MINIMIZATION OF DFA

Dr. A. Beulah

AP/CSE

LEARNING OBJECTIVE

- To construct finite automata for any given pattern and find its equivalent regular expressions
  - To understand what is Regular Expression

MYHILL NERODE ALGORITHM

- Initialize entry for each pair in table to “unmarked”.
- Mark (p,q) if $p \in F$ and $q \notin F$ or vice-versa.
- Scan table entries and repeat till no more marks can be added:

  If there exists unmarked (p,q) with $a \in \Sigma$ such that $\delta(p,a)$ and $\delta(q,a)$ are marked, then mark (p,q).
- Return as: $p \approx q$ iff (p,q) is left unmarked in table.

MYHILL NERODE ALGORITHM

begin

for p in F and q in $Q–F$ **do** mark (p, q);

for each pair of distinct states (p, q) in $F \times F$ or $(Q–F) \times (Q–F)$ **do**

if for some input symbol a, $(\delta(p, a), \delta(q, a))$ is marked **then**

begin

mark (p, q);

recursively mark all unmarked pairs on the list for (p, q) and on the lists of other pairs that are marked at this step.

end

else /* no pair $(\delta(p, a), \delta(q, a))$ is marked */ **for** all input symbols a **do**

put (p, q) on the list for $(\delta(p, a), \delta(q, a))$ unless $\delta(p, a) = \delta(q, a)$

end

EXAMPLE

δ	a	b
→A	B	C
B	B	D
C	B	C
*D	B	C

F: (D) 1

NF: (A B C)

X

→A	=			
B	X	=		
C	=	X	=	
*D	X	X	X	=
	A	B	C	D

A,B: δ(A,a), δ(B,a) = B,B
δ(A,b), δ(B,b) = C,D X

A,C: δ(A,a), δ(C,a) = B,B
δ(A,b), δ(C,b) = C,C =

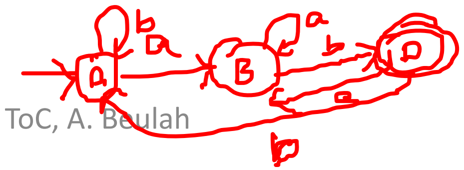
B,C: δ(B,a), δ(C,a) = B,B
δ(B,b), δ(C,b) = D,C X

(AC) (B) (D)*

A B D

c

δ	a	b
→A	B	A
B	B	D
*D	B	A



EXAMPLE

- (a/b)*abb

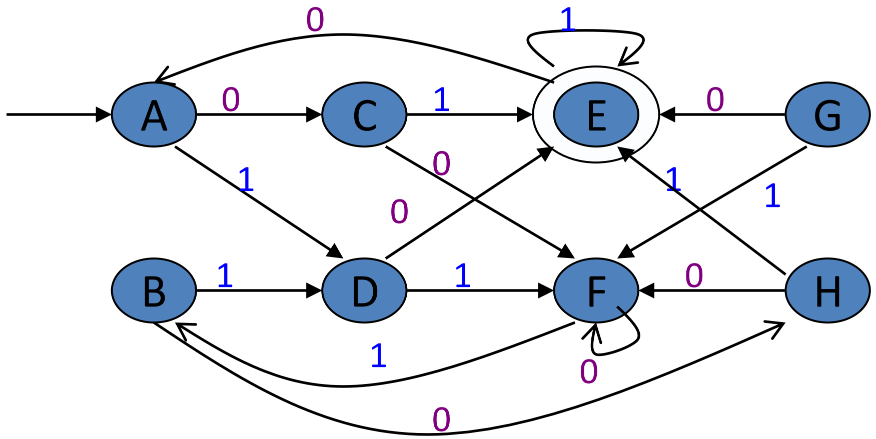
δ	a	b
→A	B	C
B	B	D
C	B	C
D	B	E
*E	B	C

δ	a	b
A	B	A
B	B	D
D	B	E
E	B	A

ABCD (E)

(AC) (B) (D) (E*)

EXAMPLE



	A	B	C	D	E	F	G	H
A	=							
B	=	=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F	X	X	X	X	X	=		
G	X	X	X	=	X	X	=	
H	X	X	=	X	X	X	X	=

<u>Pass #0</u>

1. Mark accepting states ≠ non-accepting states

<u>Pass #1</u>

1. Compare every pair of states
2. Distinguish by one symbol transition
3. Mark = or ≠ or blank(tbd)

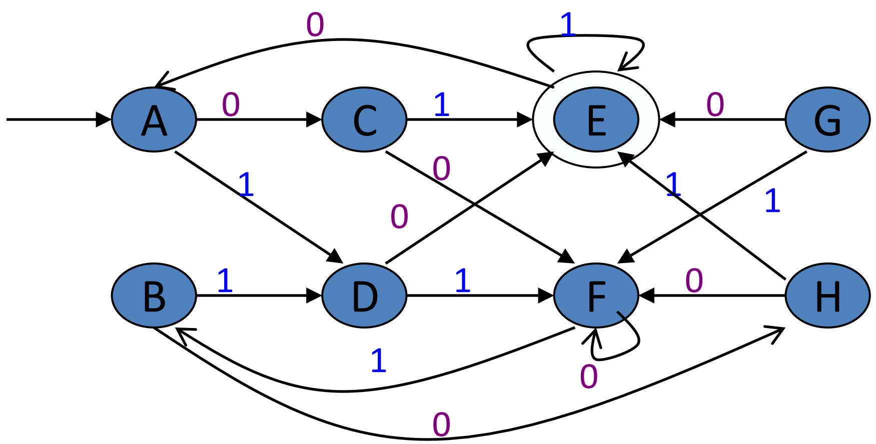
<u>Pass #2</u>

1. Compare every pair of states
2. Distinguish by up to two symbol transitions (until different or same or tbd)

....

(keep repeating until table complete)

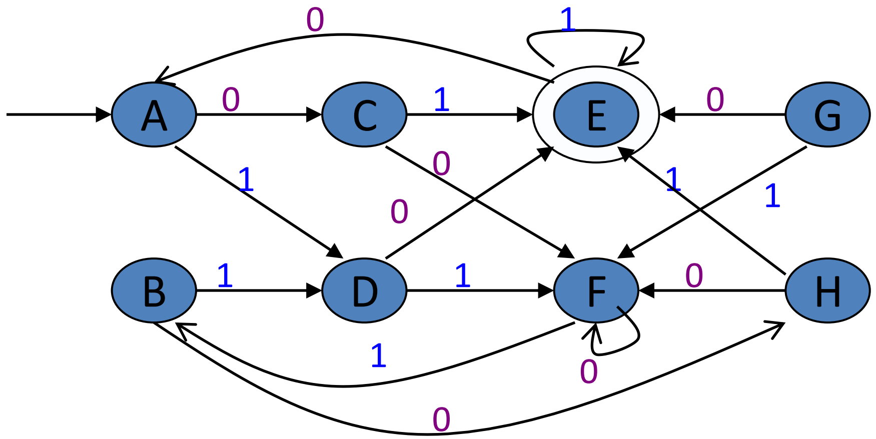
EXAMPLE



1. Mark X between accepting vs. non-accepting state

A	=							
B		=						
C			=					
D				=				
E	X	X	X	X	=			
F					X	=		
G					X		=	
H					X			=
	A	B	C	D	E	F	G	H

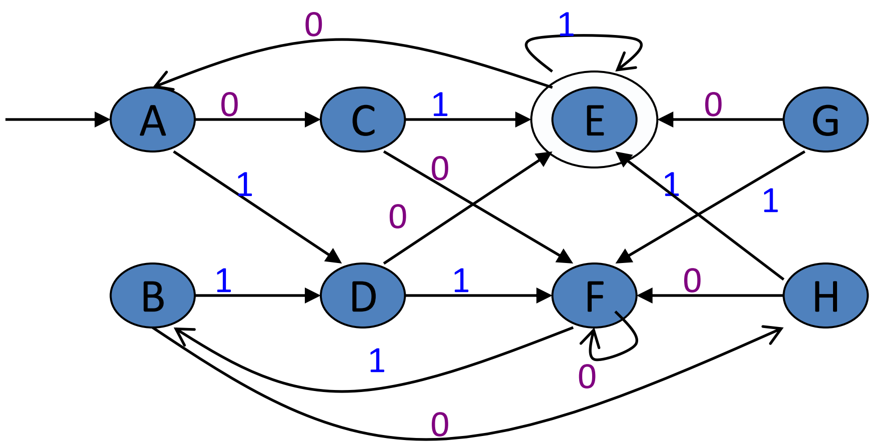
EXAMPLE



1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings

A	=							
B		=						
C	X		=					
D	X			=				
E	X	X	X	X	=			
F					X	=		
G	X				X		=	
H	X				X			=
	A	B	C	D	E	F	G	H

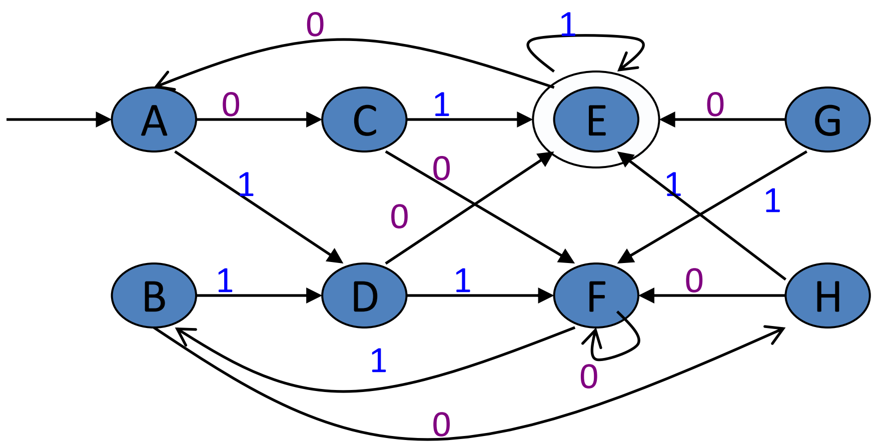
EXAMPLE



	A	B	C	D	E	F	G	H
A	=							
B		=						
C	X	X	=					
D	X	X		=				
E	X	X	X	X	=			
F					X	=		
G	X	X			X		=	
H	X	X			X			=

1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings

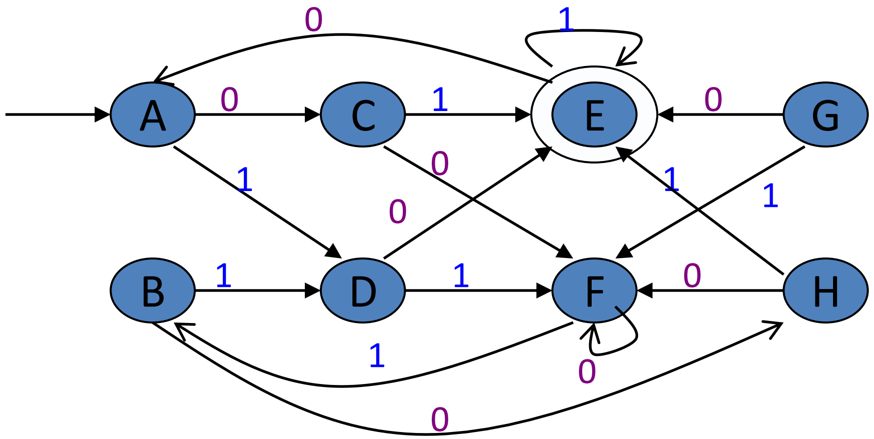
EXAMPLE



	A	B	C	D	E	F	G	H
A	=							
B		=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F			X		X	=		
G	X	X	X		X		=	
H	X	X	=		X			=

1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings

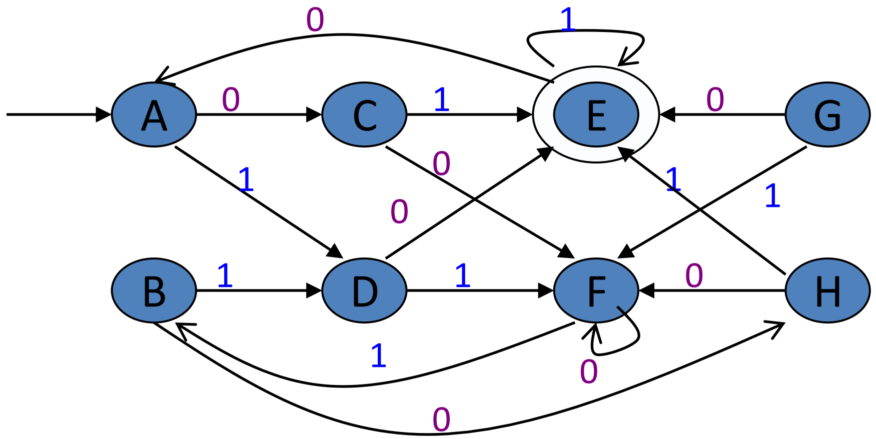
EXAMPLE



1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings

A	=							
B		=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F			X	X	X	=		
G	X	X	X	=	X		=	
H	X	X	=	X	X			=
	A	B	C	D	E	F	G	H

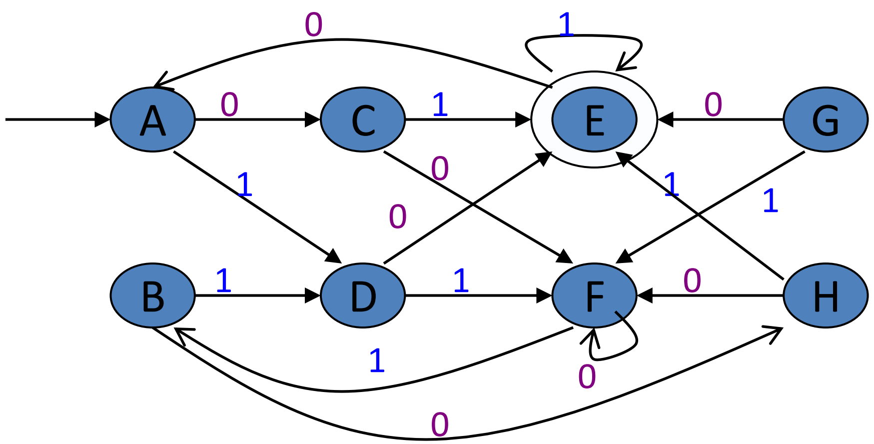
EXAMPLE



	A	B	C	D	E	F	G	H
A	=							
B		=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F			X	X	X	=		
G	X	X	X	=	X	X	=	
H	X	X	=	X	X	X		=

1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings

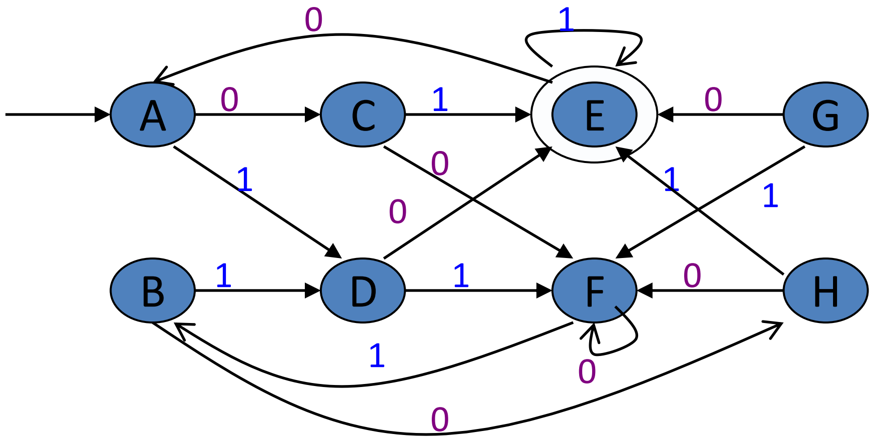
EXAMPLE



1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings

A	=							
B		=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F			X	X	X	=		
G	X	X	X	=	X	X	=	
H	X	X	=	X	X	X	X	=
	A	B	C	D	E	F	G	H

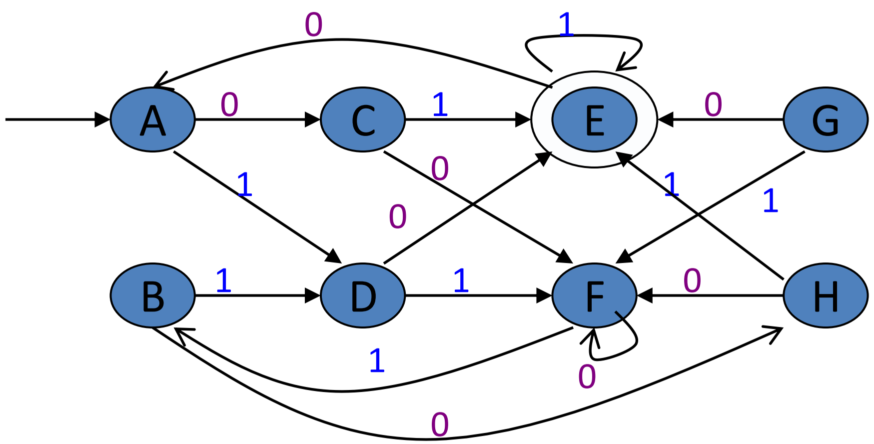
EXAMPLE



	A	B	C	D	E	F	G	H
A	=							
B	=	=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F	X	X	X	X	X	=		
G	X	X	X	=	X	X	=	
H	X	X	=	X	X	X	X	=

1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings
3. Look 2-hops away for distinguishing states or strings

EXAMPLE

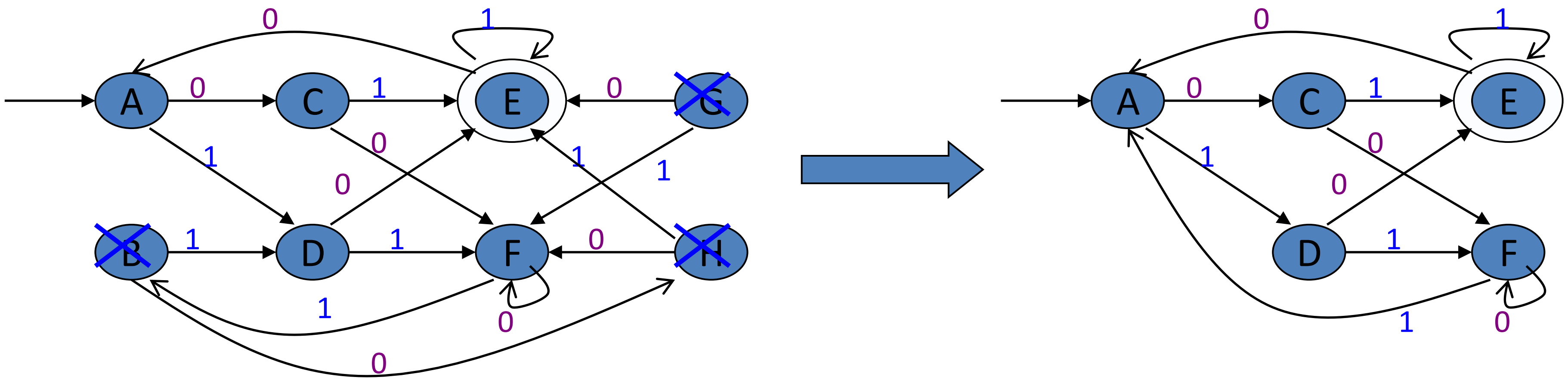


1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings
3. Look 2-hops away for distinguishing states or strings

A	=							
B	=	=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F	X	X	X	X	X	=		
G	X	X	X	=	X	X	=	
H	X	X	=	X	X	X	X	=
	A	B	C	D	E	F	G	H

Equivalences:
- A=B
- C=H
- D=G

EXAMPLE

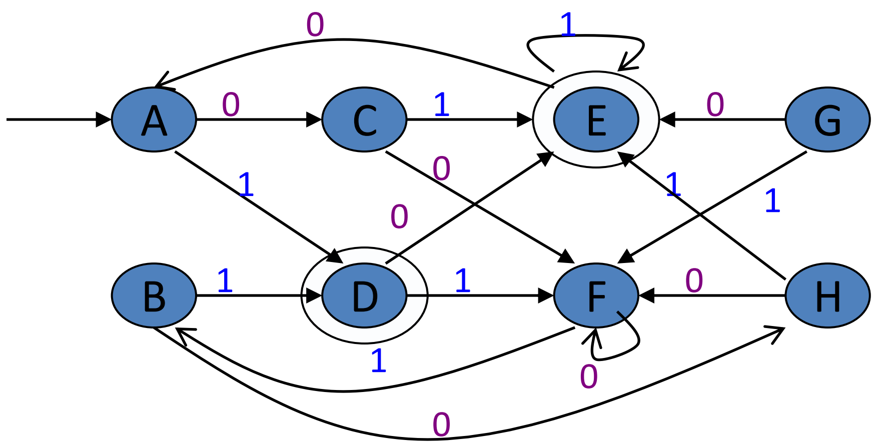


Retrain only one copy for each equivalence set of states

<u>Equivalences:</u>

- A=B
- C=H
- D=G

EXAMPLE



	A	B	C	D	E	F	G	H
A	=							
B		=						
C			=					
D				=				
E				?	=			
F						=		
G							=	
H								=

Q) What happens if the input DFA has more than one final state?
Can all final states initially be treated as equivalent to one another?

CONSTRUCTION OF Π_{FINAL} FROM Π

Algorithm: Minimizing the number of states of a DFA

- Input. A DFA M with set of states S, set of inputs $\sum$, transitions defined for all states and inputs, start state s_0, and a set of accepting states F.
- Output. A DFA $M^{'}$ accepting the same language as M and having as few states as possible.
- Method.
    1. Construct an initial partition $\prod$ of the set of states with two groups: the accepting states F and non-accepting states $S - F$.
    2. Partition $\prod$ to $\prod_{new}$.
    3. If $\prod_{new} = \prod$, let $\prod_{final} = \prod$ and go to step (4). Otherwise, repeat step (2) with $\prod := \prod_{new}$.
    4. Choose one state in each group of the partition $\prod_{final}$ as the *representative* for that group.
    5. Remove dead states.

CONSTRUCTION OF Π_{FINAL} FROM Π

```
for each group G of Π do begin
        partition G into subgroups such that two states s and t
            of G are in the same subgroup if and only if for all
            input symbols a, states s and t have transitions on a
            to states in the same group of Π;
        /* at worst, a state will be in a subgroup by itself */
        replace G in Π_new by the set of all subgroups formed
end
```

SUMMARY

- Procedure to minimize a DFA using Myhill – Nerode algorithm

LEARNING OUTCOME

On successful completion of this topic, the student will be able to:

- Understand the concepts of DFA minimization (K3)

TEST YOUR KNOWLEDGE

- Are the given two patterns equivalent?

  (1) gray|grey

  (2) gr(a|e)y

- Conversion of a regular expression into its corresponding NFA :

  a) Thompson's Construction Algorithm

  b) Powerset Construction

  c) Kleene's algorithm

  d) None of the mentioned

REFERENCE

- Hopcroft J.E., Motwani R. and Ullman J.D, “Introduction to Automata Theory, Languages and Computations”, Second Edition, Pearson Education, 2008

REFERENCE

- Hopcroft J.E., Motwani R. and Ullman J.D, “Introduction to Automata Theory, Languages and Computations”, Second Edition, Pearson Education, 2008