If P=NP Then There Is a Program For SAT

William Gasarch
What if P=NP but the proof is nonconstructive so that we don’t have an algorithm?

1. **Good News:** If P=NP then ∃ program you can write that will decide a finite variant of SAT in P-time.
2. **Good News:** You can write the program NOW!
3. **Good News?:** The finite number of fml the program is WRONG on are NOT in SAT.
4. **Bad News:** The program is completely impractical.
5. **Factoring:** Similar except that the program is always right.
6. **Credit:** The result on SAT is attributed to Levin.
7. **Credit:** The result on factoring I did this morning though I am sure its known.
1) If $P=NP$ then there is a poly-time program for $F$:
   \begin{align*}
   \text{Input:} & \phi \\
   \text{Output:} & \begin{cases} 
   \text{NO} & \text{if } \phi \notin \text{SAT} \\
   \text{Lex Least } \vec{x} \text{ such that } \phi(\vec{x}) = T & \text{if } \phi \in \text{SAT}
   \end{cases}
   \end{align*}

2) Let $M_1, M_2, \ldots$ be a list of TMs with clocks so that $M_i$ on input of length $n$ runs for $i + n^i$ steps. If it hasn’t finished then output BLAH.

   If $F$ (or any function) is computable in poly time then there is some $i$ such that $M_i$ computes $F$.

3) In the next slide we denote $\lceil \lg \lg i \rceil$ by $LL(i)$. 
Assume $P = NP$

The program:

Input($\phi$)
For $i = 1$ to $LL(n)$
  Run $M_i$ on all fml of length $\leq LL(n)$.
  Compute $F$ (by brute force) on all fmls of length $\leq LL(n)$
  If $M_i$ and $F$ agree on all fmls of length $\leq LL(n)$ then
    Compute $x = M_i(\phi)$
    If $z = NO$ then output NO and halt.
    If $z = \overline{x} \land \phi(\overline{x}) = T$ then output $\overline{x}$ and halt.
    If none of those happen then goto the next $i$.

If got this far then none of the $i$ work. Oh well. Just determine $F(\phi)$ by brute force.

Why the program is in $P$ and works on next slide.
Why the Program Works and is in P

Assume $F \in P$. Let $i_0$ be the least $i$ such that $M_i$ computes $F$.

There exists $n_o$ such that for all $n \geq n_o$, for all $\phi$, $|\phi| = n$, when the program is run on $\phi$ the loop will stop during $i = i_0$ and be correct. Hence the program runs in poly-time.

For all $\phi$, $|\phi| \geq n$, the program is correct.
Whenever the program outputs $\vec{x}$, $\phi(\vec{x}) = T$.
Hence when the program is incorrect, $\phi \not\in SAT$. 
1) If FACTORING in in P then there is a poly-time program for $F$:
   Input: $n$
   Output:
   - NO if $n$ is prime
   - some $m$ where $m$ is a nontrivial factor of $n$ if $n$ is not prime

Note: $F$ is not a function, its a multi-function.

2) Let $M_1, M_2, \ldots$ as before.

3) In the next slide we denote $\lceil \lg \lg \lg i \rceil$ by $LLL(i)$. 
Assume FACTORING in P

The program:

Input($n$)

1. Test $n \in PRIME$ using known primes algorithm. If YES then output NO and halt.

For $i = 1$ to $LLL(n)$

Run $M_i$ on all nums of length $\leq LLL(n)$.

Compute $F$ (by brute force) on all nums of length $\leq LLL(n)$

If $M_i$ and $F$ agree on all nums of length $\leq LLL(n)$ then

Compute $x = M_i(n)$

If $x$ divides $n$ then output NO and halt.

If not then goto the next $i$.

If got this far then none of the $i$ work. Oh well. Just determine a factor of $n$ by brute force.

Why the program is in P and works on next slide.
Assume $P = NP$ so $F \in P$ via $M_i$

Assume $F \in P$. Let $i_o$ be the least $i$ such that $M_i$ computes $F$.

There exists $n_o$ such that for all $n \geq n_o$, for all $\phi$, $|\phi| = n$, when the program is run on $\phi$ the loop will stop during $i = i_o$ and be correct. Hence the program runs in poly-time.

If $n$ is prime the program is correct.
If $n$ is not prime then the program only outputs an $m$ that is a nontrivial divisor or $n$.
So the program is never wrong!

Still wouldn’t use it.