Linear Algorithm for Regular Expressions within P vs. NP Question

Mirzakhmet Syzdykov
Independent researcher, Astana, Kazakhstan

ABSTRACT

We present an almost polynomial matching algorithm for regular expressions with extensions like intersection, complement and subtraction which runs in practically applicable time and space, which outperforms any existing methods to the present time and converges to the theoretically known minimal lower bound, thus, making the extended regular expression matching almost practical in the main-stream domain as regular expressions are widely used to the present time and can be used, for instance, in parsing techniques or other well-known problems like matching of high-size input strings in any practical domains to the present time as originally regular expressions were developed to make it possible, it still needs a better approach for extended sub-problem like match with extended operators as intersection, complement and subtraction. Since our exhibition of the equivalence of complexity classes, we have met that they are still not giving correct and coherent solution for problems known as unsolvable or NP-complete by Cook-Levin theorem “P versus NP”, in contrary, in this work we present the proof of nonequivalent relation between polynomial and non-polynomial classes and show that NP-complete problem cannot be approximated or solved on a hypothetical computational device within the proof by contradiction

Keywords: finite automata, regular expression, matching, Cook-Levin theorem, proof.

INTRODUCTION

Since the constructions presented by Ken Thompson (Thompson, 1968), this tendency became a general approach of using non-deterministic finite automata (NFA) for regular expression matching and construction, however, after the practical application of regular expression in programming environment was stand, the number of arising optimizations were implemented, for example, the well-known tagging approach by Ville Laurikari (Laurikari, 2000), which was used in solving the submatch addressing and extraction using compensated, or limited, number of time and space.

In sense, the regular expression matching with extended operators like intersection, subtraction and complement is almost double exponential for deterministic finite automata, which was previously shown by Gelade and Neven (Gelade & Neven, 2012). Samuel Hsieh presented product construction (Hsieh, 2010), however, for DFA and this is also an NP-complete design and solution (Cook, 2003), which was previously proved from the side of computational complexity, thus, by meaning ‘practical’, we suppose that algorithm is terminated in any polynomial, or observable, time of computation and memory space consumption on any existing Turing machine.

There were number of tries to make the problem of extended regular expression matching with operators like, at least, intersection, subtraction and complement practically possible (Sen & Roşu, 2003; Kupferman & Zuhovitzky, 2002), however, the quadratic or even exponential grow of number of steps of matching algorithm make these methods almost ‘unpractical’.

We will present our approach which runs in time $O(k*m*n)$, where k is the number of extended operators in pattern, m is the size of regular expression and n is the size of the input string to be matched, while considering that pattern practically is much less than the matching string, we make it practically applicable for any general suite.

As we have shown in our prior works that are worth to study by presenting the ‘activator’ state based upon the logical events incoming from the type of activator corresponding to regular language operation like intersection $(A \& B)$, complement $(\sim A)$ and subtraction $(A - B)$ – this logically can be

encoded as the number of events during the matching process and gains no grow of complexity of this process.
To make the algorithm fully correct, we are using Ville Laurikari's tagging transitions for each of the incoming activator and corresponding ending activator which verifies the set of obtained tags according to its type like logical intersection, complement and subtraction.
Thus, if the number of tags never exceeds the complexity of *O(m*n)* during matching routine, it gives us the resulting complexity of *O(k*m*n)* in time and consumed memory space as we separate each tag and give the set of 'matched activator states' in order for our method to be correct.
The theorem whether the problem can be solved as dependent polynomial function as it can be verified is known as "P versus NP" and was tried to be proved for the past time (Sipser, 1992), however, it led to the resulting statement till the present time.
The sample model used in our proof resembles to the parallel computing strategy (Kaur et al., 2011) as parallel computations is a way of giving the more improved solution to algorithm or method, however, this is only an analogy to our modeled virtual machine which hypothetically can solve any NP-complete problem.

VIRTUAL MACHINE

Imagine that any query to the oracle in the given Virtual Machine (VM) can be processed in time at least *O(1)* as it will take some operation and time to check and verify the query towards global optimal solution. The oracle itself compares the result with the given one in the same time and totally in a separate thread in time *O(1)* for overall queue of incoming queries.
We then can construct the universal problem solver for any NP-complete problem based upon the number of parallel threads operating on this hypothetical VM, which will run in polynomial time, thus giving the following statement:

$$\lim_{N \to O(NP)} \frac{NP}{N} = P,$$

where in above statement *NP* is an NP-complete problem complexity, *P* is an oracle polynomial verification complexity and N is the number of threads operating on VM.
With the general assumption we can state that if *N* converges to infinity the whole solution process in observable time in above VM still takes polynomial time, thus we can assume the following:

$$\lim_{N \to \infty} \frac{NP}{N} = P.$$

However, we can also assume that the whole complexity with the infinite number of working threads will take the tame of at least *O(N)* for the given NP-complete problem *NP*, thus, to show our proof, we will assume that *P = NP*, then we have the following:

$$\lim_{N \to \infty} \frac{1}{N} = 1 \Rightarrow 0 = 1 \Rightarrow P \neq NP.$$

The above resolution is a contradiction, thus, $P \neq NP$ on VM even if the number of threads is infinitely large to check and verify each of the solution from the problem *NP*.
We have gone through the long way of the complexity theory and its outcomes, however, the presented result and the following outlet demonstrate that more than practice is needed in order to overcome the

existing barriers lying in the proof of *P = NP*, which supposedly can give a way to approach 'the almost unsolvable problem', however, the experiment and its result tell the different answer.
This dilemma can be seen as promising, but hard to present, since solutions often are described as algorithmic steps, which we avoid in order to rise our experiment.

DISCUSSION

As per our work on Berry-Sethi and Antimirov derivative-based algorithms on construction automaton from regular expression, we can conclude that operational layer can be used when syntax processing occurs on abstract syntax trees – this is a good optimization technique.
As per "P versus NP" problem, it can be admitted that there is a limited model for memory and resource consumption, however, the presented above 'virtual machine' is a typical form of abstract to be made possible in our hypothetical experiment, thus proving that in general, or in some other case, **complexity class *P* doesn't equal complexity class *NP*** – and other classes in the time-dependent hierarchy.

CONCLUSION

We have given a practical and almost polynomial algorithm of matching for regular expressions, languages and automata with extended regular expressions like intersection, subtraction and complement which run in almost polynomial time $O(k*m*n)$ – the memory consumption is obviously same, where k is the number of extended operators, m is the size of regular expression and n is the number of input string. Considering that the pattern is relatively small to the input string, we can consider this algorithm almost polynomial and practical.
We also hope that our approach will be mainstreamed in the present regular expressions with negative or positive look-aheads and look-behinds with respect to the logical activator state and its tracing tag.
The term 'activator state', which was introduced before, plays a vital and general role in constructing NFA for extended regular expressions with intersection, complement or subtraction, it's typically can be considered as the starting and ending state in the same construction for extended operators and matching on these states is logically bound according to the state logic like the corresponding logical operators of 'and', 'not' or 'subtraction' operating on regular language sets.
We have also gained the new improved bound over already practically accepted and best-known $O(m^2*n)$ complexity.
We could also show that DFA construction for extended operators and its implementation within NFA could guarantee the linear matching, however, this is bound by the exponential limit and, thus, is unpractical laying in the complexity EXPTIME.
We have shown on our model that polynomial oracle *P* for non-polynomial problem *NP* will be never equal due to the contradiction on universal solver like VM for NP-complete problems – this is due to the fact that when infinite number of threads are performed on our abstract model and classes are equal it will take zero time in order to find the optimal and correct solution which contradicts with the 'at least' time $O(1)$ for polynomial method since it will take at least single step in order to produce the final result, which generally always exist and its minimal dimension can be two like 'true' or 'false', thus classes *P* and *NP* aren't equal, or $P \neq NP$, with this contradiction followed from assumption that these classes can be equal, considering the fact that VM can solve any problem in *NP* as well as in *P*.
From that and on, it follows that classes are not equal, which can be discussed further, for this purpose we have used a hypothetical approach in order to simulate the best scenario, however, even with abstract and fully theoretical approach, it still remains clear that classes will never coincide, which may give a general proof of "P versus NP" theorem since our model is abstract and follows the assumption of the solution of any NP-complete problem, against which we have made a theoretical experiment and devised the exposed result.

Still we stake that the proposed method is a general case, since the given method uses the approximation to the absolute parity using infinite number of threads on abstract and hypothetical VM, at least it can be partial, but the given scenario is a way of abstracting from that ‘partiality’.
As before, we have investigated that the differential comparison between O-complexity of any class of complexity can give us the ‘strict map’ according to the dependency and relation between these complexity classes – this is a general consequence from the modeled experiment on the hypothetical virtual machine.

REFERENCES

Thompson, K. (1968). Programming techniques: Regular expression search algorithm. Communications of the ACM, 11(6), 419-422.

Laurikari, V. (2000, September). NFAs with tagged transitions, their conversion to deterministic automata and application to regular expressions. In Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000 (pp. 181-187). IEEE.

Gelade, W., & Neven, F. (2012). Succinctness of the complement and intersection of regular expressions. ACM Transactions on Computational Logic (TOCL), 13(1), 1-19.

Hsieh, S. C. (2010). Product construction of finite-state machines. In Proc. of the World Congress on Engineering and Computer Science (pp. 141-143).

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

Sen, K., & Roşu, G. (2003). Generating optimal monitors for extended regular expressions. Electronic Notes in Theoretical Computer Science, 89(2), 226-245.

Kupferman, O., & Zuhovitzky, S. (2002, August). An improved algorithm for the membership problem for extended regular expressions. In International Symposium on Mathematical Foundations of Computer Science (pp. 446-458). Berlin, Heidelberg: Springer Berlin Heidelberg.

Sipser, M. (1992, July). The history and status of the P versus NP question. In Proceedings of the twenty-fourth annual ACM symposium on Theory of computing (pp. 603-618).

Kaur, P., Singh, D., Singh, G., & Singh, N. (2011). Analysis, comparison and performance evaluation of BNP scheduling algorithms in parallel processing. International Journal of Information Technology and Knowledge Management, 4(1), 279-284.