

Re: Oracles

Source: <http://sci.tech-archive.net/Archive/sci.math/2006-08/msg00424.html>

- *From:* "Chip Eastham" <hardmath@xxxxxxxx>
 - *Date:* 9 Aug 2006 09:10:23 -0700
-

matt.zellman@xxxxxxxx wrote:

Why are oracles even noteworthy in discussing computational complexity?

Several things I've read sound like the fact that some oracles make $P=NP$ and others make $P \neq NP$ is a surprising result. I don't see why it should be.

If your solution to an NP-complete problem uses an oracle that is for a problem that is NP-hard, then it will make $P=NP$, otherwise, $P \neq NP$. The very fact that an oracle makes an NP problem solvable in P indicates that the oracle is NP-hard, because you have just reduced an NP problem to it in polynomial time.

How is it that even though $P \neq NP$ has been proved given the existence of certain oracles, it hasn't been recognized that the existence of any oracle is bound to move NP closer to P by reducing the total complexity of a computation?

If $P \neq NP$ given the existence of any oracle, then shouldn't $P \neq NP$, period?

I suspect that you are unclear about the meanings of some of these terms.

An "oracle" was introduced by Turing as a hypothetical adjunct to his deterministic Turing machines. Why is a hypothetical machine useful in complexity theory?

The motivation is to explore whether there are degrees of computationally "hard" and "easy" problems. Access to an "oracle" machine that solves any of a certain class of problems allows us to formulate a "relativization" of one class of problems to another, focusing on what the computational effort of "translation" amounts to between these different classes.

Re: Oracles

The classification of "NP-complete" (for example) gets defined in this way. A problem is said to be in "NP" if a non-deterministically provided answer can be verified in polynomial time (ie. time bounded by a polynomial function of the size of the problem's statement/input). In other words NP is short for "non-deterministic polynomial".

A problem is "NP-complete" if any problem in NP can be relativized/translated to an instance of this problem in polynomial time. Intuitively these problems are as computationally "hard" as any problem in NP could be.

The problems in P are characterized by (deterministic) polynomial time algorithms, ie. finding and verifying an answer in time bounded by a polynomial function in the size of the input. It is known, for example, that there's a polynomial-time deterministic algorithm to decide primality of an integer. Here the size of the input would be the logarithm of the integer, e.g. proportional to the number of bits required to express it.

Clearly P is contained in NP, but it is not known if P is a proper subset of NP. Despite the intuition that this should be true, there are no compelling results to support what is for me the natural expectation.

Consider the problem of factoring an integer. It is in NP because if an "oracle" were to hand us the factors, we could verify in polynomial time their correctness by multiplication (using a "fast" algorithm).

This notion, that modulo an "oracle" which solves an NP-complete problem, all NP problems can be solved (by translation into the former) in polynomial time, doesn't obviate the distinction between P and NP. Theoretically it is possible that $P = NP$ could turn out to be true, but this would be unexpected so far as we know now.

Lot's of thought has gone into trying to construct a problem in NP but not in P, but this must be very difficult to do because we don't have much to show for our efforts. Perhaps it will turn out to be one of those things that is easiest to establish in some non-constructive way, ie. without being able to explicitly exhibit the problem that belongs to NP and not to P.

regards, chip

Re: Oracles