Minimization of DFA

Beulah A.
AP/CSE

Myhill Nerode Theorem

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);

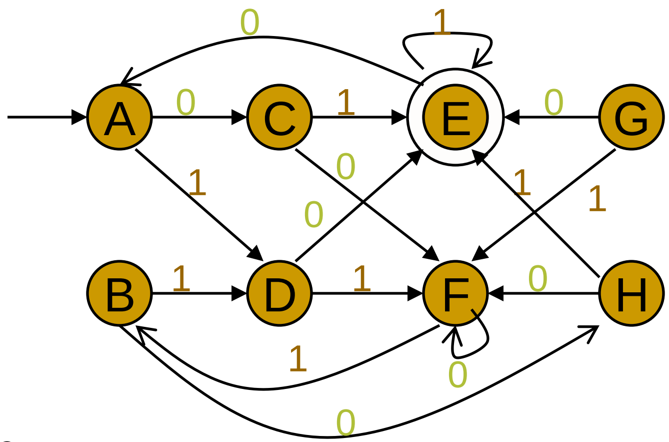
recursively mark all unmarked pairs on the list for (p, q) and on the lists of other pairs that are marked at this step.

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



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

<u>Pass #0</u>

1. Mark accepting states ≠ non-accepting states

<u>Pass #1</u>

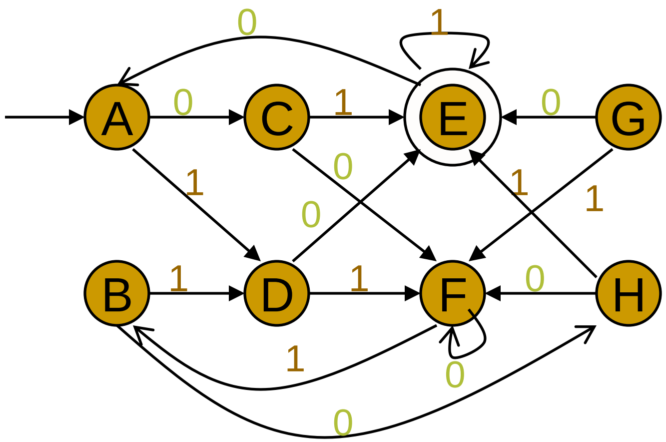
1. Compare every pair of states
2. Distinguish by one symbol transition
3. Mark = or ≠ or blank(tbd)

<u>Pass #2</u>

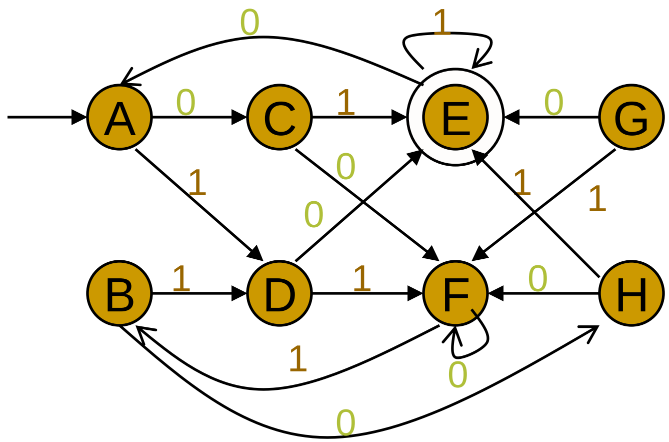
1. Compare every pair of states
2. Distinguish by up to two symbol transitions (until different or same or tbd)

….

(keep repeating until table complete)

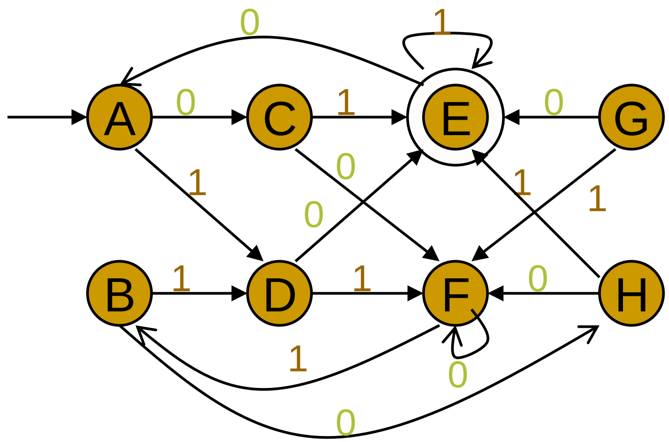


A	=							
B		=						
C			=					
D				=				
E					=			
F						=		
G							=	
H								=
	A	B	C	D	E	F	G	H



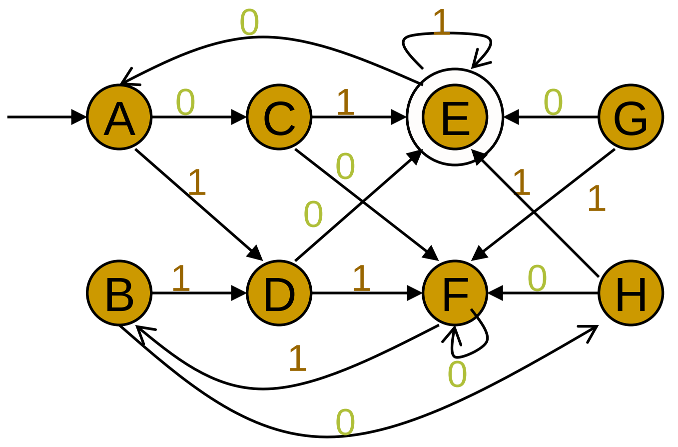
	A	B	C	D	E	F	G	H
A	=							
B		=						
C			=					
D				=				
E	X	X	X	X	=			
F					X	=		
G					X		=	
H					X			=

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



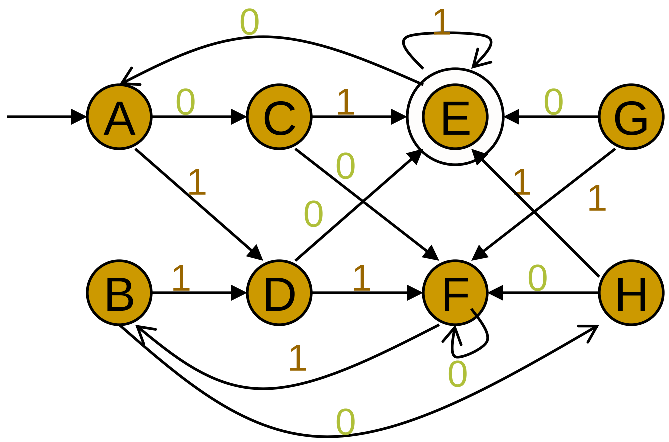
1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings

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



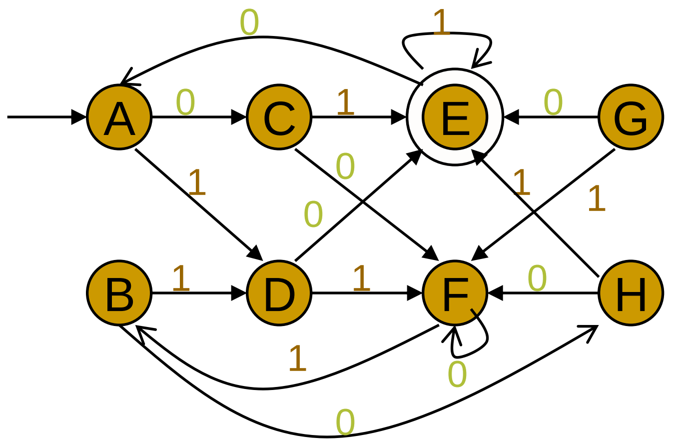
	A	B	C	D	E	F	G	H
A	=							
B		=						
C	X	X	=					
D	X	X		=				
E	X	X	X	X	=			
F					X	=		
G	X	X			X		=	
H	X	X			X			=

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



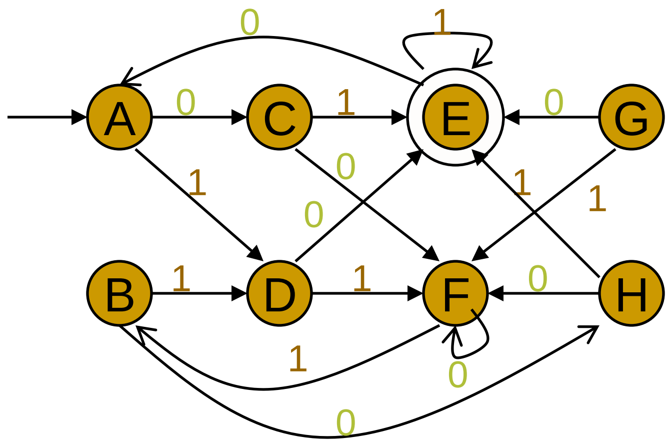
A	=							
B		=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F			X		X	=		
G	X	X	X		X		=	
H	X	X	=		X			=
	A	B	C	D	E	F	G	H

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



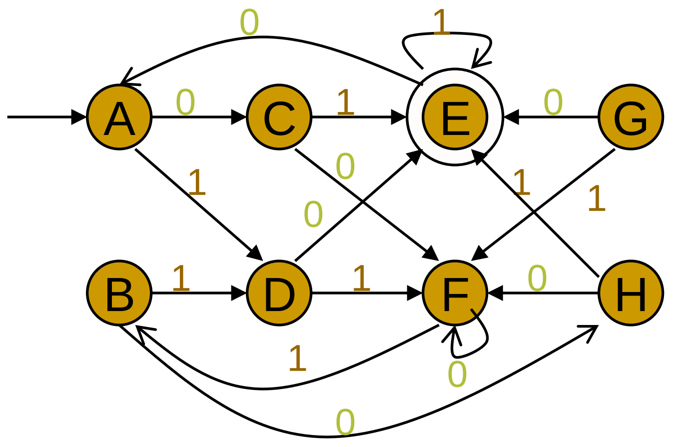
	A	B	C	D	E	F	G	H
A	=							
B		=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F			X	X	X	=		
G	X	X	X	=	X		=	
H	X	X	=	X	X			=

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



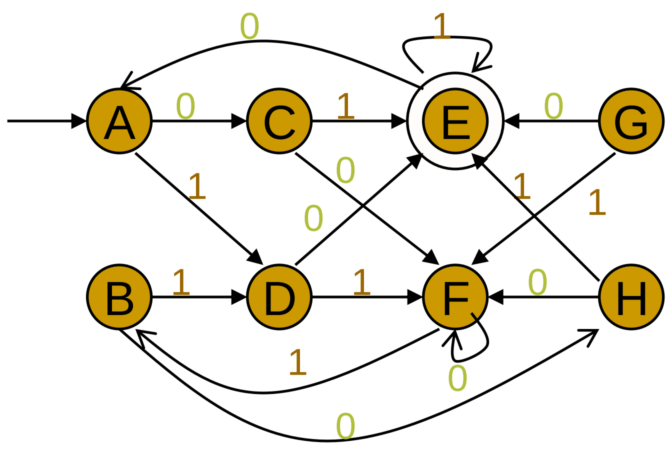
	A	B	C	D	E	F	G	H
A	=							
B		=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F			X	X	X	=		
G	X	X	X	=	X	X	=	
H	X	X	=	X	X	X		=

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



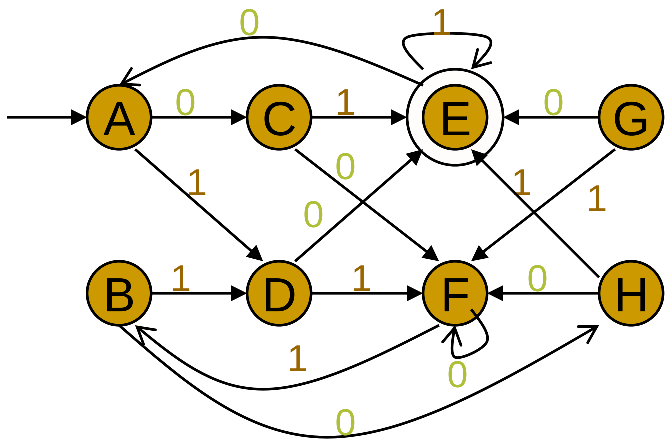
A	=							
B		=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F			X	X	X	=		
G	X	X	X	=	X	X	=	
H	X	X	=	X	X	X	X	=
	A	B	C	D	E	F	G	H

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

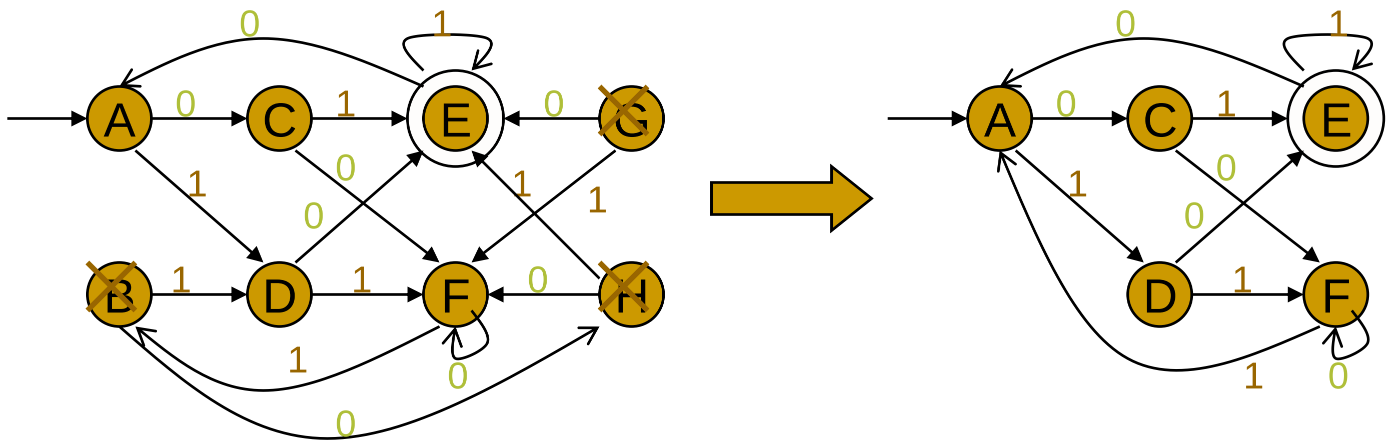
1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings
3. Look 2-hops away for distinguishing states or strings



	A	B	C	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	=

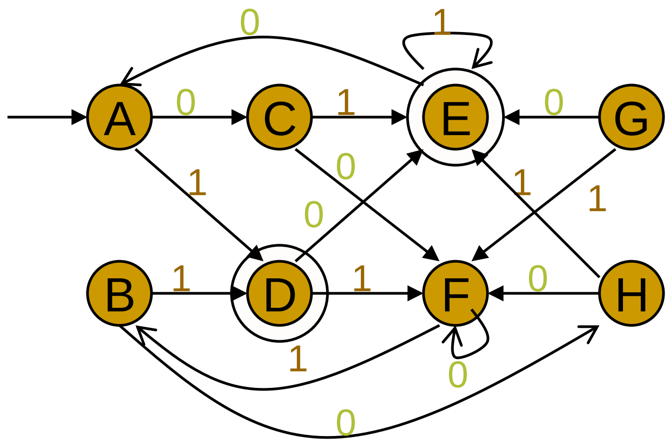
1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings
3. Look 2-hops away for distinguishing states or strings

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



Retrain only one copy for
each equivalence set of states

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



	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

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