Reduction problem

Week 14 discussion

Yintao, Xu | Fall 2019 | CS101
Reference: Rui Fan's CS140

Reduction

- **What does it mean to say problem B is harder than problem A?**
- It means if you can solve B, you can also solve A.
  - Algebra is harder than arithmetic, because if you can do algebra, you can also do arithmetic.
- (definition) So if I have an algorithm for solving B, I can use it to solve A.
  - **We say A reduces to B.**
  - **Write $A \leq_R B$.**
  - Read this as "A is equally or less difficult than B"
  - Note the direction of the inequality.

Reduction

If the mapping function from A to B runs in polynomial time, then it is a **polynomial time reduction**, and we write $A \leq_P B$

Reduction

- **(definition)** An **instance** of a problem consists of an input for the problem.
  - An instance of the sorting problem is a set {3,1,2,4} that we want to sort
- **(formal definition)** Problem X **polynomial-time (Cook) reduces** to problem Y if arbitrary instances of problem X can be solved using:
  - Polynomial number of standard computational steps, plus
  - Polynomial number of calls to oracle that solves problem Y.
- We write $\boldsymbol{A \leq_R B}$

Reduction

- To show $\boldsymbol{A} \leq_{\boldsymbol{R}} \boldsymbol{B}$, just give the mapping f:
- If $\boldsymbol{A} \leq_{\boldsymbol{R}} \boldsymbol{B}$, then we can use an algorithm for B to solve A:
  - To solve an instance of A, first map it to an instance of B using f.
  - Then run the B algorithm.
  - Return the same answer for A as the B algorithm gives.
  - By definition, A is true equals to f(A) is true
  - Similar to problems other than decision problems.

Decision, search and optimal

Motivation: reduction is formally defined on decision problem, so how we generalize to other types of problem?

(definition) **A decision problem** asks us to check if something is true.

(definition) **A search problem** asks us to find a solution with certain properties if such a solution exists.

(definition) **A optimization problem** asks us to find among all solutions the one with the best performance in some metric.

Refer to MIT Fall 09 lecture notes

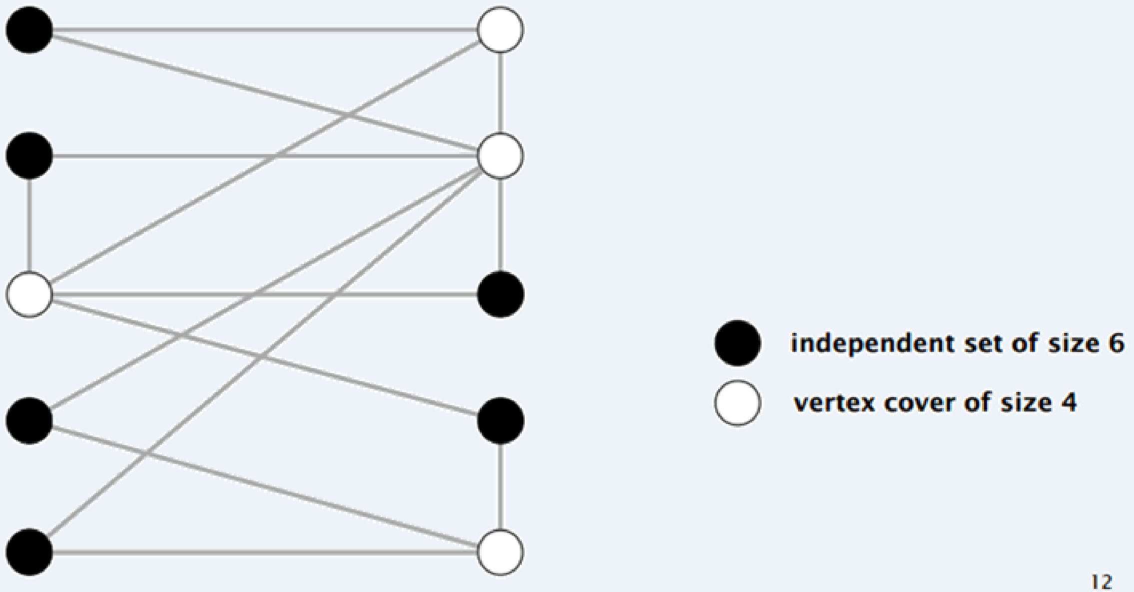
Decision, search and optimal

Review topics on lecture

VERTEX-COVER. Given a graph $G = (V, E)$ and an integer k, is there a subset of k (or fewer) vertices such that each edge is incident to at least one vertex in the subset?

Ex. Is there a vertex cover of size ≤ 4 ?

Ex. Is there a vertex cover of size ≤ 3 ?



Decision, search and optimal

VERTEX-COVER. Does there exist a vertex cover of size $\leq k$?

FIND-VERTEX-COVER. Find a vertex cover of size $\leq k$.

Theorem. VERTEX-COVER $\equiv_P$ FIND-VERTEX-COVER.

Pf. $\leq_P$ Decision problem is a special case of search problem. ▪

Pf. $\geq_P$

To find a vertex cover of size $\leq k$:

- Determine if there exists a vertex cover of size $\leq k$.
- Find a vertex v such that $G - \{v\}$ has a vertex cover of size $\leq k - 1$. (any vertex in any vertex cover of size $\leq k$ will have this property)
- Include v in the vertex cover.
- Recursively find a vertex cover of size $\leq k - 1$ in $G - \{v\}$. ▪

delete v and all incident edges

Decision, search and optimal

FIND-VERTEX-COVER. Find a vertex cover of size $\leq k$.

FIND-MIN-VERTEX-COVER. Find a vertex cover of minimum size.

Theorem. FIND-VERTEX-COVER $\equiv_P$ FIND-MIN-VERTEX-COVER.

Pf. $\leq_P$ Search problem is a special case of optimization problem. ▪

Pf. $\geq_P$ To find vertex cover of minimum size:

- Binary search (or linear search) for size k^* of min vertex cover.
- Solve search problem for given k^*. ▪

Reduction of search

See Figure 2 for how to use functions f and g to produce an algorithm for problem A given an algorithm for problem B.

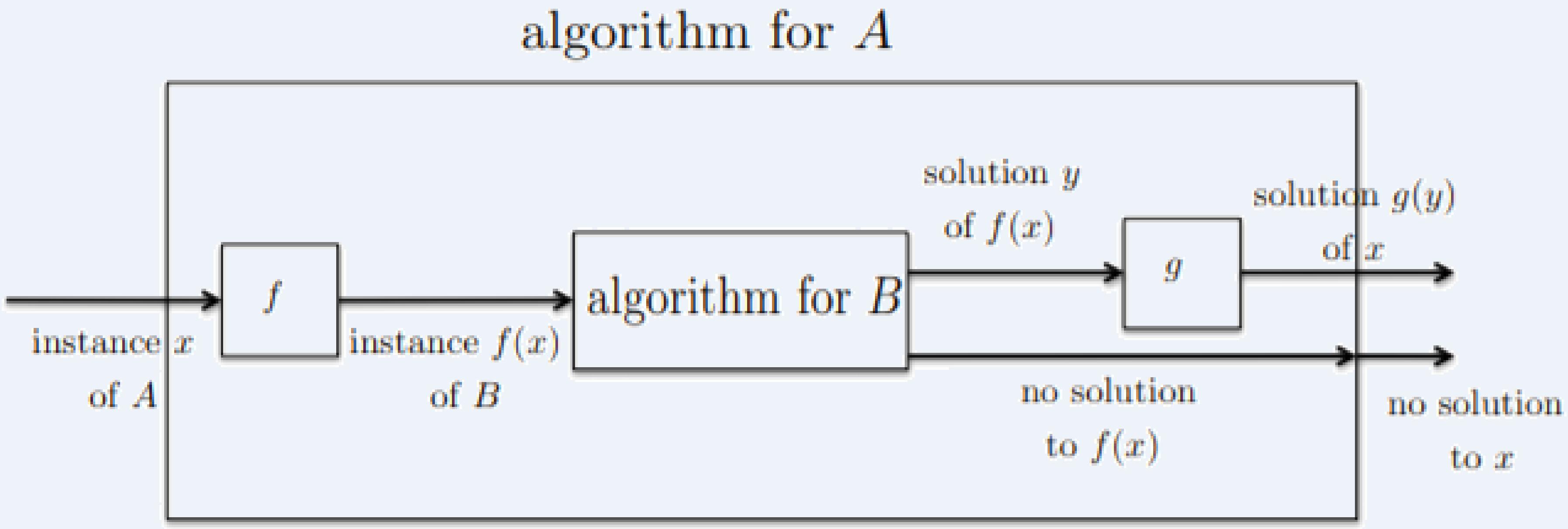


Figure 2: Reduction *from* search problem A *to* search problem B

Refer to MIT Fall 09 lecture notes

Case study 1

3-SAT $\leq_P$ Hamiltonian Cycle

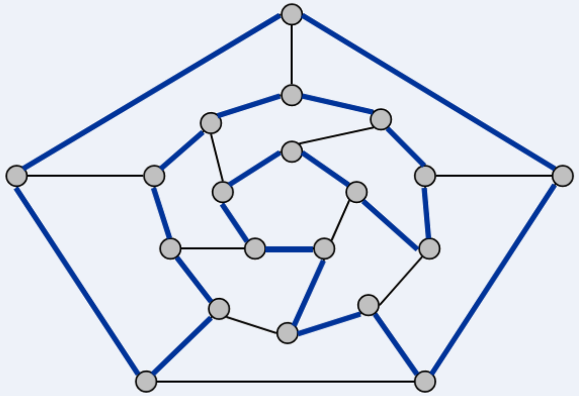
Refer to Rui, Fan's lecture

3-SAT $\leq_P$ Hamiltonian Cycle

- **(definition)** A **decision problem** is a problem with a yes / no answer.
- **(definition)** Given a decision problem, the set of **yes (resp. no) instances** are the instances of the problem for which the answer is yes (resp. no).
  - *11 is a yes instance to the prime problem, 10 is a no instance.*

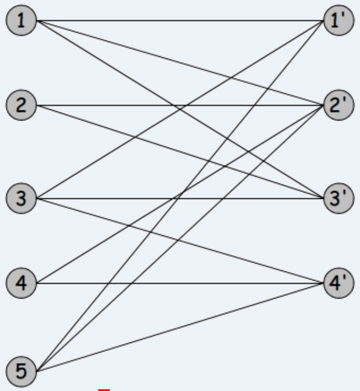
3-SAT $\leq_P$ Hamiltonian Cycle

(definition) **Hamilton-Cycle** given an undirected graph $G = (V, E)$, does there exist a simple cycle Γ that contains every node in V.



Yes instance

vertices and faces of a dodecahedron



No instance

bipartite graph with odd number of nodes

3-SAT $\leq_P$ Hamiltonian Cycle

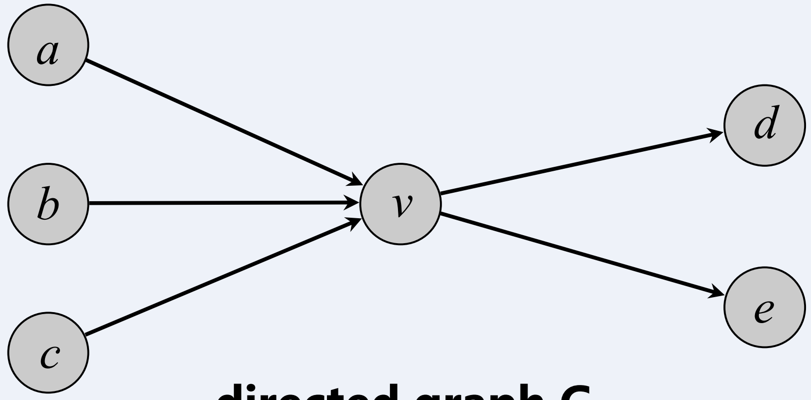
(definition) **3-SAT:** Given a set of clauses C1, . . . , Ck , each of length 3, over variables X = {x1, . . . , xn} is there a satisfying assignment?

$$\Phi = \left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(\overline{x_1} \vee x_2 \vee x_4\right)$$

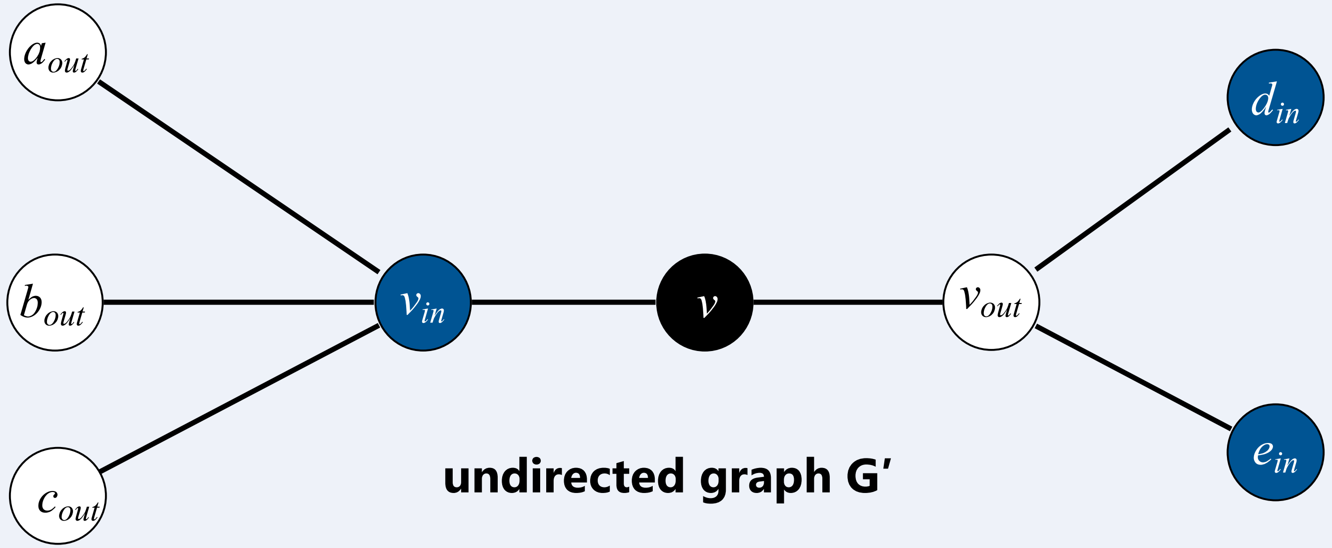
yes instance: x_1 = true, x_2 = true, x_3 = false, x_4 = false

3-SAT $\leq_P$ Hamiltonian Cycle

- **(definition)** **Directed-Hamilton-Cycle** Given a directed graph $G = (V, E)$, does there exist a directed cycle Γ that visits every node exactly once?
- **Theorem** Directed-Hamilton-Cycle $\leq_P$ Hamilton-Cycle
- Pf. Given a directed graph $G = (V, E)$, construct a graph G' with $3n$ nodes.



directed graph G



undirected graph G′

3-SAT $\leq_P$ Hamiltonian Cycle

- **Lemma** G has a directed Hamilton cycle iff G' has a Hamilton cycle.

Pf. $\Rightarrow$ (completeness)

- Suppose G has a directed Hamilton cycle Γ.
- Then G' has an undirected Hamilton cycle (same order). ▪

3-SAT $\leq_P$ Hamiltonian Cycle

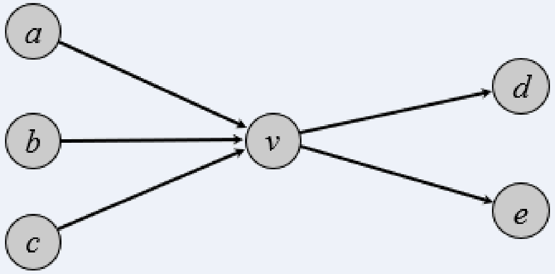
- **Lemma** G has a directed Hamilton cycle iff G' has a Hamilton cycle.

  Pf. $\Leftarrow$ (soundness)

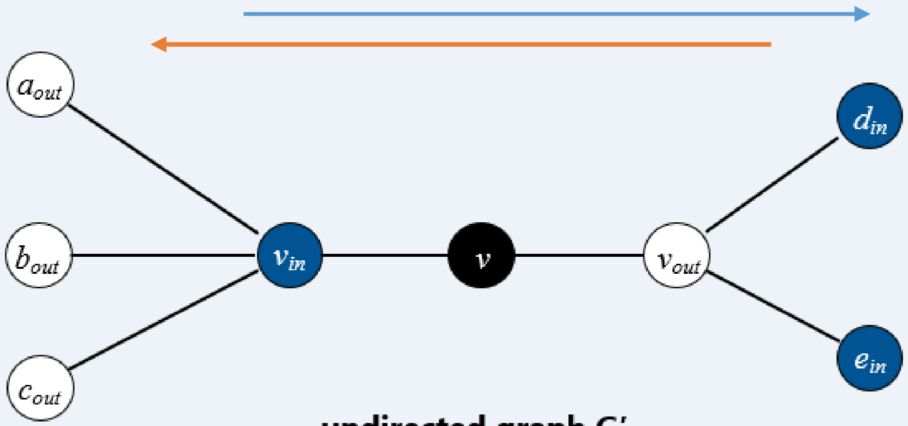
  - Suppose G' has an undirected Hamilton cycle Γ'.
  - Γ' must visit nodes in G' using one of following two orders:

    $\ldots, black, white, blue, black, white, blue, black, white, blue, \ldots$

    $\ldots, black, blue, white, black, blue, white, black, blue, white, \ldots$
  - Black nodes in Γ' comprise either a directed Hamilton cycle Γ in G, or reverse of one. ▪



directed graph G



undirected graph G′

3-SAT $\leq_P$ Hamiltonian Cycle

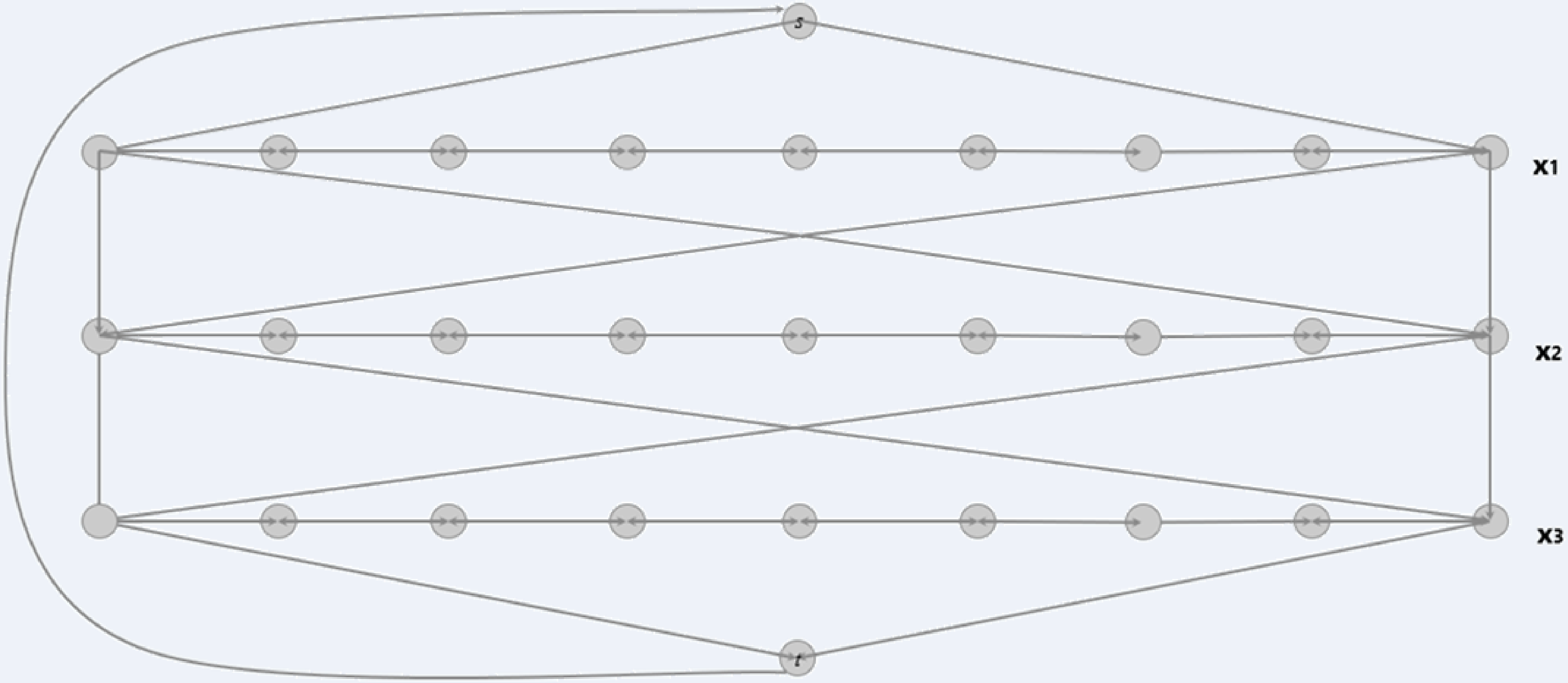
- **Lemma** 3-Sat $\leq_{\mathrm{P}}$ Directed-Hamilton-Cycle.

Pf Given an instance Φ of 3-Sat, we construct an instance G of Directed-Hamilton-Cycle that has a Hamilton cycle iff Φ is satisfiable.

Construction overview Let n denote the number of variables in Φ. We will construct a graph G that has 2^n Hamilton cycles, with each cycle corresponding to one of the 2^n possible truth assignments.

3-SAT $\leq_P$ Hamiltonian Cycle

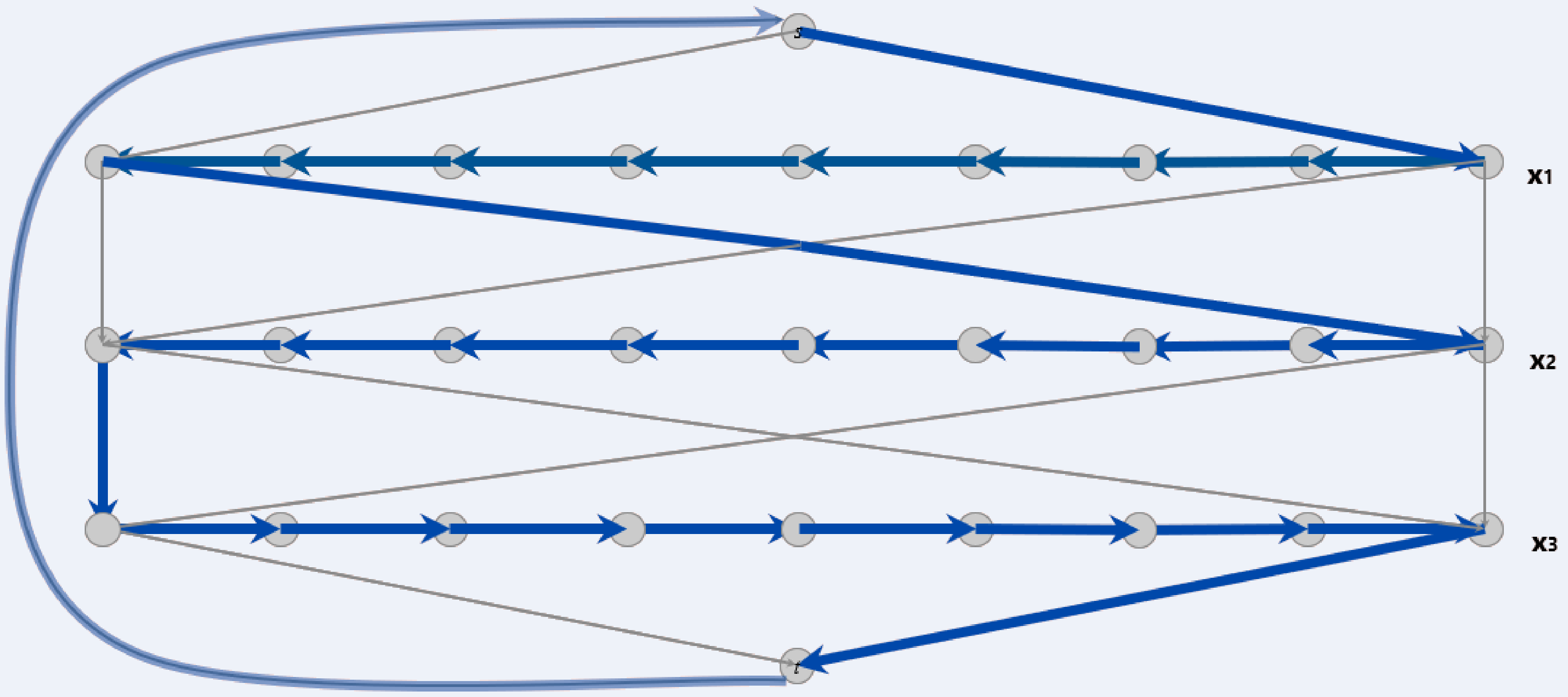
- **Construction** Given 3-Sat instance Φ with n variables x_i and k clauses.
  - Construct G to have 2^n Hamilton cycles.
  - Intuition: traverse path i from left to right $\Leftrightarrow$ set variable $x_i = true$.



3-SAT $\leq_P$ Hamiltonian Cycle

- **Which is truth assignment corresponding to Hamilton cycle below?**

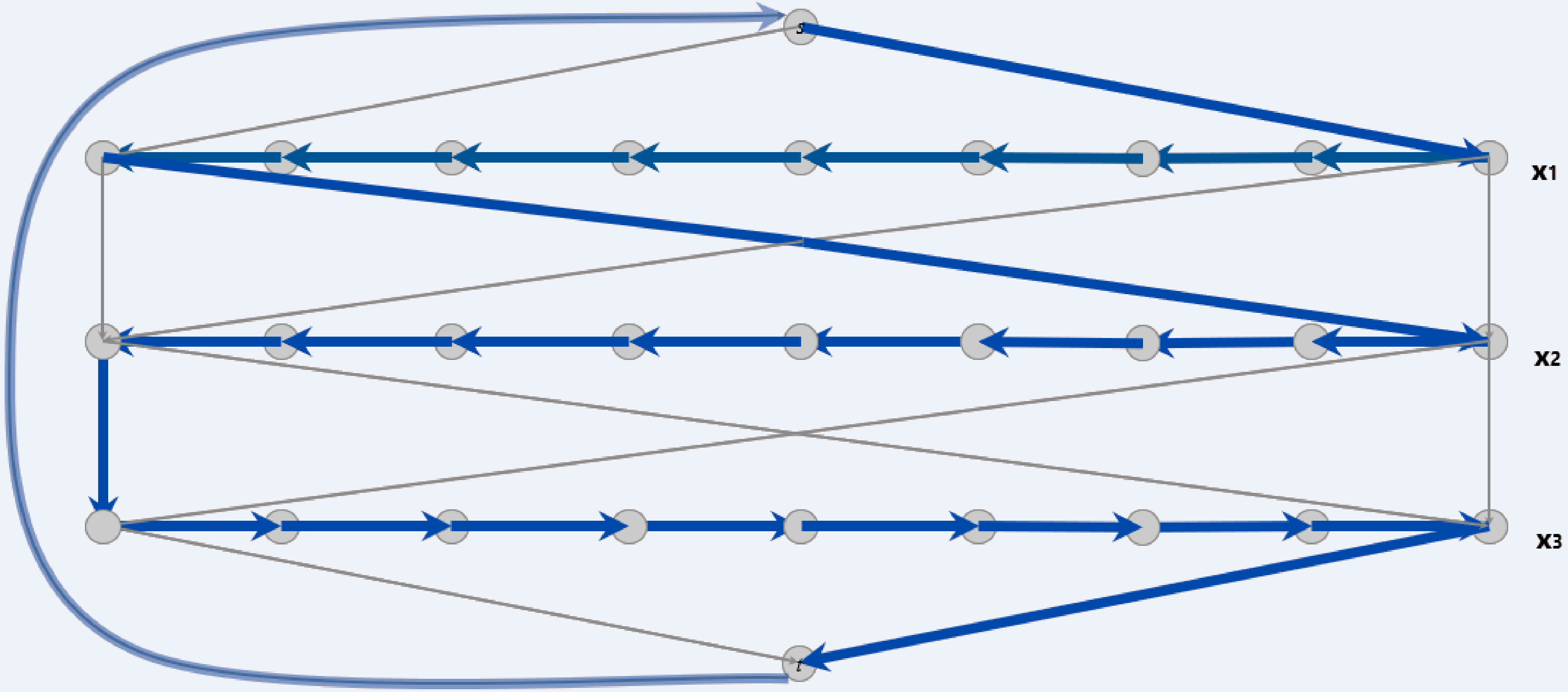
$x_1 = ?, x_2 = ?, x_3 = ?$ traverse path i from left to right $\Leftrightarrow$ set variable $x_i = true$



3-SAT $\leq_P$ Hamiltonian Cycle

- **Which is truth assignment corresponding to Hamilton cycle below?**

$x_1 = F, x_2 = T, x_3 = T$ traverse path i from left to right $\Leftrightarrow$ set variable $x_i = true$



3-SAT $\leq_P$ Hamiltonian Cycle

- **Recall:**

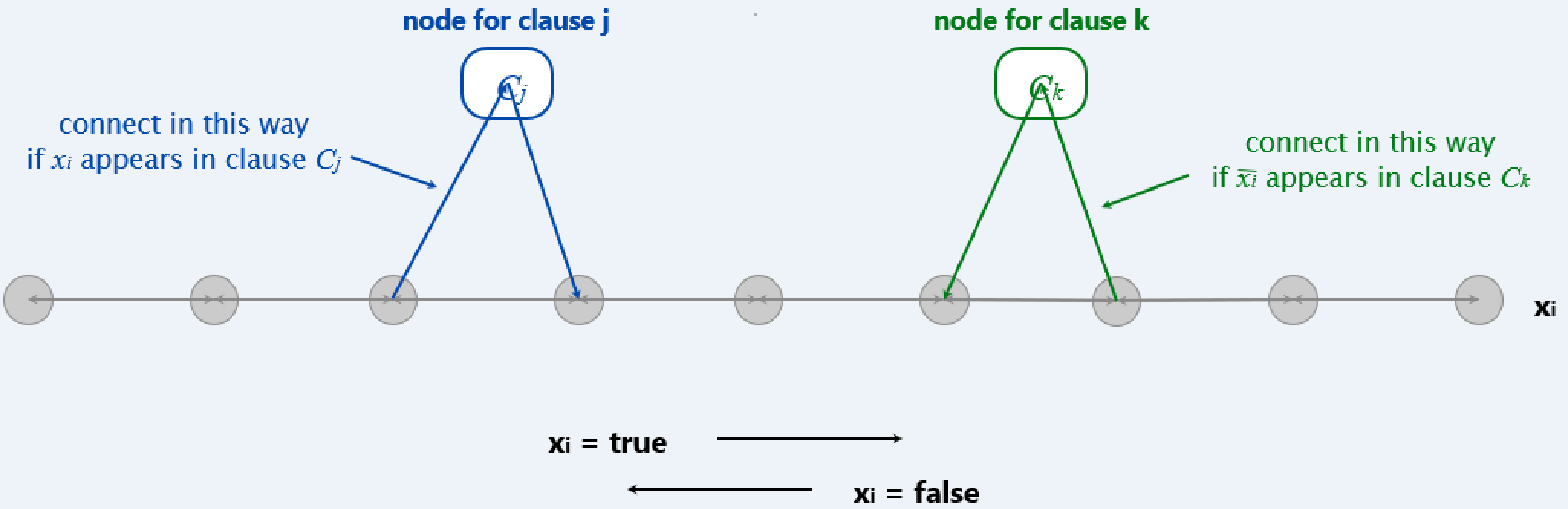
- **Literal** A Boolean variable or its negation. x_i or $\overline{x_i}$
- **Clause** A disjunction of literals. $C_j = x_1 \vee \overline{x_2} \vee x_3$
- **Conjunctive normal form (CNF)** A propositional formula Φ that is a conjunction of clauses. $\Phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$

3-SAT $\leq_P$ Hamiltonian Cycle

Construction Given 3-Sat instance Φ with n variables x_i and k clauses.

- For each clause: add a node and 2 edges per literal.

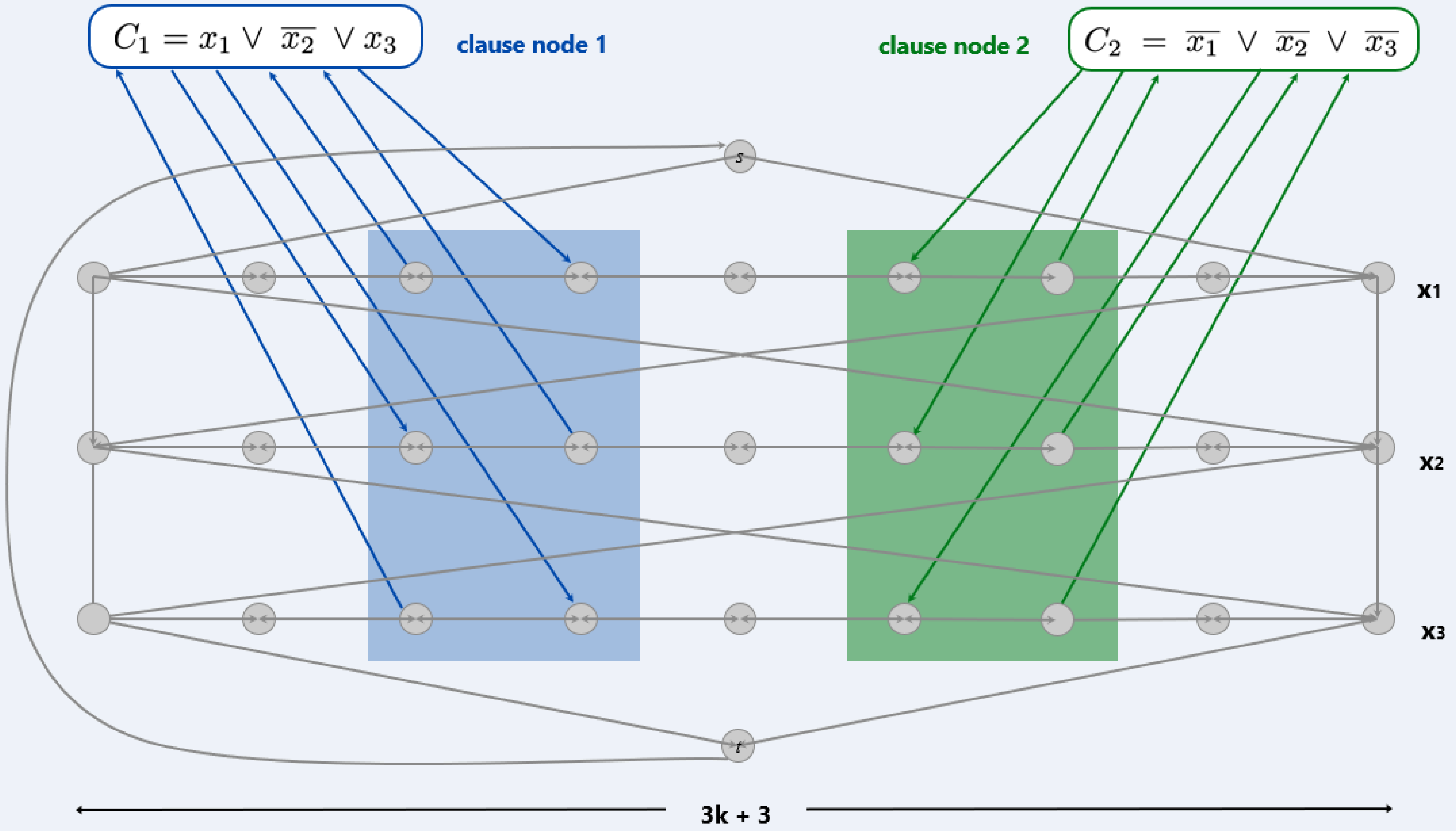
$$C_j = x_1 \vee \overline{x_2} \vee x_3$$



3-SAT $\leq_P$ Hamiltonian Cycle

Construction Given 3-Sat instance Φ with n variables x_i and k clauses.

- For each clause: add a node and 2 edges per literal.



3-SAT $\leq_P$ Hamiltonian Cycle

Lemma. Φ is satisfiable iff G has a Hamilton cycle.

Pf. $\Rightarrow$ (completeness)

- Suppose 3-Sat instance Φ has satisfying assignment x^*.
- Then, define Hamilton cycle Γ in G as follows:

-if $x^*_i = true$, traverse row i from left to right

-if $x^*_i = false$, traverse row i from right to left

-for each clause C_j , there will be at least one row i in which we are going in “correct” direction to splice clause node C_j into cycle

(and we splice in C_j exactly once) ▪

3-SAT $\leq_P$ Hamiltonian Cycle

Lemma. Φ is satisfiable iff G has a Hamilton cycle.

Pf. $\Leftarrow$ (soundness)

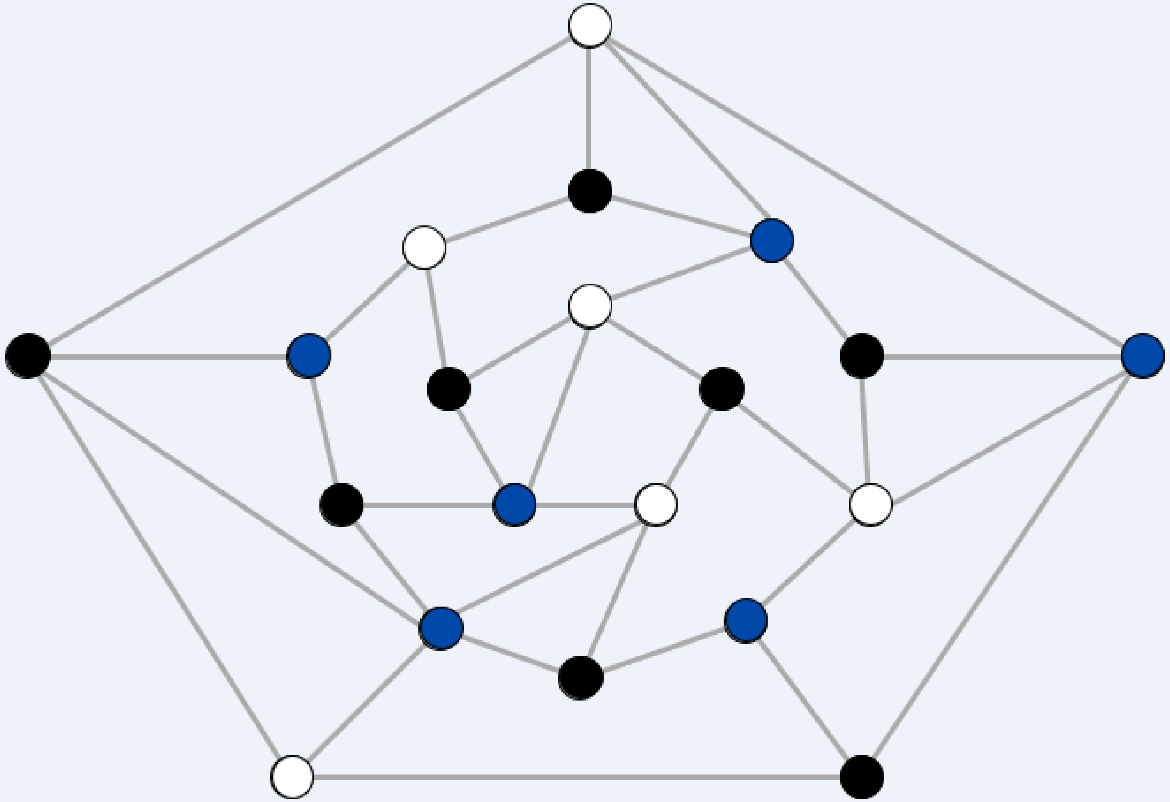
- Suppose G has a Hamilton cycle Γ.
- If Γ enters clause node C_j , it must depart on mate edge.
  - -nodes immediately before and after C_j are connected by an edge $e \in E$
  - -removing C_j from cycle, and replacing it with edge e yields Hamilton cycle on $G - \{ C_j \}$
- Continuing in this way, we are left with a Hamilton cycle Γ' in $G - \{ C_1 , C_2 , \ldots, C_k \}$.
- Set $x^*_i = true$ if Γ' traverses row i left-to-right; otherwise, set $x^*_i = false$.
- traversed in “correct” direction, and each clause is satisfied. ▪

Case study 2

3-SAT $\leq_p$ 3-Color

3-SAT $\leq_P$ 3-Color

3-COLOR Given an undirected graph G, can the nodes be colored black, white, and blue so that no adjacent nodes have the same color?



3-SAT $\leq_P$ 3-Color

Register allocation. Assign program variables to machine registers so that no more than k registers are used and no two program variables that are needed at the same time are assigned to the same register.

Interference graph. Nodes are program variables; edge between u and v if there exists an operation where both u and v are "live" at the same time.

Observation. [Chaitin 1982] Can solve register allocation problem iff interference graph is k-colorable.

Fact. 3-Color $\leq_P$ K-Register-Allocation for any constant $k \geq 3$.

3-SAT $\leq_P$ 3-Color

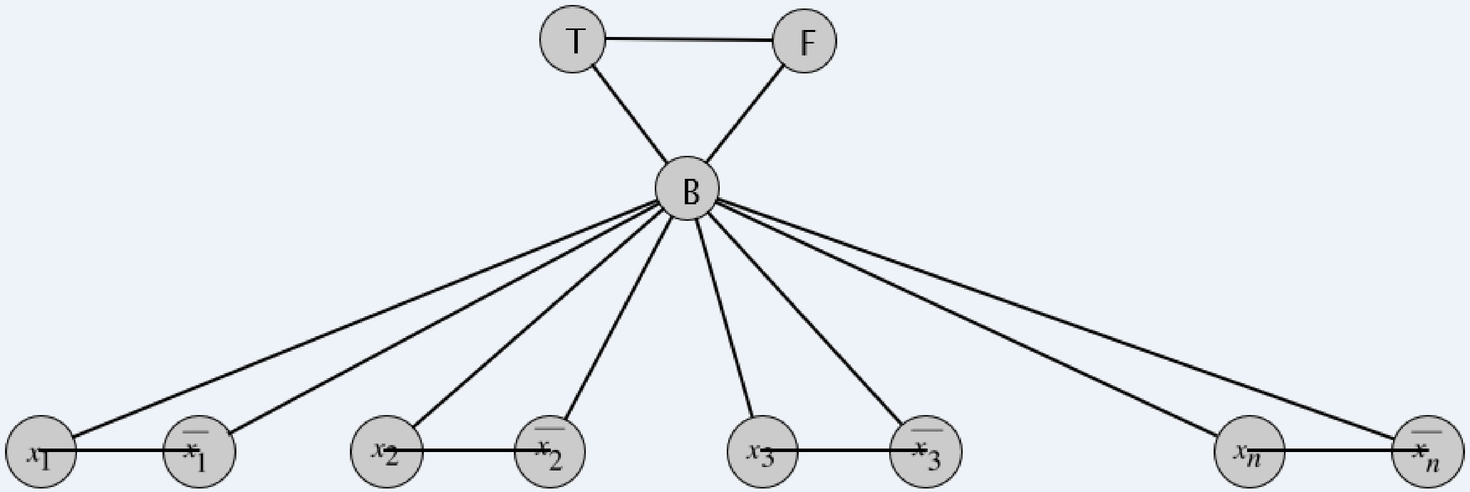
Theorem. 3-Sat $\leq_{\text{P}}$ 3-Color.

Pf. Given 3-Sat instance Φ, we construct an instance of 3-Color that is 3-colorable iff Φ is satisfiable.

3-SAT $\leq_P$ 3-Color

Construction.

(i) Create a graph G with a node for each literal.
(ii) Connect each literal to its negation.
(iii) Create 3 new nodes T, F, and B; connect them in a triangle.
(iv) Connect each literal to B.
(v) For each clause C_j, add a gadget of 6 nodes and 13 edges. (described later)



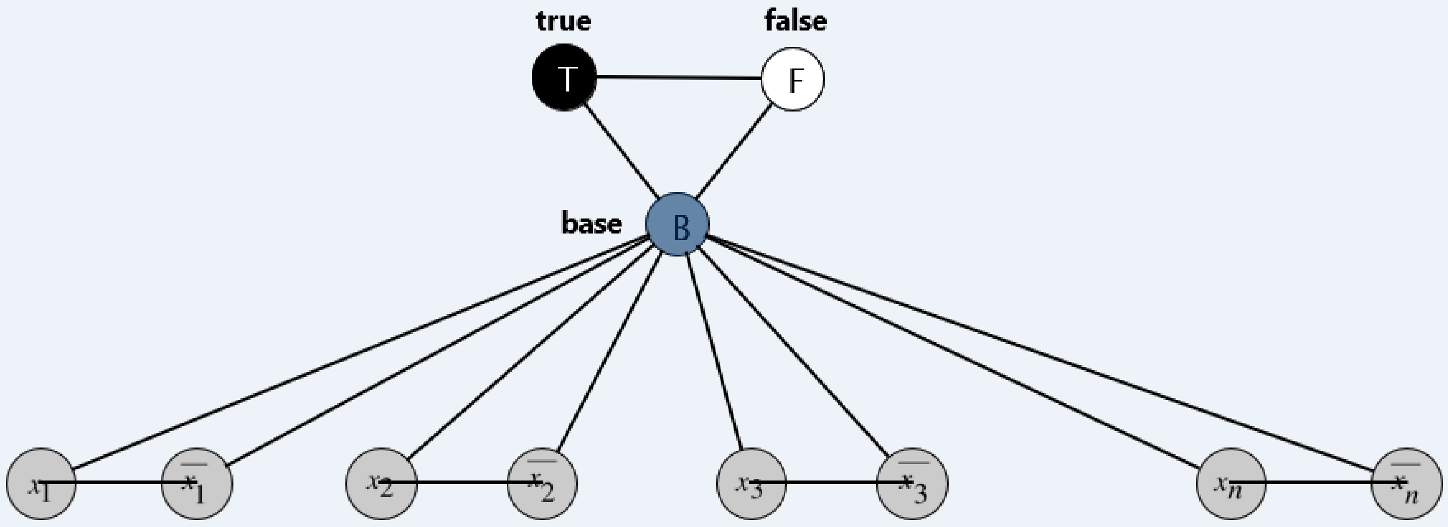
3-SAT $\leq_P$ 3-Color

Lemma. Graph G is 3-colorable iff Φ is satisfiable.

Pf. $\Rightarrow$ (completeness) Suppose graph G is 3-colorable.

WLOG, assume that node T is colored *black*, F is *white*, and B is *blue*.

- Consider assignment that sets all *black* literals to *true* (and *white* to *false*).
- (iv) ensures each literal is colored either *black* or *white*.
- (ii) ensures that each literal is *white* if its negation is *black* (and vice versa).

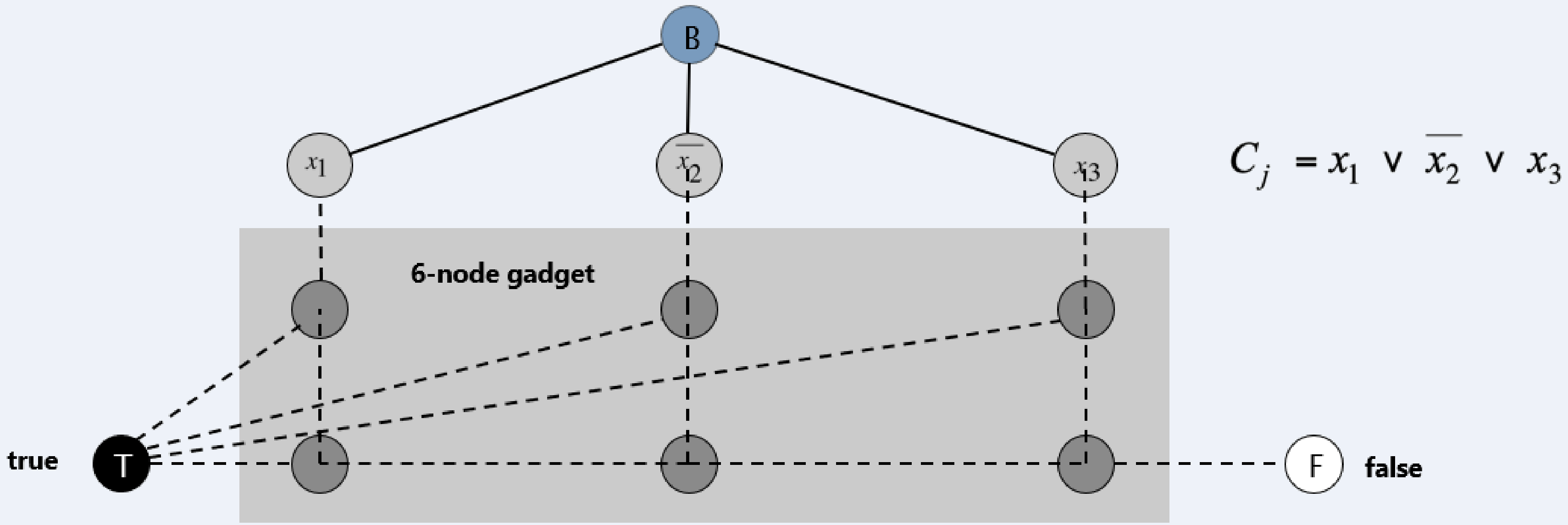


3-SAT $\leq_P$ 3-Color

Lemma. Graph G is 3-colorable iff Φ is satisfiable.

Pf. $\Rightarrow$ Suppose graph G is 3-colorable.

- WLOG, assume that node T is colored *black*, F is *white*, and B is *blue*.
- Consider assignment that sets all *black* literals to *true* (and *white* to *false*).
- (iv) ensures each literal is colored either *black* or *white*.
- (ii) ensures that each literal is *white* if its negation is *black* (and vice versa).
- (v) ensures at least one literal in each clause is *black*.

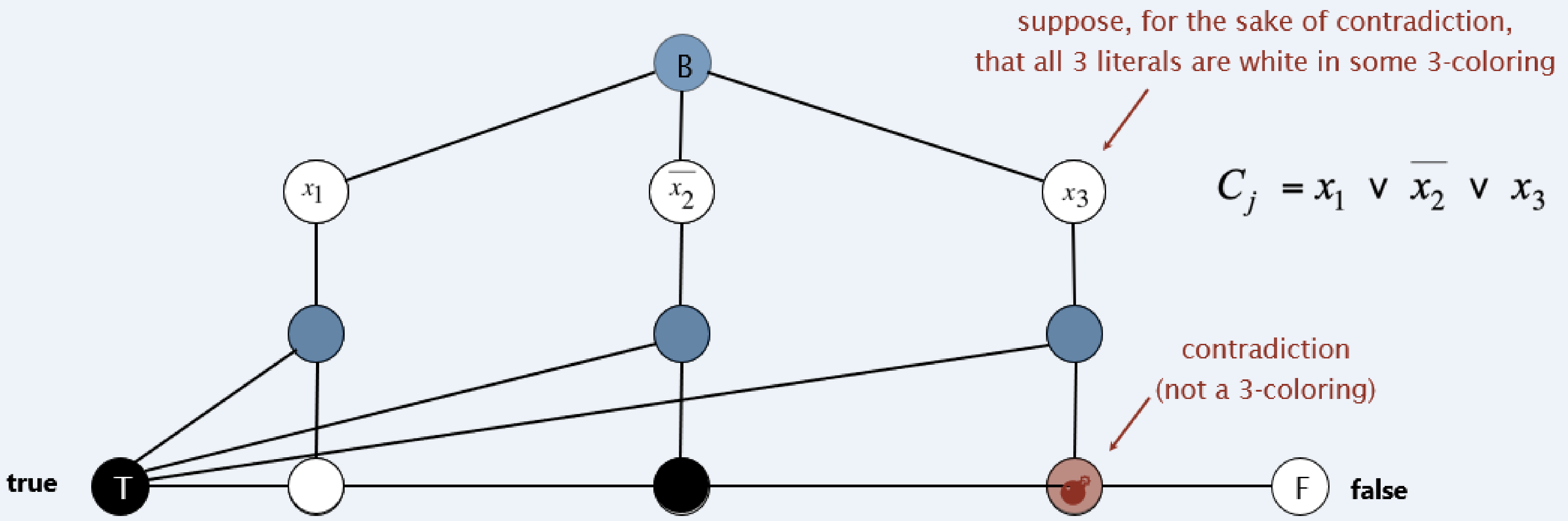


3-SAT $\leq_P$ 3-Color

Lemma. Graph G is 3-colorable iff Φ is satisfiable.

Pf. $\Rightarrow$ Suppose graph G is 3-colorable.

- WLOG, assume that node T is colored *black*, F is *white*, and B is *blue*.
- Consider assignment that sets all *black* literals to *true* (and *white* to *false*).
- (iv) ensures each literal is colored either *black* or *white*.
- (ii) ensures that each literal is *white* if its negation is *black* (and vice versa).
- (v) ensures at least one literal in each clause is *black*.

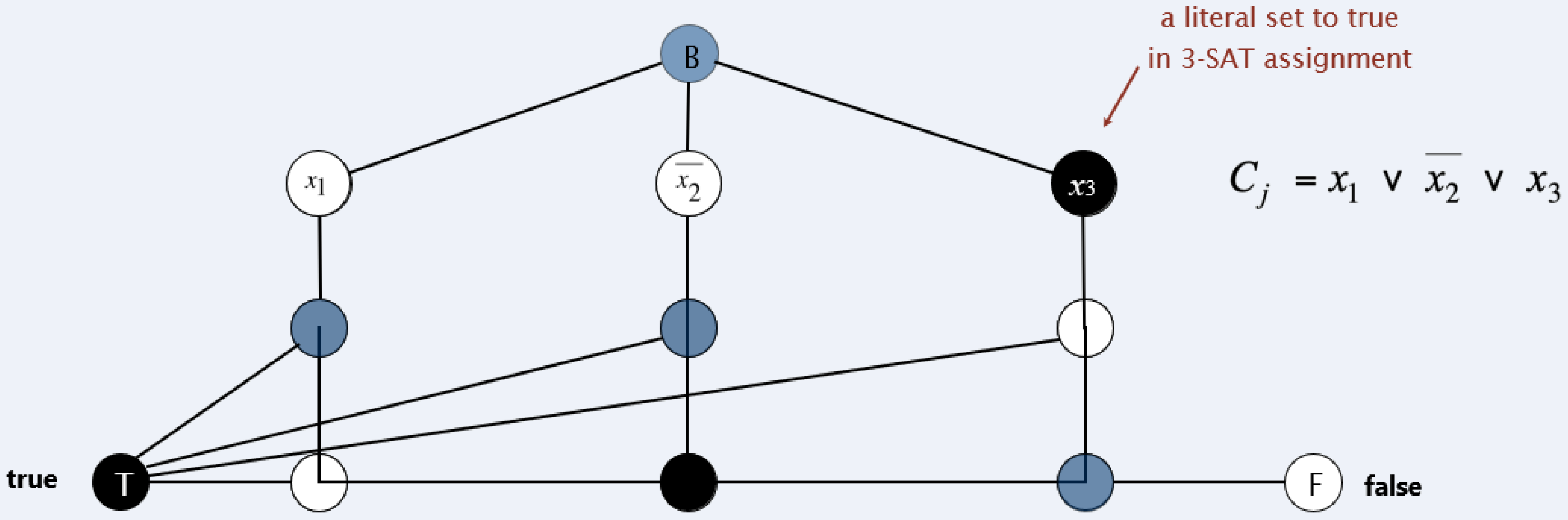


3-SAT $\leq_P$ 3-Color

Lemma. Graph G is 3-colorable iff Φ is satisfiable.

Pf. $\Leftarrow$(soundness) Suppose 3-Sat instance Φ is satisfiable.

- Color all *true* literals *black* and all *false* literals *white*.
- Pick one *true* literal; color node below that node *white*, and node below that *blue*.
- Color remaining middle row nodes *blue*.
- Color remaining bottom nodes *black* or *white*, as forced. ▪



Quiz Time