If P=NP Then There Is a Program For SAT

William Gasarch

Fear

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

1. Good News: If P=NP then $\exists$ 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.

Setting up The Program for SAT

1) If P=NP then there is a poly-time program for F:

Input: ϕ

Output:

$$\begin{array}{rl} \text{NO} & \text{if } \phi \notin \mathrm{SAT} \\ \text{Lex Least } \vec{x} \text{ such that } \phi(\vec{x}) = T & \text{if } \phi \in \mathrm{SAT} \end{array}$$

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 computers F.

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

Assume P=NP

The program:

Input(ϕ)

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 $z = \vec{x} \wedge \phi(\vec{x}) = T$ then output $\vec{x}$ 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_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 ϕ, $|\phi| = n$, when the program is run on ϕ the loop will stop during $i = i_o$ and be correct. Hence the program runs in poly-time.

For all ϕ, $|\phi| \geq n$, the program is correct.
Whenever the program outputs $\vec{x}$, $\phi(\vec{x}) = T$.
Hence when the program is incorrect, $\phi \notin \mathrm{SAT}$.

Setting up The Program for FACTORING

1) If FACTORING in in P then there is a poly-time program for F:
Input: n
Output:

$$\begin{array}{ll} \text{NO} & \text{if } n \text{ is prime} \\ \text{some } m \text{ where } m \text{ is a nontrivial factor of } n & \text{if } n \text{ is not prime} \end{array}$$

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 numbs of length $\leq LLL(n)$.
   Compute F (by brute force) on all numbs of length $\leq LLL(n)$
   If M_i and F agree on all numbs 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 ϕ, $|\phi| = n$, when the program is run on ϕ 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.