CS 401

NP-Complete

Xiaorui Sun

Stuff

Teaching evaluation

- Extra 1% score for all the students if overall response rate >= 80%
- Additional to the final score cut
- May improve the final grade (if overall response rate >= 80%)

Last Lecture

Decision problem: Problems that only outputs yes or no.

P: Decision problems for which there is a poly-time algorithm.

NP Decision problems for which there exists a poly-time certifier.

A certifier is to verify if a given certificate (proposed proof) t correctly proves that input s is a yes instance

- If the the certificate t proves that s is a yes instance, then the algorithm output yes
- If the the certificate t does not prove that s is a yes instance, then the algorithm output no
- (If t is a wrong proof for a yes input, the certifier still outputs no. If s is a no instance, the certifier should always output no because no certificate can prove s to be a yes instance.)

P, NP, EXP

P: Decision problems for which there is a poly-time algorithm.

EXP: Decision problems for which there is an exponential-time algorithm.

NP: Decision problems for which there is a poly-time certifier.

Claim $P \subseteq NP$.

Pf. Consider any problem X in P.

- By definition, there exists a poly-time algorithm A(s) that solves X.
- Certificate: t = empty string, certifier C(s, t) = A(s).

Claim $NP \subseteq EXP$.

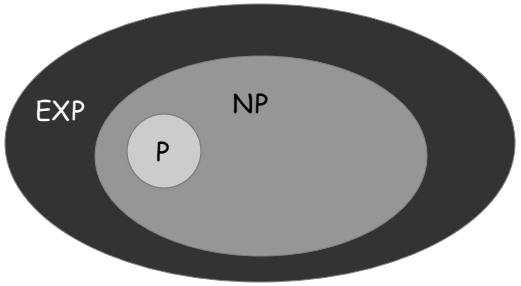
Pf. Consider any problem X in NP.

- By definition, there exists a poly-time certifier C(s, t) for X.
- To solve input s, run C(s, t) on all strings t with $|t| \leq p(|s|)$.
- Return `yes`, if C(s, t) returns `yes` for any of these. ▪

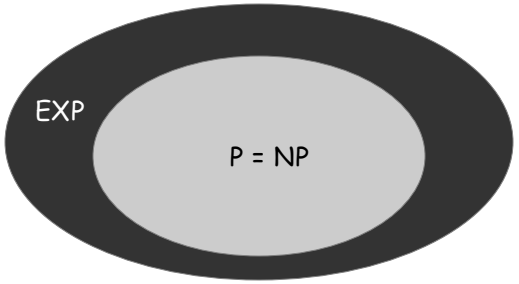
The Main Question: P Versus NP

Does P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]

- Is the decision problem as easy as the certification problem?
- Clay $1 million prize.



If P ≠ NP



If P = NP

would break RSA cryptography
(and potentially collapse economy)

If yes: Efficient algorithms for 3-COLOR, TSP, FACTOR, SAT, …

If no: No efficient algorithms possible for 3-COLOR, TSP, SAT, …

Consensus opinion on P = NP? Probably no.

The Simpson's: P = NP?

Summary

P: Decision problems for which there is a poly-time algorithm.

EXP: Decision problems for which there is an exponential-time algorithm.

NP: Decision problems for which there is a poly-time certifier.

Claim $P \subseteq NP$, $NP \subseteq EXP$.

Open question: Does P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]

- Is the decision problem as easy as the certification problem?

would break RSA cryptography (and potentially collapse economy)

If yes: Efficient algorithms for 3-COLOR, TSP, FACTOR, SAT, …

If no: No efficient algorithms possible for 3-COLOR, TSP, SAT, …

NP-Complete

NP Completeness

Complexity Theorists Approach: We don't know how to prove any problem in NP is hard. So, let's find hardest problems in NP.

NP-complete: A problem Y in NP with the property that for every problem X in NP, $X \leq_p Y$.

Motivations:

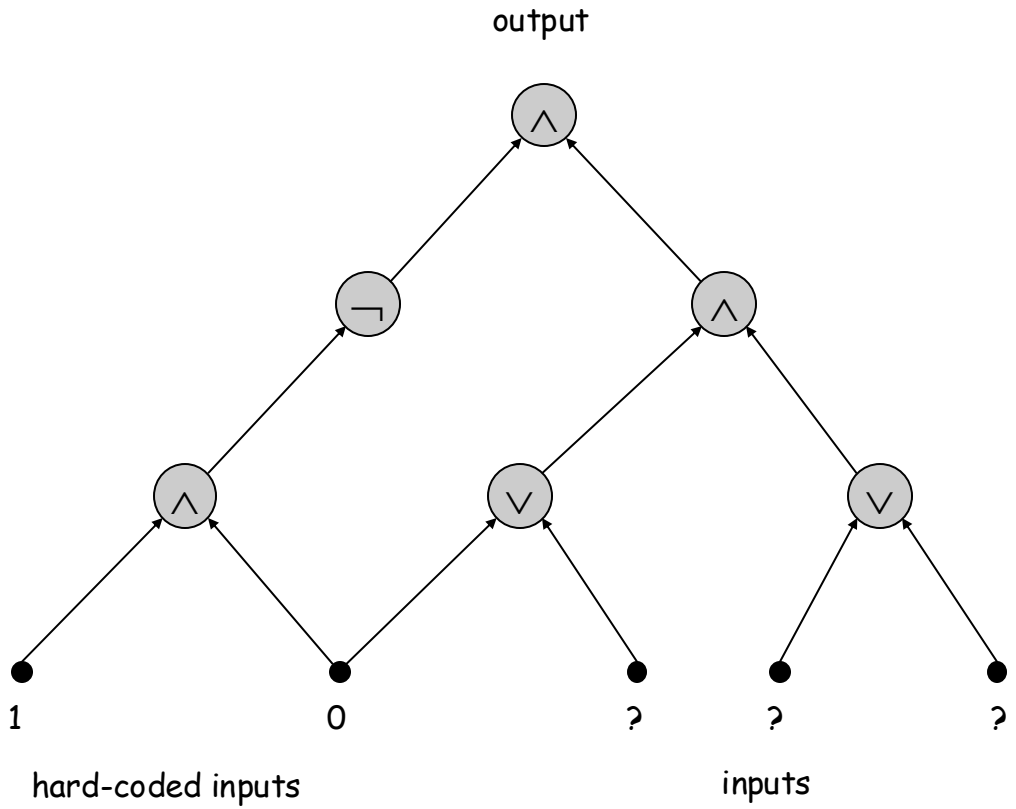
If $P \neq NP$, then every NP-Complete problems is not in P. So, we shouldn't try to design polytime algorithms

To show $P = NP$, it is enough to design a polynomial time algorithm for just one NP-complete problem.

The "First" NP-Complete Problem

CIRCUIT-SAT. Given a combinational circuit built out of AND, OR, and NOT gates, is there a way to set the circuit inputs so that the output is 1?

Theorem. CIRCUIT-SAT is NP-complete. [Cook 1971, Levin 1973]



yes: 1 0 1

Read KT 8.4 for the proof

Establishing NP-Completeness

Remark. Once we establish first "natural" NP-complete problem, others fall like dominoes.

Recipe to establish NP-completeness of problem Y.

- Step 1. Show that Y is in NP.
- Step 2. Choose a known NP-complete problem X.
- Step 3. Prove that $X \leq_p Y$.

Justification. If X is an NP-complete problem, and Y is a problem in NP with the property that $X \leq_P Y$ then Y is NP-complete.

Pf. Let W be any problem in NP. Then $W \leq_P X \leq_P Y$.

- By transitivity, $W \leq_P Y$.
- Hence Y is NP-complete. ▪

by definition of NP-complete

by assumption

3-SAT is NP-Complete

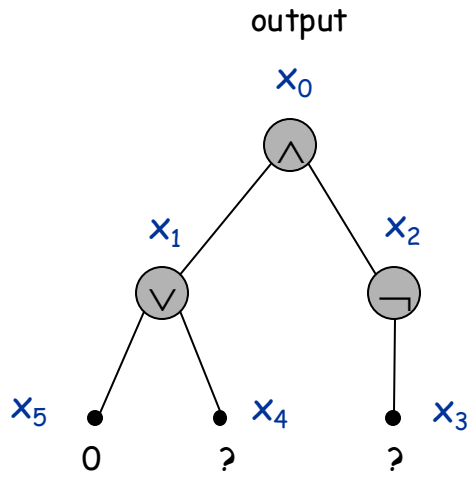
Theorem. 3-SAT is NP-complete.

Pf. Suffices to show that CIRCUIT-SAT $\leq_P$ 3-SAT since 3-SAT is in NP.

- Let K be any circuit.
- Create a 3-SAT variable x_i for each circuit element i.
- Make circuit compute correct values at each node:
  - $x_2 = \neg x_3$ $\Rightarrow$ add 2 clauses: $x_2 \vee x_3$, $\overline{x_2} \vee \overline{x_3}$
  - $x_1 = x_4 \vee x_5$ $\Rightarrow$ add 3 clauses: $x_1 \vee \overline{x_4}$, $x_1 \vee \overline{x_5}$, $\overline{x_1} \vee x_4 \vee x_5$
  - $x_0 = x_1 \wedge x_2$ $\Rightarrow$ add 3 clauses: $\overline{x_0} \vee x_1$, $\overline{x_0} \vee x_2$, $x_0 \vee \overline{x_1} \vee \overline{x_2}$

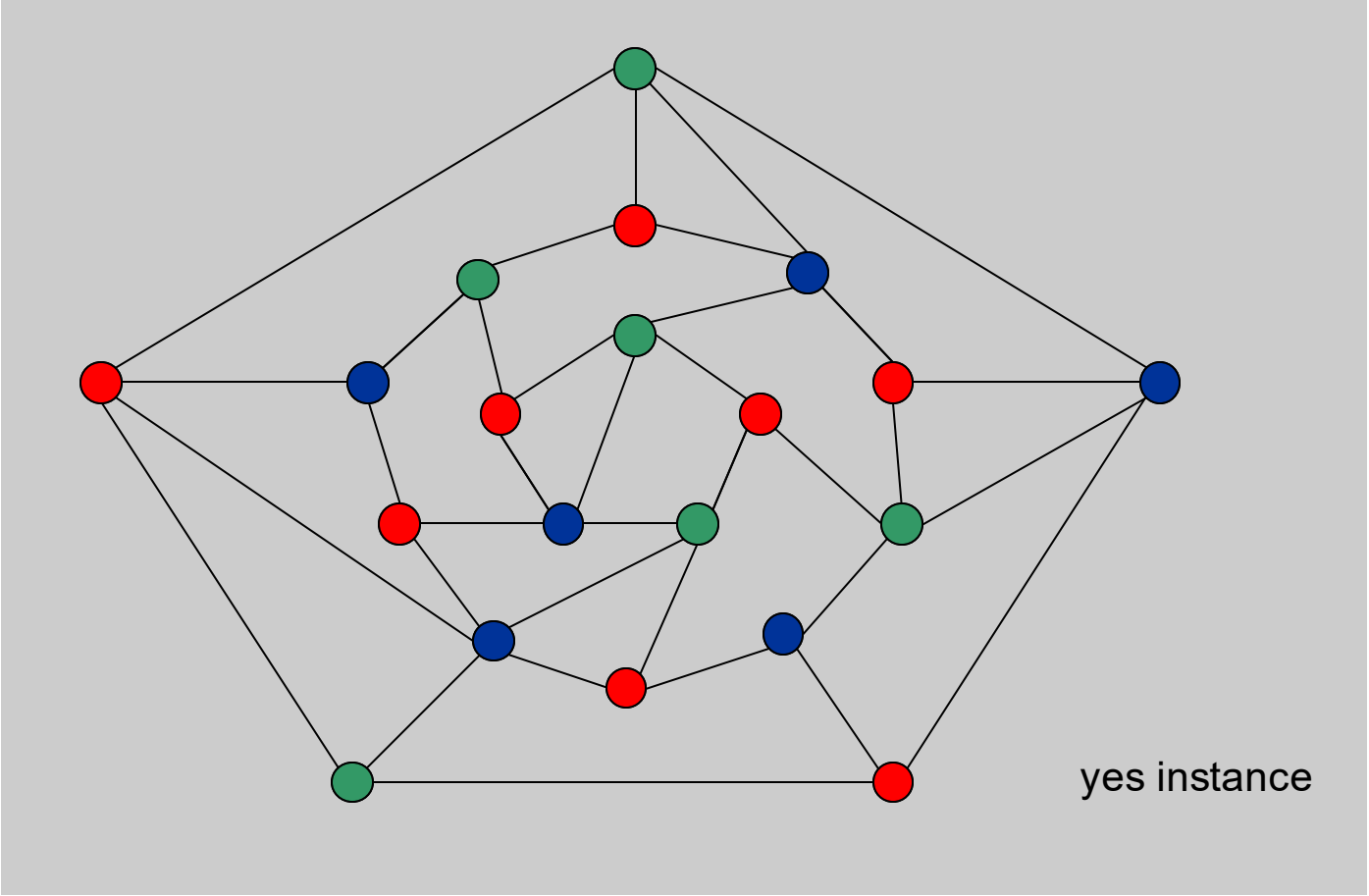
- Hard-coded input values and output value.
  - $x_5 = 0$ $\Rightarrow$ add 1 clause: $\overline{x_5}$
  - $x_0 = 1$ $\Rightarrow$ add 1 clause: x_0

- Final step: turn clauses of length < 3 into clauses of length exactly 3. ▪



3-Colorability

3-COLOR: Given an undirected graph G does there exists a way to color the nodes red, green, and blue so that no adjacent nodes have the same color?



3-Colorability

Claim. 3-SAT $\leq_P$ 3-COLOR.

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

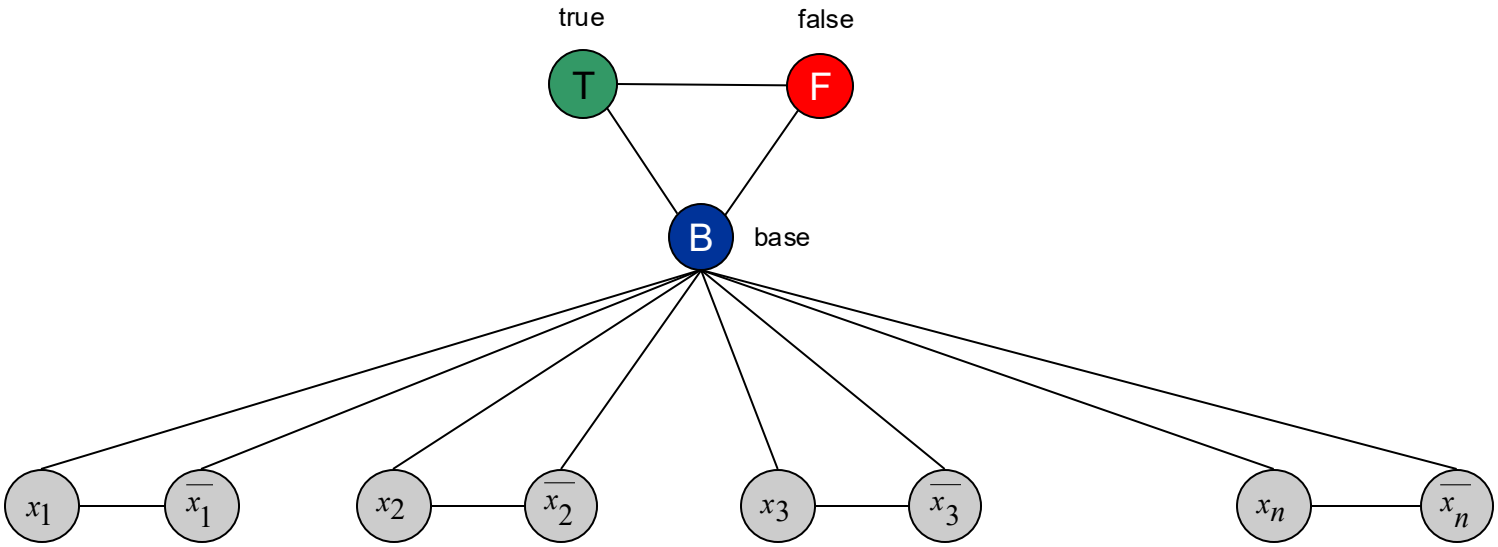
3-Colorability

Construction.

i. For each literal, create a node.

ii. Create 3 new nodes T, F, B; connect them in a triangle, and connect each literal to B.

iii. Connect each literal to its negation.



3-Colorability

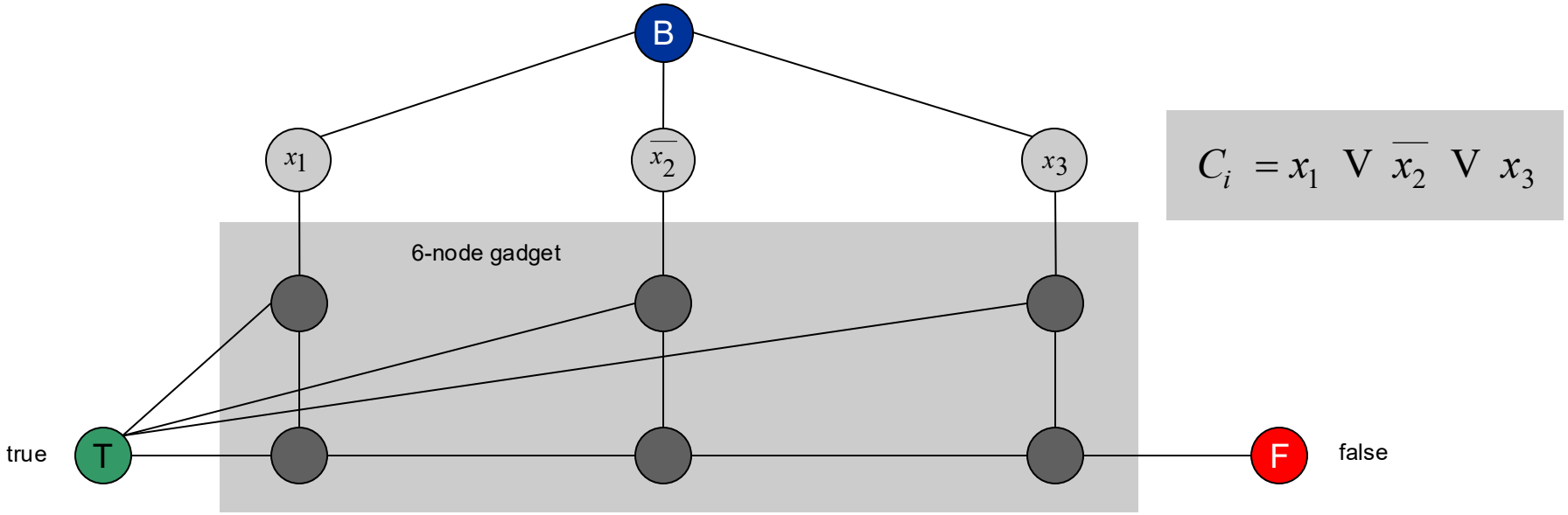
Construction.

i. For each literal, create a node.

ii. Create 3 new nodes T, F, B; connect them in a triangle, and connect each literal to B.

iii. Connect each literal to its negation.

iv. For each clause, add gadget of 6 nodes and 13 edges.

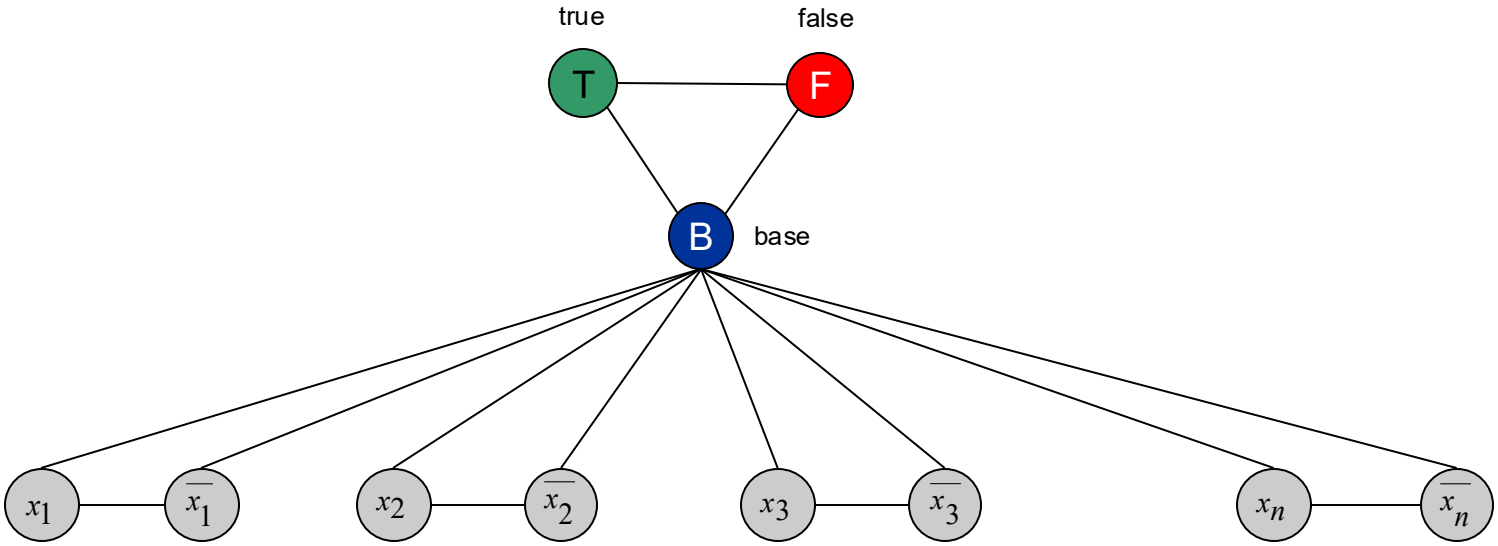


3-Colorability

Claim. Graph is 3-colorable iff Φ is satisfiable.

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

- Consider assignment that sets all T literals to true.
- (ii) ensures each literal is T or F.
- (iii) ensures a literal and its negation are opposites.

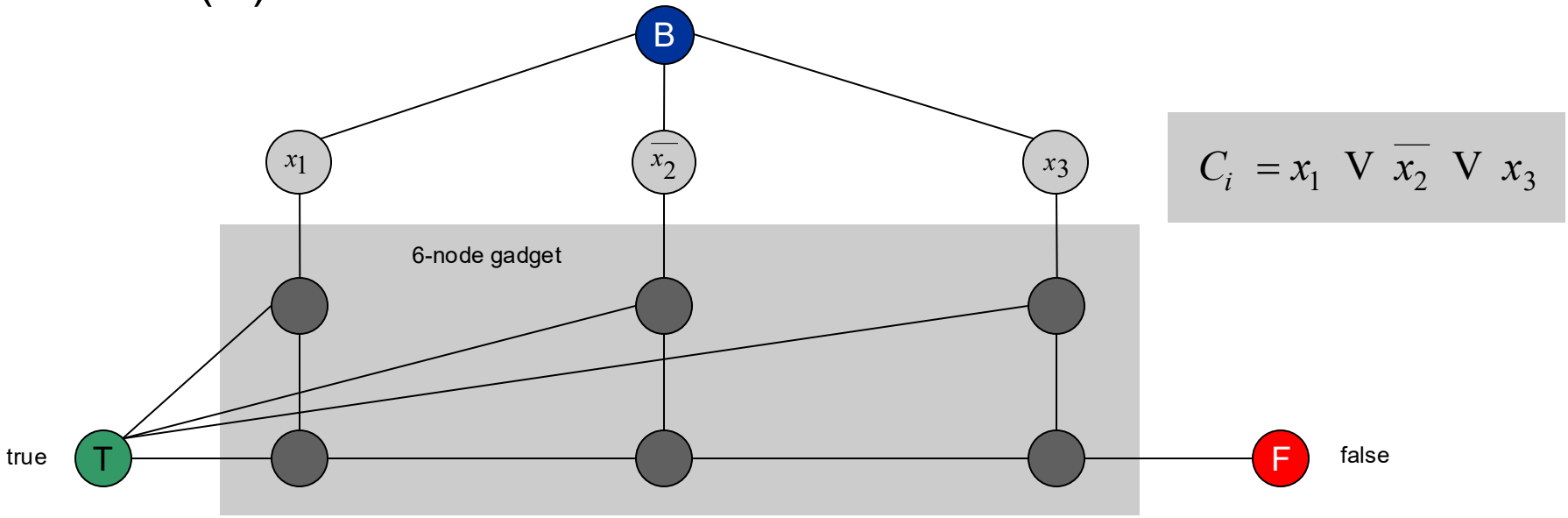


3-Colorability

Claim. Graph is 3-colorable iff Φ is satisfiable.

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

- Consider assignment that sets all T literals to true.
- (ii) ensures each literal is T or F.
- (iii) ensures a literal and its negation are opposites.
- (iv) ensures at least one literal in each clause is T.

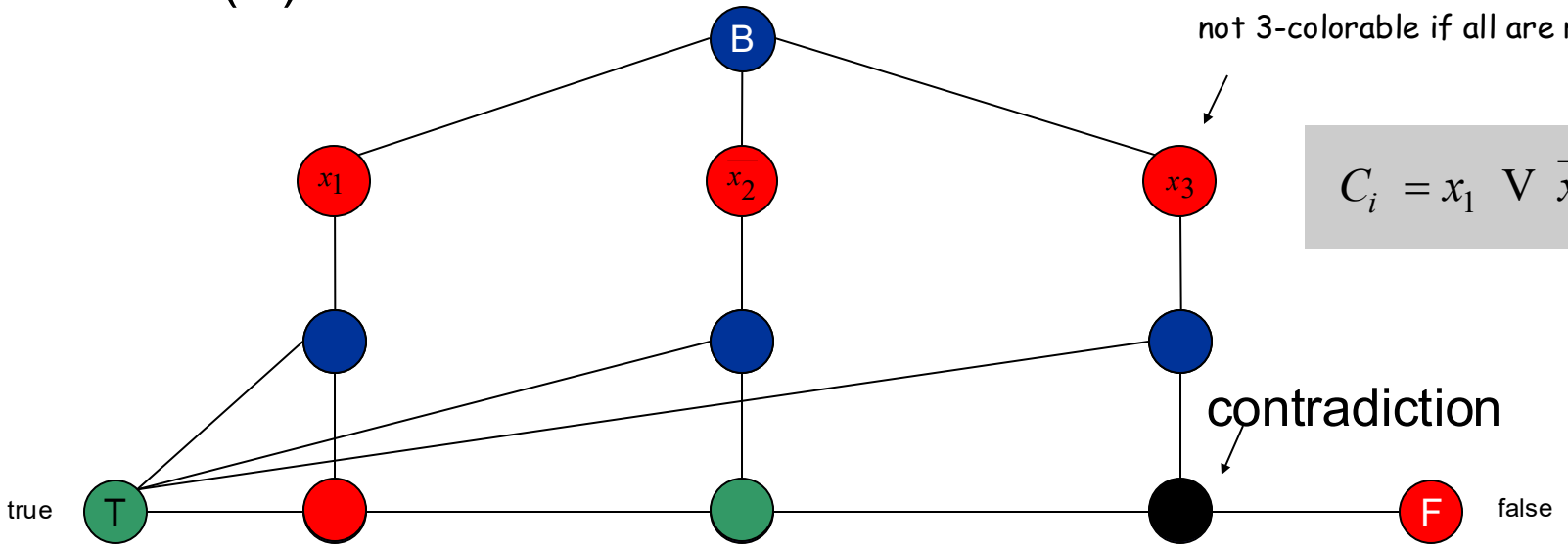


3-Colorability

Claim. Graph is 3-colorable iff Φ is satisfiable.

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

- Consider assignment that sets all T literals to true.
- (ii) ensures each literal is T or F.
- (iii) ensures a literal and its negation are opposites.
- (iv) ensures at least one literal in each clause is T.

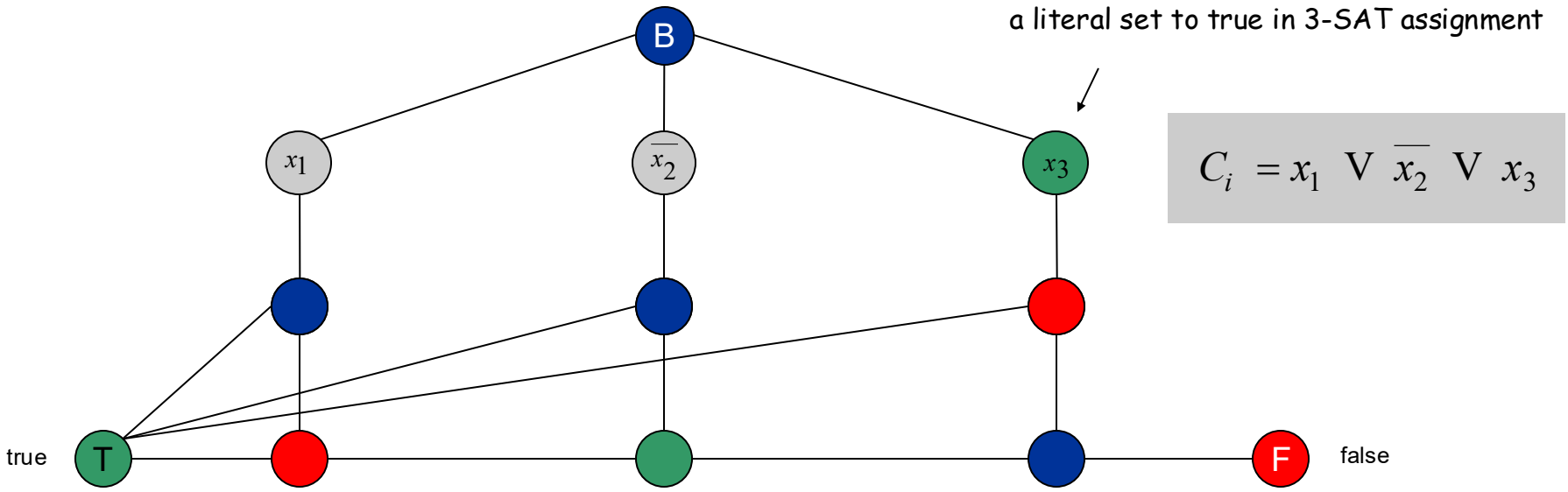


3-Colorability

Claim. Graph is 3-colorable iff Φ is satisfiable.

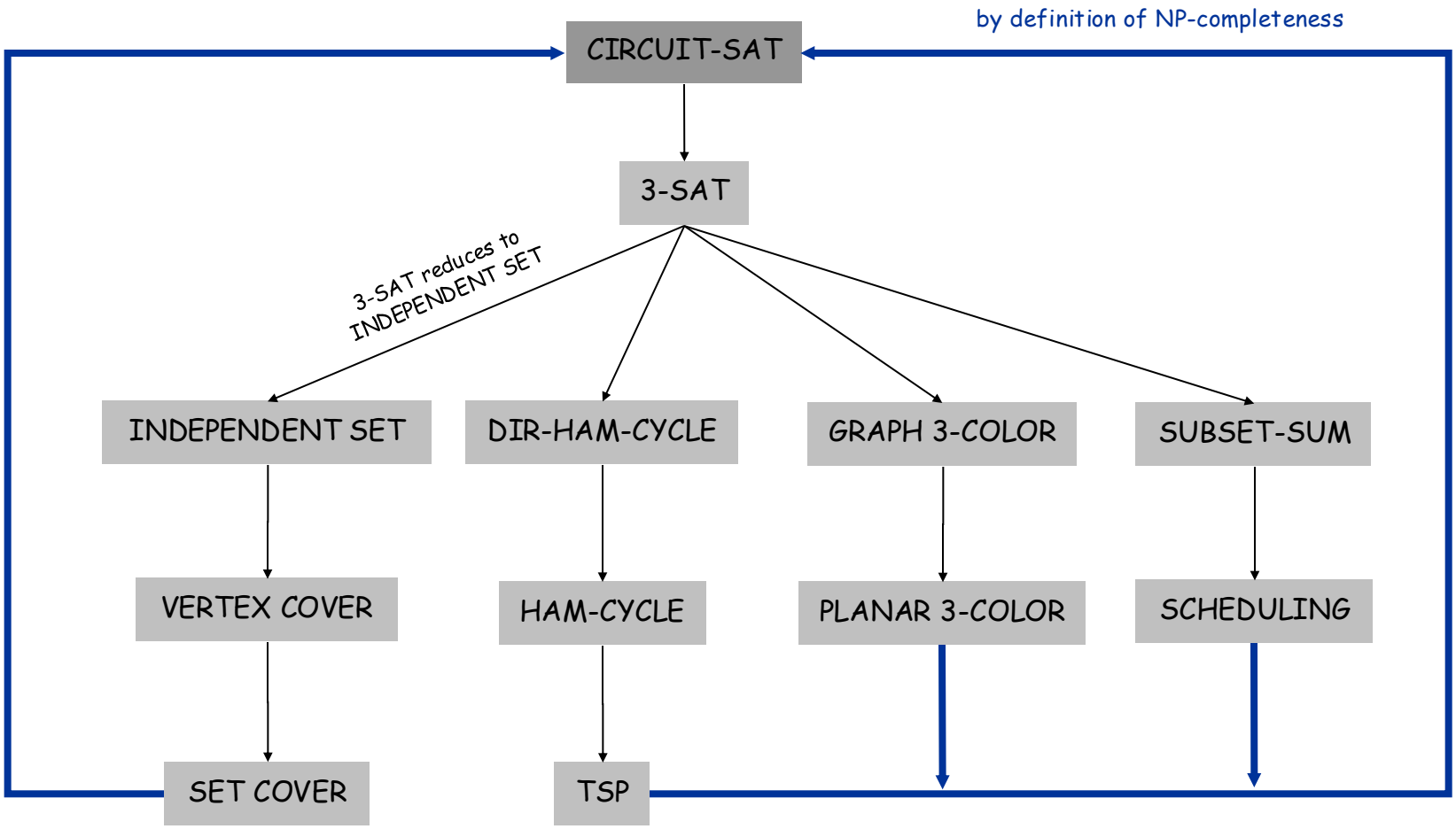
Pf. $\Leftarrow$ Suppose 3-SAT formula Φ is satisfiable.

- Color all true literals T.
- Color node below green node F, and node below that B.
- Color remaining middle row nodes B.
- Color remaining bottom nodes T or F as forced. ▪



NP-Completeness

Observation. All problems below are NP-complete and polynomial reduce to one another!



Some NP-Complete Problems

Six basic genres of NP-complete problems and paradigmatic examples.

- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING 3-COLOR.
- Numerical problems: SUBSET-SUM, KNAPSACK.

Practice. Most NP problems are either known to be in P or NP-complete.

Notable exceptions. Factoring, graph isomorphism.

Determine if a composite number c has a factor $\leq k$

Summary

We learned the crucial idea of polynomial-time reduction. This can be even used in algorithm design, e.g., we know how to solve max-flow so we reduce image segmentation to max-flow

Polynomial-time reductions are transitive relations

NP: Set of all decision problems for which there exists a poly-time certifier.

NP-complete problems are the hardest problem in NP