More NP-complete problems [P&S §15.6, 15.7]

7 July

Recall: Definitions of polynomial transformation, NP-completeness.

Yesterday: Cook's theorem shows that SAT is NP-complete.

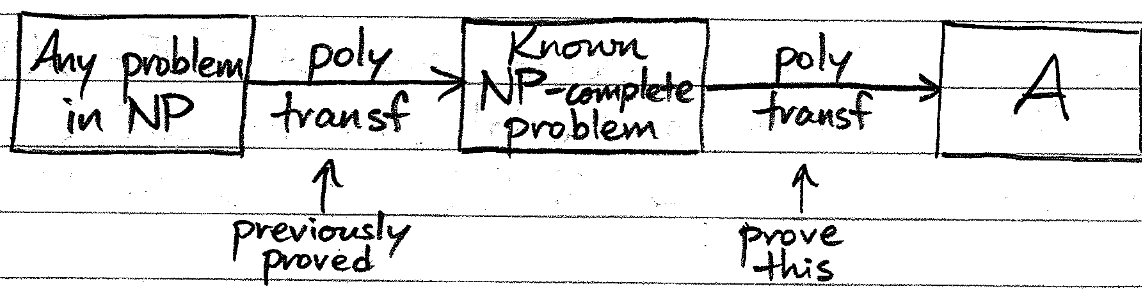
Actually, directly from that proof, because the constructed formula $F(x)$ is in conjunctive normal form:

Corollary. CNF-SAT is NP-complete.

Now that we have an NP-complete problem, we can use the typical method for showing that a decision problem A is NP-complete:

1. Show that $A \in NP$.

2. Show that some decision problem previously proven to be NP-complete can be polynomially transformed to A.



Think: If I had a hypothetical algorithm to solve A, how could I use it to solve some known NP-complete problem?

— Be careful not to get the reasoning backward. You want to polynomially transform a known NP-complete problem to A, not vice versa.

Thm. ILP is NP-complete.

Pf. Polynomial transformation from CNF-SAT to ILP.
(See notes from yesterday.) □
— See also P&S Example 15.8 in §15.3.

Corollary. 0-1 ILP is NP-complete.

Pf. The poly transf from CNF-SAT to ILP actually creates an instance of 0-1 ILP (all domains restricted to $\{0,1\}$). □
— Note: P&S call 0-1 ILP, ZOLP.

Defn. 3-SAT.
Instance: A propositional formula F in conjunctive normal form on the Boolean variables $x_1, \dots, x_n$ such that every clause contains exactly three literals.
Question: Is F satisfiable?

Thm [P&S Thm 15.2]: 3-SAT is NP-complete.

[Note: 2-SAT can be solved in linear time! See P&S Chap. 15 Problem 6.]

Pf. 3-SAT is a special case of CNF-SAT, which in turn is a special case of SAT, which are both in NP, so 3-SAT is also in NP. ✓

Cont'd →

7 July

We show that CNF-SAT polynomially transforms to 3-SAT.

Let F be a CNF formula with clauses $C_1, \ldots, C_m$.

For each clause C_i:

1. If C_i has exactly three literals, leave it alone.
2. If C_i has $k \geq 4$ literals, say $C_i = \lambda_1 \vee \lambda_2 \vee \cdots \vee \lambda_k$, replace C_i with the clauses
$$(\lambda_1 \vee \lambda_2 \vee w_1) \wedge (\overline{w}_1 \vee \lambda_3 \vee w_2) \wedge (\overline{w}_2 \vee \lambda_4 \vee w_3) \wedge \cdots$$
$$\cdots \wedge (\overline{w}_{k-3} \vee \lambda_{k-1} \vee \lambda_k),$$
where $w_1, \ldots, w_{k-3}$ are fresh variables not appearing anywhere else.
   — Use <u>different</u> sets of w_i's for each clause with more than three literals.

Here y, z are variables not appearing in F, but you can reuse y, z for different clauses C_i.

3. If C_i has exactly one literal, say $C_i = \lambda$, replace C_i with the clause $\lambda \vee y \vee z$.
4. If C_i has exactly two literals, say $C_i = \lambda_1 \vee \lambda_2$, replace C_i with the clause $\lambda_1 \vee \lambda_2 \vee y$.

Here α, β are variables not appearing in F.

5. If steps 3 or 4 were used, force y to be false by <u>adding</u> the clauses
$$(\overline{y} \vee \alpha \vee \beta) \wedge (\overline{y} \vee \alpha \vee \overline{\beta}) \wedge (\overline{y} \vee \overline{\alpha} \vee \beta) \wedge (\overline{y} \vee \overline{\alpha} \vee \overline{\beta}).$$
6. Similarly force z to be false if step 3 was used, by adding the clauses
$$(\overline{z} \vee \alpha \vee \beta) \wedge (\overline{z} \vee \alpha \vee \overline{\beta}) \wedge (\overline{z} \vee \overline{\alpha} \vee \beta) \wedge (\overline{z} \vee \overline{\alpha} \vee \overline{\beta}).$$

<u>Exercise</u>: The resulting 3-SAT formula is satisfiable <u>iff</u> F is satisfiable.

— And this transformation can be done in polynomial time. $\square$

Defn. CLIQUE.

Instance: A graph $G=(V,E)$ and an integer k.

Question: Does G contain a clique of size k?

(i.e., a subset $K \subseteq V$ with $|K|=k$ such that every two vertices in K are adjacent)

Thm. [P&S Thm 15.3] CLIQUE is NP-complete.

Pf. We have seen previously that CLIQUE $\in$ NP. Certificate is just K. (See P&S Example 15.5.)

We show that 3-SAT polynomially transforms to CLIQUE. Let F be a 3-SAT formula on the Boolean variables $x_1, \dots, x_n$ having clauses $C_1, \dots, C_m$.

Defn. A partial truth assignment is an assignment of truth values to some of the variables $x_1, \dots, x_n$, leaving the others unspecified.

Notation: We will write, for example: if the variables are $x_1, \dots, x_5$, then the partial truth assignment $x_2 = \text{TRUE}$, $x_5 = \text{FALSE}$ will be written ␣T␣␣F.

(To do: Fix the terrible wording of that sentence.)

Our goal is to construct an instance of CLIQUE, i.e., a graph $G=(V,E)$ and an integer k, whose answer is "yes" iff F is satisfiable.

7 July

[Note: WLOG no clause contains both a variable and its negation— such clauses are tautologies and can just be deleted.]

The vertex set V is constructed as follows: each clause C_i has seven corresponding vertices, labeled with the partial truth assignments that assign truth values to and only to the three variables appearing in C_i, except the one that would make C_i false (by making all of its literals false).

[See Example, next page.]

There is an edge joining every pair of vertices whose labels are compatible partial truth assignments (they don't assign opposite truth values to any variable). This defines E.

Take $k = m$, the number of clauses.

Exercise: $G=(V,E)$ constructed in this way has a clique of size m if and only if F is satisfiable.
(See P&S for full details of this justification.)

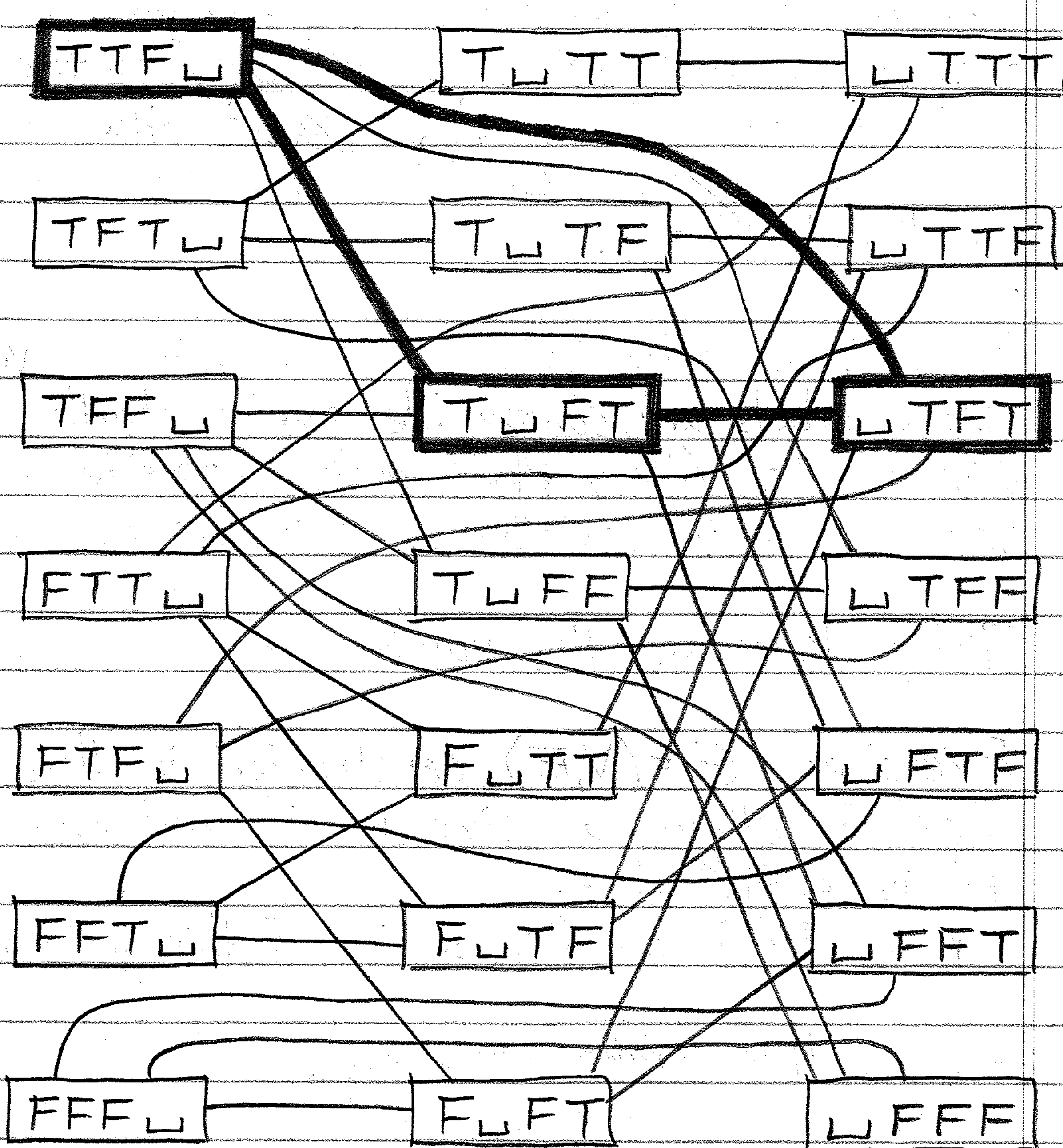
—And this transformation can be done in polynomial time. $\square$

Corollary. INDEPENDENT SET is NP-complete.

Proof. INDEPENDENT SET is in NP, and CLIQUE polynomially transforms to INDEPENDENT SET (see lecture notes from July 2). $\square$

Example: 3-SAT $\xrightarrow{\text{poly transf}}$ CLIQUE.

$$F = (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4)$$



The highlighted clique corresponds to the satisfying assignment x_1 = TRUE, x_2 = TRUE, x_3 = FALSE, x_4 = TRUE.

[See P&S Figure 15–4 for another example.]