Efficient Signature Matching with Multiple Alphabet Compression Tables

Shijin Kong
Randy Smith
Cristian Estan

Presented at SecureComm 2008, Istanbul, Turkey

Signature Matching

- Signature Matching a core component of network devices

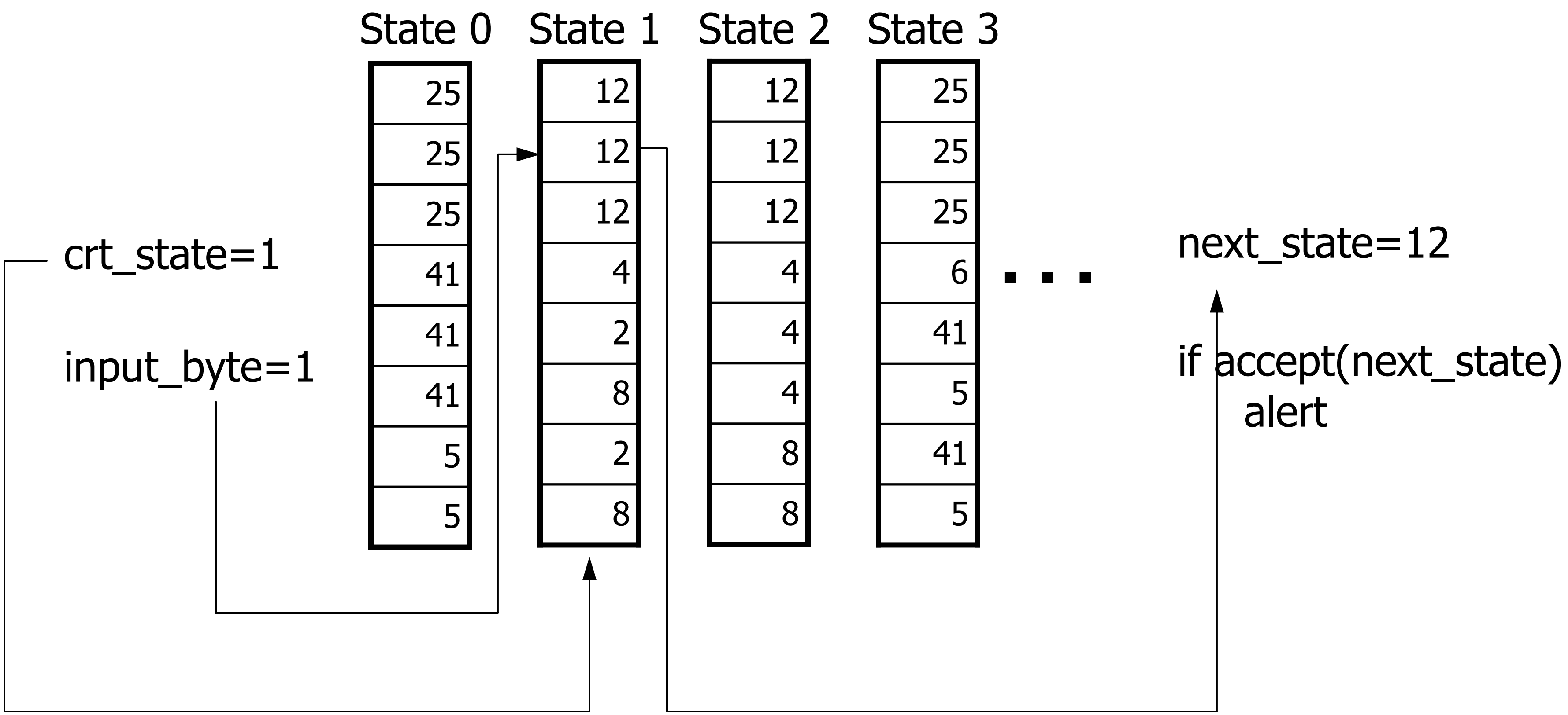
- Operation (ideal): For a set of signatures, match all relevant sigs in a single pass over payload

- Many constraints
  - Evolving, complex signatures
  - Wirespeed operation
  - Limited memory
  - Active adversary

Regular Expressions and DFAs

- Regular expressions standard for *writing* sigs
  - Buffer overflow: `/^RETR\s[^\n]{100}/`
  - Format string attack: `/^SITE\s+EXEC[^\n]*%[^\n]*%/`
- DFAs used for *matching* to input

DFA Operation



Matching with DFAs

- Advantages
  - Fast – minimal per-byte processing
  - Composable – combine many DFAs into one

- Disadvantages
  - States are heavyweight (1 KB each!)
  - State-space explosion occurs when DFAs combined

- Memory exhausted with only a few DFAs!
- Workaround: many DFAs matched in parallel

Key: Reduce memory usage

Strategy: aggressively reduce memory footprint, keep exec time low

Reduce size of transition tables

State 0	State 1	State 2	State 3
25	12	12	25
25	12	12	25
25	12	12	25
41	4	4	6
41	2	4	41
41	8	4	5
5	2	8	41
5	8	8	5

. . .

Reduce number of states

Main Contribution

- Multiple Alphabet Compression Tables
  - Lightweight, applicable to hardware or software
  - Compatible with other techniques
  - Worst case = average case

- Results (in software)
  - 4x to 70x memory reduction
  - 35% - 85% execution time increase

Outline

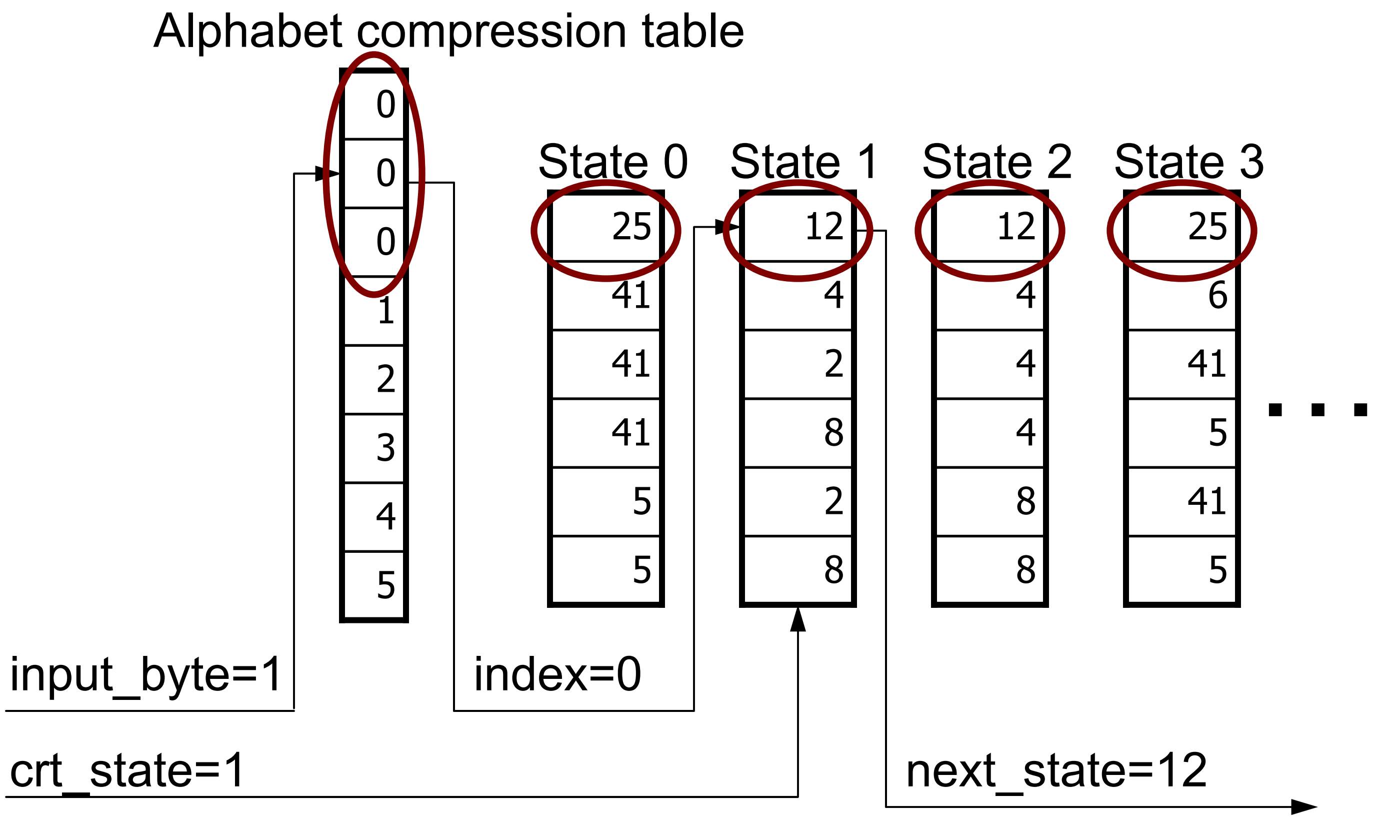
- Introduction
- Alphabet Compression Tables
- Interacting with D^2FAs
- Experimental Results

Alphabet Compression: core observation

Some input symbols are equivalent; the transitions on those symbols at any state are identical.

State 0	State 1	State 2	State 3	
25	12	12	25	
25	12	12	25	
25	12	12	25	
41	4	4	6	. . .
41	2	4	41	
41	8	4	5	
5	2	8	41	
5	8	8	5	

Alphabet Compression Tables



Even further compression…

Alphabet compression table

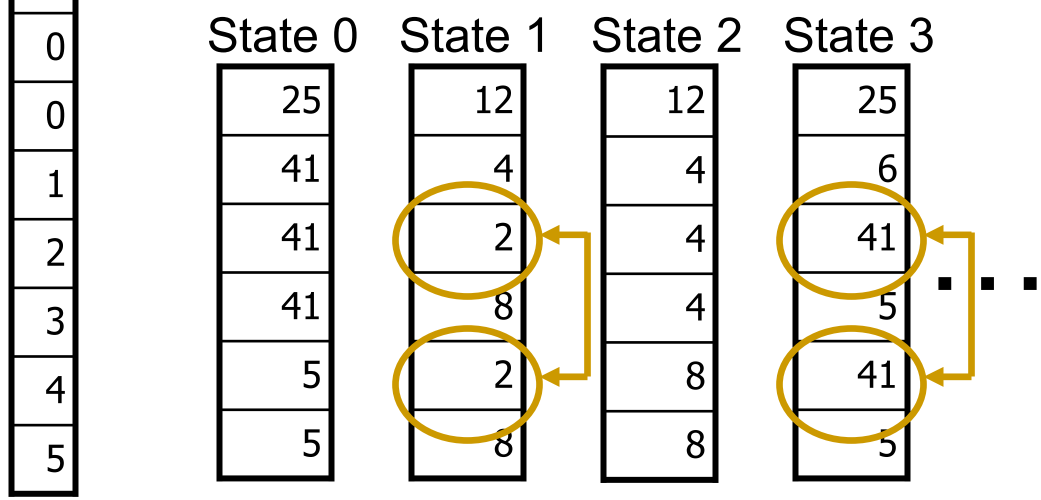
0
0
0
1
2
3
4
5

State 0	State 1	State 2	State 3
25	12	12	25
41	4	4	6
41	2	4	41
41	8	4	5
5	2	8	41
5	8	8	5

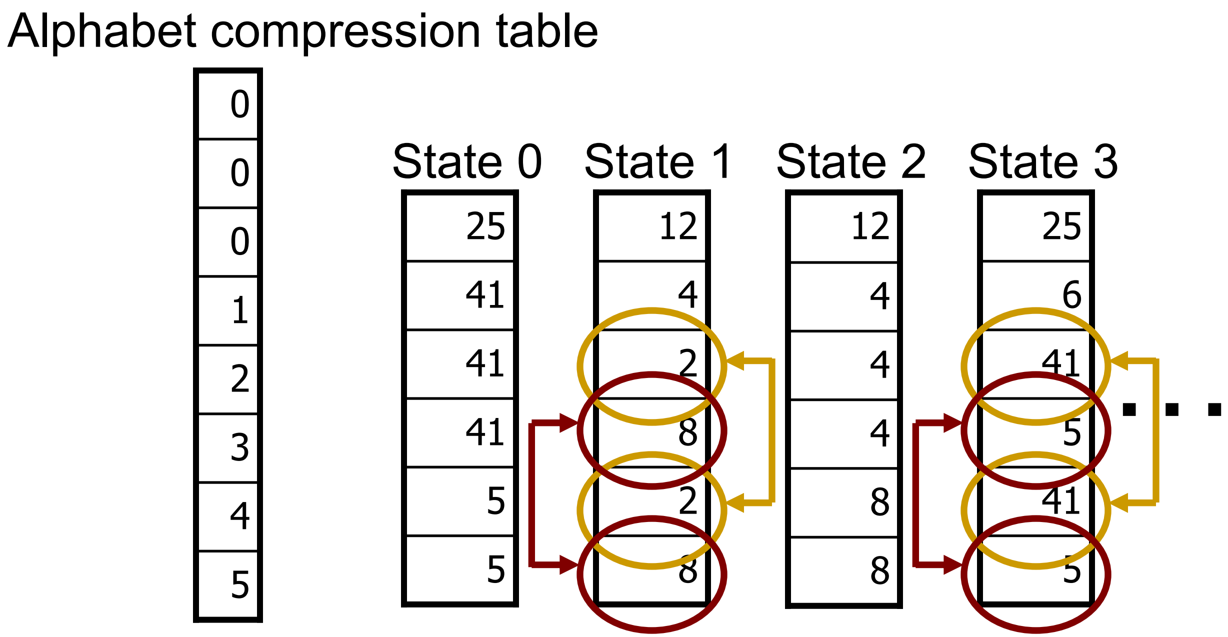
. . .

Even further compression…

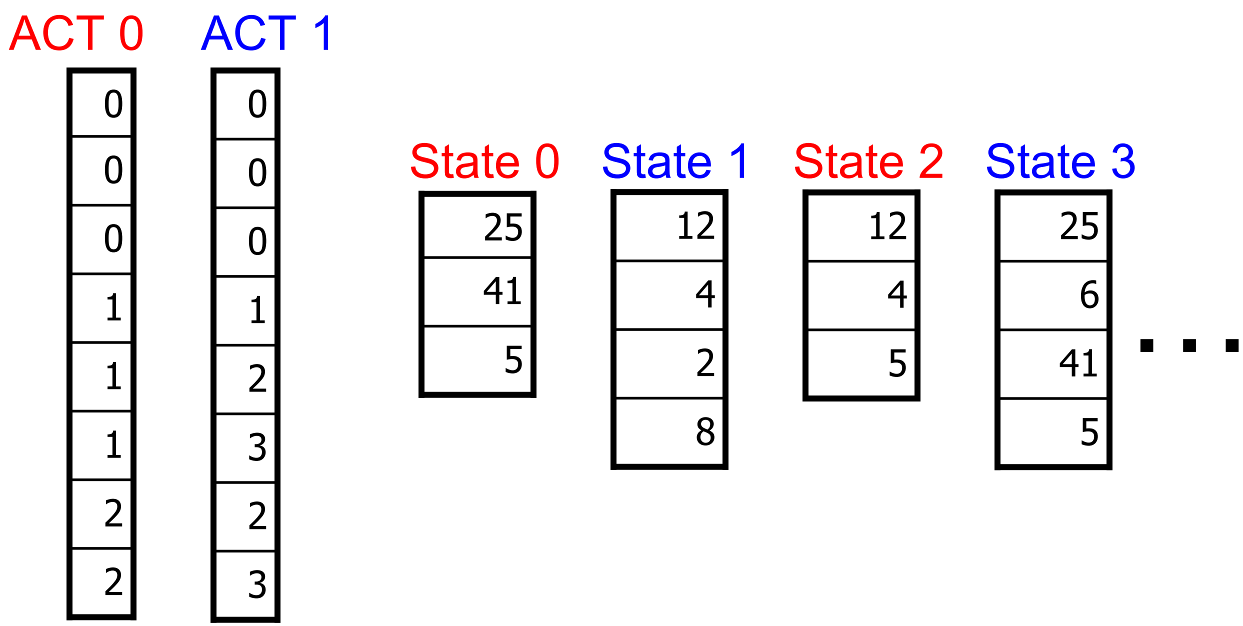
Alphabet compression table



Even further compression…

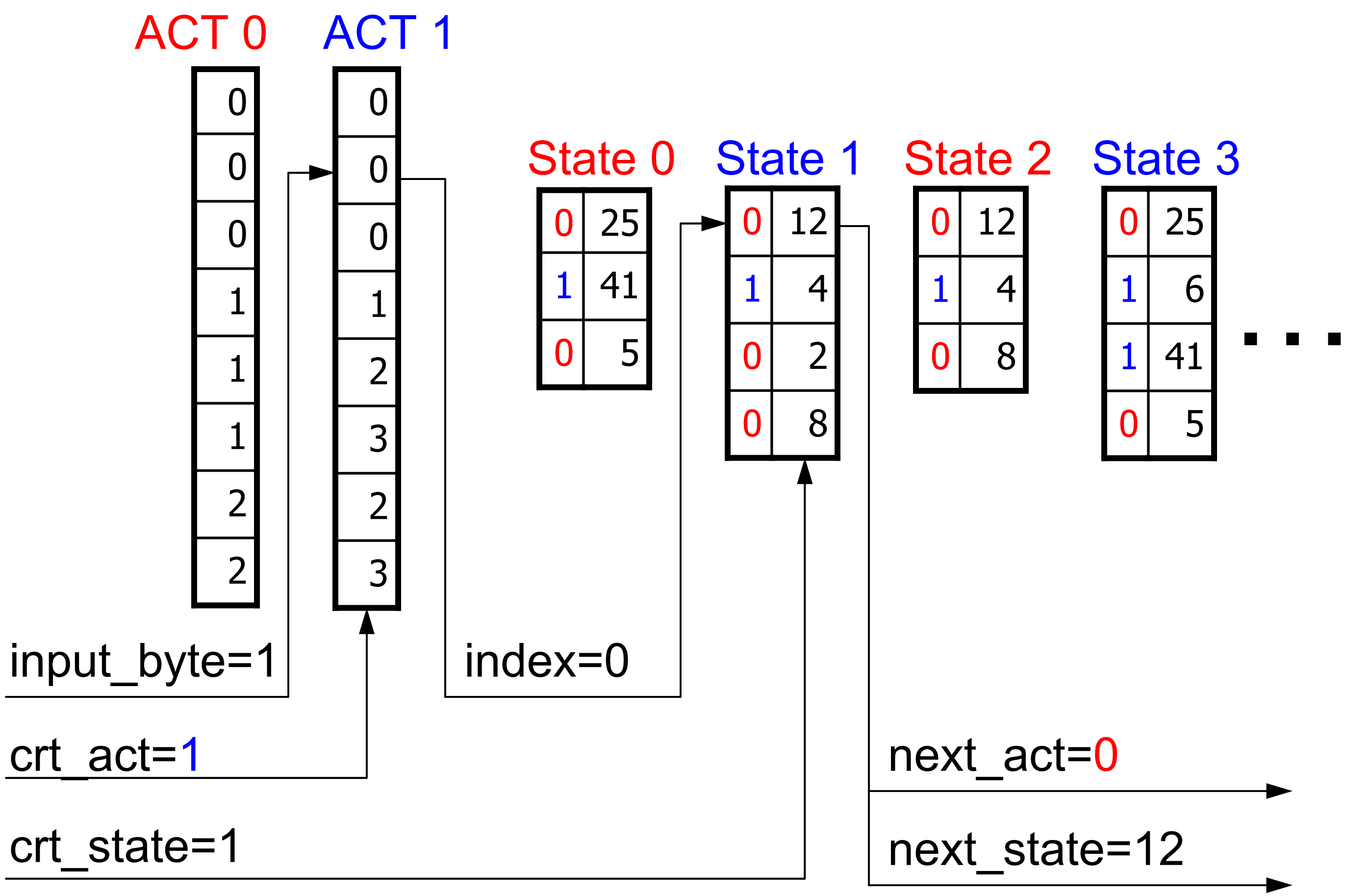


Multiple ACTs



How do we know which ACT to use with which state?

Multiple ACTs



Constructing Multiple ACTs

- Partition states appropriately
  - for example:

    $\{S_1, S_2, S_3, \ldots, S_n\} \rightarrow \{\ \{S_1, S_8,\},\ \{S_2, S_3, S_9,\},\ \ldots\ \}$

- Construct single ACT for each group of states
  - See algorithm in paper

Partitioning States for ACTs

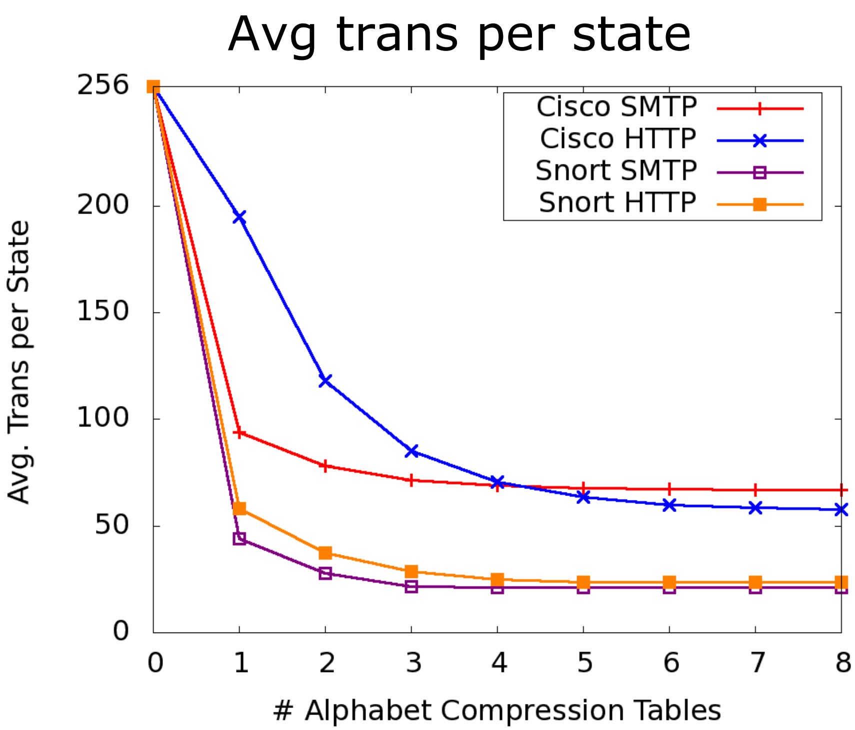
- Input: number of ACTs to use *m*, DFA *D*
- Output: a partition of states into *m* subsets

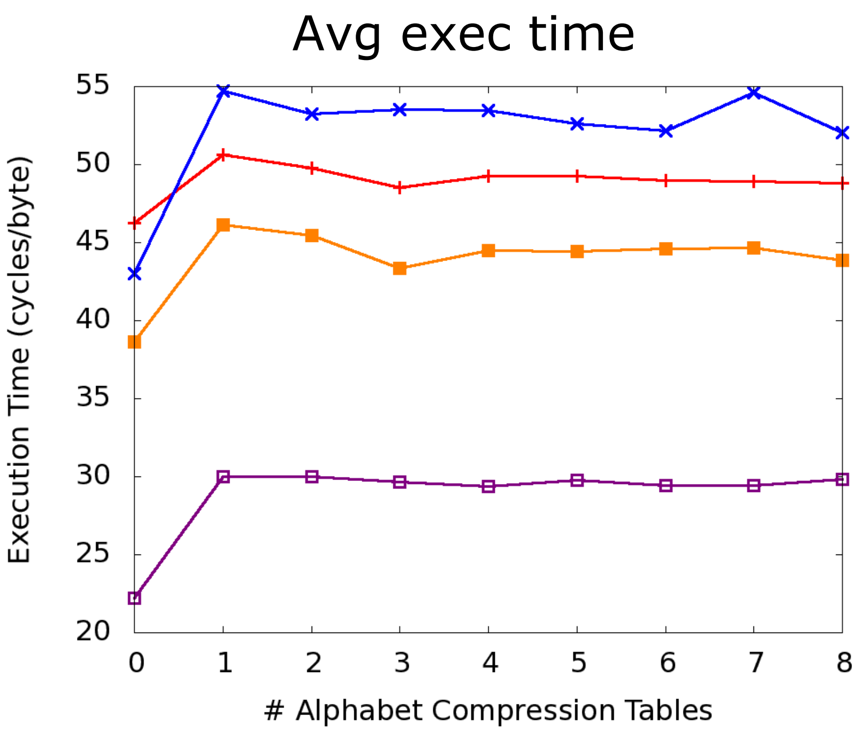
- Use greedy, heuristic approach:

```
States = Set of all states in D;
while (m>1) {
        Subset = GetEquivClassPartition(States);
        AddToResult(Subset);
        States = States - Subset;
        m--;
}
return Result;
```

How many Compression Tables?

- Eight ACTs is enough



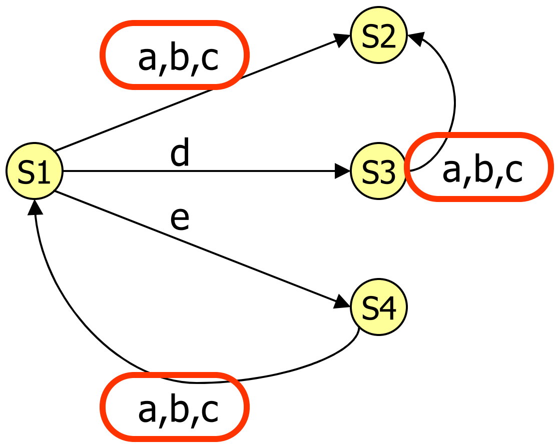


Outline

- Introduction
- Alphabet Compression Tables
- Interacting with D^2FAs
- Experimental Results

ACTs and D²FAs

- Two kinds of redundancy

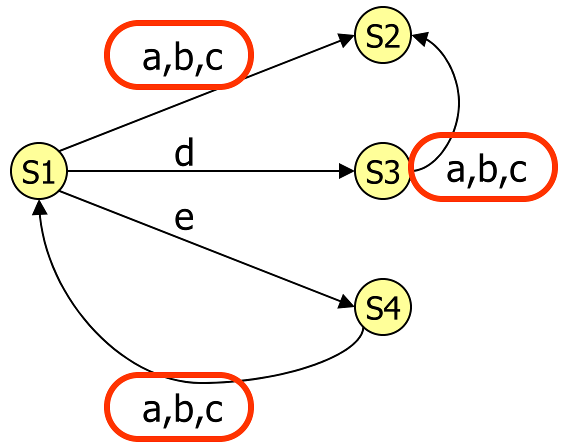


Symbols have identical behavior
for large subsets of states

Compress with (multiple) ACTs

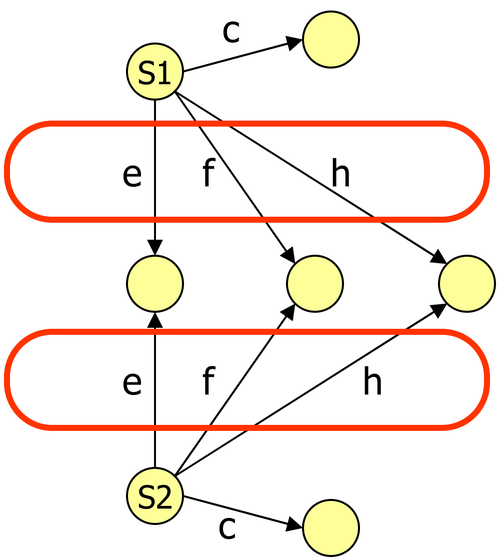
ACTs and D^2FAs

- Two kinds of redundancy



Symbols have identical behavior for large subsets of states

Compress with (multiple) ACTs



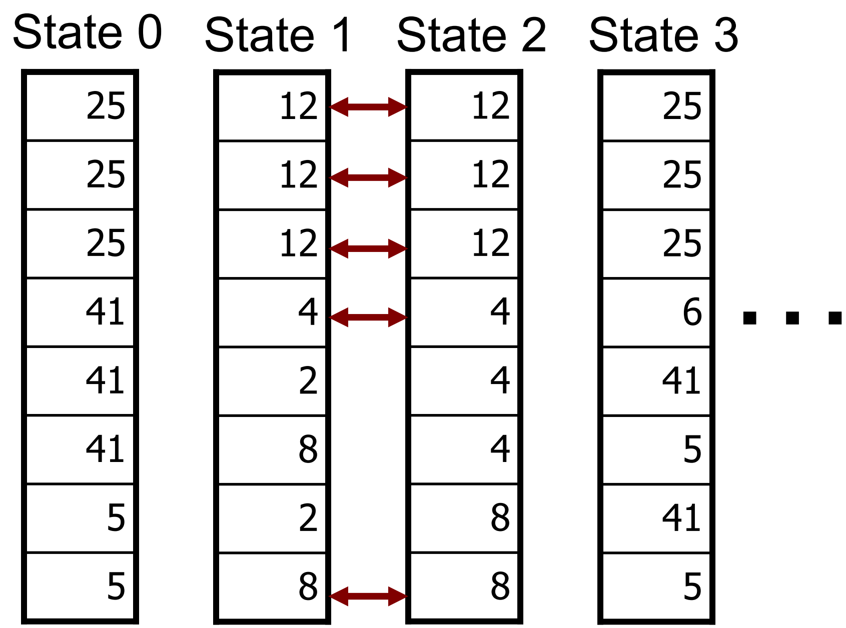
Symbols at many states lead to common next states

Compress with D^2FAs

D²FAs: core observation

For many pairs of states, the transitions for most characters are identical!

Idea: store only one copy

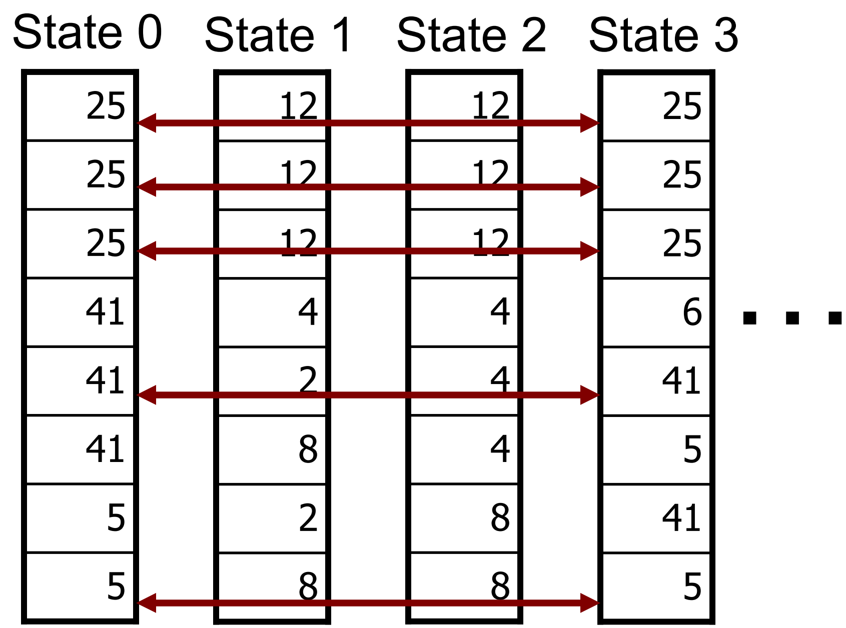


Kumar *et al* (Sigcomm 2006); Kumar *et al* (ANCS 2006); Becchi *et al* (ANCS 2007)

D²FAs: core observation

For many pairs of states, the transitions for most characters are identical!

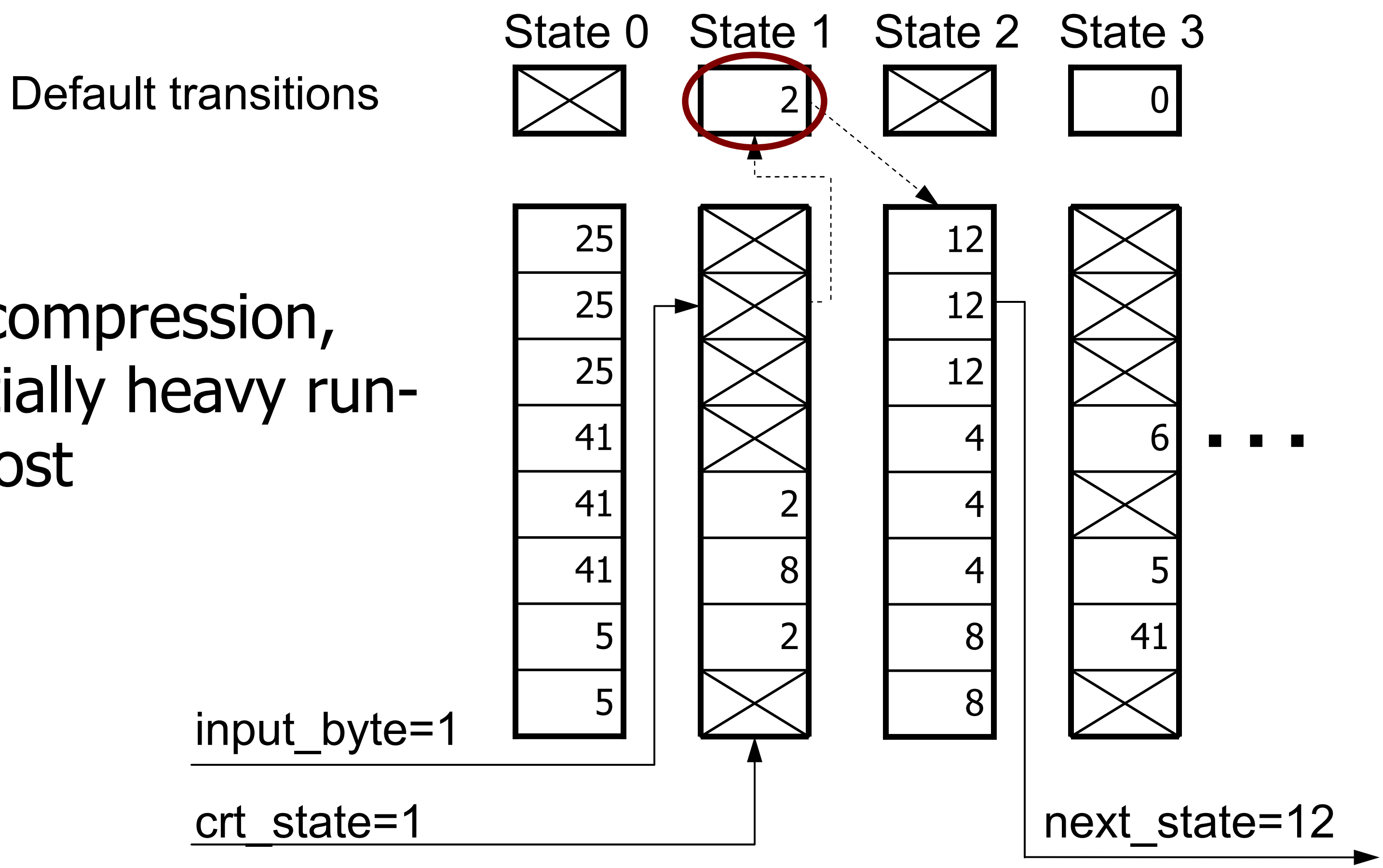
Idea: store only one copy



Kumar *et al* (Sigcomm 2006); Kumar *et al* (ANCS 2006); Becchi *et al* (ANCS 2007)

D^2FAs

Issue:
good compression, potentially heavy run-time cost



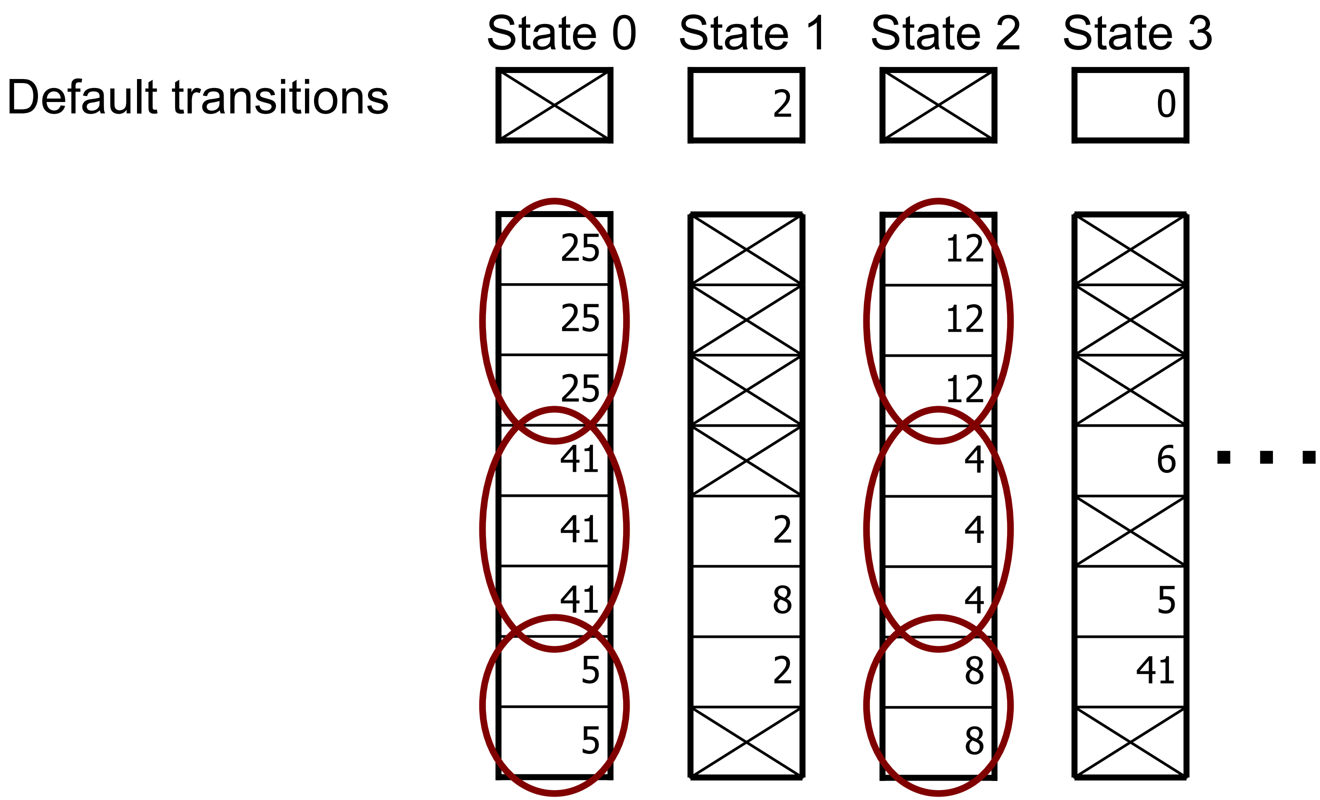
ACTs and D^2FAs Together

- Combine ACTs and D^2FAs to address both kinds of redundancy

- Procedure:
  1. Apply D^2FA compression to DFAs
  2. Apply multiple ACT compression to D^2FA results

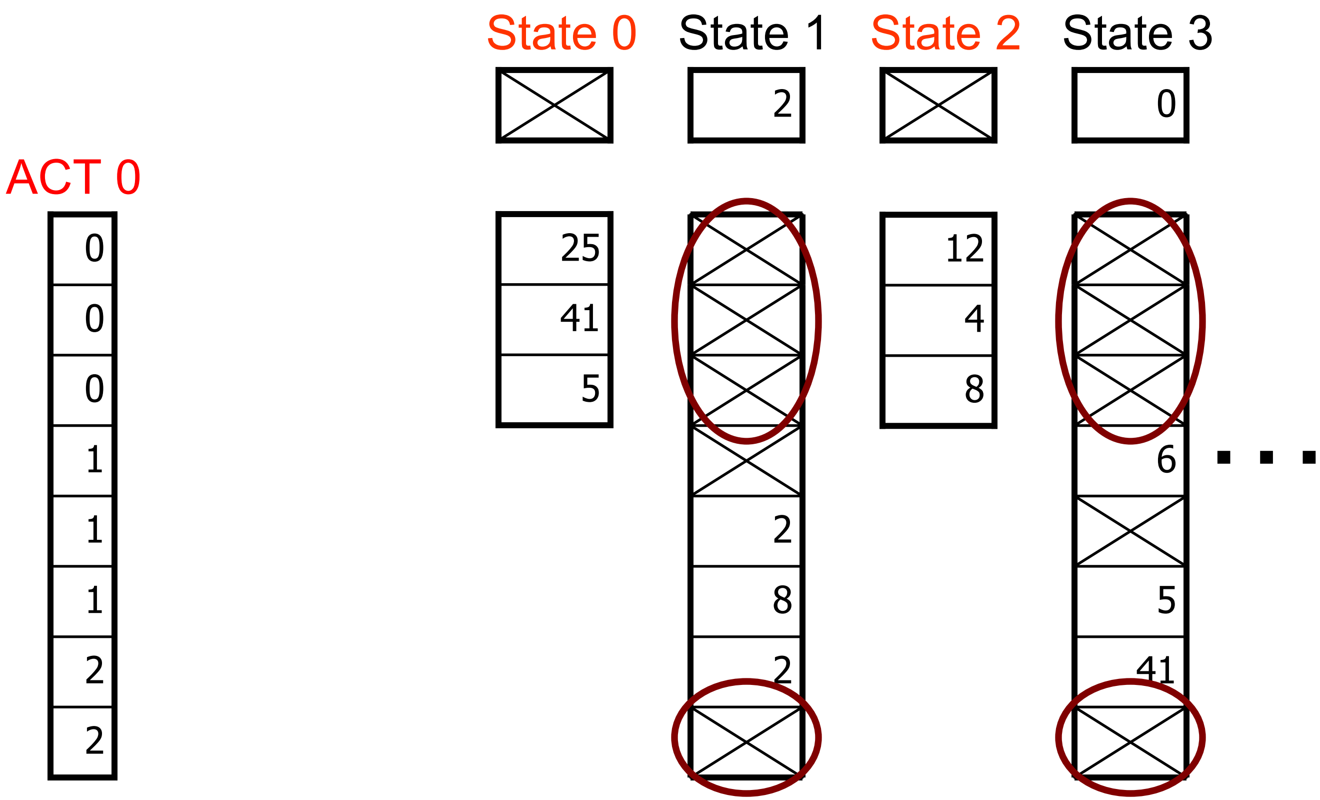
- Only slight modification to ACT construction
  - Add “not handled here” symbol
  - Deal with default transitions

ACTs + D^2FAs

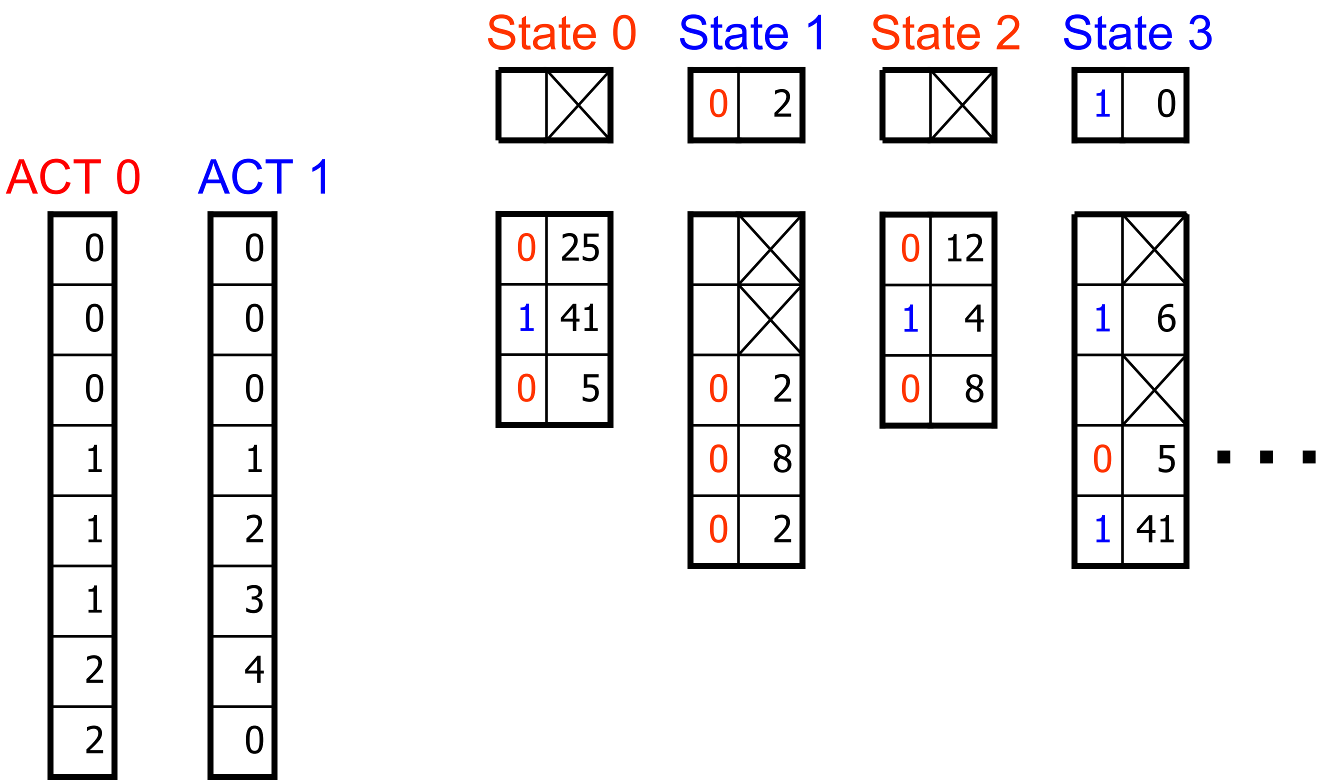


State 0
State 1
State 2
State 3
Default transitions
2
0
25
25
25
41
41
41
5
5
2
8
2
12
12
12
4
4
4
8
8
6
5
41
. . .

ACTs + D^2FAs

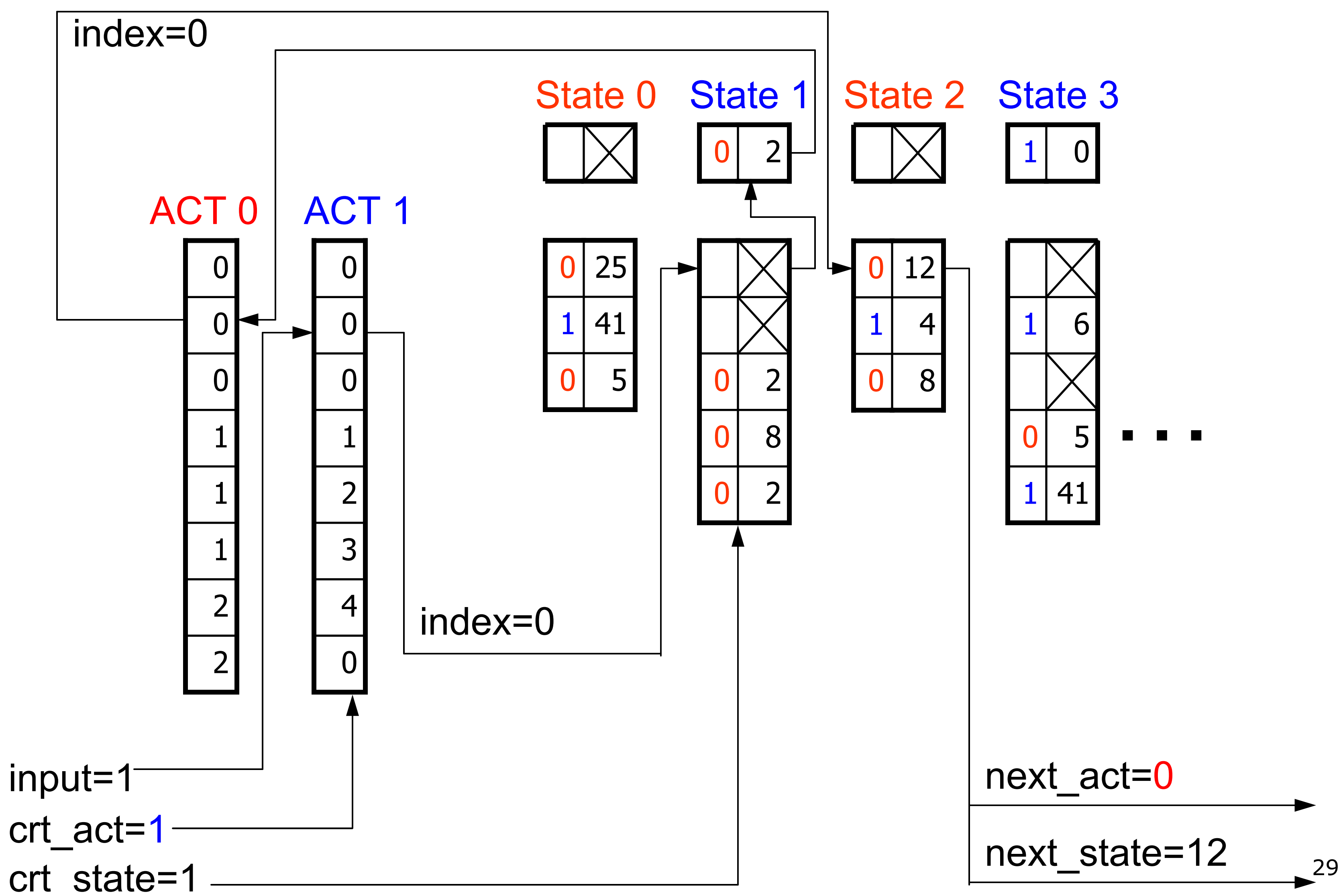

ACT 0
0
0
0
1
1
1
2
2
State 0
25
41
5
State 1
2
2
8
2
State 2
12
4
8
State 3
0
6
5
41
. . .

ACTs + D²FAs



State 0
State 1
State 2
State 3
ACT 0
ACT 1
0 2
1 0
0 25
1 41
0 5
0 2
0 8
0 2
0 12
1 4
0 8
1 6
0 5
1 41
0 0 0 1 1 1 2 2
0 0 0 1 2 3 4 0
. . .

ACTs + D^{2}FAs


index=0
State 0
State 1
State 2
State 3
0
2
1
0
ACT 0
ACT 1
0
0
0
1
1
1
2
2
0
0
0
1
2
3
4
0
0
25
1
41
0
5
0
2
0
8
0
2
0
12
1
4
0
8
1
6
0
5
1
41
. . .
index=0
input=1
crt_act=1
crt_state=1
next_act=0
next_state=12

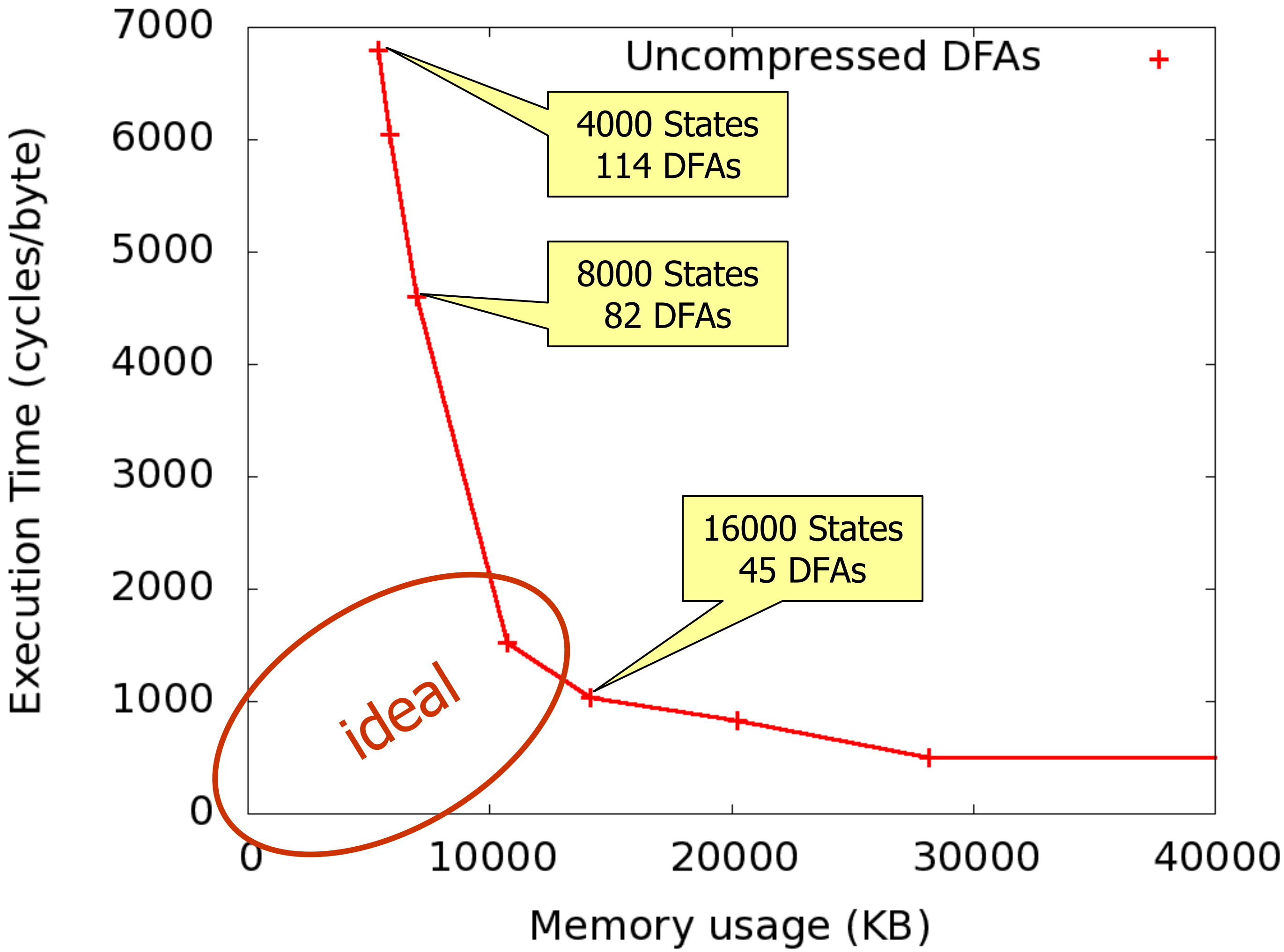
Outline

- Introduction
- Alphabet Compression Tables
- Interacting with D^2FAs
- Experimental Results

Experimental Setup

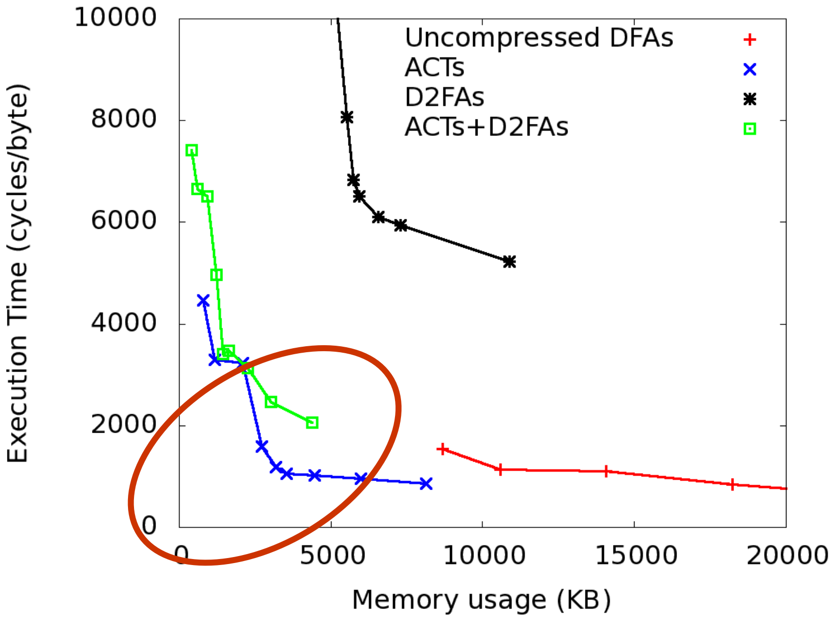
- 1550 HTTP, SMTP, FTP signatures
  - Grouped by protocol and rule set (Snort or Cisco)
- *DFA Set Splitting* (Yu, 2006) to cluster DFAs
  - Provide memory bound *a priori*
  - Heuristically combine into as few DFAs as possible
- Experiment Environment
  - 10 GB traces, run on 3.0 GHz P4
  - Exec time measured with cycle-accurate counters

Memory vs Time



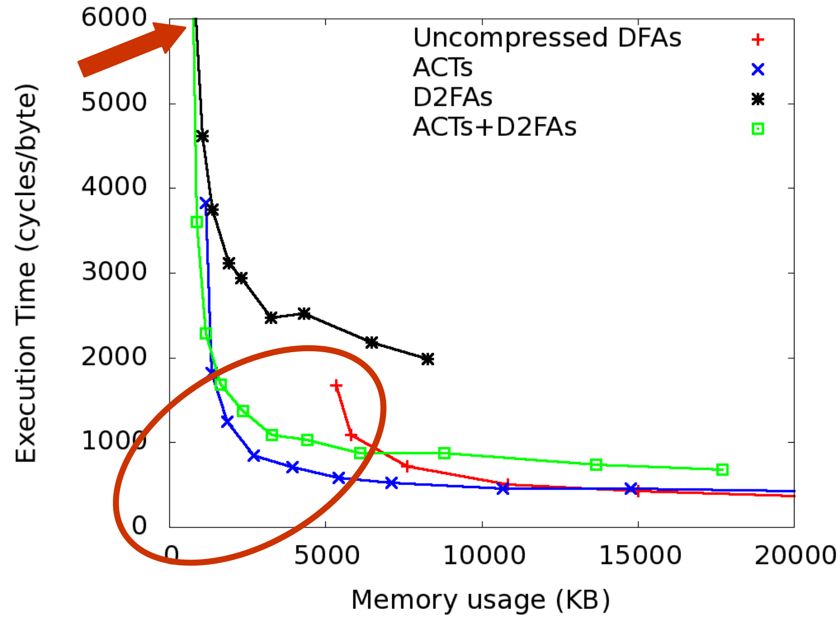
Memory vs Time

Snort HTTP



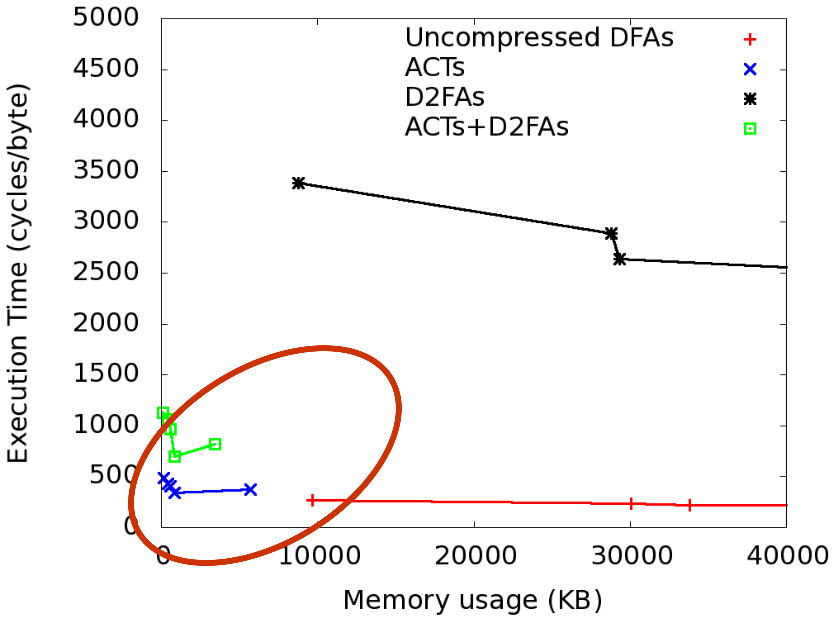
Cisco IPS HTTP

Lowest mem, highest exec!



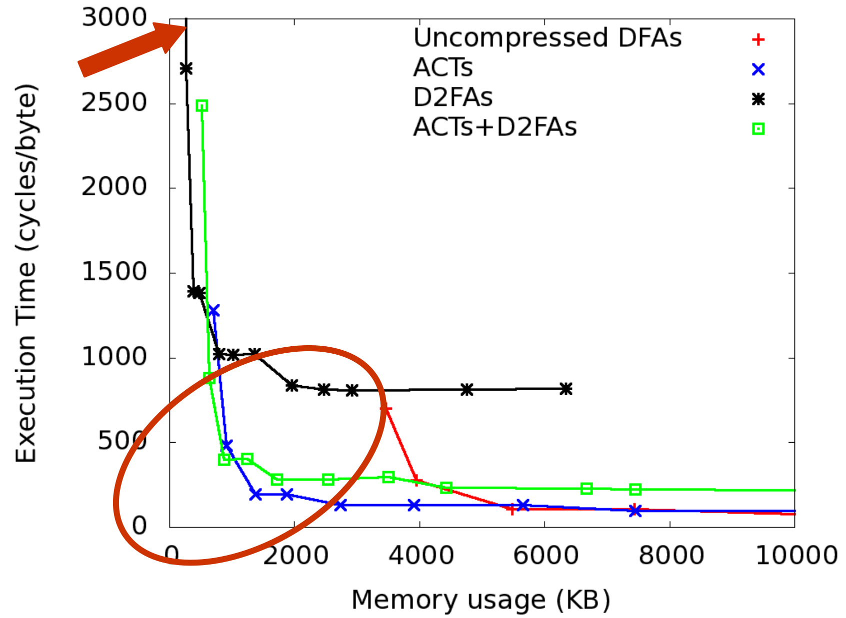
Memory vs Time

Snort SMTP



Cisco IPS SMTP

Lowest mem, highest exec!



Conclusion

- Multiple alphabet compression tables
  - Lightweight
  - Applicable to hardware or software platforms
  - Compatible with other techniques
- Provides better time vs. space performance
  - 4x to 70x memory reduction
  - 35% to 85% execution time increase
- Best technique a function of time, memory limits
  - ACTs add superior design points

Efficient Signature Matching with Multiple Alphabet Compression Tables

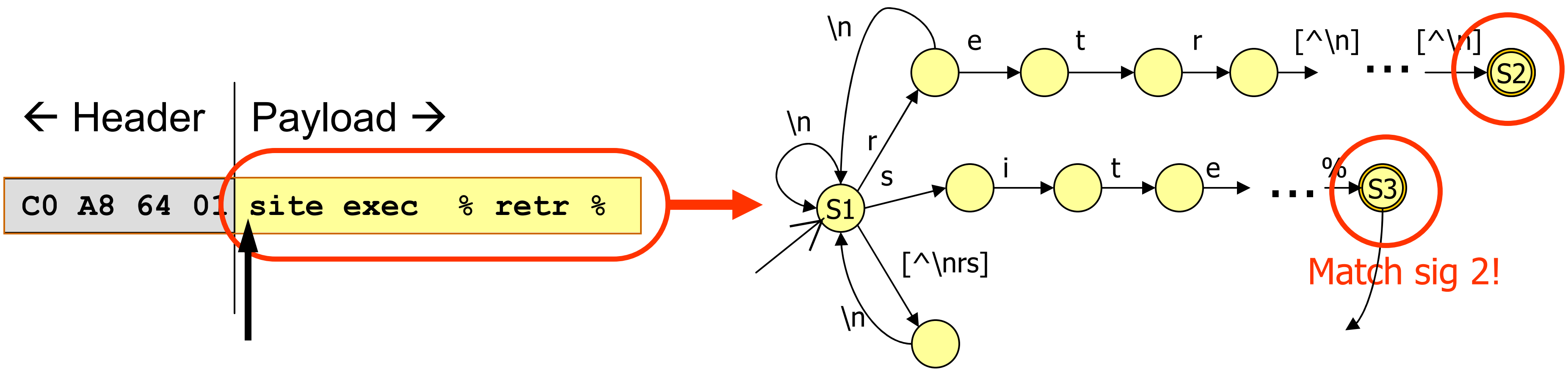
Thank you

intentionally blank

Regular Expressions and DFAs

- Regular expressions standard for *writing* sigs
  - Buffer overflow: `/^RETR\s[^\n]{100}/`
  - Format string attack: `/^SITE\s+EXEC[^\n]*%[^\n]*%/`

- DFAs used for *matching* to input



Memory Usage

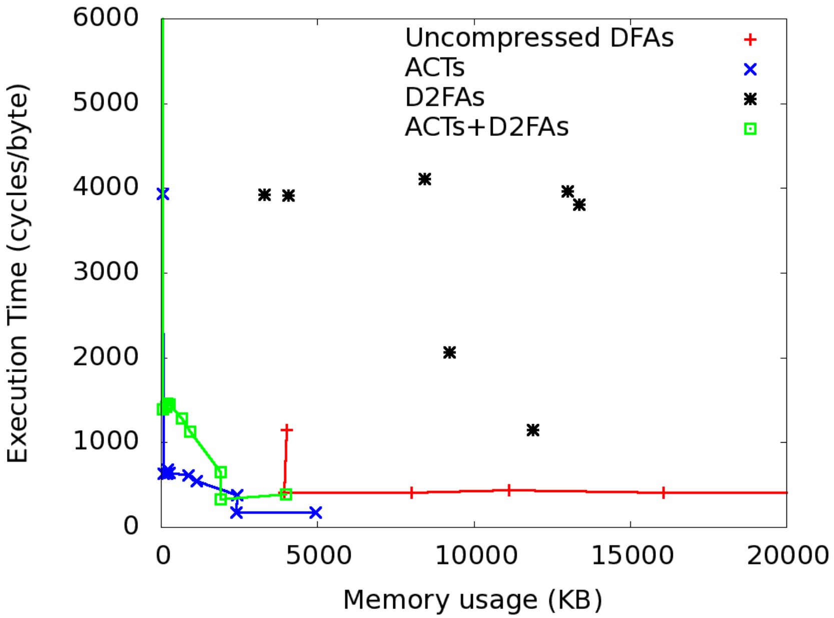
	DFA	ACT	D^2FA	ACT + D^2FA
Snort HTTP	74	8.1	8.8	**4.3**
Snort SMTP	98	5.7	42	**3.4**
Snort FTP	94	4.9	9.2	**3.9**

	DFA	ACT	D^2FA	ACT + D^2FA
Cisco HTTP	116	30	**4.7**	17
Cisco SMTP	110	29	**3.0**	18
Cisco FTP	83	5.1	**1.7**	1.9

All results reported in megabytes (MB)

Memory vs Time

Snort FTP



Cisco IPS FTP