NFAs accept the Regular Languages

Equivalence of Machines

Definition:

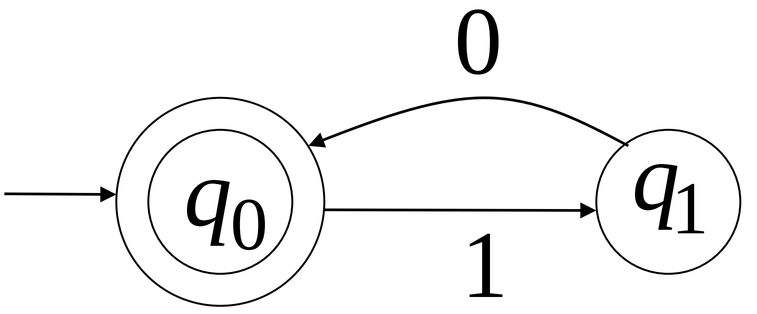
Machine M_1 is equivalent to machine M_2

if $L(M_1) = L(M_2)$

Example of equivalent machines

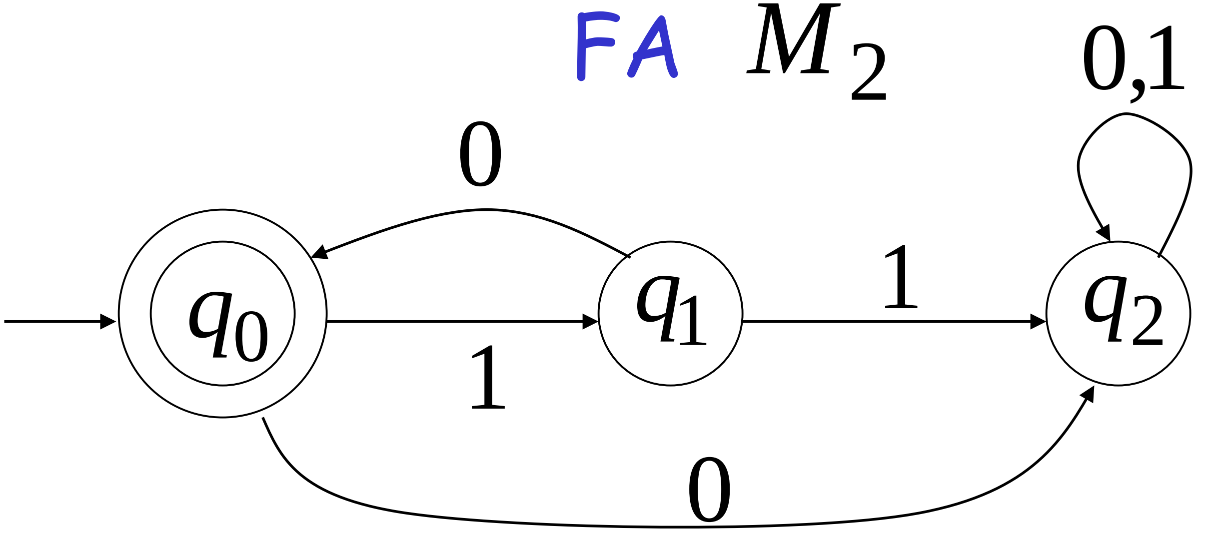
$$L(M_1) = \{10\}^*$$

NFA M_1



$$L(M_2) = \{10\}^*$$

FA M_2



We will prove:

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{accepted} \\ \text{by NFAs} \end{array} \right\} = \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

Languages
accepted
by FAs

NFAs and FAs have the same computation power

Proof: Given M_N, we use the procedure *nfa-to-dfa* below to construct the transition graph G_D for M_D. To understand the construction, remember that G_D has to have certain properties. Every vertex must have exactly $|\Sigma|$ outgoing edges, each labeled with a different element of Σ. During the construction, some of the edges may be missing, but the procedure continues until they are all there.

procedure: nfa-to-dfa

1. Create a graph G_D with vertex $\{q_0\}$. Identify this vertex as the initial vertex.

2. Repeat the following steps until no more edges are missing.

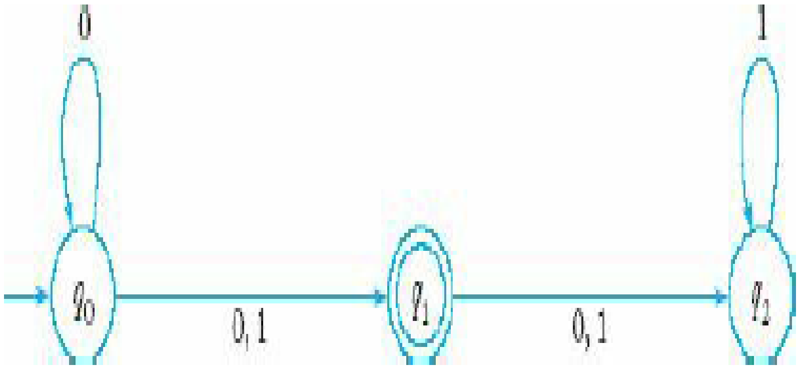
Take any vertex $\{q_i,q_j,\ldots,q_k\}$ of G_D that has no outgoing edge for some $a \in \Sigma$ Compute $\delta_N^*(q_i,a), \delta_N^*(q_j,a), \ldots, \delta_N^*(q_k,a)$. If

$$\delta_N^*(q_i,a) \cup \delta_N^*(q_j,a) \cup \ldots \cup \delta_N^*(q_k,a) = \{q_l, q_m, \ldots, q_n\},$$

create a vertex for G_D labeled $\{q_l,q_m,\ldots,q_n\}$ if it does not already exist. Add to G_D an edge from $\{q_i,q_j,\ldots,q_k\}$ and label it with a.

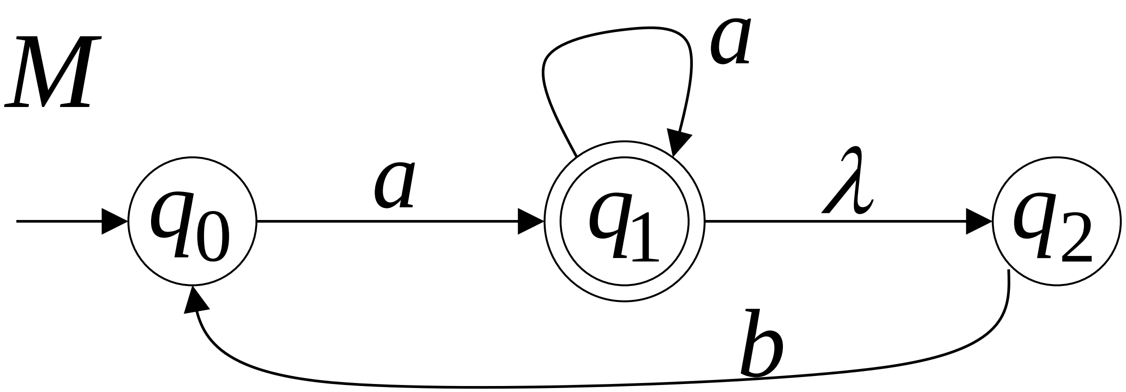
3. Every state of G_D whose label contains any $q_f \in F_N$ is identified as a final vertex.

4. If M_N accepts λ, the vertex $\{q_0\}$ in G_D is also made a final vertex.

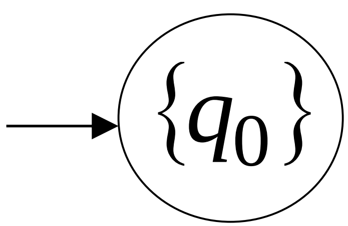

0
1
q_0
0, 1
q_1
0, 1
q_2

Convert NFA to FA

NFA M

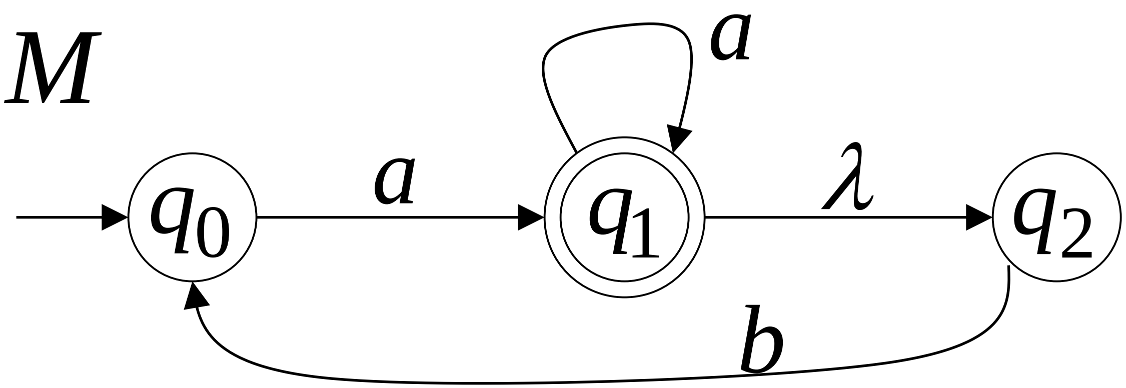


FA M'

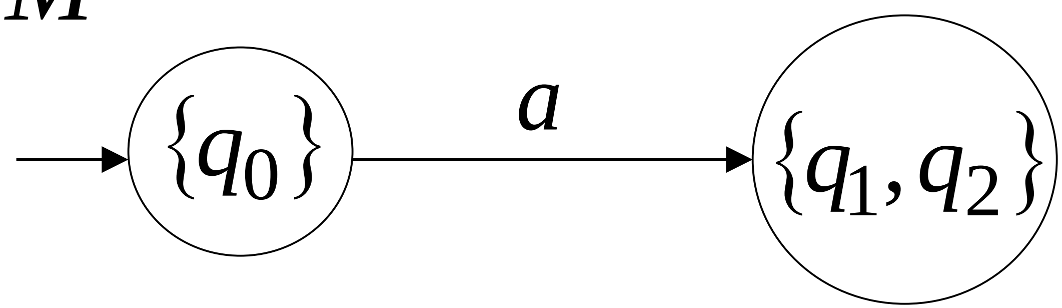


Convert NFA to FA

NFA M

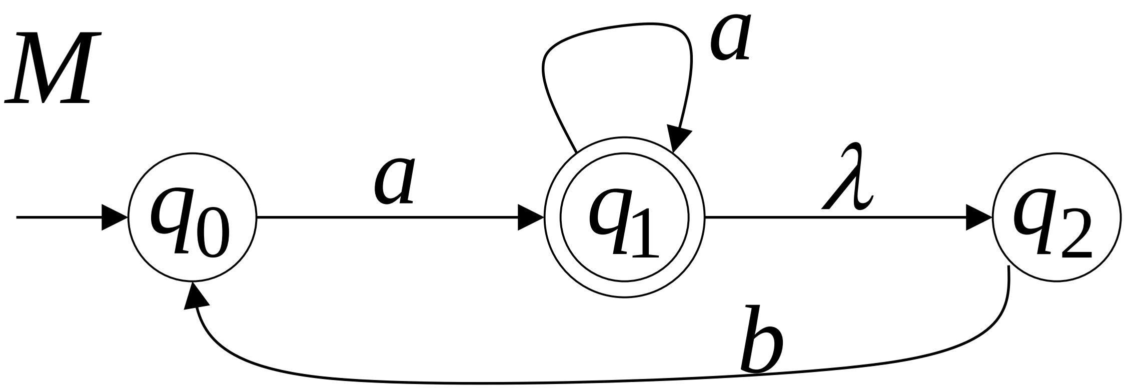


FA M'

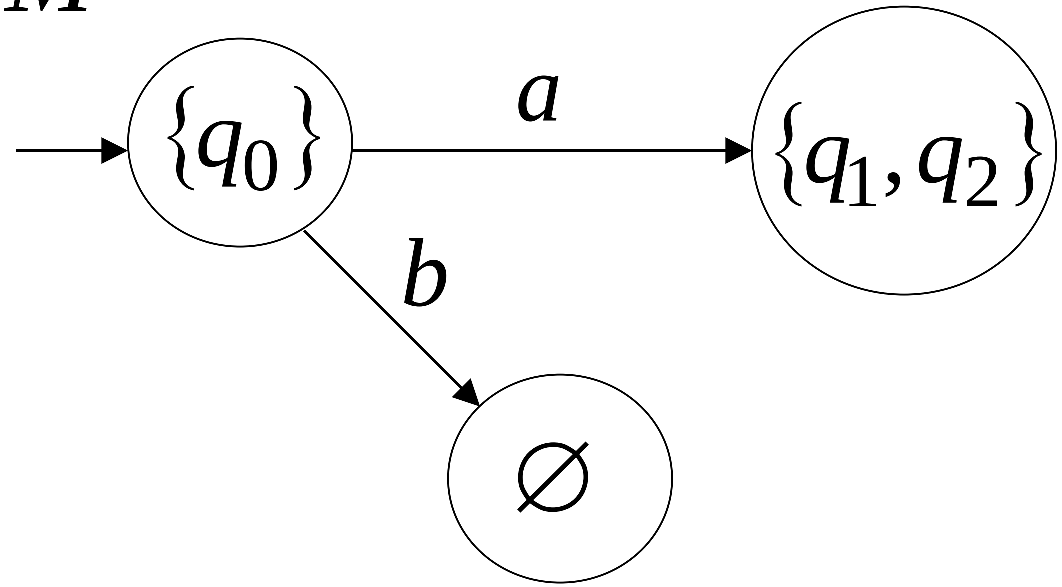


Convert NFA to FA

NFA M

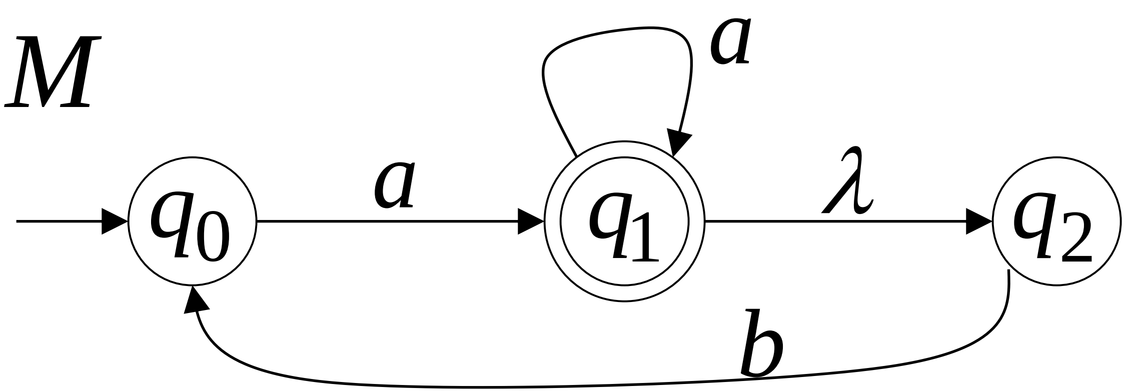


FA M'

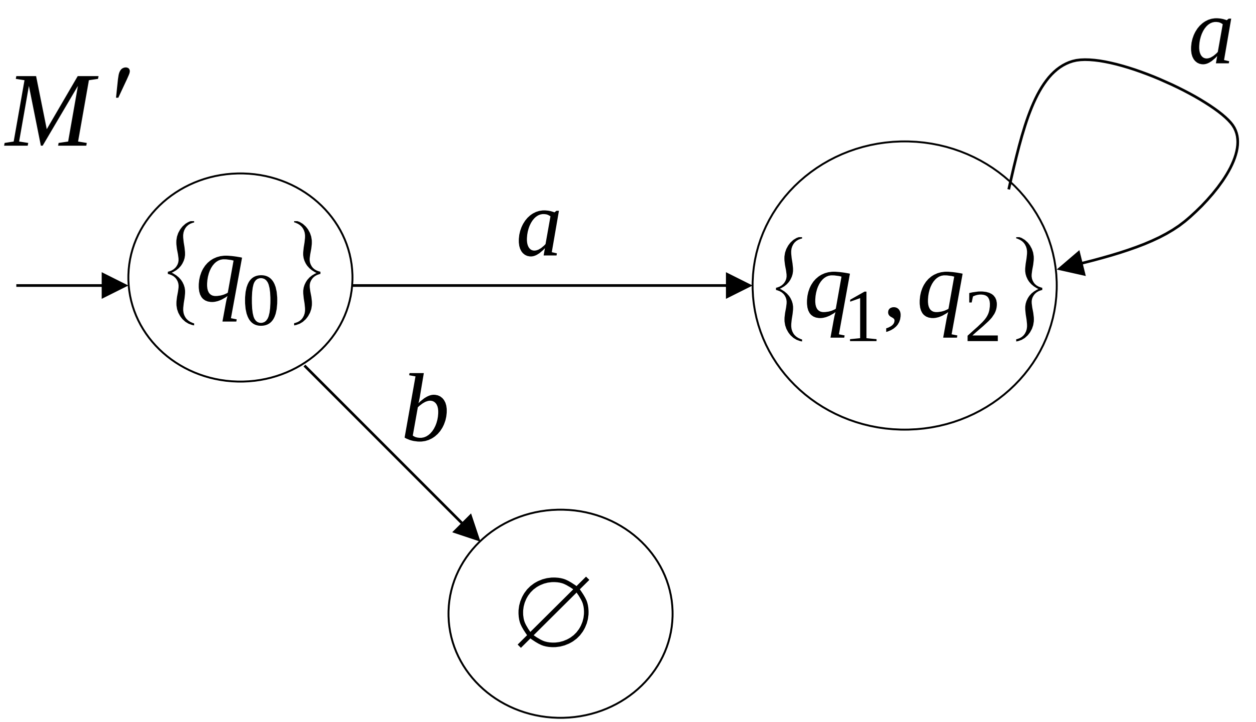


Convert NFA to FA

NFA M

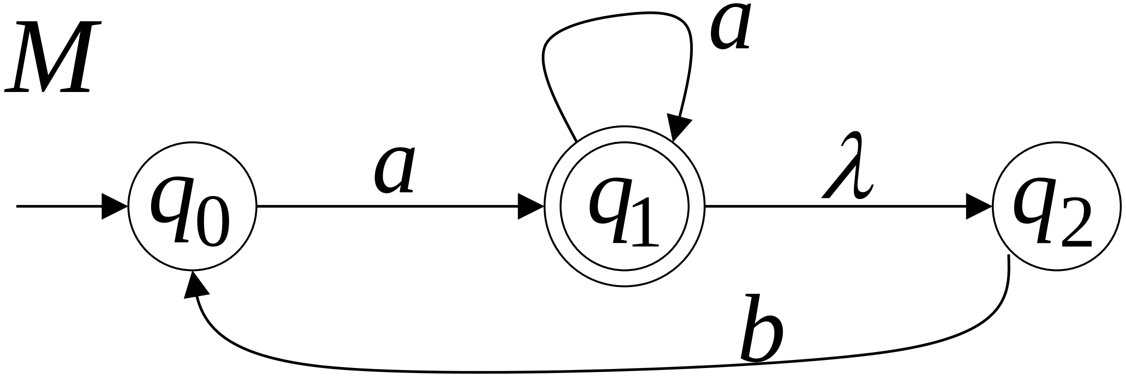


FA M'

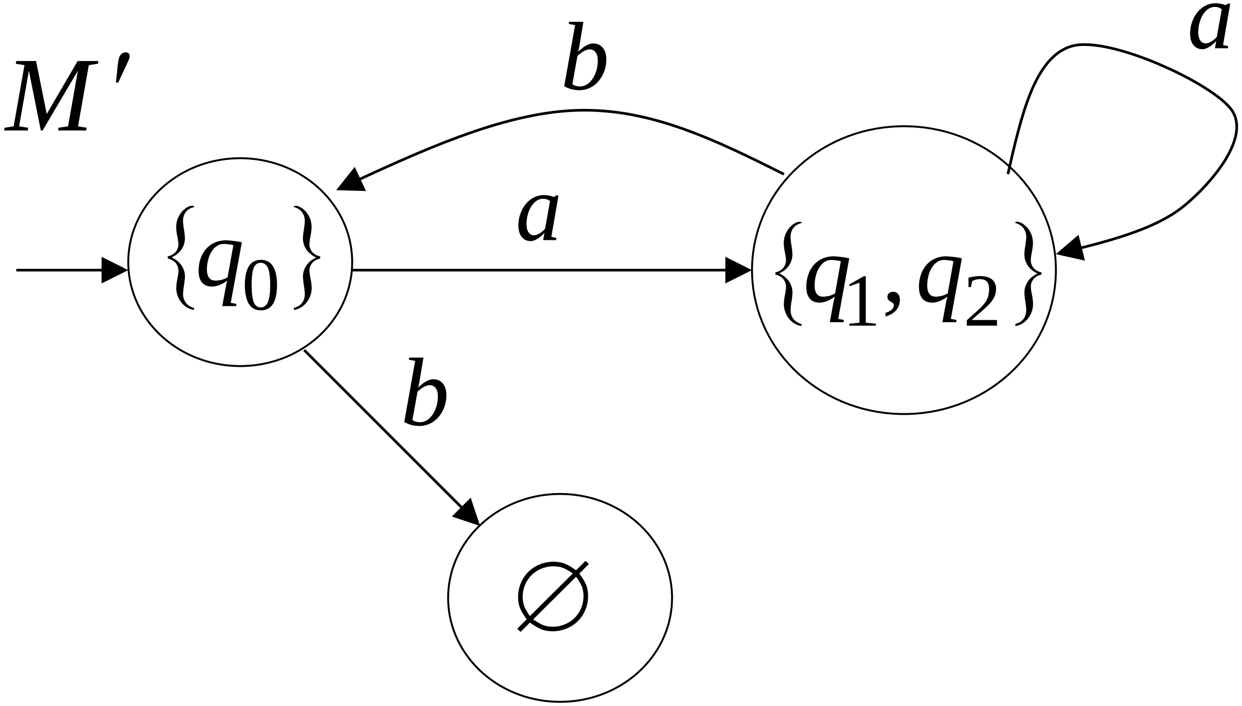


Convert NFA to FA

NFA M

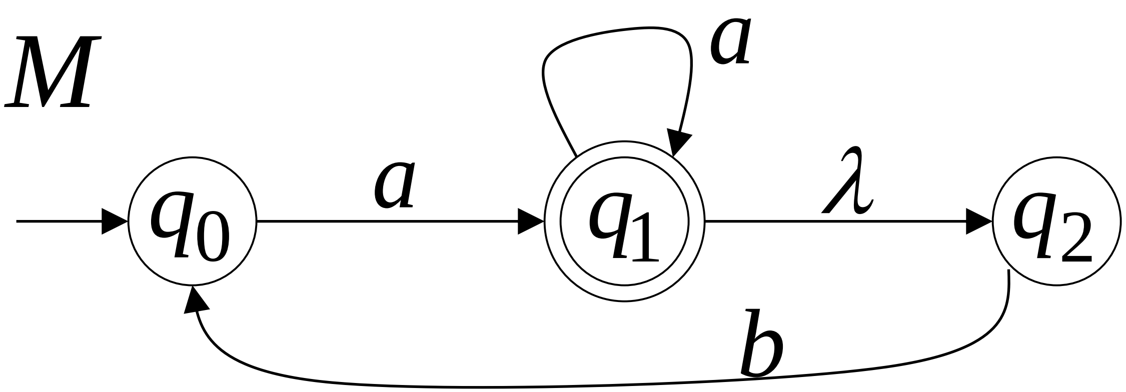


FA M'

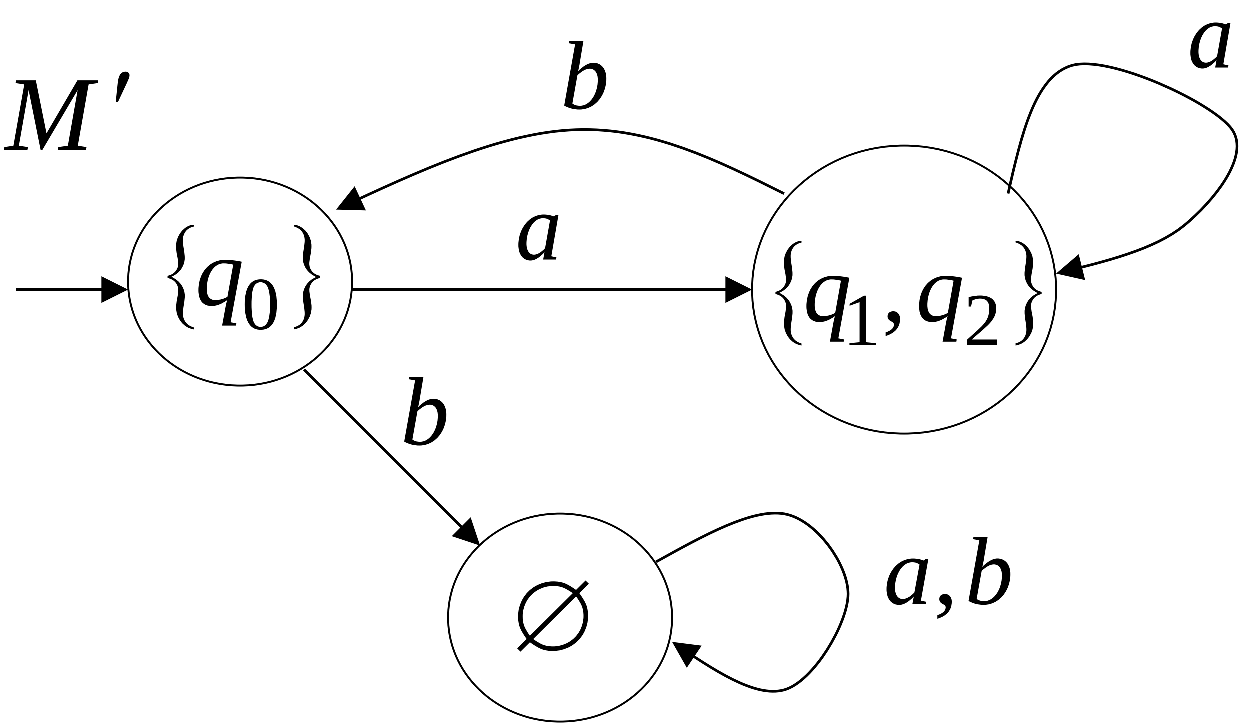


Convert NFA to FA

NFA M

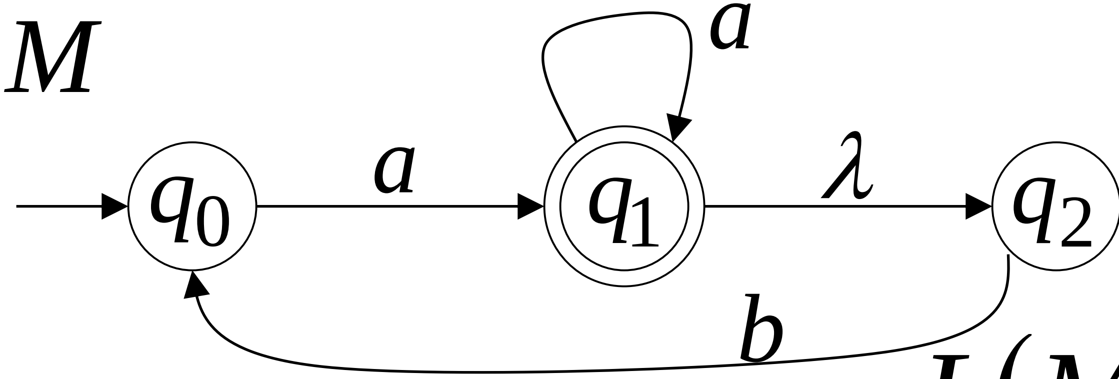


FA M'



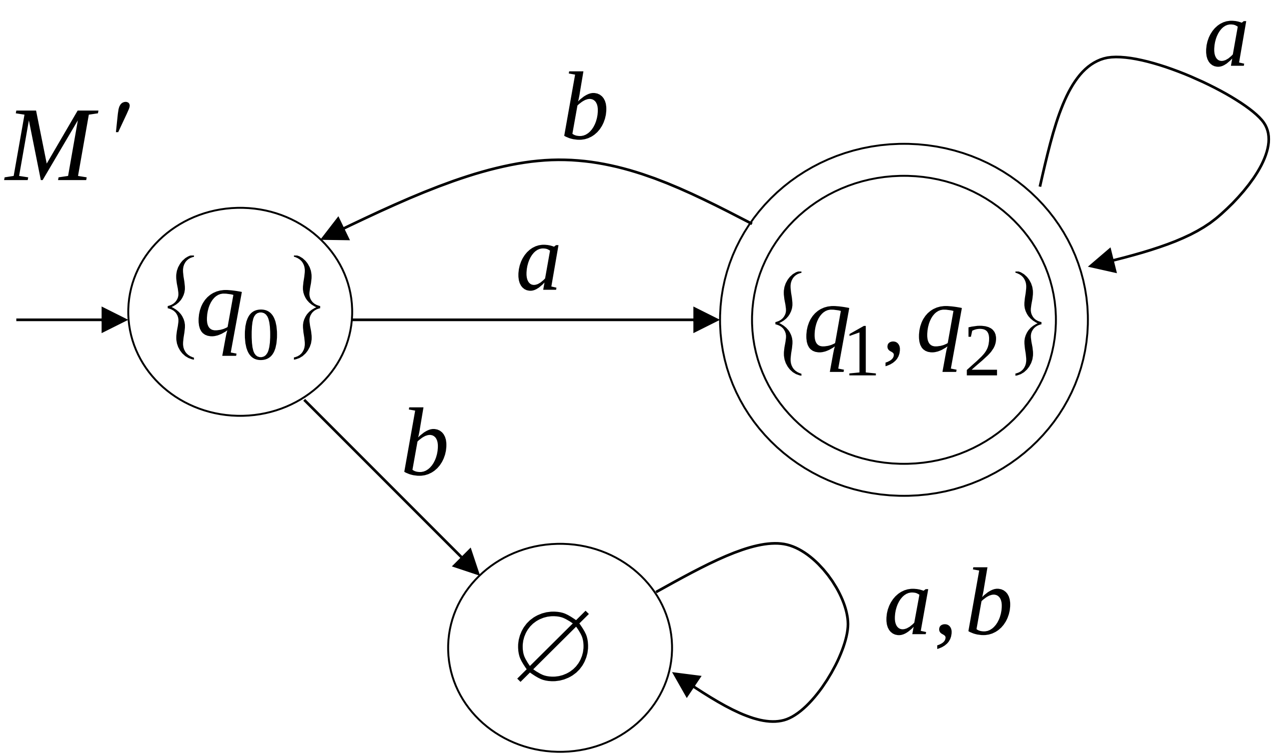
Convert NFA to FA

NFA M



$$L(M) = L(M')$$

FA M'



NFA to FA: Remarks

We are given an NFA M

We want to convert it
to an equivalent FA M'

With $L(M) = L(M')$

If the NFA has states

$$q_0, q_1, q_2, \ldots$$

the FA has states in the powerset

$$\emptyset, \{q_0\}, \{q_1\}, \{q_1, q_2\}, \{q_3, q_4, q_7\}, \ldots.$$

Single Accepting State for NFAs

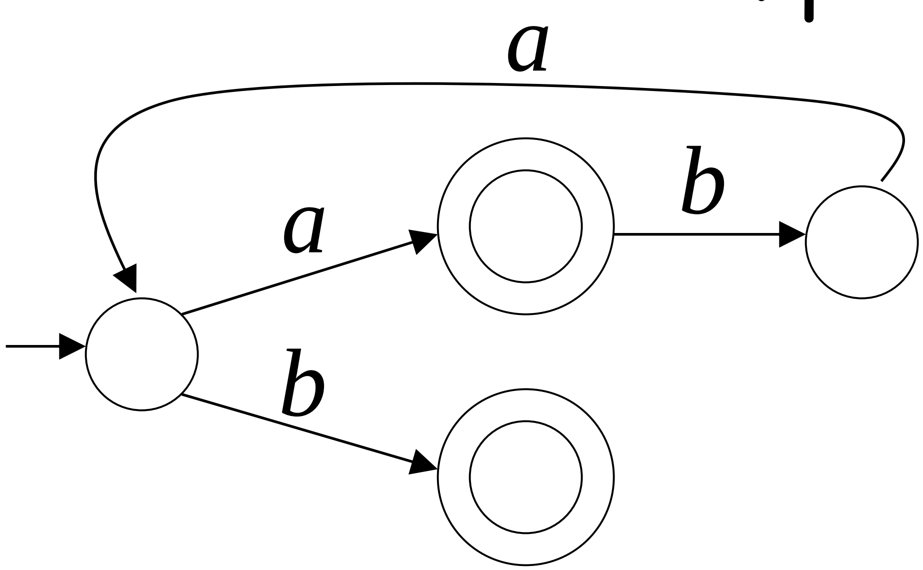
Any NFA can be converted

to an equivalent NFA

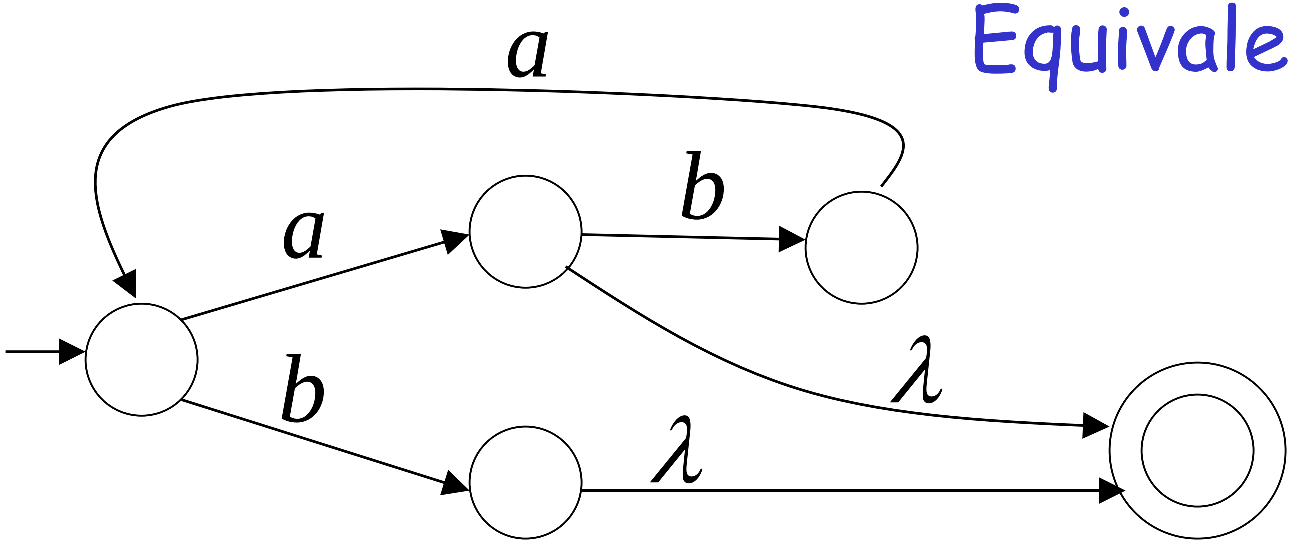
with a single accepting state

Example

NFA

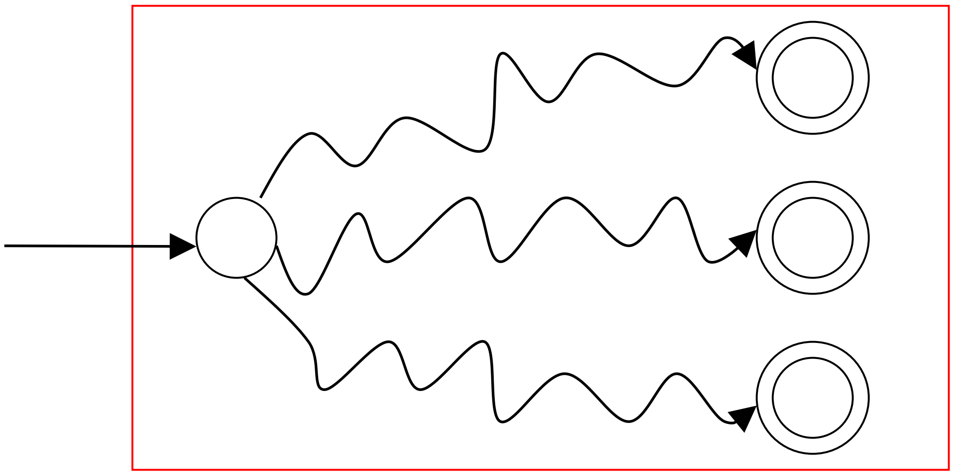


Equivalent NFA

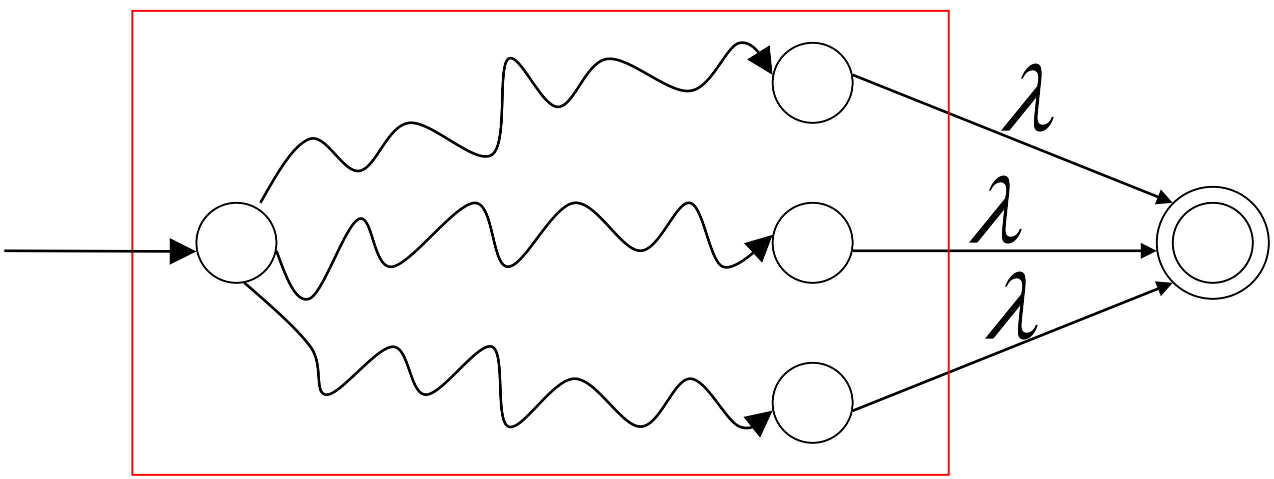


In General

NFA



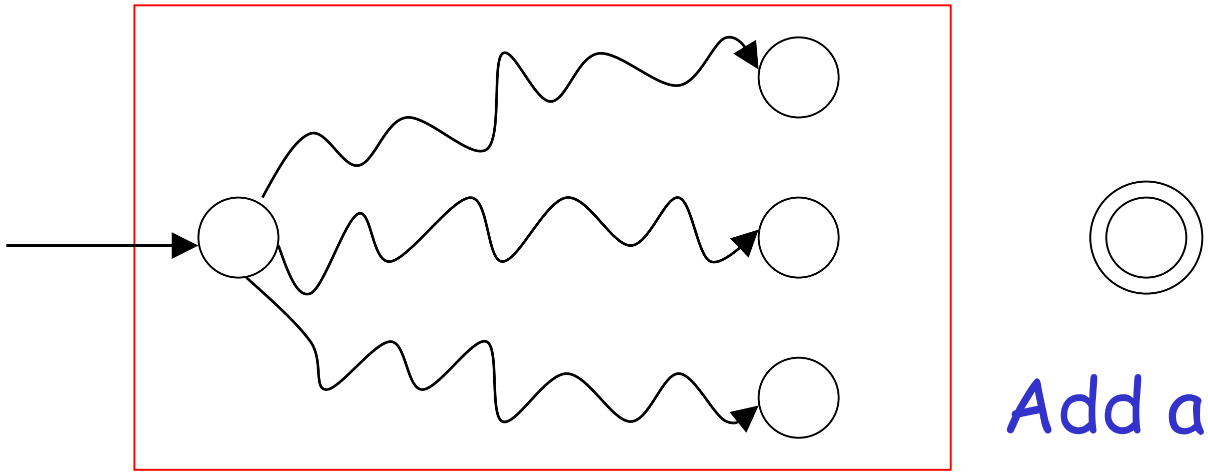
Equivalent NFA



Single accepting state

Extreme Case

NFA without accepting state



Add an accepting state
without transitions

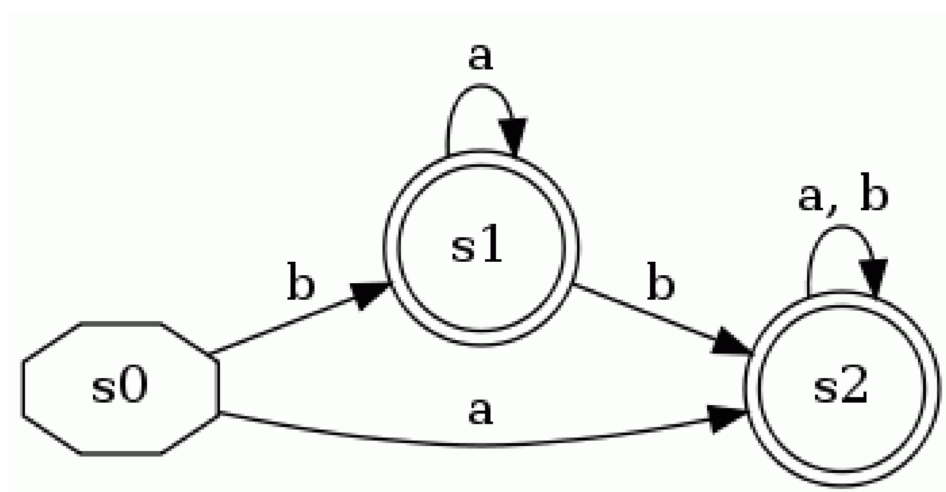
Minimization of DFA

Algorithm

First delete any state that is unreachable from the start state;
For each pair of states where one is a final state and the other is non-final,
mark them as distinguishable;
For each pair of states q_i and q_j
For each symbol in the alphabet,
If q_i takes the automaton to q_m and q_j to q_n and
If q_m and q_n are already marked as distinguishable,
Then mark q_i and q_j as distinguishable;
Repeat the above until no more pairs can be marked;
All the pairs of states that are not marked are indistinguishable;
Collapse indistinguishable pairs to single states and merge their transitions.

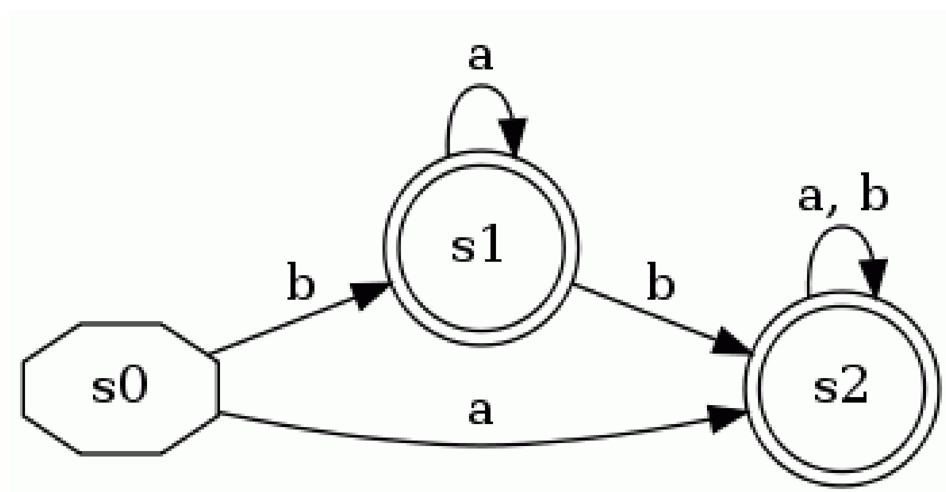
Very Simple Example

s0			
s1			
s2			
	s0	s1	s2



Very Simple Example

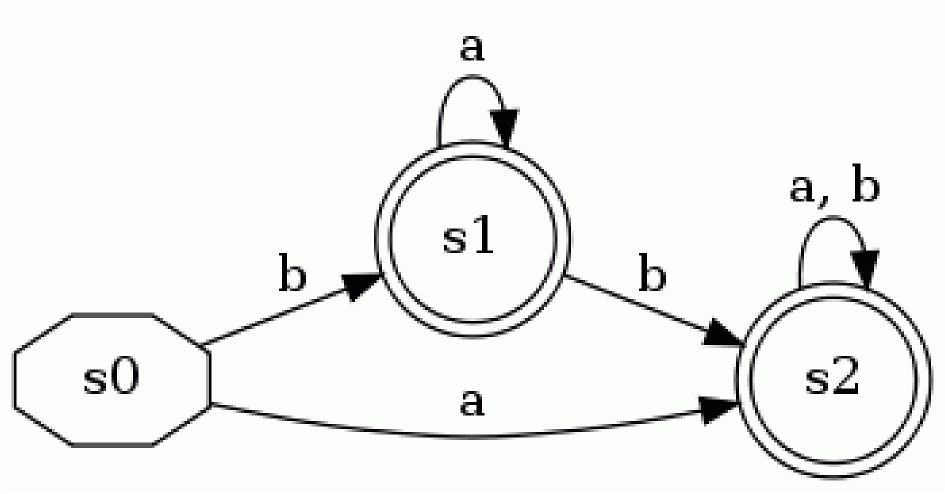
s0			
s1	X		
s2	X		
	s0	s1	s2



Label pairs with ε where one is a final state and the other is not

Very Simple Example

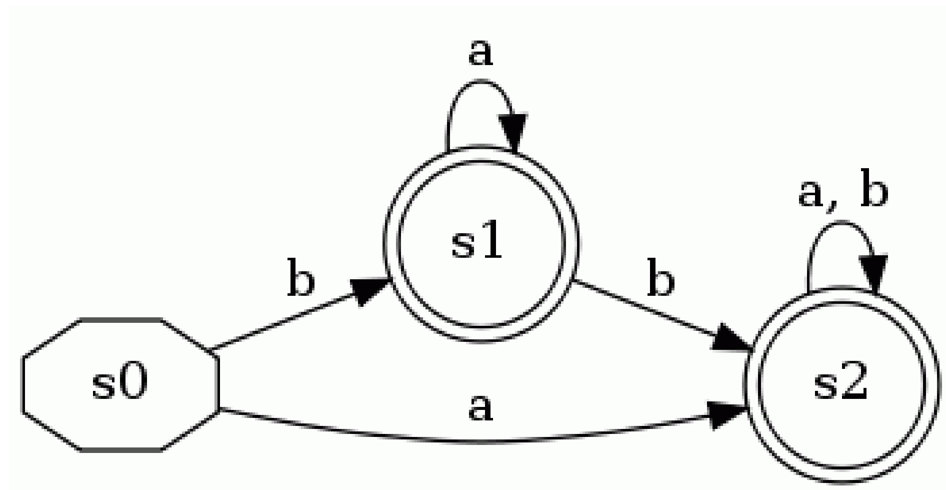
s0			
s1	X		
s2	X		
	s0	s1	s2



Main loop (no changes occur)

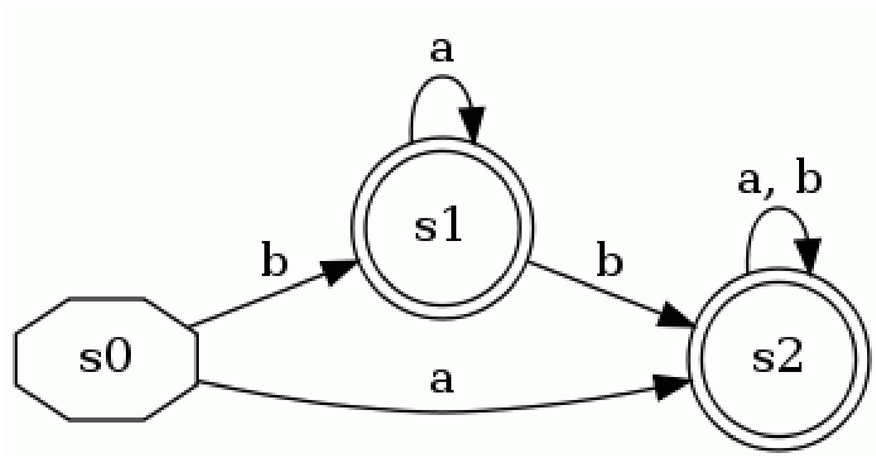
Very Simple Example

s0			
s1	X		
s2	X		
	s0	s1	s2

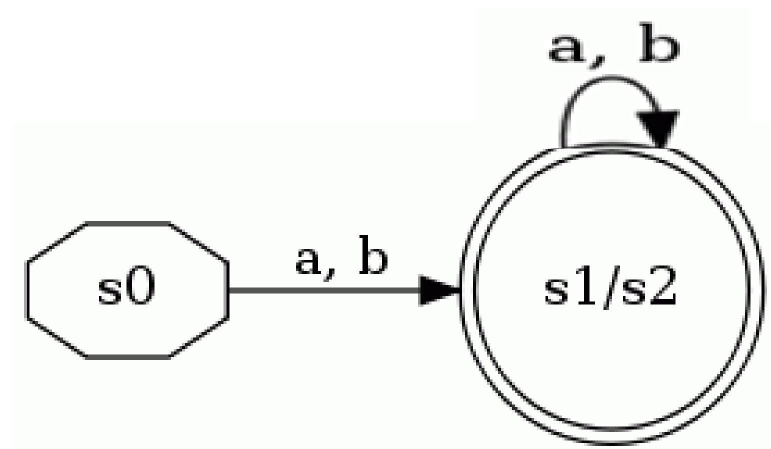


DISTINGUISHABLE(s1, s2) is empty, so s1 and s2 are equivalent stat

Very Simple Example



Merge s1 and s2

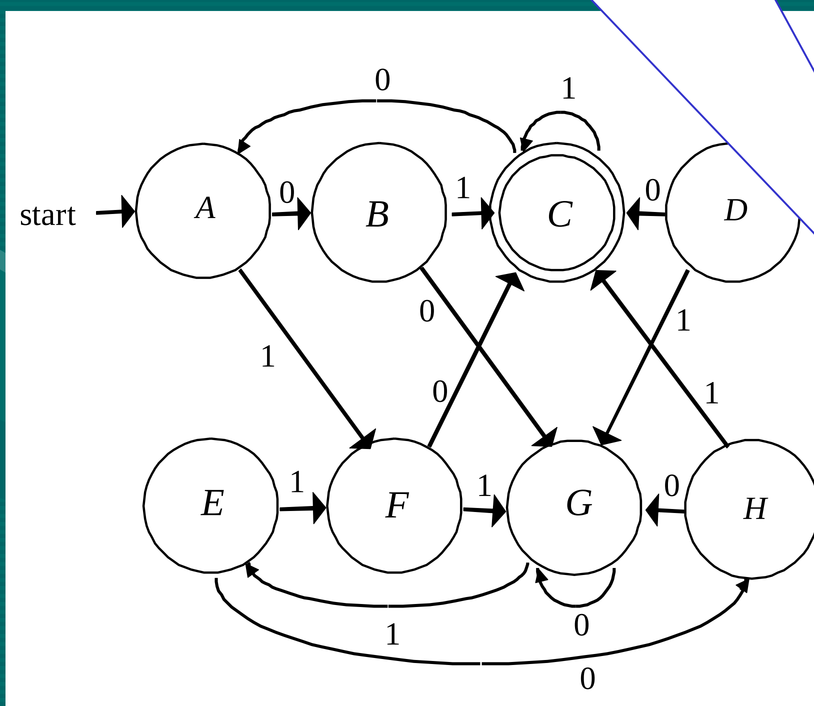


Example2

$\{r, s\} = \{B, G\}, \{F, C\}$ with $\{F, C\}$ already marked, so mark $\{p, q\} = \{A, B\}$

$\{r, s\} = \{B, G\}, \{E, F\}$ with both unmarked, so put $\{A, G\}$ into lists of $\{B, G\}$ and $\{E, F\}$

$\{r, s\} = \{C, E\}$ which is marked already, so mark $\{B, G\}$ and also $\{A, G\}$ in the list



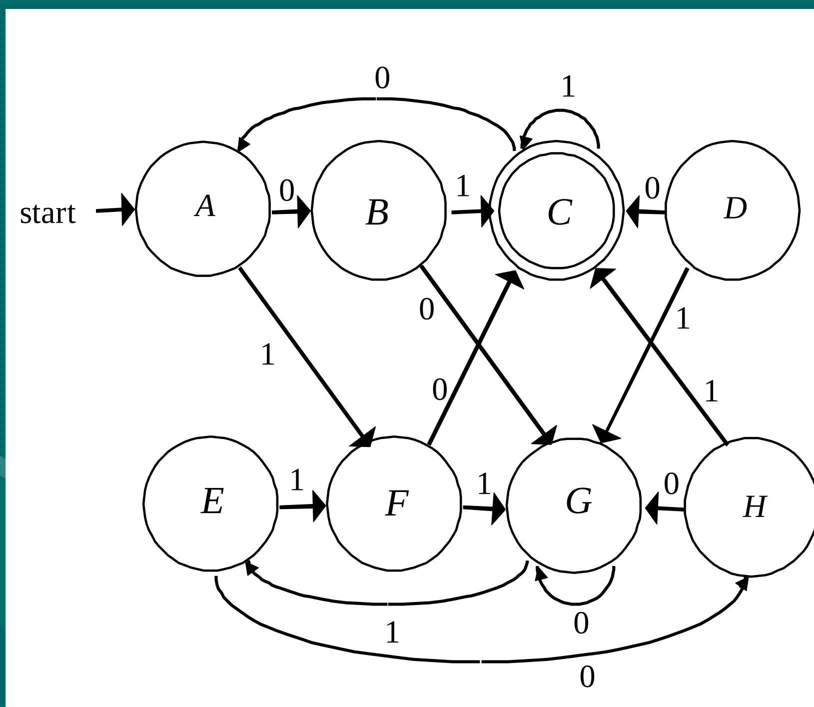
	A	B	C	D	E	F	G
B	X						
C	X	X					
D			X				
E			X				
F							
G	X	X	X				
H			X				

List = $\{A, G\}$

List = $\{A, G\}$

Example 2 (contd…_

– Final results are as follows.



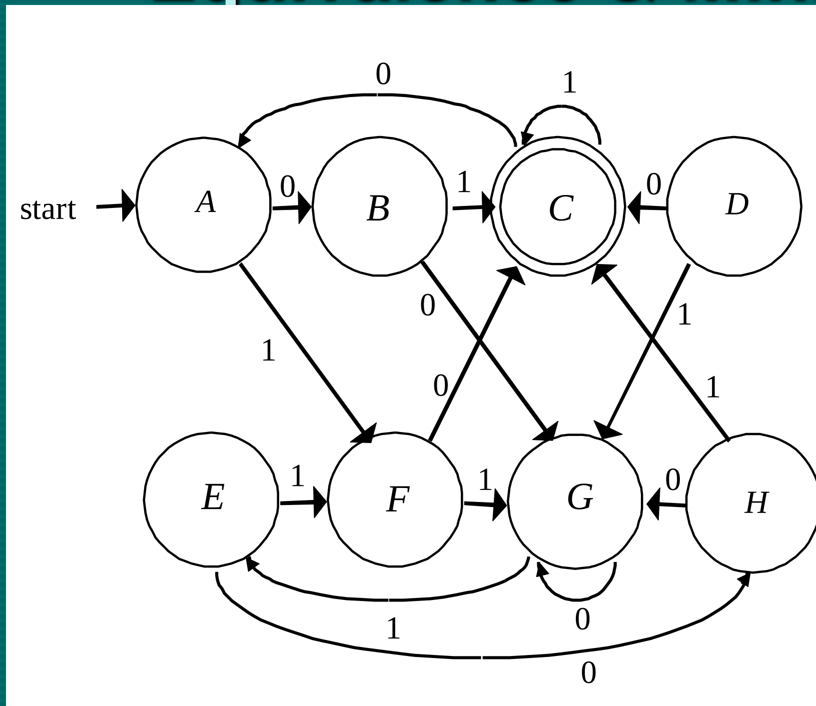
B	X						
C	X	X					
D	X	X	X				
E		X	X	X			
F	X	X	X		X		
G	X	X	X	X	X	X	
H	X		X	X	X	X	X
	A	B	C	D	E	F	G

– Then, **what???**

Equivalence & Minimization of Automata

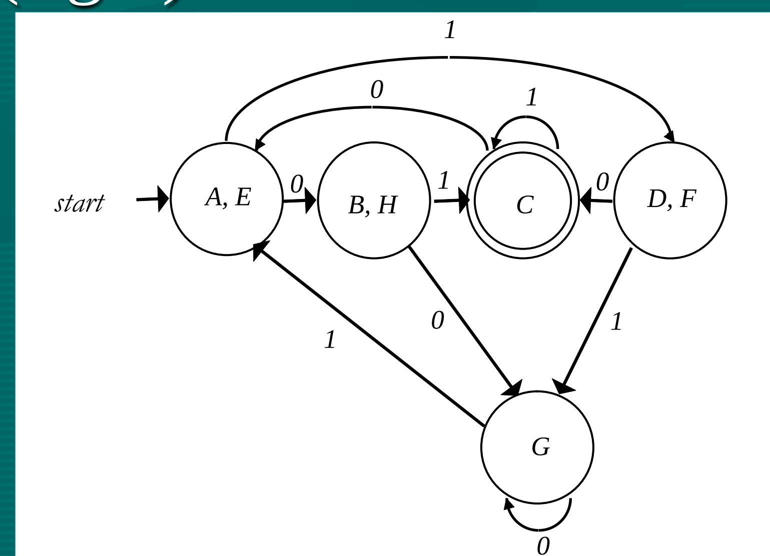
- **If two states are not distinguishable by the table-filling algorithm, then they are equivalent.**
- **Minimization of DFA's**
  – Group equivalent states into a block and regard each block as a new state in the minimized DFA.
  – Take the block containing the old start state as the new start state.
  – Take the new accepting states as those blocks which contain old accepting states.

Equivalence & Minimization of Automata



- The final result below says (A, E), (B, H), (D, F) are equivalent states and can be put into 3 blocks as states of the new DFA. The final new DFA is as follows (right).

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



Example 3

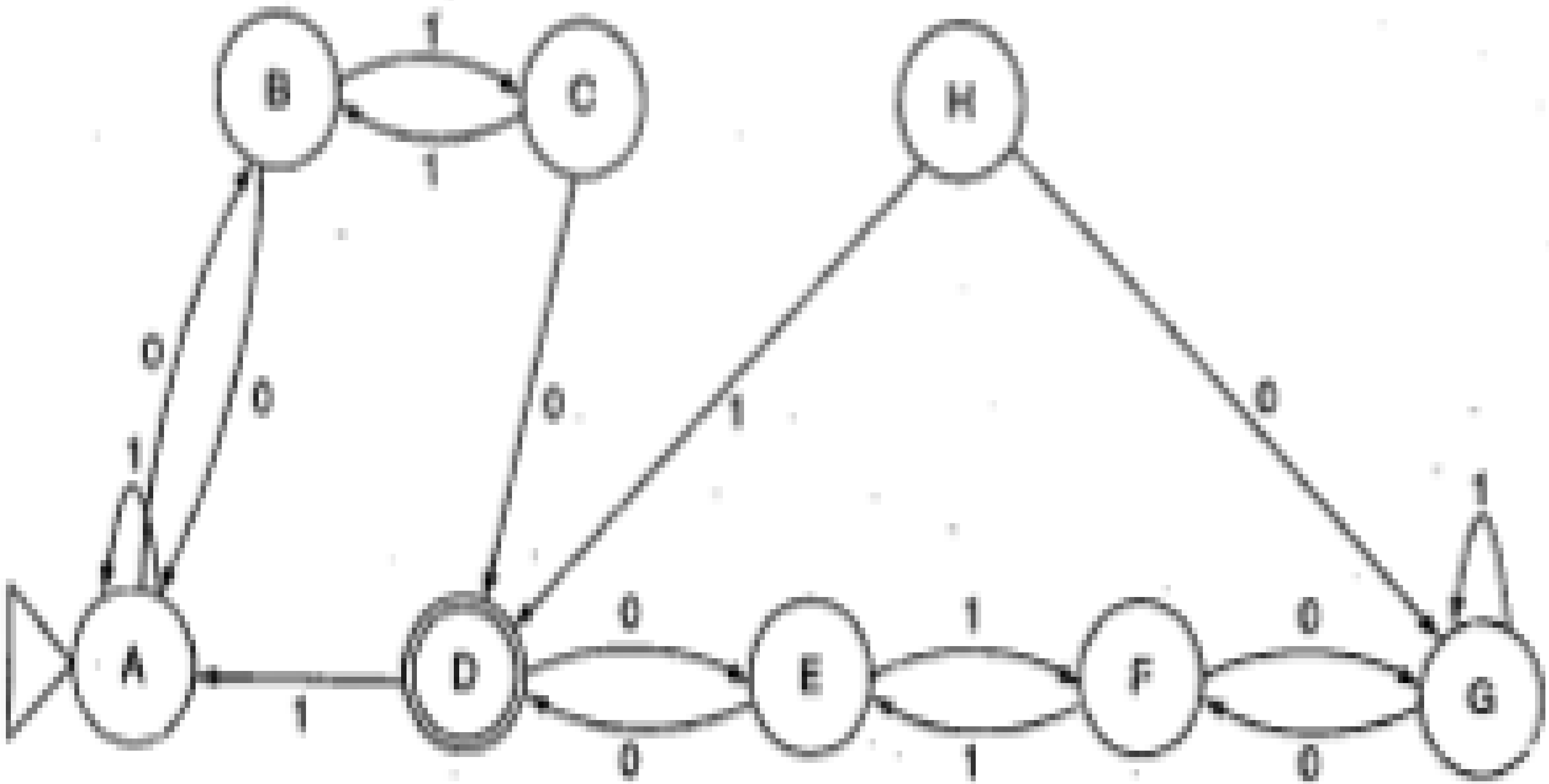
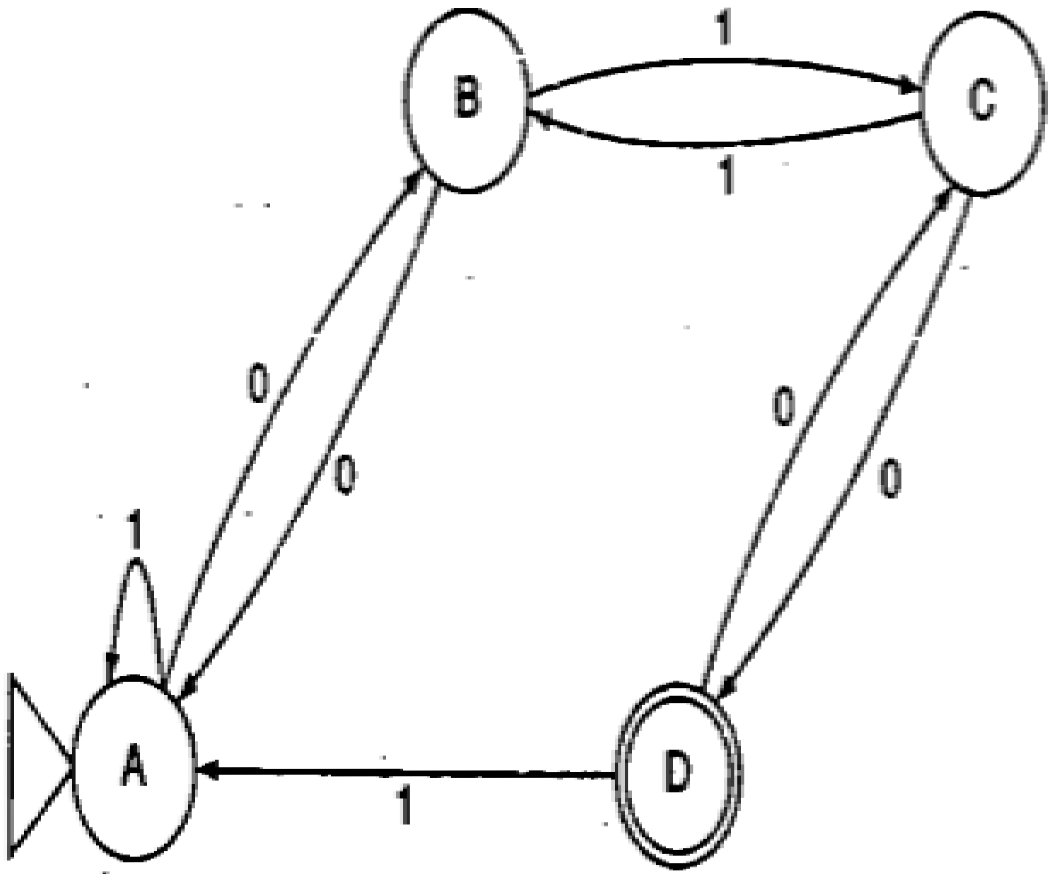
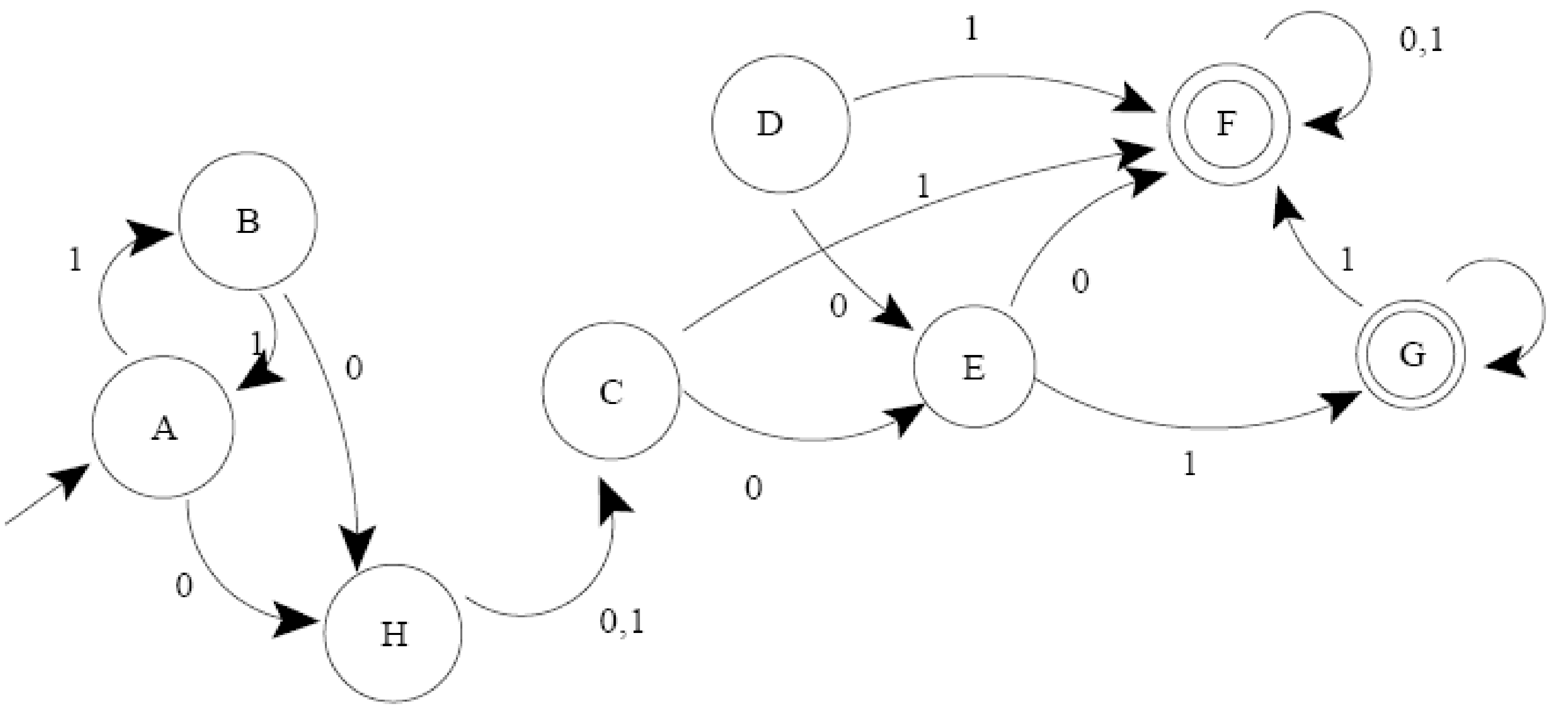

B
C
H
A
D
E
F
G
1
1
0
0
1
0
1
0
1
1
0
0
1
1
0
0

TABLE 3.4 Marking Distinguishable States for Minimizing a DFA

B	on 1 (A×C)						
C	on 0	on 0					
D	fnf	fnf	fnf				
E	on 0	on 0	?	fnf			
F	on 1 (A×E)	?	on 0	fnf	on 0		
G	?	on 1 (C×G)	on 0	fnf	on 0	on 1 (E×G)	
H							
States	**A**	**B**	**C**	**D**	**E**	**F**	**G**


1
B
C
1
0
0
0
0
1
A
D
1

More Complex Example



More Complex Example

Check for pairs with one state final and one not:

b							
c							
d							
e							
f	ϵ	ϵ	ϵ	ϵ	ϵ		
g	ϵ	ϵ	ϵ	ϵ	ϵ		
h						ϵ	ϵ
	a	b	c	d	e	f	g

More Complex Example

First iteration of main loop:

b							
c	1	1					
d	1	1					
e	0	0	0	0			
f	ϵ	ϵ	ϵ	ϵ	ϵ		
g	ϵ	ϵ	ϵ	ϵ	ϵ		
h			1	1	0	ϵ	ϵ
	a	b	c	d	e	f	g

More Complex Example

Second iteration of main loop:

b							
c	1	1					
d	1	1					
e	0	0	0	0			
f	ϵ	ϵ	ϵ	ϵ	ϵ		
g	ϵ	ϵ	ϵ	ϵ	ϵ		
h	1	1	1	1	0	ϵ	ϵ
	a	b	c	d	e	f	g

More Complex Example

Third iteration makes no changes

- Blank cells are equivalent pairs of states

b							
c	1	1					
d	1	1					
e	0	0	0	0			
f	ϵ	ϵ	ϵ	ϵ	ϵ		
g	ϵ	ϵ	ϵ	ϵ	ϵ		
h	1	1	1	1	0	ϵ	ϵ
	a	b	c	d	e	f	g

More Complex Example

Combine equivalent states for minimized DFA: