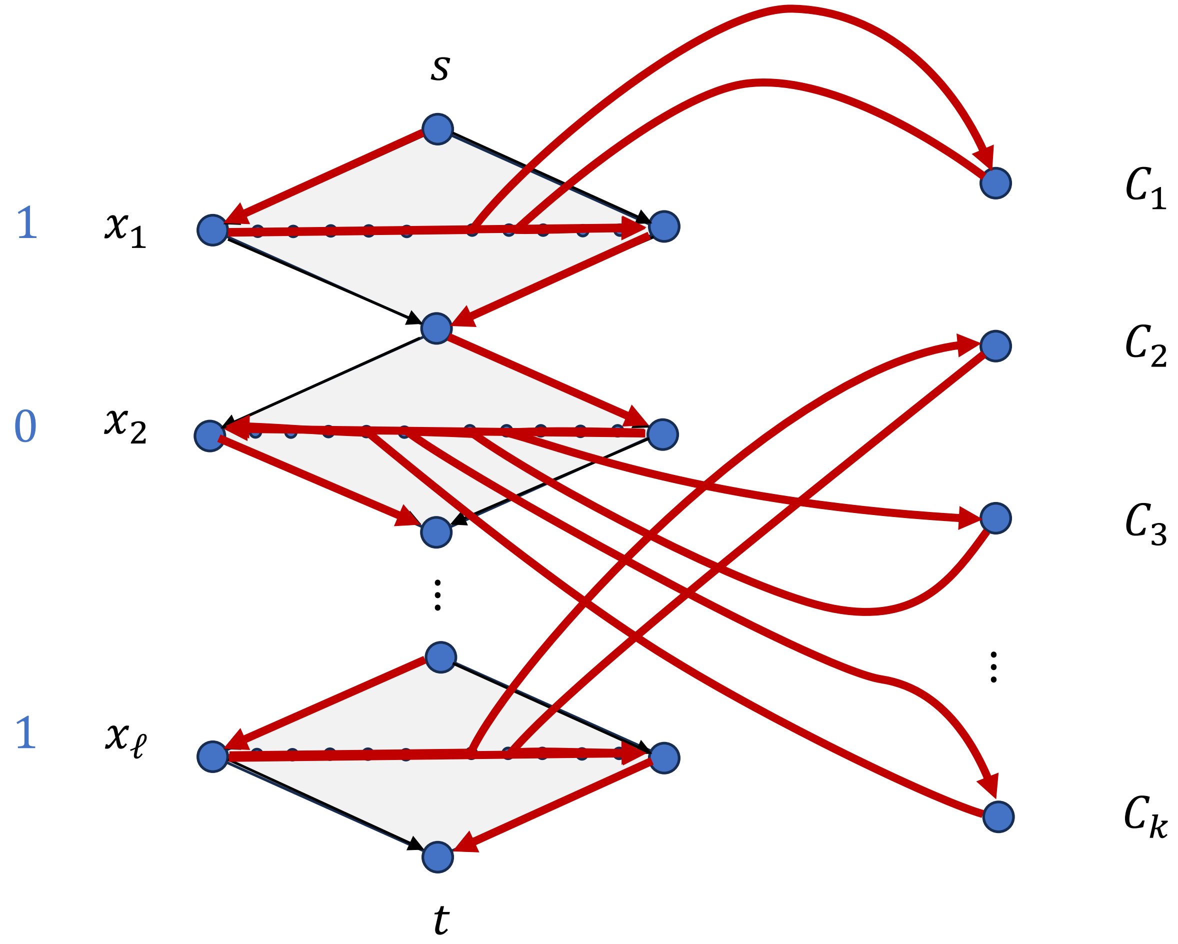
Poly-time computable ✔



- $6k - 1$ nodes inside diamond (enough for all possible detours)

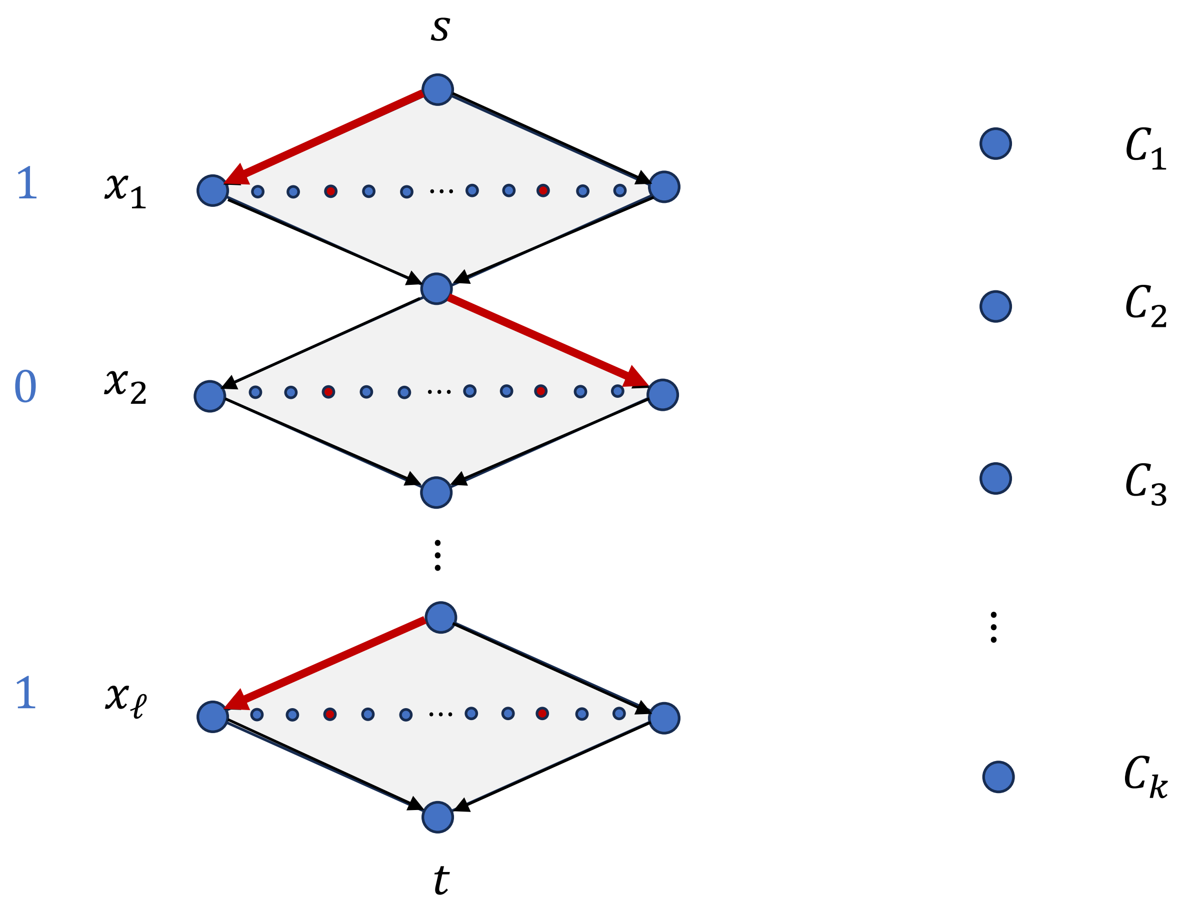
Proof that 3-SAT $\leq_P$ DIRECTED-HAM-PATH

- YES maps to YES: Let x be a satisfying assignment to ϕ
- Depending on assignment to x_i, we "zig-zag" or "zag-zig" through x_i diamond
- Each clause has a satisfied literal; insert the corresponding detour



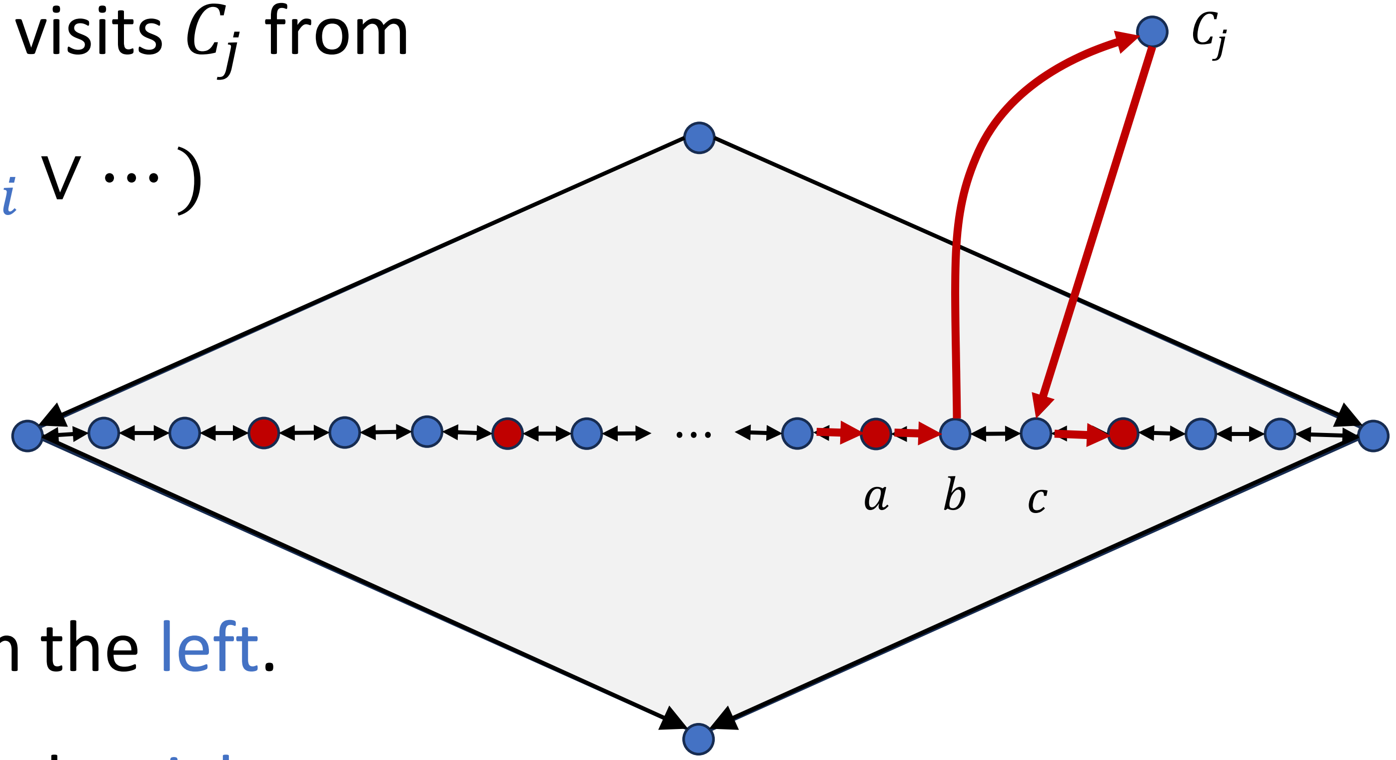
Proof that 3-SAT $\leq_P$ DIRECTED-HAM-PATH

- NO maps to NO: Consider any Hamiltonian path from s to t
- Assign value to x_i based on edge traversed from top of x_i diamond
- We must show that this assignment satisfies every clause



Proof that 3-SAT $\leq_P$ DIRECTED-HAM-PATH

- Consider any clause node C_j. Path visits C_j from some x_i diamond. WLOG, $C_j = (x_i \vee \cdots)$
- **Claim:** The path goes from C_j back to that same diamond (x_i)
- **Proof:** The path must enter a from the left. Therefore, the path must exit a to the right. Therefore, the path must exit c to the right, so the path must enter c from C_j



Proof that 3-SAT $\leq_P$ DIRECTED-HAM-PATH

- Consequence: If we traverse a "TRUE" edge, then we can only take "positive detours" in that diamond
- If we traverse a "FALSE" edge, then we can only take "negative detours" in that diamond
- Therefore, every clause is satisfied