A Proof that P≠NP

By Pascal Hitzler, Kno.e.sis Center, Wright State University, Dayton, Ohio
September 2010

Abstract
We demonstrate the separation of the complexity class NP from its subclass P.

Preliminaries

Preliminary definitions and background can be found in [Sudkamp, 2006], and the following are taken from [Sudkamp, 2006].

[Sudkamp, 2006, Section 8.7]: Every nondeterministic Turing Machine can be simulated by a deterministic Turing Machine. Hence, they give rise to the same notion of computability.

[Sudkamp, 2006, Definition 8.8.1]: A deterministic (k-tape) Turing Machine *enumerates* a language L if all of the following hold.

- The computation begins with all tapes blank.
- With each transition, the tape head on tape 1 (the output tape) remains stationary or moves to the right.
- At any point in the computation, the nonblank portion of tape 1 has the form
  B#u1#u2#...#uk# or B#u1#u2…#uk#v
  where u1,u2,… are in L and v is a string over the tape alphabet.
- A string u will be written on tape 1 preceded and followed by # if, and only if, u is in L.

[Sudkamp, 2006, Theorem 8.8.6]: A language is recursively enumerable if, and only if, it can be enumerated by a deterministic Turing Machine.

The following is easily shown from the above. We include a proof for completeness.

Theorem 1
A language is recursively enumerable if, and only if, it can be enumerated by a nondeterministic Turing Machine.

Proof.
By the results cited above, a language is recursively enumerable if, and only if, it can be enumerated by a deterministic Turing Machine, while deterministic Turing Machines can simulate nondeterministic ones (and vice versa). qed.

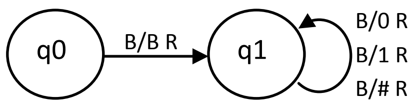
Results

We now proceed to the new results.

Theorem 2
Every set of non-negative integers is recursively enumerable.

Proof.
Let S be an arbitrary set of non-negative integers. Let L be the language containing exactly those strings over {0,1} which are binary representations of a number in S.

Now consider the following (1-tape) nondeterministic Turing Machine M, where q0 is the start state, and B stands for a blank read from the tape.



Obviously, there is a computation of M which produces L (and therefore S). By Theorem 1 we have that L, and therefore S, is recursively enumerable. Since S was chosen arbitrarily, any set of non-negative numbers is recursively enumerable.   qed.

Corollary 1
The set of all subsets of the non-negative integers is countable.

Proof.
Since every Turing Machine can be described by a finite string (or, use Gödel numbering), the set of all Turing Machines is countable. Since every subset of the non-negative integers can be enumerated by a Turing Machine (Theorem 2), the set of all these subsets must be countable.   qed.

Corollary 2
The theoretical foundations of Computer Science are contradictory.

Proof.
Georg Cantor has shown (using a diagonalization argument) that the set of all subsets of the non-negative integers is uncountable, which contradicts Corollary 1.   qed.

Corollary 3
P ≠ NP.

Proof.
Since the theoretical foundations of Computer Science are contradictory, the statement follows immediately.   qed.

References

[Sudkamp, 2006] Sudkamp, T.A. (2006). Languages and Machines. Addison Wesley, 3rd edition.