Repetitive Substructures for Efficient Representation of Automata

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Michal Šedý

supervised by doc. Mgr. Lukáš Holík, Ph.D.

Excel@FIT 2024

Motivation

In many automata, especially those representing regular expressions, there exist repetitive substructures that cannot be eliminated by the state-of-the-art tool RABIT/Reduce [1]. This automaton is depicted below in Figure 1.

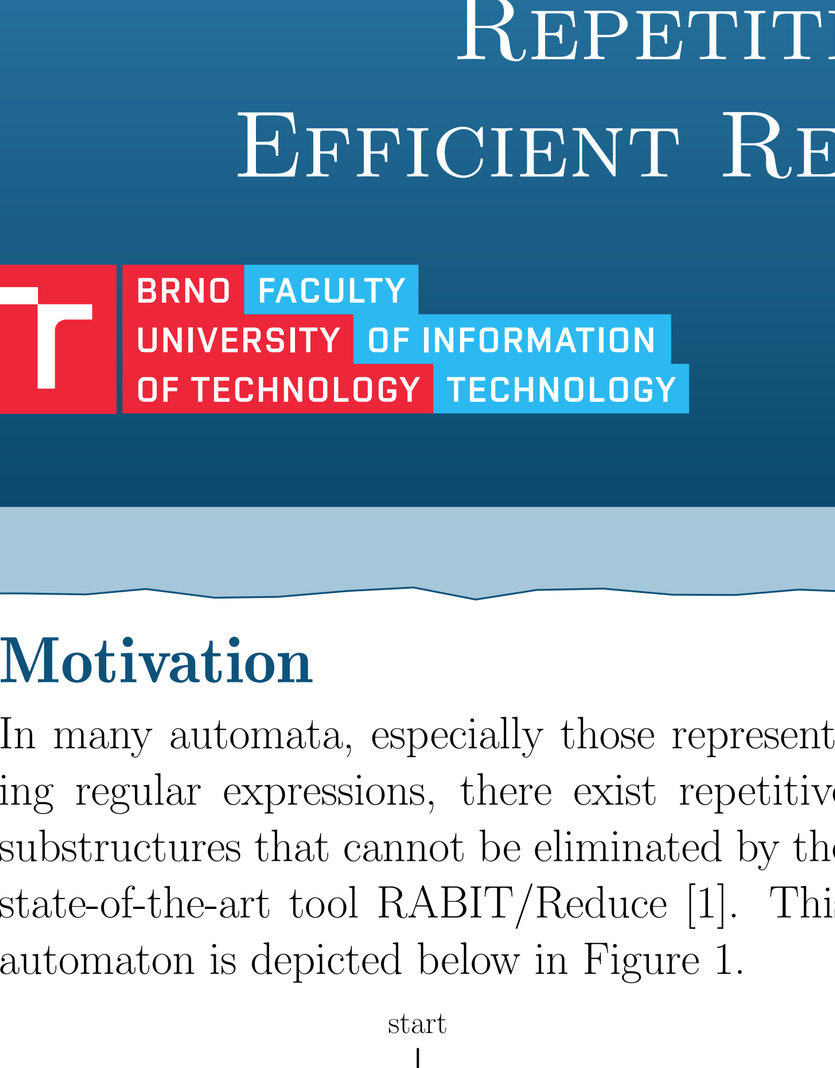

Fig. 1: Network filtering automaton.

We propose a new method based on pushdown automata and the so-called procedures that represent repetitive substructures only once.

Usage of a pushdown automaton and procedures is analogous to the call stack and functions from programming languages.

One Procedure No Duplicates

To represent automata efficiently, without duplicate substructures, we introduce a new concept called procedures. Each set of similar substructures is represented by just one procedure. The automaton uses a stack to determine the original substructure from which the procedure is entered and where to return. The symbol on the stack can also guard transitions specific to some substructures that the procedure represents.

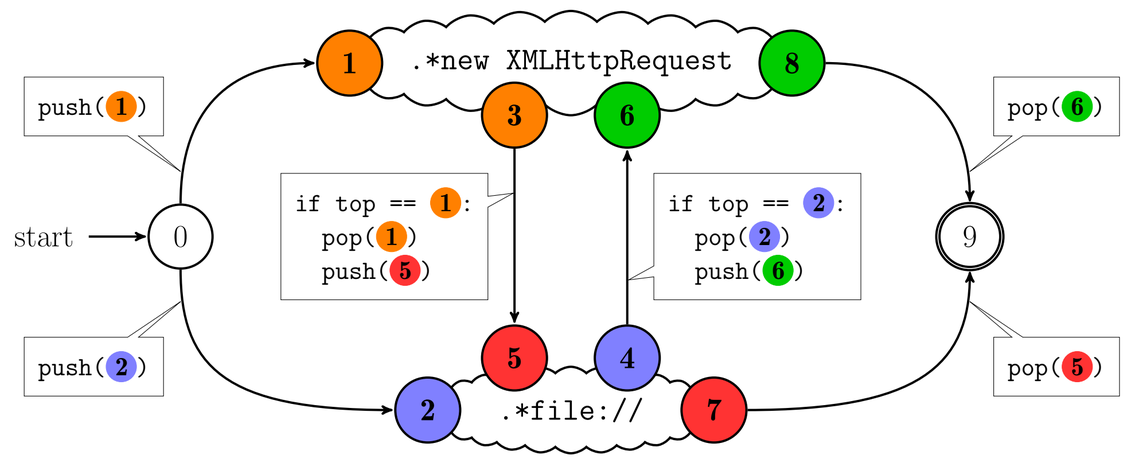

Fig. 2: Reduced network filtering automaton with two procedures.

Parametric Regular Expressions

We evaluated the reduction potential of procedures on 3'656 automata, with an average of 207 states and 2'584 transitions, generated from parametric regular expressions from [2].

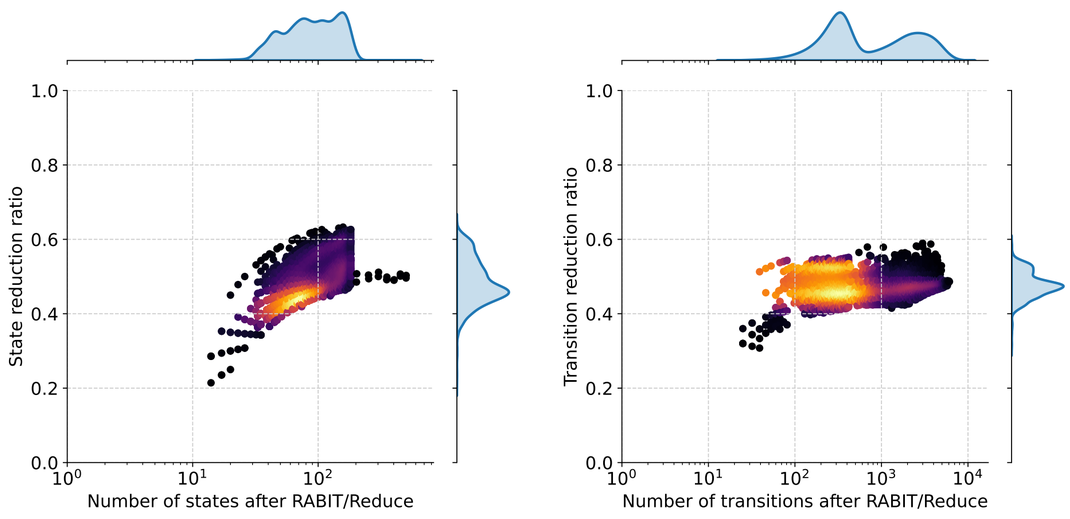

Fig. 3: The reduction ratios achieved by applying procedures to RABIT/Reduce results. On average, procedures improved reductions by 50.3% in states and 47.9% in transitions.

The standalone usage of RABIT/Reduce resulted on average in a reduction of 52.5% states and a reduction of 48.4%. The further reduction performed by our algorithm can be seen in Figure 3. The application of procedures reduced the automata to half the size given by RABIT/Reduce.

Network Intrusion Detection System

To test the reduction capability of procedures in a real-world scenario, we used rules from Snort (a well-known NIDS). We generated seven automata, each representing a union of regular expressions from a single category of Snort rules.

Snort rules	Q_{in}	δ_{in}	Q_{RAB}	δ_{RAB}	$Q_{Proc} + \Gamma_{Proc}$		δ_{Proc}	
p2p	33	1'090	32	1'084	25+6	(96.9%)	570	(52.6%)
worm	50	3'880	34	290	24+8	(94.1%)	284	(97.9%)
shellcode	162	3'328	56	579	48+2	(89.3%)	486	(83.9%)
mysql	235	30'052	91	14'430	45+18	(69.2%)	7'142	(49.5%)
chat	408	23'937	113	1'367	71+25	(76.7%)	1'058	(77.4%)
specific-threats	459	57'292	236	31'935	99+32	(55.5%)	12'680	(39.7%)
telnet	829	7'070	309	2'898	155+82	(50.0%)	2'164	(74.7%)

Tab. 1: Reduction results of RABIT/Reduce (RAB) and procedures (Proc) on seven sets of Snort rules. Q denotes the number of states δ the number of transitions, and Γ the number of stack symbols. The percentages refer to the results of RABIT/Reduce.

The two most significant reductions are highlighted in Table 1. The best reduction was achieved on the automaton created from the `specific-threats` rule set. RABIT/Reduce reduced the automaton by 48.6% of states and by 44.3% of transitions. By further applying procedures, another 43.5% reduction in states and 60.3% reduction in transitions were achieved. This experiment demonstrates that procedures can achieve significant reductions even in real-world examples.

References

[1] Abdulla, P. A. et al. *RABIT and Reduce*. `https://languageinclusion.org`.

[2] Gange, G. et al. Unbounded Model-Checking with Interpolation for Regular Language Constraints. Springer. 2013.