Repetitive Substructures for Efficient Representation of Automata

Michal Šedý*
supervised by doc. Mgr. Lukáš Holík, Ph.D.

Abstract

Nondeterministic finite automata are widely used across almost every field of computer science, such as for the representation of regular expressions, in network intrusion detection systems for monitoring high-speed networks, in abstract regular model checking, program verification, or even in bioinformatics for searching sequences of nucleotides in DNA. To obtain smaller automata, and thus reduce computational resources, state-of-the-art minimization techniques, such as state merging and transition pruning, are used. However, these methods can still leave duplicate substructures in the resulting automata. This work presents a novel approach to automata minimization based on the transformation of an NFA into a nondeterministic pushdown automaton. The transformation identifies and represents similar substructures only once by so-called procedures. By doing so, we can further reduce automata by up to another 60%.

*xsedym02@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Nondeterministic finite automata (NFA), as their name suggests, can nondeterministically transition from one state to multiple states based on the same input. This property allows NFAs to represent languages more compactly than their deterministic counterparts, which can only be in one state at a time. Despite their hard minimization, NFAs find applications in many fields of computer science, such as representing regular expressions, network intrusion detection systems for monitoring high-speed networks [1, 2], abstract regular model checking [3], verifying programs that manipulate strings [4], and decision procedures in the WS1S and WS2S logic [5, 6].

Minimization Techniques

To reduce computational resources when working with NFAs, it is crucial to reduce their size. For this purpose, the state merging [7, 8, 9] and transition pruning [8, 10] techniques are being used. The state merging technique can merge two states if one of them fully covers the logic of the other. On the other hand, transition pruning removes a transition if there already exists a duplicate transition with the same logic. These methods are implemented in the state-of-the-art tool RABIT/Reduce [11].

Repetitive Substructures

Despite the good reduction potential that standard minimization techniques offer, the resulting automata can still contain redundant substructures. These automata often represent regular expressions, such as those used in network intrusion detection systems (NIDSs) for network traffic scanning. They are constructed as the union of regular expressions. Additionally, there are types of automata that cannot be minimized by these methods.

Our Novelty Approach

In our work, we present a novel reduction approach that involves transforming an NFA into a nondeterministic pushdown automaton (NPDA) that utilizes a stack. This approach identifies similar substructures within the automaton and represents them only once using so-called procedures. The stack is then utilized to track the states from which the procedure has been entered and where to return. This transformation can be likened to converting a purely sequential program into one that utilizes functions and a call stack. By applying our approach to the results of standard minimization techniques, we were able to achieve an additional reduction of up to 60% in both the states and the transitions of the automaton.

2. Motivation

The automaton representing a regular expression (`.*new XMLHttpRequest.*file://)|(.*file:// .*new XMLHttpRequest`) from network intrusion detection system Snort [12] is shown in ⌈Figure 1⌉. Besides the epsilon transitions and `.*`, this is the most minimal form that can be achieved by standard minimization techniques. This is caused by the lack of language inclusions as *Request* and *File* substructures are compactly different. As a result, the automaton contains two substructures, each of which has redundant copy, making the NFA representation unefficient.

3. One Procedure No Duplicate

At this point, we identified repetitive substructures for *Request* and *File*, which represent duplicate information. Each of these substructures is represented by a corresponding procedure. This transformation into procedures can be seen in ⌈Figure 2⌉.

Entering the Procedure

To recognize if the procedure for *Request* has been entered from state 1 as the first part of the regular expression or from state 6 as the second part, the symbol 🟠 or 🟢 is pushed onto the stack, respectively.

Changing Procedures

When directly transferring from the *Request* procedure into the *File* procedure, it is necessary to ensure that this transition will be used only once and only after first entering the *Request* from state 1. This is done by testing for the symbol 🟠 at the top of the stack. If the top matches the required symbol, it is replaced with the symbol 🔴. The same approach applies for the transition from the *File* procedure into the *Request* procedure.

Returning From the Procedure

Transitions exiting the *Request* procedure can only be used when the stack contains the corresponding symbol that is popped afterward. For the transition between states 8 and 9, it is the symbol 🟢, indicating that the *Request* procedure represents the last part of the regular expression that started with the *File*.

4. Experiments

We tested our reduction method on parametric and real-world regular expressions used in network filtering. The highest reductions were obviously achieved on larger automata, as there is a greater chance of the existence of similar substructures. Since our tool is designed to follow after standard reduction methods, the percentage reduction is calculated relative to the resulting automata of RABIT/Reduce.

4.1 Parametric Regular Expressions

The set of 3'656 automata, with an average of 207 states and 2'584 transitions, was obtained from four families of parametric regular expressions [13]. The reduction ratio of states and transitions can be seen in ⌈Figure 3⌉. It can be observed that our tool achieved a 50% reduction on average. The graph's x-axis represents the size of RABIT/Reduce results (input of our tool), and the y-axis represents the reduction against its input. The graph is enhanced with temperature coloring and a distribution function for each axis.

4.2 Network Intrusion Detection Systems

To test our reduction algorithm on real-world examples, seven automata were created as unions of sets of regular expressions from seven different families of Snort rules. The results of reduction performed by standalone usage of the RABIT/Reduce (*RAB*) tool and with the additional application of our method based on procedures, with the two most significant results highlighted are shown in ⌈Table 1⌉. The best result obtained by applying our approach to the results of RABIT/Reduce achieved a reduction of 43.5% in states and 60.3% in transitions.

5. Conclusion

In this work, we introduced a novel approach to automata reduction. This reduction transforms NFAs into NPDAs, noting a similarity with transforming a pure sequential program into a program with functions and call stack. Applying our reduction approach to the results of the state-of-the-art reduction tool RABIT/Reduce resulted in reductions of up to 52.5% in states and up to 60.3% in transitions. These results indicate that our approach can potentially play an important role in automata reduction in the future.

References

[1] I. Sourdis and D. Pnevmatikatos. Fast, large-scale string match for a 10gbps FPGA-based network intrusion detection system. In *FPL'03*. Springer, 2003.

[2] M. Ceška, V. Havlena, L. Holík, et al. Deep packet inspection in FPGAs via approximate nondeterministic automata. In *FCCM'19*, 2019.

[3] A. Bouajjani, P. Habermehl, A. Rogalewicz, and T. Vojnar. Abstract regular (tree) model checking. *STTT'12*, Apr 2012.

[4] P. A. Abdulla, M. F. Atig, Y.F. Chen, L. Holík, A. Rezine, P. Rümmer, and J. Stenman. String constraints for verification. In *CAV'14*. Springer, 2014.

[5] C. Fu, Y. Deng, D. Jansen, and L. Zhang. On equivalence checking of nondeterministic finite automata. In *SETTA'17*. Springer, 2017.

[6] T. Fiedor, L. Holík, O. Lengál, and T. Vojnar. Nested antichains for WS1S. *Acta Informatica*, 2019.

[7] A. Aziz, V. Singhal, R. Brayton, and G.M. Swamy. Minimizing interacting finite state machines: a compositional approach to language containment. In *ICCD'94*, 1994.

[8] D. Bustan and O. Grumberg. Simulation based minimization. In D. McAllester, editor, *CADE'20*. Springer.

[9] L. Ilie, G. Navarro, and S. Yu. *On NFA Reductions*. Springer, 2004.

[10] L. Clemente and R. Mayr. Efficient reduction of nondeterministic automata with application to language inclusion testing. *CoRR*, 2017.

[11] P. A. Abdulla, Y.F. Chen, et al. RABIT and Reduce. https://languageinclusion.org.

[12] Snort. Snort - Network Intrusion Detection & Prevention System. https://snort.org.

[13] G. Gange, J. A. Navas, P. J. Stuckey, H. Søndergaard, and P. Schachte. Unbounded model-checking with interpolation for regular language constraints. In *TACAS'13*. Springer, 2013.