Repetitive Substructures for Efficient Representation of Automata

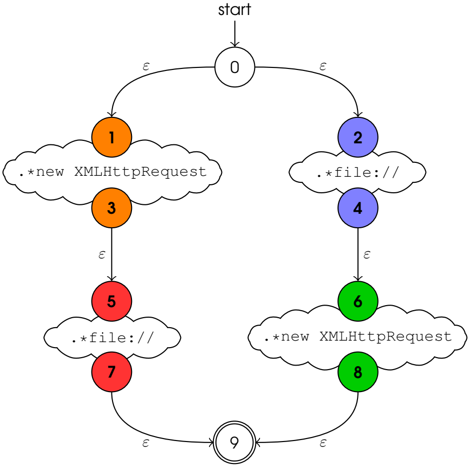
Michal Šedý

Supervisor: doc. Mgr. Lukáš Holík, Ph.D.

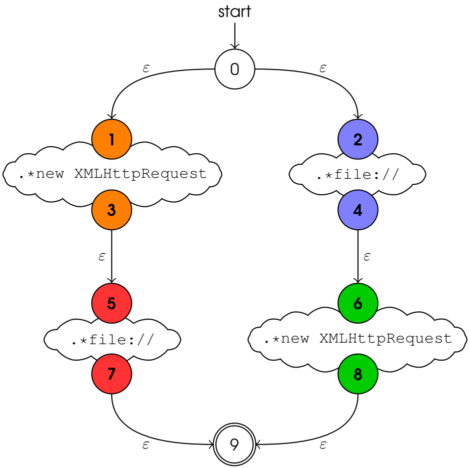
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

September 4, 2024

- Algorithms suffer from state explosion when processing large automata.

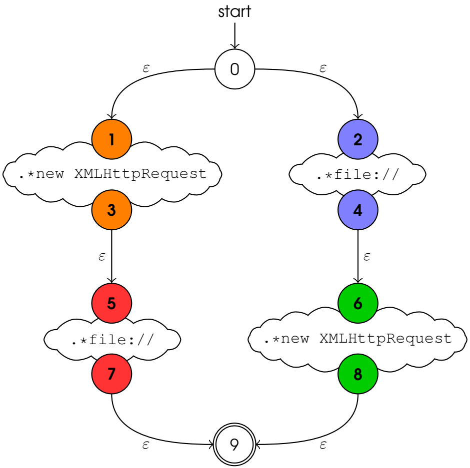


- Algorithms suffer from state explosion when processing large automata.
- State-of-the-art minimization methods (state merging and transition pruning) can leave redundant substructures in the resulting automata.

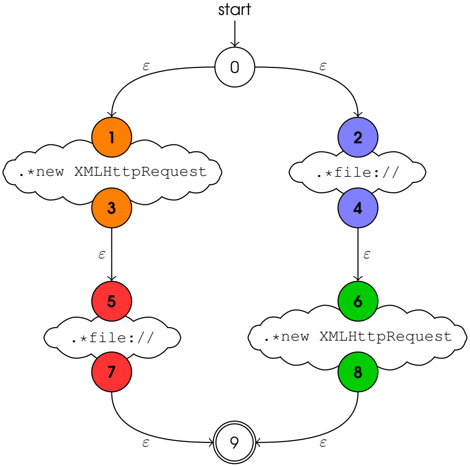


Motivation

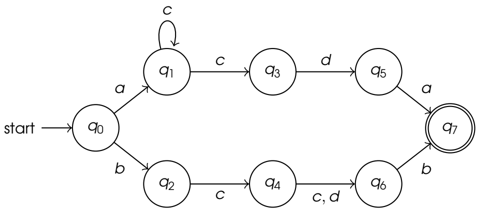
- Algorithms suffer from state explosion when processing large automata.
- State-of-the-art minimization methods (state merging and transition pruning) can leave redundant substructures in the resulting automata.
- Smaller automata means faster and cheaper computations, generally more efficient (can be used within hardware for high-speed network filtering).

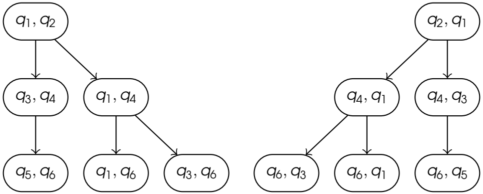


- Algorithms suffer from state explosion when processing large automata.
- State-of-the-art minimization methods (state merging and transition pruning) can leave redundant substructures in the resulting automata.
- Smaller automata means faster and cheaper computations, generally more efficient (can be used within hardware for high-speed network filtering).
- Why not take inspiration from programming languages and use procedures and a stack for repetitive substructures?



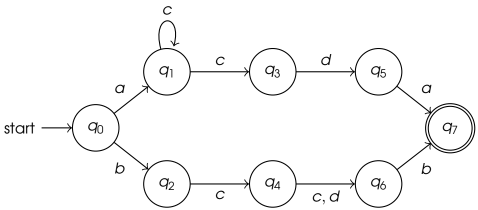
- Utilization of a superproduct.
  - $A = (Q, \Sigma, \delta, q_0, F)$
  - $A' = (Q, \Sigma, \delta, Q, Q)$
  - The superproduct of A is $A' \times A'$

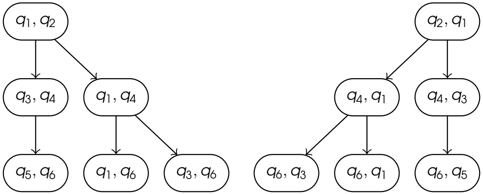




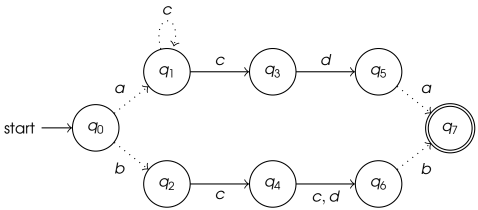
Procedure Finding

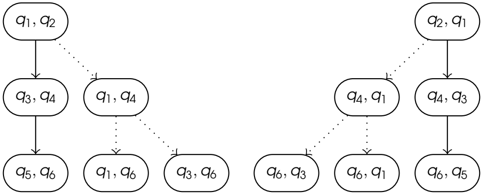
- Utilization of a superproduct.
  - $A = (Q, \Sigma, \delta, q_0, F)$
  - $A' = (Q, \Sigma, \delta, Q, Q)$
  - The superproduct of A is $A' \times A'$
- Each subgraph of the superproduct represents a procedure candidate.
- It is important to choose a subgraph with the highest reduction potential.

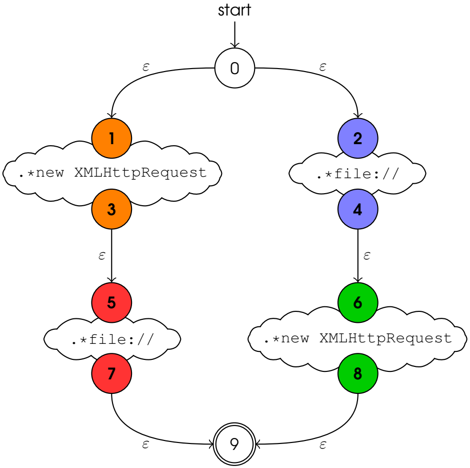




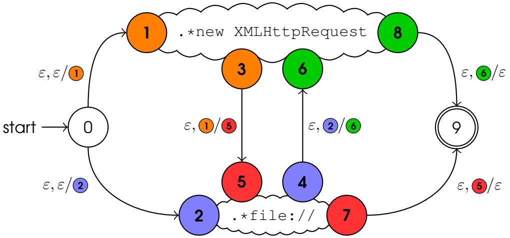
- Utilization of a superproduct.
  - $A = (Q, \Sigma, \delta, q_0, F)$
  - $A' = (Q, \Sigma, \delta, Q, Q)$
  - The superproduct of A is $A' \times A'$
- Each subgraph of the superproduct represents a procedure candidate.
- It is important to choose a subgraph with the highest reduction potential.
  - Give priority to subgraphs with the most redundant transitions.
  - Avoid state repetition.



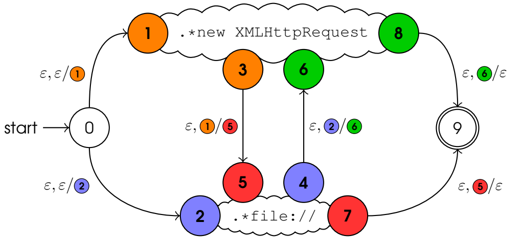




- Each substructure is assigned a unique stack symbol to differentiate its transitions.
- Repetitive substructures are substituted with a single procedure, following the procedure candidate derived from the superproduct.

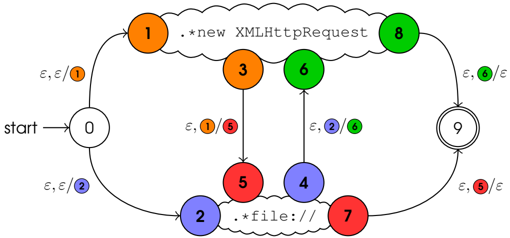


- Unique stack symbols for each procedure are not necessary.
- Only those symbols that meet in the same state must be distinct.

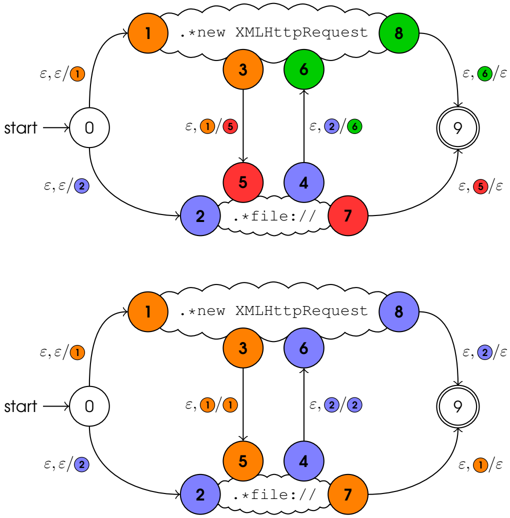


Reduction of Stack Alphabet

- Unique stack symbols for each procedure are not necessary.
- Only those symbols that meet in the same state must be distinct.
  - Meet is an equivalence relation.
  - $a \sim_{meet} b$ iff there exists such a state where a or b can be on the stack.
  - Stack alphabet can be partitioned into equivalence classes according to the meet relation.



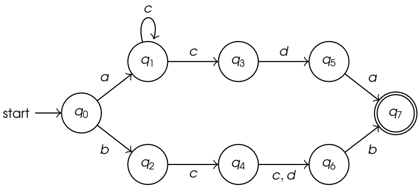
- Unique stack symbols for each procedure are not necessary.
- Only those symbols that meet in the same state must be distinct.
  - Meet is an equivalence relation.
  - $a \sim_{meet} b$ iff there exists such a state where a or b can be on the stack.
  - Stack alphabet can be partitioned into equivalence classes according to the meet relation.
- The minimal number of necessary stack symbols is equal to the size of the greatest equivalence class.



Reduction Metrics

- State reduction is determined by the difference in the number of states and the size of the non-reduced stack alphabet.
- Transition reduction is given solely by the difference in the number of transitions.

- State reduction is determined by the difference in the number of states and the size of the non-reduced stack alphabet.
- Transition reduction is given solely by the difference in the number of transitions.

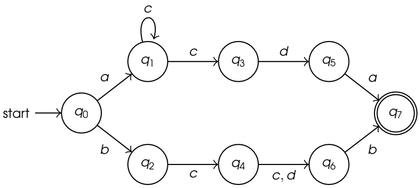


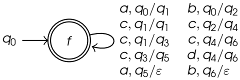
- 8 states
- 0 stack symbols

- 10 transitions

Reduction Metrics

- State reduction is determined by the difference in the number of states and the size of the non-reduced stack alphabet.
- Transition reduction is given solely by the difference in the number of transitions.



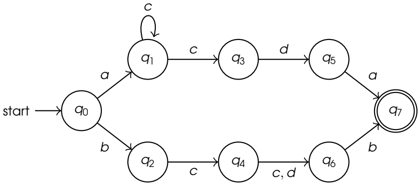


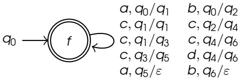
- 8 states
- 0 stack symbols
- 10 transitions

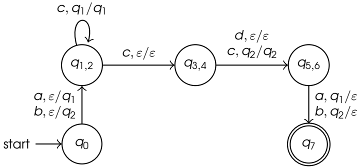
- 1 state
- 7 stack symbols
- 0% reduction in states
- 10 transitions
- 0% reduction in transitions

Reduction Metrics

- State reduction is determined by the difference in the number of states and the size of the non-reduced stack alphabet.
- Transition reduction is given solely by the difference in the number of transitions.







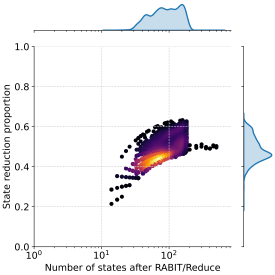
- 8 states
- 0 stack symbols
- 10 transitions

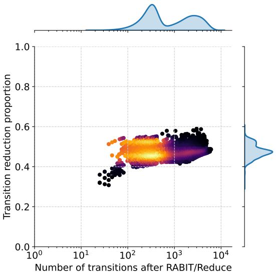
- 1 state
- 7 stack symbols
- 0% reduction in states
- 10 transitions
- 0% reduction in transitions

- 5 states
- 2 stack symbols
- 12.5% reduction in states
- 8 transitions
- 20% reduction in transitions

Parametric Regular Expressions

- Total of 3,656 automata
- Max: 503 states and 6,101 transitions
- Average state reduction: 48.4%
- Average transition reduction: 47.9%



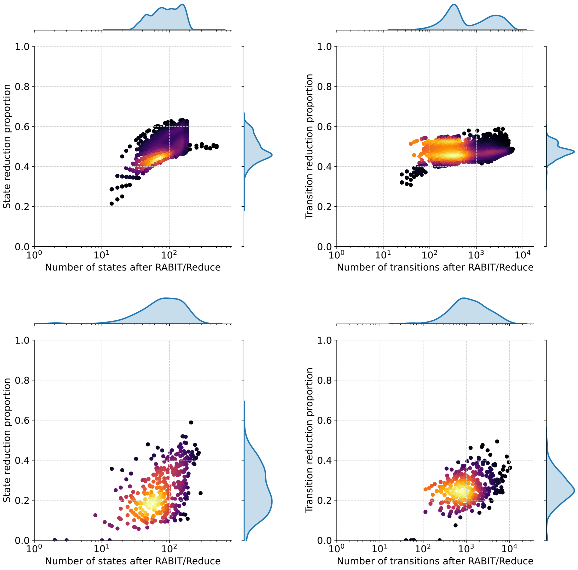


Parametric Regular Expressions

- Total of 3,656 automata
- Max: 503 states and 6,101 transitions
- Average state reduction: 48.4%
- Average transition reduction: 47.9%

Email Validations

- Total of 362 automata
- Max: 289 states and 10,333 transitions
- Average state reduction: 29%
- Average transition reduction: 28.6%



- Total of 3,616 regular expressions from seven families of Snort rules.
- Each automaton represents a union of regular expressions from one family.
- The table illustrates the further minimization achieved by utilizing procedures, following the initial automaton reduction using the Rabit/Reduce tool, which employs state merging and transition pruning.

Snort rules	Q_{in}	δ_{in}	Q_{RAB}	δ_{RAB}	$Q_{Proc} + \Gamma_{Proc}$		δ_{Proc}		Γ^{red}_{Proc}
p2p	33	1,090	32	1,084	25+6	(-3.1%)	570	(-47.4%)	2
worm	50	3,880	34	290	24+8	(-5.9%)	284	(-2.1%)	2
shellcode	162	3,328	56	579	48+2	(-10.7%)	486	(-16.1%)	2
mysql	235	30,052	91	14,430	45+18	(-30.8%)	7,142	(-50.5%)	5
chat	408	23,937	113	1,367	71+25	(-15.0%)	1,058	(-22.6%)	3
specific-threats	459	57,292	236	31,935	99+32	(-44.5%)	12,680	(-60.3%)	6
telnet	829	7,070	309	2,898	155+82	(-23.3%)	2,164	(-25.3%)	4

$Q_{Proc} + \Gamma_{Proc}$: Number of states and stack symbols after procedure mapping.
Γ^{red}_{Proc}: Number of stack symbols after stack alphabet reduction.

Future Work

- Investigate the impact of the stack on the performance of automata operations.
- Incorporate automata with a stack in hardware to scan high-speed networks.
- Improve the detection of similar substructures.
- Effectively utilize a greater stack depth.

Future Work

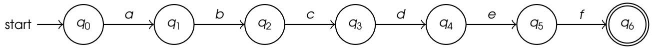
- Investigate the impact of the stack on the performance of automata operations.
- Incorporate automata with a stack in hardware to scan high-speed networks.
- Improve the detection of similar substructures.
- Effectively utilize a greater stack depth.

Thank you for your attention!

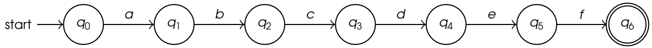
Reduce or Not to Reduce?

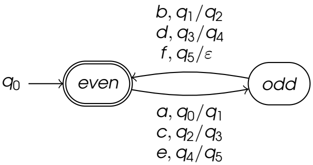
- Why is the size of the stack alphabet reported before reduction?

- Why is the size of the stack alphabet reported before reduction?



- Why is the size of the stack alphabet reported before reduction?

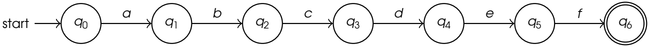


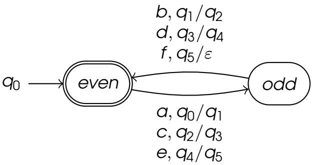


- 2 states
- 6 stack symbols
- no reduction

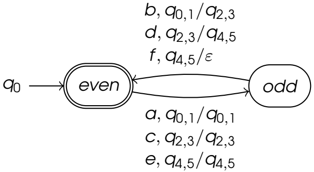
Reduce or Not to Reduce?

- Why is the size of the stack alphabet reported before reduction?





- 2 states
- 6 stack symbols
- no reduction



- 2 states
- 3 stack symbols
- 28.6% reduction (Or is it?)