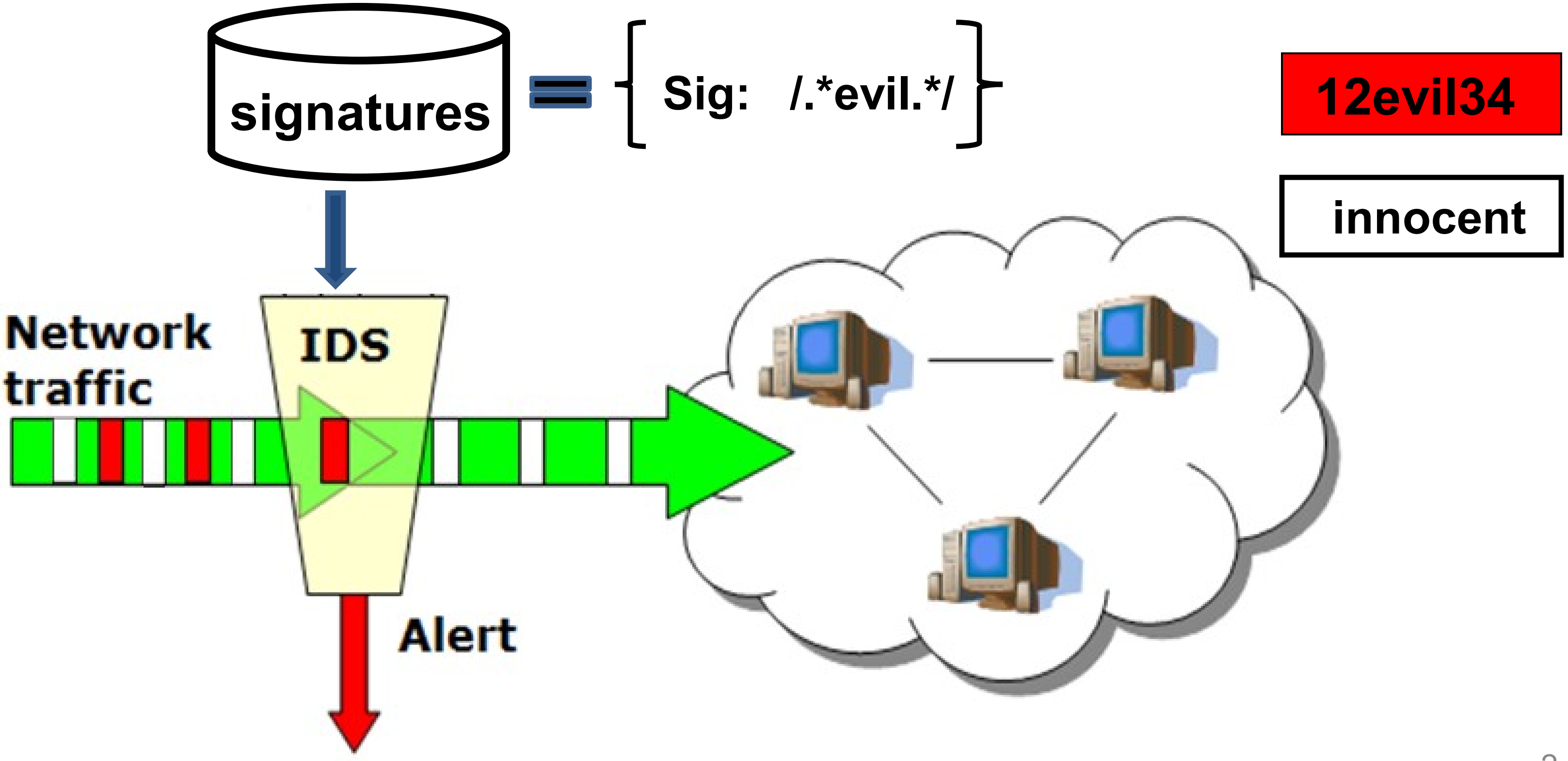
Improving Signature Matching using Binary Decision Diagrams

Liu Yang, Rezwana Karim, Vinod Ganapathy
Rutgers University

Randy Smith
Sandia National Labs

Signature matching in IDS

- Find instances of network packets that match attack signatures



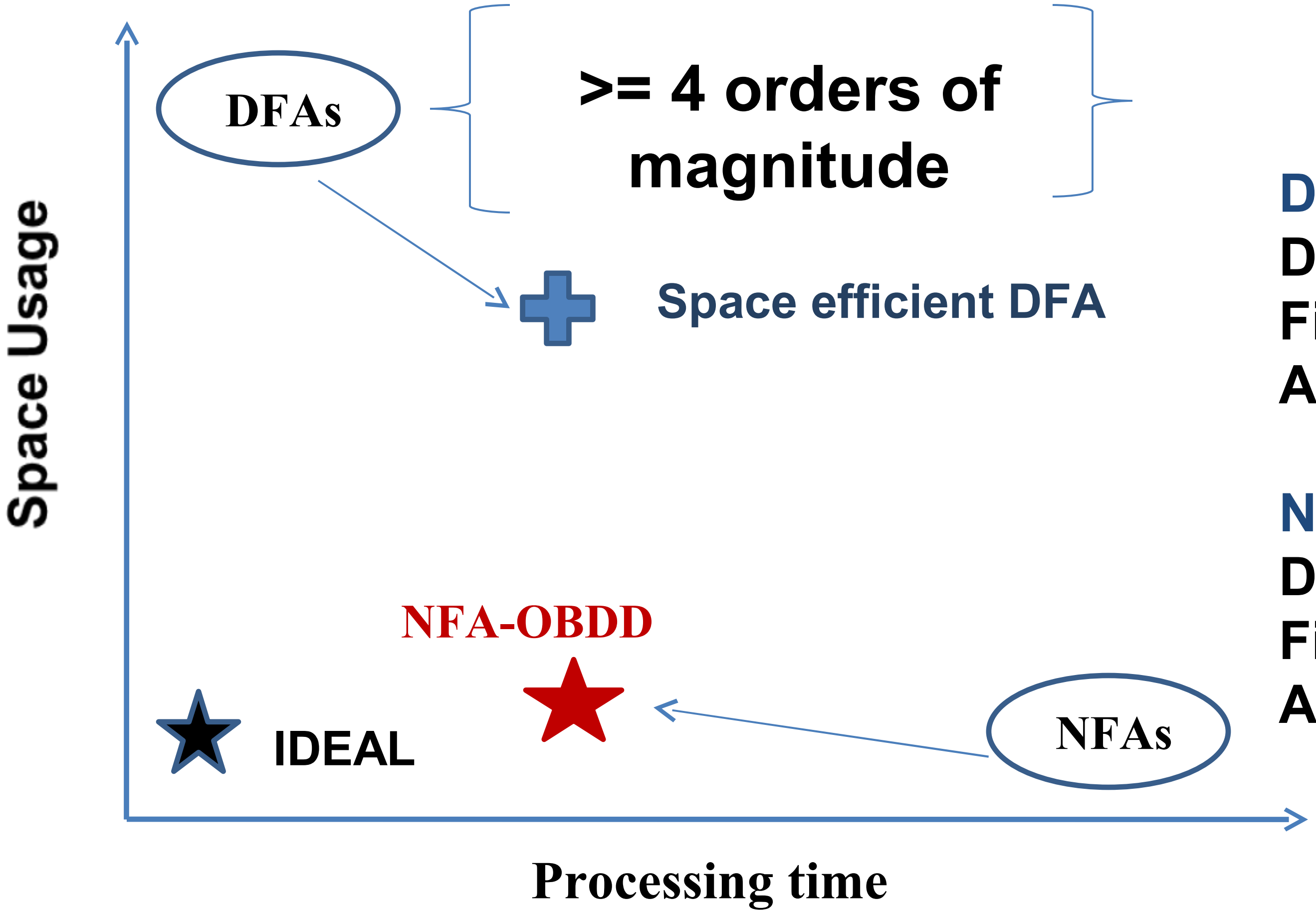
Matching regular expressions

- Signatures represented as regular expressions (RE)
- Finite Automata
  - Represent and operate regular expressions

Time-Space Tradeoff in Finite Automata

- Time Efficiency
  - Throughput must cope with Gbps link speeds
- Space Efficiency
  - Must fit in main memory of NIDS

Time/Space tradeoff



DFAs

>= 4 orders of magnitude

Space efficient DFA

NFA-OBDD

IDEAL

NFAs

Space Usage

Processing time

DFA : Deterministic Finite Automaton

NFA : Non-Deterministic Finite Automaton

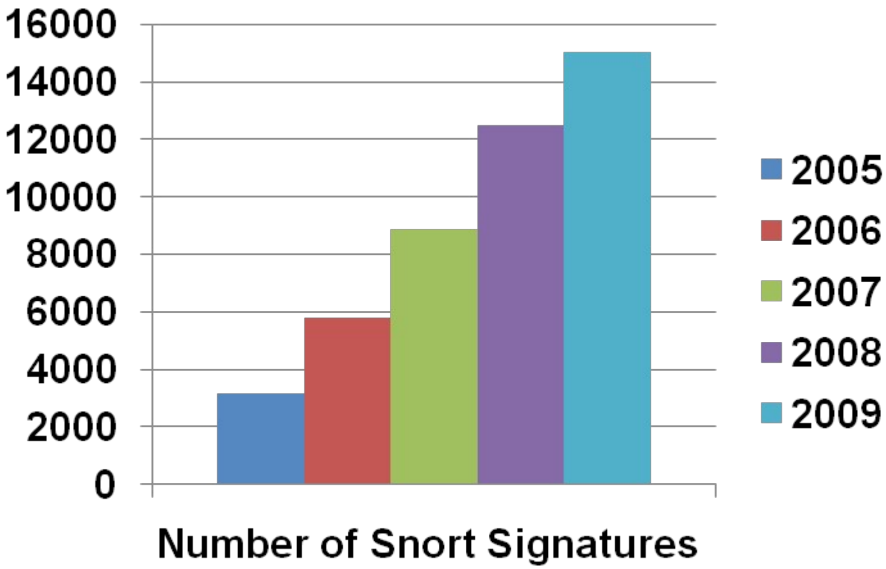
Contributions of our paper

- **NFA-OBDD**: New data structure that offers fast regular expression matching with space consumption comparable to that of NFAs

Key Idea: Boolean encoding of NFA operation

- Up to 1645x faster than NFAs with comparable memory consumption
- Speed is competitive with DFA variants
  - DFA runs out of memory for our signature sets
- Outperforms or is competitive with PCRE

Trends and challenges



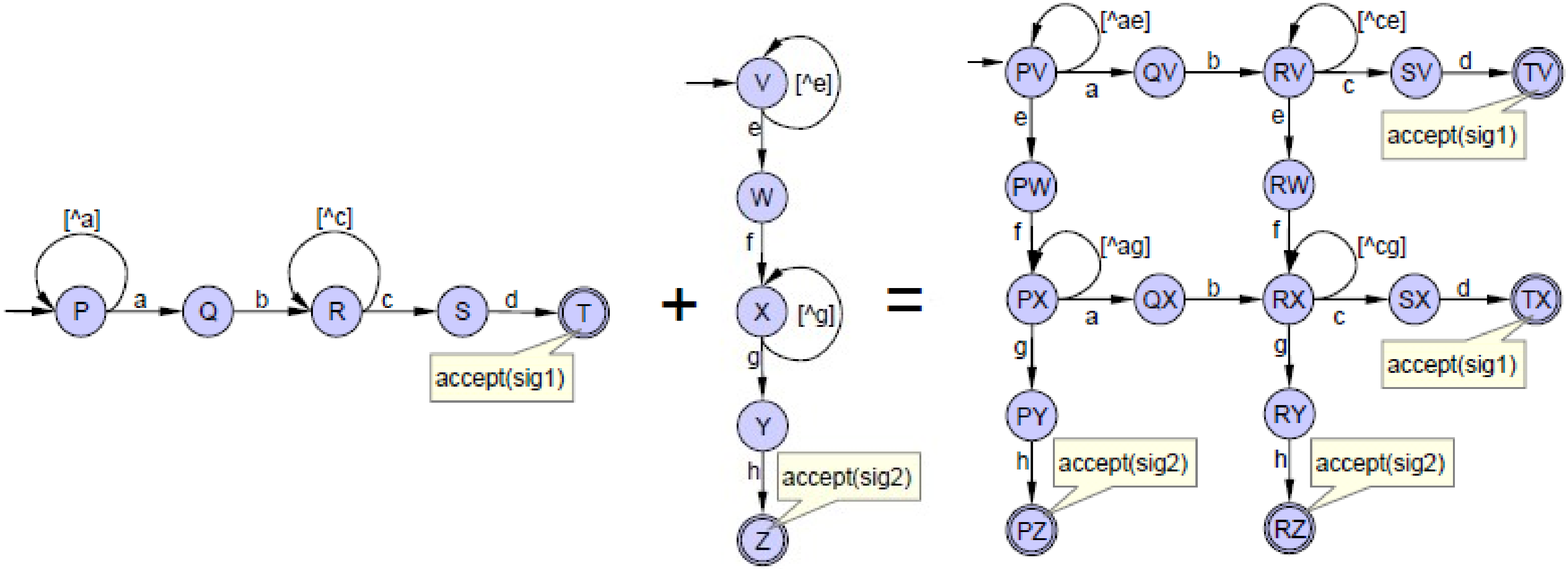
Signature set size increased 5x in the last 5 years

Challenge

Perform fast matching with low memory consumption

Combining DFAs

Multiplicative increase in number of states

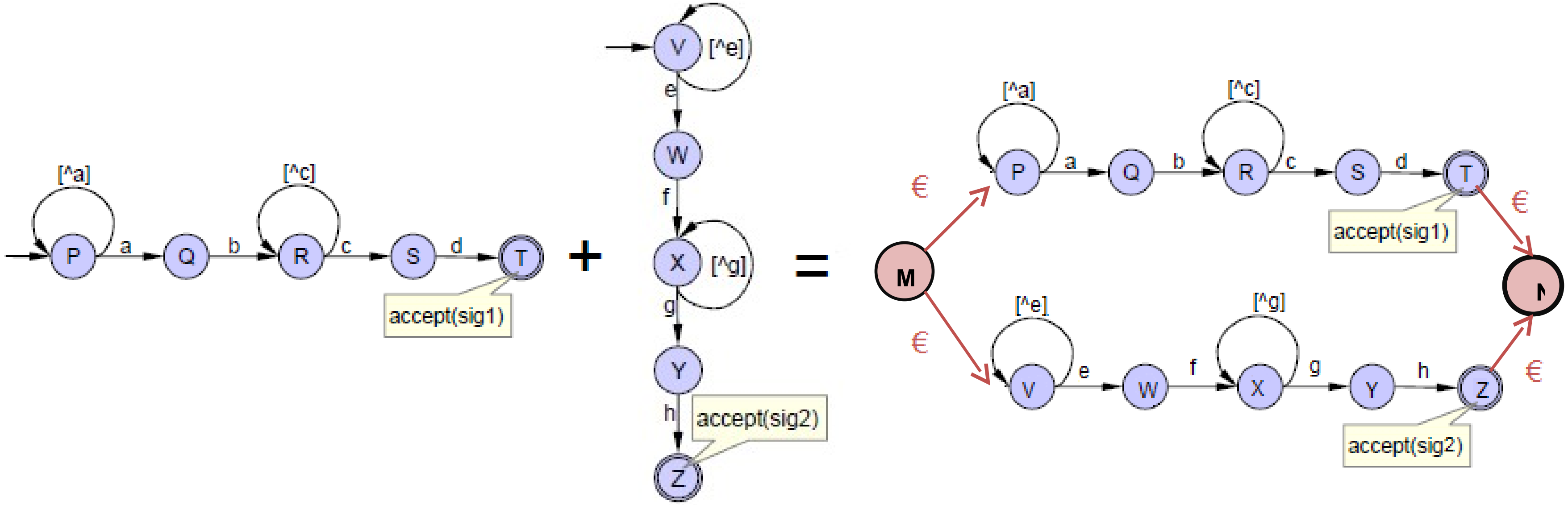


/.*ab.*cd/ **/.*ef.*gh/** **/.*ab.*cd | .*ef.*gh/**

Picture courtesy : [Smith et al. Oakland'08]

Combining NFAs

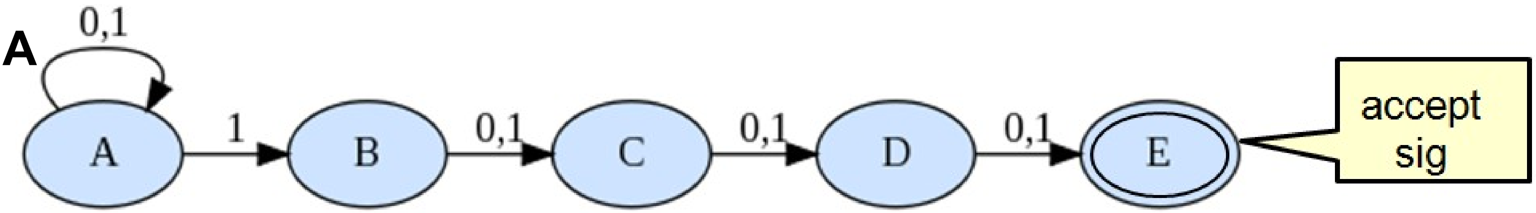
Additive increase in number of states



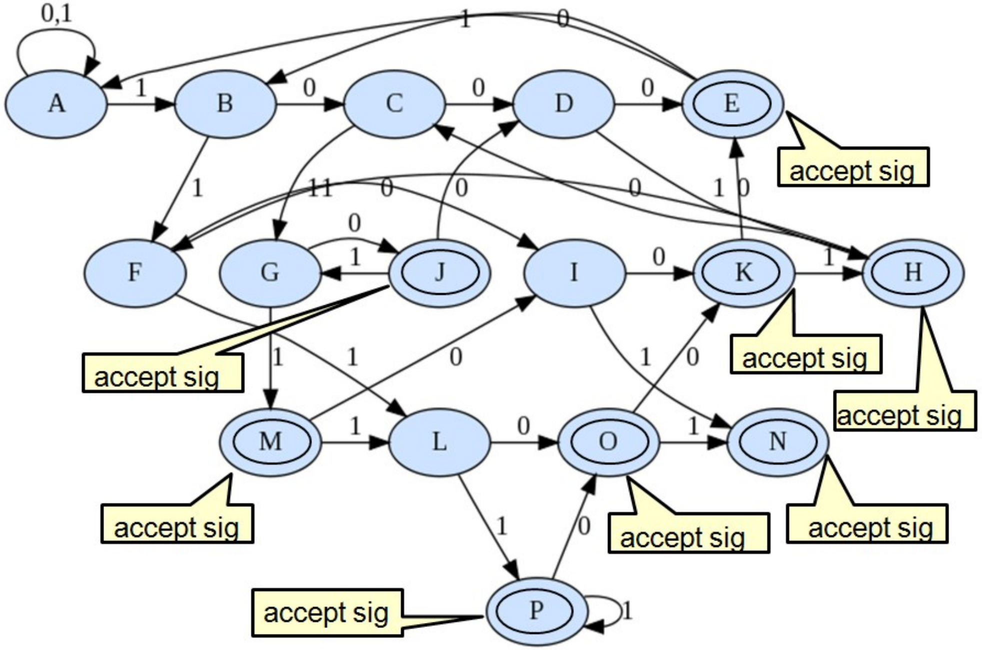
/.*ab.*cd/ **/.*ef.*gh/** **/.*ab.*cd | .*ef.*gh/**

NFAs more compact than DFAs

NFA



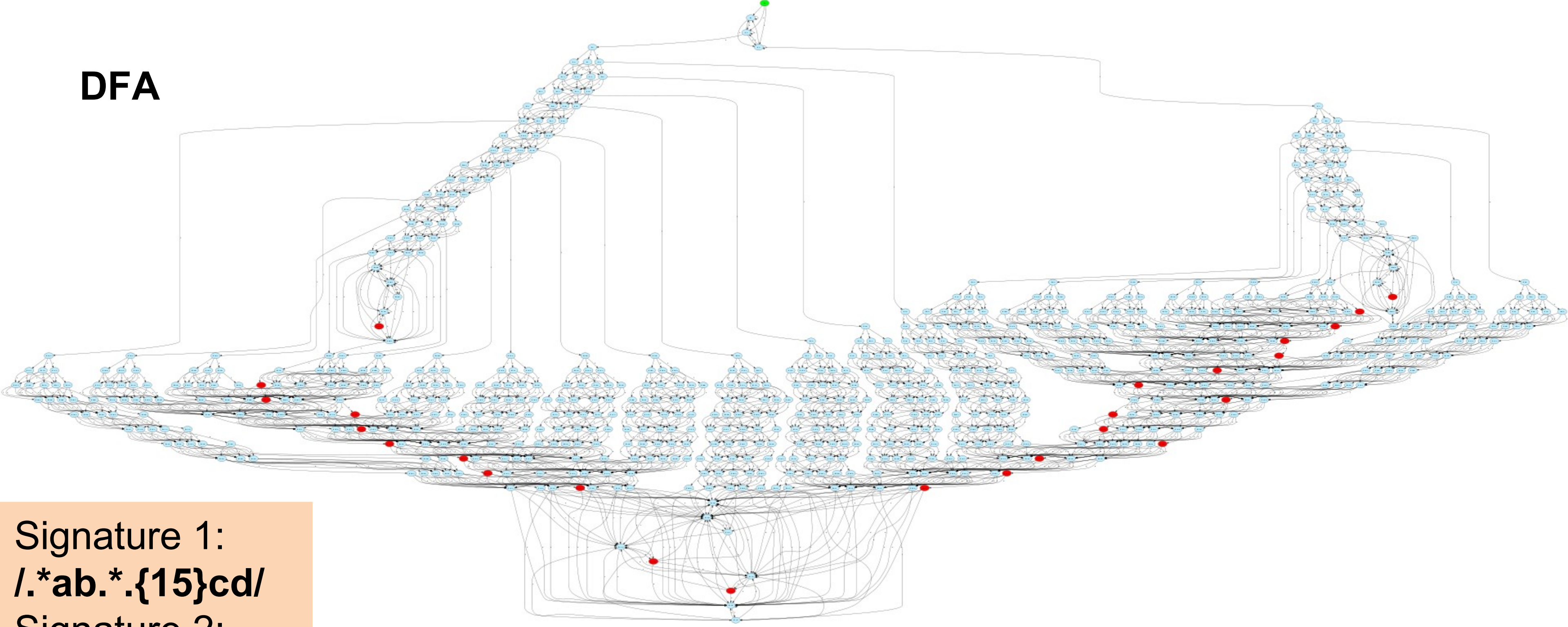
DFA



Signature
/.*1[0|1] {3} /

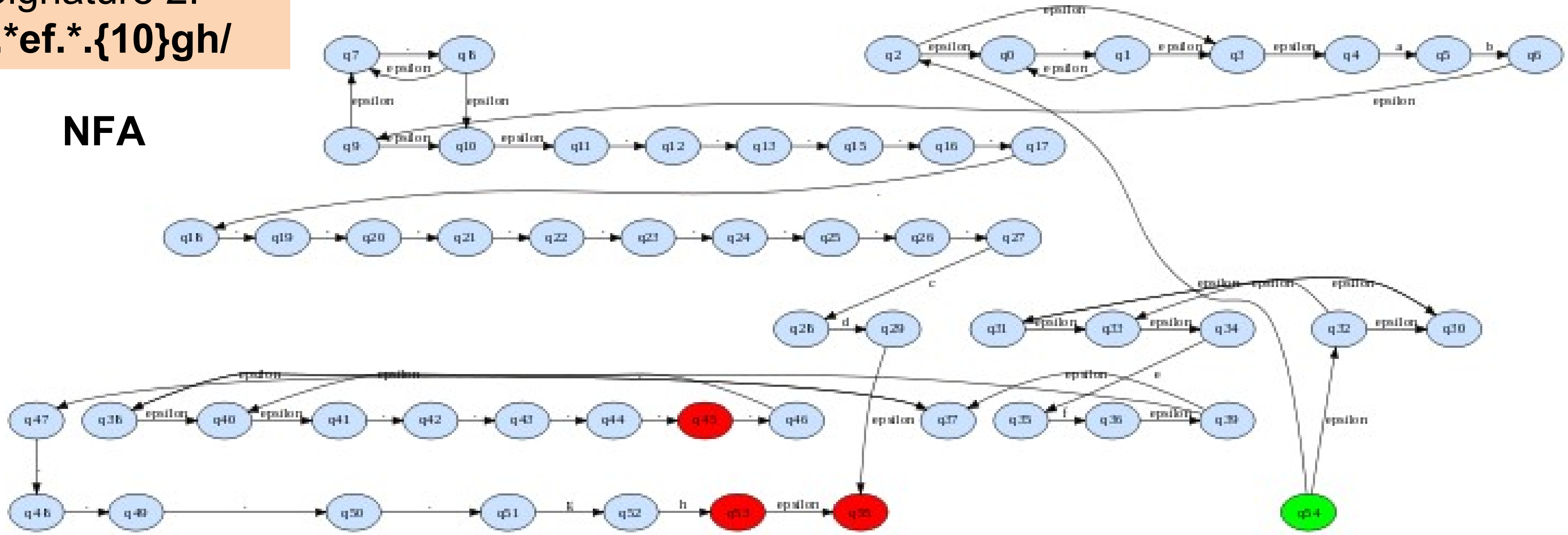
Real Snort signatures have large counter values

DFA



Signature 1:
/.*ab.*.{15}cd/
Signature 2:
/.*ef.*.{10}gh/

NFA



Outline

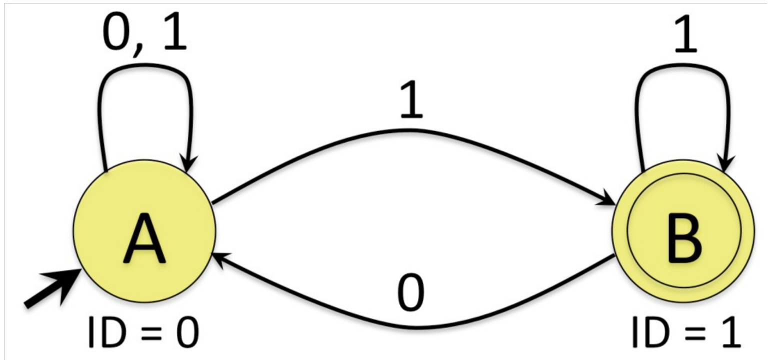
- Problem Definition
- Our Contribution
- NFA-OBDD model and operation
- Implementation
- Evaluation
- Related Work
- Conclusion

NFA-OBDDs: Main idea

- Why are NFAs slow?
  - NFA frontiers may contain multiple states
  - Each frontier state must be processed for each symbol in the input.
- **Idea:** Represent and operate NFA frontiers symbolically using Boolean functions
  - Entire frontier can be modified using a single Boolean formula
  - Use ordered binary decision diagrams (OBDDs) to represent Boolean formulae

Encoding NFA transition functions

- An NFA for (0|1)*1, ∑ = {0,1}



- Frontier: Set of current states
  - Size: O(n); n= # of states
- Set membership function
  -Disjunction of binary values of member states

- Transition function

x	i	y	*f*(*x*,*i*,*y*)
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

NFA operation

- Determine new frontier after processing input:

  Next set of states =

$$
\begin{aligned}
\text{Map}_{y\to x}\ (\ \exists_{x,i}\quad & \text{Transition\_Function}(x,i,y) \\
& \wedge\ \text{Frontier}(x) \\
)\quad & \wedge\ \text{Input\_Symbol}(i)
\end{aligned}
$$

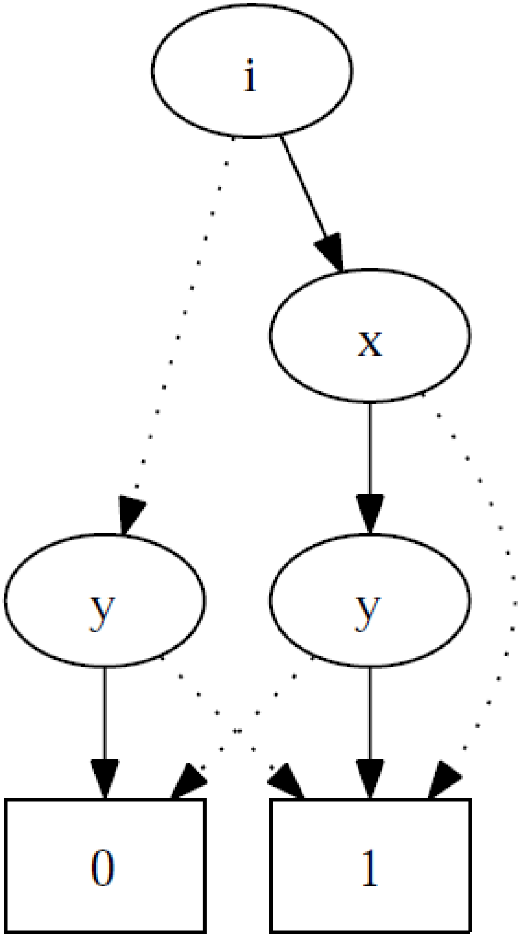
- Checking acceptance:

$$\text{SAT}(\text{Set\_of\_ Accept\_States}(x) \wedge \text{Frontier}(x))$$

Ordered binary Decision Diagram (OBDD) [Bryant 86]

- Compact representation of Boolean function

x	i	y	f(x,i,y)
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



The BDD of *f*(*x*,*i*,*y*) with order i < x < y

NFA-OBDD

- NFA-OBDD: NFA representation and operation using OBDDs
- OBDD Representation of
  - Transitions
  - Frontiers
    - Current set of states
  - Input symbols
  - Set of accepting states
  - Set of start states

Space efficiency of NFA-OBDDs

- NFA-OBDD construction:
  - Uses same combination algorithm as NFAs
  - OBDD data structure itself utilizes the redundancy of the binary function table

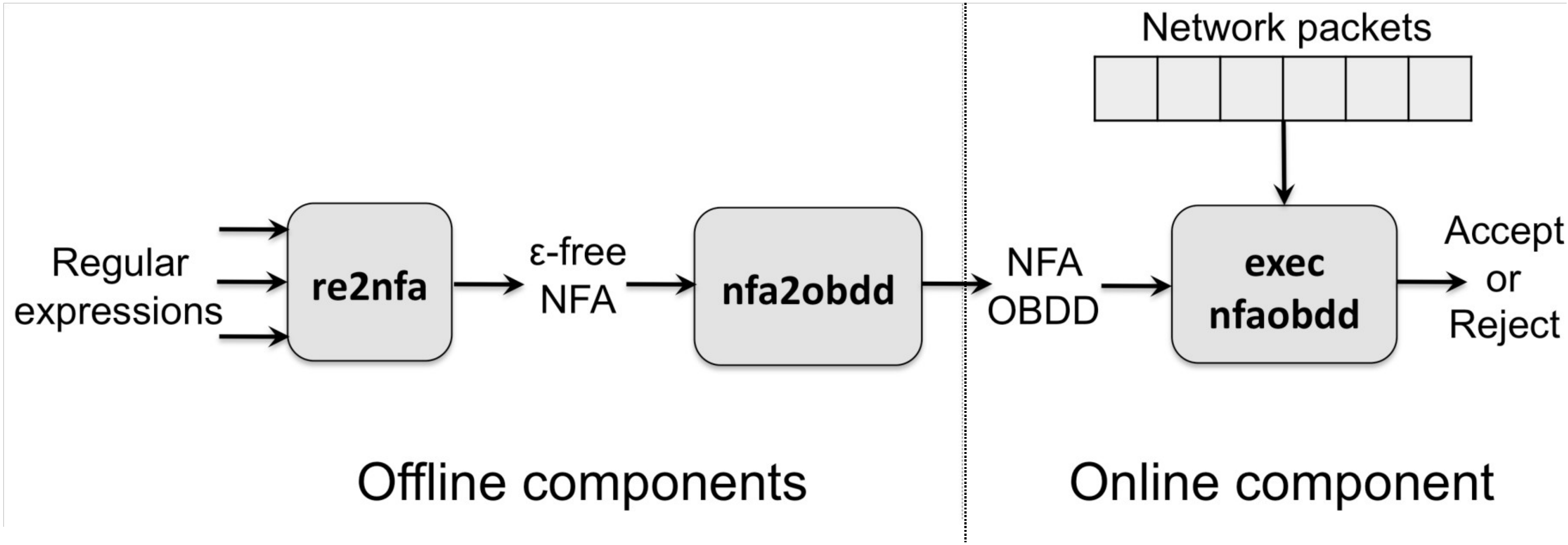
Rapid growth of signature set has little impact on NFA-OBDD space consumption (unlike DFAs)

Outline

- ❑Problem Definition
- ❑Our Contribution
- ❑NFA-OBDD model and operation
- ❑Implementation
- ❑Evaluation
- ❑Related Work
- ❑Conclusion

Experimental apparatus

- C++ and CUDD package for OBDDs



Regular expression sets

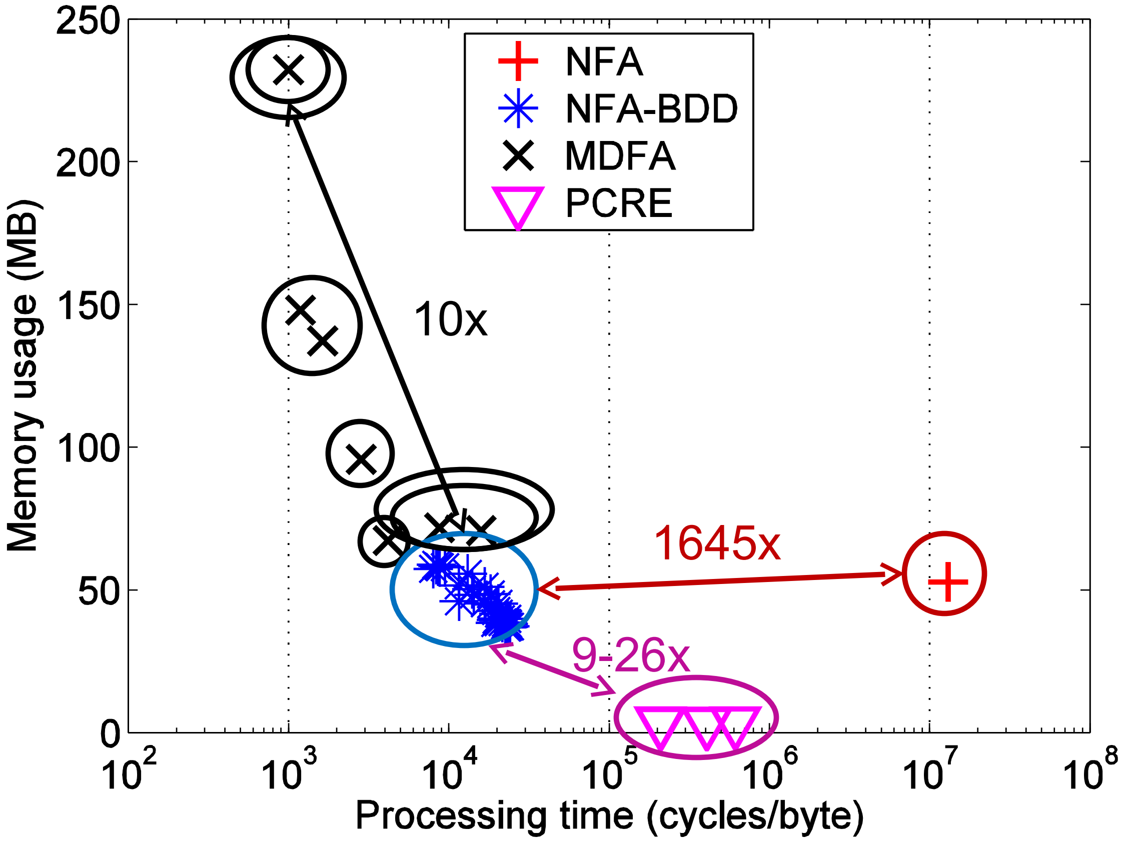
- Snort HTTP signature set
  - 1503 regular expressions from March 2007
  - 2612 regular expressions from October 2009
- Snort FTP signature set
  - 98 regular expressions from October 2009
- Extracted regular expressions from `pcre` and `uricontent` fields of signatures

Traffic traces

- HTTP traces
  - 33 traces
  - Size: 5.1MB –1.24 GB
  - One week period in Aug 2009 from Web server of the CS department at Rutgers
- FTP Traces
  - 2 FTP traces
  - Size: 19.4MB, 24.7 MB
  - Two week period in March 2010 from FTP server of the CS department at Rutgers

Experimental results

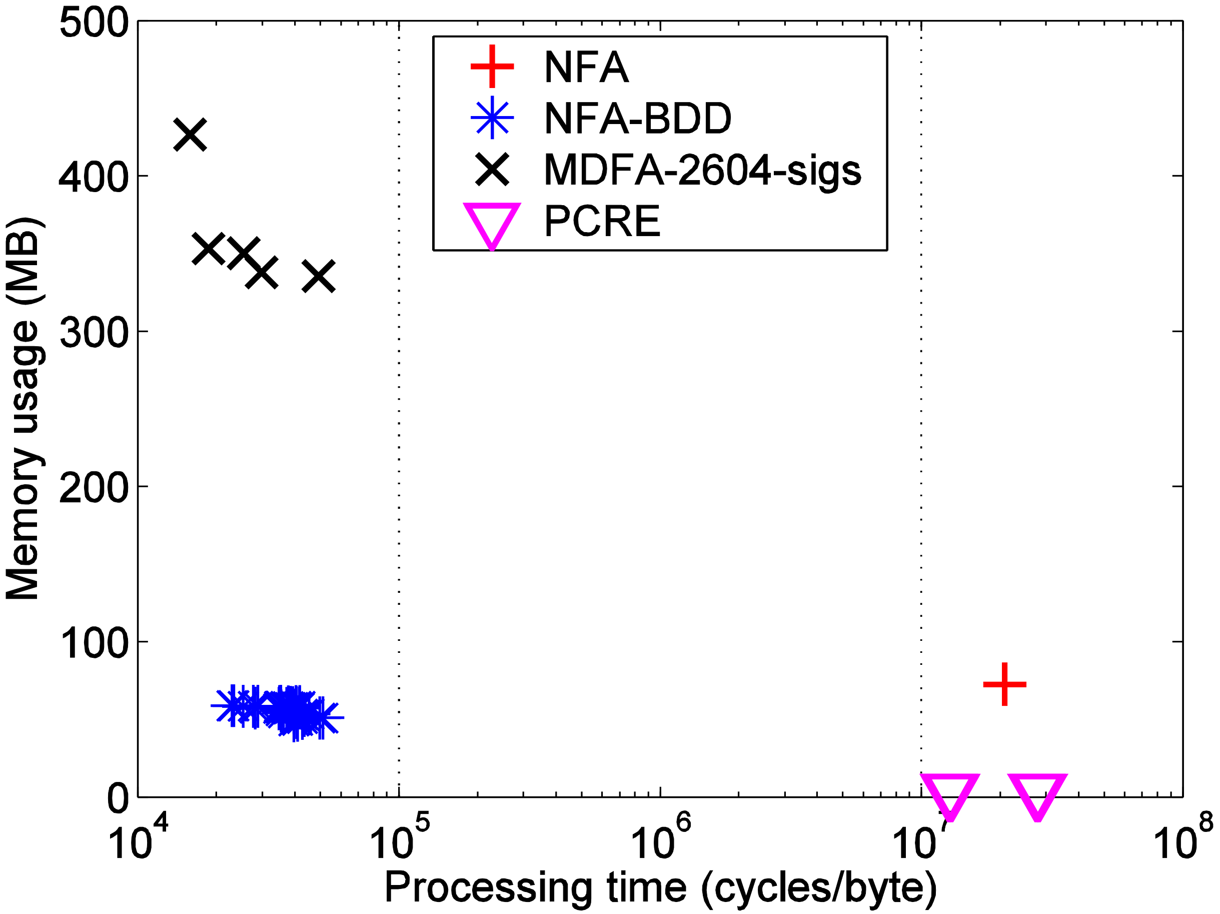
- For 1503 REs from HTTP Signatures



Intel Core2 Duo E7500, 2.93GHz; Linux-2.6; 2GB RAM

Experimental results

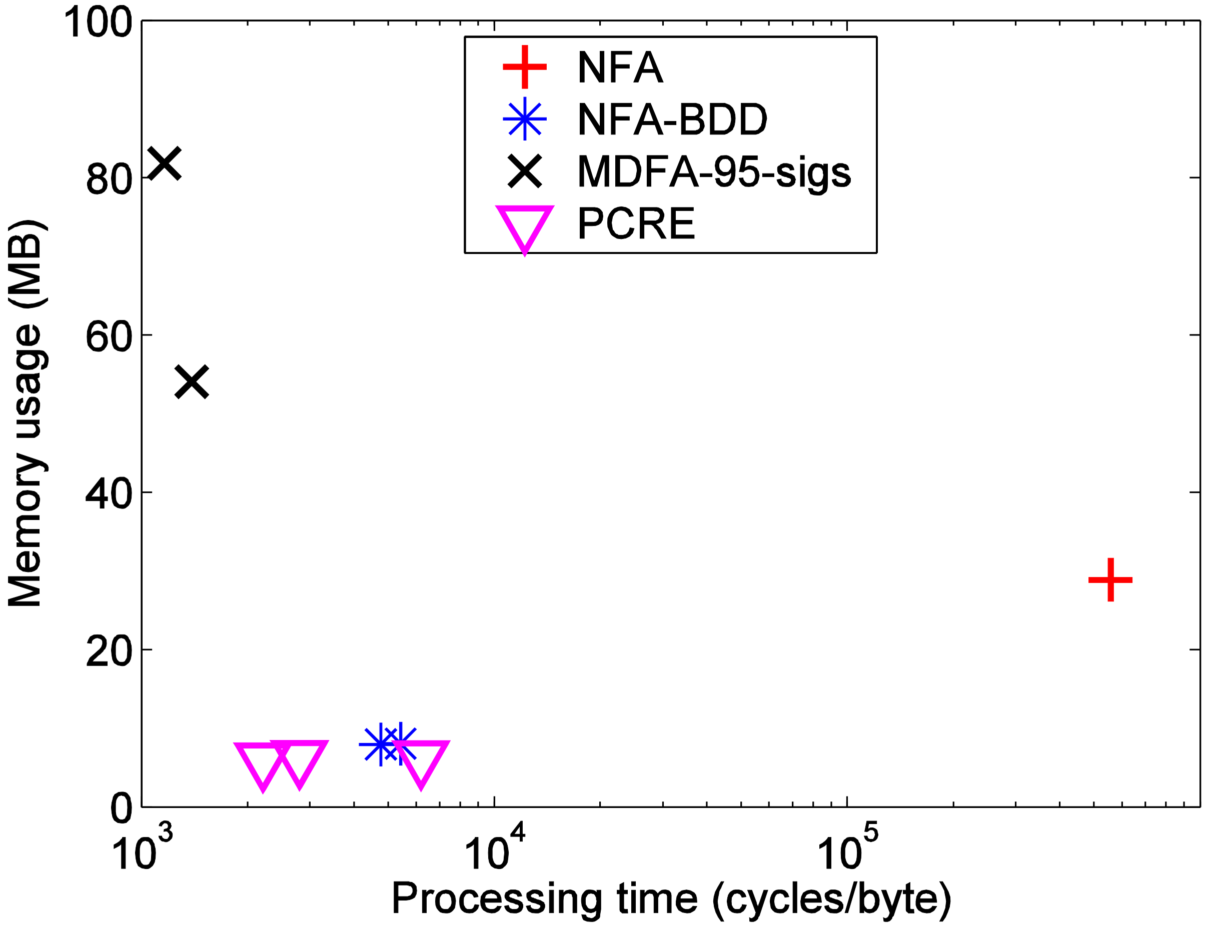
- For 2612 REs from HTTP signatures



MDFA ran out of memory for 2612 REs

Experimental results

- For 98 RE from FTP signatures



MDFA ran out of memory for 98 REs

Multibyte matching

- Matches k>1 input symbol in a single step
- Also possible with NFA-OBDDs
  - Use OBDDs to represent k-step transitive closure of NFA transition function
  - See paper for details.
- Brief summary of experimental results
  - 2-stride NFA-OBDD doubles the throughput
  - Outperforms 2-stride NFA by 3 orders of magnitude

Related work

- Multiple DFAs [Yu *et al.*, ANCS'06]
- Extended finite automata [Smith *et al.*, Oakland'08, SIGCOMM'08]
- D^{2}FA [Kumar *et al.*, SIGCOMM'06]
- Hybrid finite automata [Becchi *et al.*, ANCS'08]
- Multibyte speculative matching [Luchaup *et al.*, RAID'09]
- Many more – see paper for details

Conclusion

- NFA-OBDDs
  - Outperform NFAs by three orders of magnitude
    - Up to 1645× in the best case
    - Retain space efficiency of NFAs
  - Outperform or competitive with the PCRE package
  - Competitive with variants of DFAs but drastically less memory-intensive

Thank You

Improving Signature Matching using Binary Decision Diagrams

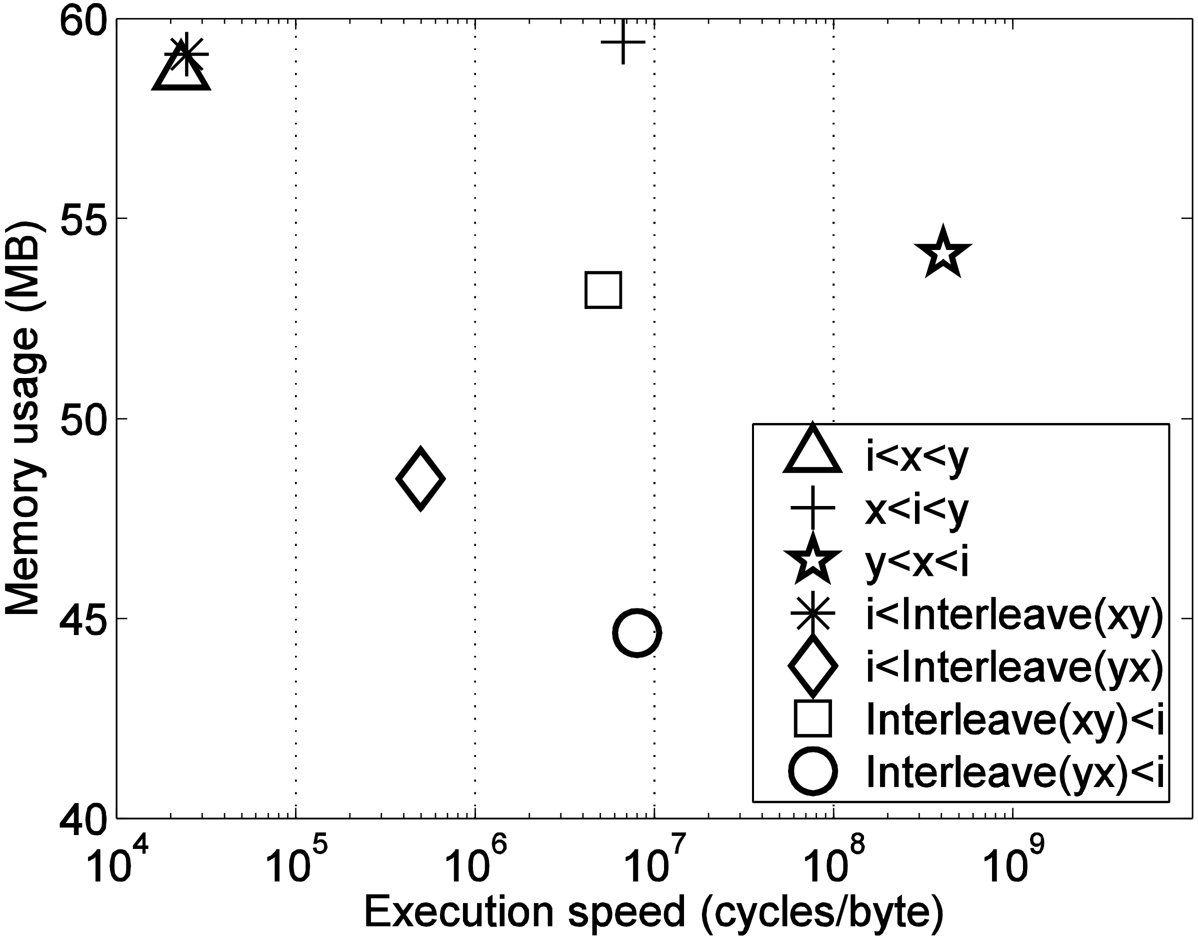
Liu Yang - lyangru@cs.rutgers.edu

Rezwana Karim - rkarim@cs.rutgers.edu

Vinod Ganapathy - vinodg@cs.rutgers.edu

Randy Smith - ransmit@sandia.gov

BDD var ordering


Memory usage (MB)
10^4
10^5
10^6
10^7
10^8
10^9
Execution speed (cycles/byte)
△ i<x<y
+ x<i<y
☆ y<x<i
✳ i<Interleave(xy)
◇ i<Interleave(yx)
□ Interleave(xy)<i
○ Interleave(yx)<i