NP-complete Problems can be Solved and Verified in Polynomial Time

Preprint · October 2025

1 author:

Mirzakhmet Syzdykov

126 PUBLICATIONS **54** CITATIONS

SEE PROFILE

NP-complete Problems can be Solved and Verified in Polynomial Time

Mirzakhmet Syzdykov
Independent researcher
mirzakhmets@icloud.com

ABSTRACT

In this short paper we give the final resolution of “P versus NP” theorem according to our previous results obtained in the field of automaton implementation for extended regular expressions and generally intersection operator.

Keywords: P versus NP, proof, P = NP.

INTRODUCTION

The statement was given before by Stephen Cook [1] in his seminal work, where there’s a clear definition of oracle with ‘certificate’ and the problem solution itself defined as an automaton.
Prior to this ‘milestone work’, we have worked on narratives of experimental proof and showed that the equivalence function exists [2], this is, however, not a definitive answer with the full composition of the problem according to oracle and problem languages.
Before we have shown that intersection of languages can be simulated on finite automaton with the ‘witness state’ [3-5].

STATEMENT

Let’s define the language proposed by our solution as $L_{solution}(A)$, where A is the alphabet and also let’s define the language recognized by set of oracles as $L_{oracle}(A)$ over the same alphabet.
We are to show that P equals NP and, thus, the problem can be solved and verified in any polynomial time.

PROOF

Let’s assume that P not equals NP and thus, we have:

$$t\{L_{solution}(A)\} \gg t\{L_{oracle}(A)\}.$$

For our proof let’s assume that states in our solution automaton with oracles are final states as they are actually accepting any existing problem solution.
As we know the only condition in order for the arbitrary word w to be a solution is defined as:

$$w \in L_{solution}(A) \wedge w \in L_{oracle}(A).$$

From the above it follows that:

$$w \in [L_{solution}(A) \cap L_{oracle}(A)].$$

Let’s construct the modified state automaton for the intersection of the languages above as it was first stated in [3-5] – this algorithm shows that we have the least number of operations in order to compute by our algorithm:

$$t(w) = f_{poly}[NFA_{Aho-Corasick}\{L_{solution}(A)\}, NFA\{L_{oracle}(A)\}] = |w|^{O(1)}.$$

From the above it follows that the time for solving the co-existence of certificate path and path followed by solution branch in our automaton construction is polynomial, thus the languages can be found on both paths of our branching, since our automaton is finite along the sought path.
This is proved by the fact that automaton for solution language is finite by definition and the automaton for oracle can be as large as the whole set of certificates amortized by the Aho-Corasick tree which can efficiently encode large data and has the complexity of $O(|A|^{k*log(|w|)})$ - it's also bounded by the polynomial.
Since $NFA_{Aho\text{-}Corasick}$ can give the paths for each of the language of arbitrary length, it will give the fixed polynomial number of steps along the path of the oracle word.
For more proof, we will consider 3-SAT or even MAX-SAT problem as it can be converted into our modified non-deterministic automaton with extended operator like intersection according to proposed logic with the size $O(n*k)$, where n is the number of variables and k is the number of constraints – literally for each of the logical clause we construct 'n-length' expression with binary alphabet, replacing each of the operand in this clause with "0" or "1" according to its form which can be negated, thus giving zero value, rather than one, after each of these steps, we get the final expression which is the intersection of each of the expressions corresponding to each of the clause and its definition.
Thus, we can easily simulate by our extended membership problem algorithm [3-5] and show that the solution exists as well as verify it.

Thus it follows that complexity classes are equal on case of any regular language $L_{solution}(A)$ or any SAT problem:

$$P = NP.$$

DISCUSSION

Generally speaking, we have already shown the thesis on whether any non-deterministic finite automaton can be as functional as any deterministic one – since we have a strong proof of the linear complexity of our algorithm on non-deterministic automaton [5], it follows that it's as performing as on deterministic automaton [3, 4].
The above proof could be possible when even simulating automaton, however, we apply the intersection as we have in general the set of certifying states which is polynomial as it was stated in the original theorem [1].
In contrast, our proof could be seen improper if there would be so-called product constructions for the set operators on finite automatons describing any language, whereas our intersection algorithm on these automatons propose the best possible solution within the time frame as we go through the whole simulation on both branches of construction. More generally, our method can be seen as product construction, however, there is the solution with fewer states on the automaton describing solution language.

Thus, in this paper, we have shown that extended finite automaton can easily simulate any of the SAT-language $L_{SAT}(A)$ over alphabet A as well as get the certifying state within any of the member of accepting states.

CONCLUSION

We have shown the proof the equivalence of general complexity classes by our finite automaton construction for intersection operator and the way of its simulation which is to be linear. For better understanding of our methodology, it's recommended to read our algorithm in the reference articles, which is omitted here as it was outlined many times before with the general aim of implementation of

effective solutions on finite automaton with extended operators like intersection, subtraction and complement.
At first glance, we used Aho-Corasick trees for efficient computation, however, on account of SAT problem type we can get the automaton with extensions, which were seen before in our continuous series of articles, which are not limited to the presented in reference list.

ACKNOWLEDGMENTS

We express our sympathy towards effective algorithms and their verification as it goes beyond ‘extended operators’ and, thus, gives a more broader sense towards proof of “P versus NP” theorem.

REFERENCES

1. Cook, S. (2003). The importance of the P versus NP question. *Journal of the ACM (JACM), 50*(1), 27-29.

2. Syzdykov, M., & Kardeis, Y. (2025). ON EXPERIMENTAL PROOF OF "P VERSUS NP" THEOREM. *Journal of Mathematics, Mechanics and Computer Science*, *127*(3). https://doi.org/10.26577/JMMCS202512735

3. Syzdykov, M. (2015). Algorithm to Generate DFA for AND-operator in Regular Expression. *International Journal of Computer Applications*, *124*(8).

4. Syzdykov, M. (2017). Deterministic automata for extended regular expressions. *Open Computer Science*, *7*(1), 24-28.

5. Syzdykov, M. (2022). Extended Regular Expressions in Finite Automata Revisited. *ADVANCED TECHNOLOGIES AND COMPUTER SCIENCE*, (1), 4-7.