More about $P \neq NP$

The first issue to clarify is: what would qualify as a solution to this problem? After all, proofs in ordinary English can be constructed which are easily seen not to be valid, but it isn't always easy to explain why. The statement of Richard's paradox can be given an English proof, as can its negation.

Here is an example of an obviously wrong proof of a fact, the fact that an even number added to an odd number results in an answer which is an odd number. If we were to say, "assume the answer is even, now Richard's theorem is true, but we know it false, the contradiction proves the sum is odd," we'd see immediately that this is a meaningless proof.

Now, we set up the $P \neq NP$ problem like this. We set up a formal language and theory, with a two variable function symbol S, such that $S(m, n)$ encodes the statement that the m'th Turing machine (or Javascript function) when given the input $S(m, n)$, fails to return a proof of the same sentence $S(m, n)$ within n steps. I have explained elsewhere how it is possible to do this, in the way Godel formalized the Richard paradox in first order language.

Now, suppose we adjoin all $S(m, n)$ as axioms of the theory, at least for all m and n less than some large finite bound, and now fix algorithms $m_1, m_2, ...$ which ignore their input and m_i simply state $S(1, i)$ is an axiom, and $S(2, i)$ is an axiom, and $S(3, i)$ is an axiom....

It is quite possible that some of the m_i produce a proof of $S(m_i, n)$ within n steps for some n.

How did *this* paradox arise? It arises because we have made no attempt to assert that things provable in the theory should be true in real life. There is no type of axiom saying "provable implies true." As we see that $S(m_i, n)$ can be false while it is taken as an axiom of our theory.

We need to ensure that we do not create a paradox in the real world, and then to deduce what we're trying to prove only from the paradox.

Now, suppose it were true that for some natural number a, algorithm a were a polynomial time proof-finder, meaning, as we've described it before, that there is a number c such that whenever a statement w of length s has a proof of length k then a when given input w, returns a possibly different proof within $(k+s)^c$ steps.

Earlier we wished to include enough axioms in our theory so that for each m and n there is a short proof of $S(m,n)$. We should not be allowed to use a proof which is paradoxical in real life, even if it is a valid formal proof. The sense of the evident real life proof is that if $S(m,n)$ were false, it would have a proof which would be found in n steps by an algorithm, namely the algorithm m, and the absurdity of a false statement having a proof implies it is true instead. But if we use this real-life proof as justification for adjoining the axiom $\forall m \forall n S(m,n)$, then there can exist choices of m which can prove $S(m,n)$ within n steps, and the meaning of $S(m,n)$ has changed in real life, to a false statement. That is, the truth or falsehood of $S(m,n)$ within the formal langauge depends on our theory and its axiomatization, and although the $S(m,n)$ are evidently true in real life, once we adjoin them as axioms of the theory, they become provable quickly and hence false in real life. So that it is not that they are 'true in real life' to start with, but they are 'true in real life' if the axiomatization we choose has 'real life' as a model.

So that we are in a situation where adjoining as axioms to the formal theory things which are true in real life, makes them become false in real life even while provable algorithmically in the theory.

That is, adjoining to the theory axioms which are true in life, can make them become quickly provable formally and hence false in real life.

We might be very happy if $S(m,n)$ could be provable at all in the theory, algorithmically, say, but by a different algorithm than m, and by a proof whose length $f(n)$ is as we should expect logarithmic in n. This contradicts m being a polynomial time theorem prover for if m could prove $S(m,n)$ within $(length(S(m,n)+f(n))^c$ steps, we would have $(lengthS(m,n)+f(n))^c > n$ a contradiction for n large.

The difficulty was, we could not use m as the algorithm which is going to provide a proof of $S(m,n)$, because as we saw, a paradox arises.

Instead of using the $S(a,n)$ as axioms, we could try using the $T(a,b,n)$ in an attempt to avoid the diagonal.

For a still being our presumed polynomial time proof-finding algorithm, let's try adjoining as an axiom $\forall w \forall n T(w,a,n)$.

The $T(w,a,n)$ are provable in our theory from quantifier elimination.

If we believe that the real world is a model of our theory, then the $T(w,a,n)$ must all be true, and so it must be true that when any algorithm w is given $T(w,a,n)$ as an input, for any number n, it does not return a proof of $T(w,a,n)$ within n steps unless $w=a$.

We seem to have removed the immediate contradiction in the real world, there is no longer any difficulty with the notion that a finds a polynomial time proof. Note that adding in extra axioms cannot make a fail to find whatever proof it made using the original axioms. Because there is a short algorithmic proof of each $T(w,a,n)$ to within a very small error, a finds a proof within $length(T(w,a,n))^c$ steps. In fact it is consistent now for a to merely carry out the quantifier elimination.

The issue is now that within this particular theory it is provable that there is no other algorithm besides a which does this.

So let's start over and posit that there are two different polynomial time theorem provers, a and b. Now we go through what we've just done, adjoining the axiom $\forall w \forall n T(w, a, n)$.

And algorithm number b is one of the choices of w. So within our theory, which has the real world as a model, $T(b, a, n)$ is true, and has a proof which, presumably is length $f(n)$ logarithmic in n [this step is currently missing]. Now as before we deduce $n < length(T(b, a, n) + f(n))^c$ and this statement which becomes false for large n is our contradiction which proves that such algorithms as a, b cannot both exist.

The proof is still vague and bordering on paradoxical, or surely it still is paradoxical, but not quite as stupid as the proof using the $S(m, n)$.

Perhaps the situation is that *any* real-life proof is paradoxical, and all we can do is hide the paradox deeper and deeper. With the problem of proving $P \neq NP$ there is no difficulty believing the statement, nor is there any difficulty producing the first proof which comes to mind, which is paradoxical. And the difficulty is just hiding the paradox, a step which must be done in constructing any proof, is more difficult in this case because of the quantification over algorithms.

May 2017