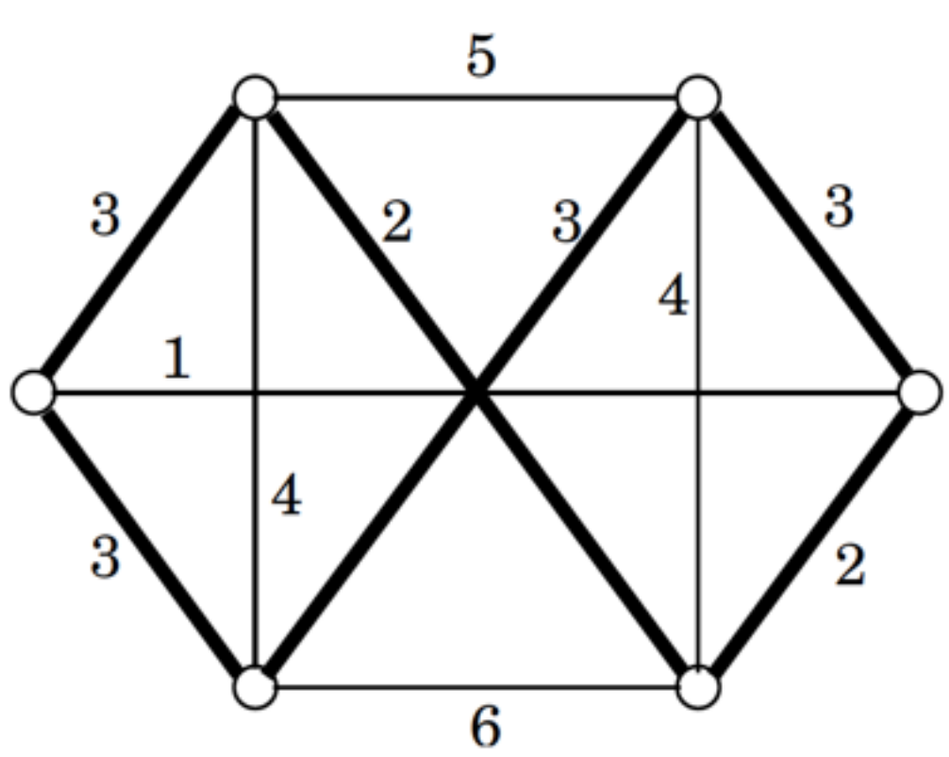
NP-Complete Problems

CSC 282



The Story So Far

- NP
  - Decision problems that can be solved in polynomial time on a non-deterministic Turing Machine
  - Search problems whose solution can be verified in polynomial time
- Reduction: a polynomial time transformation A→B
  - Of problems in class A to problems in class B, and
  - Of solutions in class B to solutions in class A
  - Hardness goes →, easiness goes ←
  - SAT → CNF-SAT → 3SAT

Graph Coloring Problem

Graph Coloring Problem

Given a graph G, can you color the nodes with $\leq k$ colors such that the endpoints of every edge are colored differently?

Notation: A k-coloring is a function $f : V \to \{1, \ldots, k\}$ such that for every edge $\{u, v\}$ we have $f(u) \neq f(v)$.

If such a function exists for a given graph G, then G is k-colorable.

Special case of $k = 2$

How can we test if a graph has a 2-coloring?

Check if the graph is bipartite.

Unfortunately, for $k \geq 3$, the problem is NP-complete.

Theorem

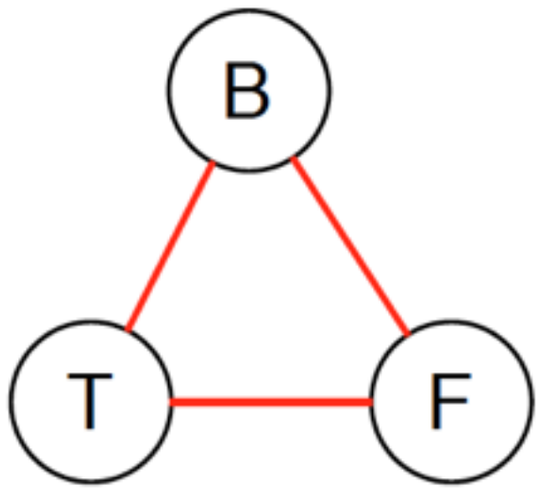
3-Coloring is NP-complete.

Reduction from 3-SAT

We construct a graph G that will be 3-colorable iff the 3-SAT instance is satisfiable.

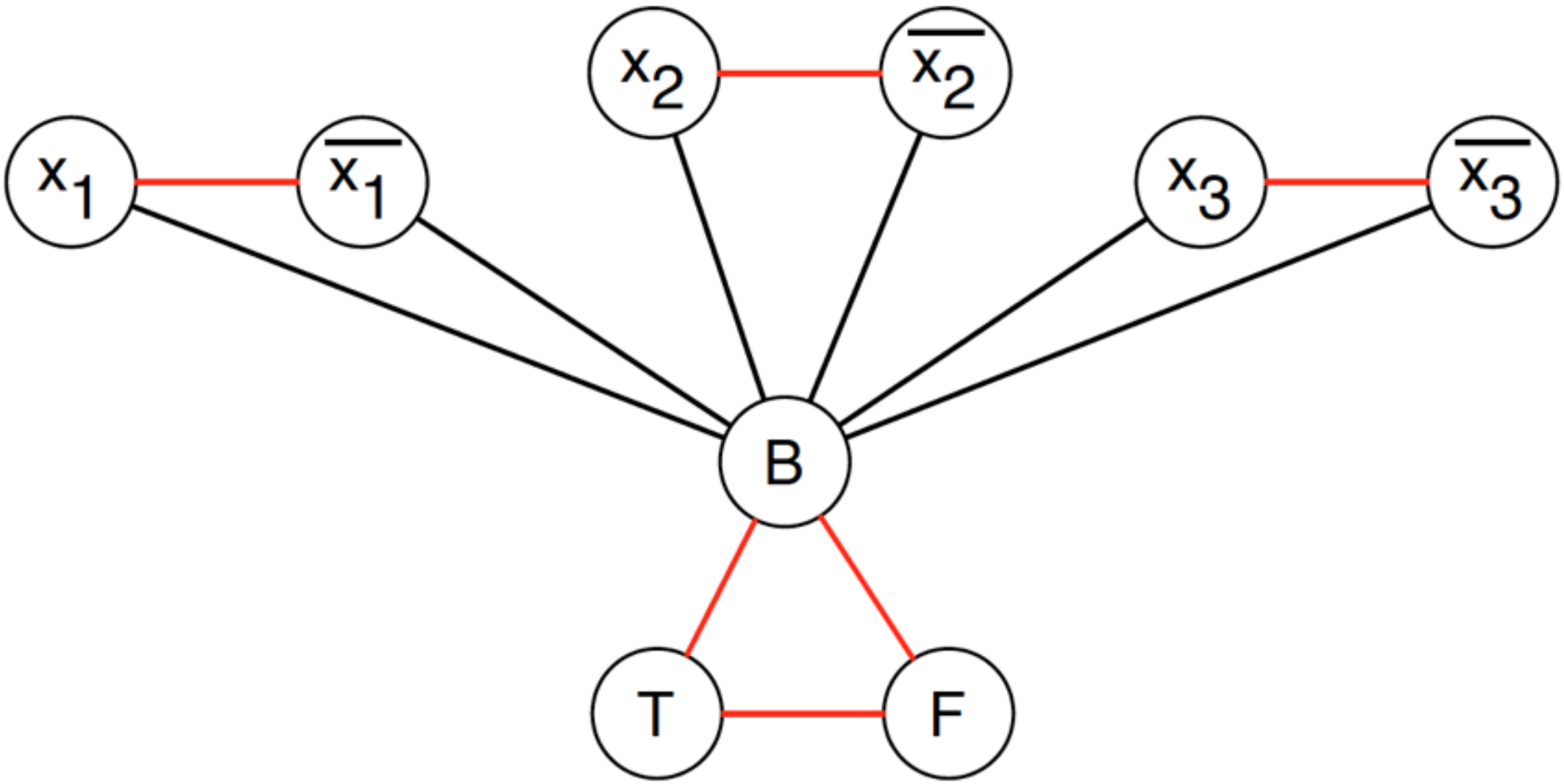
For every variable x_i, create 2 nodes in G, one for x_i and one for $\bar{x}_i$. Connect these nodes by an edge:

Create 3 *special nodes* T, F, and B, joined in a triangle:



Connecting them up

Connect every variable node to B:



Properties

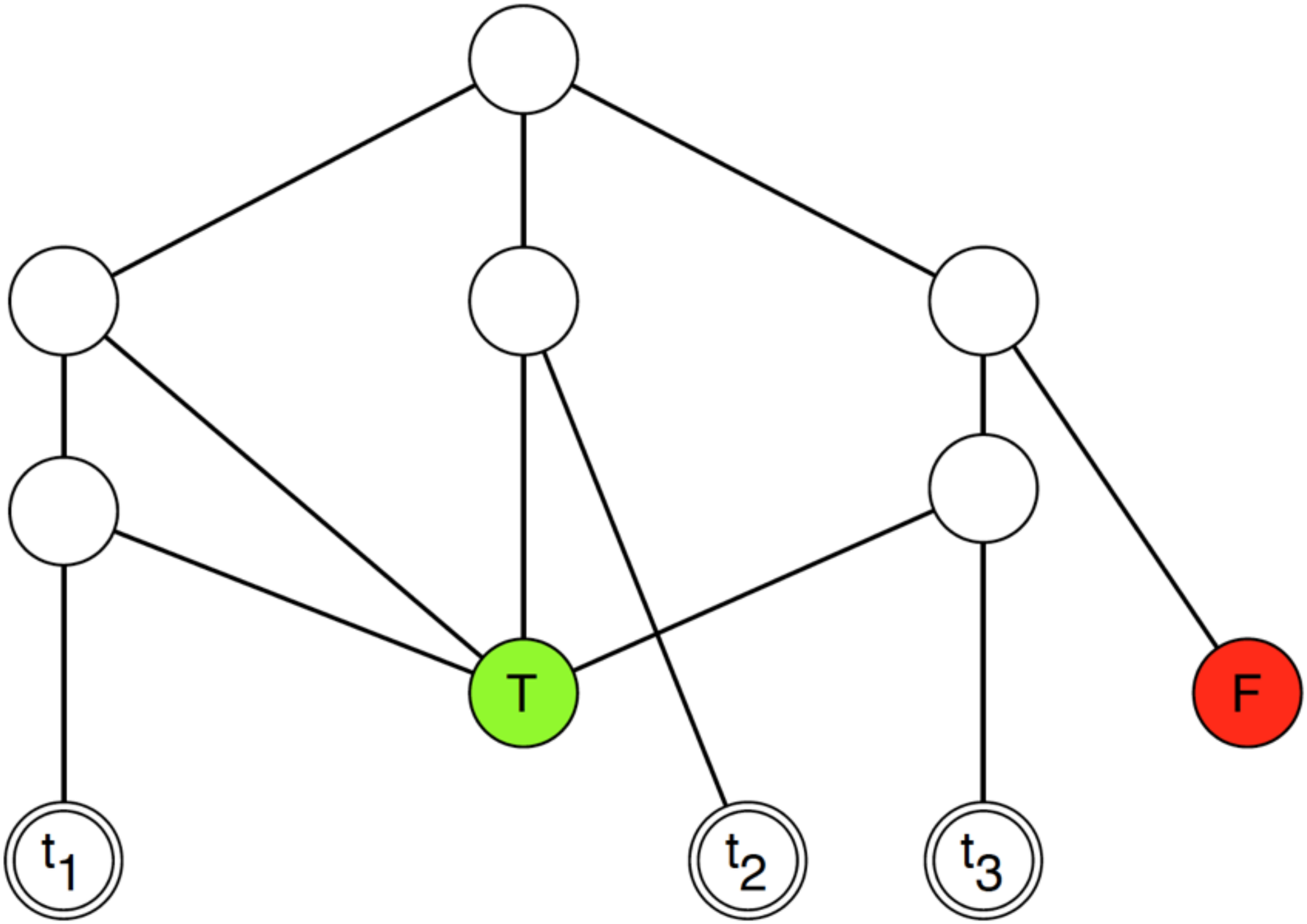
Properties:

- Each of x_i and $\bar{x}_i$ must get different colors
- Each must be different than the color of B.
- B, T, and F must get different colors.

Hence, any 3-coloring of this graph defines a valid truth assignment!

Still have to constrain the truth assignments to satisfy the given clauses, however.

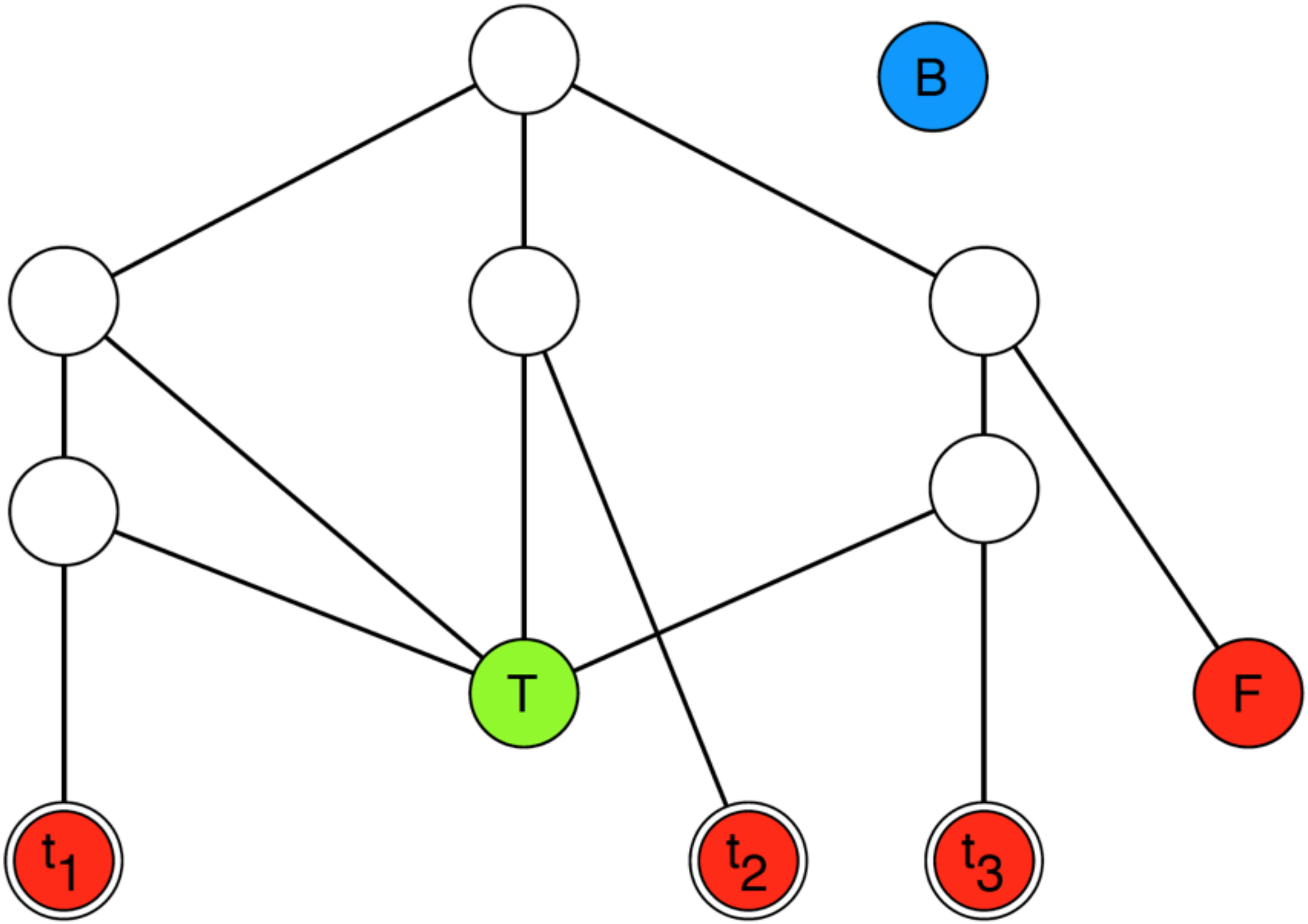
Connect Clause (t_1, t_2, t_3) up like this:



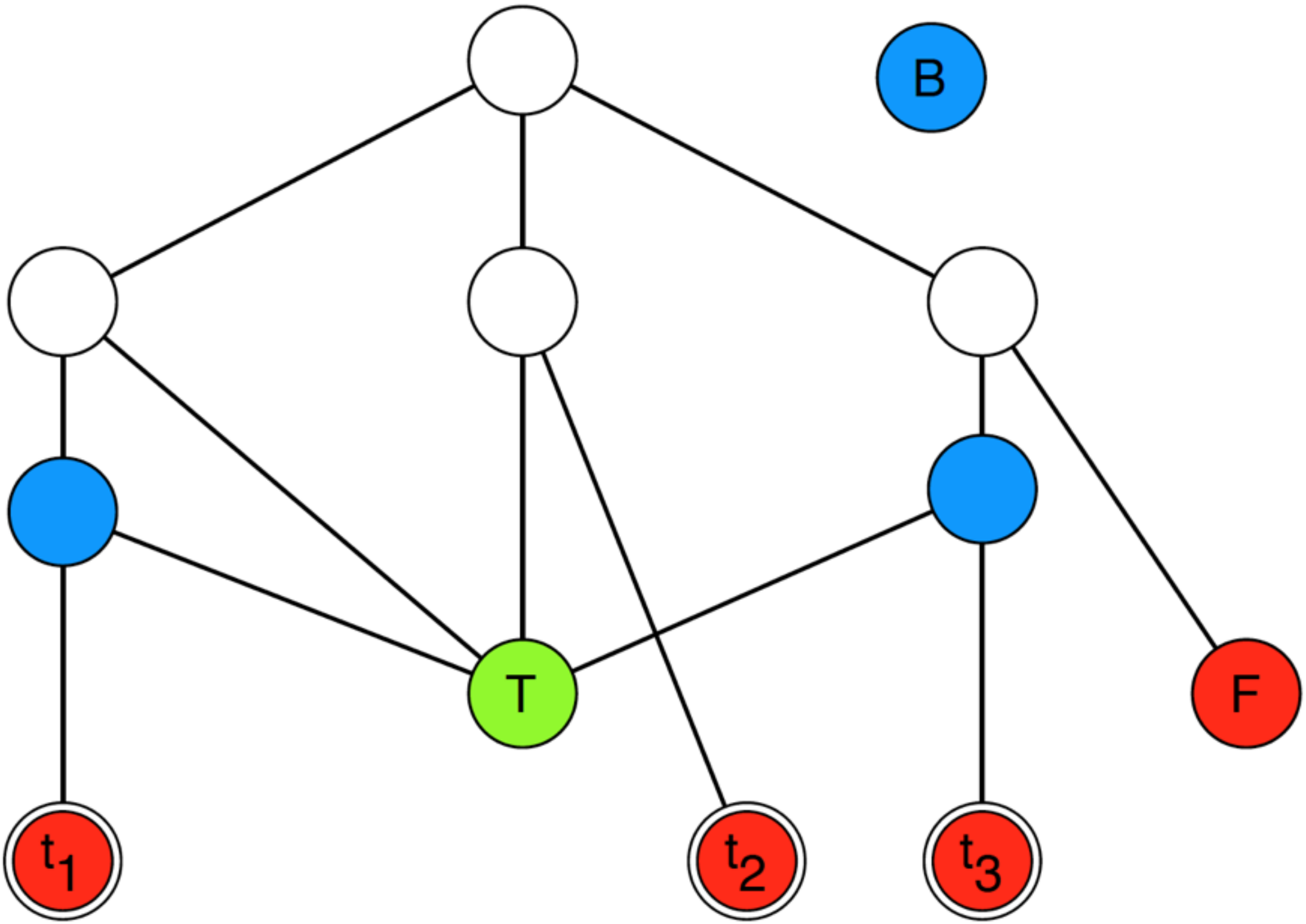
Suppose Every Term Was False

What if every term in the clause was assigned the **false** color?

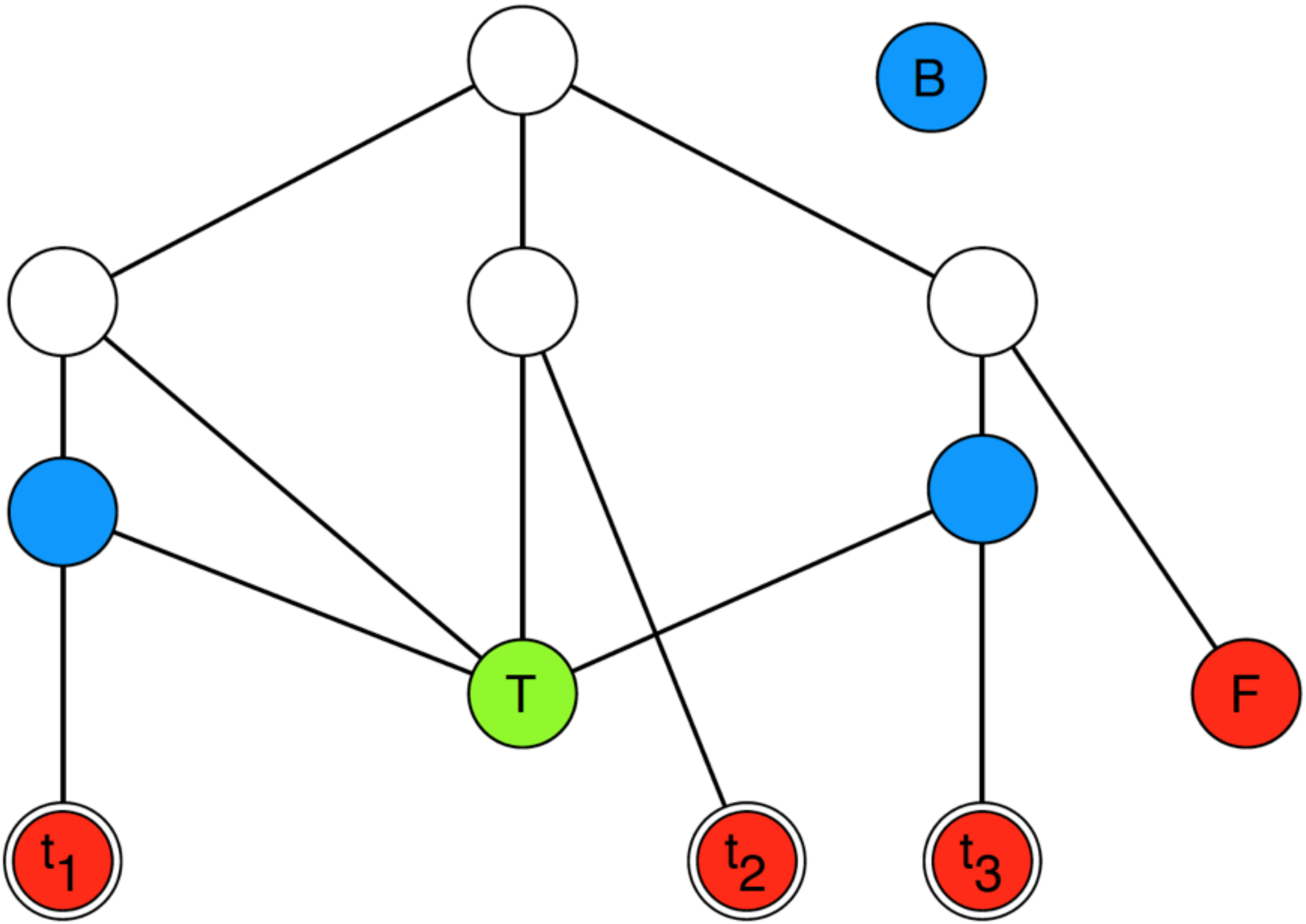
Connect Clause (t_1, t_2, t_3) up like this:



Connect Clause (t_1, t_2, t_3) up like this:



Connect Clause (t_1, t_2, t_3) up like this:



Suppose there is a 3-coloring

Top node is colorable iff one of its terms gets the **true** color.

Suppose there is a 3-coloring.

We get a satisfying assignment by:

- Setting $x_i =$ **true** iff v_i is colored the same as T

Let C be any clause in the formula. At least 1 of its terms must be true, because if they were all false, we couldn't complete the coloring (as shown above).

Suppose there is a satisfying assignment

Suppose there is a satisfying assignment.

We get a 3-coloring of G by:

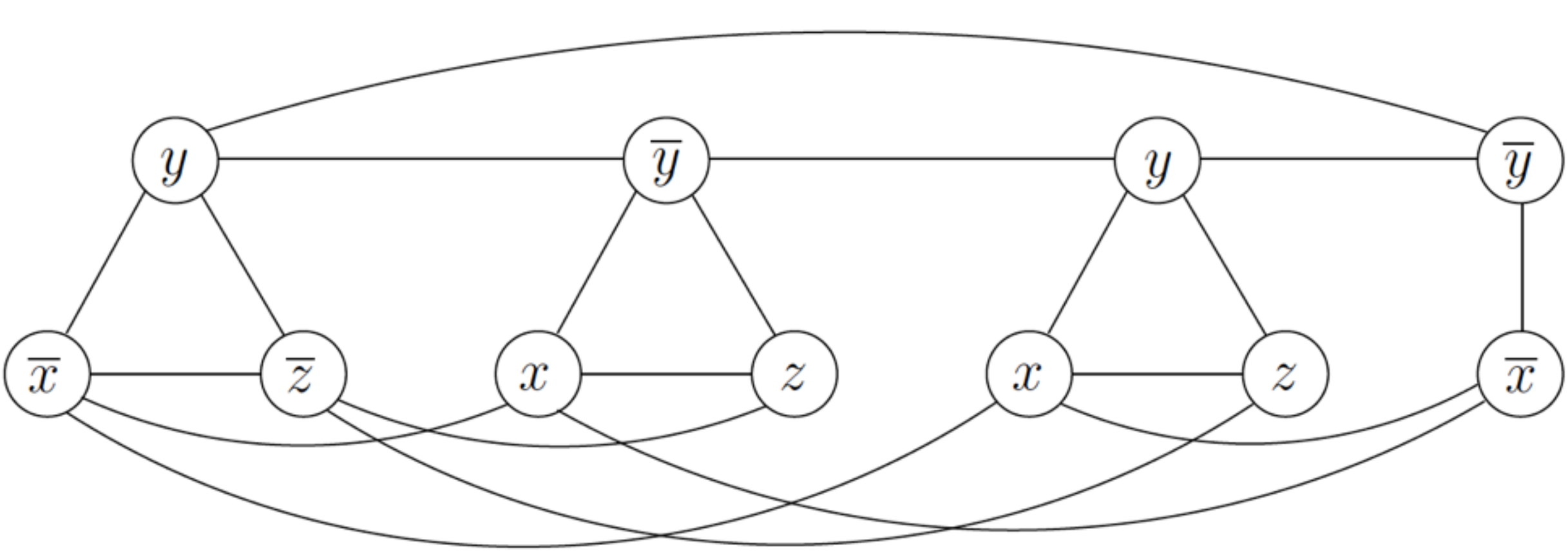
- Coloring T, F, B arbitrarily with 3 different colors
- If $x_i =$ **true**, color v_i with the same color as T and $\bar{v}_i$ with the color of F.
- If $x_i =$ **false**, do the opposite.
- Extend this coloring into the clause gadgets.

Hence: the graph is 3-colorable iff the formula it is derived from is satisfiable.

3SAT→Independent Set

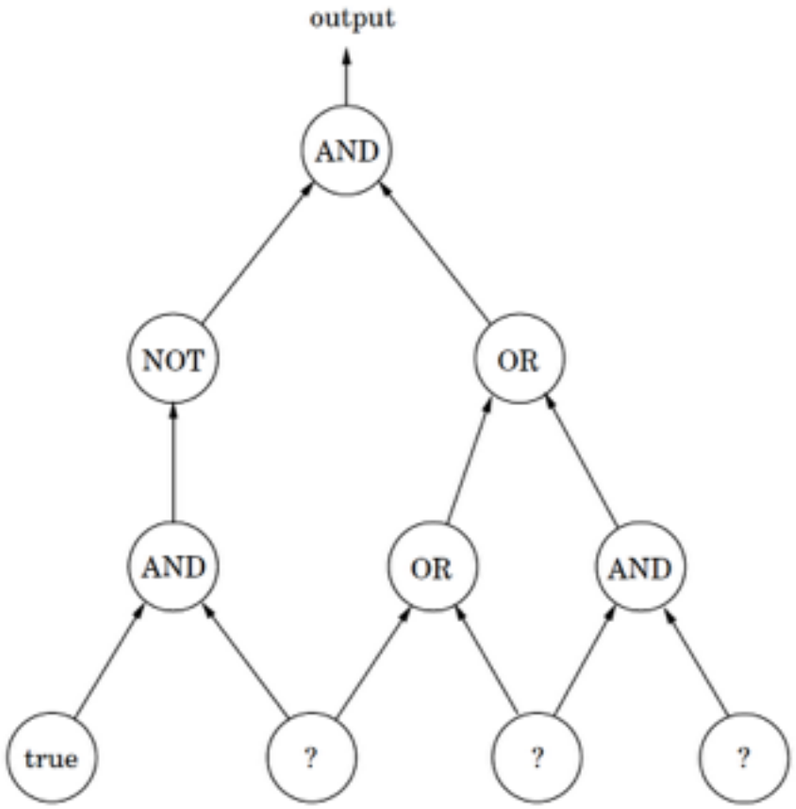
- Independent Set: Given a graph G and number g, find a set of g pairwise non-adjacent vertices

The graph corresponding to $(\overline{x} \vee y \vee \overline{z})\ (x \vee \overline{y} \vee z)\ (x \vee y \vee z)\ (\overline{x} \vee \overline{y})$**.**



NP-Complete

- A problem is NP-Complete if
  - It is in NP
  - Any problem in NP can be reduced to it
- SAT is NP-Complete
  - Short proof:
    - Circuit-SAT → SAT
    - Any problem solvable by a Turing Machine in polynomial time can be solved by circuit of polynomial size



A Slightly Longer Proof

- Instead of a non-deterministic Turing Machine, consider an equivalent model of computing:
  - An ordinary (von-Neumann) style computer with a polynomial amount of (binary) RAM
  - Non-deterministic “choose” instruction

```
choose { x=1;} or { x=2;}
```

  - `FAIL` instruction means answer is “no”
  - Answer is “yes” is some series of choices does not end in a fail

Simulation

- Write a Boolean formula that simulates the operation of this machine
- Boolean variable for each memory location at each time step (recall that time is bounded!)
- Boolean variables for the instruction counter and registers at each time step
- Write implications that capture the instruction set

State Transition Formulas

at time 1:
instruction counter is 0001 &
instruction at location 0001 is “increment” &
argument to instruction is location 0101 &
value at location 0101 is “0” ⇒

at time 2:
value at location 0101 is “1”

Looking Ahead

- Last assignment #5 given out Monday

- Will not be turned in

- Instead, material will be covered in exam on last day of class, Dec 12

Kinds of Transitions

- Deterministic transition: formula exactly specifies the state after the execution of the instruction

- Non-deterministic transition: formula allows 2 or more states to possibly result from the execution of an instruction

- Transition to FAIL: a deliberately inconsistent subformula

Reduction

- For any problem in NP, there is an algorithm with some time polynomial time bound p
- Encode a non-deterministic computer with time and space bound p
- Encode the program itself and its input as the initial state of the computer
- Formula is satisfiable iff program does not necessarily fail (answer “no”)