Optimization of Pattern Matching Algorithm for Memory Based Architecture

Cheng-Hung Lin, Yu-Tang Tai, and Shih-Chieh Chang

National Tsing Hua University, Taiwan, R.O.C

Outline

- Memory architecture for string matching
- Basic idea
- Novel Algorithm for memory architecture
- Experimental results and conclusions

Introduction

- Network Intrusion Detection System is used to detect network attacks by identifying attack patterns.

- Software-only approaches can no longer meet the high throughput of today's networking

- Hardware approaches for acceleration.
  - Logic architecture
  - Memory architecture

Advantage of Memory Architecture

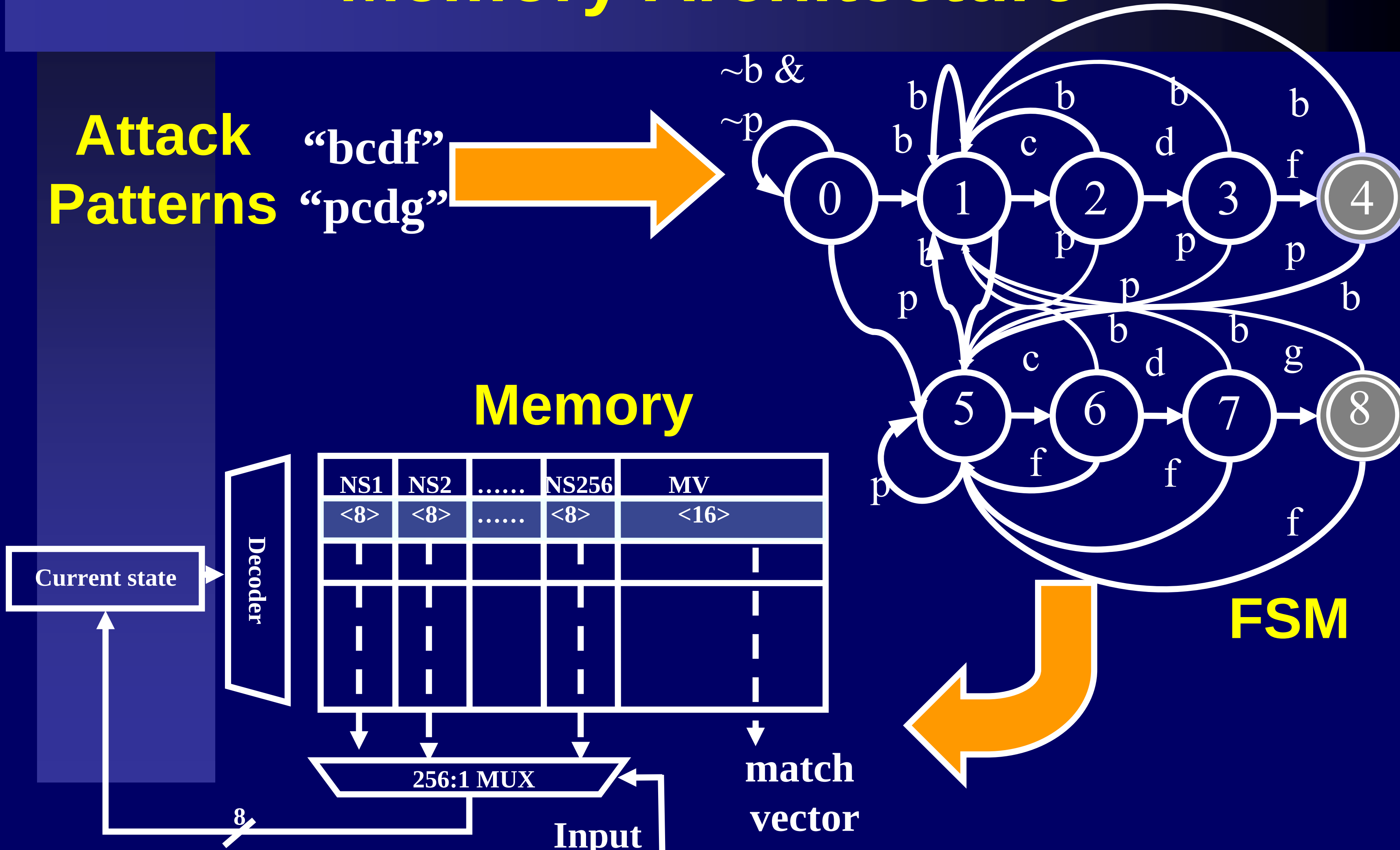
- The memory architecture has attracted a lot of attention because of its easy re-configurability and scalability.

✓Young H. Cho and William H. Mangione-Smith, “A Pattern Matching Co-processor for Network Security,” in *Proc. 42nd IEEE/ACM Design Automation Conference,* Anaheim, CA, June 13-17, 2005.

✓M. Aldwairi*, T. Conte, and P. Franzon. “Configurable String Matching Hardware for Speeding up Intrusion Detection,” in *Proc*. *ACM SIGARCH Computer Architecture News*, 33(1):99–107, 2005.

✓S. Dharmapurikar and J. Lockwood. “Fast and Scalable Pattern Matching for Content Filtering,” in *Proc. Symposium on Architectures for Networking and. Communications Systems (ANCS)*, Oct 2005.

Memory Architecture



Major Issue of Memory Architecture

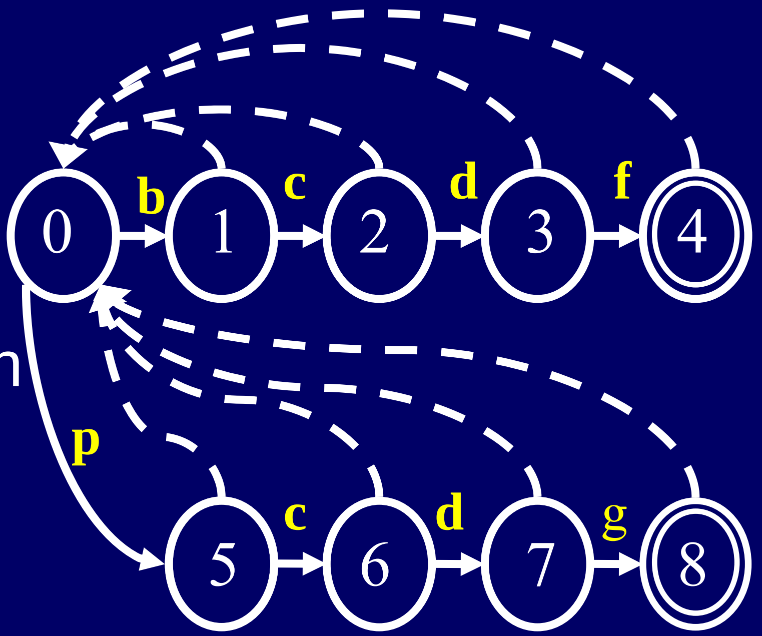
- Due to the increasing number of attacks, the required memory increases tremendously
  - The performance, cost, and power consumption are related to the memory size
  - Reducing the memory size has become imperative

Outline

- Memory architecture for string matching
- Basic idea
- Novel algorithm for memory architecture
- Experimental results and Conclusions

Review of Aho-Corasick Algorithm

- Aho-Corasick (AC) algorithm can reduce large number of state transitions and memory size.
  - Solid line represents valid transitions.
  - Dotted line represents failure transitions.
  - Introduce the failure transition to reduce the outgoing transitions.

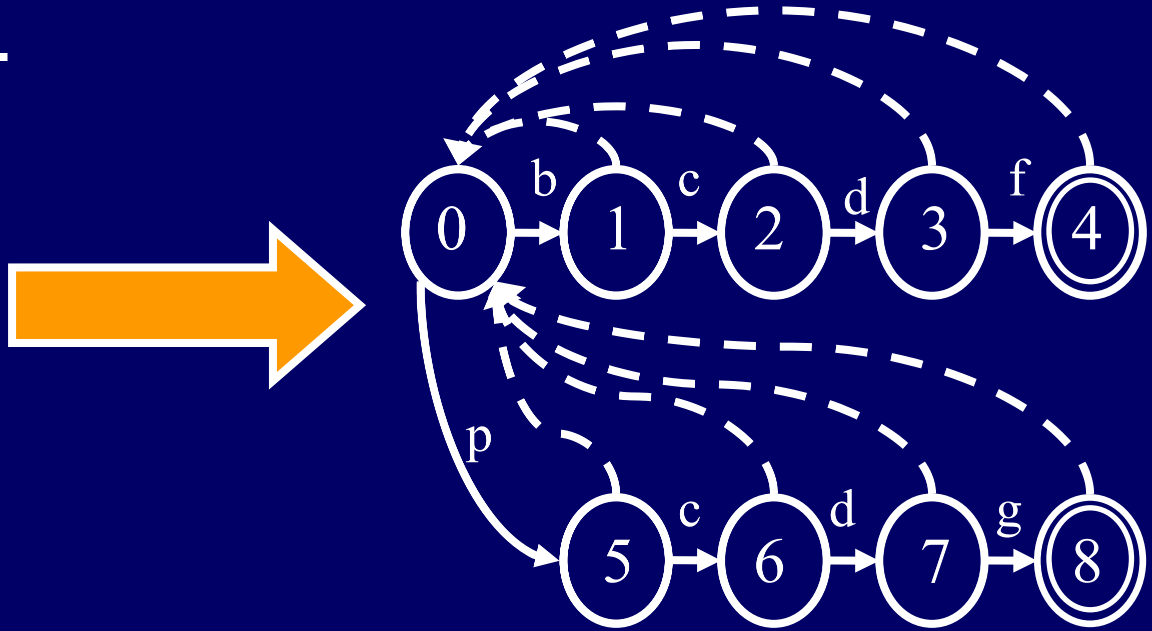


AC state machine of "bcdf"and "pcdg"

Observation

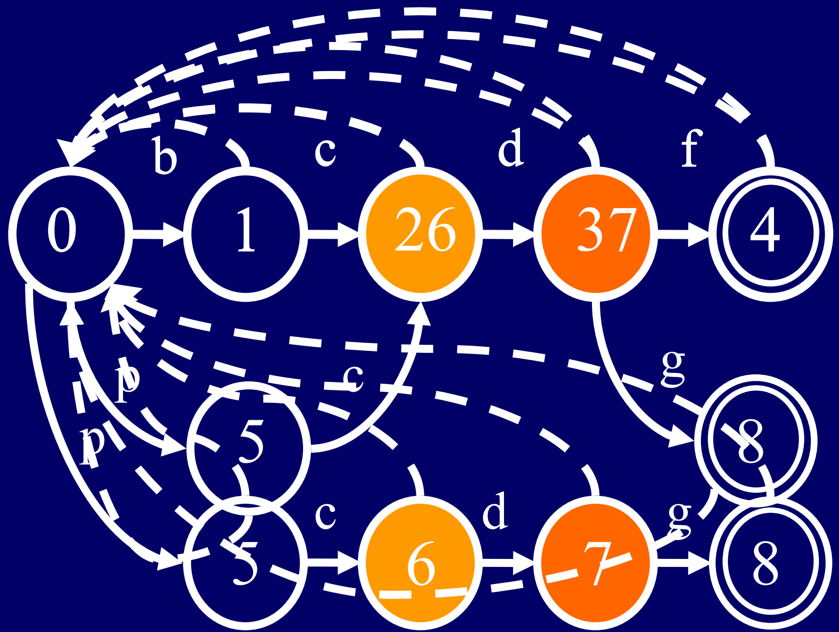
- Many string patterns are similar because of common sub-strings
- The similarity does not lead to a small state machine.

“bcdf”
“pcdg”



AC state machine

Merge Similar States



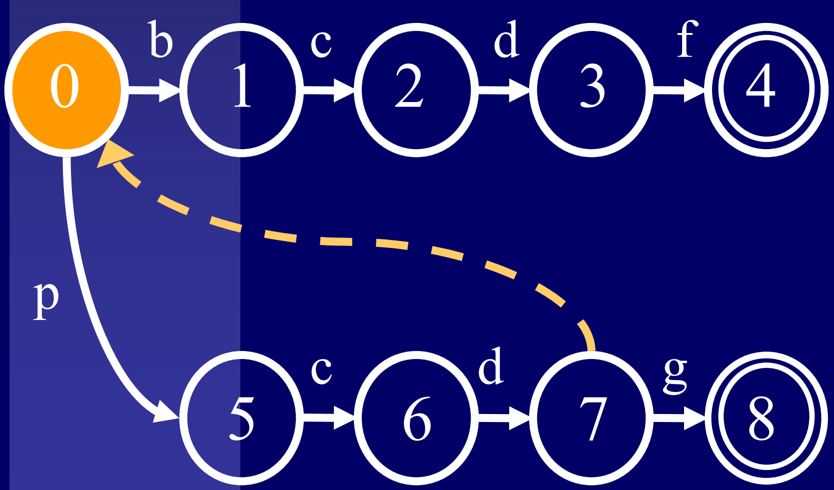
merg_FSM

- The merg_FSM is a different machine
  - smaller number of states and transitions.
  - smaller memory in memory architecture.

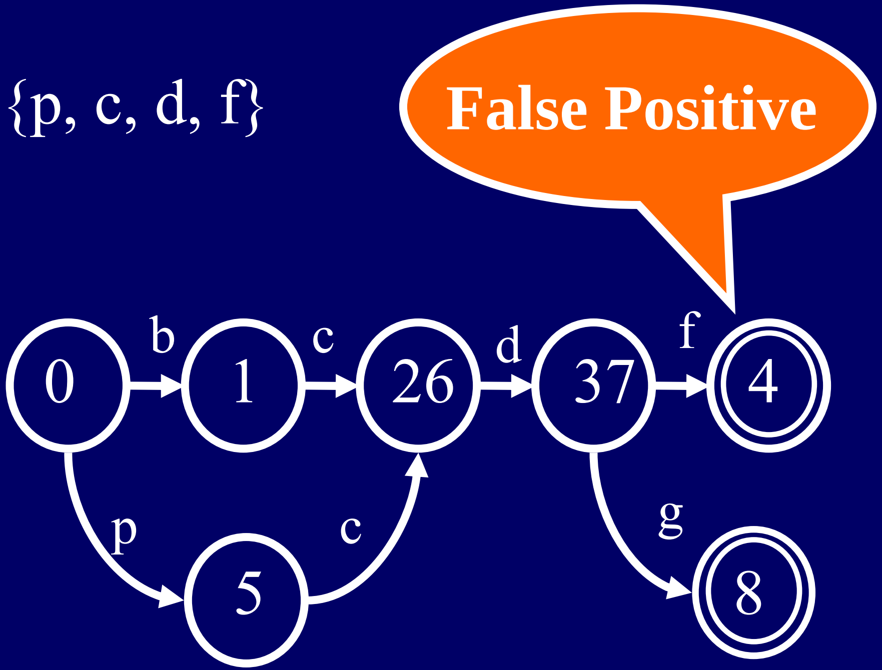
Problem of merg_FSM

- Directly merging similar states results in an erroneous state machine.

input stream = {p, c, d, f}



AC state machine



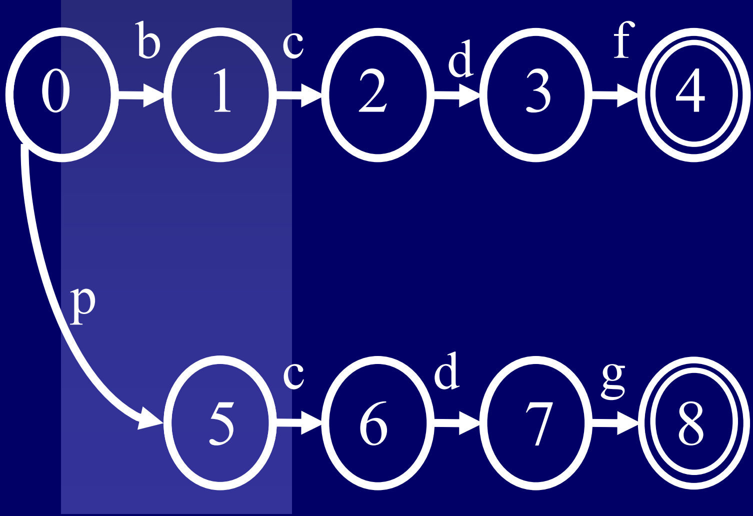
merg_FSM

Outline

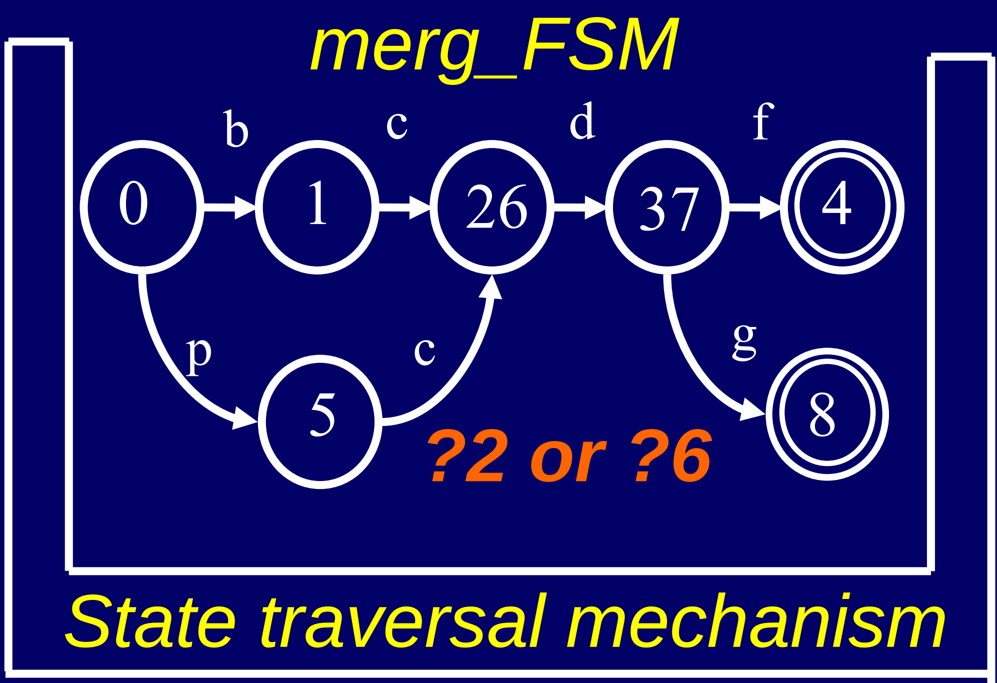
- Memory architecture for string matching
- Basic Idea
- Novel Algorithm for memory architecture
- Experimental results and Conclusions

State Traversal Mechanism

- Store *merg_FSM* table in memory
- *State traversal mechanism* is used to memorize the precedent state and differentiate merged states.



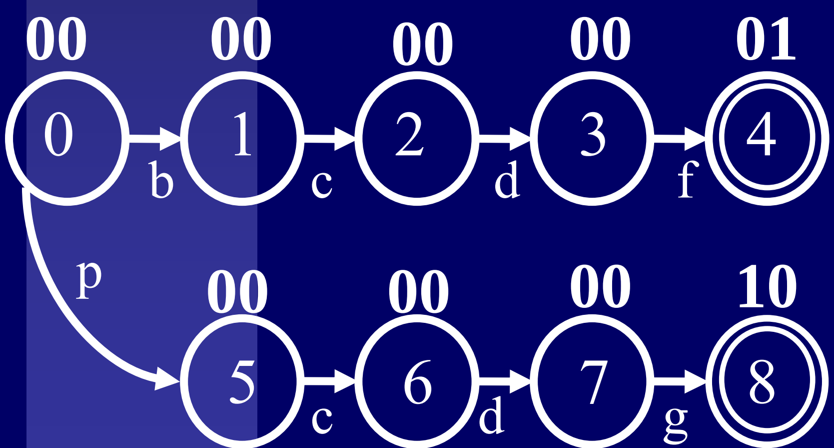
AC state machine



New State Information

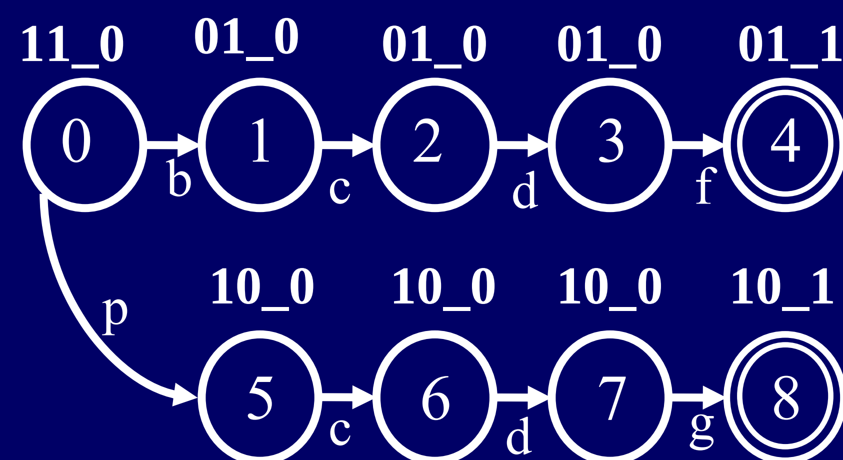
- AC state machine stores *match vector*.
- New state machine stores
  - *PathVec* stores path information.
  - *IfFinal* indicates whether the state is a final state.

match vector



AC State Machine

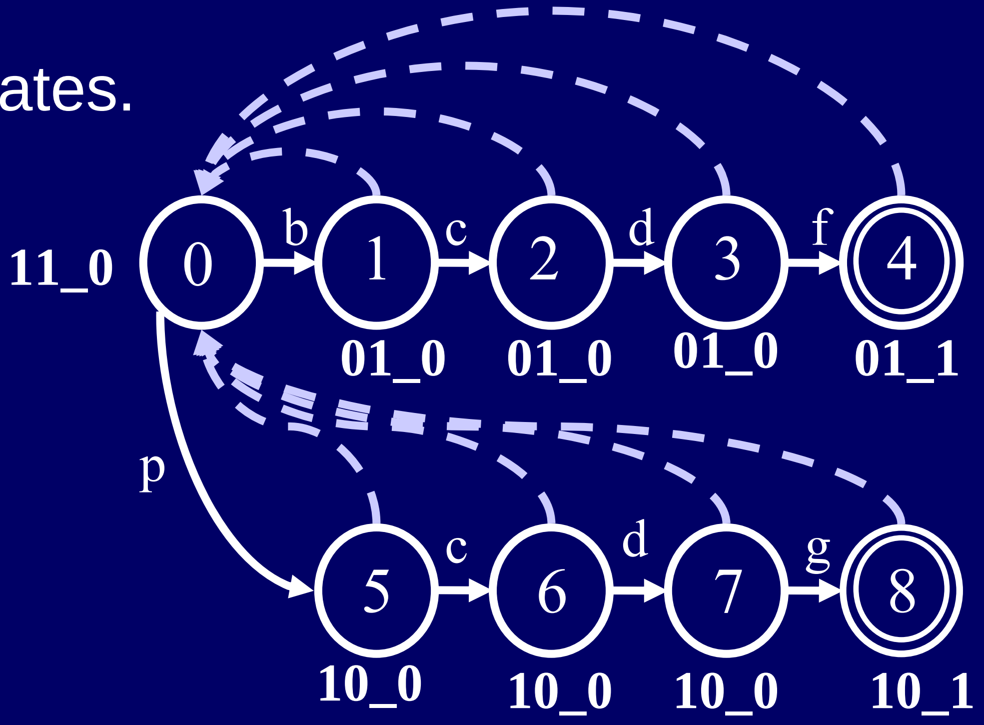
pathVec_ifFinal



New State Machine

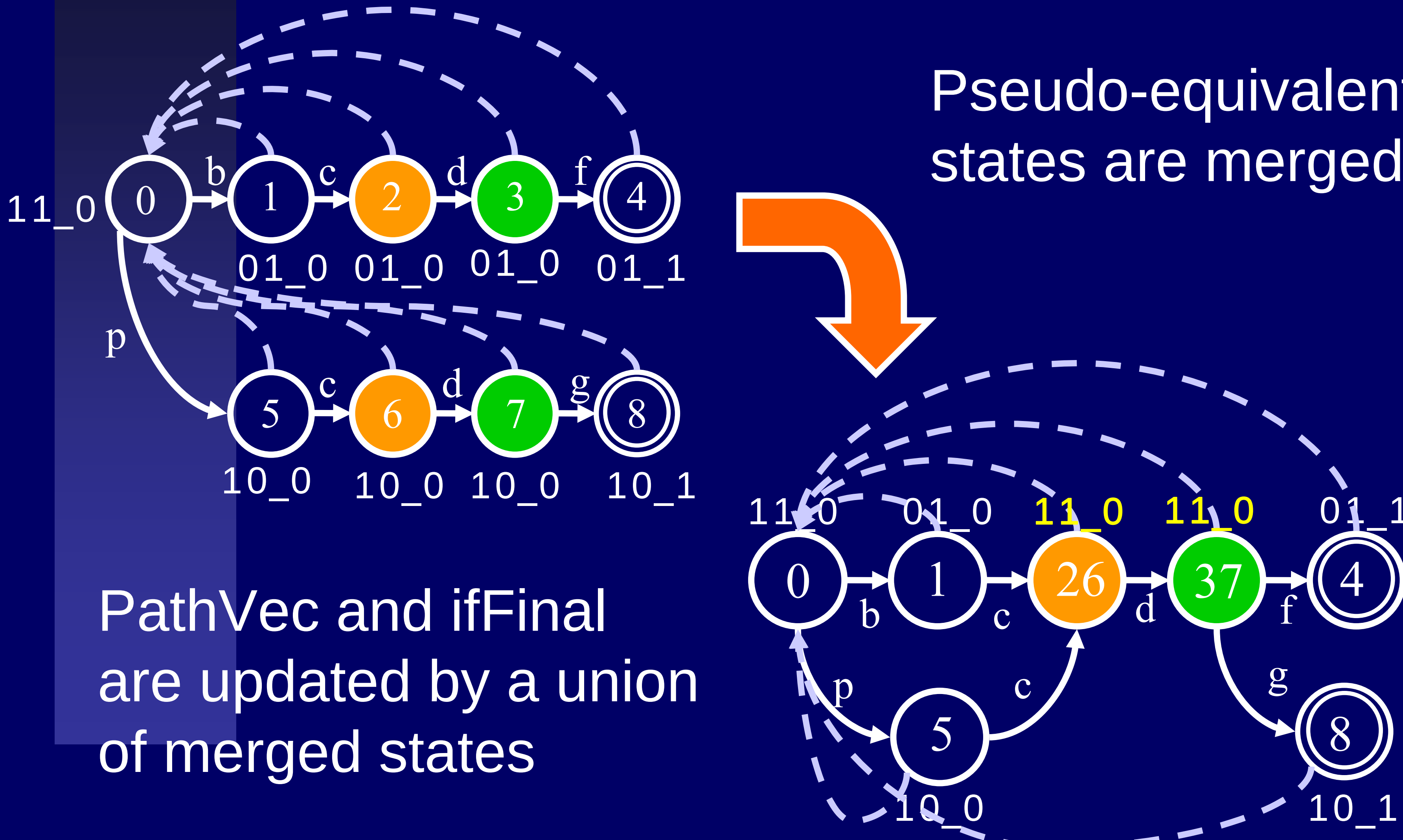
Pseudo-Equivalent States

- Definition: Two states are ***pseudo-equivalent*** if they have
  - identical input transitions
  - identical failure transitions
  - identical ifFinal
  - but different next states.



Merge Pseudo-Equivalent States

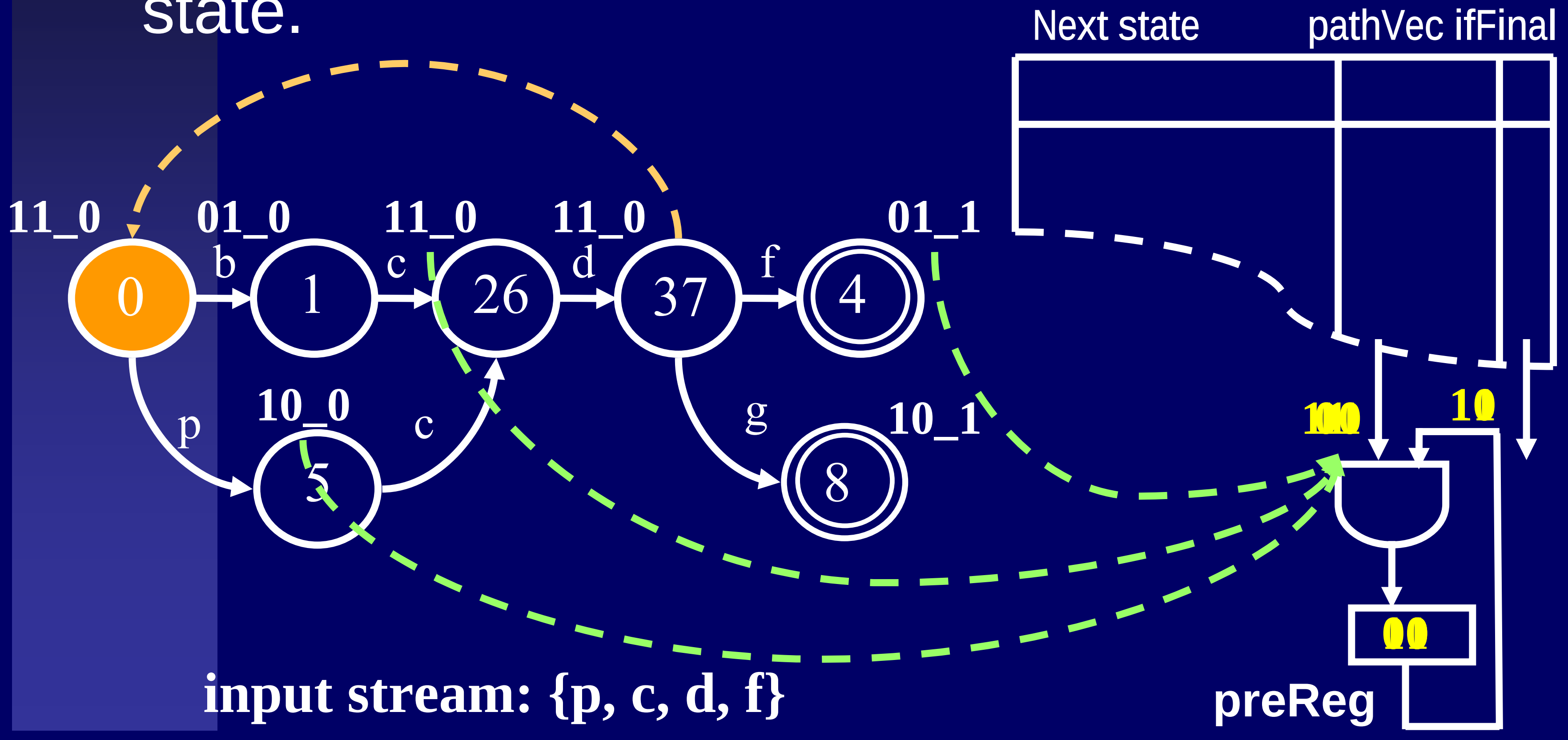
Pseudo-equivalent states are merged.



PathVec and ifFinal are updated by a union of merged states

State Traversal Mechanism

- PreReg traces the precedent pathVec in each state.



Outline

- Memory architecture for string matching
- Basic Idea
- Novel algorithm for memory architecture
- Experimental results and Conclusions

Experiment I

- Perform experiments on Snort rule sets.
- Compare our approach with the Aho-Corasick algorithm .

- A.V. Aho and M.J. Corasick. Efficient String Matching: An Aid to Bibliographic Search. In *Communications of the ACM 1975*.

Compare with Traditional AC

Rule Sets	# of patterns	# of char.	Tradition AC [24]			Our algorithm			
			# of trans.	# of states	Memory (bytes)	# of trans.	# of states	Memory (bytes)	Memory Reduct.
Oracle	**138**	**4,674**	**2,180**	**2,185**	**880,009**	**1,389**	**1,221**	**452,533**	**49%**
Sql	44	1,089	421	422	129,290	321	284	87,011	33%
Backdoor	57	599	563	565	191,253	523	497	152,268	20%
Web-iis	113	2,047	1,533	1,537	569,651	1,273	1,155	428,072	25%
Web-php	115	2,455	1,670	1,675	620,797	1,295	1,142	423,254	32%
Web-misc	310	4,711	3,576	3,587	1,444,664	3,031	2,734	1,101,119	24%
Web-cgi	347	5,339	3,407	3,419	1,377,002	2,672	2,358	949,685	31%
					⋮				
Total rules	1,595	20,921	17,472	17,522	8,745,668	14,704	13,381	6,248,927	29%
Ratio			**1**	**1**	**1**	**84%**	**76%**	**71%**	**29%**

Experiment II

- Enhance the bit-split algorithm with our method
  - The results are compared with the original bit-split algorithm.

- L. Tan and T. Sherwood. A high throughput string matching architecture for intrusion detection and prevention. In *ISCA'05*.

Compare with Traditional Bit-Split

Rule Sets	# of patterns	# of char.	Bit-split [8]			Bit-split + Our algorithm			
			# of trans.	# of states	Memory (bytes)	# of trans.	# of states	Memory (bytes)	Memory Reduct.
Oracle	138	4,674	6,645	6,665	633,175	4,146	3,603	358,499	43%
Sql	44	1,089	1,211	1,215	110,565	866	769	72,671	34%
Backdoor	57	599	1,697	1,705	155,155	1,441	1,305	126,585	18%
Web-iis	113	2,047	4,869	4,885	464,075	3,844	3,374	335,713	28%
Web-php	115	2,455	4,991	5,011	476,045	3,871	3,345	332,828	30%
Web-misc	310	4,711	10,959	11,003	1,067,291	8,861	7,816	797,232	25%
Web-cgi	347	5,339	9,901	9,949	965,053	7,875	6,957	709,614	26%
⋮									
Total ruls	1,595	20,921	53,930	54,130	5,467,130	43,550	38,701	4,237,760	22%
Ratio			**1**	**1**	**1**	**81%**	**71%**	**78%**	**22%**

Conclusion

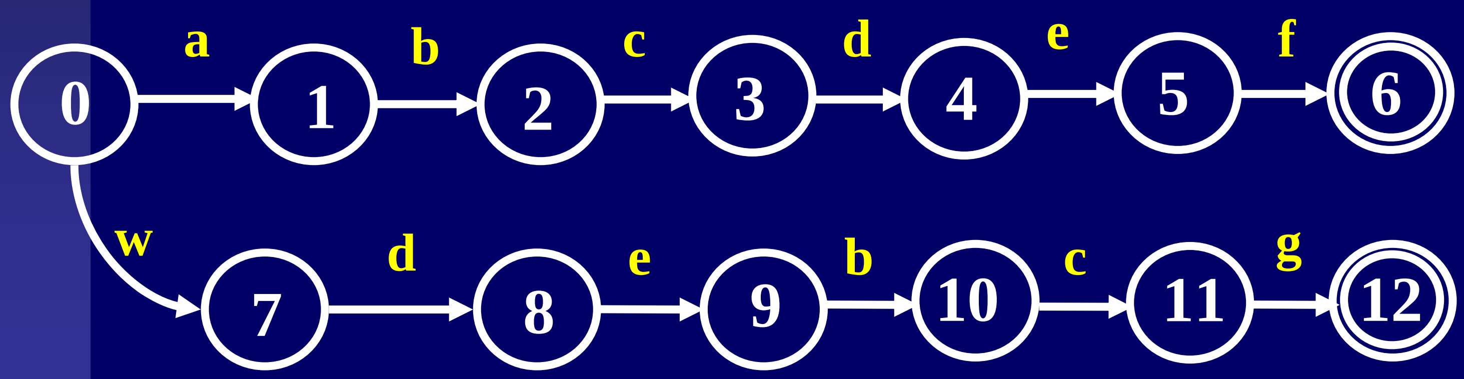
- Provide a concept of merging pseudo-equivalent states to reduce the number of states and transitions.
- Propose a state traversal mechanism working with the merg_FSM without false positive matching results.
- Experimental results demonstrate a significant reduction in memory requirement.

Thank You!

Backup

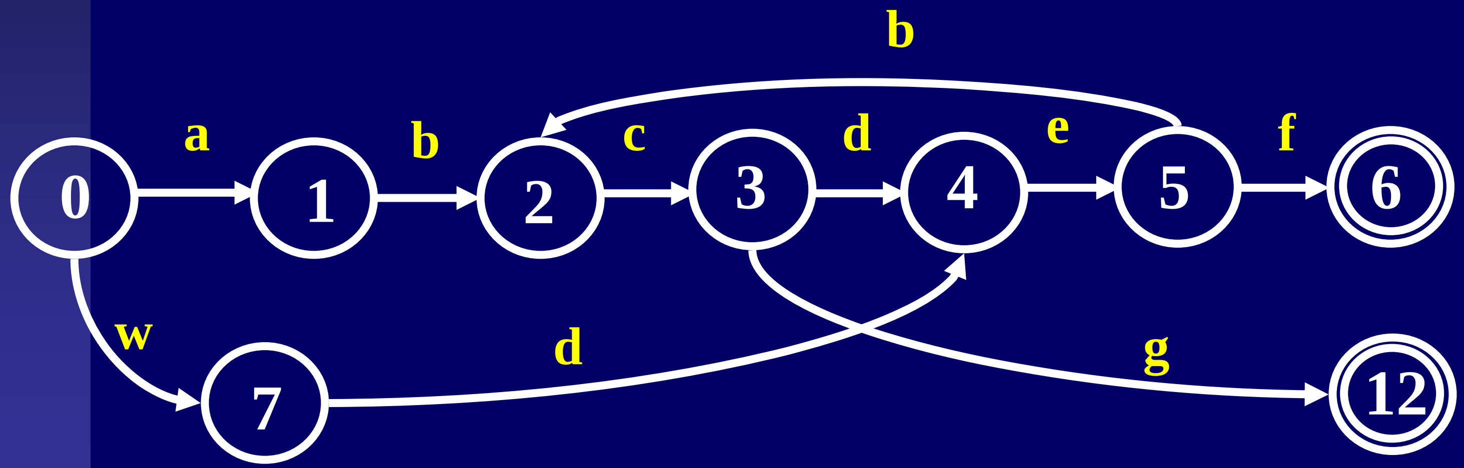
Cycle Problem

- Merging disorder sections of pseudo-equivalent states creates cycle problem.



Cycle Problem

- For example, the input string “abcdebcdef” will be mistaken as a match of the pattern “abcdef.”



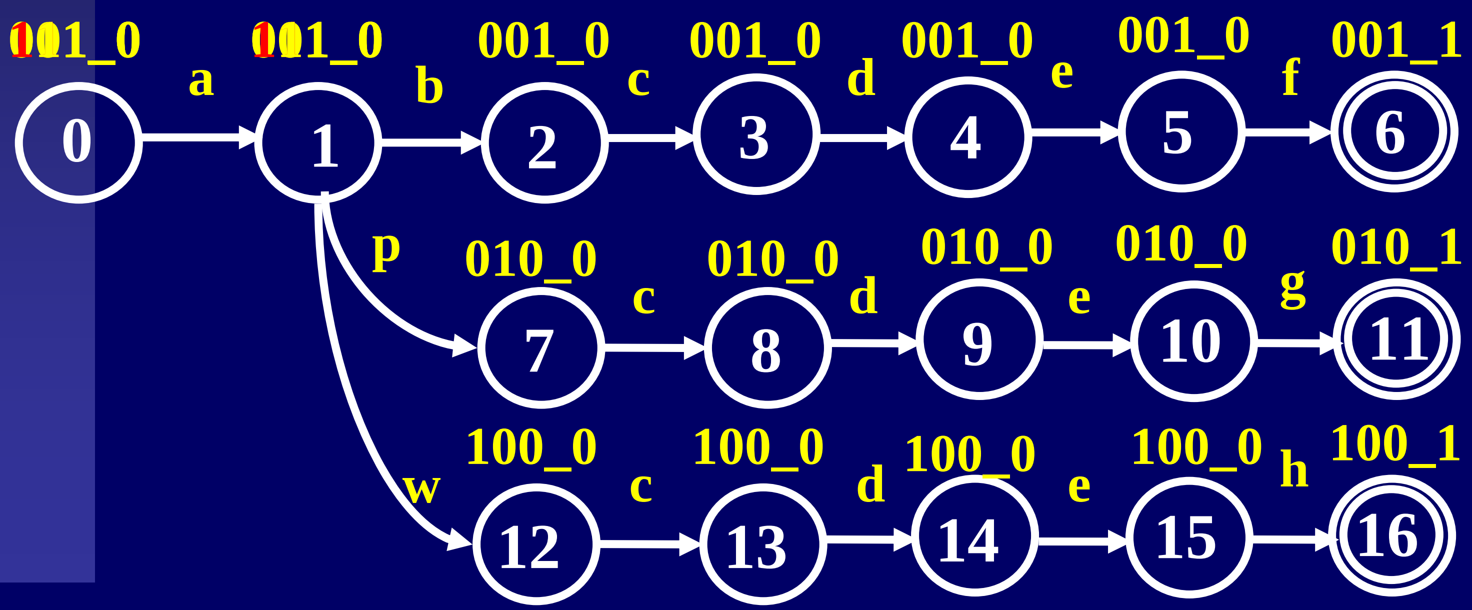
Construction of State Traversal Machine

- Construction of the state traversal machine consists of two steps
  - Step1: Construct valid transitions, failure transitions, pathVec, and ifFinal function.
  - Step2: Merge the pseudo-equivalent states.

Example

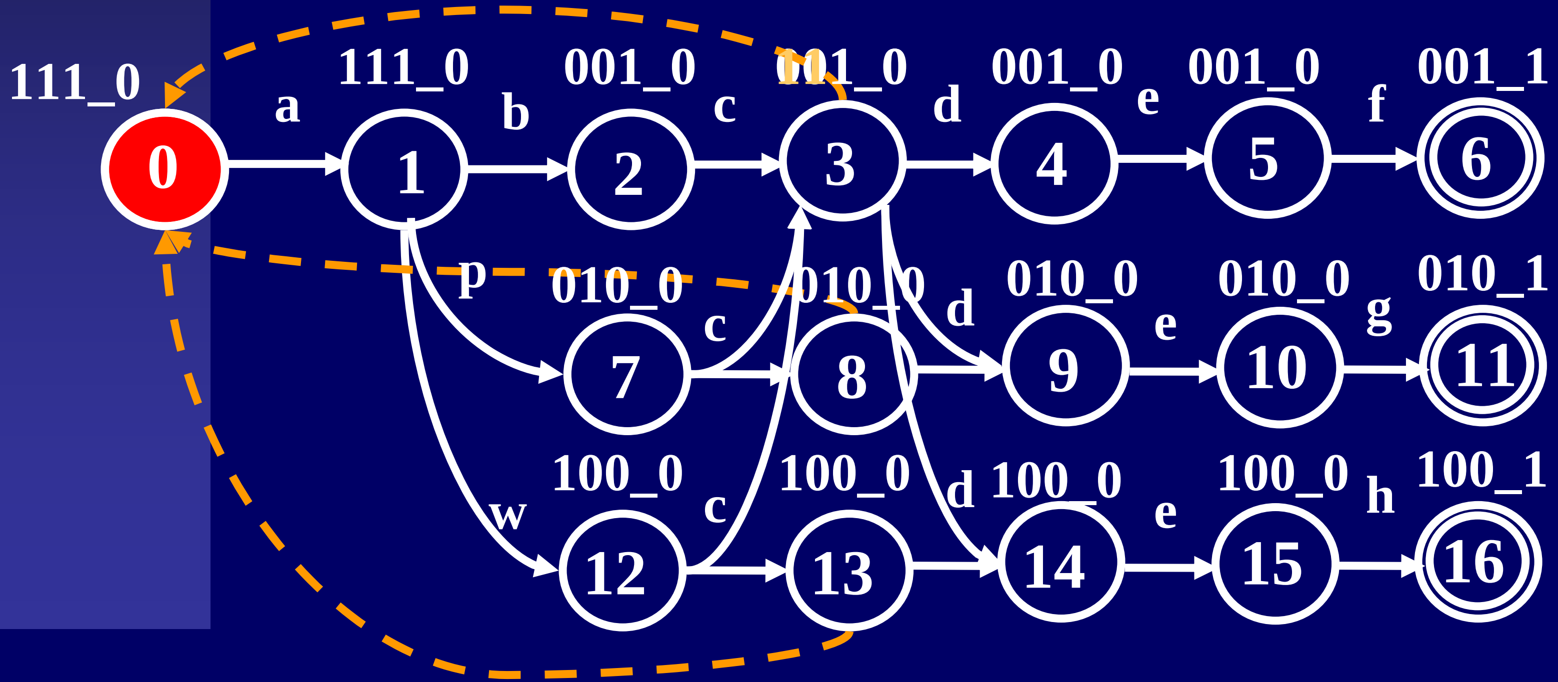
- Consider three patterns “abcdef”, “apcdeg”, “awcdeh”.

16 states

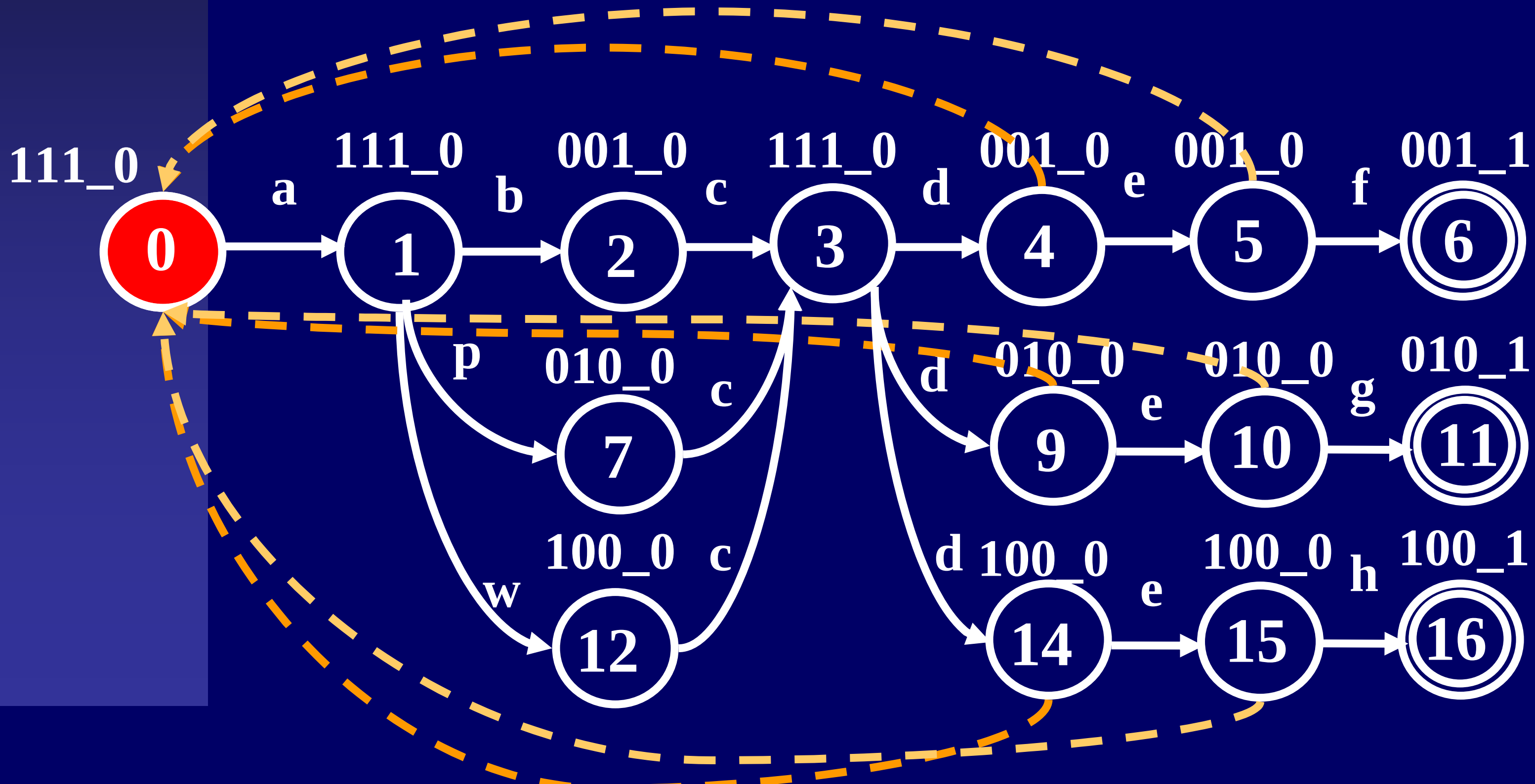


Merging Pseudo-equivalent States

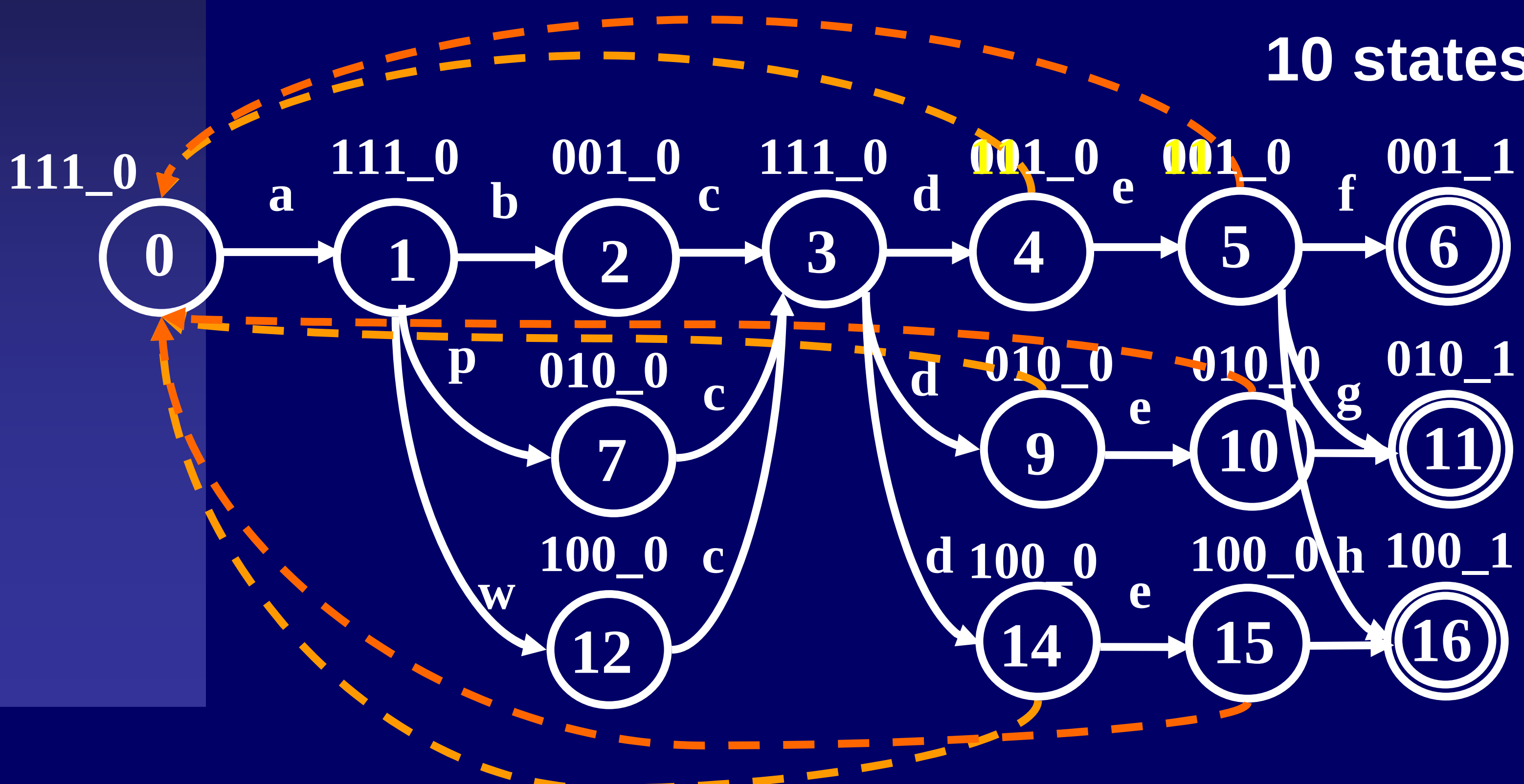
- merging the failure transitions
- performing the union on the pathVec of the merged states



Merging Pseudo-equivalent States



Merging Pseudo-equivalent States



State Traversal Algorithm

Algorithm: State traversal pattern matching algorithm

Input: A text string x=*a*1*a*2…*a*n where each *a*i is an input symbol and a state traversal machine *M* with valid transition function *g*, failure transition function *f*, path function *pathVec* and final function *ifFinal*.

Output: Locations at which keywords occur in x.

Method:

```
begin
    state←0
    preReg←1….1   //all bits are initiated to 1.
    for i←until n do
      begin
        preReg = preReg & pathVec(state)
        while g(state, ai) == fail || preReg == 0 do
          begin
              state←f (state)
              preReg←1….1
          end

        state←g(state, ai)
        if ifFinal(state) = 1 then
          begin
              print i
              print preReg
          end
      end
end
```