

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 a 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, the 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 novelty approach to automata minimization based on the transformation of a NFA into a nondeterministic pushdown automaton. The transformation identifies and represents similar substructures only once by so called procedures. By this, we can further reduce automata by up to another 50%.

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

1. Introduction

Nondeterministic finite automat (NFA), as its name suggests, can nondeterministically traverse from one state into multiple states based on the same input. This allows NFAs to represent the language more compactly than its deterministic counterpart that can be only at one state at the time. Despite the complicated minimization caused by nondeterminism, NFAs are applied in many fields of computer science, such as representing regular expressions, network intrusion detection systems for monitoring high-speed networks [1, 2], in abstract regular model checking [3], in verifying programs that manipulate strings [4] or in 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 the transition pruning removes a transition if there already exists a duplicity transition with the same logic. This minimization approaches are implemented in state-of-the-art tool RABIT/Reduce [11].

There are Repetitive Substructures

Despite the good minimization potential that standard minimization techniques carry, the resulting automata can still contain redundant substructures. This automata often represents a regular expressions. For example an automaton for network traffic scanning containing the union of regular expression from network intrusion detection systems (NIDS). There are also types of automata that cannot be minimized by this methods at all.

Our Novelty Approach

In our work we present a novelty reduction approach, that incorporates a transformation of NFA to a nondeterministic pushdown automaton (NPDA) that uses a stack. The method identifies automaton's substructures, with the similar logic, and represents them only once by so-called procedure. The stack is then used to denote from which state the procedure have been entered and where to return at the end. This transformation can be understood as the conversion of a purely sequential program to program that uses functions and call stack. By applying our approach on results of the standard minimization techniques we were able to achieve an additional reduction of automaton by up to 50% states and transitions.



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 where each has one redundant copy, making the NFA representation inefficient.



3. One Procedure No Duplicate

At this point we identify repetitive substructure for *Request* and *File*, that represents a duplicate information. Each of this substructures is going to be represented by corresponding procedure. This transformation to 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 expression or from state 6 as the second part, the symbol  respective  is pushed onto the stack.

Changing Procedures

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

Returning From the Procedure

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

4. Experiments

We tested our reduction method on parametric and real-world regular expressions from network filtering. The higher reduction was achieved obviously on larger automata as there is a bigger change for existence of similar substructures. Because our tool is conceived as next level that follows after standard reduction

methods, the percentage reduction is referred to the resulting automata of RABIT/Reduce.

4.1 Parametric Regular Expressions

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

4.2 Network Intrusion Detection Systems

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

5. Conclusion

In this work we introduced a novelty approach to automata reduction. The reduction transforms NFA into NPDA using a similarity with transforming sequential program into a program with functions. Applying our reduction approach as on results of state-of-the-art reduction tool RABIT/Reduce resulted in up to 52.5% of states and up to 60.3% of transitions. This result indicates that our approach can play, in the future, an important role in the automata reduction.

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. Češka, V. Havlena, L. Holík, et al. Deep packet inspection in fpgas via approximate non-deterministic 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] Parosh Aziz Abdulla, Mohamed Faouzi Atig, Yu-Fang Chen, Lukáš Holík, Ahmed Rezine, Philipp Rümmer, and Jari Stenman. String constraints for verification. In *CAV'14*. Springer, 2014.
- [5] Chen Fu, Yuxin Deng, David Jansen, and Lijun 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] Doron Bustan and Orna Grumberg. Simulation based minimization. In David McAllester, editor, *CADE'20*. Springer.
- [9] Lucian Ilie, Gonzalo Navarro, and Sheng 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] Parosh Aziz Abdulla, Yu-Fang Chen, et al. RABIT and Reduce. <https://languageinclusion.org>.
- [12] Snort. Snort - Network Intrusion Detection & Prevention System. <https://snort.org>.
- [13] Graeme Gange, Jorge A. Navas, Peter J. Stuckey, Harald Søndergaard, and Peter Schachte. Unbounded model-checking with interpolation for regular language constraints. In *TACAS'13*. Springer, 2013.