Theory of Computation

Introduced By:

Prof. Safia Abbas

Text Books:

"***An Introduction to the theory of computation***" by Michael Sipser, 2nd Edition, PWS Publishing Company, Inc. 2006; ISBN: 0-534-95097-3

"***An Introduction to Formal Languages & Automata***" by Peter Linz, 5th Edition, Jones & Bartlett Publishers, Inc. January 2012; ISBN:978-1-4496-1552-9 .

Agenda

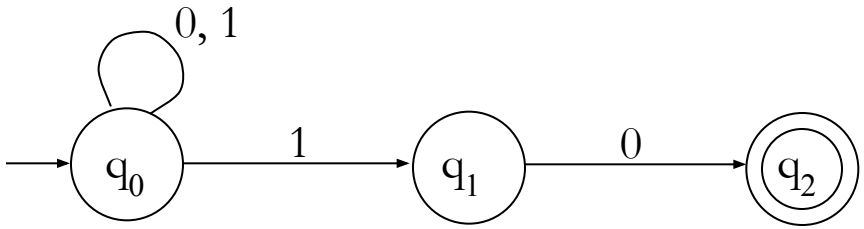
- Conversion between DFA and NFA
- Closure properties of Regular language
- Regular Expression

How to convert from NFA to DFA

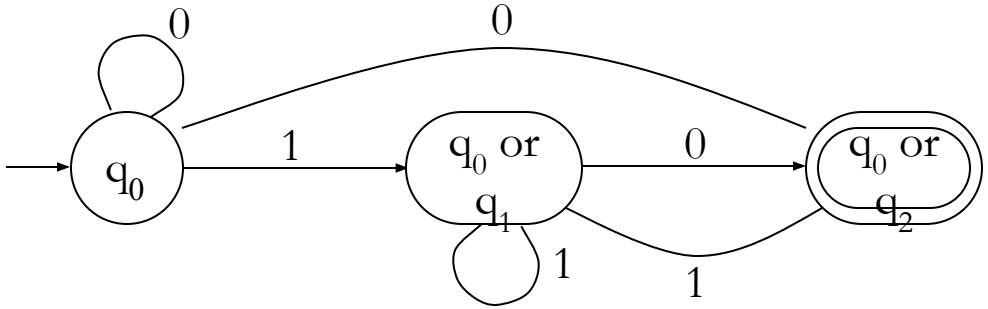
- **Create a Graph G_D with vertex $\{q_0\}$**
- **Repeat until no more edges are missing**
  - Take any vertex $\{q_i, q_j, .. q_k\}$ of G_D that has no outgoing edge
  - Compute $\delta^*(q_i, a), \delta^*(q_j, a).. \delta^*(q_k, a)$
  - Form a union of all these δ^* $\{q_l, q_m, .. q_n\}$
  - Create a vertex for G_D labeled $\{q_l, q_m, .. q_n\}$, if does not exist
  - Add an edge a from $\{q_i, q_j, .. q_k\}$ to $\{q_l, q_m, .. q_n\}$
- **Any vertex has $q_f \in F_N$ is identified as a final state**
- **If M_N accepts λ, the vertex $\{q_0\}$ in G_D is a final state**

NFA to DFA conversion intuition

NFA:

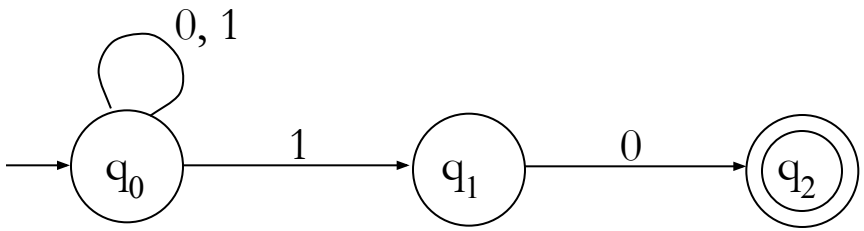


DFA:

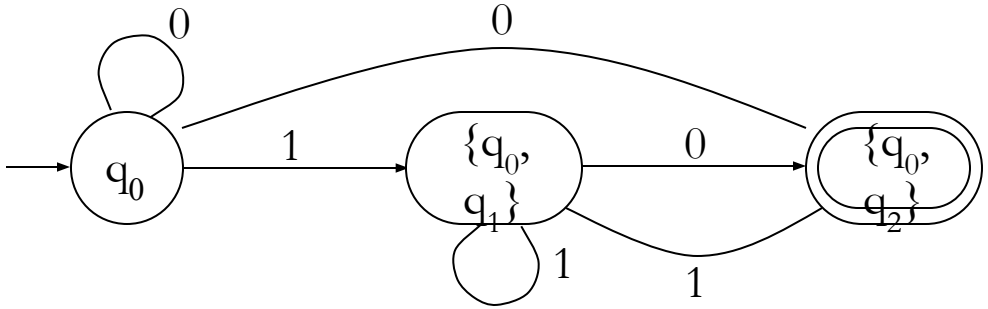


NFA to DFA conversion intuition

NFA:

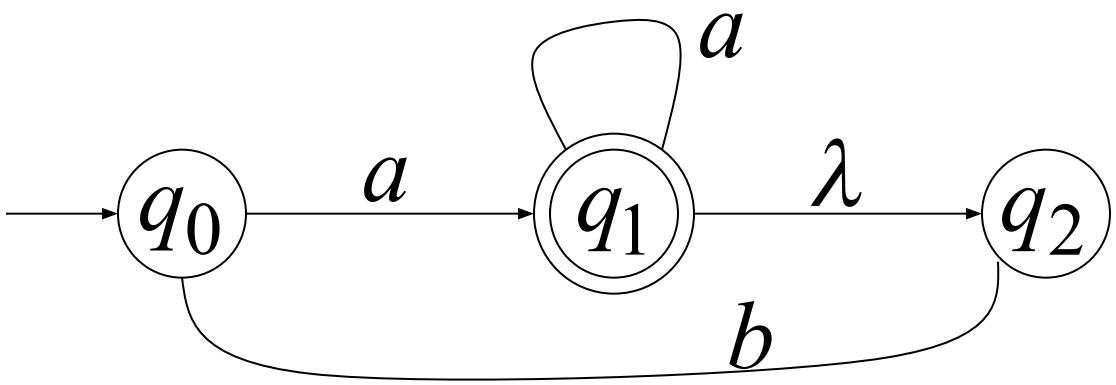


DFA:

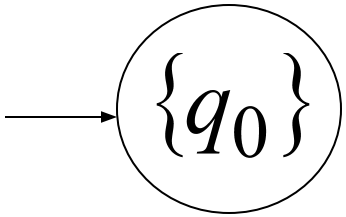


NFA to DFA

NFA

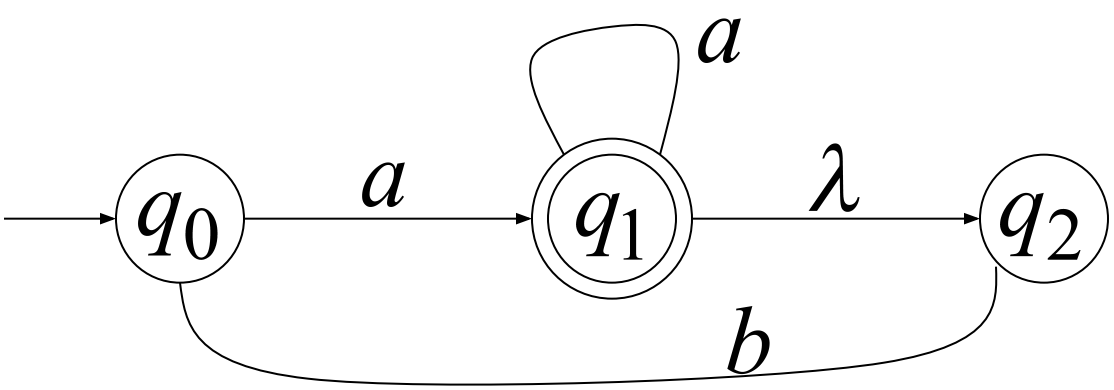


DFA

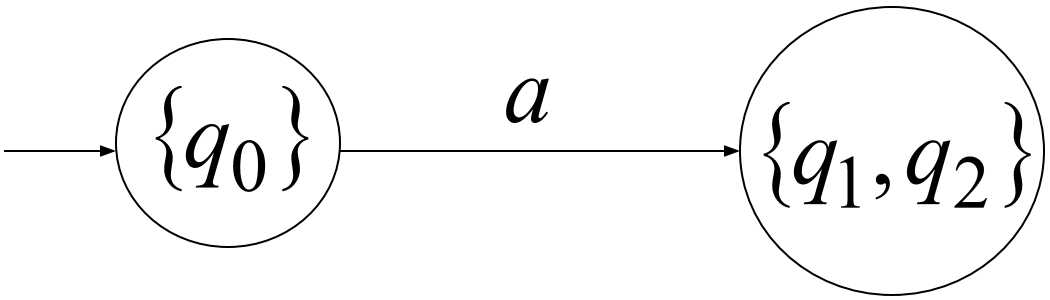


NFA to DFA

NFA

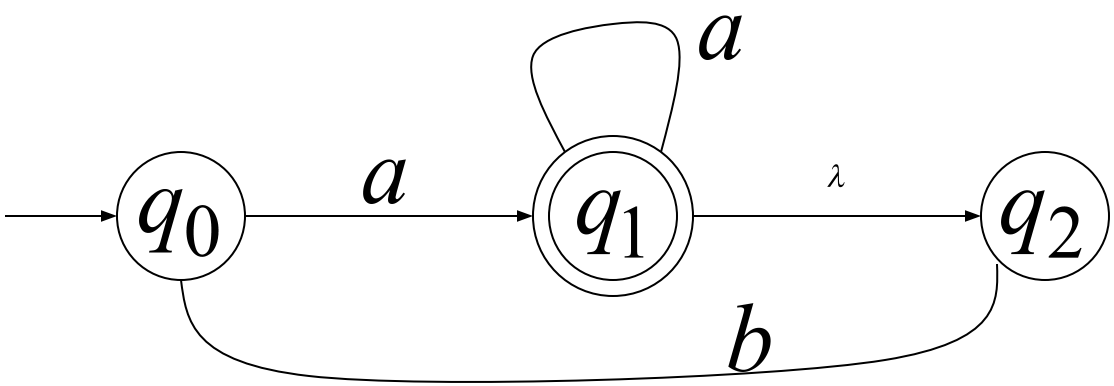


DFA

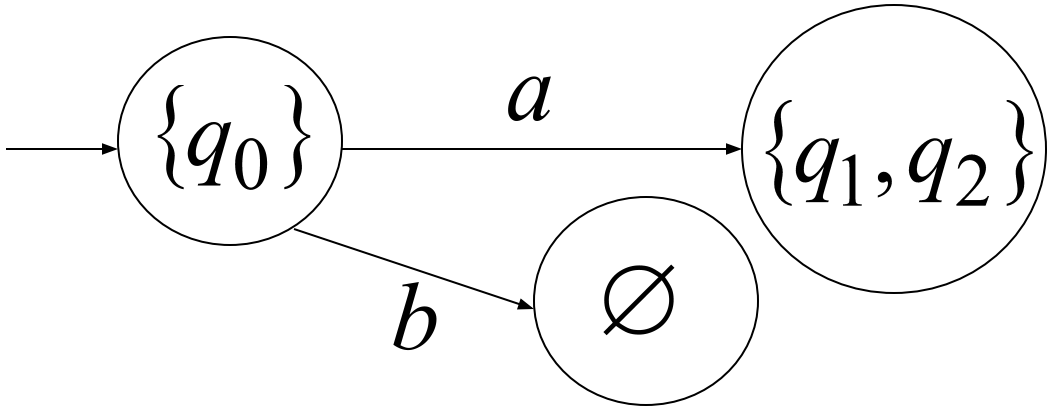


NFA to DFA

NFA

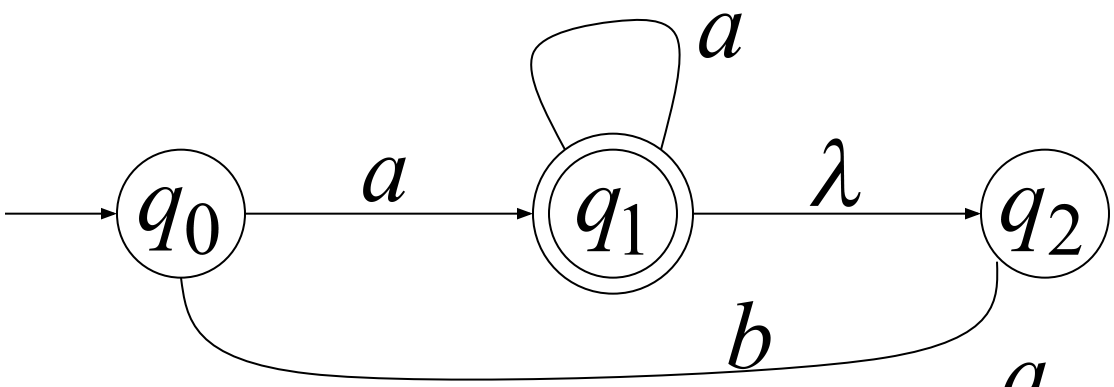


DFA

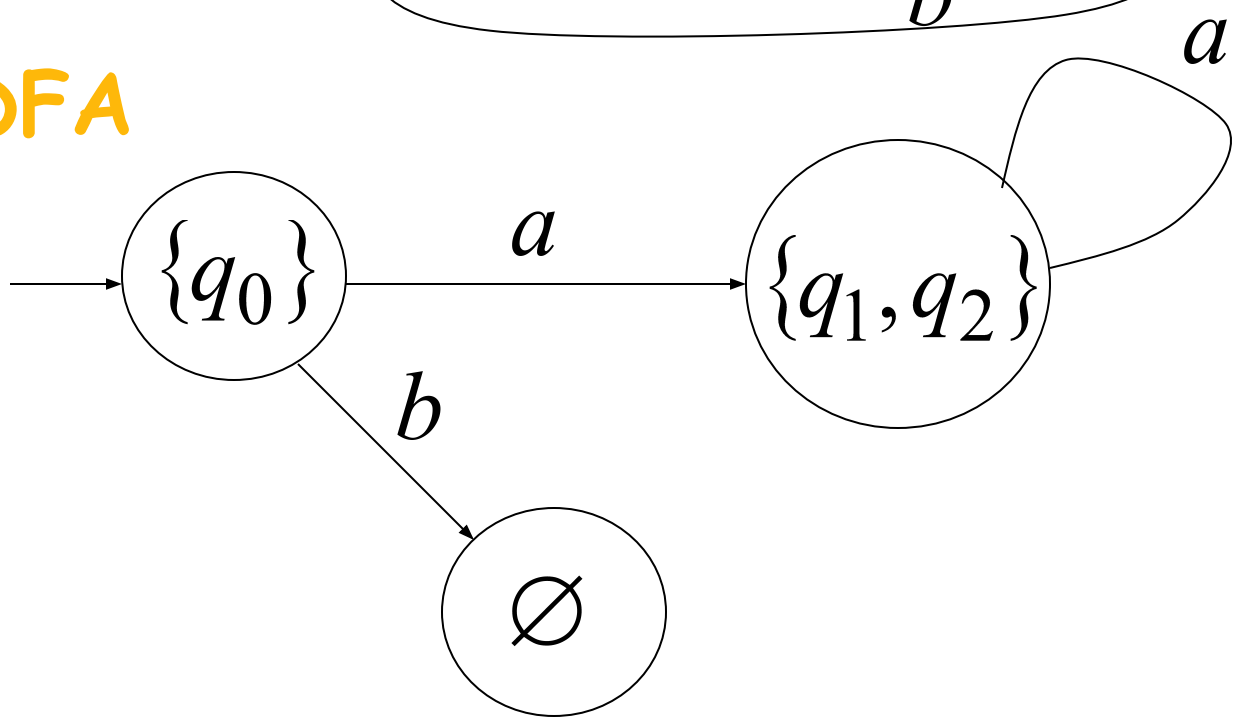


NFA to DFA

NFA

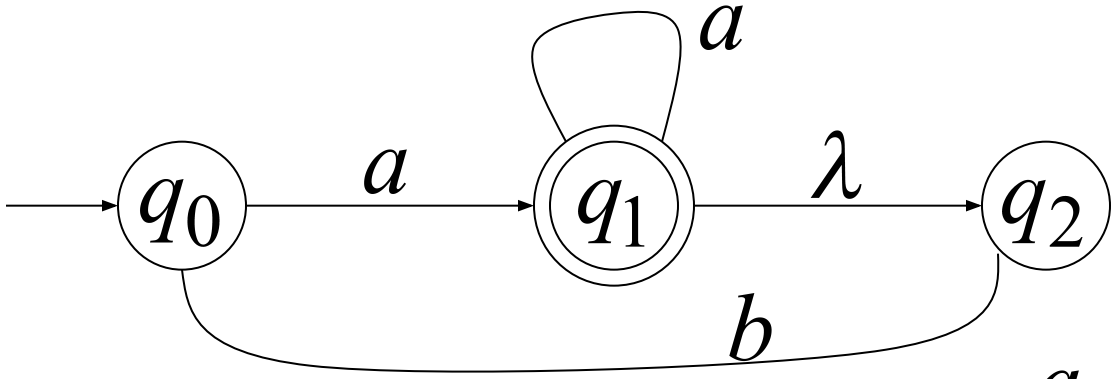


DFA

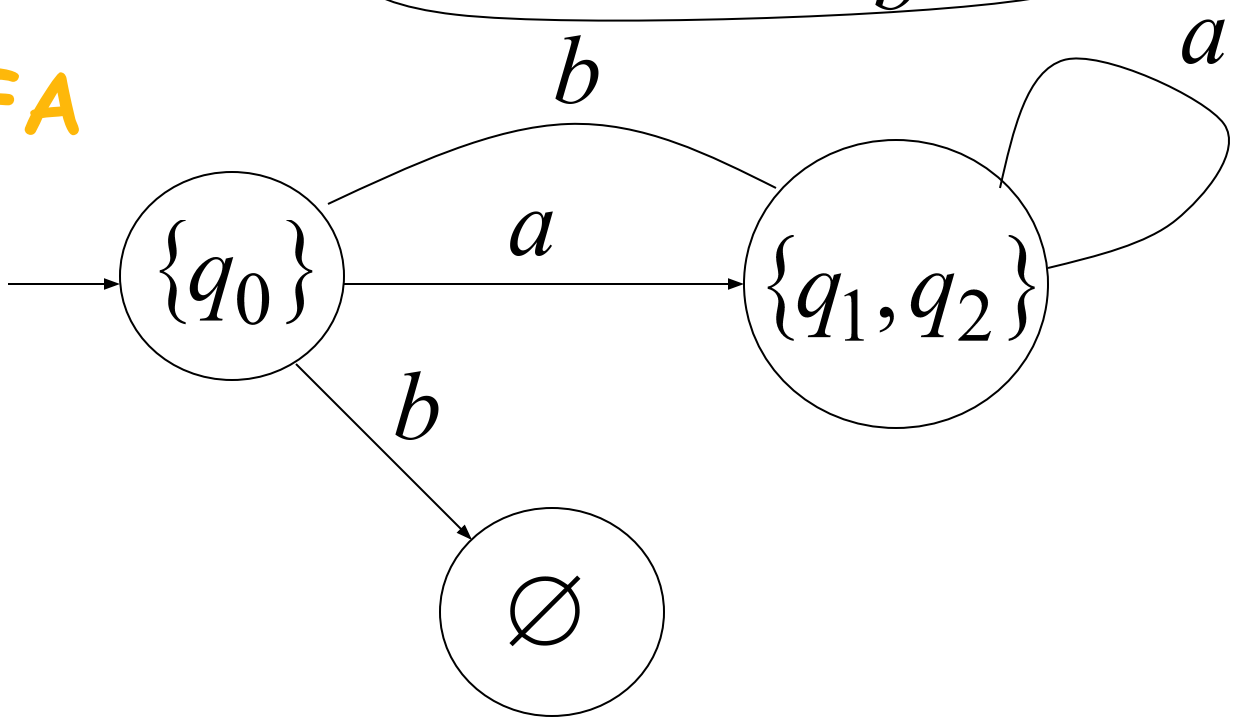


NFA to DFA

NFA

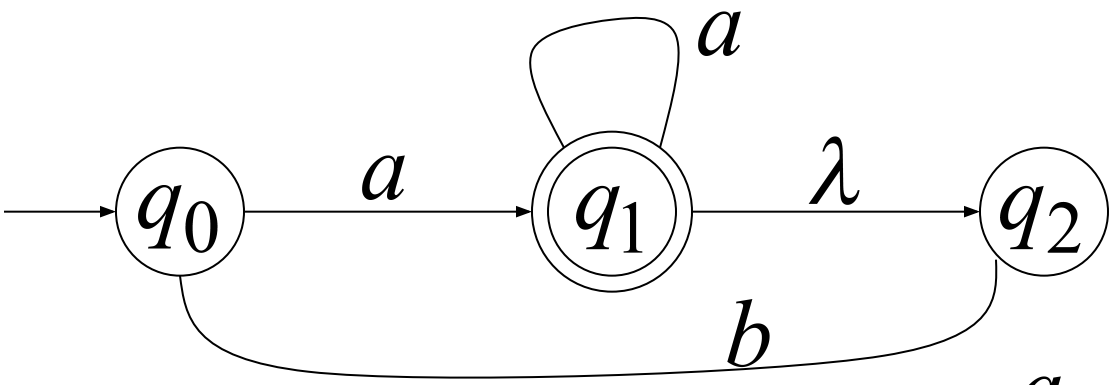


DFA

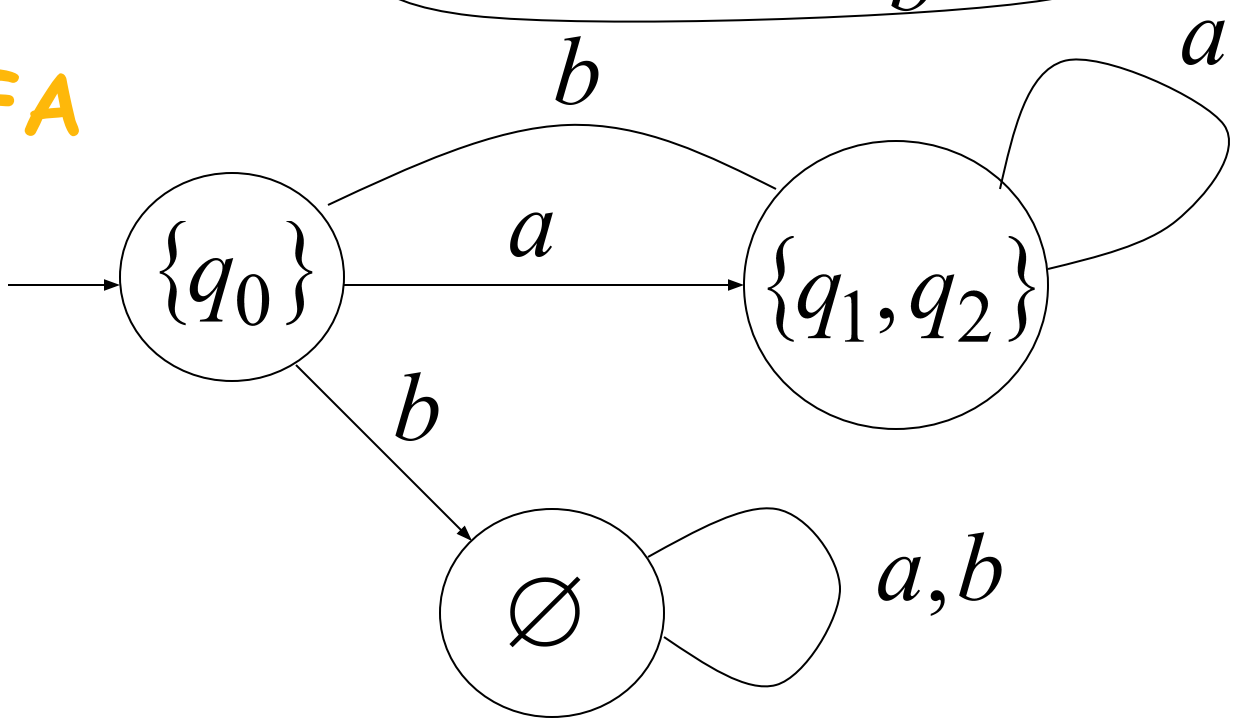


NFA to DFA

NFA



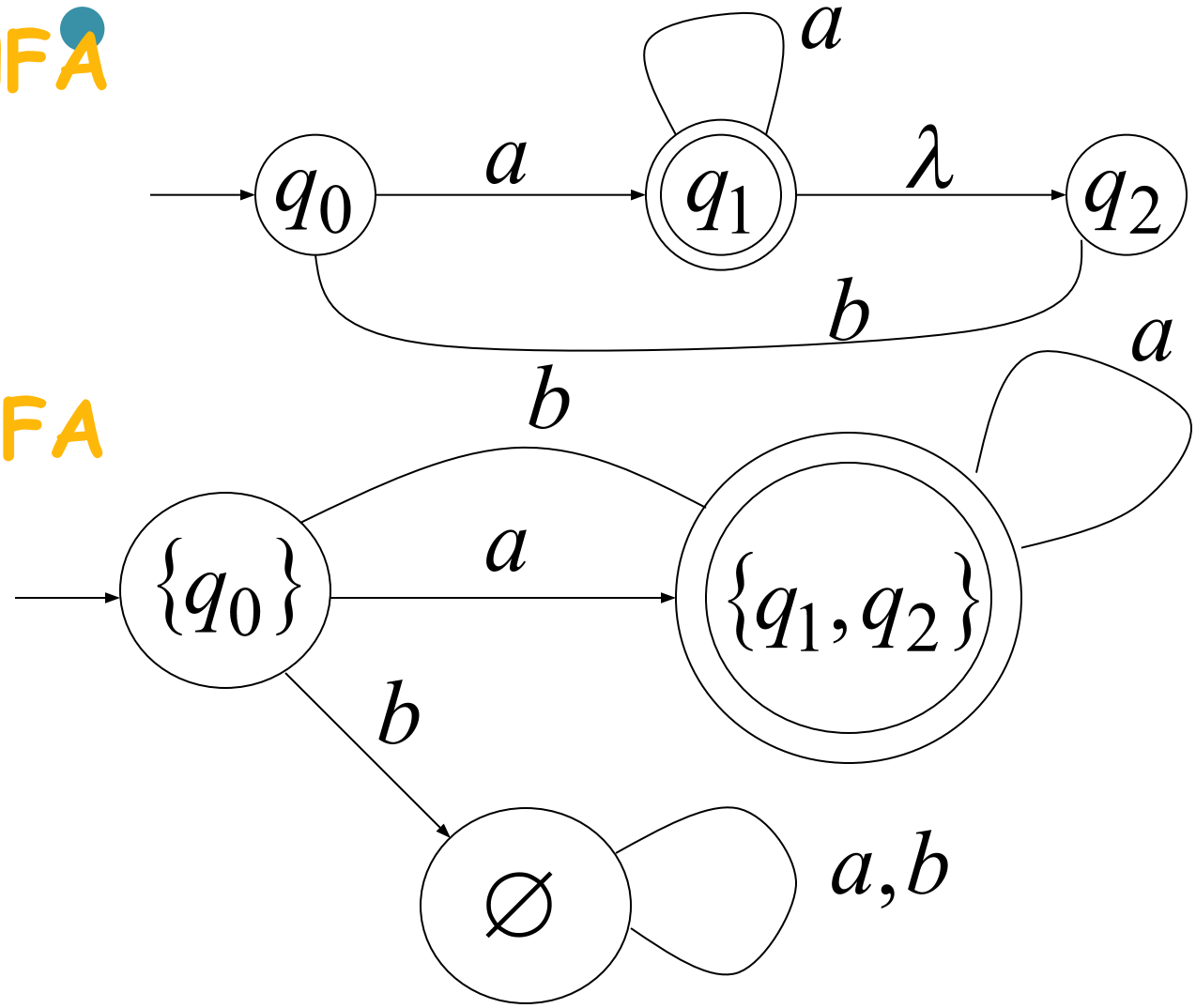
DFA



NFA to DFA

NFA

DFA



NFA to DFA: Remarks

- We are given an NFA M
- We want to convert it to an equivalent DFA M'

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

NFA to DFA

- If the NFA has states

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

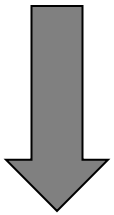
- the DFA has states in the power set

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

Procedure NFA to DFA

1. Initial state of NFA:

$$q_0$$

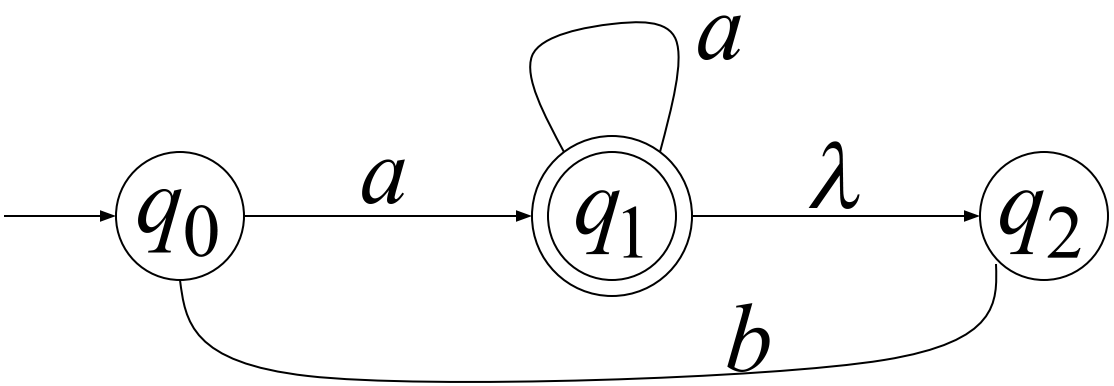


- Initial state of DFA:

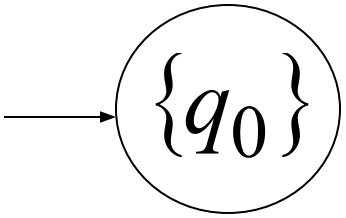
$$\{q_0\}$$

Example

NFA



DFA



Procedure NFA to DFA

2. For every DFA's state $\{q_i, q_j, \ldots, q_m\}$

- Compute in the NFA

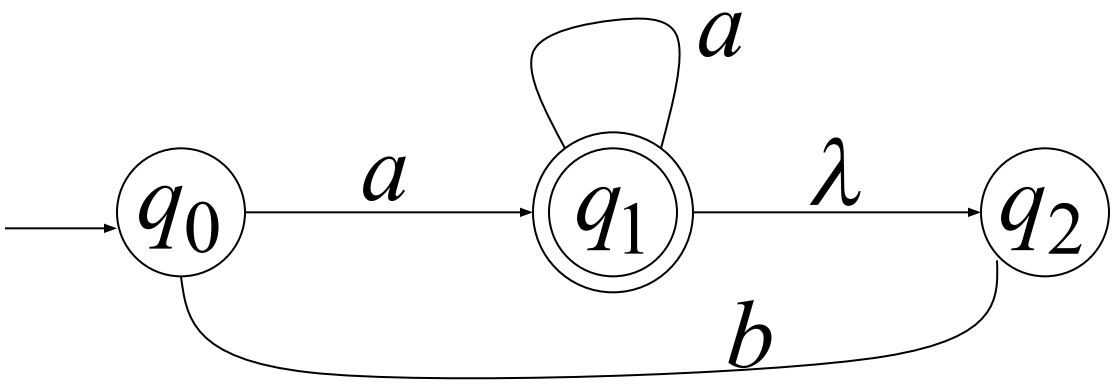
$$\left.\begin{array}{l} \delta^*(q_i, a), \\ \delta^*(q_j, a), \\ \ldots \end{array}\right\} = \{q'_i, q'_j, \ldots, q'_m\}$$

- Add transition

$$\delta(\{q_i, q_j, \ldots, q_m\}, \ a) = \{q'_i, q'_j, \ldots, q'_m\}$$

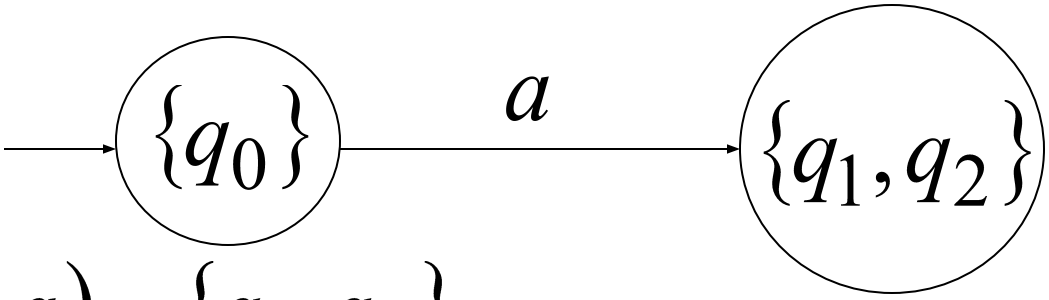
Example

NFA



$$\delta^*(q_0, a) = \{q_1, q_2\}$$

DFA



$$\delta(\{q_0\}, a) = \{q_1, q_2\}$$

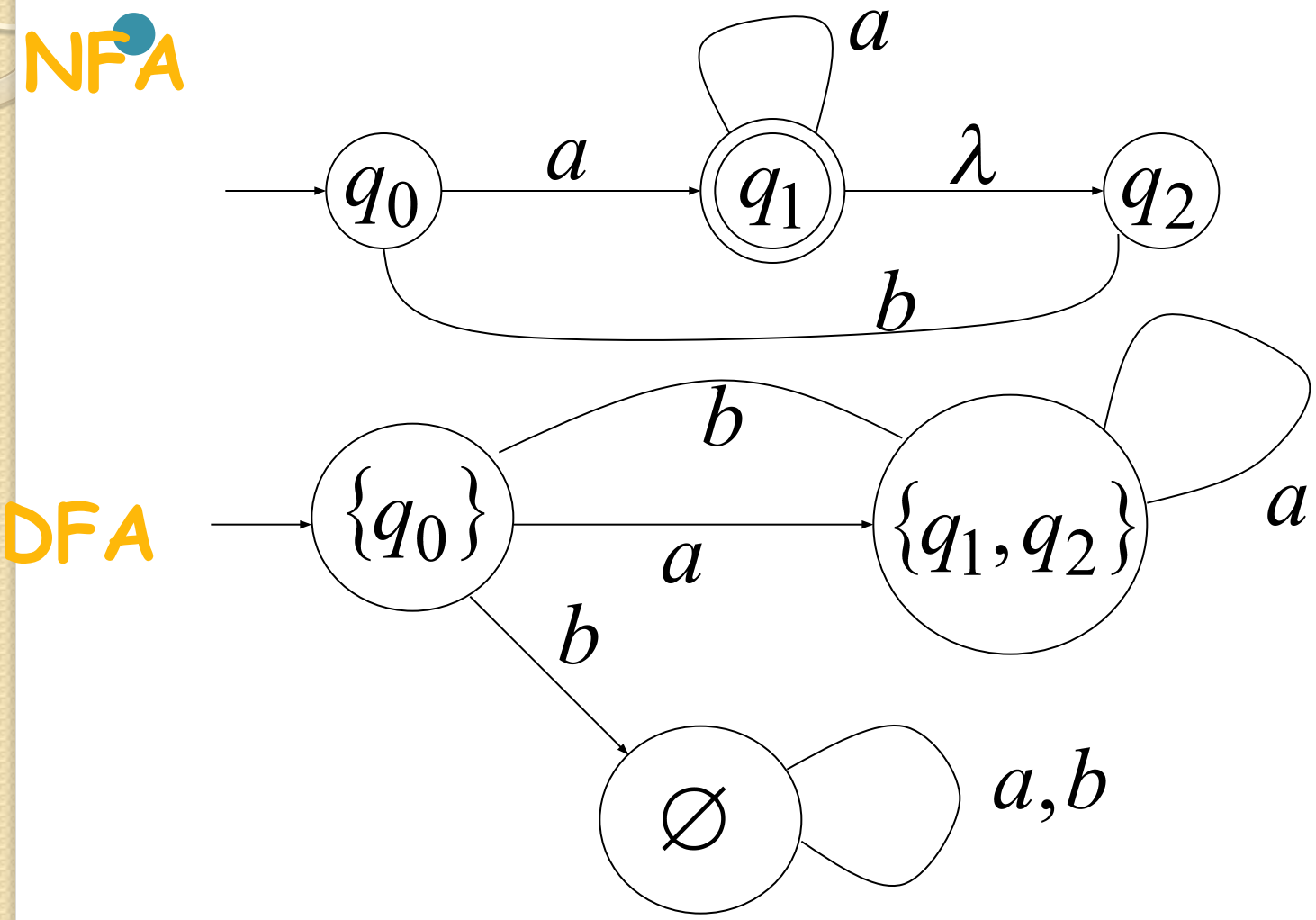
Procedure NFA to DFA

- Repeat Step 2 for all letters in alphabet, until
- no more transitions can be added.

Example

NFA

DFA



Procedure NFA to DFA $\{q_i, q_j, \ldots, q_m\}$

3. For any DFA state

If some q_j is a final state in the NFA

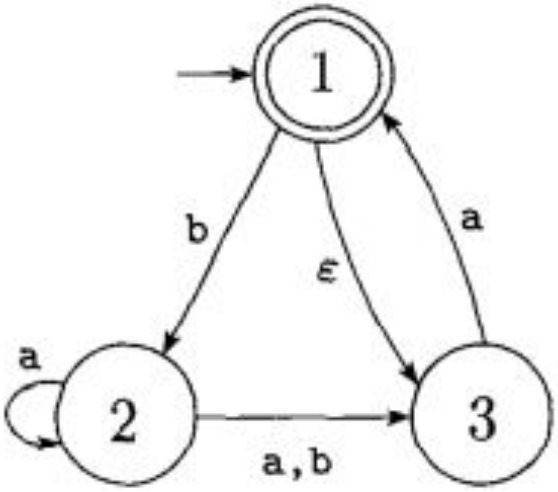
Then,

$\{q_i, q_j, \ldots, q_m\}$

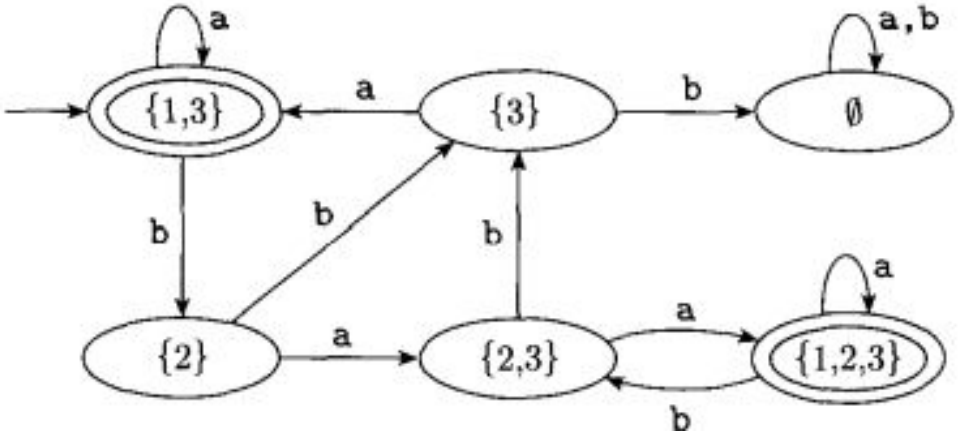
is a final state in the DFA

Example

- Convert the following NFA to the corresponding DFA.

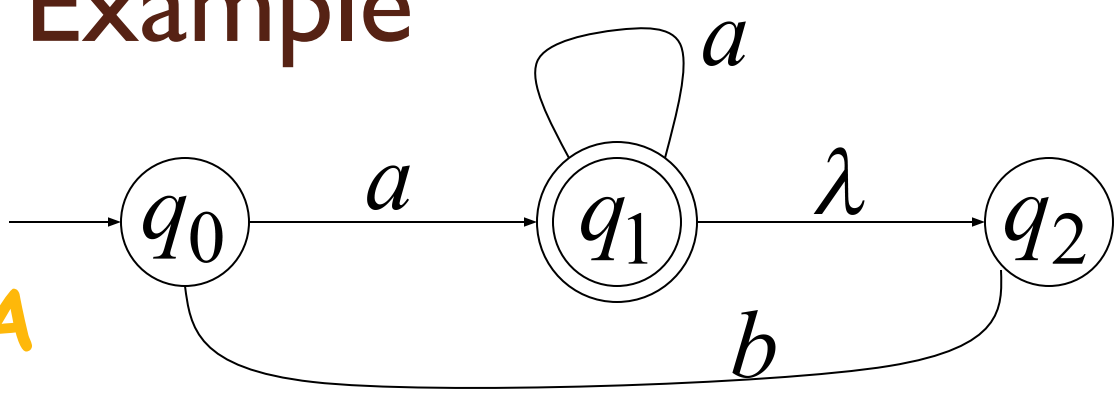


- The corresponding DFA IS:



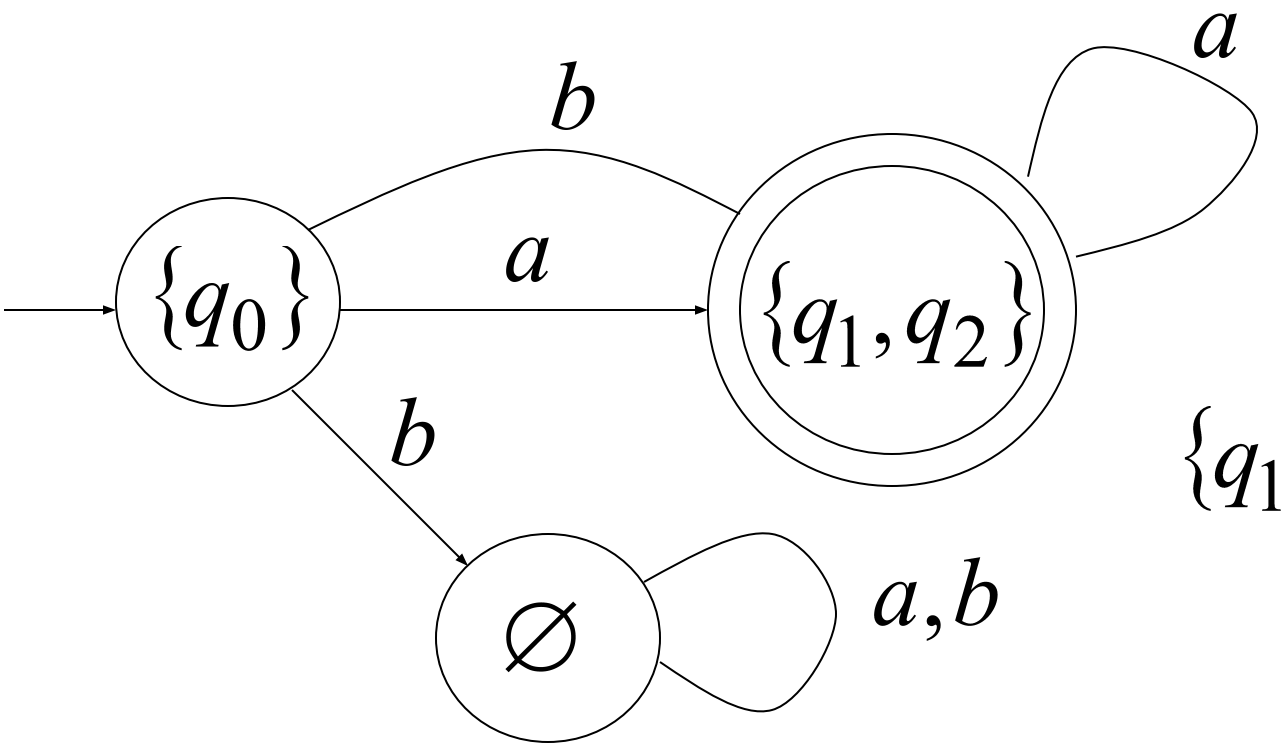
Example

NFA



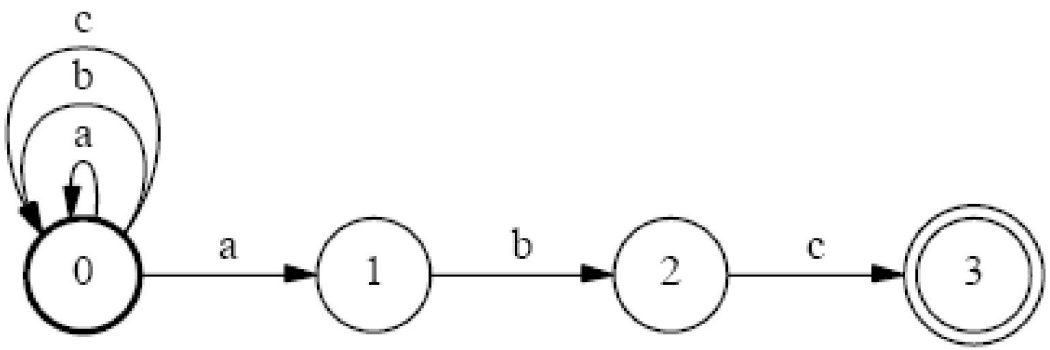
$q_1 \in F$

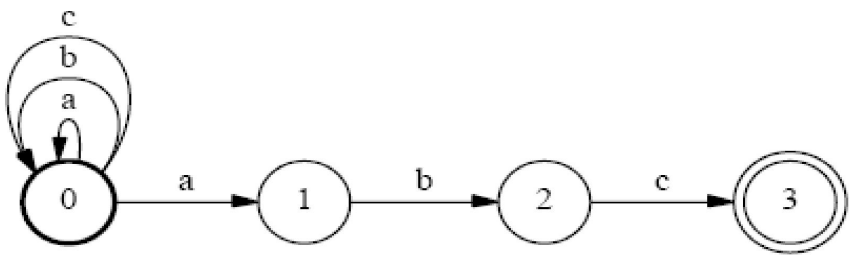
DFA

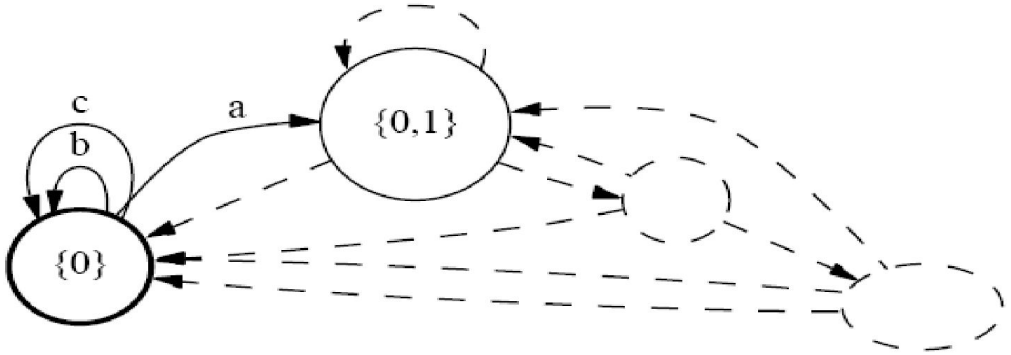


$\{q_1, q_2\} \in F'$

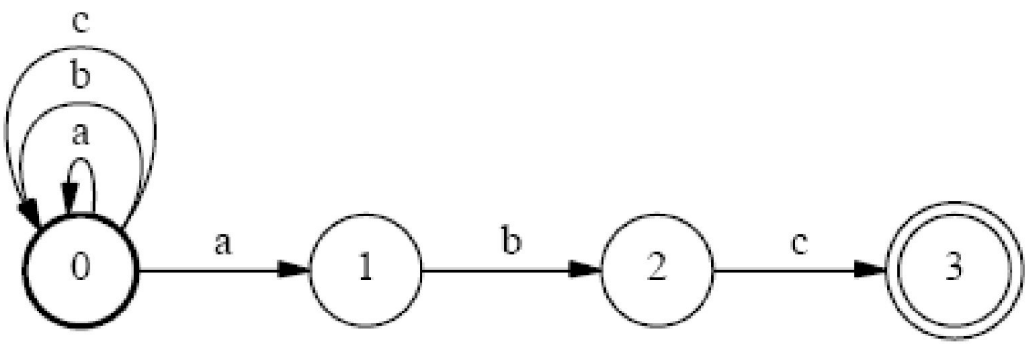
Convert the following NFA to DFA

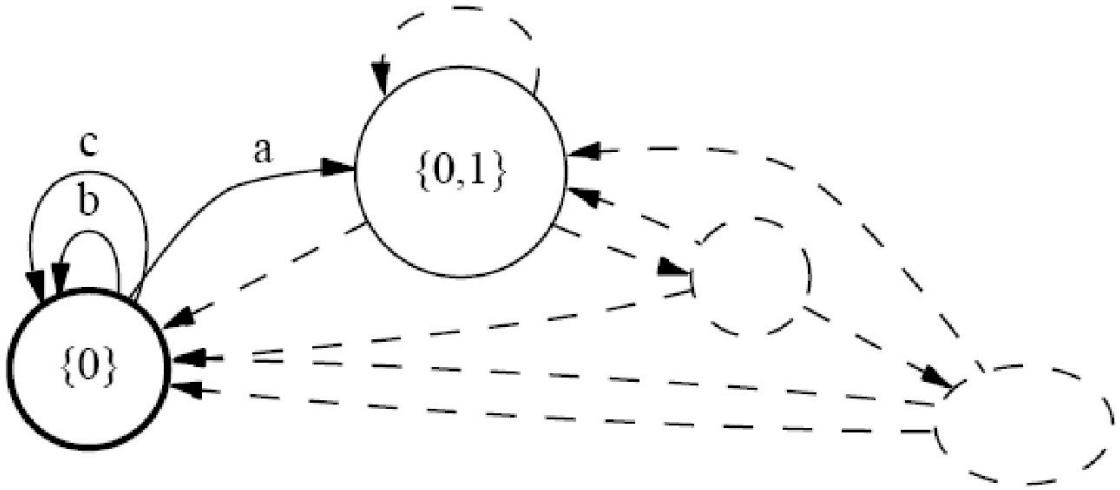




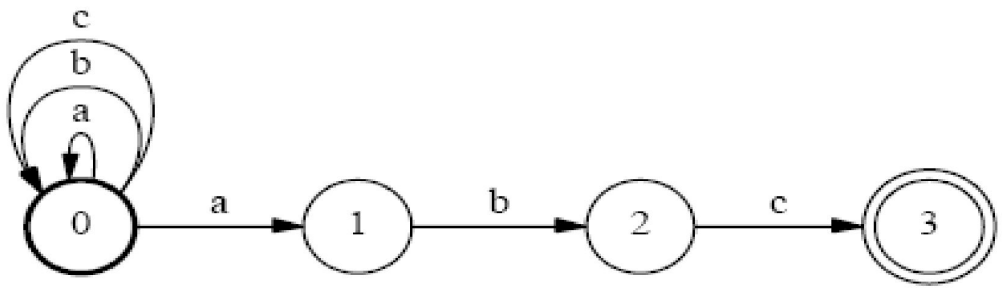


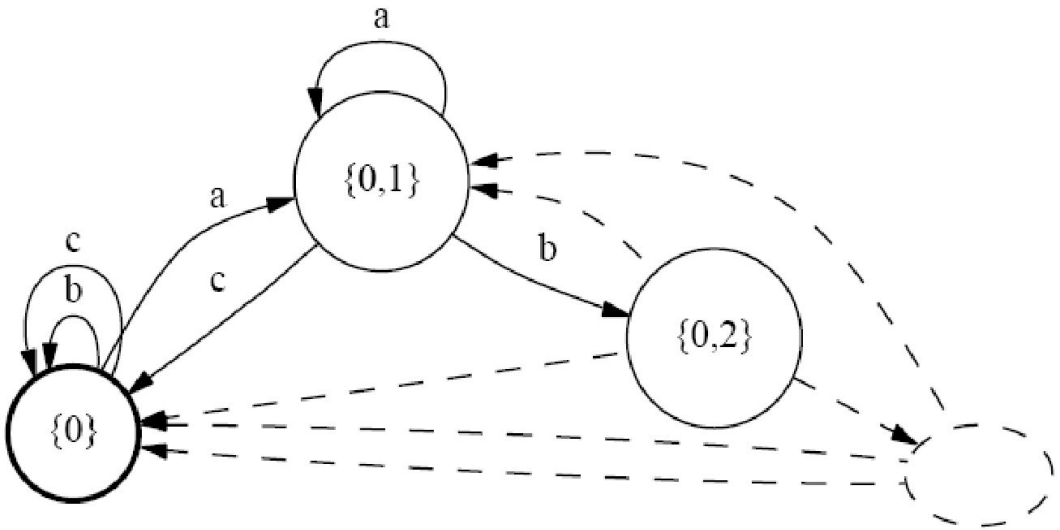
move({0},a) = move(0,a) = {0,1}
move({0},b) = move(0,b) = {0}
move({0},c) = move(0,c) = {0}



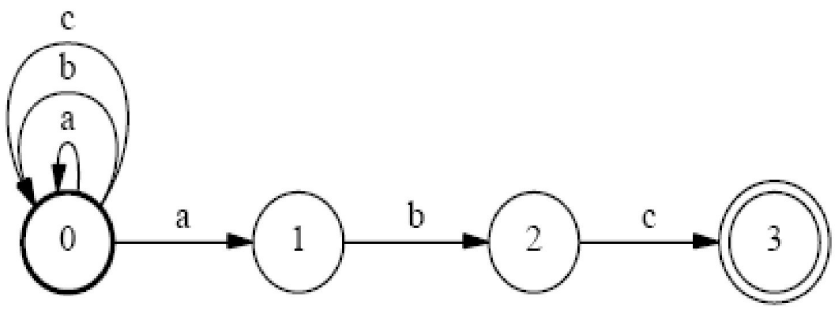


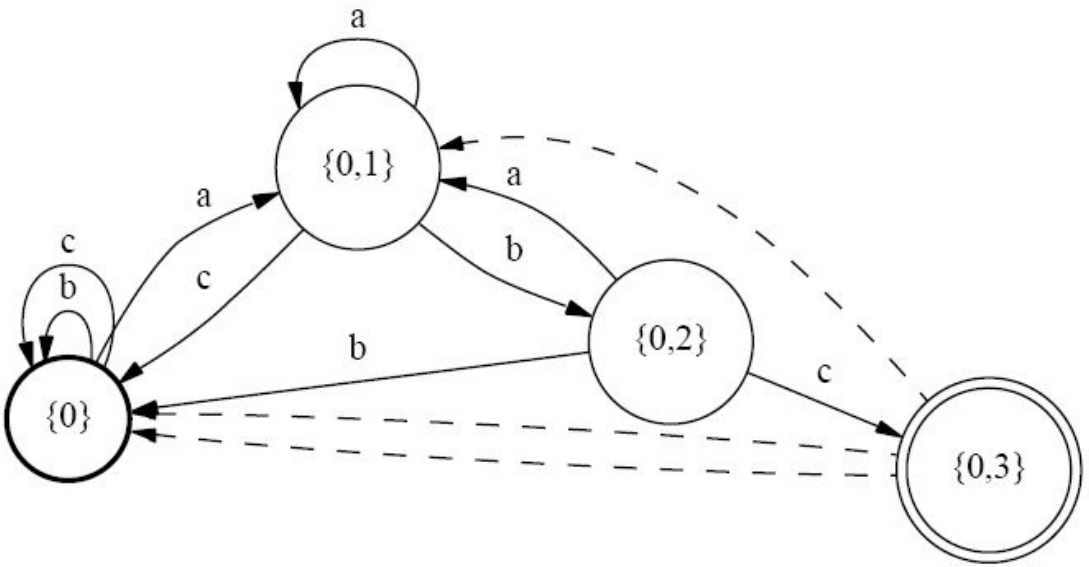
$$\begin{aligned}
\text{move}(\{0,1\},a) &= \text{move}(0,a)\cup\text{move}(1,a) = \{0,1\}\\
\text{move}(\{0,1\},b) &= \text{move}(0,b)\cup\text{move}(1,b) = \{0,2\}\\
\text{move}(\{0,1\},c) &= \text{move}(0,c)\cup\text{move}(1,c) = \{0\}
\end{aligned}$$



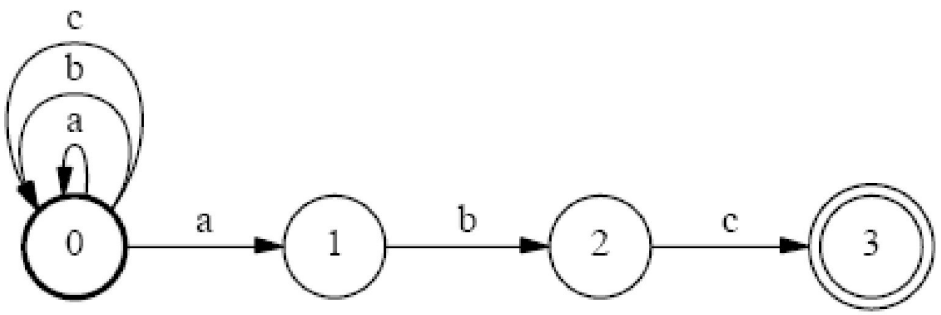


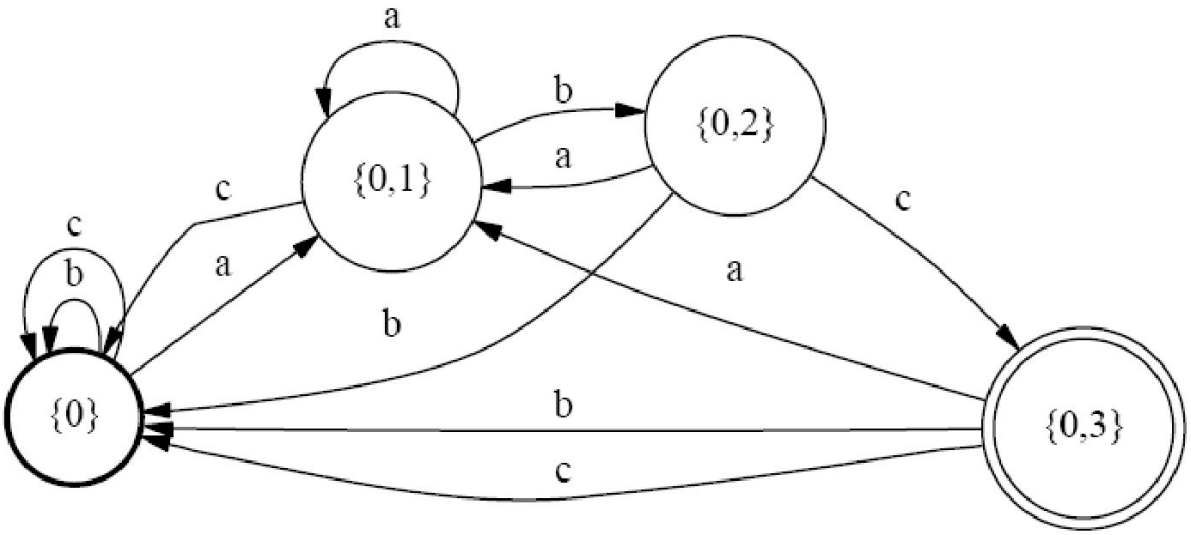
move({0,2},a) = move(0,a)∪move(2,a) = {0,1}
move({0,2},b) = move(0,b)∪move(2,b) = {0}
move({0,2},c) = move(0,c)∪move(2,c) = {0,3}



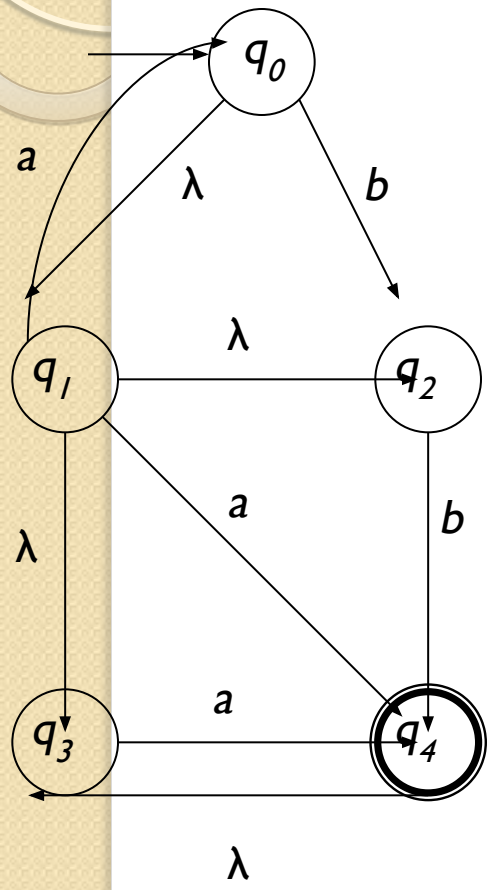


move({0,3},a) = move(0,a)∪move(3,a) = {0,1}
move({0,3},b) = move(0,b)∪move(3,b) = {0}
move({0,3},c) = move(0,c)∪move(3,c) = {0}


c
b
a
0
a
1
b
2
c
3

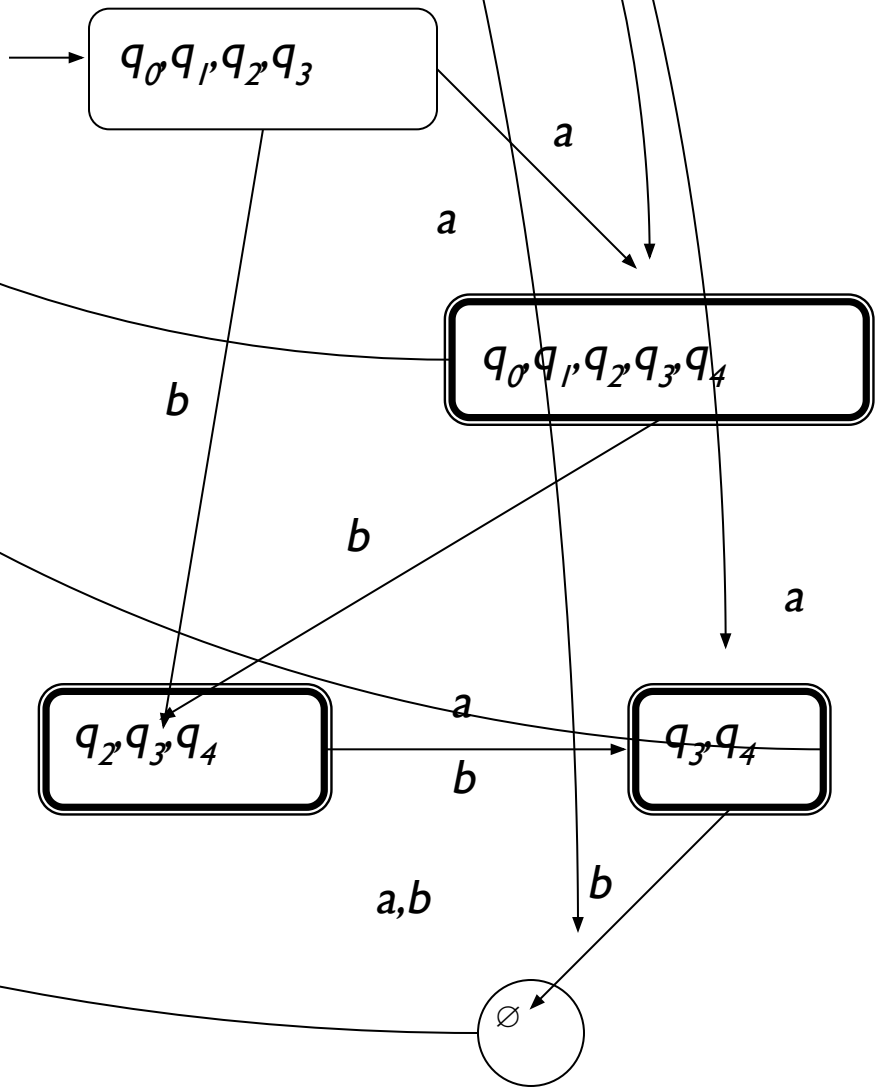

a
b
{0,2}
a
{0,1}
c
c
c
b
a
a
b
{0}
b
{0,3}
c

Find the corresponding DFA?

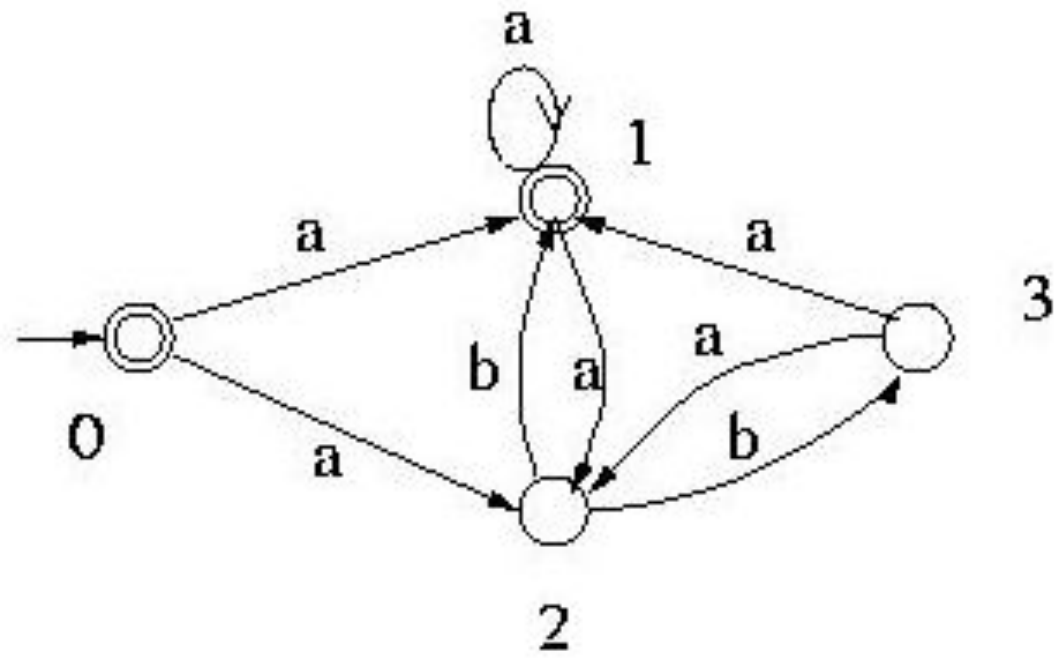


δ	q0	q1	q2	q3	q4
a	q0,q1, q2,q3, q4	q0,q1,q2, q3,q4	⌀	q3,q4	q3,q4
b	q2,q3, q4	q3,q4	q3,q4	⌀	⌀

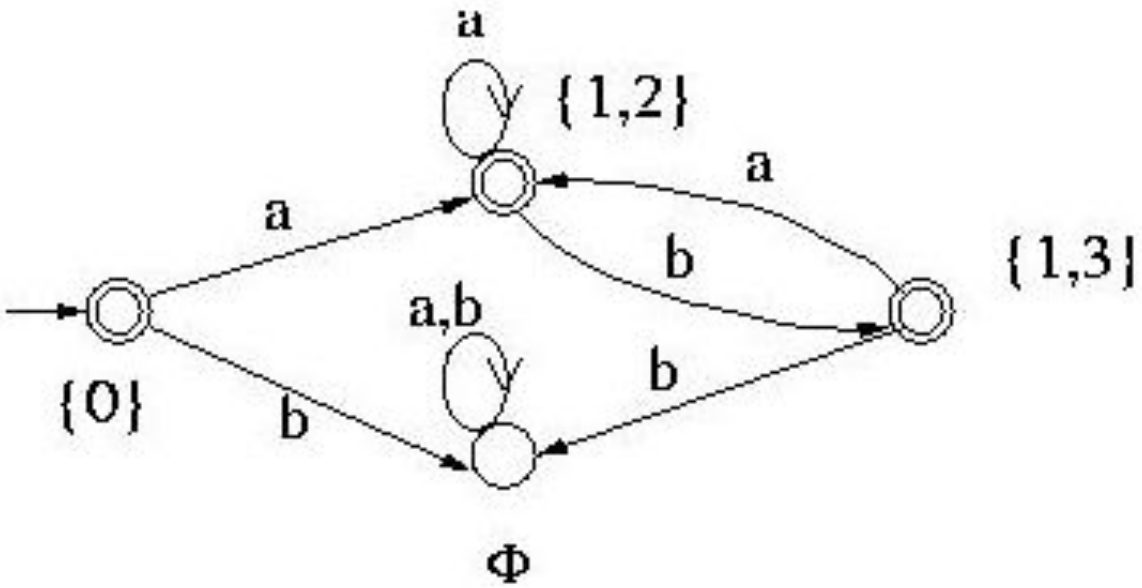
The answer



Convert the following NF to the corresponding DFA



Answer



Theorem

Take NFA M

- Apply procedure to obtain DFA M'

Then M and M' are equivalent :

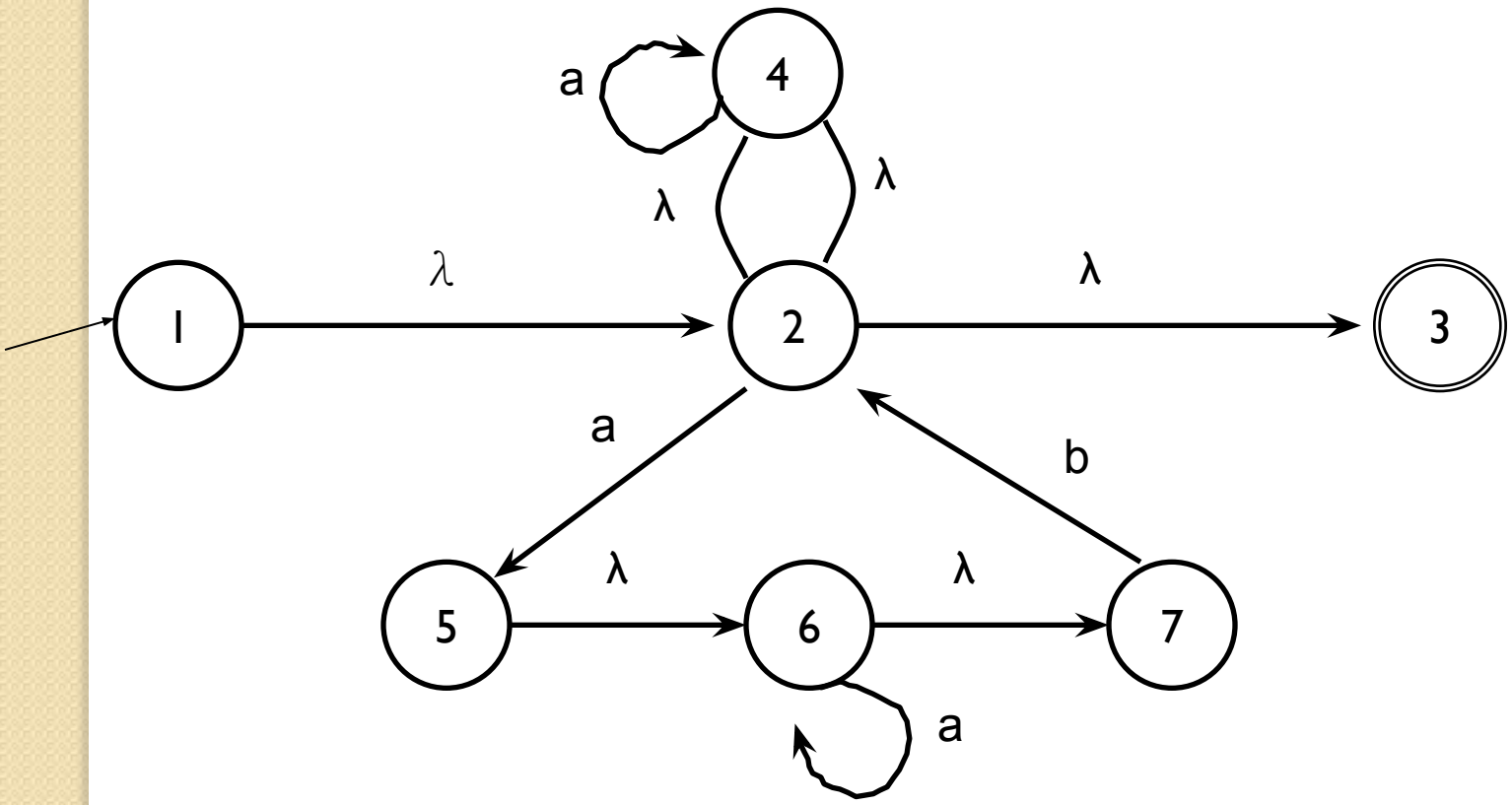
$$L(M) = L(M')$$

Finally

We have

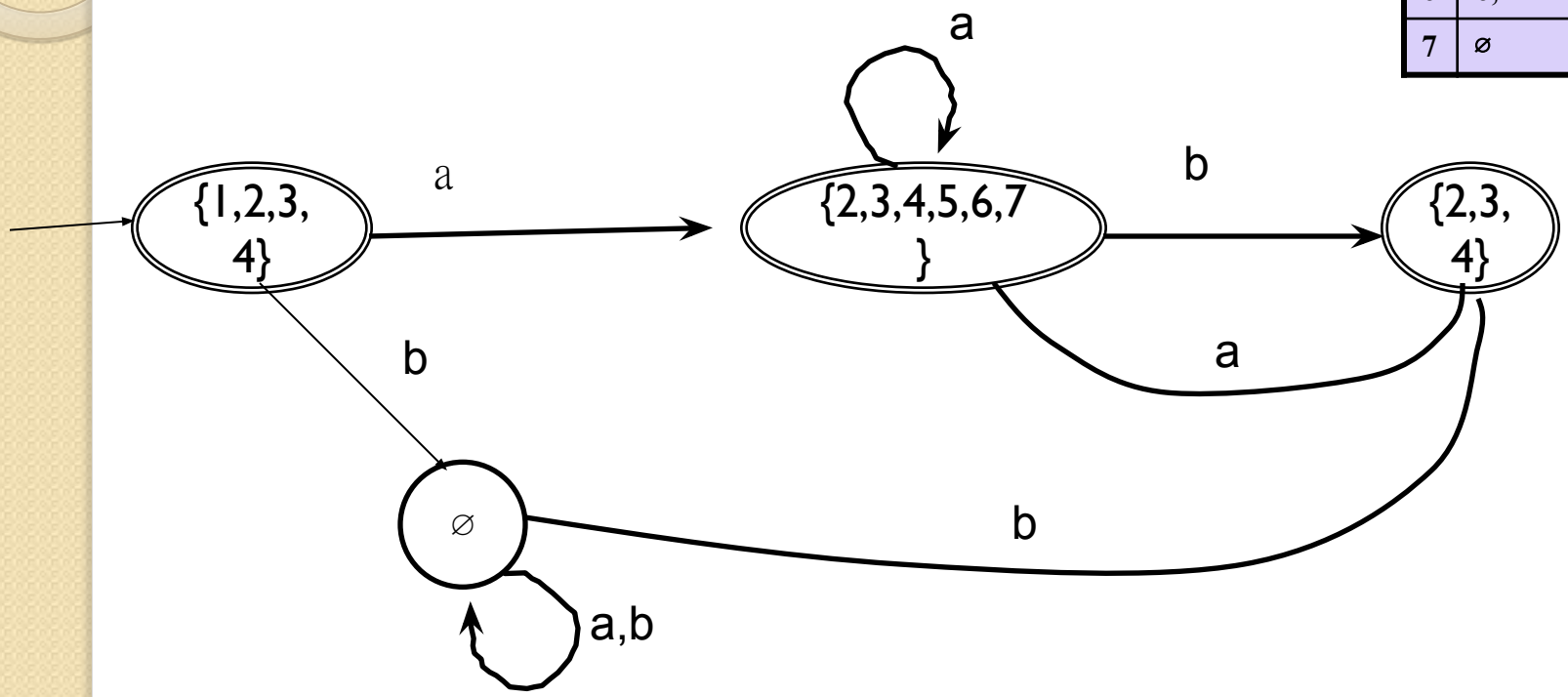
$$\left\{\begin{array}{l}\text{Languages}\\ \text{accepted}\\ \text{by NFAs}\end{array}\right\} = \left\{\begin{array}{l}\text{Languages}\\ \text{accepted}\\ \text{by DFAs}\end{array}\right\}$$

Quiz : convert the following NFA to the corresponding DFA



Answer

	a	b
1	2,3,4,5,6,7	∅
2	2,3,4,5,6,7	∅
3	∅	∅
4	2,3,4,5,6,7	∅
5	6,7	2,3,4
6	6,7	2,3,4
7	∅	2,3,4



CLOSURE UNDER THE REGULAR OPERATIONS

- The set of regular language is closed under, Union, Concatenation and Closure or star operations.

Properties of Regular Languages

For regular languages L_1 and L_2 we have

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: L_1^*

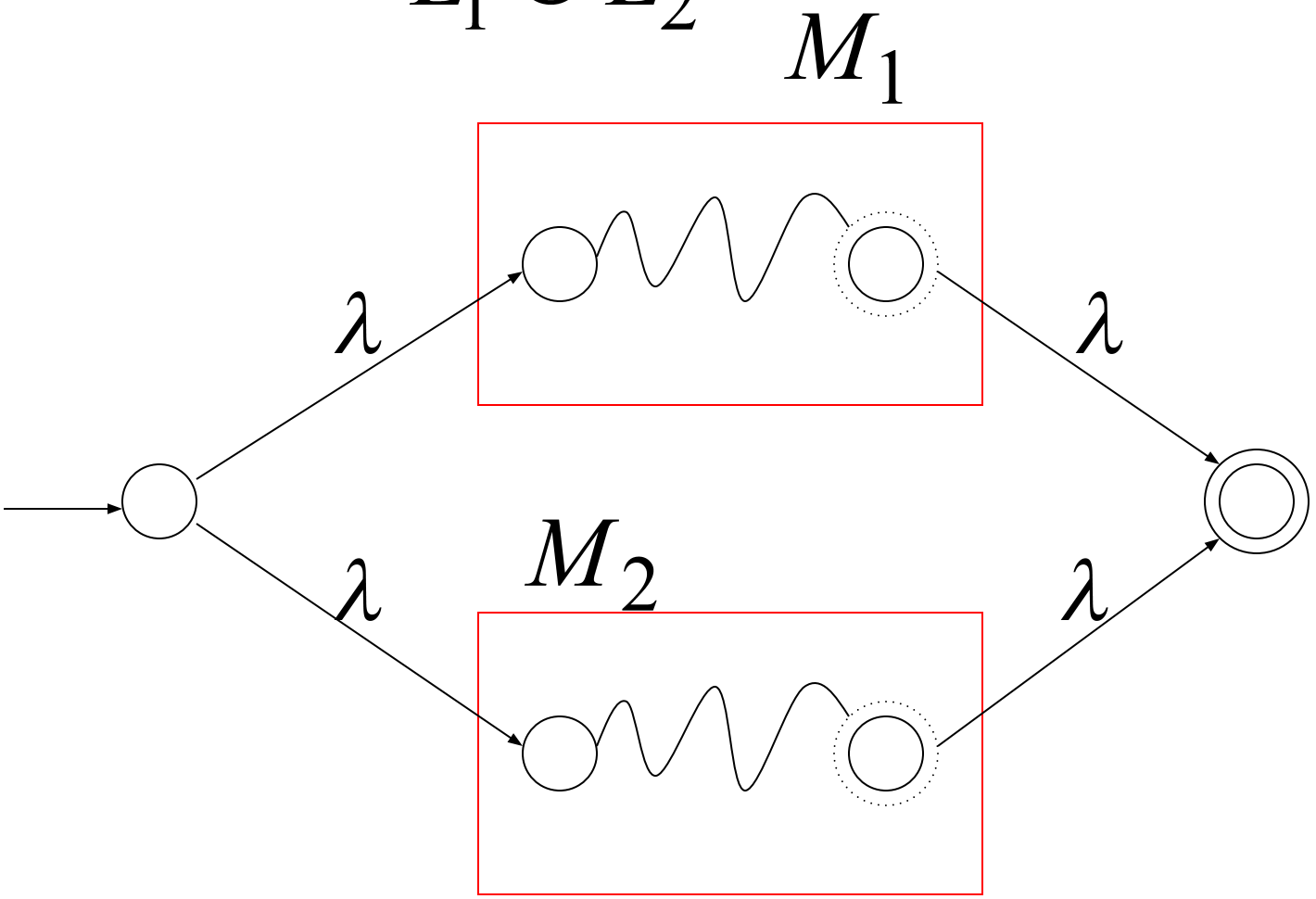
} Are regular Languages

The class of regular languages is closed under the union operation.

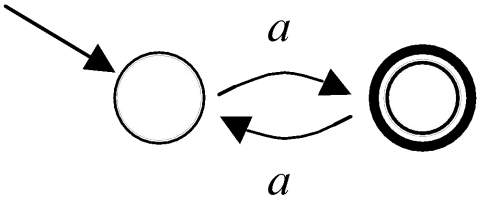
- Proof Idea: We have regular languages A1 and A2 and want to prove that Al U A2 is regular.
- The idea is to take two NFAs, N1 and N2 for A1 and A2, and combine them into one new NFA, N.
- Machine N must accept its input if either N1 or N2 accepts this input.
- The new machine has a new start state that branches to the start states of the old machines with λ arrows.
- In this way the new machine nondeterministically guesses which of the two machines accepts the input.
- If one of them accepts the input, N will accept it, too.

Union

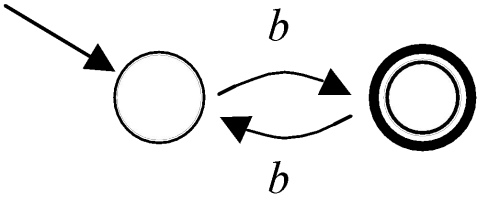
- NFA for $L_1 \cup L_2$



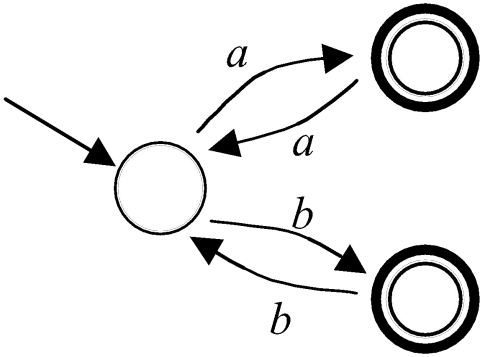
Incorrect Union



$A = \{a^n \mid n \text{ is odd}\}$

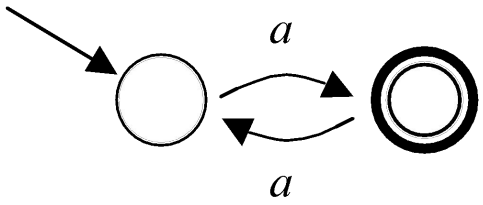


$B = \{b^n \mid n \text{ is odd}\}$

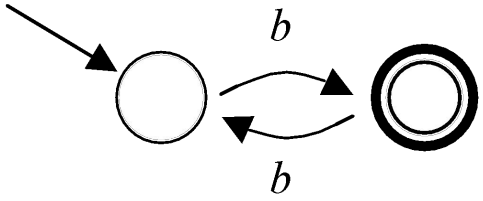


$A \cup B$?
No: this NFA accepts *aab*

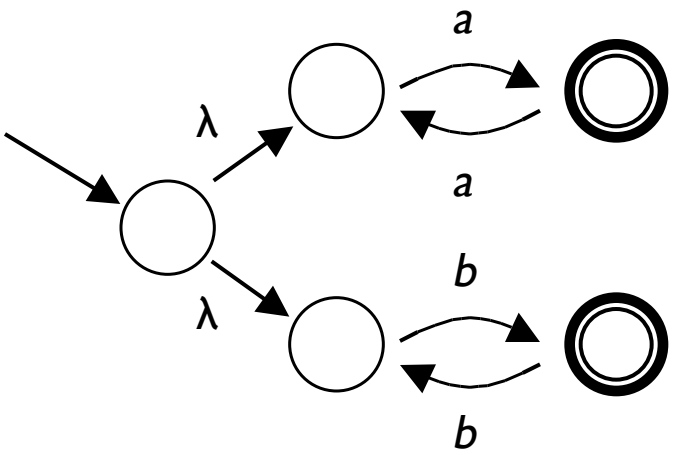
Correct Union



$A = \{a^n \mid n \text{ is odd}\}$



$B = \{b^n \mid n \text{ is odd}\}$

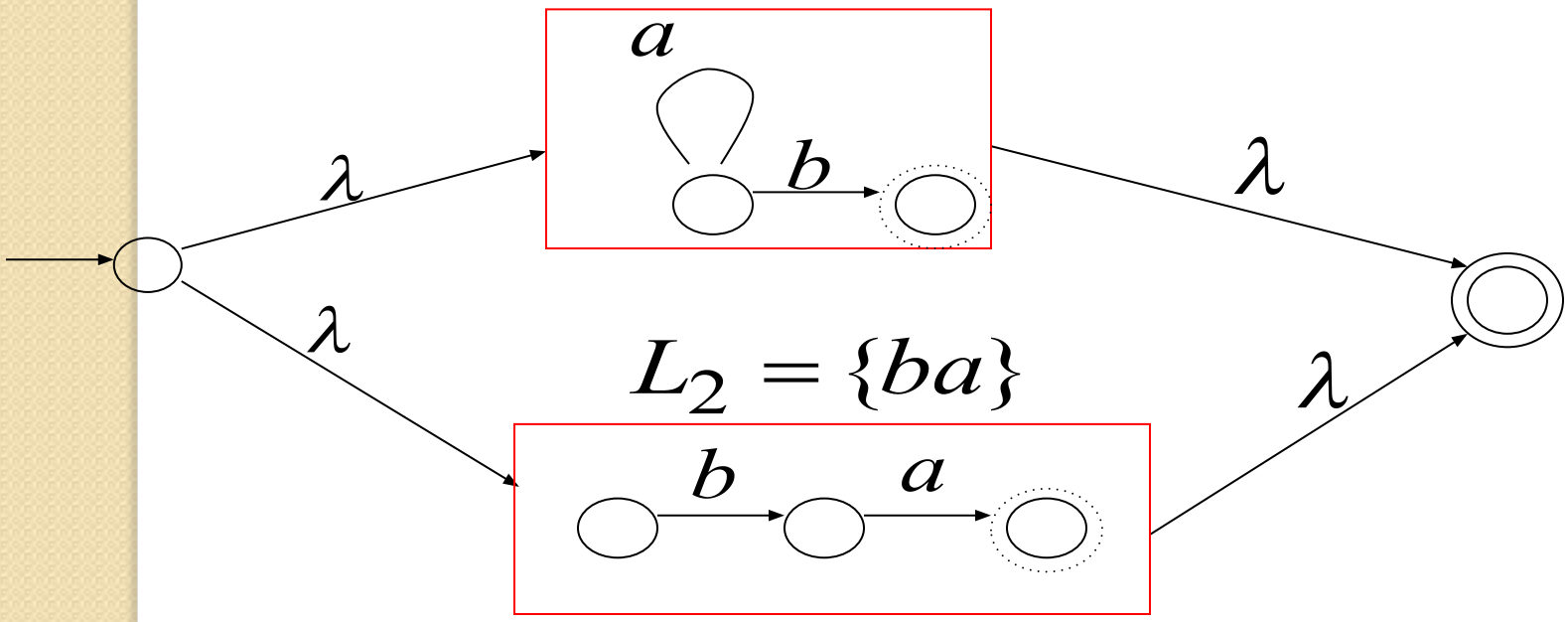


$A \cup B$

Example

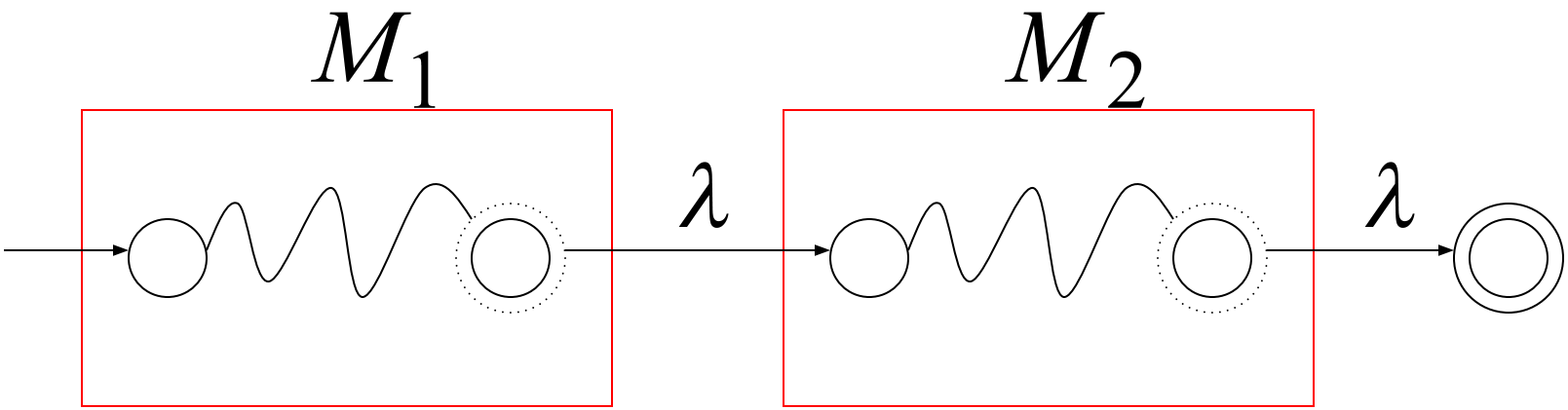
NFA for $L_1 \cup L_2 = \{a^n b\} \cup \{ba\}$

$L_1 = \{a^n b\}$



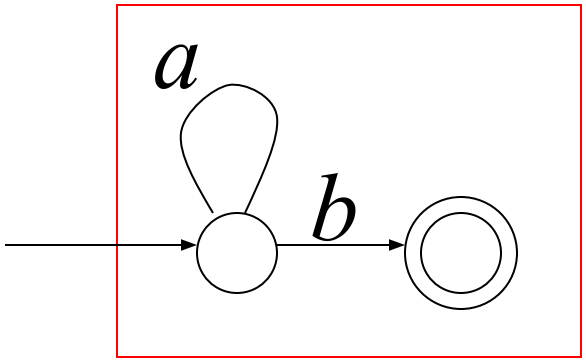
The class of regular languages is closed under the concatenation operation.

- NFA for $L_1 L_2$



Example

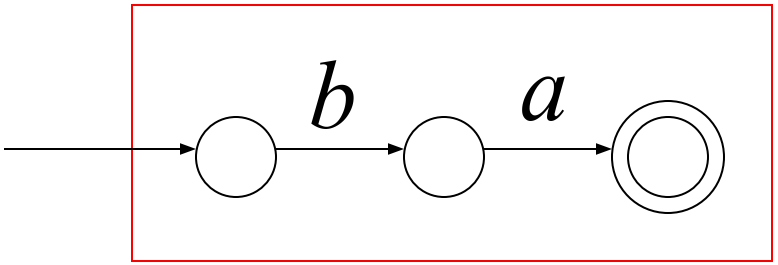
$L_1 = \{a^n b\}$



M_1

M_2

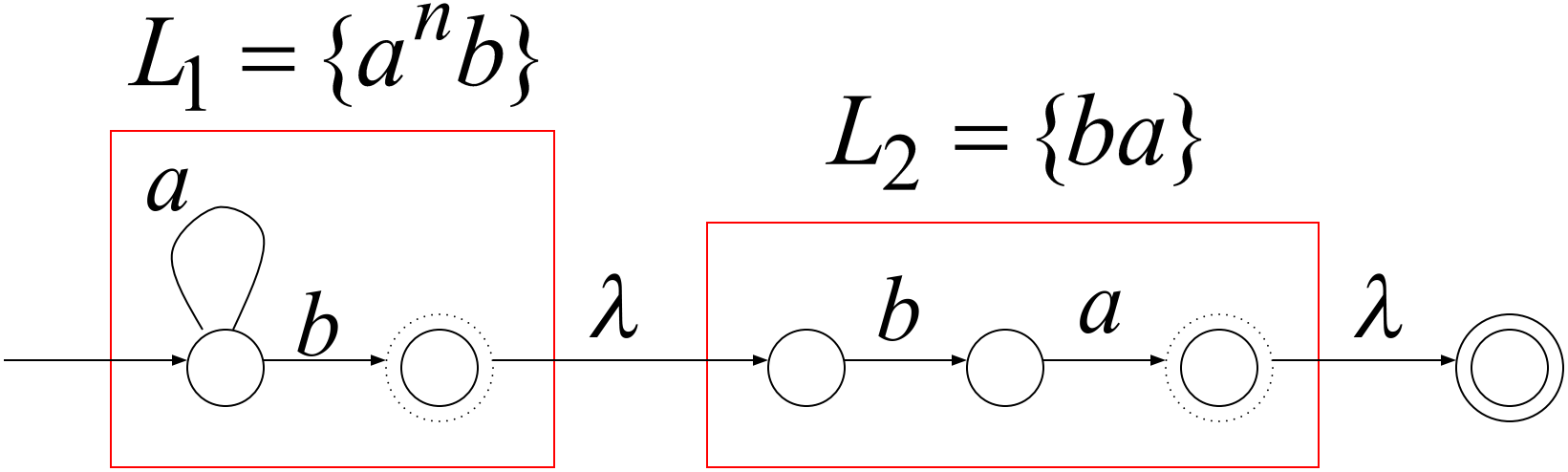
$L_2 = \{ba\}$



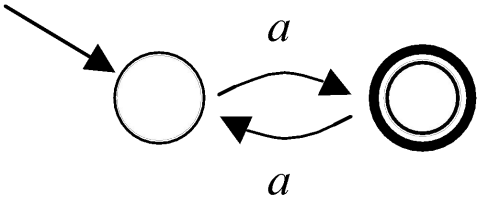
Example

- NFA for

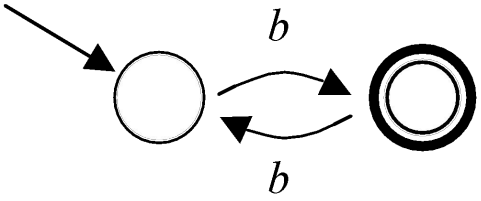
$$L_1 L_2 = \{a^n b\}\{ba\} = \{a^n bba\}$$



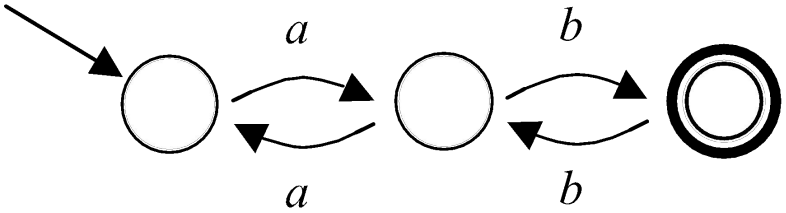
Incorrect Concatenation



$A = \{a^n \mid n \text{ is odd}\}$



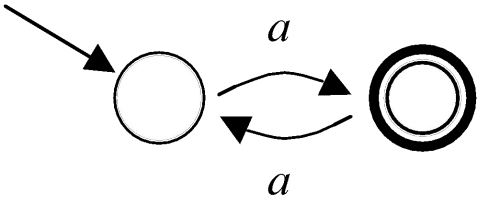
$B = \{b^n \mid n \text{ is odd}\}$



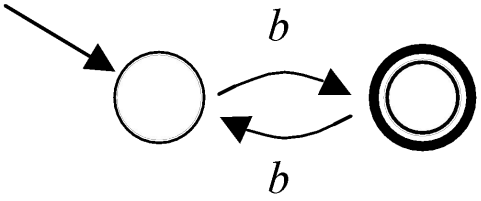
$\{xy \mid x \in A \text{ and } y \in B\}$?

No: this NFA accepts *abbaab*

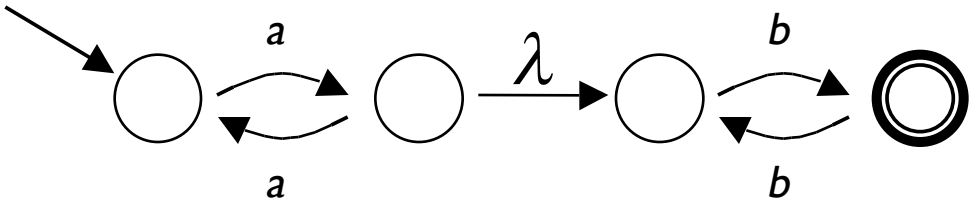
Correct Concatenation



$A = \{a^n \mid n \text{ is odd}\}$



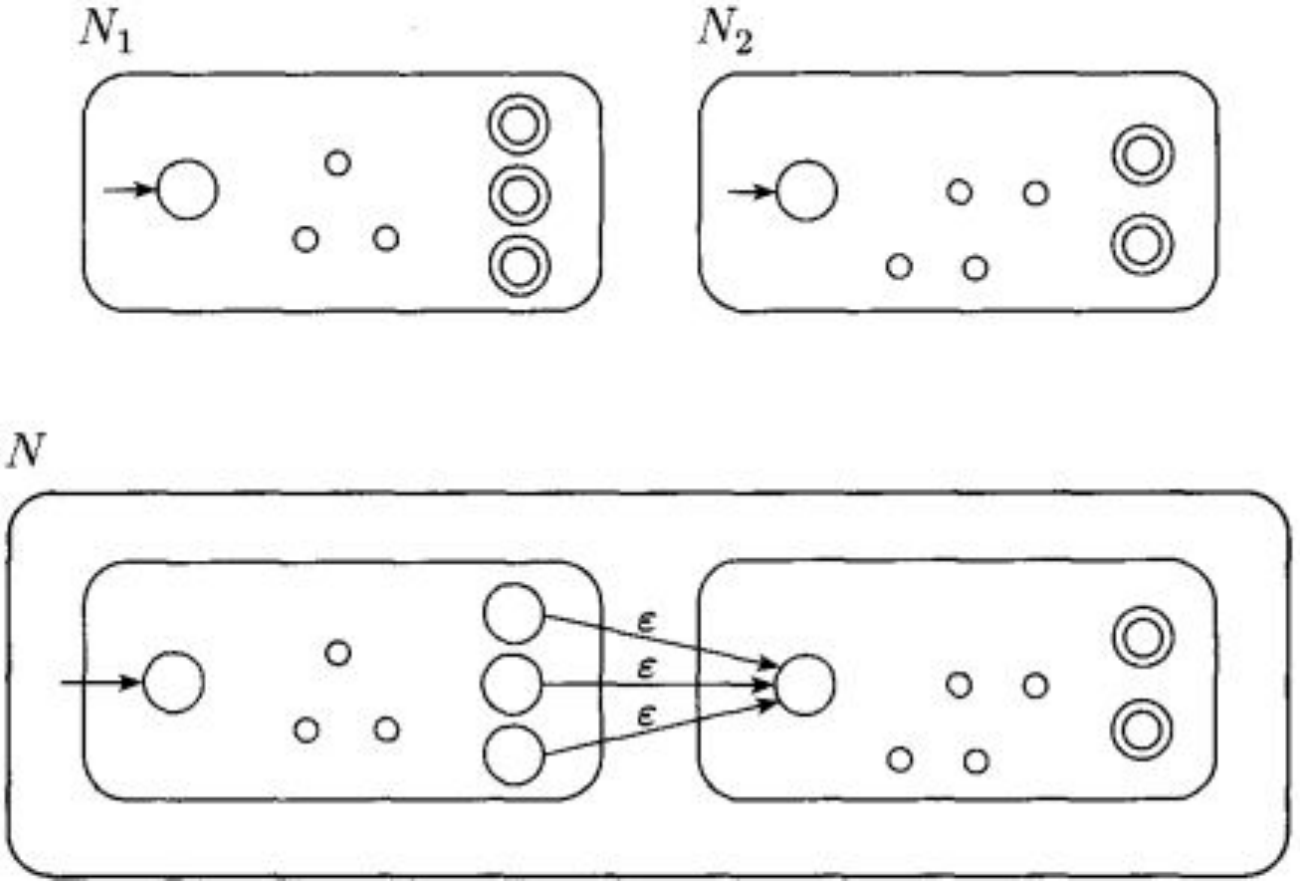
$B = \{b^m \mid m \text{ is odd}\}$



$\{xy \mid x \in A \text{ and } y \in B\}$

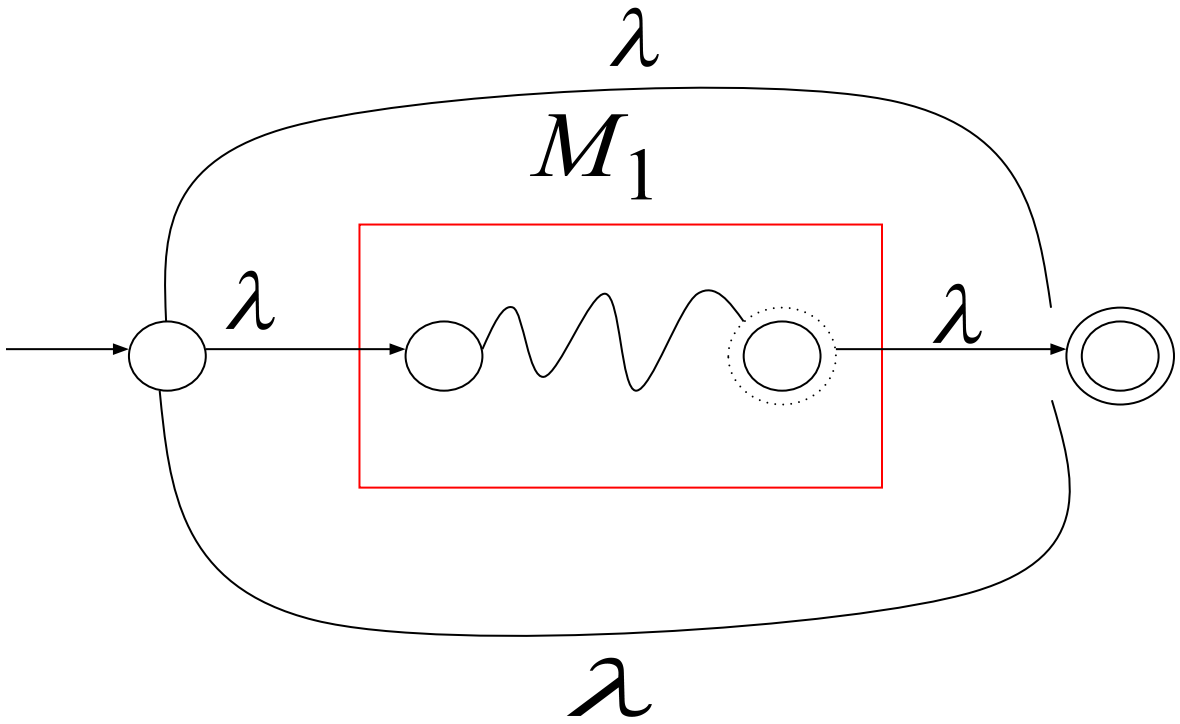
Concatenation Operation

- If the first FA contains more than one final state.



The class of regular languages is closed under the Star operation.

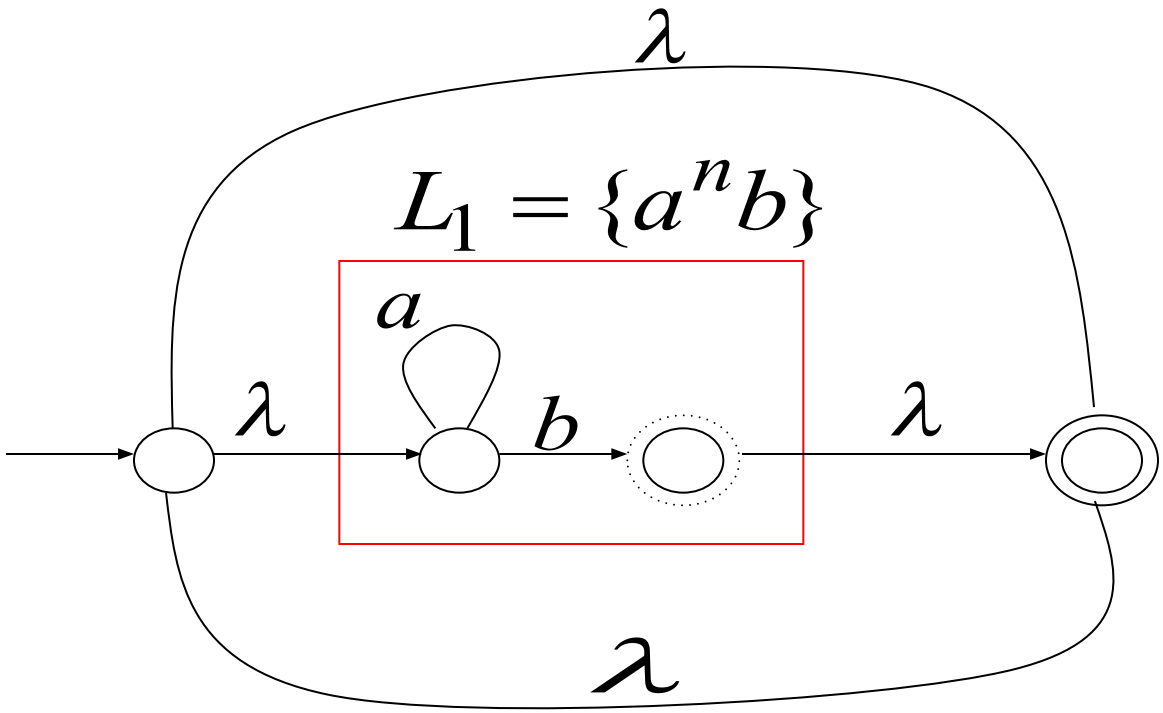
- NFA for $L_1 *$



$\lambda \in L_1 *$

Example

- NFA for $L_1^* = \{a^n b\}^*$



Regular Expressions

A regular language is a language that can be defined by a regular expression. A regular expression is often described by means of an algebraic expression called a regular expression notation .

Definition: Let Σ be a given alphabet. Then

1- $\varnothing$, λ and a $\in \Sigma$ are regular expression. These are called primitive regular expression.

2- If r1 and r2 are regular expressions, so are r1 + r2, r1 . r2, r1*, and (r1).

3- A string is a regular expression if and only if it can be derived from the primitive regular expression by a finite number of applications of the rules in 2.

Examples

A regular expression: $(a + b \cdot c)^* \cdot (c + \varnothing)$

Not a regular expression: $(a + b +)$

Languages of Regular Expressions

- $L(r)$: language of regular expression r

- Example

$$L((a+b\cdot c)^*)=\{\lambda,a,bc,aa,abc,bca,\ldots\}$$

Definition

For primitive regular expressions:

$$L(\varnothing) = \varnothing$$

$$L(\lambda) = \{\lambda\}$$

$$L(a) = \{a\}$$

Recursive Definition

Primitive regular expressions: $\emptyset$, λ, a

Given regular expressions r_1 and r_2

$r_1 + r_2$

$r_1 \cdot r_2$

r_1^*

(r_1)

} Are regular expressions

Regular Expressions

- Regular expressions
  - describe regular languages

- Example:

  describes the language $(a + b \cdot c)^*$

$$\{a, bc\}^* = \{\lambda, a, bc, aa, abc, bca, \ldots\}$$

Definition (continued)

- For regular expressions r_1 and r_2

$$L(r_1 + r_2) = L(r_1) \cup L(r_2)$$

$$L(r_1 \cdot r_2) = L(r_1)\, L(r_2)$$

$$L(r_1 *) = (L(r_1))*$$

$$L((r_1)) = L(r_1)$$

Example

- Regular expression: $(a+b)\cdot a^*$

$$
\begin{aligned}
L((a+b)\cdot a^*) &= L((a+b))\,L(a^*) \\
&= L(a+b)\,L(a^*) \\
&= (L(a)\cup L(b))\,(L(a))^* \\
&= (\{a\}\cup\{b\})\,(\{a\})^* \\
&= \{a,b\}\,\{\lambda,a,aa,aaa,\ldots\} \\
&= \{a,aa,aaa,\ldots,b,ba,baa,\ldots\}
\end{aligned}
$$

Example

- Regular expression $r = (aa)^*(bb)^* b$

$$L(r) = \{a^{2n} b^{2m} b : \quad n, m \geq 0\}$$

Question

- Are there any equivalence between regular expressions and FSM???