Almost Polynomial Matching with Extended Regular Expressions

Mirzakhmet Syzdykov
Independent researcher
mirzakhmets@icloud.com

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.

Keywords: regular expression, matching, intersection, subtraction, complement

INTRODUCTION

Since the constructions presented by Ken Thompson [1], 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 [2], 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 [3]. Samuel Hsieh presented product construction [4], however, for DFA and this is also an NP-complete design and solution [5], 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 [6, 7], 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.

ALGORITHM

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 (~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.

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 [8, 9].
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.

ACKNOWLEDGMENTS

We express gratitude to the prior works and research in order to make the regular expression matching process effective and possible without any arising limitations.

REFERENCES

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

2. 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.

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

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

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

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

7. 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.

8. Owens, S., Reppy, J., & Turon, A. (2009). Regular-expression derivatives re-examined. *Journal of Functional Programming*, *19*(2), 173-190.

9. Caron, P., Champarnaud, J. M., & Mignot, L. (2011, May). Partial derivatives of an extended regular expression. In *International Conference on Language and Automata Theory and Applications* (pp. 179-191). Berlin, Heidelberg: Springer Berlin Heidelberg.