A PROOF OF P = NP

VICTOR PORTON

Abstract. My proof that P = NP.

1. Introduction

Fix any non-contradictory formal system, containing first-order predicate calculus (such as first-order predicate calculus or ZFC). Note that our formal system can be used to prove correctness of its own proofs (in polynomial time).

In this article I use the word "proof" exclusively either to denote proofs in our formal systems or to denote the proof presented in this article. I do not use it as a synonym of "certificate". (However, certificates used are proofs.)

2. Proof

I will call an NP-*complete verifier* an algorithm that verifies an NP-complete problem in polynomial time.

Obviously, if P = NP, then there exists some NP-complete verifier.

Let $R(X)$ be the property, whether an arbitrary algorithm X (that takes any input data Y) produces a proof (in our formal system) of the statement (for every algorithm Y)

$$X(Y) = Z \Rightarrow \exists \text{algorithm } X' : X'(Z) = Y. \tag{1}$$

I remind: X is in NP means that (for every Y)

$$X(Y) = Z \Rightarrow \exists \text{polynomial-time algorithm } X' : X'(Z) = Y. \tag{2}$$

Let for either $R(X)$ or $\neg R(X)$ we have ϕ transforms X into a proof of the theorem (1) or of its negation.

Proposition 1. *If X is in* NP*, then $R(\phi X)$.*

Proof. If X is in NP, then (2), therefore (1), therefore $R(\phi X)$. □

In the usual definition Z is taken to be one bit, but we could instead allow Z to be any polynomial amount of data, without changing concepts of R and of NP-complete.

Lemma 1. *$R \circ \phi$ is an* NP*-complete problem for all algorithms X such that either $R(X)$ or $\neg R(X)$ is provable.*

Proof. ϕ preserves all information about X; R is NP-complete because it subsumes finding proofs by taking Y to be a statement to be proved and X being a proof-finding algorithm (as the last step of $(\phi X)(Y)$ is a theorem about the value of $X(Y)$ that is it contains a proof of Y). □

In the standard definition of NP we have the additional condition at the left side of the implication that $Z = \text{true}$. But let us limit further consideration to such problems that either $R(X)$ or $\neg R(X)$ is provable; then we consider $Z \in \{\text{false}, \text{true}\}$.)

Theorem 1. $\mathrm{P} = \mathrm{NP}$.

Proof. $R \circ \phi$ is an NP-complete problem for all algorithms X such that either $R(X)$ or $\neg R(X)$ is provable.

So, there is an algorithm I in NP for this problem.

Let $I(X) = Z$.

Therefore there exists a polynomial-time I' (independent of X) in P such that $I'(Z) = X$. Applying I' to an input data is a problem in $\mathrm{P} \subseteq \mathrm{NP}$. Therefore applying my definition again we get that there is a polynomial time algorithm I'' (independent of X) such that $I''(X) = Z$. I'' is a polynomial-time algorithm for our NP-complete problem. □

The above solution is constructive, for example, by my previous article.

References

[1] Porton, Victor. "It Seems, I Proved P=NP => I Have an NP-complete Algorithm." Reddit. April 6, 2021. Accessed April 07, 2021. https://www.reddit.com/r/algorithms/comments/ml85dv/it_seems_i_proved_pnp_i_have_an_npcomplete/. Forum r/algorithms

[2] Porton, Victor. "If P=NP, then I have an NP-complete verifier (second proof attempt)." Reddit. April 6, 2021. Accessed April 07, 2021. https://www.reddit.com/r/algorithms/comments/ml85dv/it_seems_i_proved_pnp_i_have_an_npcomplete/. Forum r/algorithms

[3] Wu, William. "Topic: NONCONSTRUCTIVE P=NP." Wu :: Forums. September 9, 2002. Accessed April 07, 2021. https://www.ocf.berkeleyhttps://www.ocf.berkeley.edu/~wwu/cgi-bin/yabb/YaBB.cgi?board=riddles_cs;action=display;num=1031609156.

[4] TODO

Email address: porton@narod.ru