More Finite Automata

Formal Languages and Abstract Machines

Week 03

Baris E. Suzek, PhD

Outline

- Last time
- Equivalence of Machines
- Regular Language Properties
- Regular expressions

Deterministic Finite Accepter (DFA)

$$M = (Q, \Sigma, \delta, q_0, F)$$

Q : set of states

Σ : input alphabet

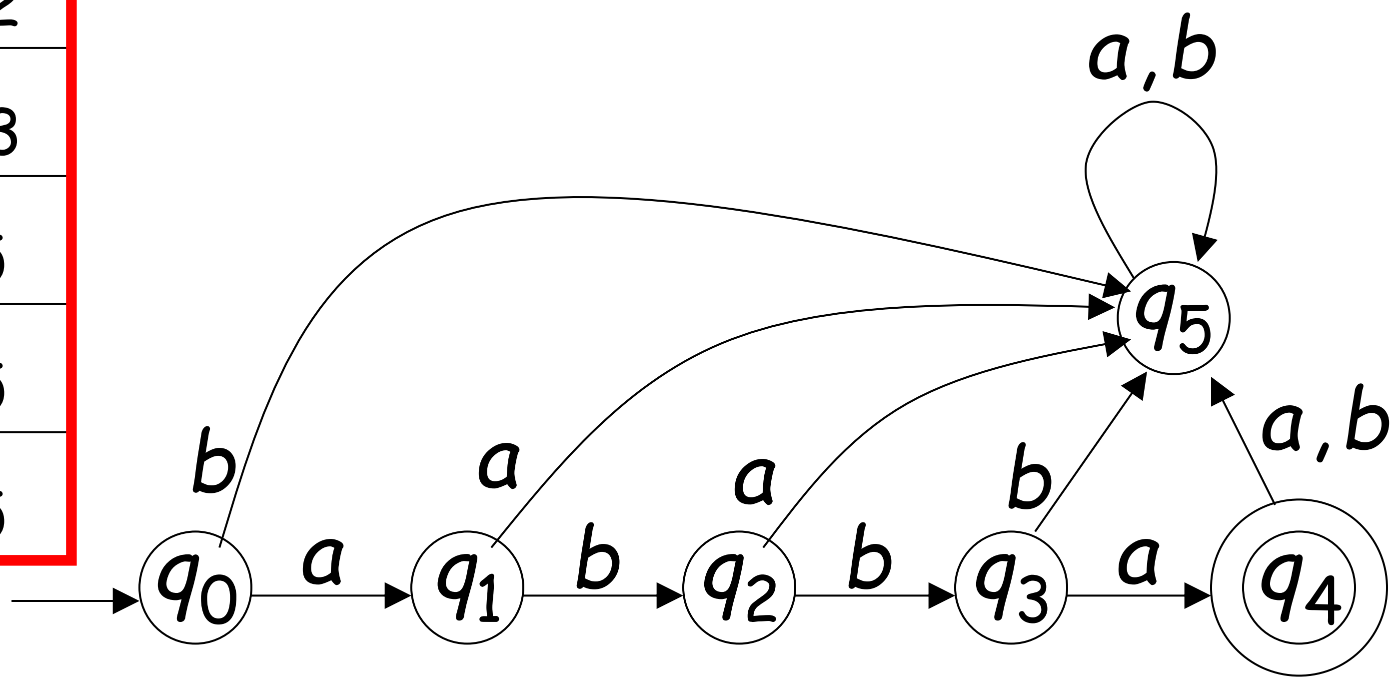
δ : transition function is a total function <u>there must be an action defined for every combination of state and symbol</u>

q_0 : initial state

F : set of final states

Transition Function δ

δ	a	b
q_0	q_1	q_5
q_1	q_5	q_2
q_2	q_5	q_3
q_3	q_4	q_5
q_4	q_5	q_5
q_5	q_5	q_5



Regular Languages

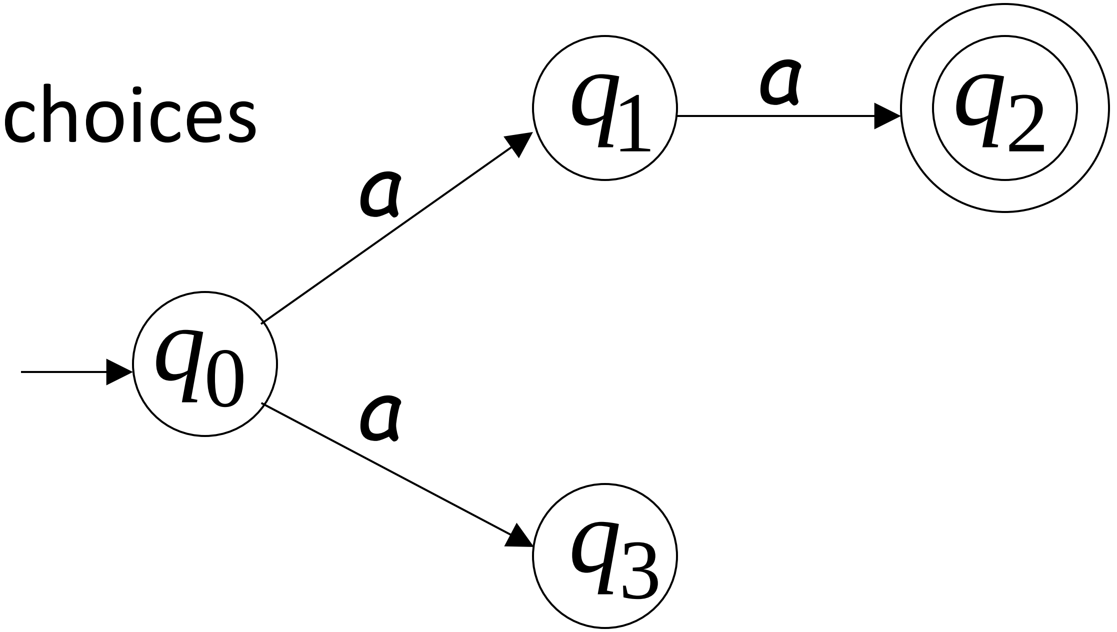
- A language L is regular if there is a DFA M such that $L = L(M)$

- All regular languages form a language family

Nondeterministic Finite Accepter (NFA)

Alphabet = $\{a\}$

Two choices



No transition

Formal Definition of NFAs

$$M = (Q,\ \Sigma,\ \delta,\ q_0,\ F)$$

-

$Q:$ Set of states, i.e. $\{q_0, q_1, q_2\}$

$\Sigma:$ Input alphabet, i.e. $\{a, b\}$

$\delta:$ Transition function $\delta : Q \times \boxed{(\Sigma \cup \{\lambda\}}) \to 2^Q.$

$q_0:$ Initial state

$F:$ Final states

NFA vs. DFA

- Transition functions range is Q vs. 2^Q (powersets of Q)
- λ can be an argument of transition function; transition without consuming a symbol
- $\delta(q_k, a)$ can be empty (not a total function)

δ	a	b
q_0	q_1	
q_1		q_2

Outline

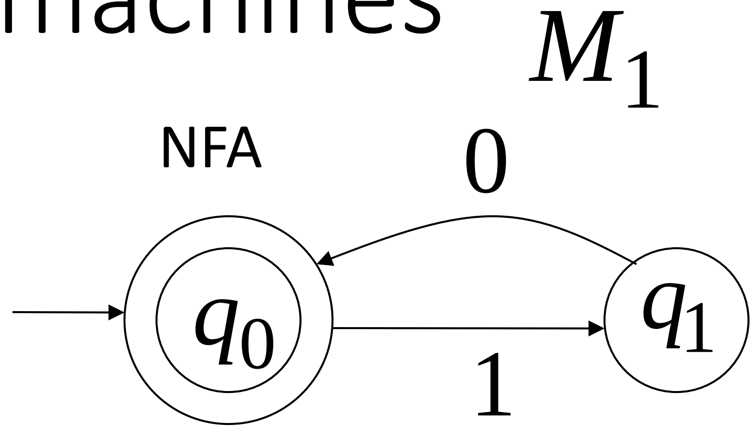
- Last time
- Equivalence of Machines
- Regular Language Properties
- Regular expressions
- Regular grammars

Equivalence of Machines

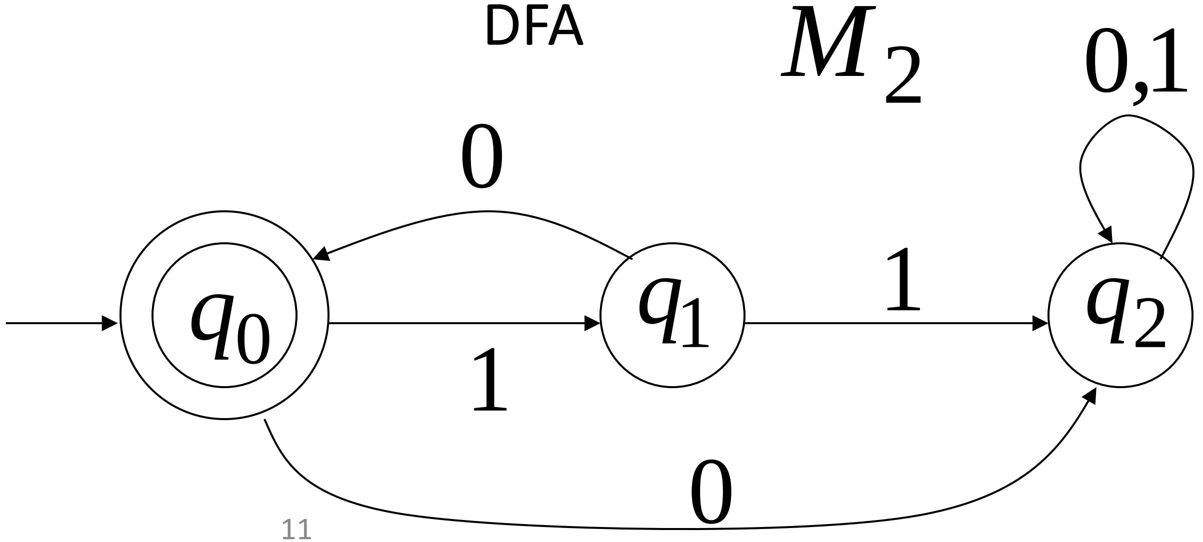
- NFAs accept the Regular Languages
- Machine M_1 is equivalent to machine M_2
- if $L(M_1) = L(M_2)$

Example of equivalent machines

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



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



We will prove:

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

(Regular Languages)

NFAs and DFAs have the same computation power

Step 1

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

Proof: Every DFA is trivially an NFA

$$\Downarrow$$

Any language L accepted by a DFA
is also accepted by an NFA

Step 2

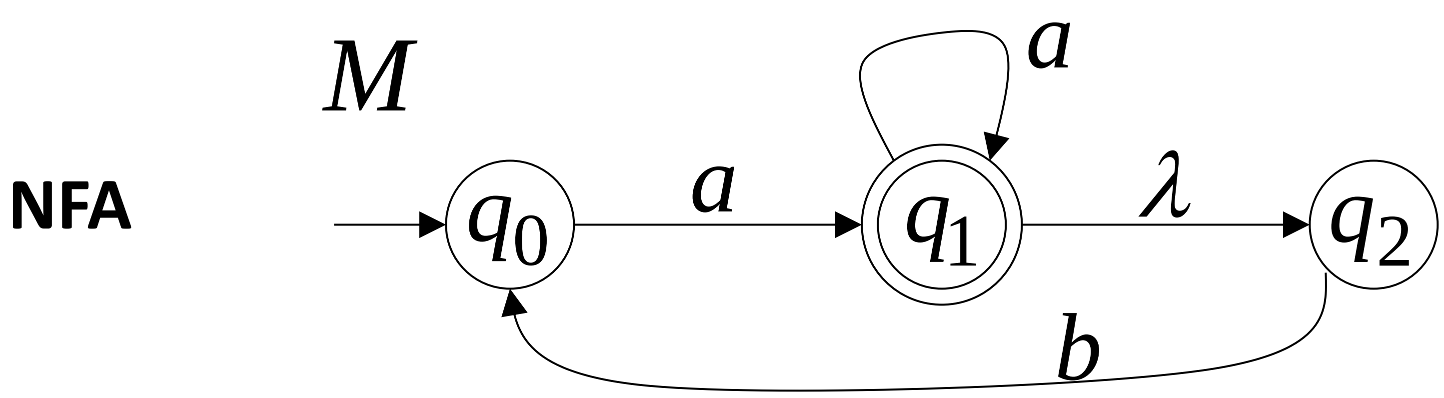
$$\left\{ \begin{array}{l} \text{Languages} \\ \text{accepted} \\ \text{by NFAs} \end{array} \right\} \subseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

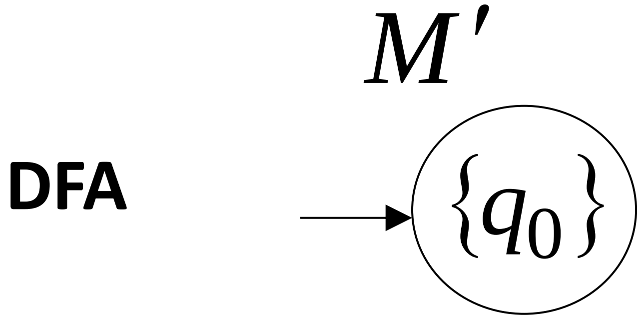
Proof: Any NFA can be converted to an equivalent DFA

↓

Any language L accepted by an NFA is also accepted by a DFA

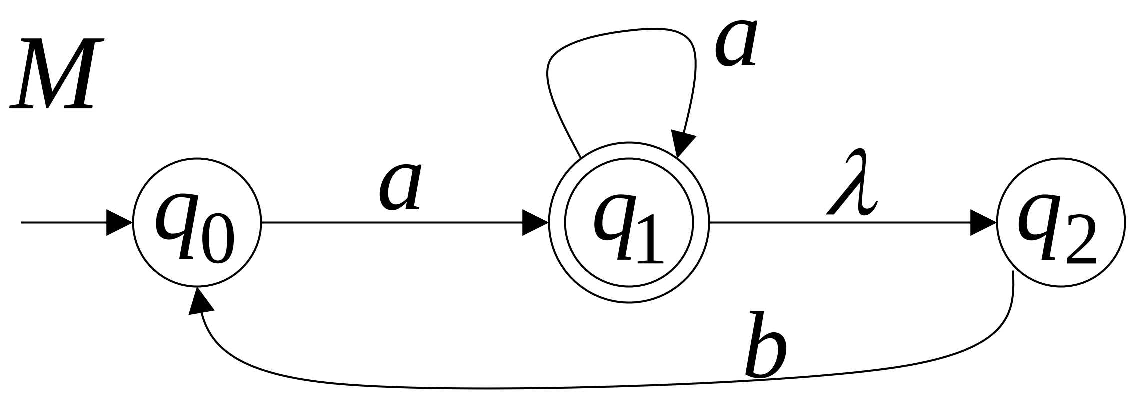
Convert NFA to DFA



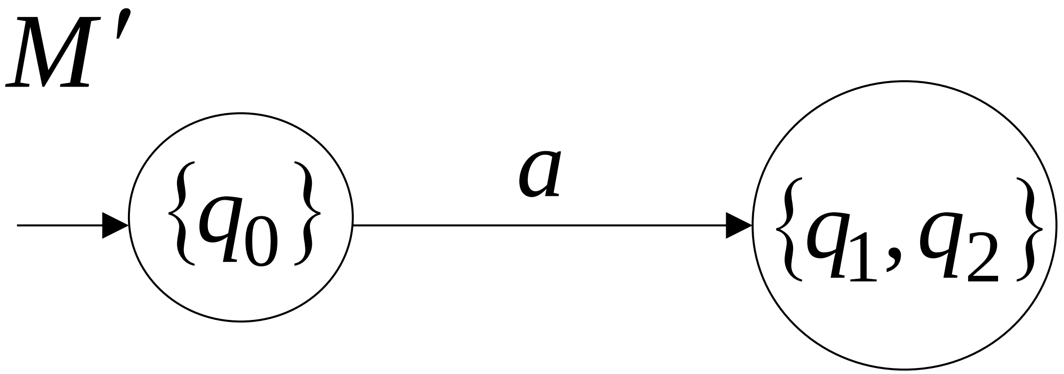


Convert NFA to DFA

NFA

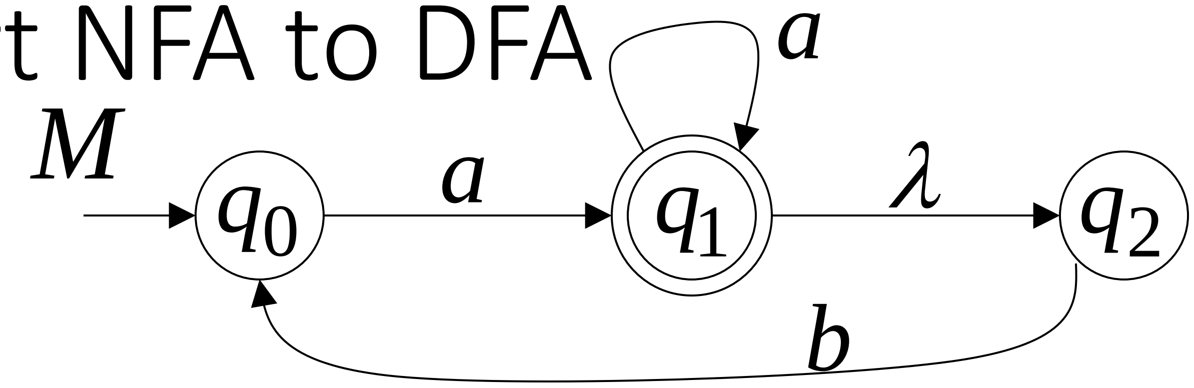


DFA

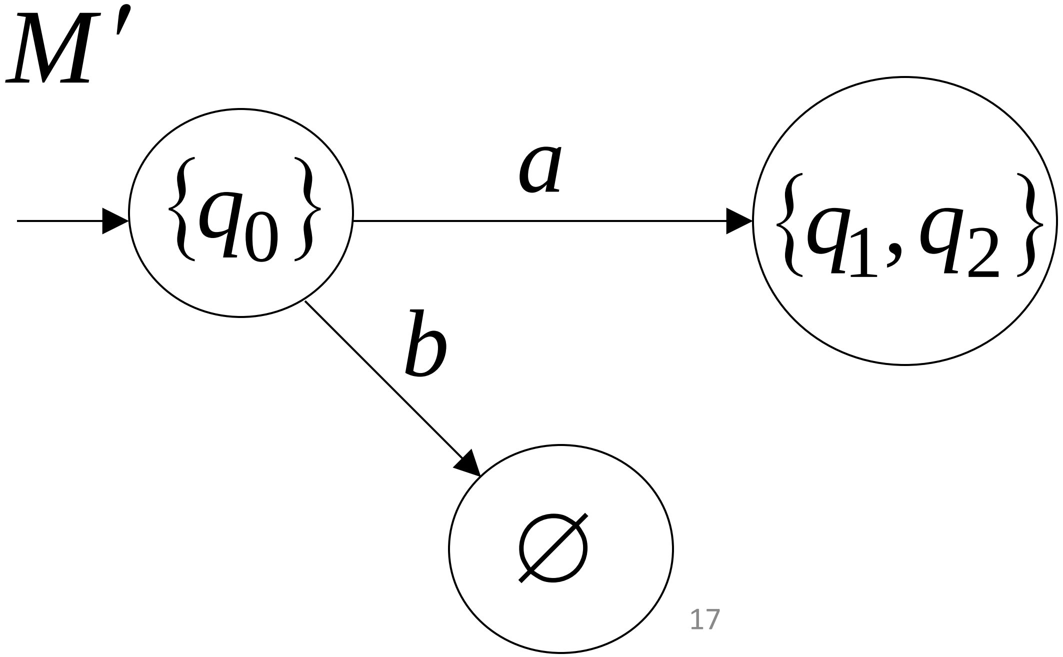


Convert NFA to DFA

NFA

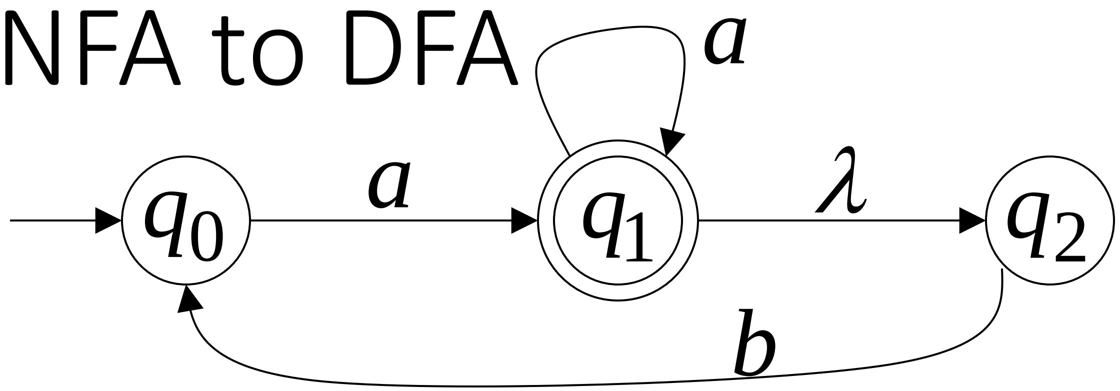


DFA

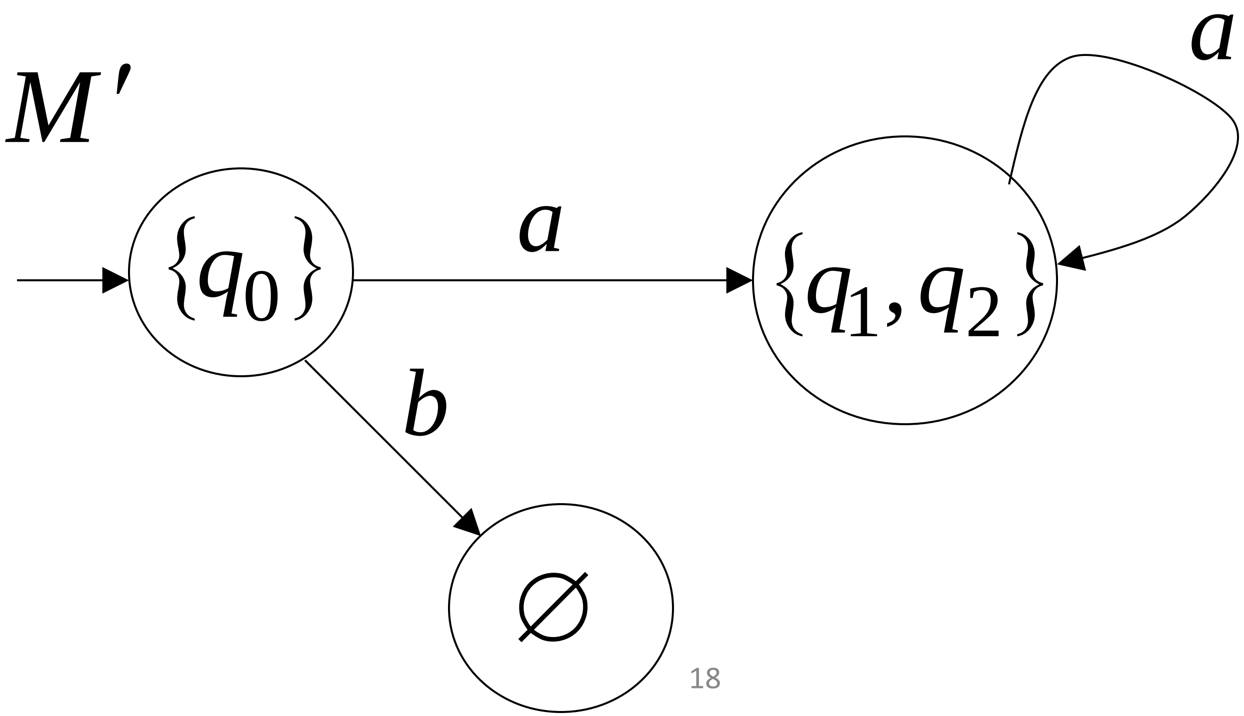


Convert NFA to DFA

- **NFA** M



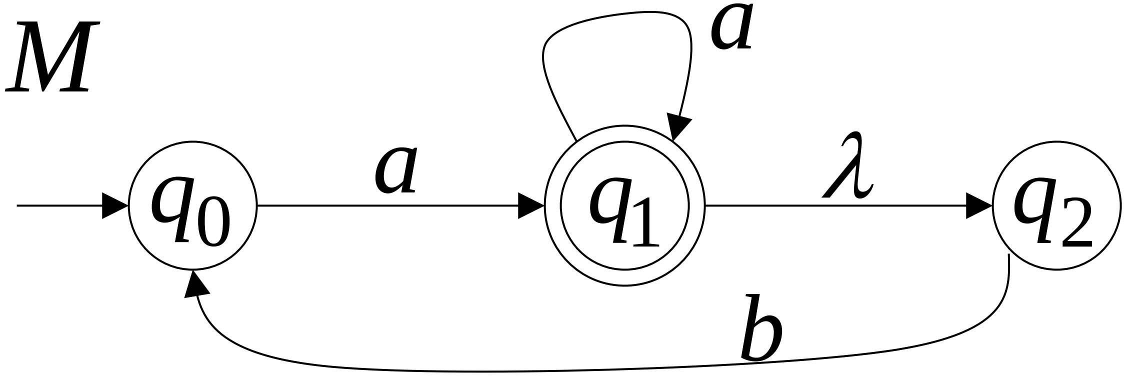
DFA M'



Convert NFA to DFA

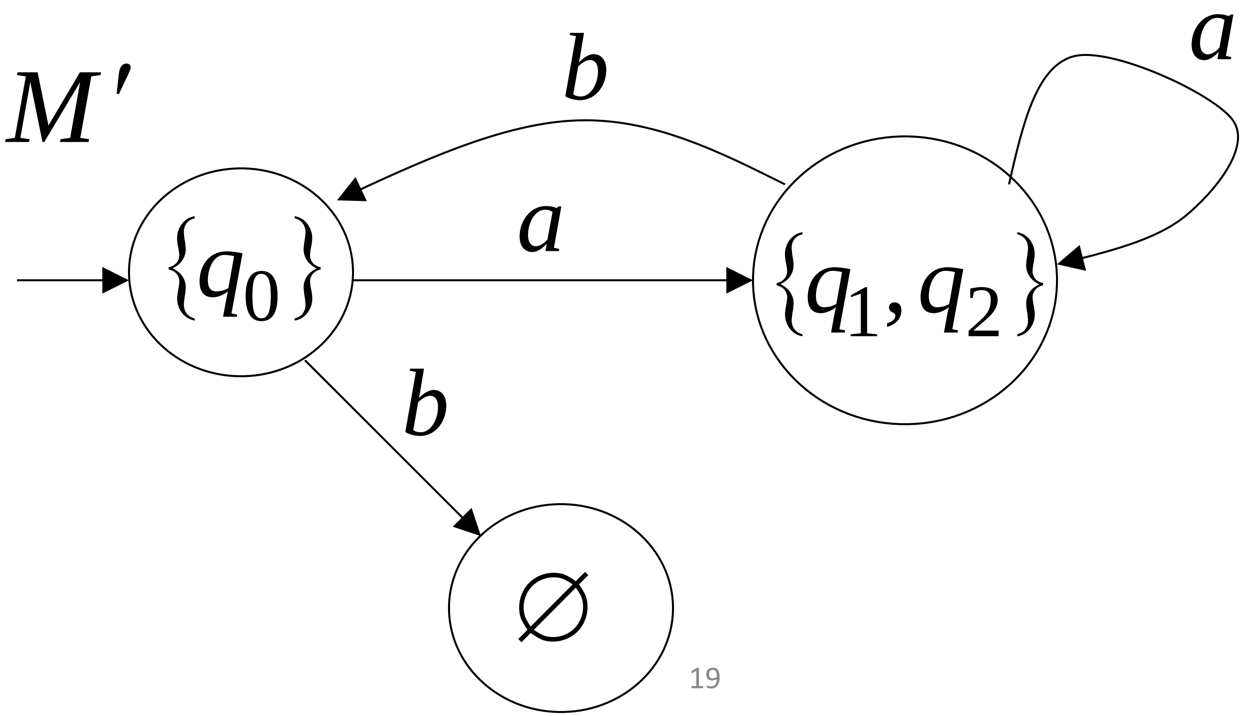
M

NFA



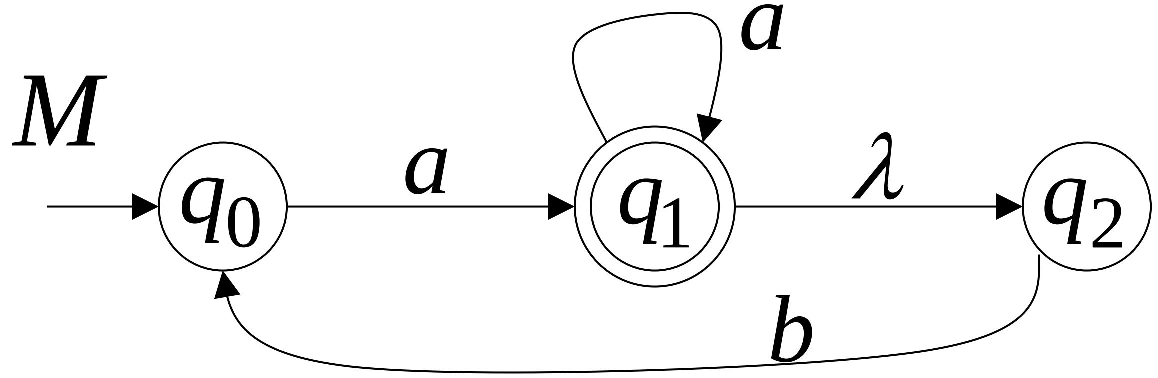
M'

DFA

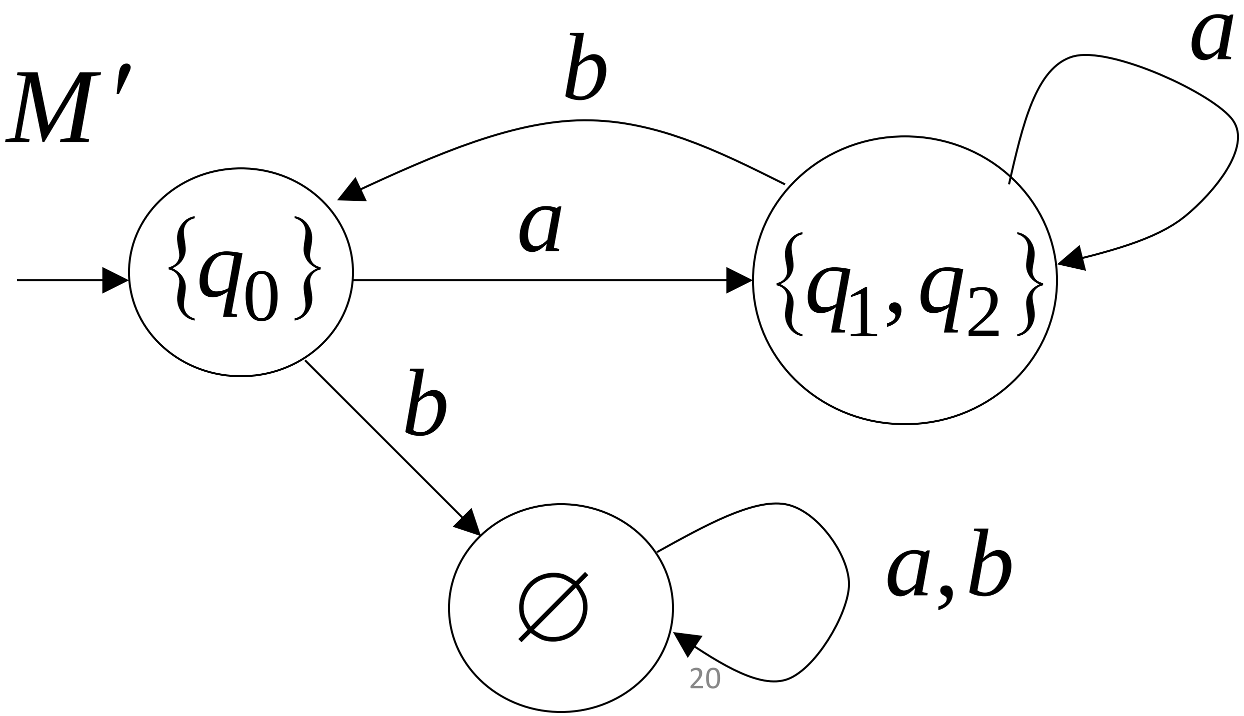


Convert NFA to DFA

NFA

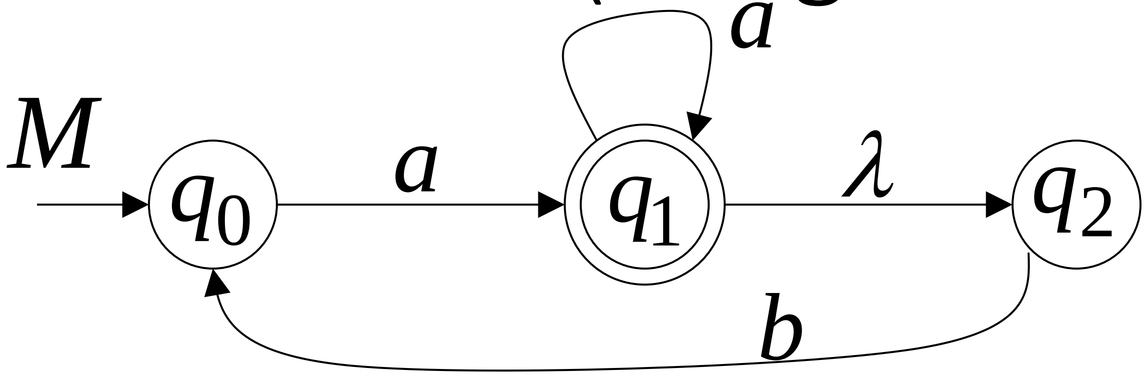


DFA



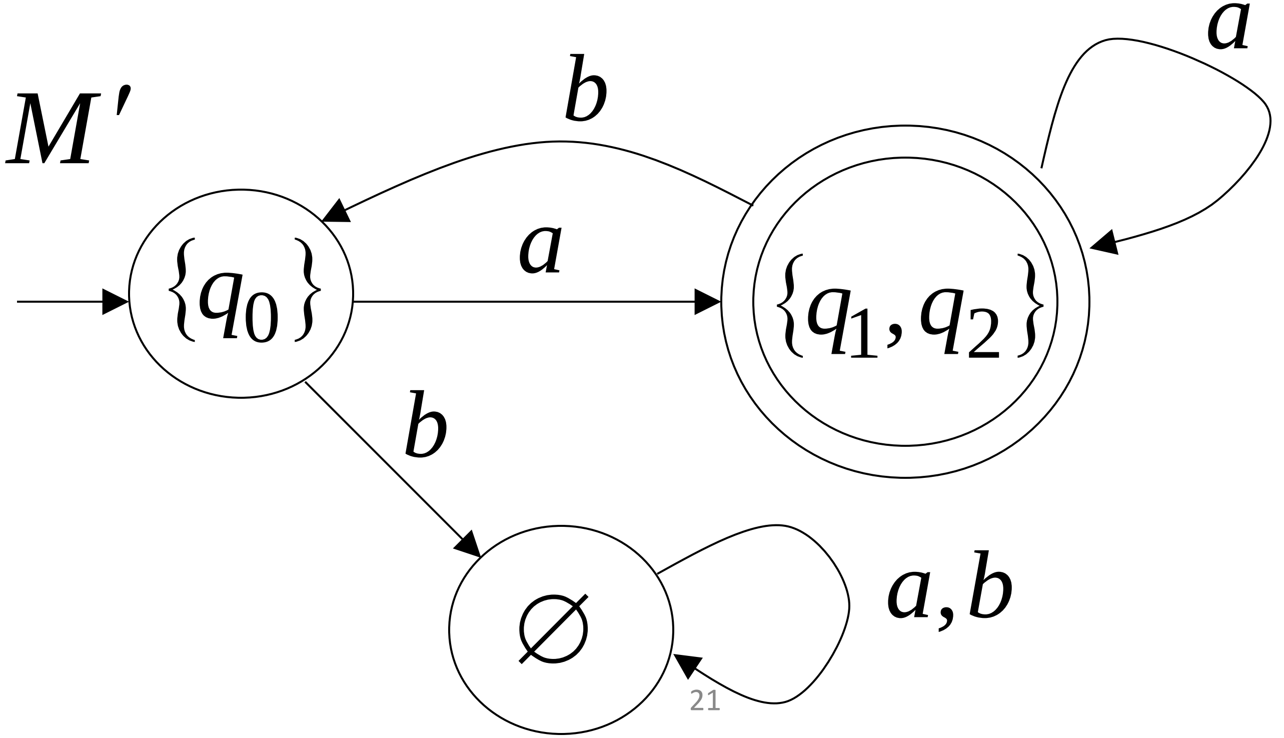
Convert NFA to DFA (Assign final state(s))

NFA

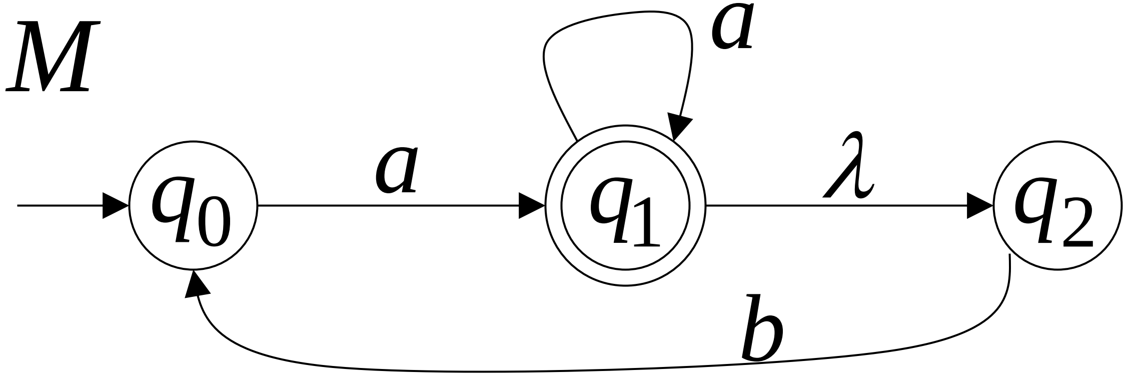


$$L(M) = L(M')$$

DFA



Convert NFA to DFA



States of DFA	a	b
{q0}	{q1, q2}	{qq}
{q1,q2}	{q1,q2}	(q0}
{qq}	{qq}	{qq}

NFA to DFA: Remarks

- We are given an NFA M

- We want to convert it M' to an equivalent DFA with $L(M) = L(M')$

- If the NFA has states $q_0, q_1, q_2, \ldots$

- the DFA has states in the powerset

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

Procedure NFA to DFA (Step 1)

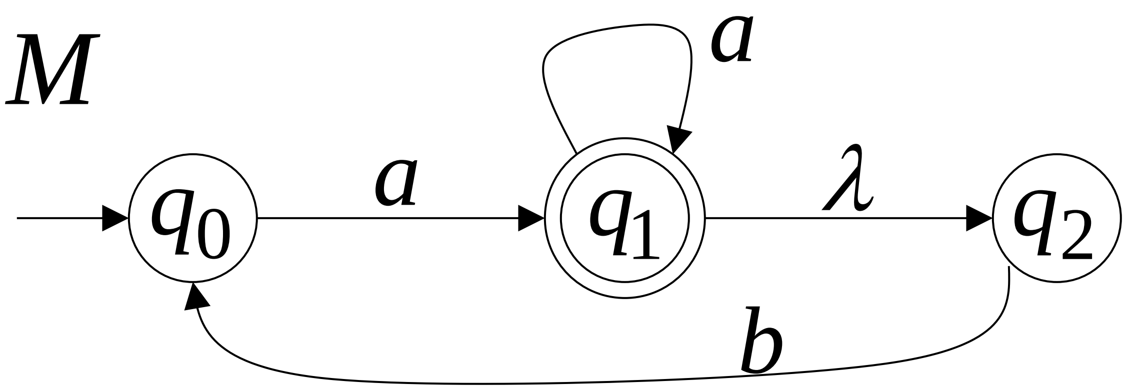
- Initial state of NFA: q_0

- Initial state of DFA: $\{q_0\}$

Example

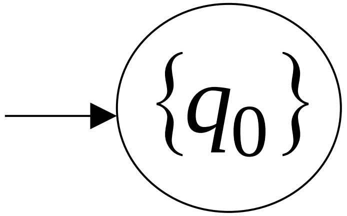
NFA

M



M'

DFA



Procedure NFA to DFA (Step 2)

- For every DFA's state $\{q_i, q_j, \ldots, q_m\}$
- Compute in the NFA

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

$$\ldots$$

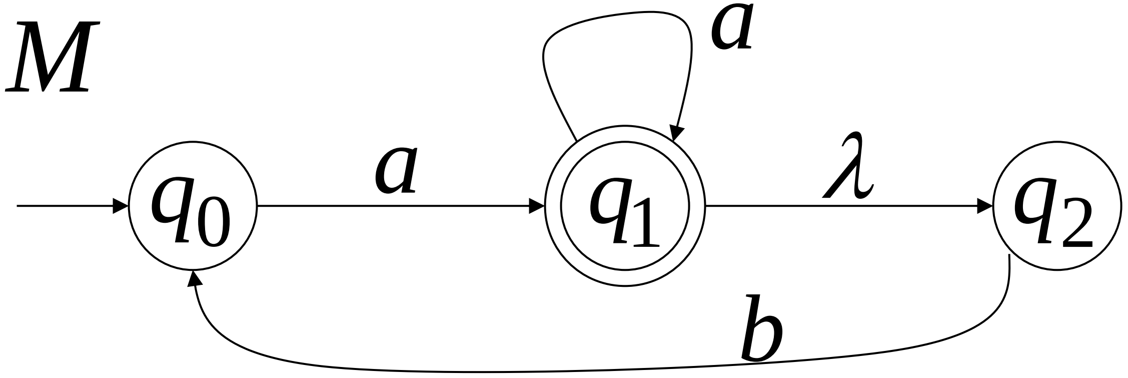
- Add transition to DFA

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

Example

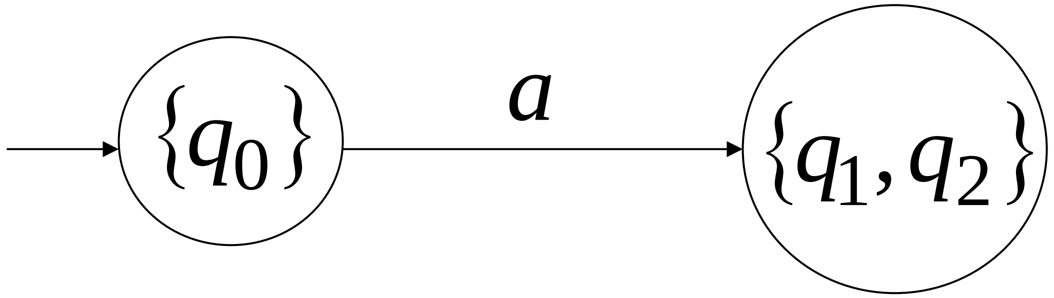
•

NFA M



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

DFA M'



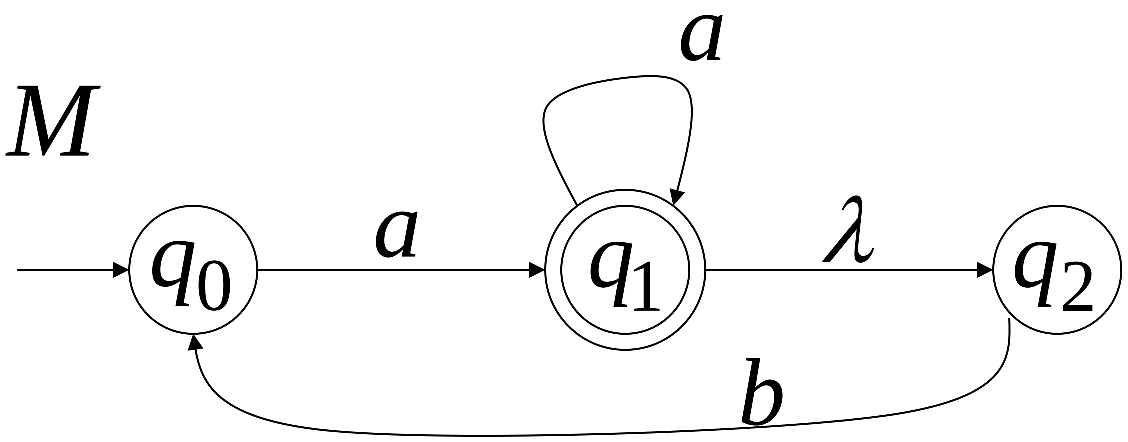
$$\delta(\{q_0\}, a) = \{q_1, q_2\}$$

Procedure NFA to DFA

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

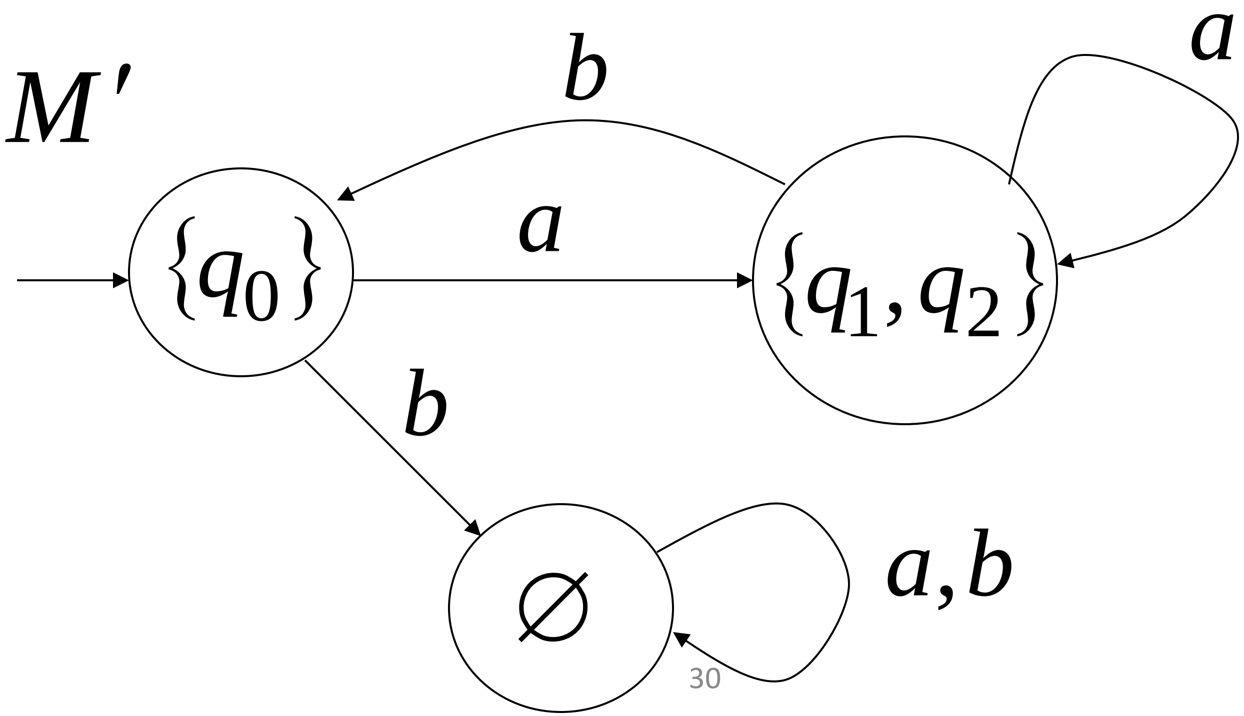
Example

NFA M



M'

DFA



Procedure NFA to DFA (Step 3)

- For any DFA state $\{q_i, q_j, \ldots, q_m\}$

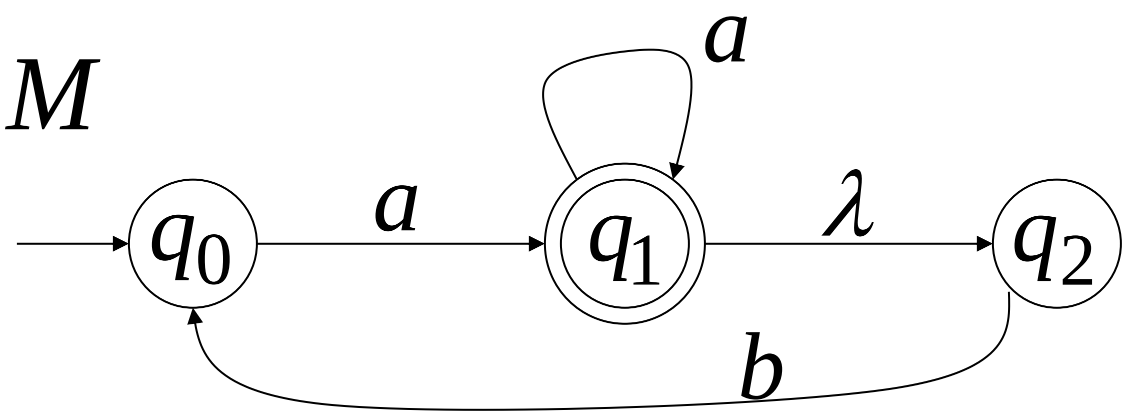
- If some q_j is a final state in the NFA

- Then, $\{q_i, q_j, \ldots, q_m\}$ is a final state in the DFA

Example

M

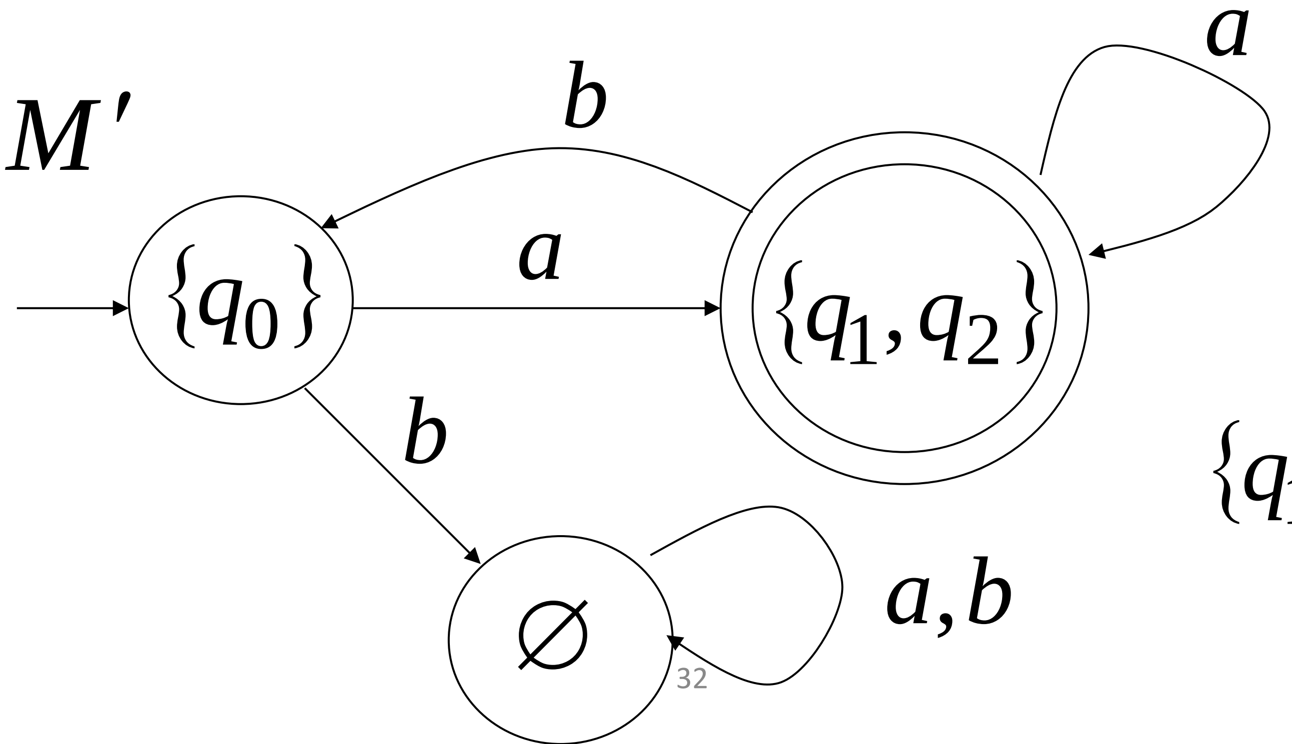
- **NFA**



$q_1 \in F$

DFA

M'

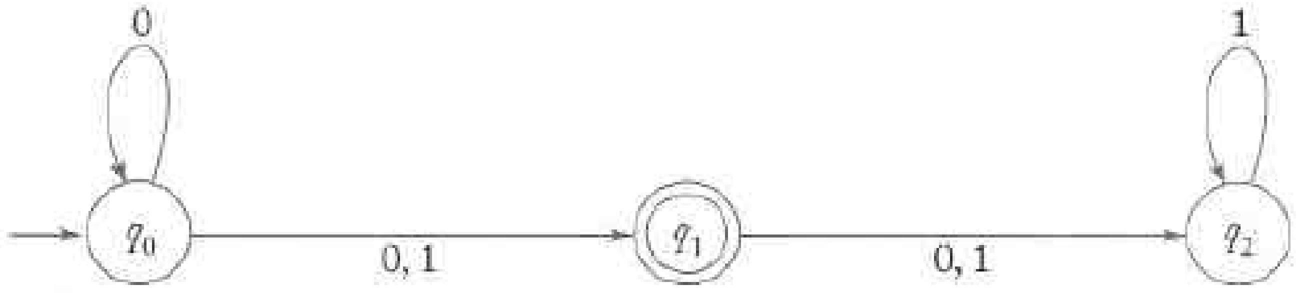


$\{q_1, q_2\} \in F'$

Theorem

- Take NFA M and apply procedure to obtain DFA M' then M and M' are equivalent.
- Also $L(M) = L(M')$

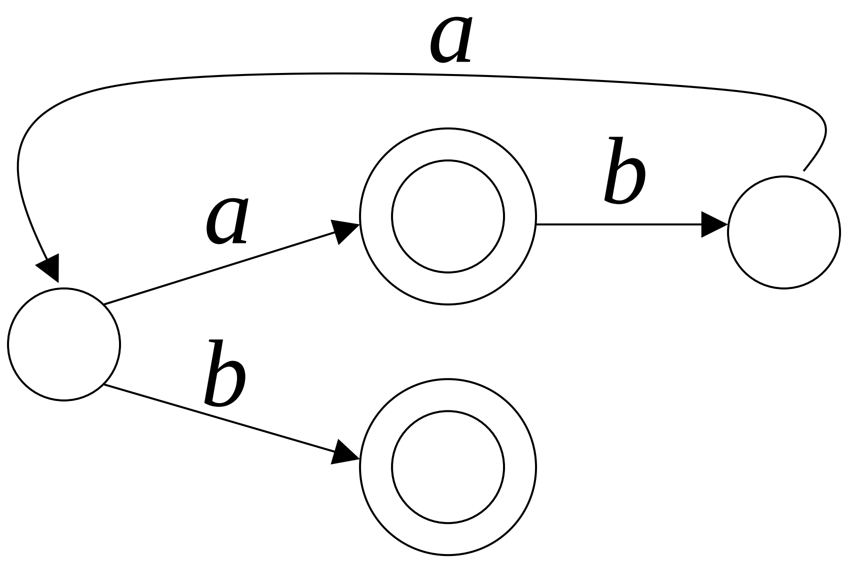
Example: Convert NFA to DFA



State	0	1
{q0}=q0	**{q0,q1}**	**{q1}**
{q0,q1}=q1	**{q0,q1,q2}**	**{q1,q2}**
{q1}=q2	**{q2}**	**{q2}**
{q0,q1,q2}=q3	**{q0,q1,q2}**	**{q1,q2}**
{q1,q2}=q4	**{q2}**	**{q2}**
{q2}=q5	**qq**	**{q2}**
qq=q6	**qq**	**qq**

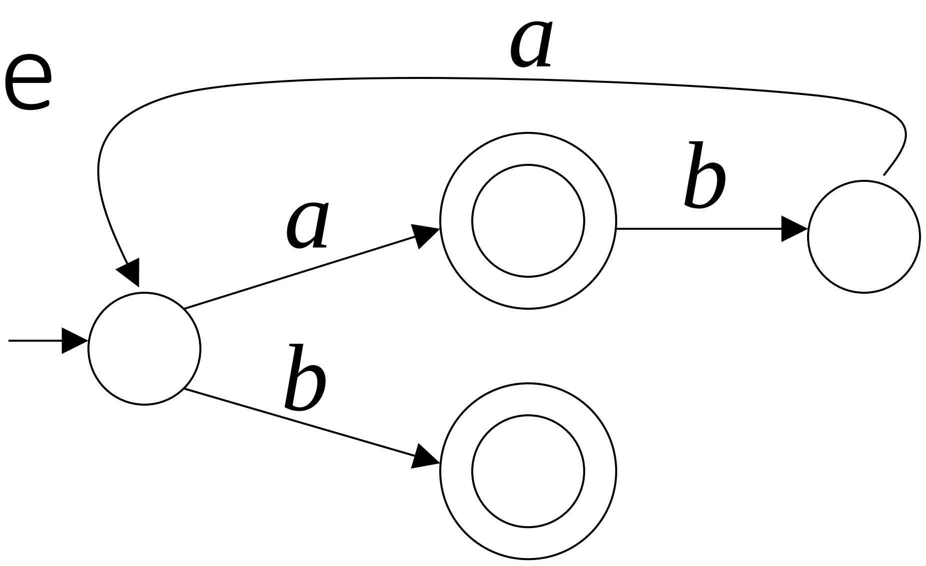
More Conversions:
Single Final State NFAs

- Any NFA can be converted to an equivalent NFA with a single final state



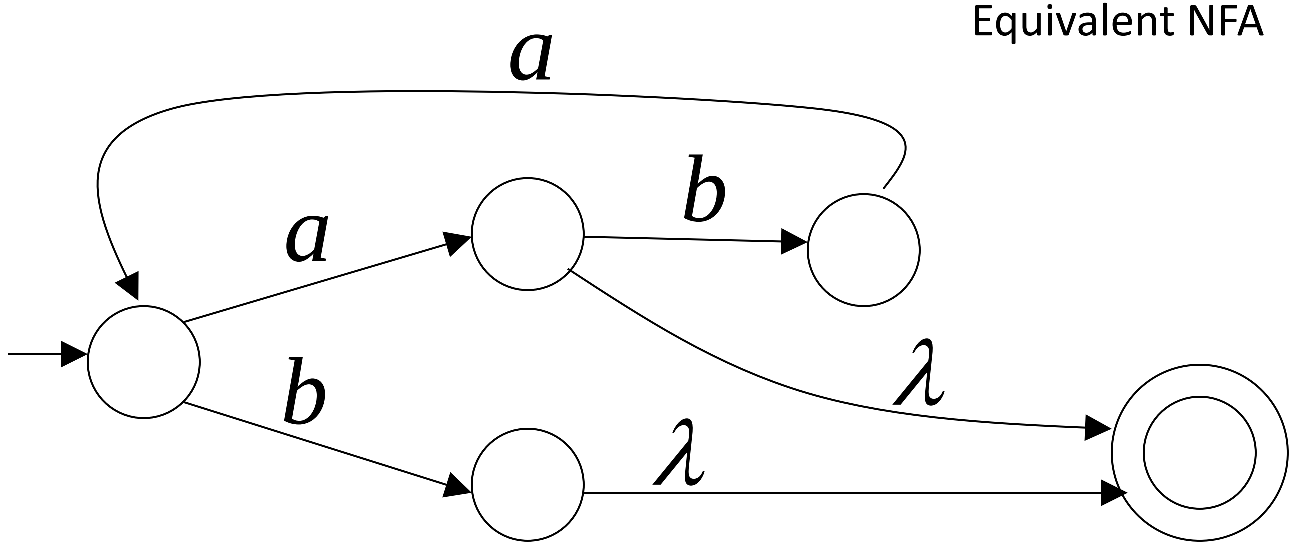
NFA

Example



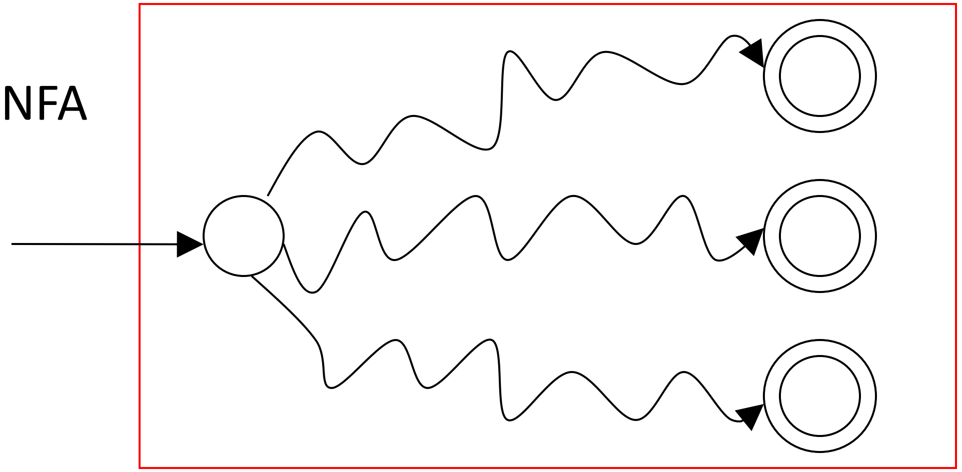
NFA

Equivalent NFA

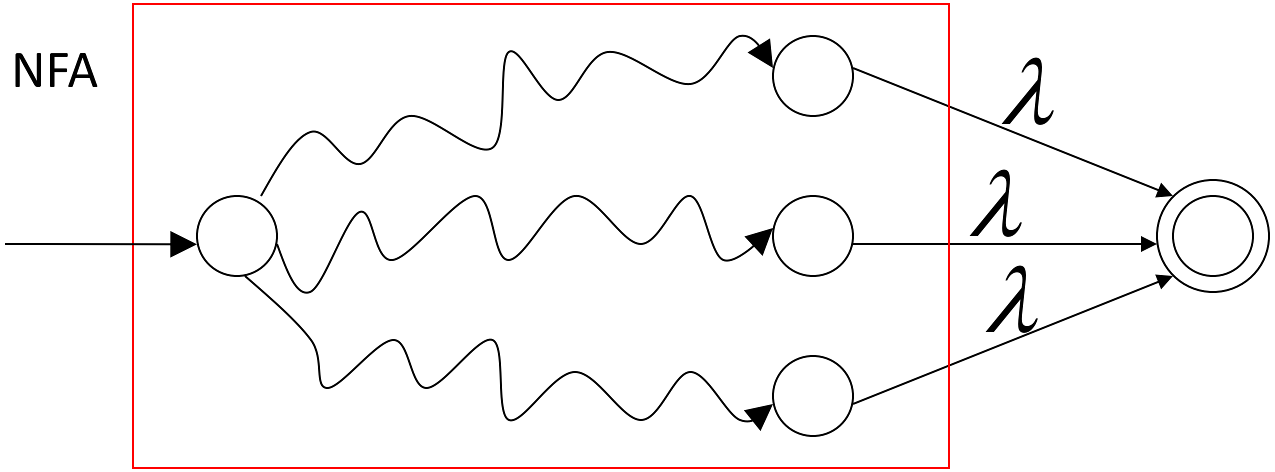


In general

NFA



Equivalent NFA



Single
final state

Conversions Covered

- NFA -> DFA
- NFA with multiple final states -> NFA with single final state
- DFA with multiple final states -> NFA with single final state

Outline

- Last time
- Equivalence of Machines
- Regular Language Properties
- Regular expressions

Regular Languages

- A language L is regular if there is a DFA M such that $L = L(M)$

- All regular languages form a language family

Regular Language Examples

$\{abba\}$ $\{\lambda, ab, abba\}$ $\{a^n b : n \geq 0\}$

{ all strings with prefix ab }

{ all strings without substring 001}

Regular Language Properties

For regular languages L_1 and L_2

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: $L_1{}^*$

Reversal: $L_1{}^R$

Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

Are regular Languages

Or we say regular languages are **closed under**

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: $L_1{}^*$

Reversal: $L_1{}^R$

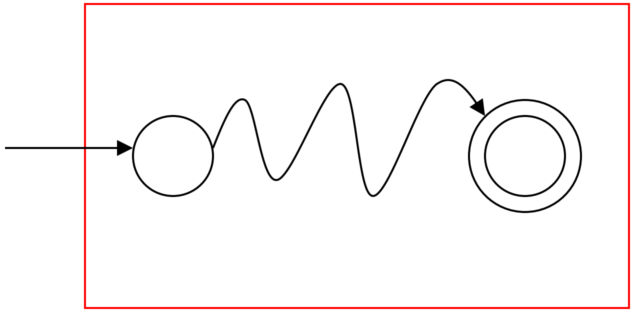
Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

Regular language L_1

$$L(M_1) = L_1$$

NFA M_1

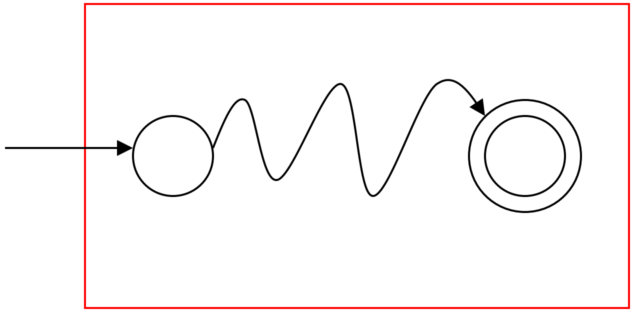


Single final state

Regular language L_2

$$L(M_2) = L_2$$

NFA M_2



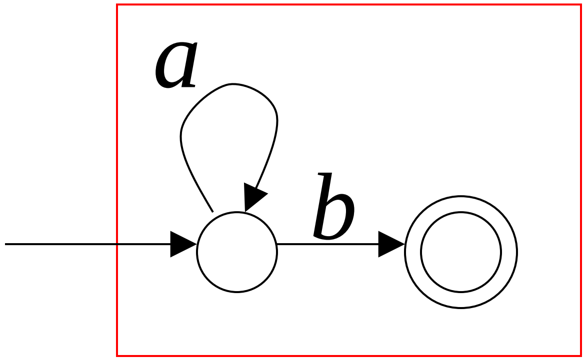
Single final state

Example

$n \geq 0$

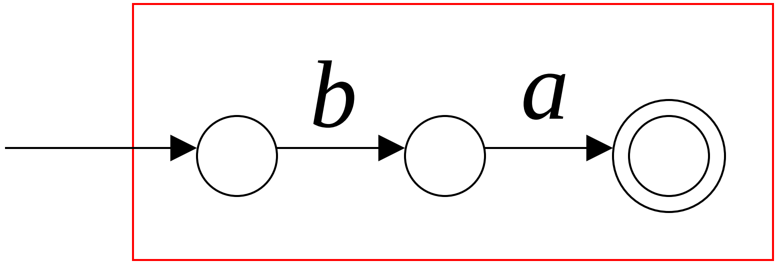
$$L_1 = \{a^n b\}$$

M_1



$$L_2 = \{ba\}$$

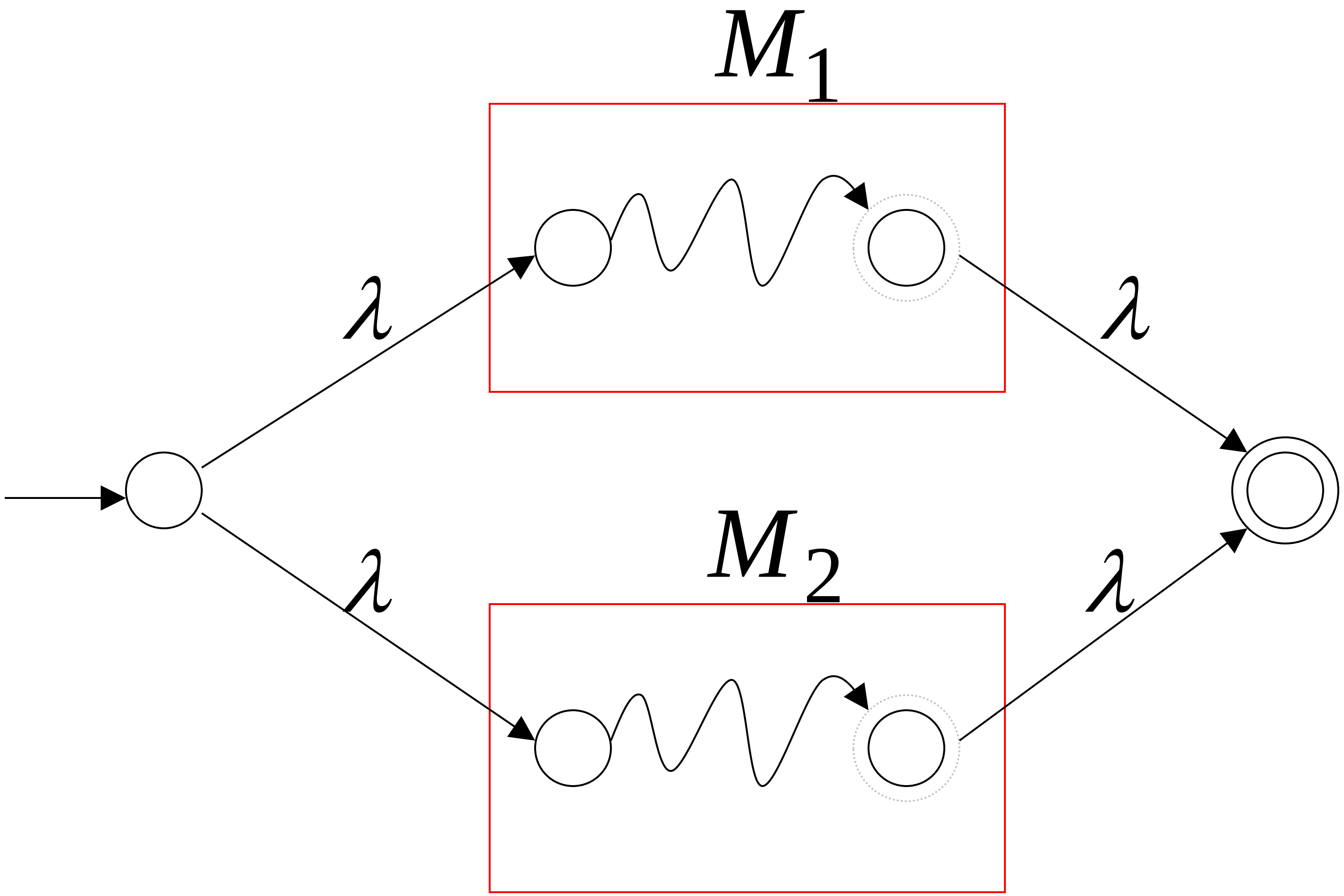
M_2



Union

$L_1 \cup L_2$

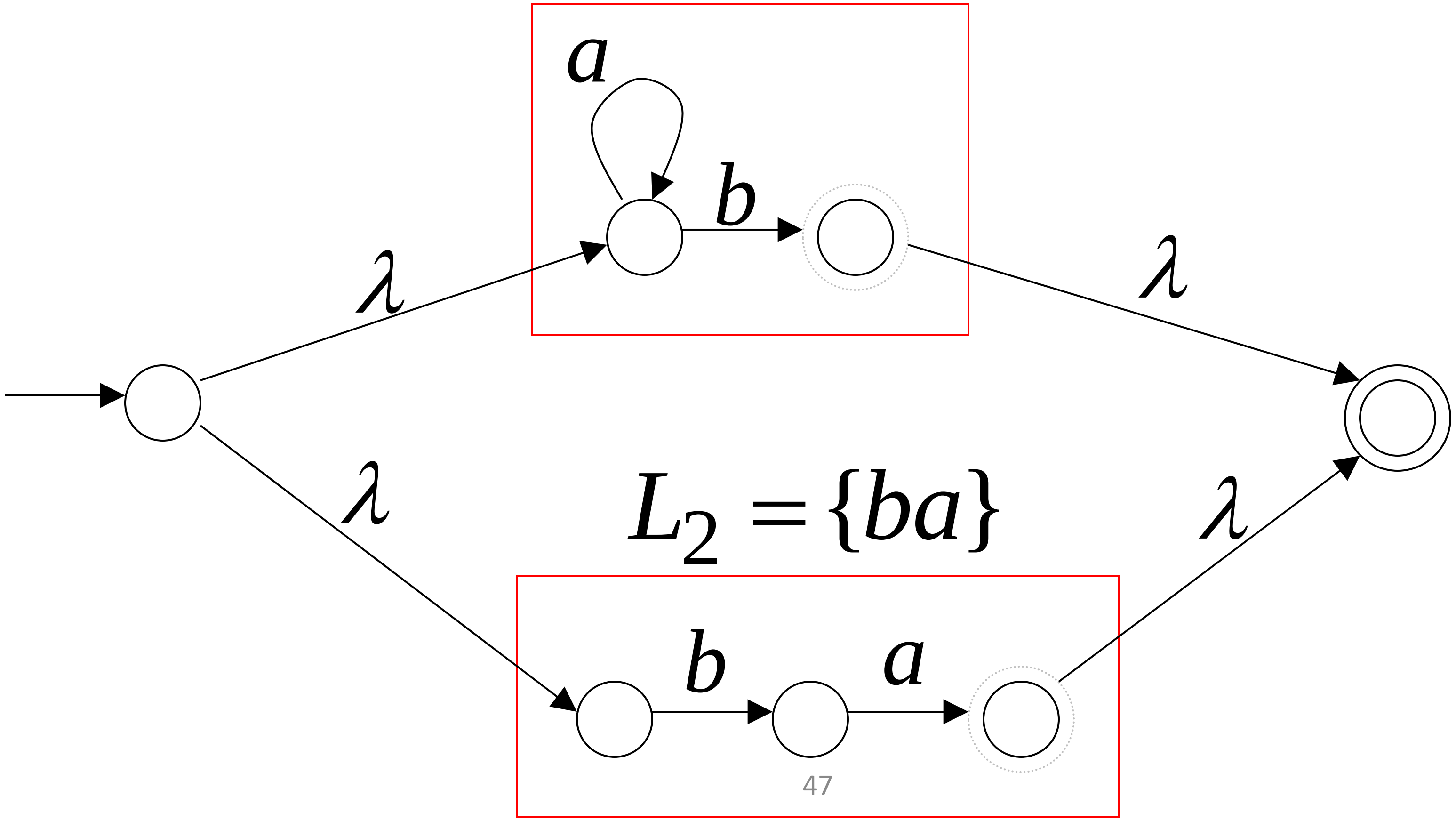
- NFA for



Example

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

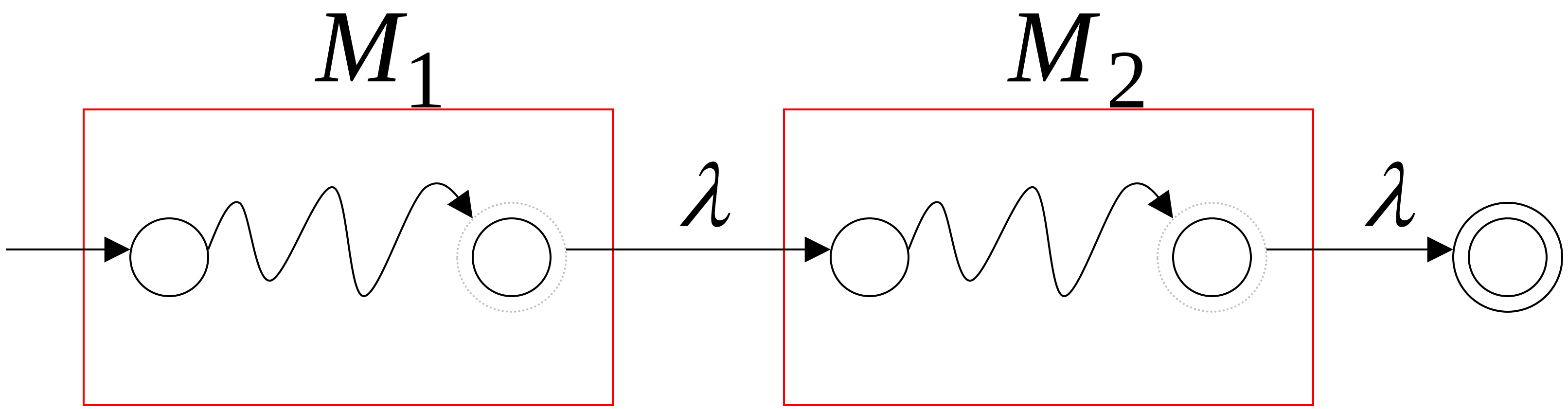
$$L_1 = \{a^n b\}$$



Concatenation

$$L_1 L_2$$

- NFA for

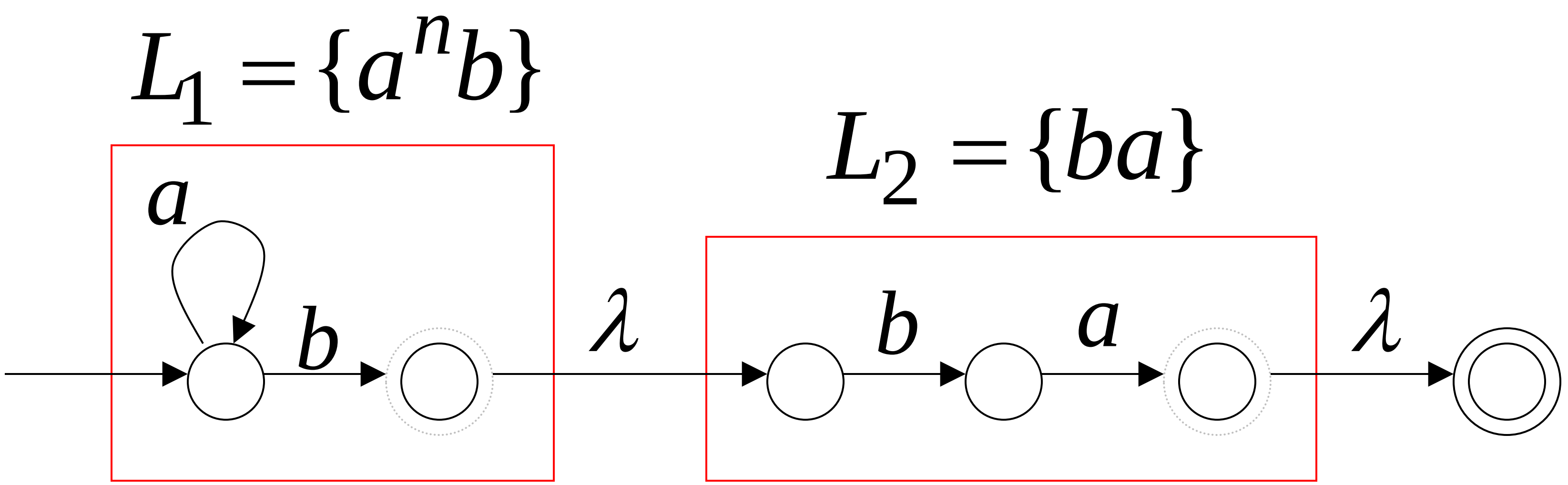


Example

$$L_1L_2 = \{a^n b\}\{ba\} = \{a^n bba\}$$

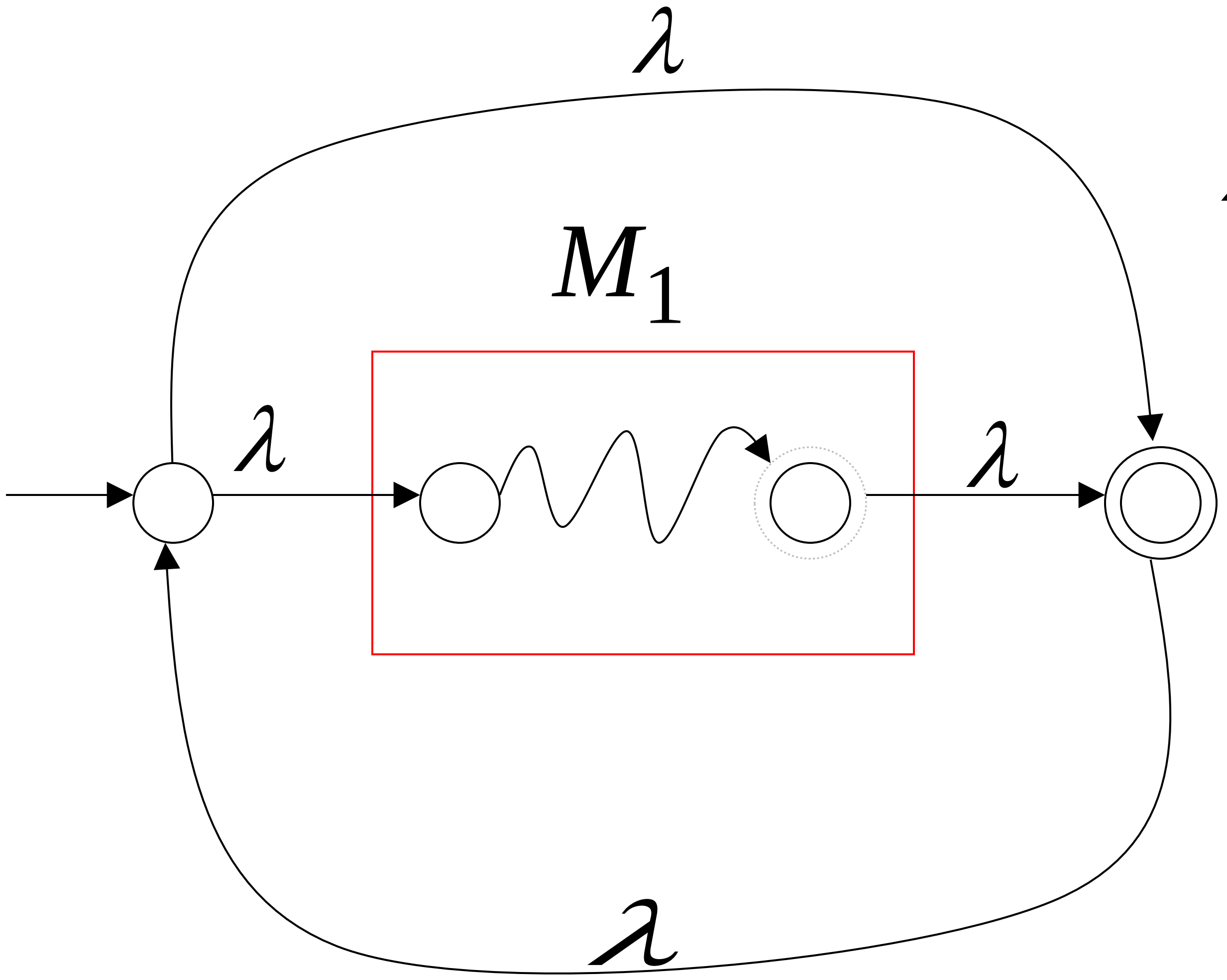
•

• NFA for



Star Operation $L_1{}^*$

- NFA for



$\lambda \in L_1{}^*$

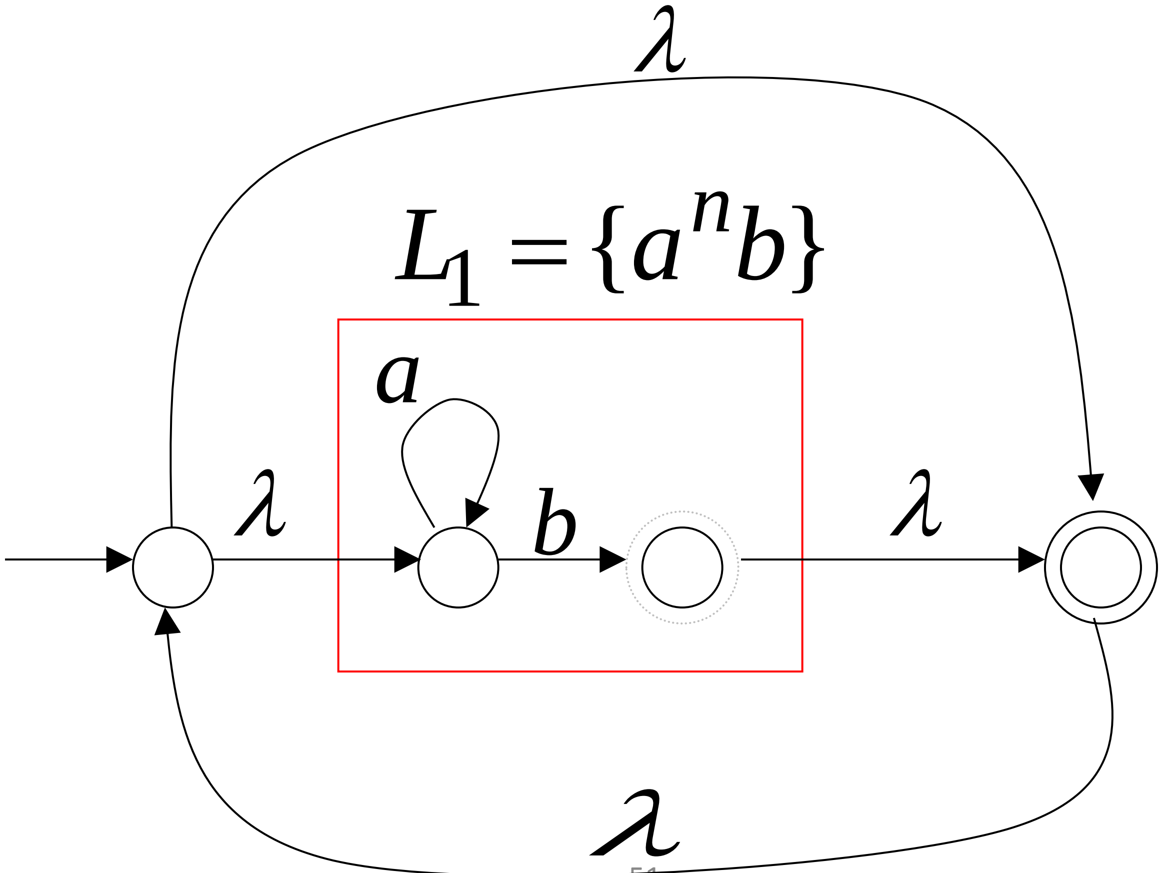
Example

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

$$w = w_1 w_2 \cdots w_k$$

$$w_i \in L_1$$

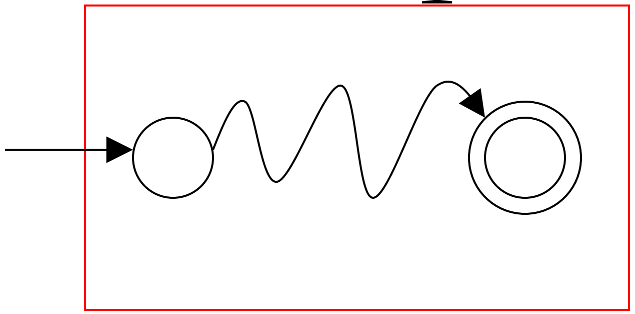
- NFA for



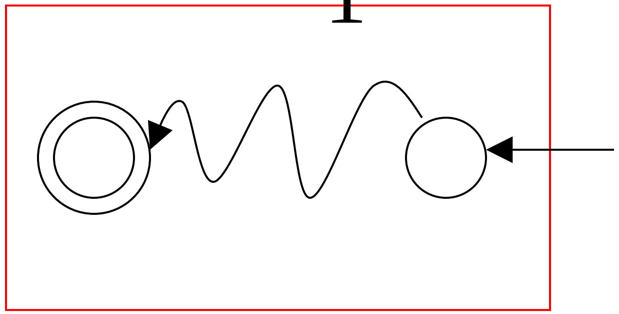
Reverse

NFA for L_1^R

L_1 M_1



M_1'



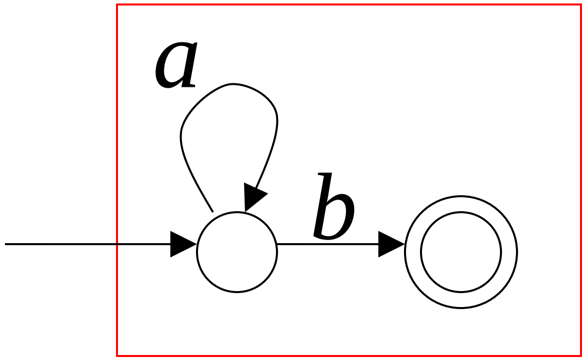
1. Reverse all transitions

2. Make initial state final state
   and vice versa

Example

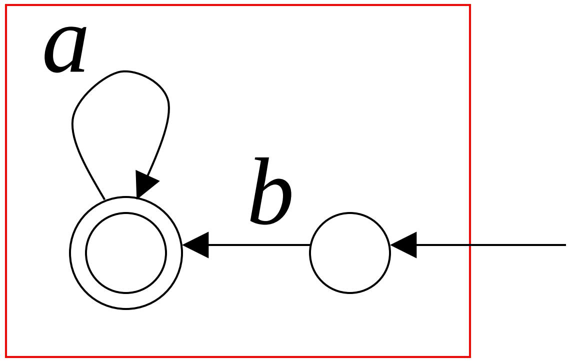
$$L_1 = \{a^n b\}$$

M_1

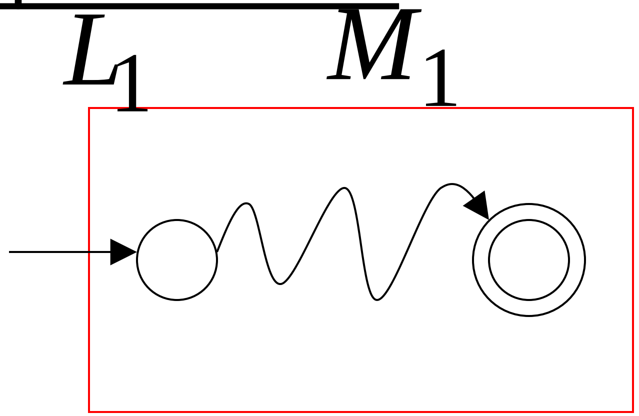


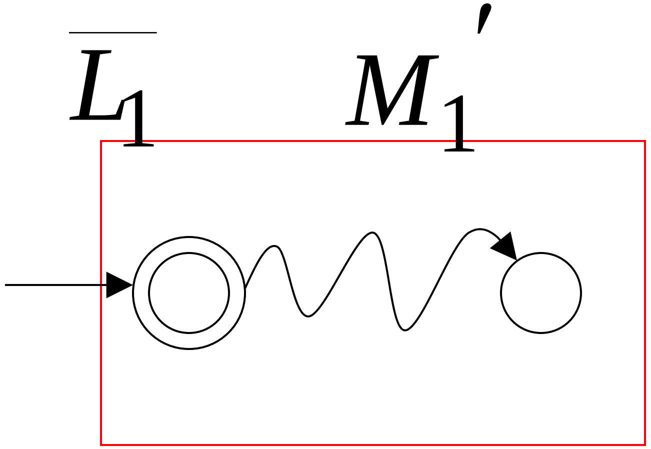
$$L_1^R = \{ba^n\}$$

M_1'



Complement





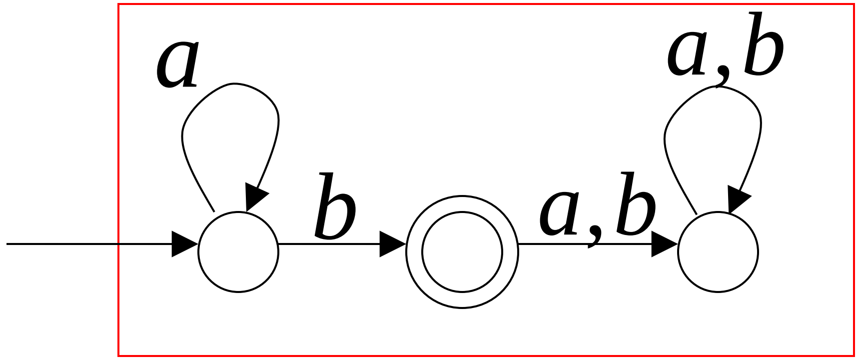
1. Take the **DFA** that accepts L_1

2. Make final states non-final,
   and vice-versa

Example

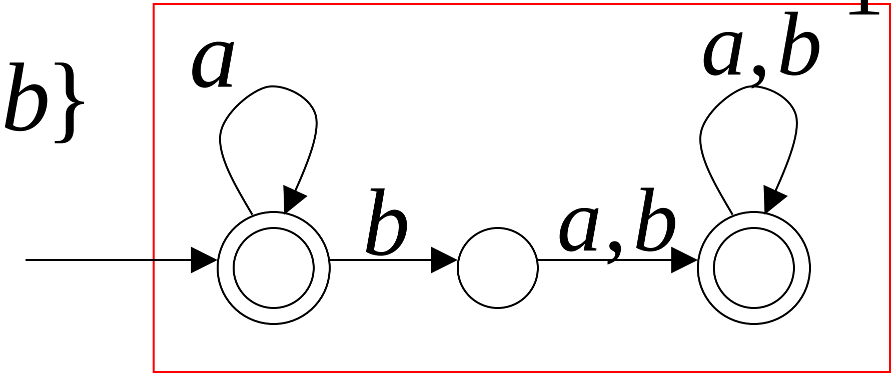
M_1

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



M_1'

$\overline{L_1} = \{a,b\}^* - \{a^n b\}$



Intersection

DeMorgan's Law: $$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

$L_1\,,\ L_2$ regular

$\Rightarrow$ $\overline{L_1}\,,\ \overline{L_2}$ regular

$\Rightarrow$ $\overline{L_1} \cup \overline{L_2}$ regular

$\Rightarrow$ $\overline{\overline{L_1} \cup \overline{L_2}}$ regular

$\Rightarrow$ $L_1 \cap L_2$ regular

Example

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

$$L_2 = \{ab, ba\}$$ regular

$$\Rightarrow L_1 \cap L_2 = \{ab\}$$

regular

Outline

- Last time
- Equivalence of Machines
- Regular Language Properties
- Regular expressions

Describing Regular Languages

- DFA or NFA (covered)
- Regular expressions
- Regular grammars

Regular Expressions

- Regular expressions describe regular languages

$$(a+b\cdot c)^*$$

- Example describes the language:

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

Recursive Definition

Primitive regular expressions: $\varnothing, \quad \lambda, \quad \alpha$

Given regular expressions r_1 and r_2

$$\left.\begin{array}{ll} \text{Union (or)} & r_1 + r_2 \\ \text{Concatenation} & r_1 \cdot r_2 \\ \text{Star closure} & r_1{}^* \\ & (r_1) \end{array}\right\} \text{Are regular expressions}$$

Examples

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

Not a regular expression: $\left(a+b+\right)$

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:

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

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

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

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: $(a+b)\cdot a^*$

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