If $P = NP$ then ...

1 Known Theorems and Definitions

Notation 1.1 P is the set of problems that are in polynomial time. Just think *can be solved quickly*.

Note that if $P = NP$ that means that one can determine quickly if a formula has a satisfying assignment. Can one also *find* a satisfying assignment if one exists? Yes:

Lemma 1.2 *If $P = NP$ then there exists a poly-time algorithm that will, on input ϕ, do the following.*

1. *If $\phi \notin \mathrm{SAT}$ then the output is NO*
2. *If $\phi \in \mathrm{SAT}$ then the output is $\vec{a}$ where $\phi(\vec{a}) = T$ (so the output is a satisfying assignment).*

Note that SAT is a $\exists$ question: Does THERE EXIST a satisfying assignment? But what about a $\exists\forall$ question? If $P = NP$ then are those also easy? Yes:

Lemma 1.3 *Assume $P = NP$ then the following are true.*

1. *Let B be a set of pairs that is in P. (Think given x, y, determining $(x, y) \in B$ can be done quickly). Let q be a polynomial. Then the following problem is in P*

$$A = \{x \colon (\exists y, |y| \le q(|x|))[(x, y) \in B\}.$$

Example Let

$$B = \{(\phi, \vec{y}) \colon \phi(\vec{y}) = T\}.$$

Then

$$A = \{\phi \colon (\exists y, |y| \le q(|x|))[(\phi, y) \in B\}.$$

Note that A is SAT.

Non-SAT Example G is a set of cities and a table that tells you how much it costs to go from one to the other. c is a cost so just a natural number. y is a sequence of cities so that you hit every one once.

$$B = \{(G, c), y)) \colon \textit{The sequence } y \textit{ costs} \le c \ \}.$$

Then

$$A = \{(G, c) \colon (\exists y)[\textit{The sequence } y \textit{ costs} \le c \ \}.$$

HENCEFORTH $(\exists^p x)$ and $(\forall^p x)$ WILL MEAN THAT THE DOMAIN OF x IS STRINGS BOUNDED BY SOME POLY IN THE LENGTH OF THE PREVIOUS VARIABLES.

2. *Let B be a set of triples that are in P (just think given x, y, z, determining $(x, y, z) \in B$ can be done quickly). Let q be a polynomial. Then the following problem is in P*

$$A = \{x \colon (\exists^p y)(\forall^p z)[(x, y, z) \in B].\}$$

Example Let $\phi(\vec{x}, \vec{y})$ be a formula with variables in $\vec{x}$ and $\vec{y}$. So its really $\phi(x_1, \ldots, x_n, y_1, \ldots, y_m)$.

$$B = \{(\phi, \vec{x}, \vec{y}) \colon \phi(\vec{x}, \phi y) = T\}.$$

Then

$$A = \{\phi(\vec{x}, \vec{y}) \colon (\exists \vec{x})(\forall \vec{y})[\phi(\vec{x}, \vec{y})].$$

Note that A is not SAT, its a $\exists\forall$ version of SAT.

3. *Let B be a set of four-tuples (or five-tuples etc.) that are in P. Similar to last part.*

Say we have that SAT is in Poly Time but perhaps with a large polynomial. Can we ASK if there is a better program? Yes, though in a limited domain:

Lemma 1.4 *If $P = NP$ then there exists a poly-time algorithm that will, on input program M, a poly q, and a number n will do the following.*

1. *If M is an algorithm for* SAT *restricted to $\leq n$ variables such that on any input on $k \leq n$ variables runs in time $\leq q(k)$ then output YES. (So if M is a FAST algorithm for* SAT *restricted to $\leq n$ variables then output YES.)*

2. *Otherwise output NO*

Proof:

For this problme we take the number of variables to be the size of a formula .

Let A be the set of all (M, q, n) such that the following are true

1. $(\forall \phi, |\phi| = m \leq n)[M(\phi)$ runs in time $\leq q(m)]$.

2. $(\forall \phi, |\phi| = m \leq n)[M(\phi) = \vec{a} \rightarrow \phi(\vec{a}) = T]$.

   If $M(\phi)$ outputs a vector, its a satisfying assignment.

3. $(\forall \phi, |\phi| = m \leq n)[M(\phi) = NO \rightarrow (\forall \vec{a})[\phi(\vec{a}) = F]$.

   If $M(\phi)$ outputs NO then ϕ is NOT satisfiable.

This can be written with quantifiers and fit into the form of Lemma 1.3. Hence the problem is in P. ∎

Can we actually FIND a better algorithm? Yes.

Lemma 1.5 *If $P = NP$ then there exists a poly-time algorithm that will, on input a poly q, and a number n will do the following: Determine if there exists an Algorithm M as in the last lemma, and if so then OUTPUT THE ALGORITHM.*

Theorem 1.6 *Assume $P = NP$ (though perhaps with a terrible algorithm). Assume there exists a better algorithm that works when the number of variables is $\leq 10^{10}$. Then we can find that algorithm.*

Proof:

Run the algorithm in Lemma 1.4 on smaller and smaller polynomials (and $n = 10^{10}$) until you find a small polynomial (small enough for your purposes) where it says YES. ∎

Note the following

1. Since you may have $P = NP$ but with a terrible algorithm, finding the better algorithm will take a long time. But its a one-time cost.

2. The inventor of the terrible $P = NP$ algorithm probably understood the algorithm, as did others who looked at it. But the new much-better algorithm was machine generated and hence it is possible, indeed likely, that nobody understands it.

3. I am assuming that there exists a good algorithm. If this is incorrect, thats sad, but the approach above will verify that there is no good algorithm.