We'll find out later what it formally means to be NP-Complete.

WHAT?
WHY?
HOW?

NP-Complete problems are a class of problems whose individual complexities are related to the whole class. More specifically, if $\exists$ a poly. time algorithm for any of these problems, all problems in NP would be poly. time solvable.

NP-COMPLETE

A good place to begin is with an NP-Complete problem called the satisfiability problem.

To approach this problem, we must first understand Boolean formulas. A BF is simply an expression composed of variables that are either true or false. These variables are called literals. A clause is any number of literals connected by a union operation.

For example:

$$x \vee y \vee \bar{z}$$

Boolean Expression

literals

THE SAT PROBLEM

- All this means is that the following Boolean formula evaluates to 1 or is <u>true</u>:

$$\phi = (\bar{x} \wedge y) \vee (x \wedge \bar{z})$$

- This tests whether or not a given formula is <u>satisfiable</u>.

COOK-LEVIN THEOREM

$$\text{SAT} \in P \text{ } \underline{\text{iff}} \text{ } P = NP$$

- Remember from Chapter 5 that when a problem A reduces to problem B, a solution to B can be used to solve A. However, now we can define reducibility in tandem with efficiency of computation.

  ↳ When a problem A efficiently reduces to problem B, an <u>efficient</u> solution of B can be used to solve A efficiently.

POLY-TIME REDUCIBILITY

First, we need some background:

① Polynomial time computable function

Formally:

$f: \Sigma^* \to \Sigma^*$ s.t. $\exists$ a TM M that halts with just $f(w)$ on its tape when started on any input w.

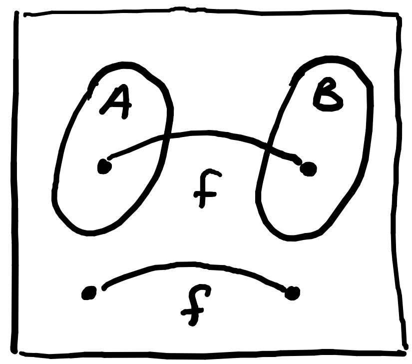
THE
DEFINITION:

Polynomial time mapping reducible (or polynomial time reducible) language A to another language B.

Formally:

If $\exists$ a polynomial time computable function $f: \Sigma^* \to \Sigma^*$, where $\forall w$:

$$w \in A \iff f(w) \in B$$



f is called the polynomial time reduction of A to B.

This is written $A \leq_p B$.

This leads to a new theorem:

- If $A \leq_p B$ and $B \in P$, then $A \in P$.

So what does a polynomial time reduction provide us? As before, a reduction of A to B provides a way to convert membership testing in A to membership testing in B, but now its done more efficiently.

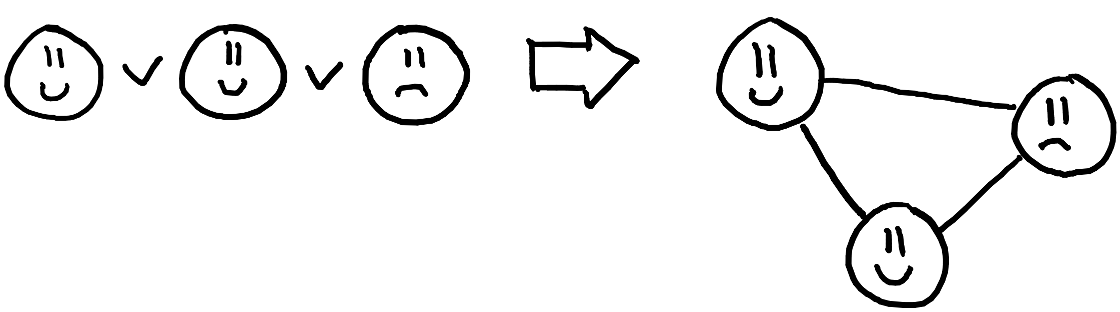
THE 3-SAT PROBLEM

- This is just a derivative of the SAT Problem where each clause contains only <u>3</u> literals.

It turns out that:

$$\text{3-SAT} \leq_p \text{CLIQUE}.$$

This means that we can convert formulas to graphs.



Now let's define what it formally means to be NP-Complete!

THE DEFINITION:

- A language B is NP-Complete if
  1. $B \in NP$
  2. $\forall A \in NP,\ A \leq_p B$
- You should also know these two theorems.

- If B is NP-Complete and $B \geq_p C$ for $C \in NP$, then C is NP-Complete.

- If B is NP-Complete and $B \in P$, then $P = NP$.

- Once we have one NP-Complete problem, we can identify others by poly. time reduction.

  We start by establishing <u>two</u> NP-Complete problems:

  1. SAT is NP-Complete
  2. 3-SAT is NP-Complete

- In the next section, you'll see how other NP-Complete problems are obtained from these two problems.