Big picture

- All languages
- Decidable

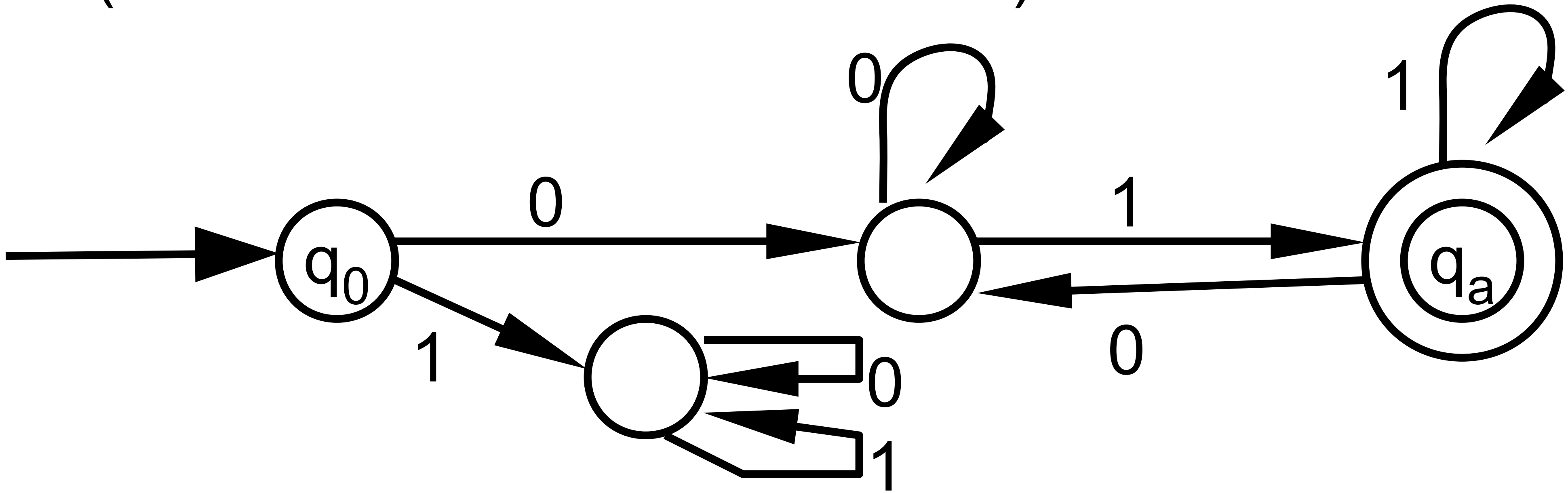
  Turing machines
- NP
- P
- Context-free

  Context-free grammars, push-down automata
- Regular

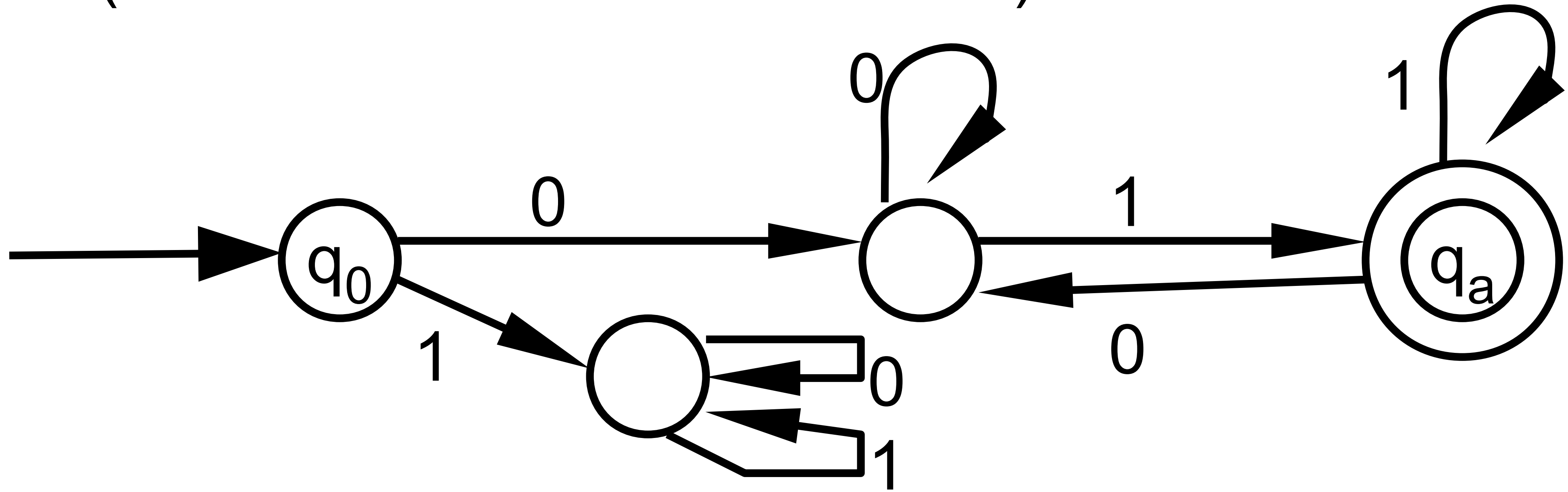
  Automata, non-deterministic automata,

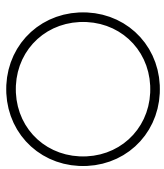
  regular expressions

DFA (Deterministic Finite Automata)



DFA (Deterministic Finite Automata)

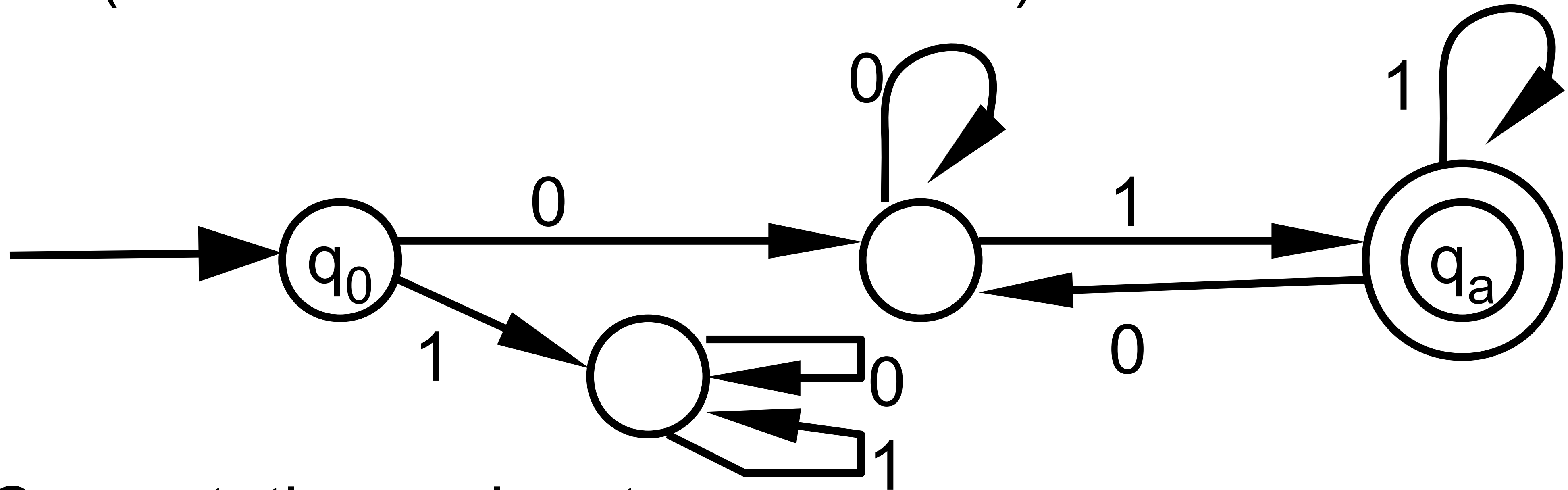


- States  , this DFA has 4 states

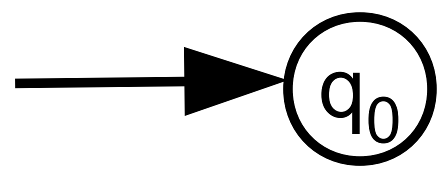
- Transitions 

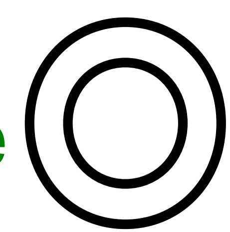
  labelled with elements of the alphabet $\Sigma = \{0,1\}$

DFA (Deterministic Finite Automata)



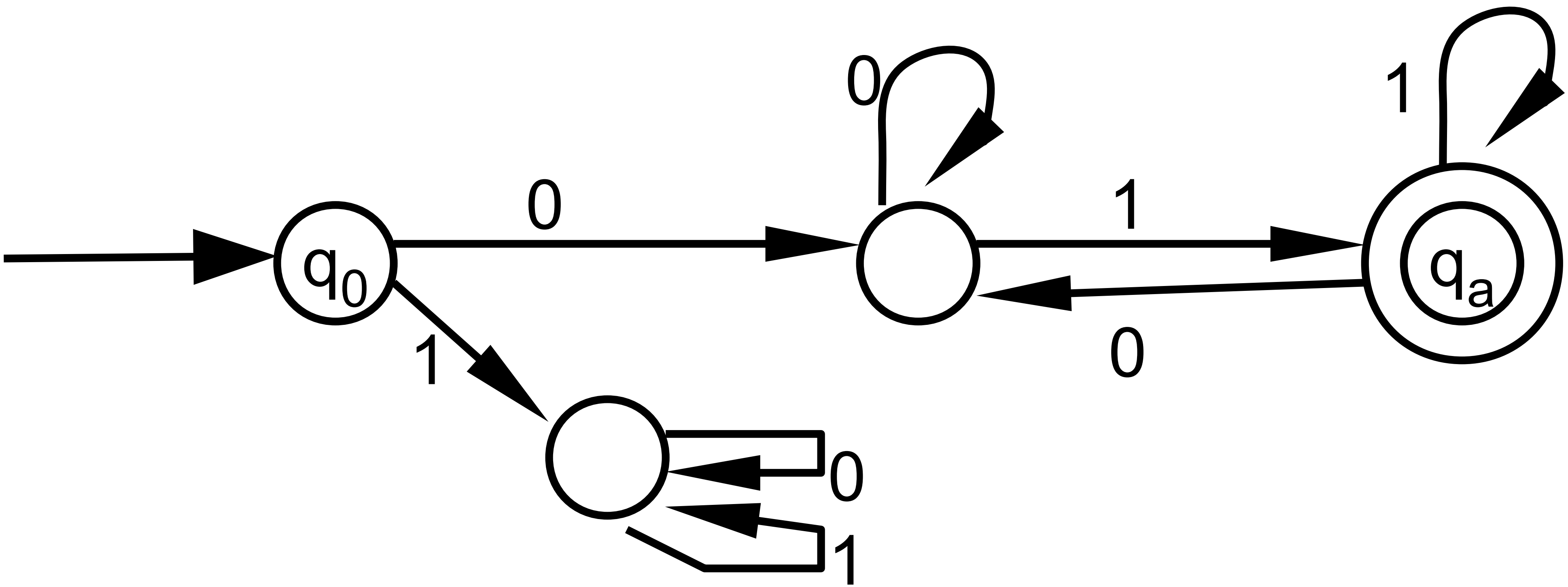
Computation on input w:

- Begin in start state 

- Read input string in a one-way fashion
- Follow the arrows matching input symbols
- When input ends: ACCEPT if in accept state 

  REJECT if not

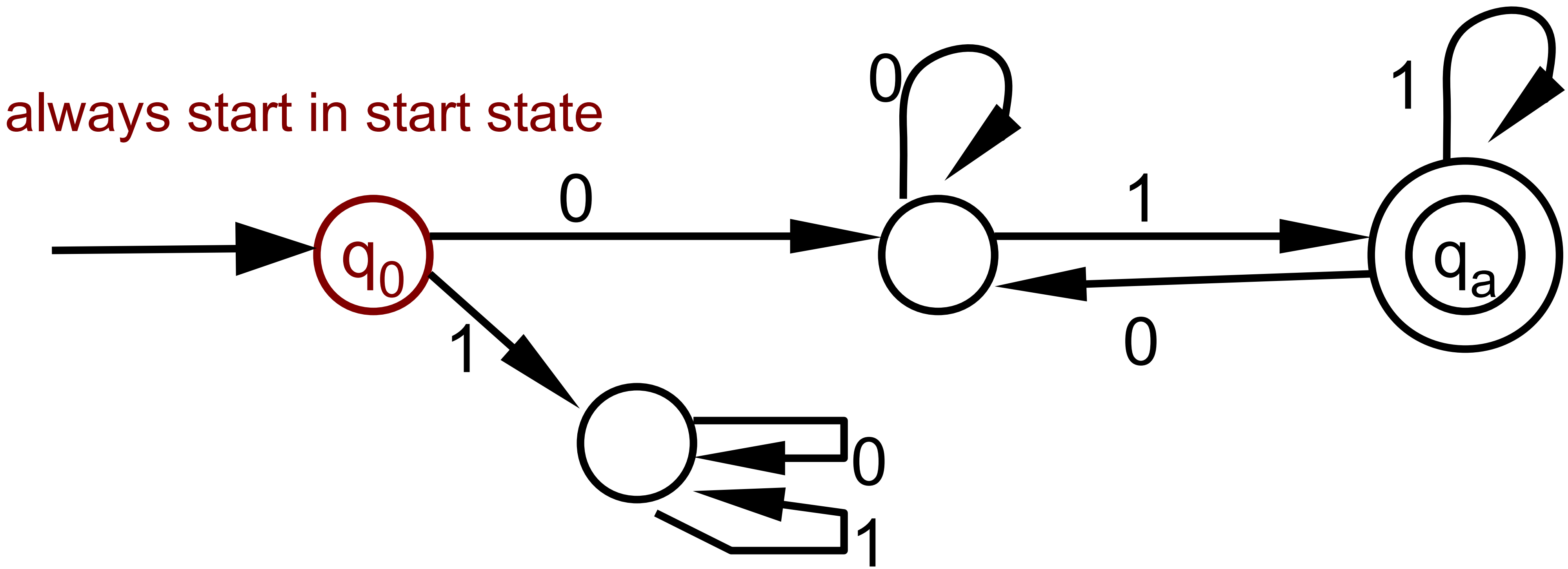
DFA (Deterministic Finite Automata)



Example: Input string

w = 0011

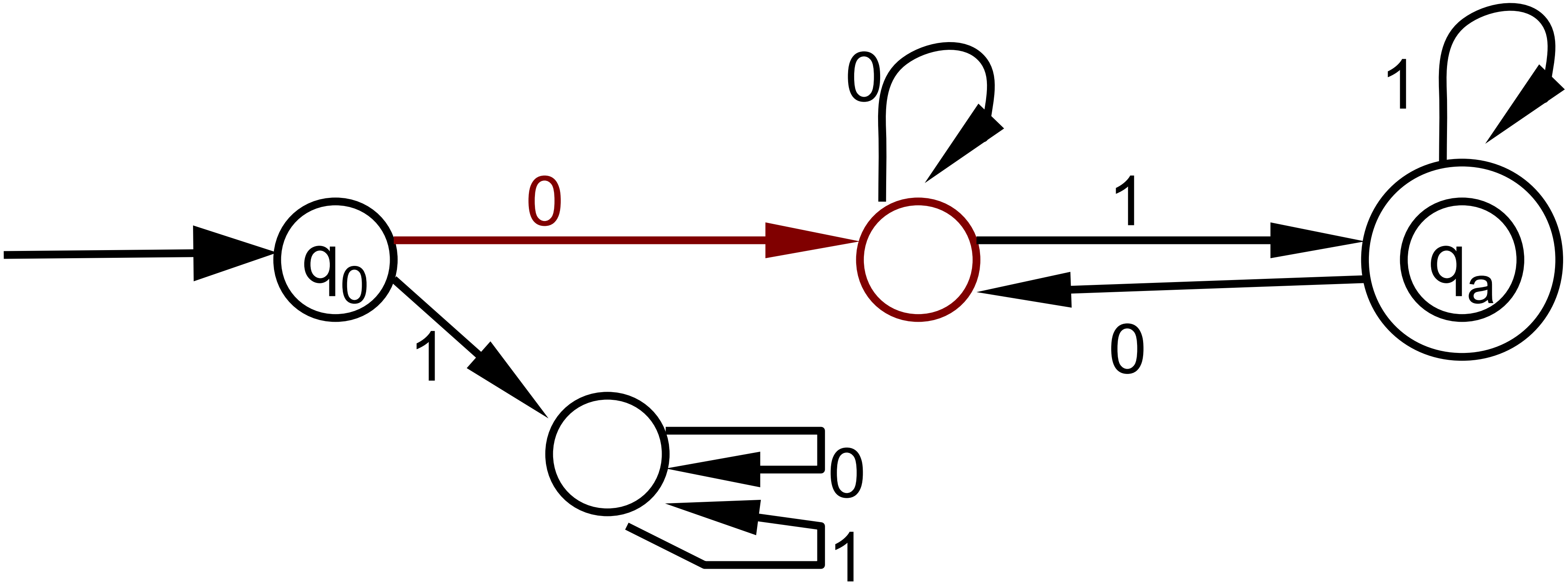
DFA (Deterministic Finite Automata)



Example: Input string

$w = 0011$

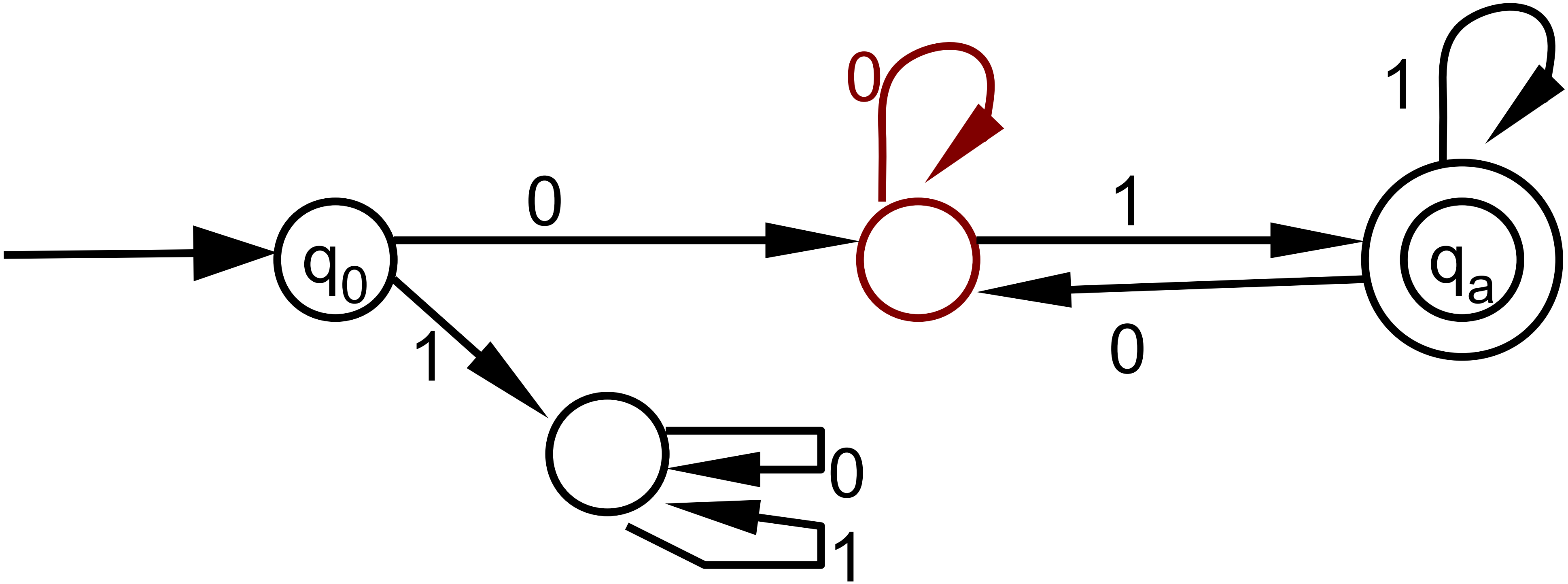
DFA (Deterministic Finite Automata)



Example: Input string

w = **0**011

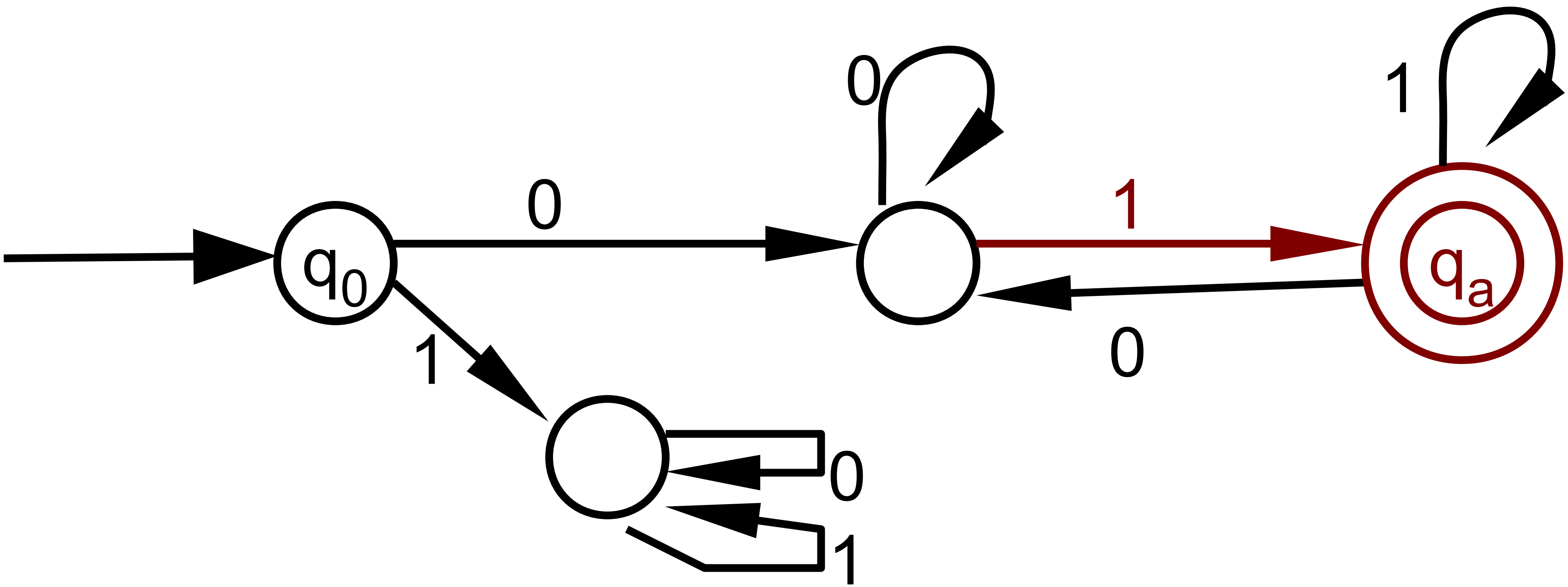
DFA (Deterministic Finite Automata)



Example: Input string

$w = 0\mathbf{0}11$

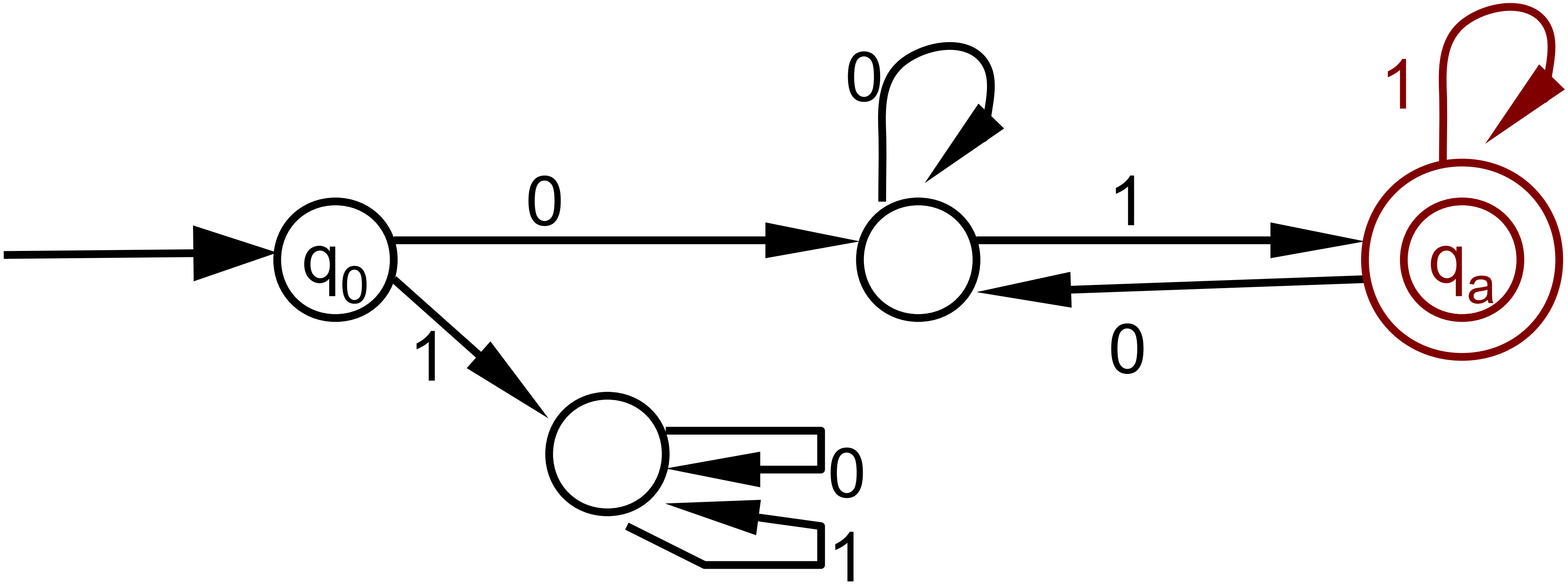
DFA (Deterministic Finite Automata)



Example: Input string

$w = 0011$

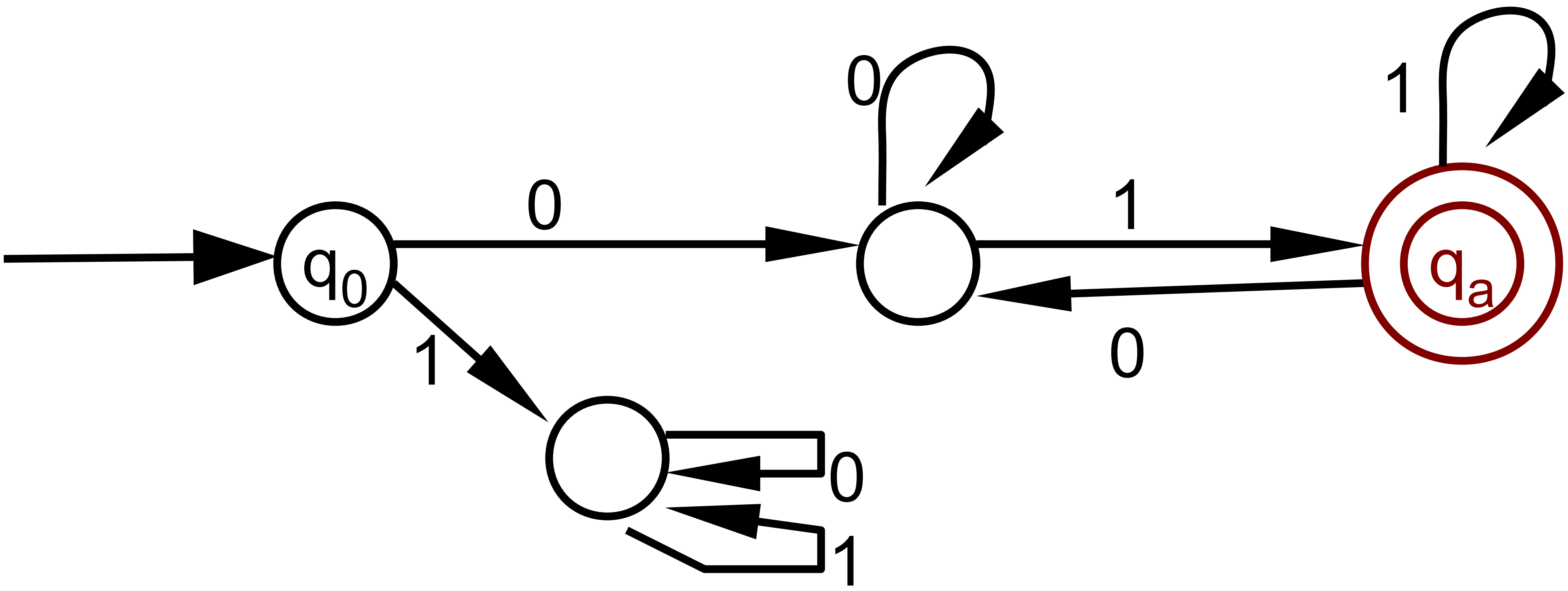
DFA (Deterministic Finite Automata)



Example: Input string

w = 001**1**

DFA (Deterministic Finite Automata)



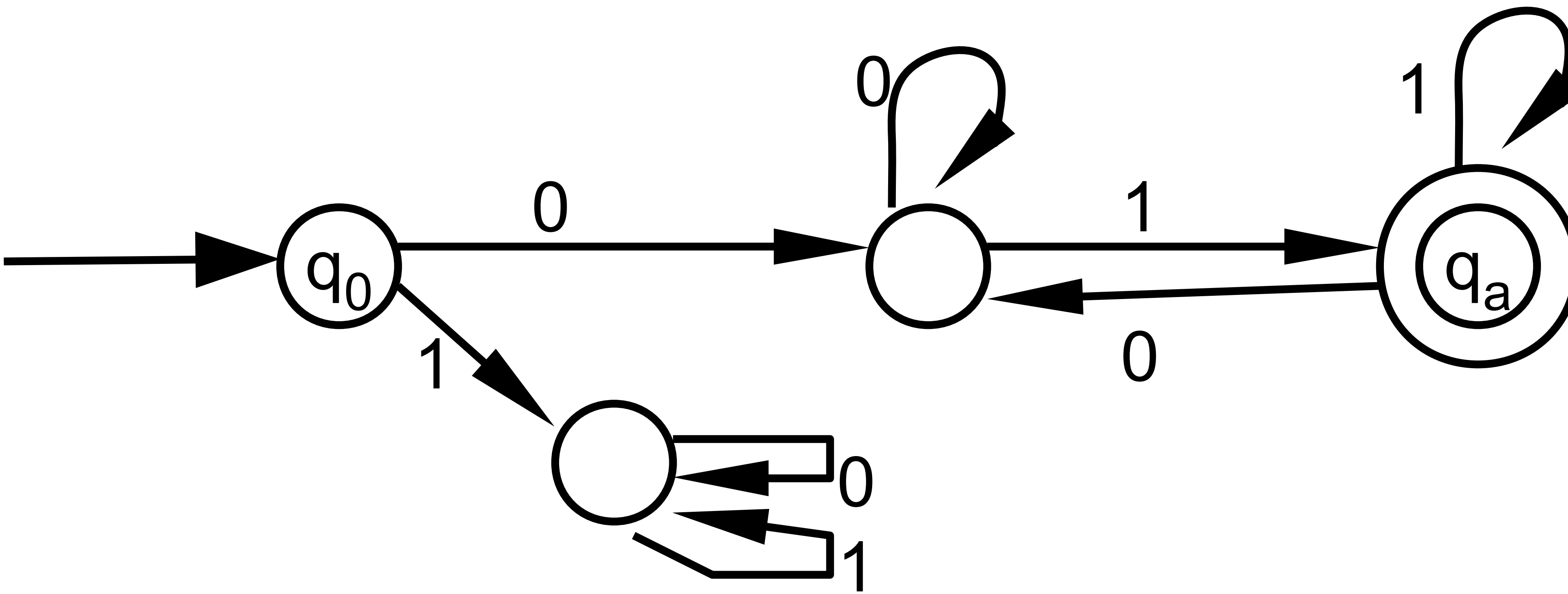
Example: Input string

w = 0011 **ACCEPT**

because end in

accept state

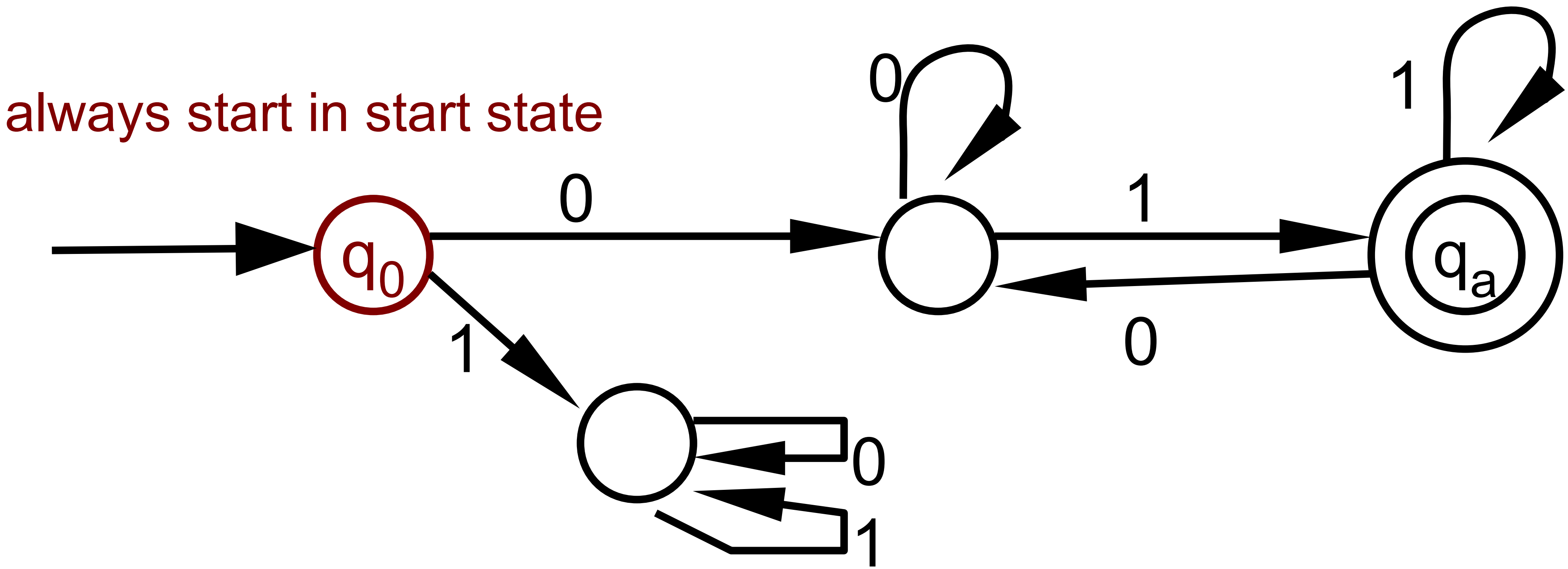
DFA (Deterministic Finite Automata)



Example: Input string

$w = 010$

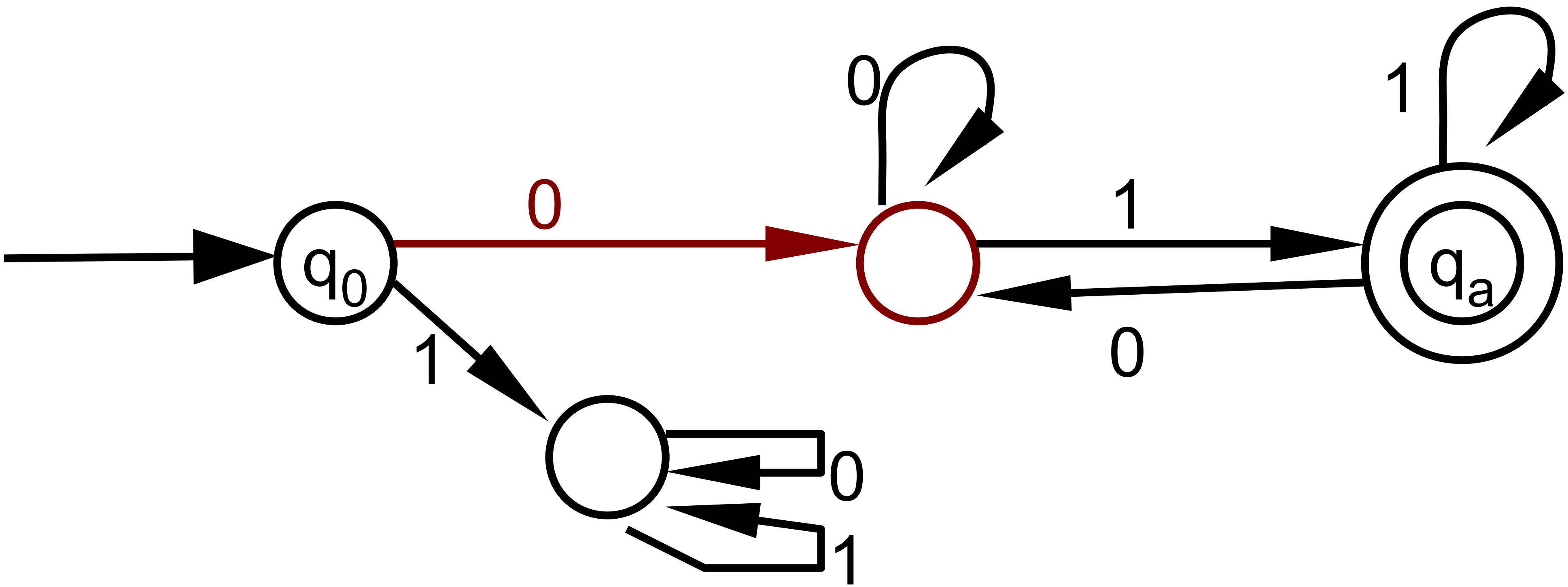
DFA (Deterministic Finite Automata)



Example: Input string

w = 010

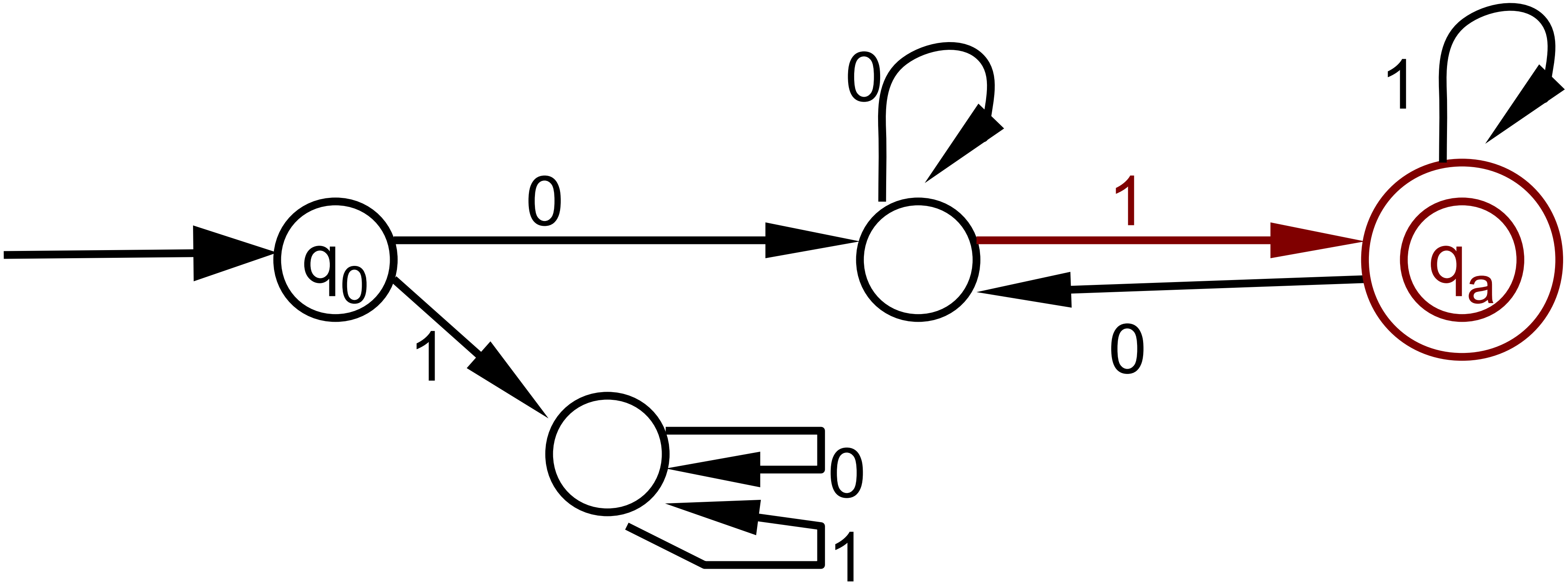
DFA (Deterministic Finite Automata)



Example: Input string

$w = \mathbf{0}10$

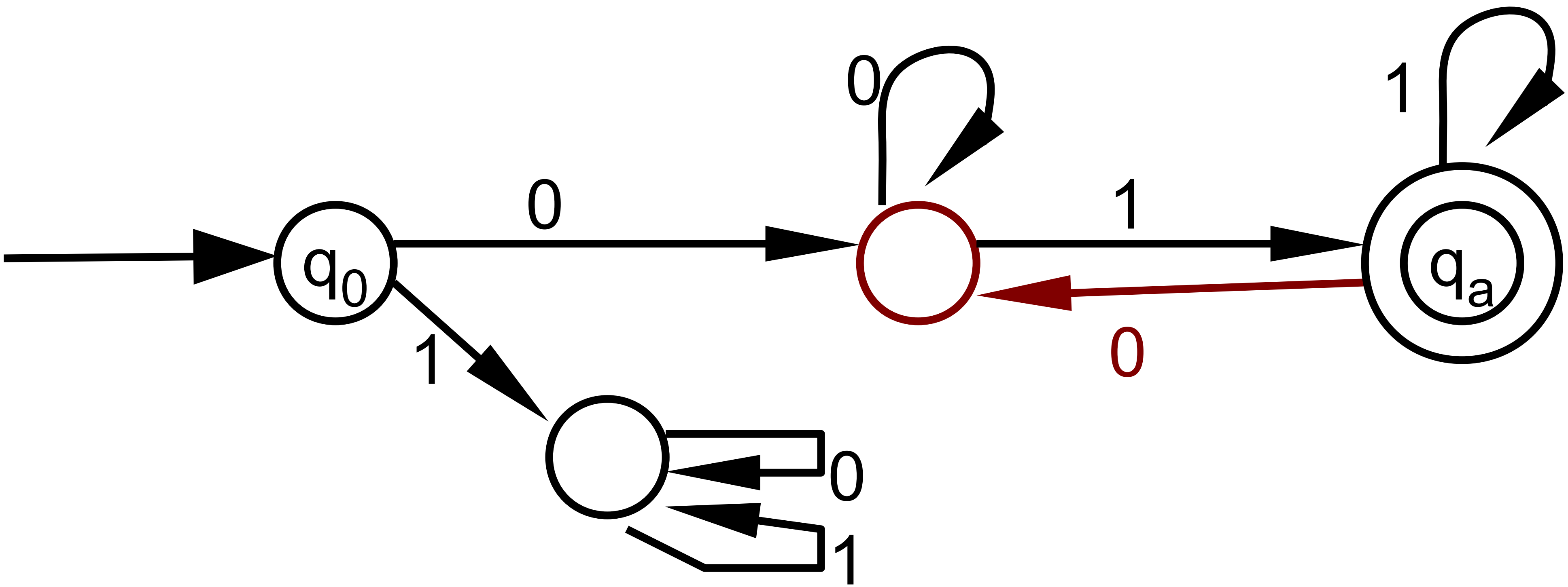
DFA (Deterministic Finite Automata)



Example: Input string

$w = 010$

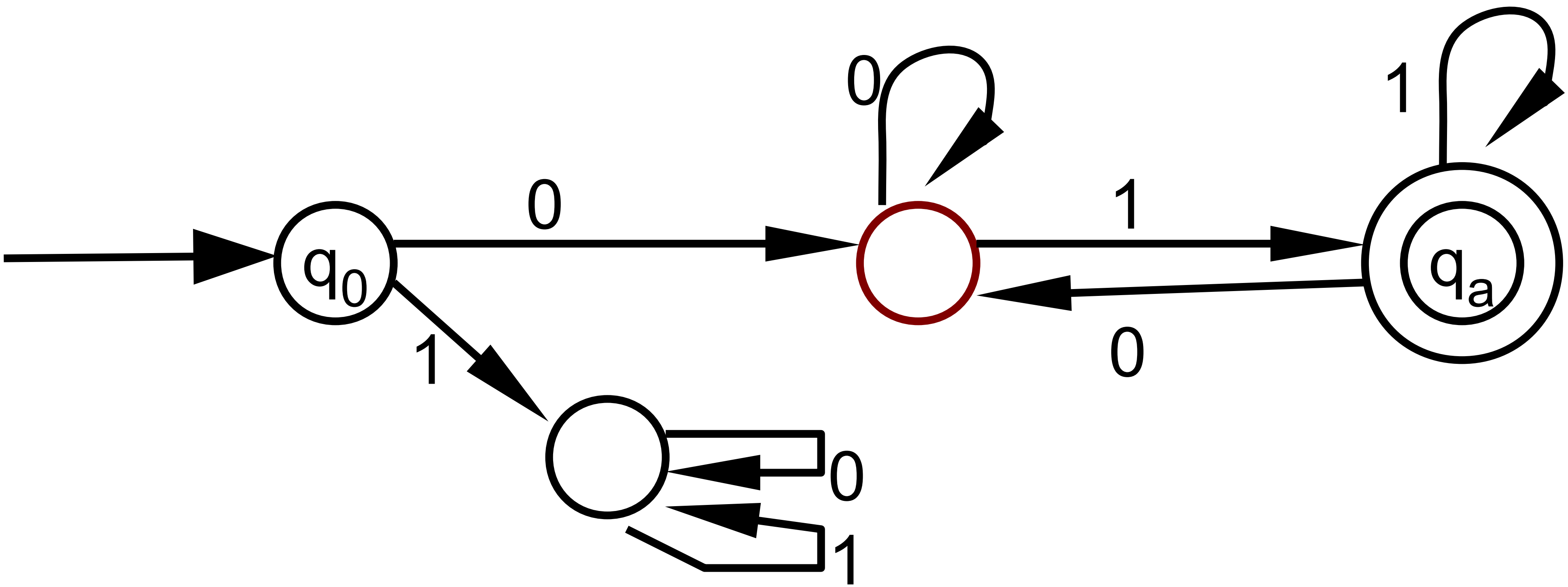
DFA (Deterministic Finite Automata)



Example: Input string

$w = 01\mathbf{0}$

DFA (Deterministic Finite Automata)



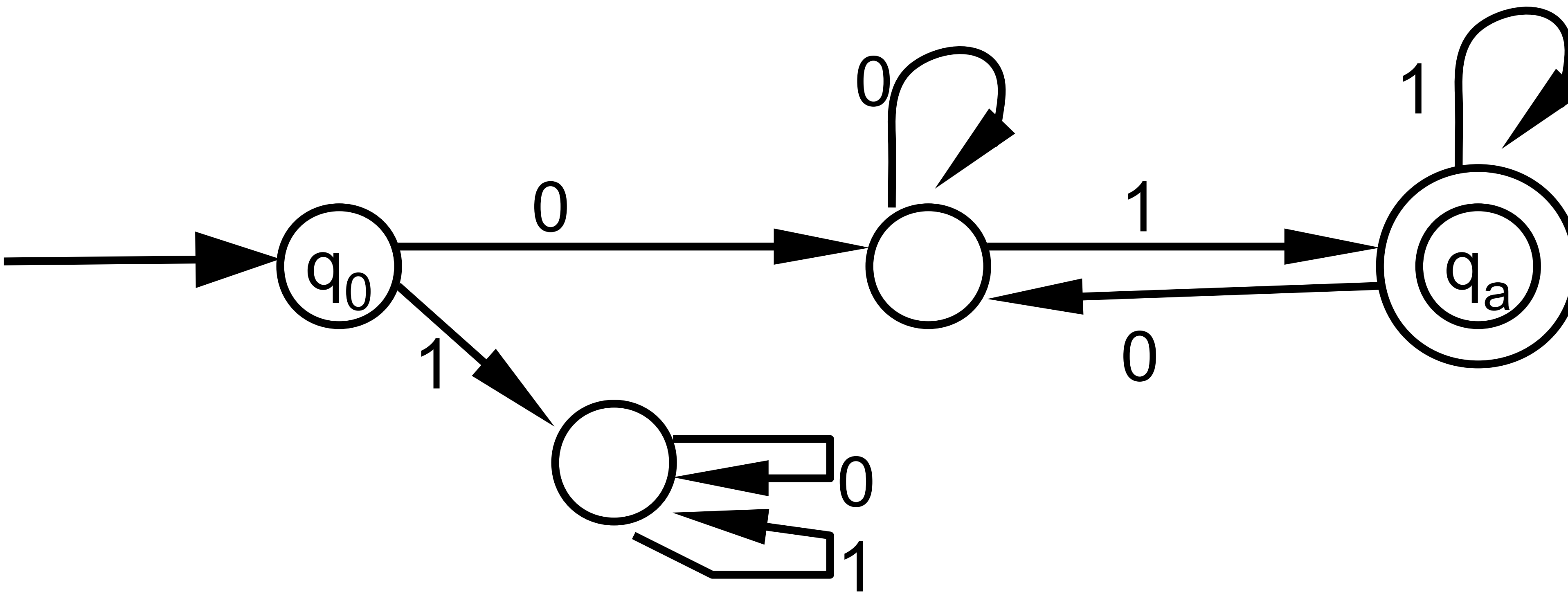
Example: Input string

$w = 010$ **REJECT**

because does not

end in accept state

DFA (Deterministic Finite Automata)



Example: Input string w = 01 ACCEPT

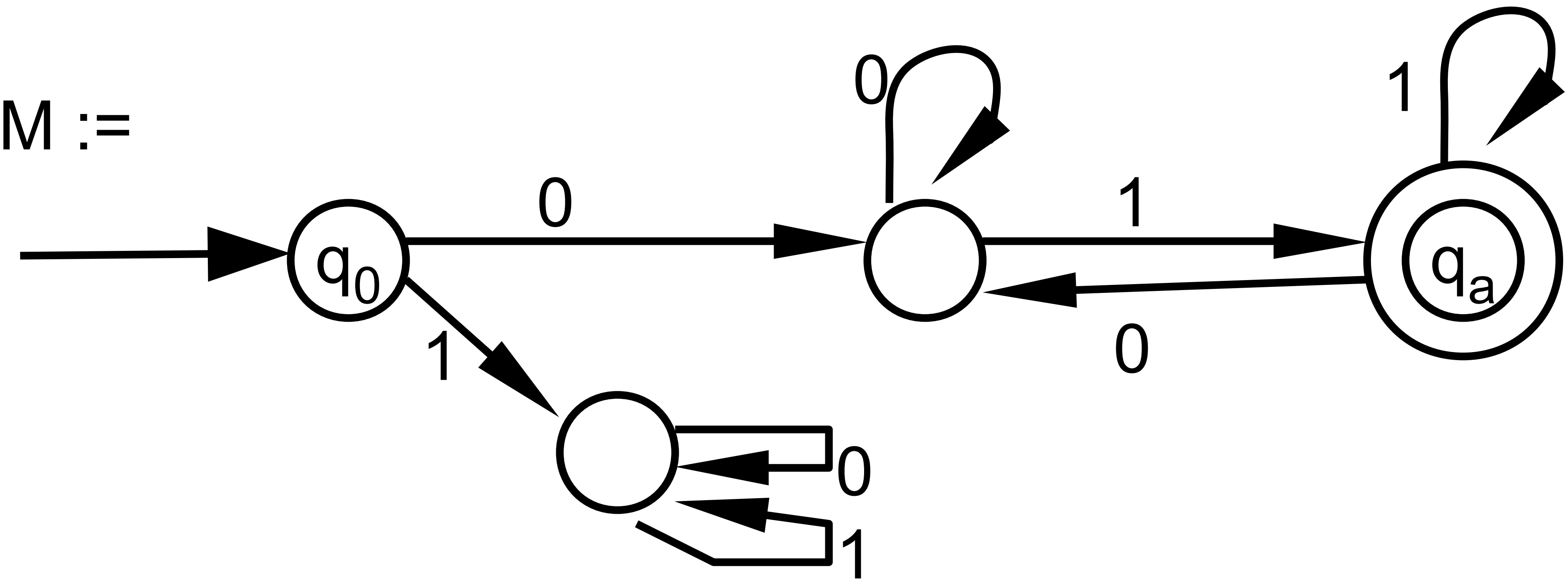
w = 010 REJECT

w = 0011 ACCEPT

w = 00110 REJECT

DFA (Deterministic Finite Automata)

M :=



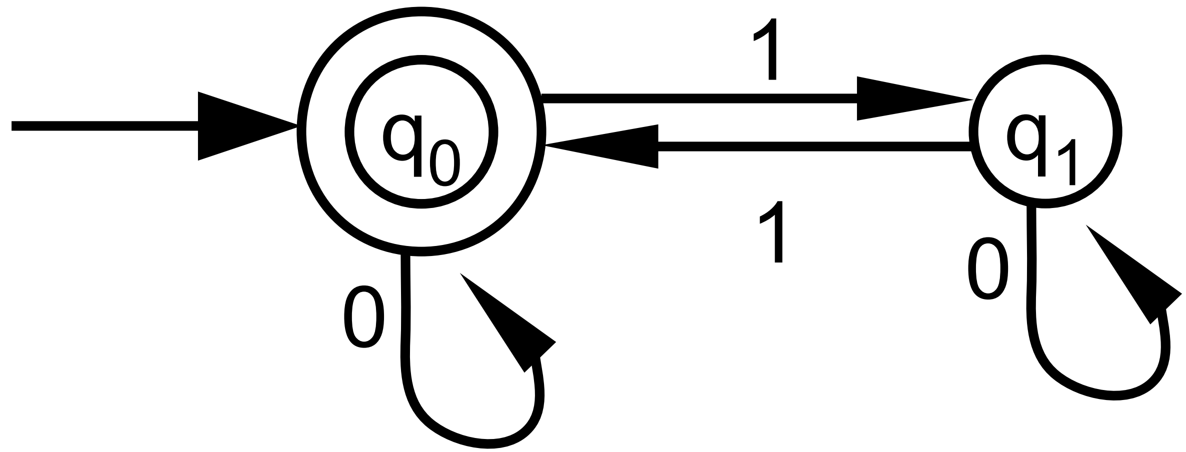
M recognizes language

L(M) = { w : w starts with 0 and ends with 1 }

L(M) is the language of strings causing M to accept

Example: 0101 is an element of L(M), 0101 $\in$ L(M)

Example

M := 

$\Sigma = \{0,1\}$

- 00 causes M to accept, so 00 is in L(M) $00 \in L(M)$
- 01 does not cause M to accept, so 01 not in L(M),

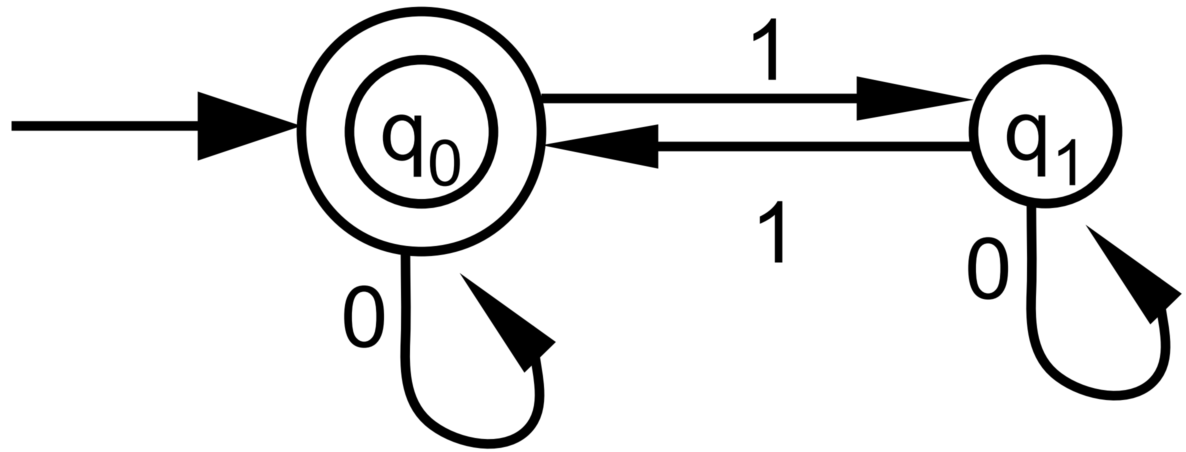
  $01 \notin L(M)$

- 0101 $\in L(M)$
- 01101100 $\in L(M)$
- 011010 $\notin L(M)$

Example

$\Sigma = \{0,1\}$

M :=



L(M) = {w : w has an even number of 1 }

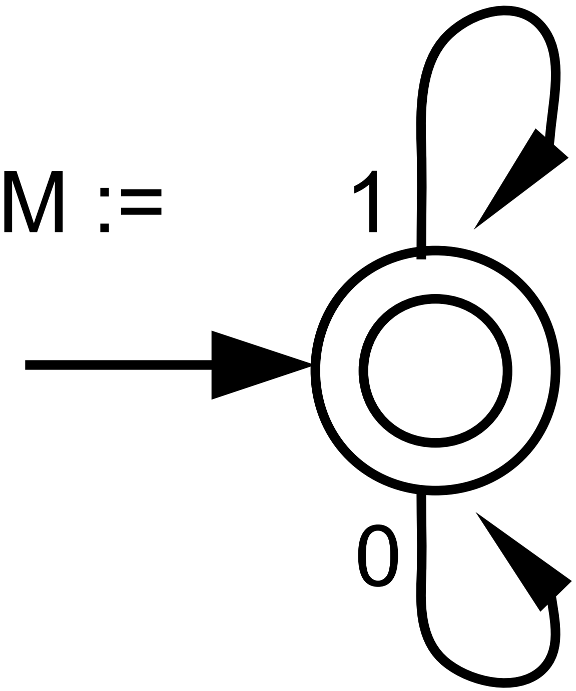
Note: If there is no 1, then there are zero 1,

zero is an even number, so M should accept.

Indeed 0000000 $\in$ L(M)

Example

M :=

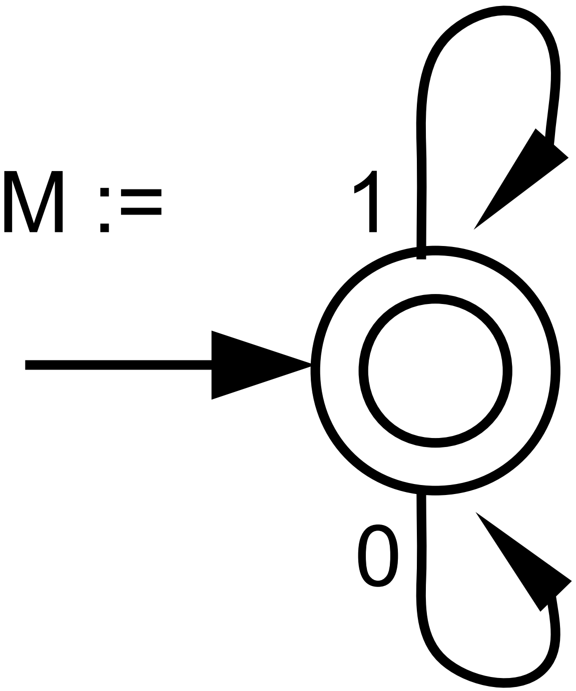


$\Sigma = \{0,1\}$

- L(M) = ?

Example

$M :=$



$\Sigma = \{0,1\}$

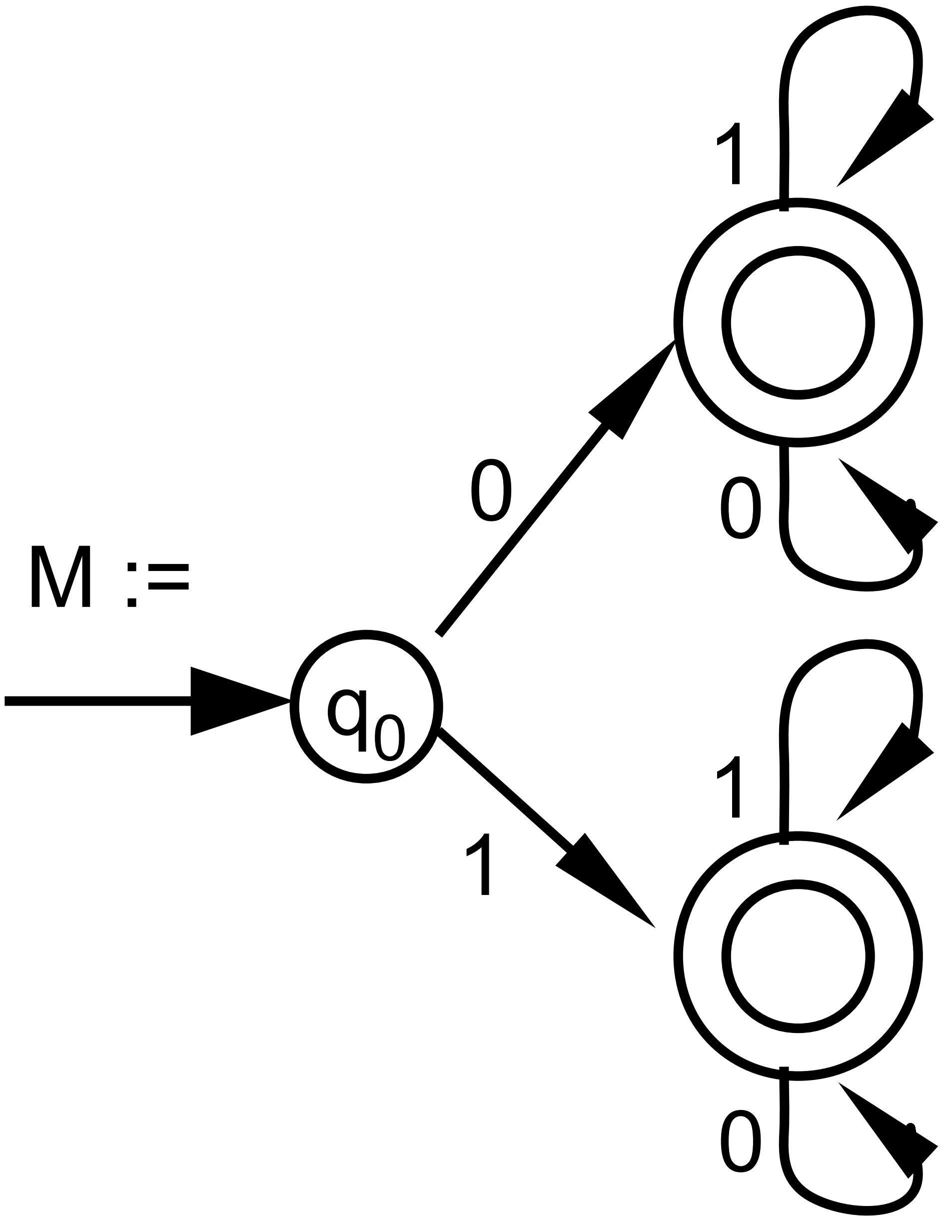
- L(M) = every possible string over {0,1}

  $= \{0,1\}^*$

Example

$\Sigma = \{0,1\}$

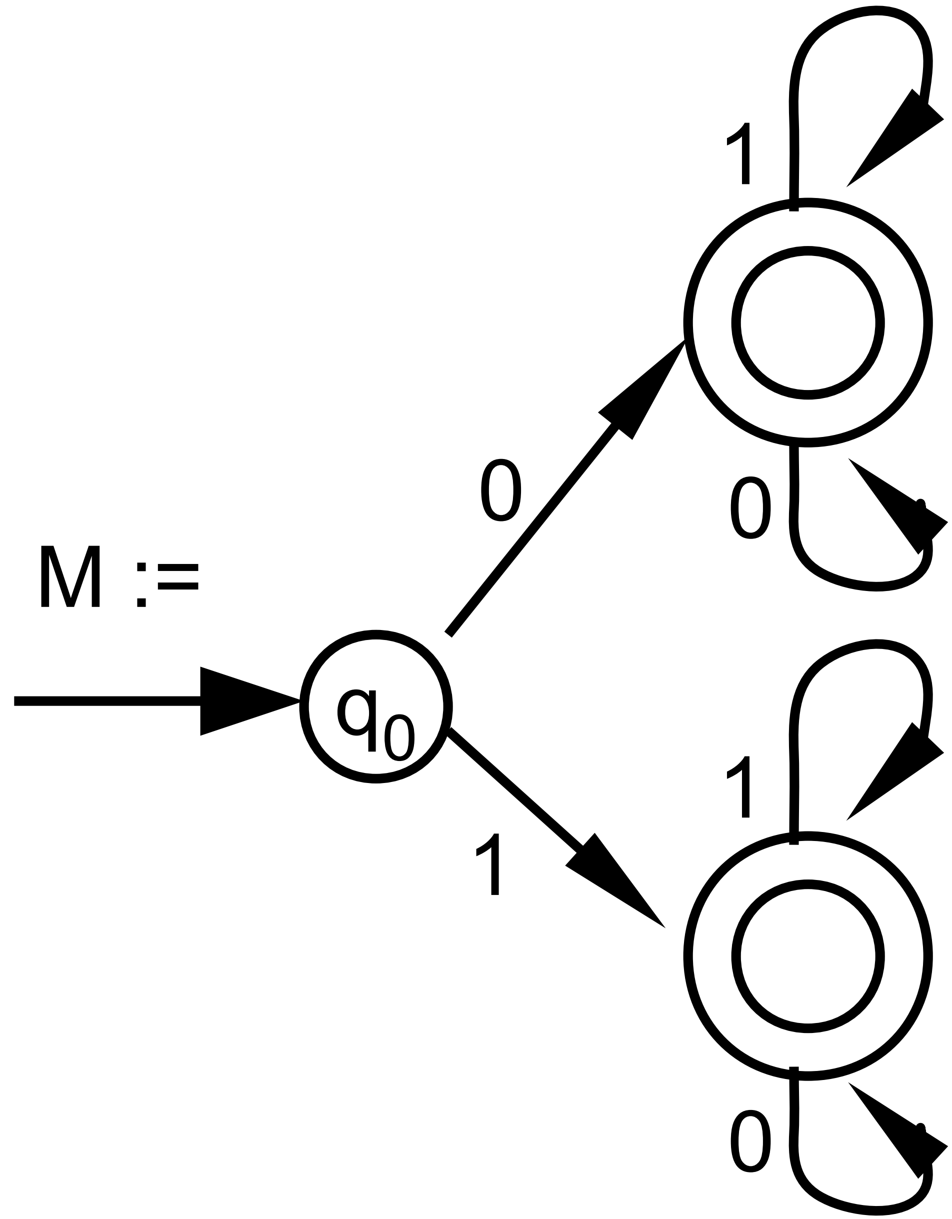
M :=



- L(M) = ?

Example

$\Sigma = \{0,1\}$



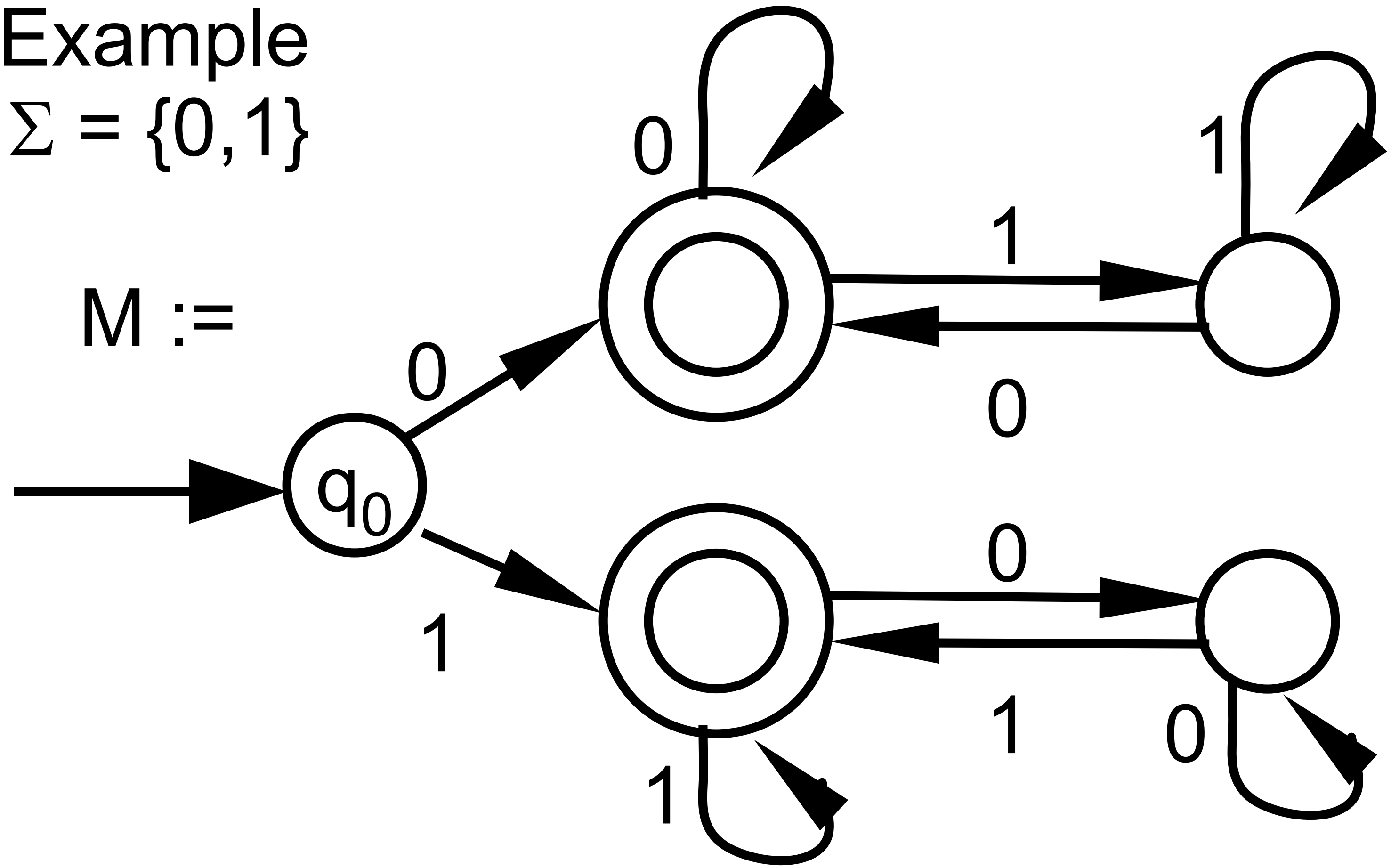
- L(M) = all strings over {0,1} except empty string ε

  $= \{0,1\}^* - \{ \varepsilon \}$

Example

$\Sigma = \{0,1\}$

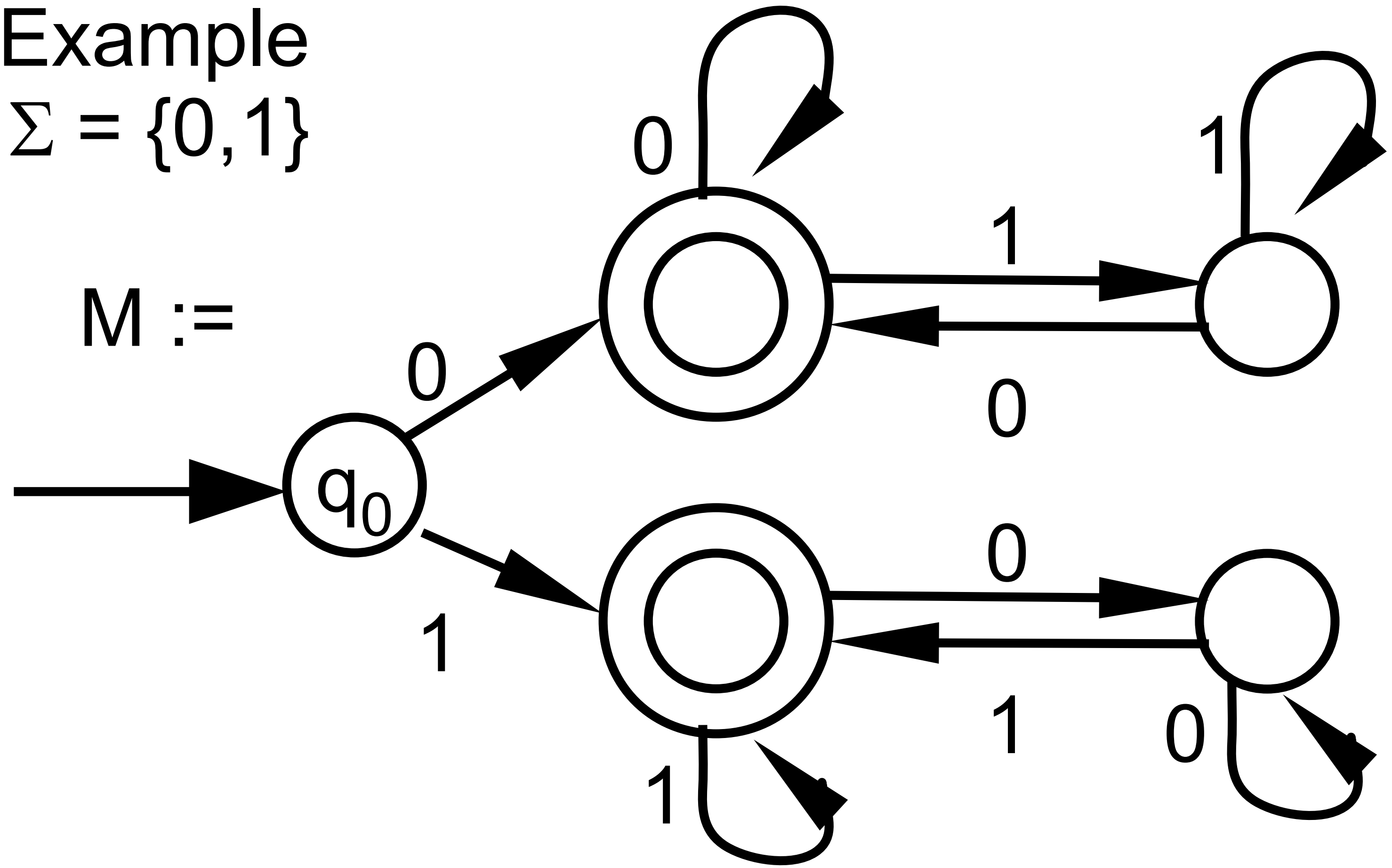
M :=



- L(M) = ?

Example
$\Sigma = \{0,1\}$

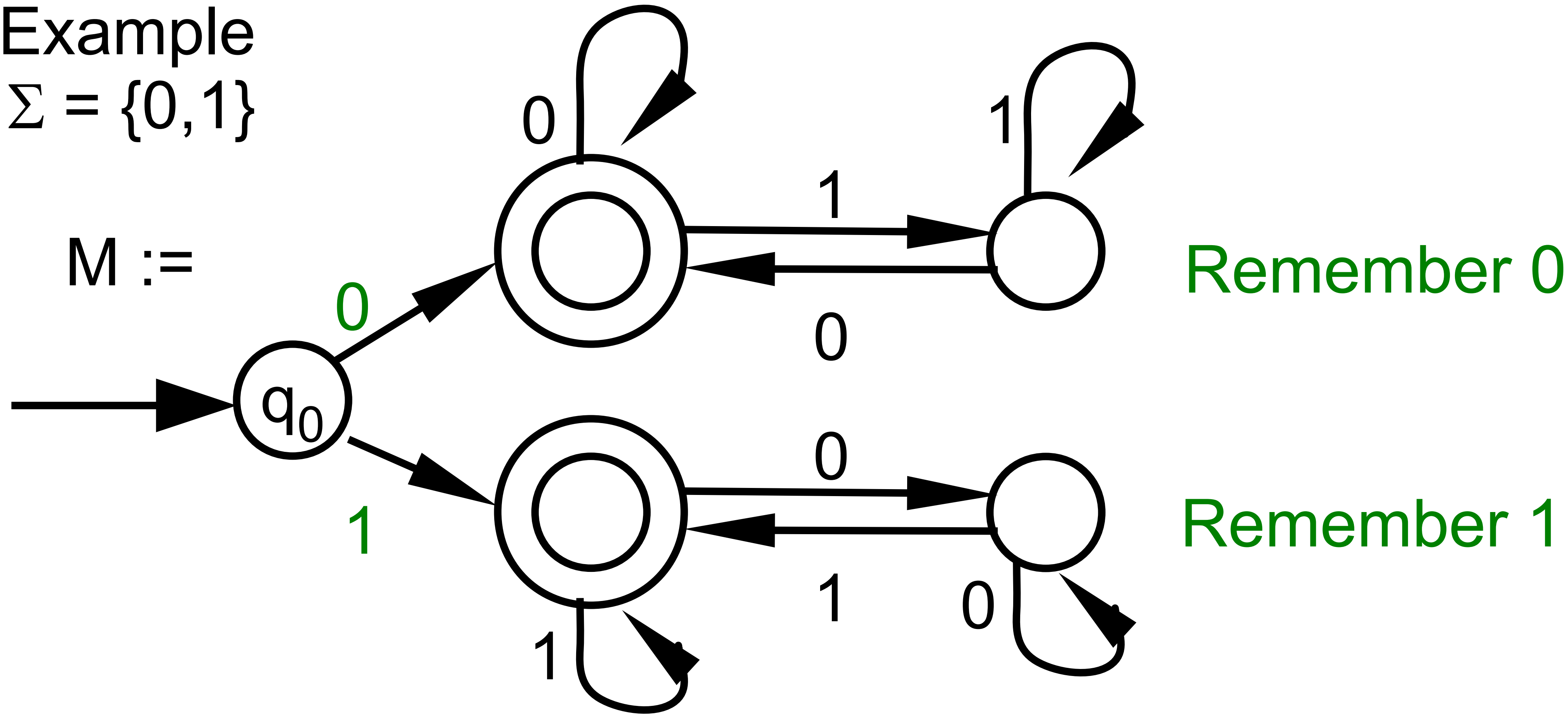
M :=



- L(M) = { w : w starts and ends with same symbol }
- Memory is encoded in … what ?

Example
$\Sigma = \{0,1\}$

M :=

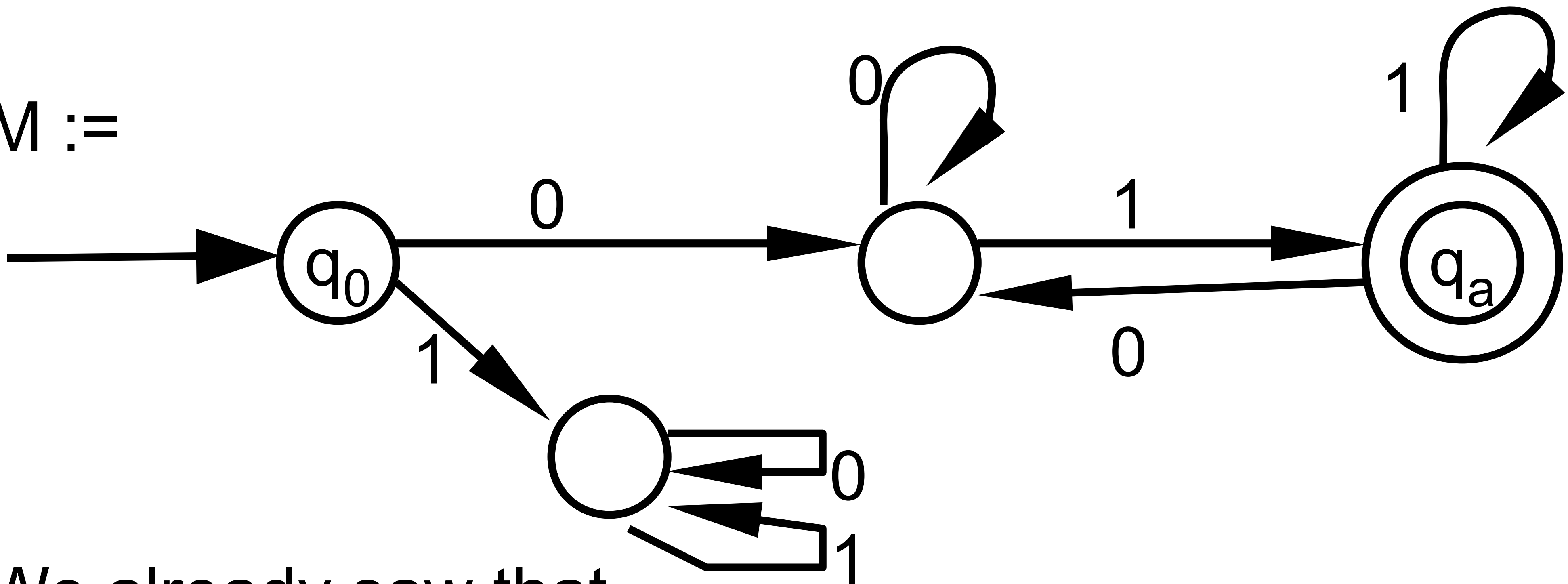


- L(M) = { w : w starts and ends with same symbol }
- Memory is encoded in states.

DFA have finite states, so finite memory

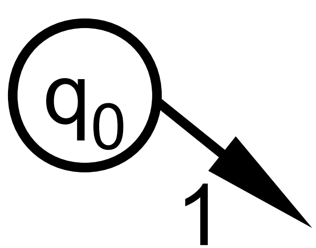
Convention:

M :=



We already saw that

L(M) = { w : w starts with 0 and ends with 1 }

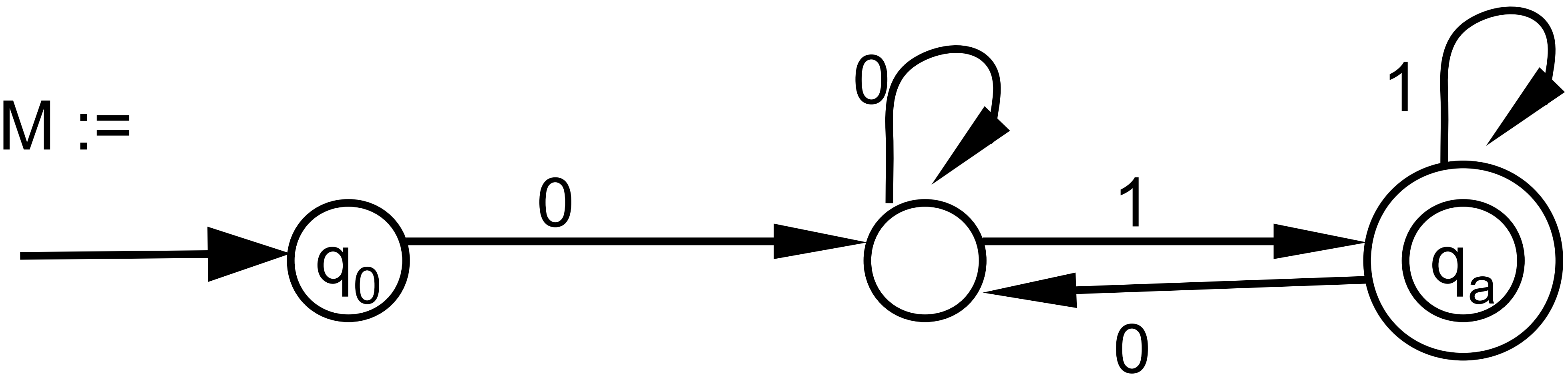
The arrow 

leads to a "sink" state.

If followed, M can never accept

Convention:

M :=



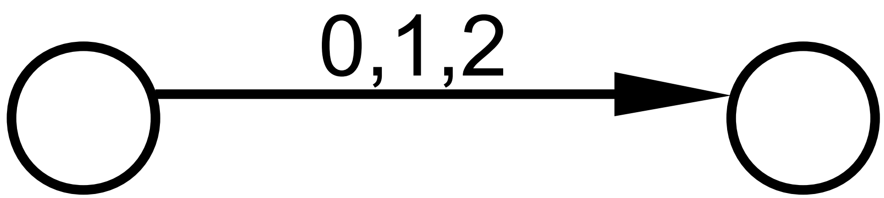
Don't need to write such arrows:

If, from some state, read symbol with no corresponding arrow, imagine M goes into "sink state" that is not shown, and REJECT.

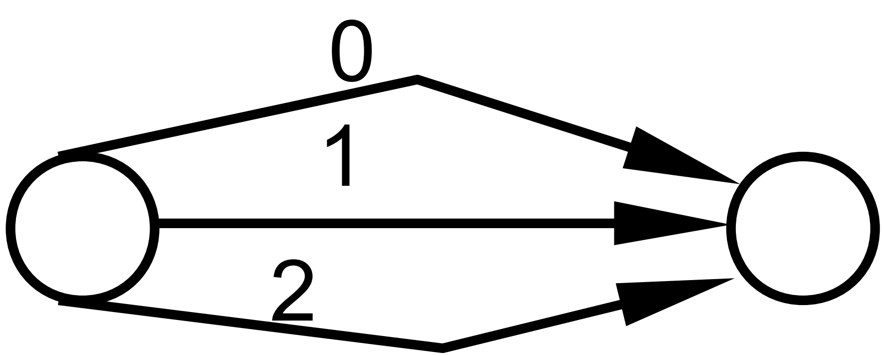
This makes pictures more compact.

Another convention:

List multiple transition on same arrow:



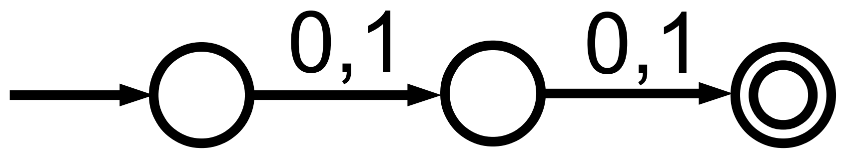
Means



This makes pictures more compact.

Example $\sum = \{0,1\}$

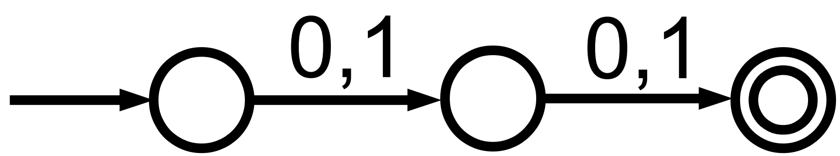
M =



L(M) = ?

Example $\Sigma = \{0,1\}$

M =

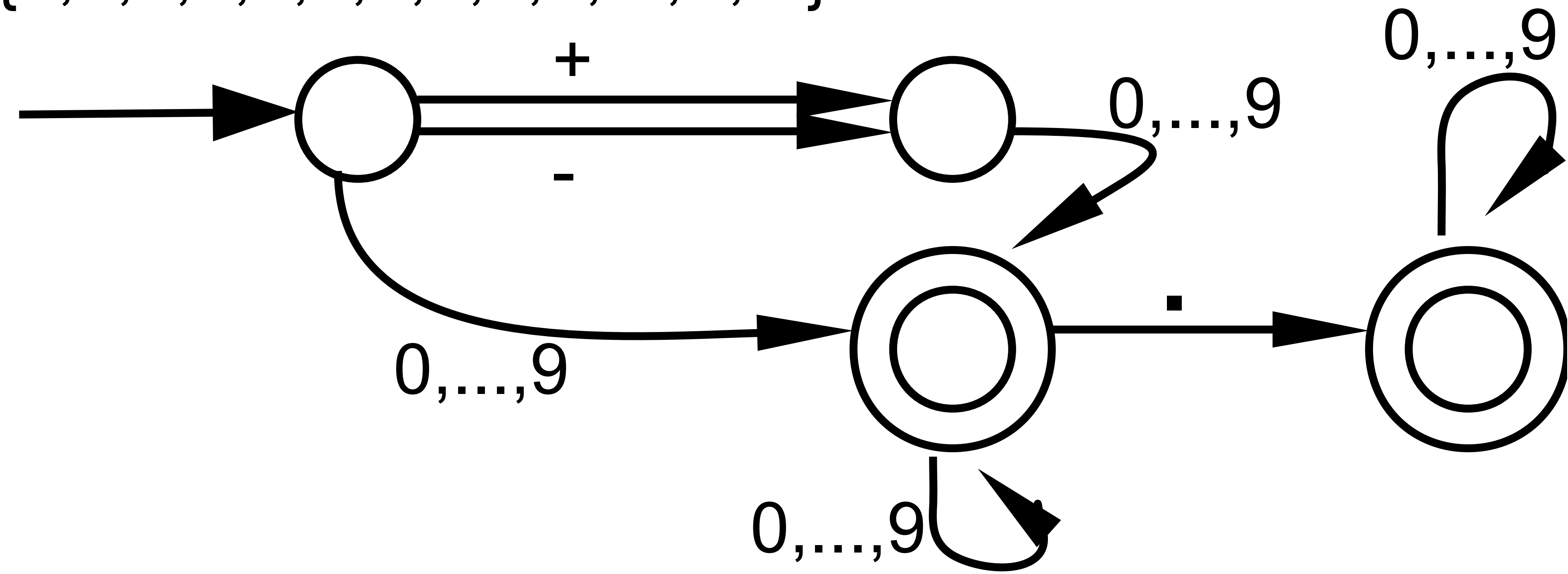


$L(M) = \Sigma^2 = \{00,01,10,11\}$

Example from programming languages:

Recognize strings representing numbers:

$\Sigma = \{0,1,2,3,4,5,6,7,8,9, +, -, .\}$

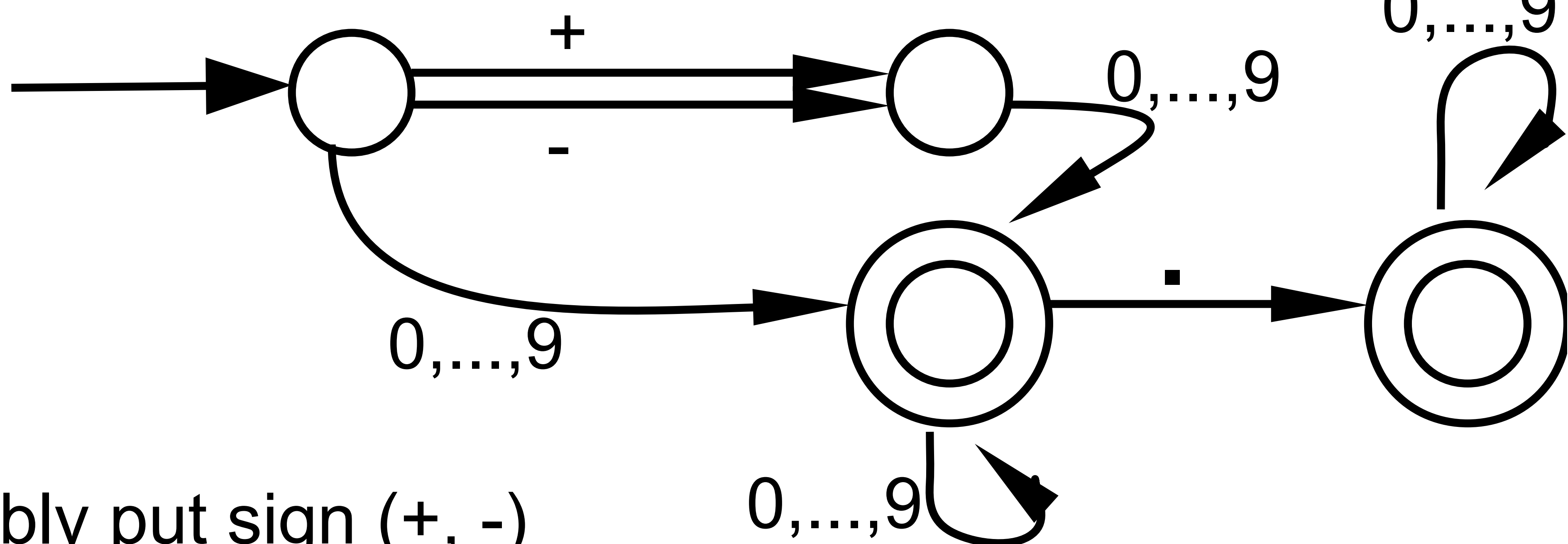


Note: 0,...,9 means 0,1,2,3,4,5,6,7,8,9: 10 transitions

Example from programming languages:

Recognize strings representing numbers:

$\Sigma = \{0,1,2,3,4,5,6,7,8,9, +, -, .\}$



Possibly put sign (+, -)

Follow with arbitrarily many digits, but at least one

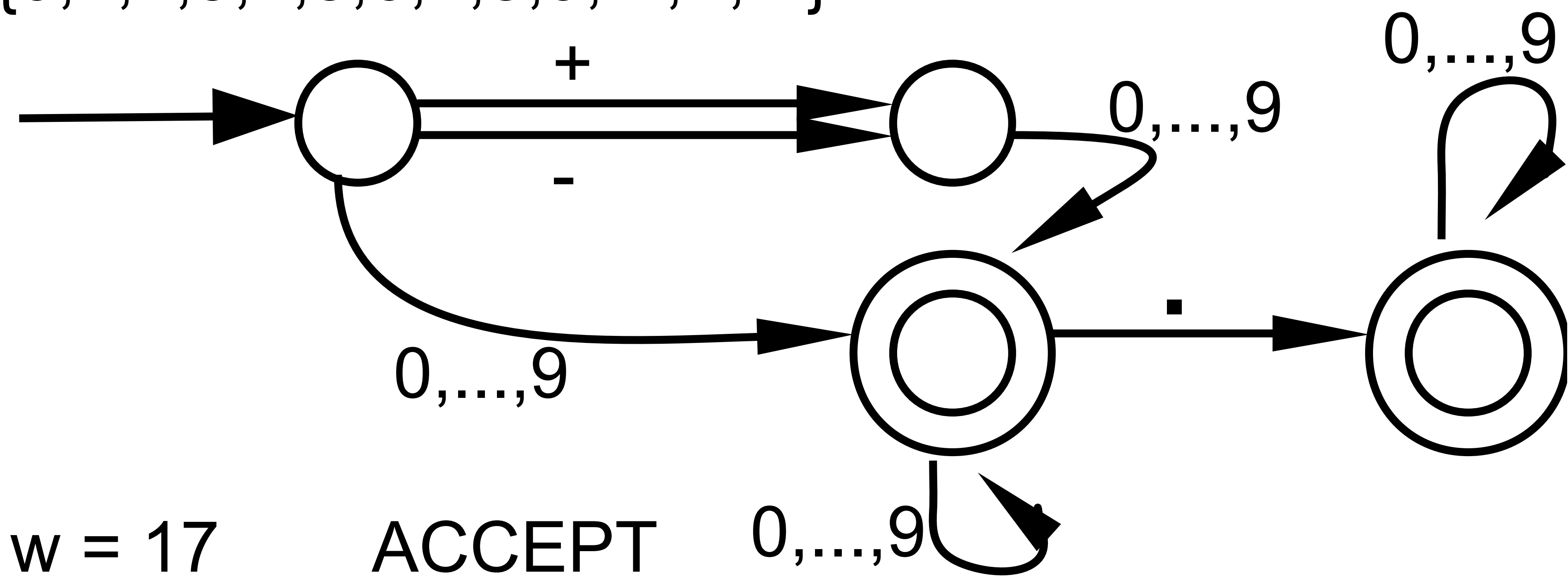
Possibly put decimal point

Follow with arbitrarily many digits, possibly none

Example from programming languages:

Recognize strings representing numbers:

$\Sigma = \{0,1,2,3,4,5,6,7,8,9, +, -, .\}$



Input w = 17 ACCEPT

Input w = + REJECT

Input w = -3.25 ACCEPT

Input w = +2.35-. REJECT

Example

$\Sigma = \{0,1\}$

- What about { w : w has same number of 0 and 1 }

- Can you design a DFA that recognizes that?

- It seems you need infinite memory

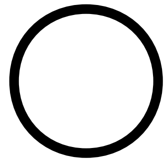
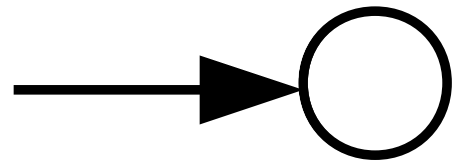
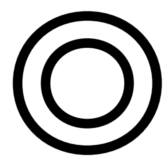
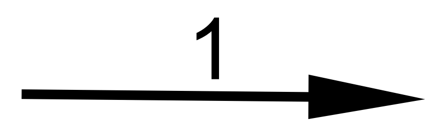
- We will prove later that

  there is no DFA that recognizes that language !

Next: formal definition of DFA

- Useful to prove various properties of DFA
- Especially important to prove that things CANNOT be
  recognized by DFA.
- Useful to practice mathematical notation

State diagram of a DFA:

- One or more states 
- Exactly one start state 
- Some number of accept states 
- Labelled transitions exiting each state, 

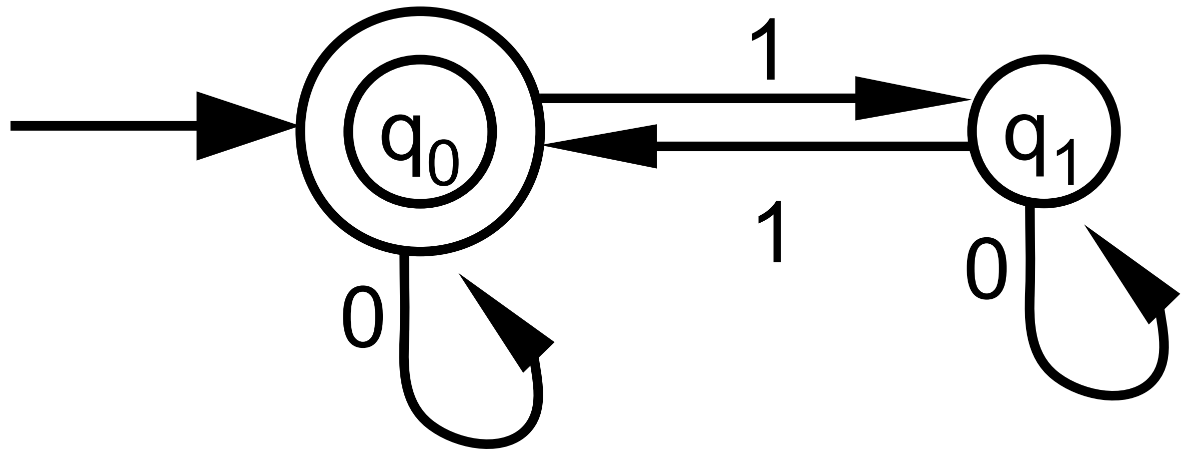
  for every symbol in Σ

- Definition: A finite automaton (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

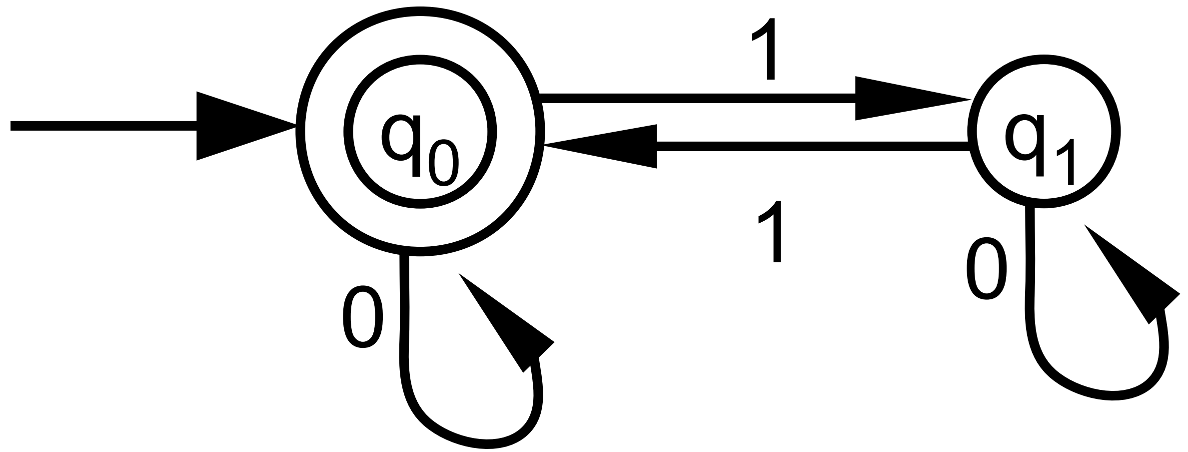
- Q is a finite set of states
- Σ is the input alphabet
- $\delta : Q \times \Sigma \rightarrow Q$ is the transition function
- q_0 in Q is the start state
- $F \subseteq Q$ is the set of accept states

$Q \times \Sigma$ is the set of ordered pairs $(a,b) : a \in Q,\ b \in \Sigma$

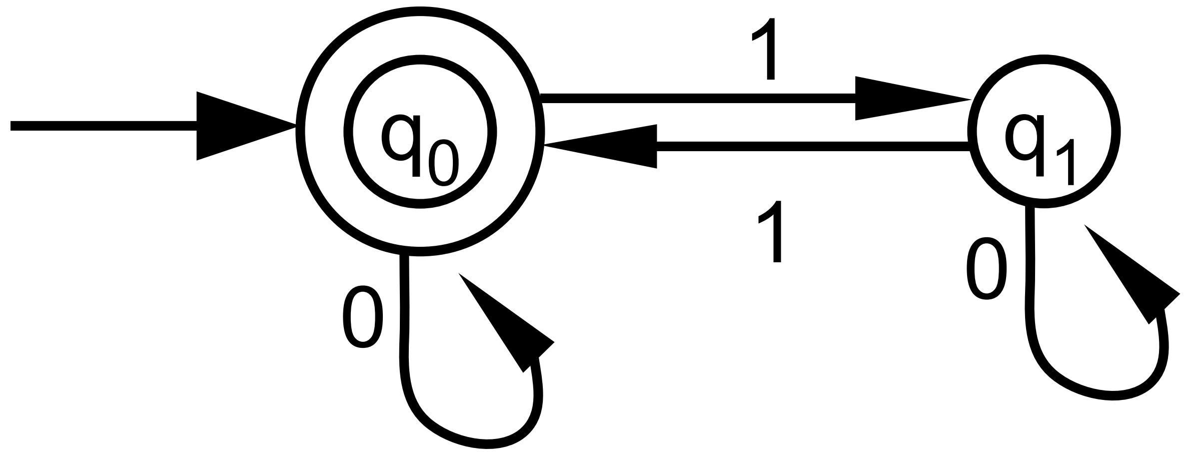
Example $\{q,r,s\} \times \{0,1\} = \{(q,0),(q,1),(r,0),(r,1),(s,0),(s,1)\}$



- Example: above DFA is 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where
- $Q = \{ q_0, q_1 \}$
- $\Sigma = \{0,1\}$
- $\delta(q_0 ,0) = ?$

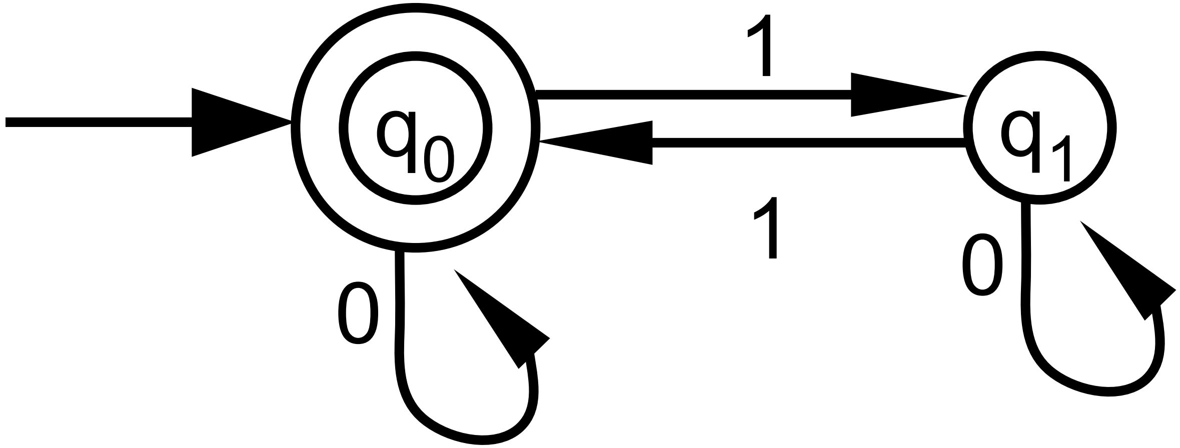


- Example: above DFA is 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where
- $Q = \{ q_0, q_1\}$
- $\Sigma = \{0,1\}$
- $\delta(q_0, 0) = q_0 \quad \delta(q_0, 1) = ?$



- Example: above DFA is 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where
- $Q = \{ q_0, q_1\}$
- $\Sigma = \{0,1\}$
- $\delta(q_0 ,0) = q_0$ $\quad \delta(q_0 ,1) = q_1$

  $\delta(q_1 ,0) = q_1$ $\quad \delta(q_1 ,1) = q_0$
- q_0 in Q is the start state
- F = ?



- Example: above DFA is 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where
- $Q = \{ q_0, q_1\}$
- $\Sigma = \{0,1\}$
- $\delta(q_0 ,0) = q_0 \quad \delta(q_0 ,1) = q_1$

  $\delta(q_1 ,0) = q_1 \quad \delta(q_1 ,1) = q_0$
- q_0 in Q is the start state
- $F = \{ q_0\} \subseteq Q$ is the set of accept states

- Definition: A DFA $(Q, \Sigma, \delta, q_0, F)$ accepts a string w if

- $w = w_1\, w_2 \ldots w_k$ where, $\forall\ 1 \leq i \leq k$, w_i is in Σ

(the k symbols of w)

- The sequence of k+1 states $r_0, r_1, .., r_k$ such that:

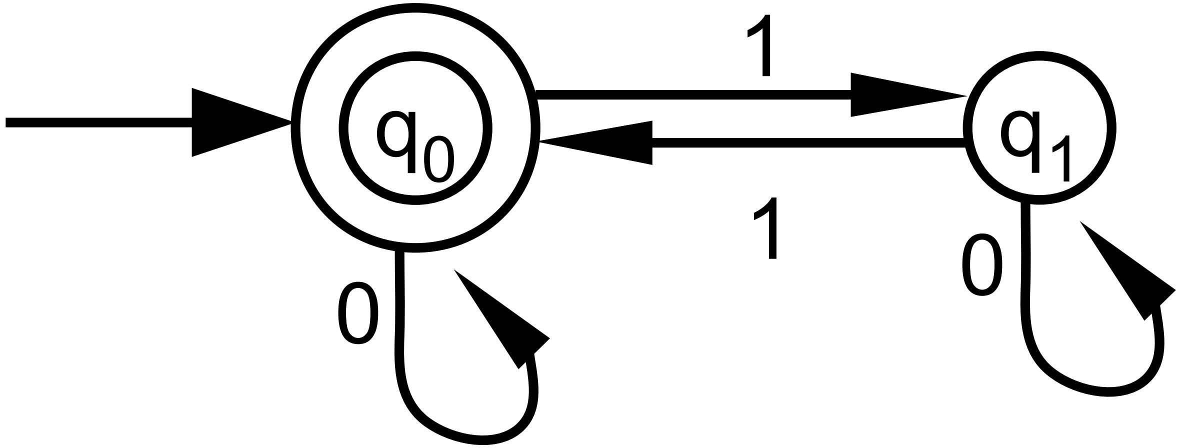
(1) $r_0 = q_0$, and

(2) $r_{i+1} = \delta(r_i, w_{i+1})\ \forall\ 0 \leq i < k$

has r_k in F

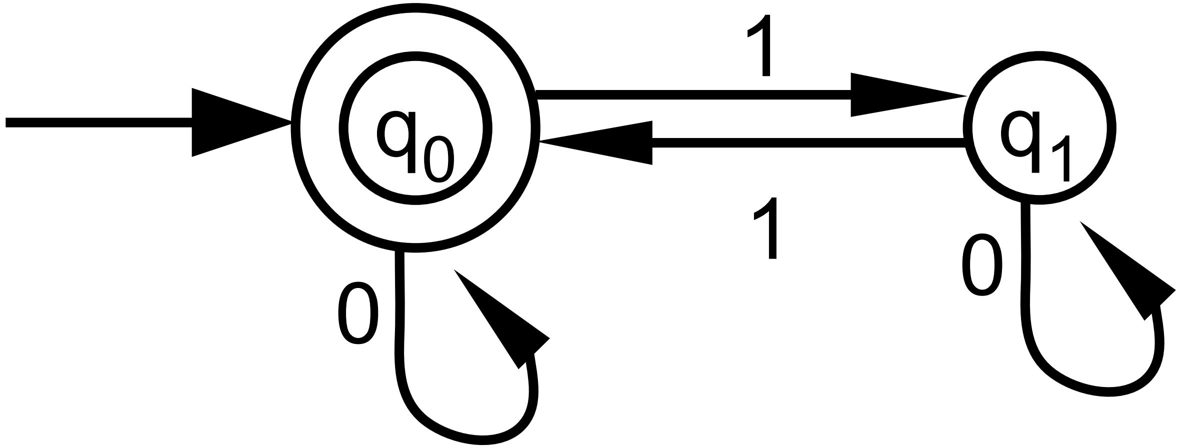
(r_i = state DFA is in after reading i-th symbol in w)

Example



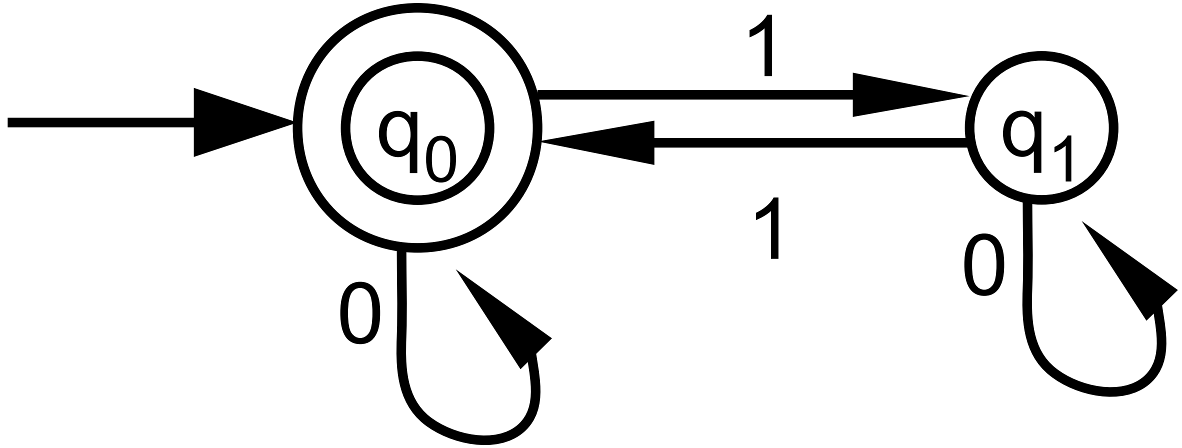
- Above DFA $(Q, \Sigma, \delta, q_0, F)$ accepts w = 011

Example



- Above DFA $(Q, \Sigma, \delta, q_0, F)$ accepts $w = 011$
- $w = 011 = w_1\, w_2\, w_3$ $\qquad w_1 = 0 \quad w_2 = 1 \quad w_3 = 1$

Example



- Above DFA $(Q, \Sigma, \delta, q_0, F)$ accepts $w = 011$
- $w = 011 = w_1\, w_2\, w_3$ $\qquad w_1 = 0 \quad w_2 = 1 \quad w_3 = 1$

We must show that

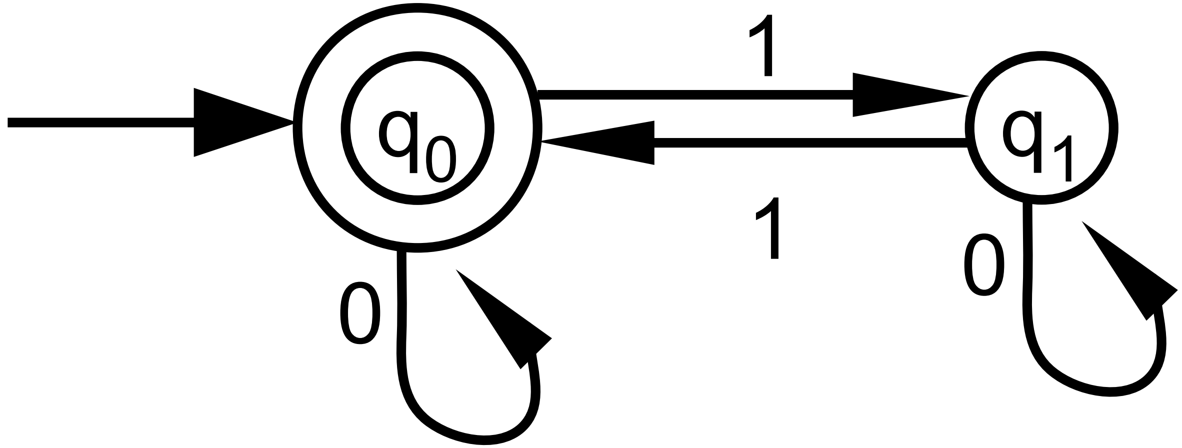
- The sequence of 3+1=4 states r_0, r_1, r_2, r_3 such that:

  (1) $r_0 = q_0$

  (2) $r_{i+1} = \delta(r_i, w_{i+1})\ \forall\ 0 \le i < 3$

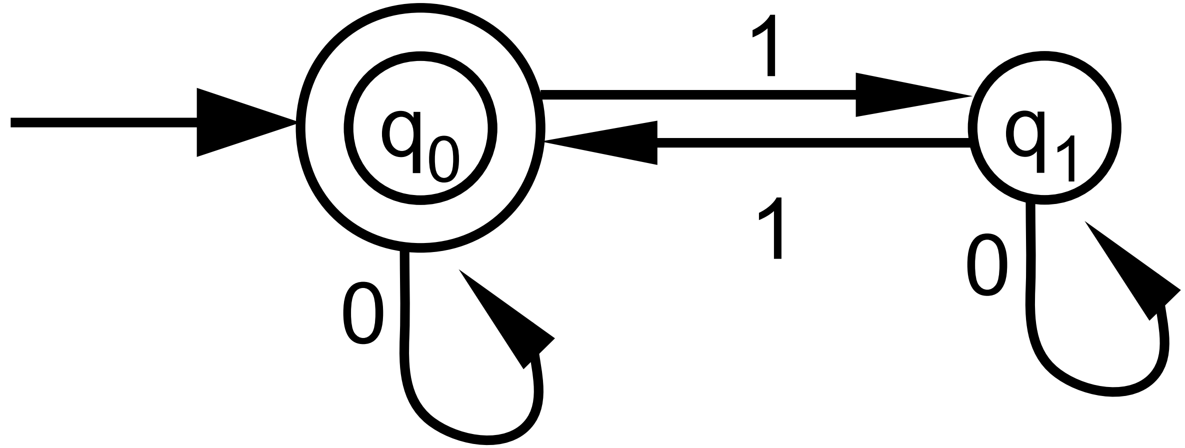
has r_3 in F

Example



- Above DFA (Q, Σ, δ, q_0, F) accepts w = 011
- w = 011 = w_1 w_2 w_3 $w_1 = 0$ $w_2 = 1$ $w_3 = 1$
- $r_0 = q_0$
- r_1 := ?

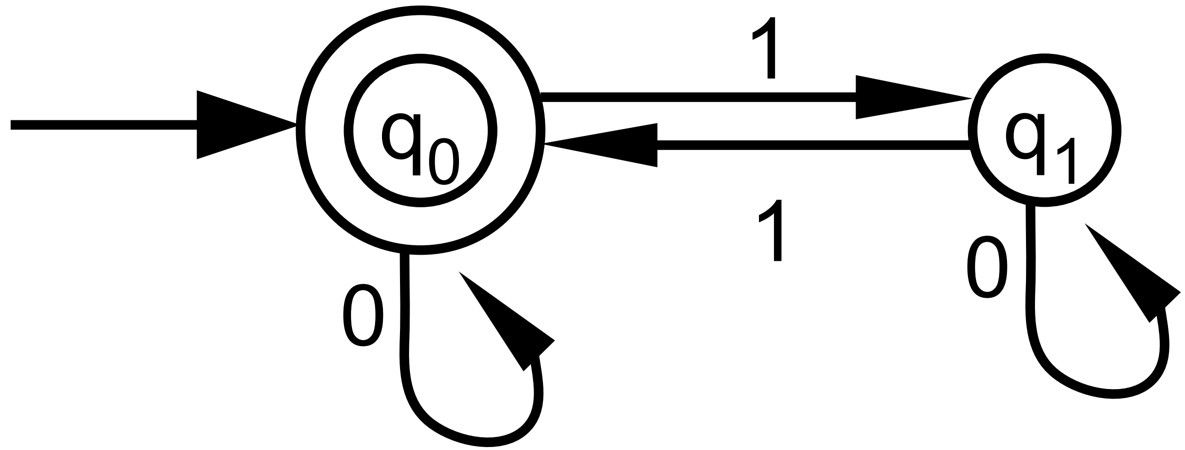
Example



- Above DFA $(Q, \Sigma, \delta, q_0, F)$ accepts $w = 011$
- $w = 011 = w_1\, w_2\, w_3$ $\quad w_1 = 0 \quad w_2 = 1 \quad w_3 = 1$

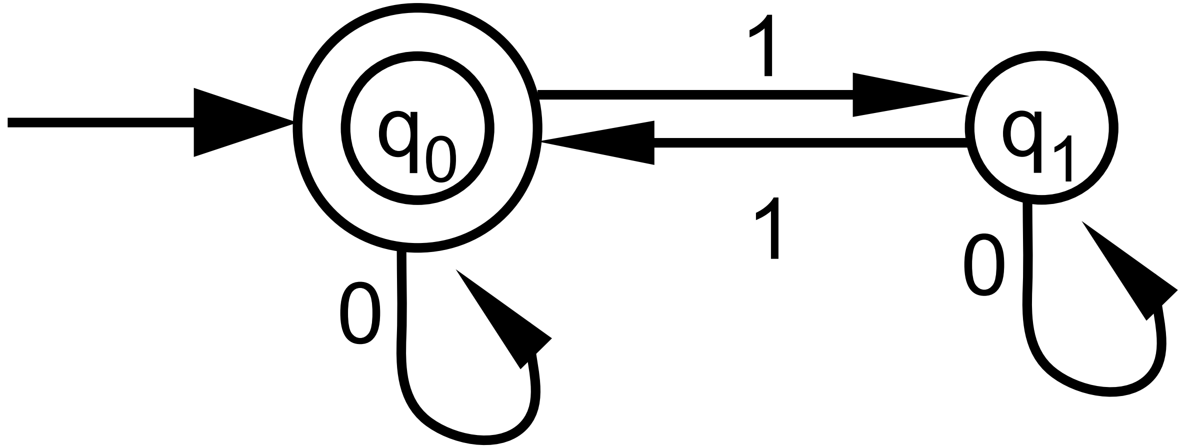
- $r_0 = q_0$
- $r_1 = \delta(r_0, w_1) = \delta(q_0, 0) = q_0$
- $r_2 := ?$

Example



- Above DFA (Q, Σ, δ, q_0, F) accepts w = 011
- $w = 011 = w_1\ w_2\ w_3$ $\qquad w_1 = 0 \quad w_2 = 1 \quad w_3 = 1$
- $r_0 = q_0$
- $r_1 = \delta(r_0, w_1) = \delta(q_0, 0) = q_0$
- $r_2 = \delta(r_1, w_2) = \delta(q_0, 1) = q_1$
- $r_3 :=$?

Example



- Above DFA $(Q, \Sigma, \delta, q_0, F)$ accepts $w = 011$
- $w = 011 = w_1\ w_2\ w_3$ $\qquad w_1 = 0 \quad w_2 = 1 \quad w_3 = 1$

- $r_0 = q_0$
- $r_1 = \delta(r_0 ,w_1)=\delta(q_0 ,0) = q_0$
- $r_2 = \delta(r_1 ,w_2)=\delta(q_0 ,1) = q_1$
- $r_3 = \delta(r_2 ,w_3)=\delta(q_1 ,1) = q_0$
- $r_3 = q_0$ in F $\qquad$ OK $\qquad$ DONE!

- Definition: For a DFA M, we denote by L(M) the set of strings accepted by M:

$$L(M) := \{ w : M \text{ accepts } w\}$$

We say M accepts or recognizes the language L(M)

- Definition: A language L is regular

$$\text{if } \exists \text{ DFA } M : L(M) = L$$

In the next lectures we want to:

- Understand power of regular languages

- Develop alternate, compact notation to specify regular languages

Example: Unix command `grep '\<c.*h\>'` *file* selects all words starting with c and ending with h in *file*

- Understand power of regular languages:

- Suppose A, B are regular languages, what about
- not A := { w : w is not in A }
- A U B := { w : w in A or w in B }
- A o B := { w_1 w_2 : w_1 in A and w_2 in B }
- A* := { w_1 w_2 … w_k : $k \geq 0$, w_i in A for every i }

- Are these languages regular?

- Understand power of regular languages:

- Suppose A, B are regular languages, what about
- not A := { w : w is not in A }
- A U B := { w : w in A or w in B }
- A o B := { w_1 w_2 : w_1 in A and w_2 in B }
- A* := { w_1 w_2 … w_k : $k \geq 0$, w_i in A for every i }

- Terminology: Are regular languages closed under not, U, o, * ?

- Theorem:

  If A is a regular language, then so is (not A)

- Theorem:

  If A is a regular language, then so is (not A)

- Proof idea: ?????????? the set of accept states

- Theorem:

  If A is a regular language, then so is (not A)

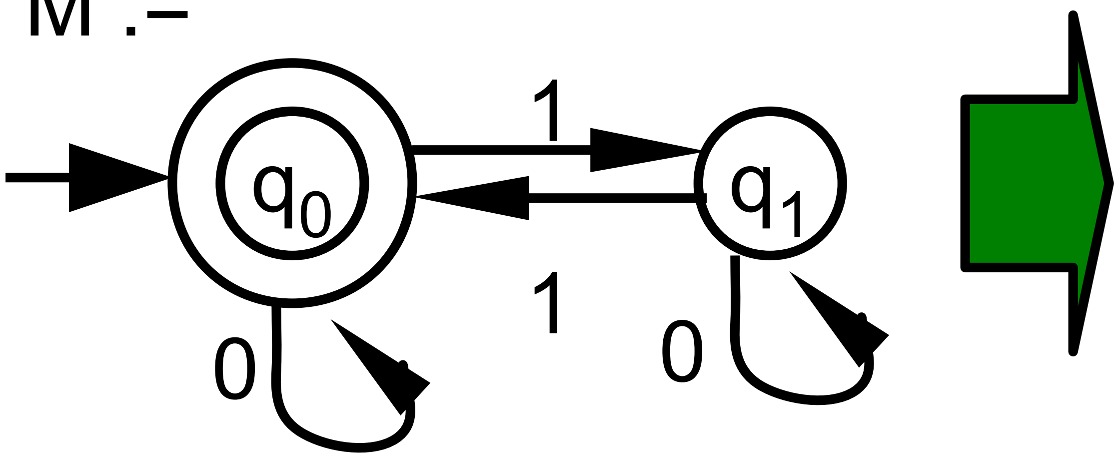
- Proof idea: Complement the set of accept states
- Example

- Theorem:

  If A is a regular language, then so is (not A)

- Proof idea: Complement the set of accept states
- Example:

M :=



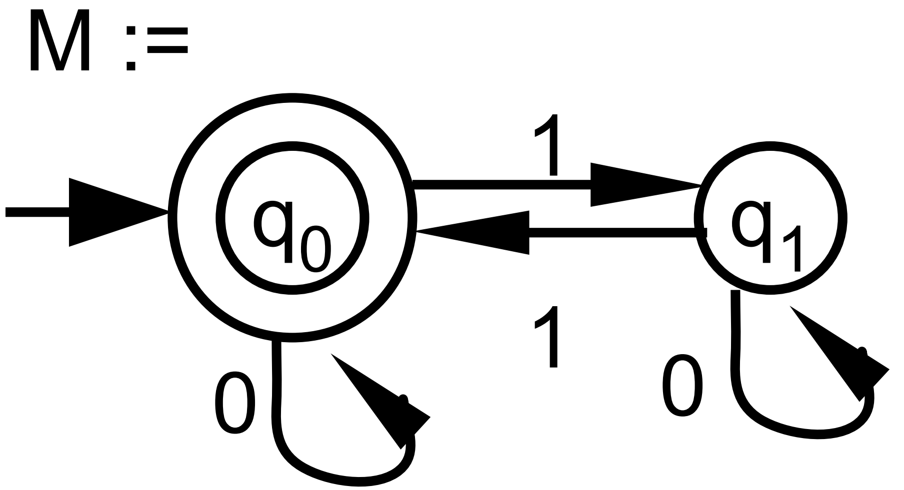
L(M) =

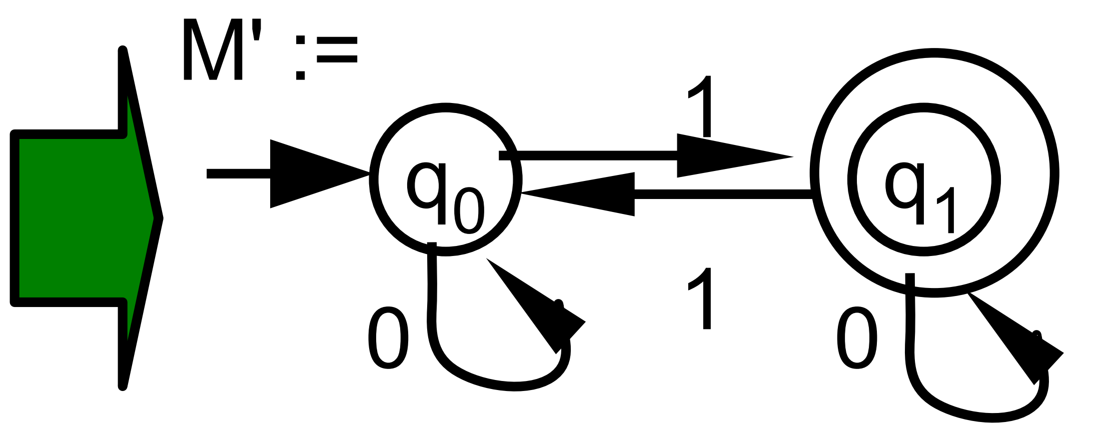
{ w : w has even number of 1}

- Theorem:

  If A is a regular language, then so is (not A)

- Proof idea: Complement the set of accept states
- Example:





L(M) =

{ w : w has even number of 1}

L(M') = not L(M) =

{ w : w has odd number of 1}

- Theorem: If A is a regular language, then so is (not A)
- Proof:

Given DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = A$.

Define DFA M' = ??????????????????????????

This definition is the creative step of this proof,

the rest is (perhaps complicated but) mechanical

“unwrapping definitions”

- Theorem: If A is a regular language, then so is (not A)
- Proof:

  Given DFA M = $(Q, \Sigma, \delta, q_0, F)$ such that L(M) = A.

  Define DFA M' = $(Q, \Sigma, \delta, q_0, F')$, where F' := not F.

- We need to show L(M') = not L(M), that is:

  for any w, ??????????????????????????

- Theorem: If A is a regular language, then so is (not A)
- Proof:

Given DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = A$.

Define DFA $M' = (Q, \Sigma, \delta, q_0, F')$, where $F' :=$ not F.

- We need to show $L(M') =$ not $L(M)$, that is:

for any w, M' accepts w $\Leftrightarrow$ M does not accept w.

- So let w be any string of length k, and consider the k+1 states $r_0, r_1, .., r_k$ from the definition of accept:

(1) $r_0 = q_0$, and

(2) $r_{i+1} = \delta(r_i, w_{i+1}) \ \forall\ 0 \le i < k$.

How do we conclude?

- Theorem: If A is a regular language, then so is (not A)
- Proof:

Given DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = A$.

Define DFA $M' = (Q, \Sigma, \delta, q_0, F')$, where $F' := \text{not } F$.

- We need to show $L(M') = \text{not } L(M)$, that is:

for any w, M' accepts w $\Leftrightarrow$ M does not accept w

- So let w be any string of length k, and consider the k+1 states $r_0, r_1, .., r_k$ from the definition of accept:

  (1) $r_0 = q_0$, and

  (2) $r_{i+1} = \delta(r_i, w_{i+1}) \ \forall\ 0 \leq i < k$.

Note that r_k in F' $\Leftrightarrow$ r_k not in F, since F' = not F. ■

What is a proof?

- A proof is an explanation, written in English, of why something is true.
- Every sentence must be logically connected to the previous ones, often by “so”, “hence”, “since”, etc.
- Your audience is a human being, NOT a machine.

- Theorem: If A is a regular language, then so is (not A)
- Proof:

DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = A$

DFA $M' = (Q, \Sigma, \delta, q_0, F')$, where $F' := \text{not } F$.

$L(M') = \text{not } L(M)$

M' accepts w $\Leftrightarrow$ M does not accept w

k+1 states $r_0, r_1, \ldots, r_k$

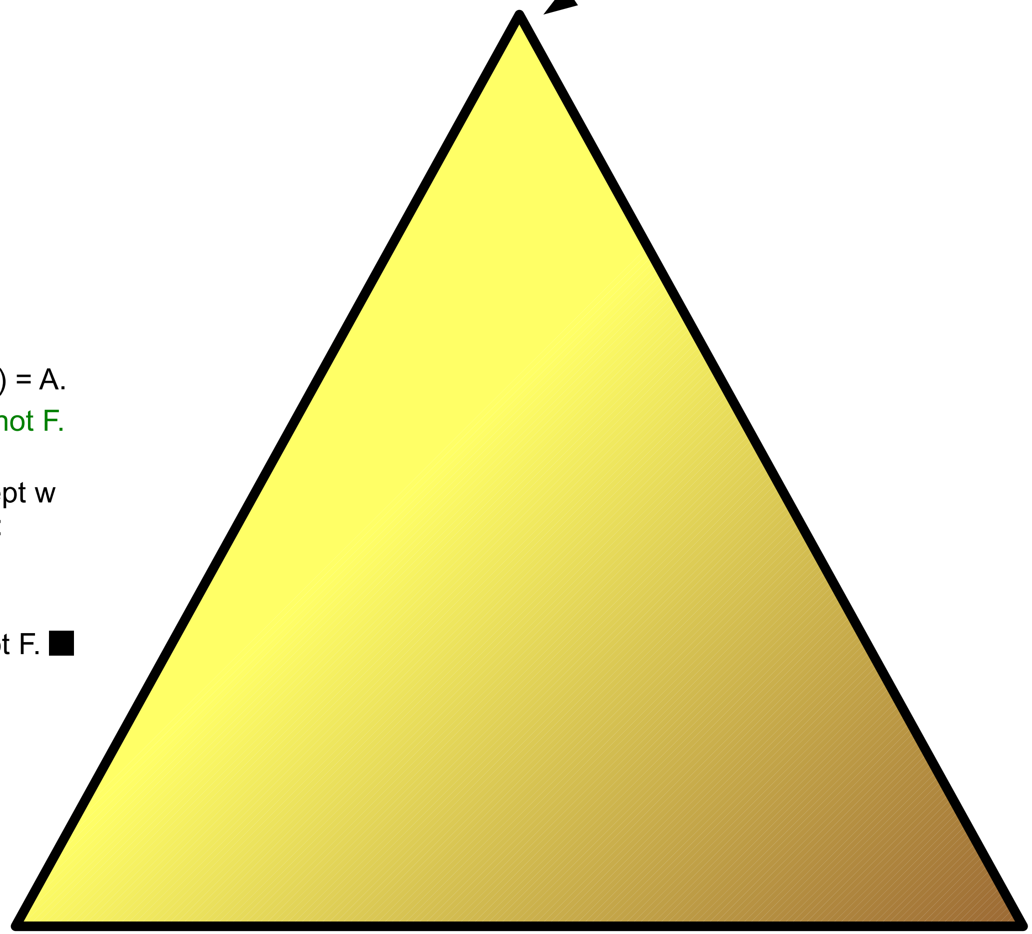
(1) $r_0 = q_0$, and

(2) $r_{i+1} = \delta(r_i, w_{i+1}) \ \forall\ 0 \le i < k$.

r_k in F' $\Leftrightarrow$ r_k not in F, F' = not F. ■

What is a proof?

Complement the set of accept states



Given DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = A$.

Define DFA $M' = (Q, \Sigma, \delta, q_0, F')$, where $F' :=$ not F.

- We need to show $L(M') =$ not $L(M)$, that is:
  for any w, M' accepts w $\Leftrightarrow$ M does not accept w
- Consider the k+1 states $r_0, r_1, .., r_k$ such that:

  (1) $r_0 = q_0$, and

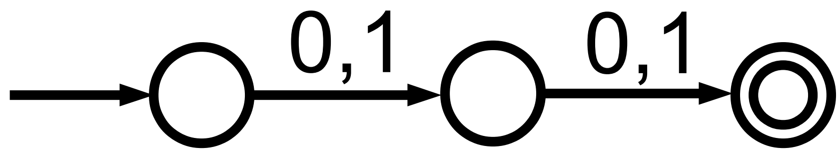
  (2) $r_{i+1} = \delta(r_i, w_{i+1})\ \forall\ 0 \le i < k$.

Note that r_k in F' $\Leftrightarrow$ r_k not in F, since $F' =$ not F. ■

To know a proof means to know all the pyramid

Example $\Sigma = \{0,1\}$

M =



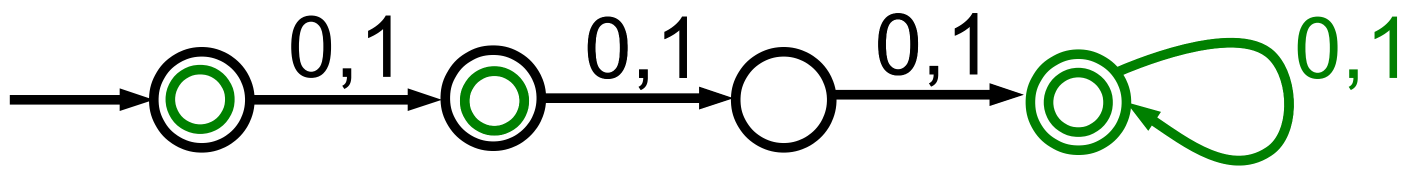
$L(M) = \Sigma^2 = \{00,01,10,11\}$

What is a DFA M' :

$L(M') = \text{not } \Sigma^2$ = all strings except those of length 2 ?

Example $\sum = \{0,1\}$

M' =



$L(M') = \text{not } \sum^2 = \{0,1\}^* - \{00,01,10,11\}$

Do not forget the convention about the sink state!

- Suppose A, B are regular languages, what about
- not A := { w : w is not in A } REGULAR
- A U B := { w : w in A or w in B }
- A o B := { w_1 w_2 : w_1 in A and w_2 in B }
- A* := { w_1 w_2 … w_k : $k \geq 0$, w_i in A for every i }

- Theorem: If A, B are regular, then so is A U B

- Proof idea: Take Cartesian product of states

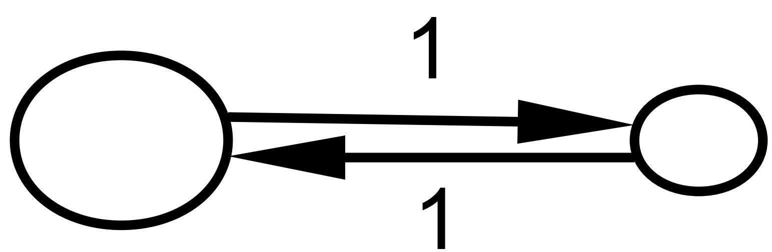
  In a pair (q,q'),

  q tracks DFA for A,

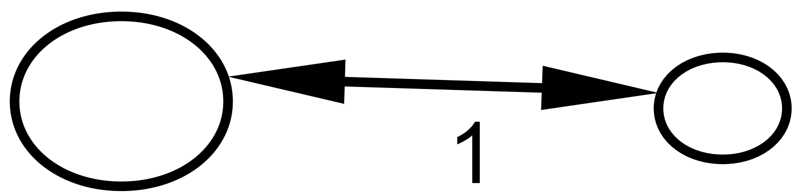
  q' tracks DFA for B.

- Next we see an example.

  In it we abbreviate

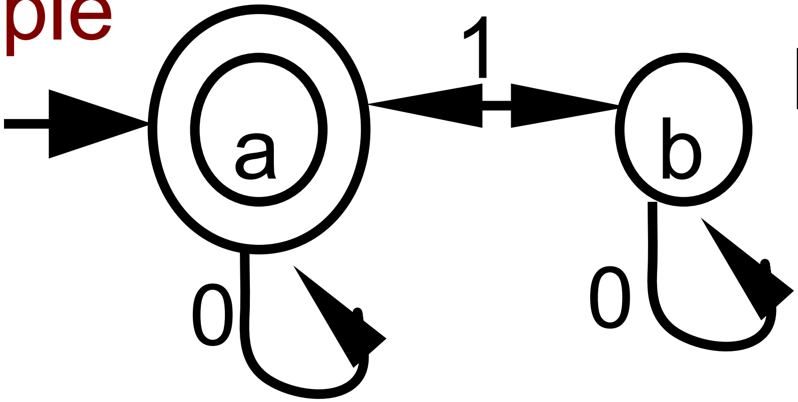


with

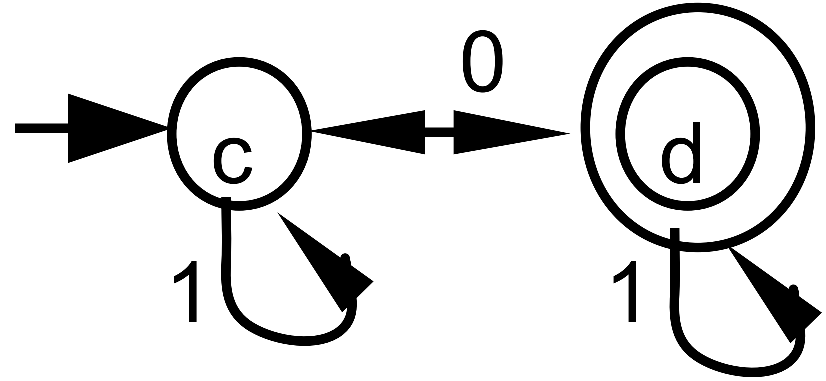


Example

$M_A :=$



$M_B :=$

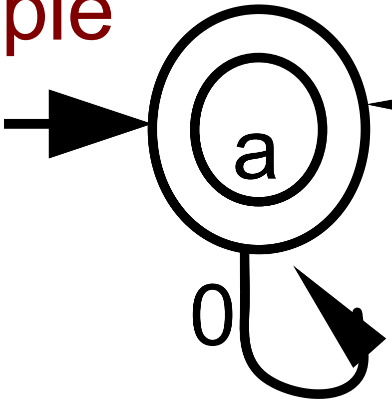


$L(M_A) = A = ?$

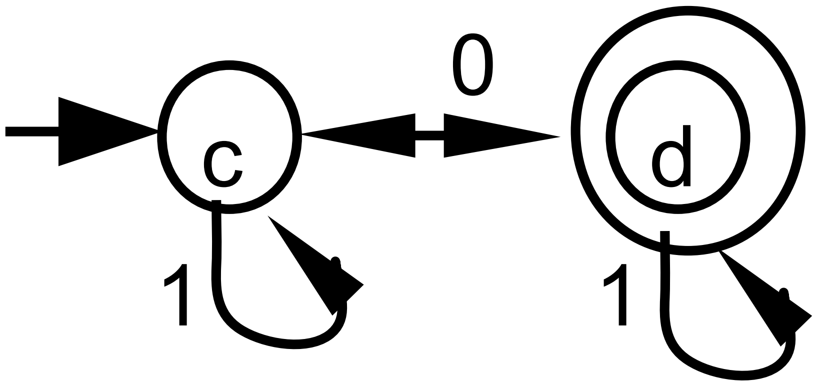
$L(M_B) = B = ?$

Example

$M_A :=$



$M_B :=$



$L(M_A) = A =$

{ w : w has even number of 1}

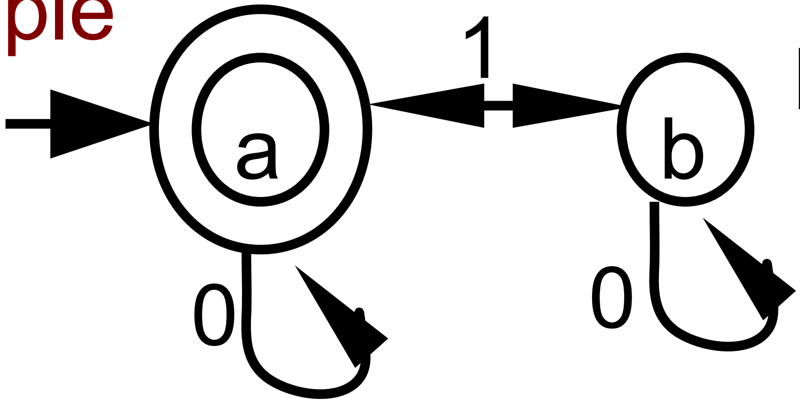
$L(M_B) = B =$

{ w : w has odd number of 0}

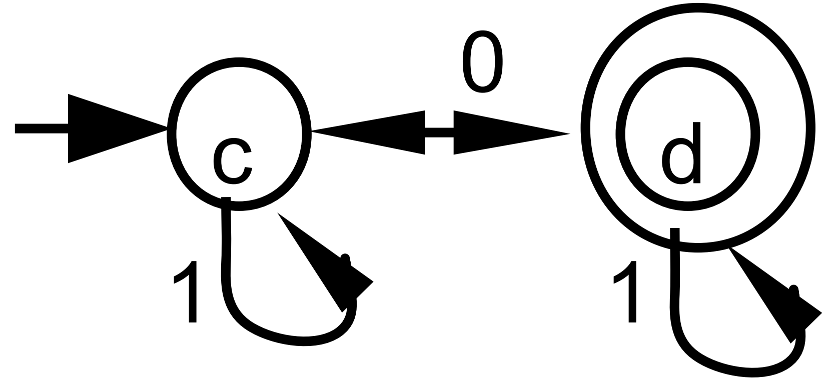
$M_{A \cup B} :=$ How many states?

Example

$M_A :=$



$M_B :=$



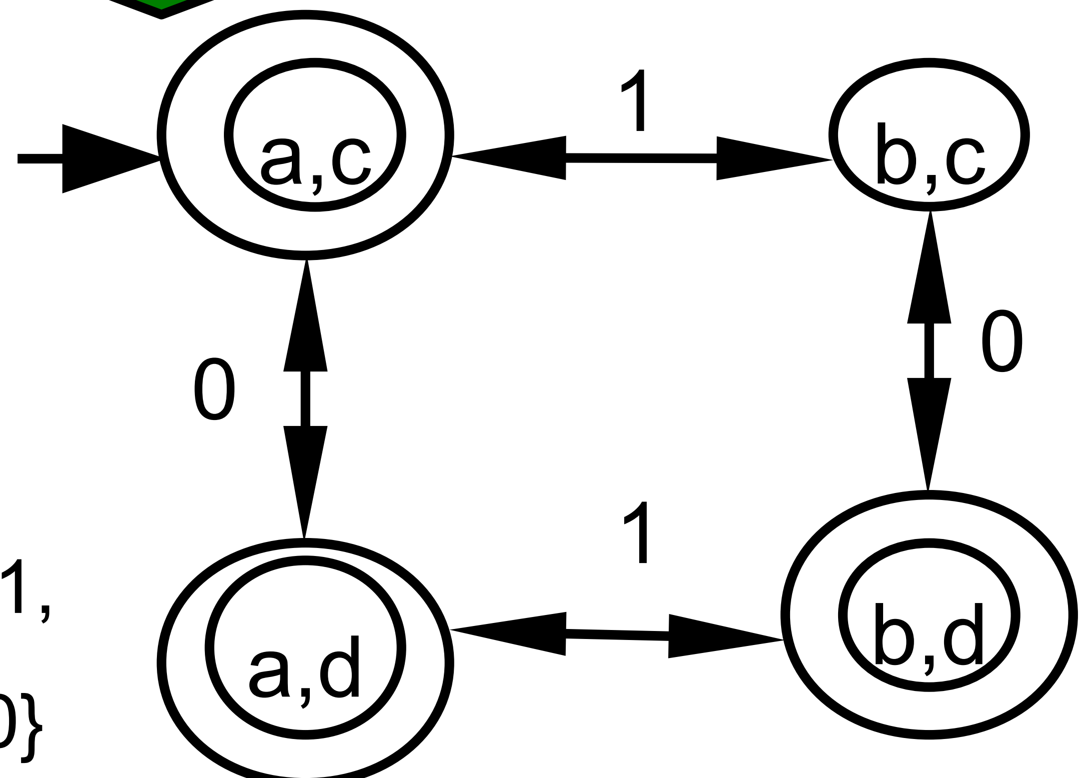
$L(M_A) = A =$

{ w : w has even number of 1}

$L(M_B) = B =$

{ w : w has odd number of 0}

$M_{AUB} :=$



$L(M_{AUB}) = AUB =$

{ w : w has even number of 1,

or odd number of 0}

- Theorem: If A, B are regular, then so is A U B
- Proof:

Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ such that L(M) = A,

DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ such that L(M) = B.

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

$Q :=$?

- Theorem: If A, B are regular, then so is A U B
- Proof:

Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ such that $L(M) = A$,

DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ such that $L(M) = B$.

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

$Q := Q_A \times Q_B$

$q_0 :=$?

- Theorem: If A, B are regular, then so is A U B
- Proof:

Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ such that L(M) = A,

DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ such that L(M) = B.

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

$Q := Q_A \text{ X } Q_B$

$q_0 := (q_A, q_B)$

$F :=$?

- **Theorem**: If A, B are regular, then so is A U B
- **Proof**:

Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ such that $L(M) = A$,

DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ such that $L(M) = B$.

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

$Q := Q_A \times Q_B$

$q_0 := (q_A, q_B)$

$F := \{(q,q') \in Q : q \in F_A \text{ or } q' \in F_B\}$

$\delta((q,q'), v) := (?, ?)$

- Theorem: If A, B are regular, then so is A U B
- Proof:

Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ such that $L(M) = A$,

DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ such that $L(M) = B$.

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

$Q := Q_A \times Q_B$

$q_0 := (q_A, q_B)$

$F := \{(q,q') \in Q : q \in F_A \text{ or } q' \in F_B\}$

$\delta((q,q'), v) := (\delta_A(q,v), \delta_B(q',v))$

- We need to show L(M) = A U B that is, for any w:

  M accepts w $\Leftrightarrow$ M_A accepts w or M_B accepts w

- Proof M accepts w $\rightarrow M_A$ accepts w or M_B accepts w
- Suppose that M accepts w of length k.
- From the definitions of accept and M, the sequence $(s_0, t_0) = q_0 = (q_A, q_B)$, $(s_{i+1},t_{i+1})=\delta((s_i,t_i), w_{i+1})=(\delta_A(s_i,w_{i+1}), \delta_B(t_i,w_{i+1}))\ \forall 0\leq i< k$ has $(s_k,t_k)\in$?

- Proof M accepts w $\rightarrow M_A$ accepts w or M_B accepts w
- Suppose that M accepts w of length k.
- From the definitions of accept and M, the sequence $(s_0, t_0) = q_0 = (q_A, q_B)$, $(s_{i+1},t_{i+1})=\delta((s_i,t_i), w_{i+1})=(\delta_A(s_i,w_{i+1}), \delta_B(t_i,w_{i+1})) \; \forall 0\leq i< k$ has $(s_k,t_k)\in F$.
- By our definition of F, what can we say about (s_k,t_k) ?

- Proof M accepts w $\rightarrow M_A$ accepts w or M_B accepts w
- Suppose that M accepts w of length k.
- From the definitions of accept and M, the sequence $(s_0, t_0) = q_0 = (q_A, q_B)$, $(s_{i+1}, t_{i+1}) = \delta((s_i, t_i), w_{i+1}) = (\delta_A(s_i, w_{i+1}), \delta_B(t_i, w_{i+1}))\ \forall 0 \le i < k$ has $(s_k, t_k) \in F$.
- By our definition of F, $s_k \in F_A$ or $t_k \in F_B$.
- Without loss of generality, assume $s_k \in F_A$.
- Then M_A accepts w because the sequence $s_0 = q_A$, $s_{i+1} = \delta_A(s_i, w_{i+1})\ \forall 0 \le i < k$, has $s_k \in F_A$.

- Proof M accepts w ← M_A accepts w or M_B accepts w
- W/out loss of generality, assume M_A accepts w, |w| =k.
- From the definition of M_A accepts w, the sequence $r_0 := q_A$, $r_{i+1} := \delta_A(r_i, w_{i+1})$ $\forall\ 0 \le i < k$, has r_k in ?

- Proof M accepts w $\leftarrow M_A$ accepts w or M_B accepts w
- W/out loss of generality, assume M_A accepts w, |w| =k.
- From the definition of M_A accepts w, the sequence $r_0 := q_A$, $r_{i+1} := \delta_A(r_i, w_{i+1})$ $\forall\, 0 \le i < k$, has r_k in F_A .
- Define the sequence of k+1 states

  $t_0 := q_B$, $t_{i+1} := \delta_B(t_i, w_{i+1})$ $\forall\, 0 \le i < k$.
- M accepts w because the sequence

  ?????????? (recall states in M are pairs)

- Proof M accepts w $\leftarrow M_A$ accepts w or M_B accepts w
- W/out loss of generality, assume M_A accepts w, $|w| = k$.
- From the definition of M_A accepts w, the sequence $r_0 := q_A$, $r_{i+1} := \delta_A(r_i, w_{i+1})\ \forall\ 0 \le i < k$, has r_k in F_A.
- Define the sequence of k+1 states

  $t_0 := q_B$, $t_{i+1} := \delta_B(t_i, w_{i+1})\ \forall\ 0 \le i < k$.
- M accepts w because the sequence $(r_0, t_0) = q = (q_A, q_B)$, $(r_{i+1}, t_{i+1}) = \delta((r_i, t_i), w_{i+1}) = (\delta_A(r_i, w_{i+1}), \delta_B(t_i, w_{i+1})\ \forall 0 \le i < k$ has (r_k, t_k) in F, by our definition of F. ■

- Suppose A, B are regular languages, what about
- not A := { w : w is not in A }   REGULAR
- A U B := { w : w in A or w in B }   REGULAR
- A o B := { w_1 w_2 : w_1 in A and w_2 in B }
- A^* := { w_1 w_2 … w_k : $k \geq 0$, w_i in A for every i }

- Other two are more complicated!

- Plan: we introduce NFA

  prove that NFA are equivalent to DFA

  reprove A U B, prove A o B, A^* regular, using NFA

Big picture

- All languages
- Decidable

  Turing machines
- NP
- P
- Context-free

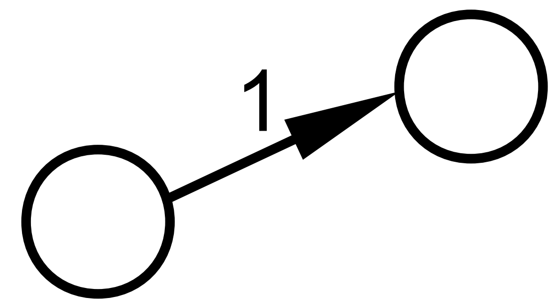
  Context-free grammars, push-down automata
- Regular

  Automata, non-deterministic automata,

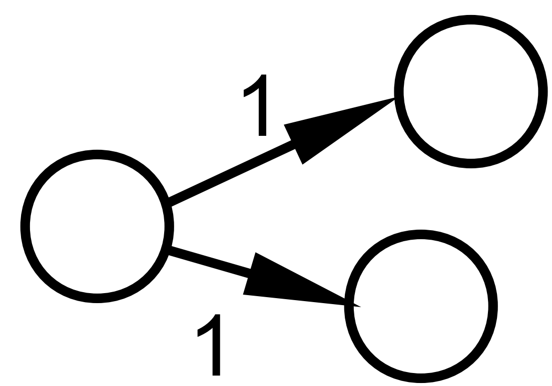
  regular expressions

Non deterministic finite automata (NFA)

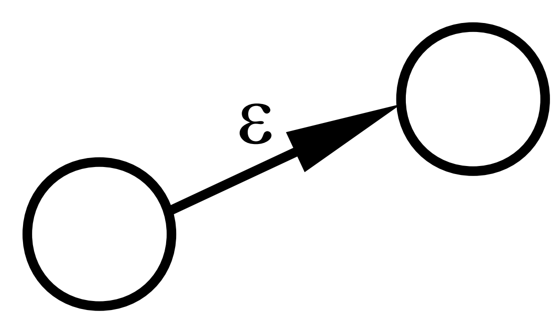
- DFA: given state and input symbol, unique choice for next state, deterministic:



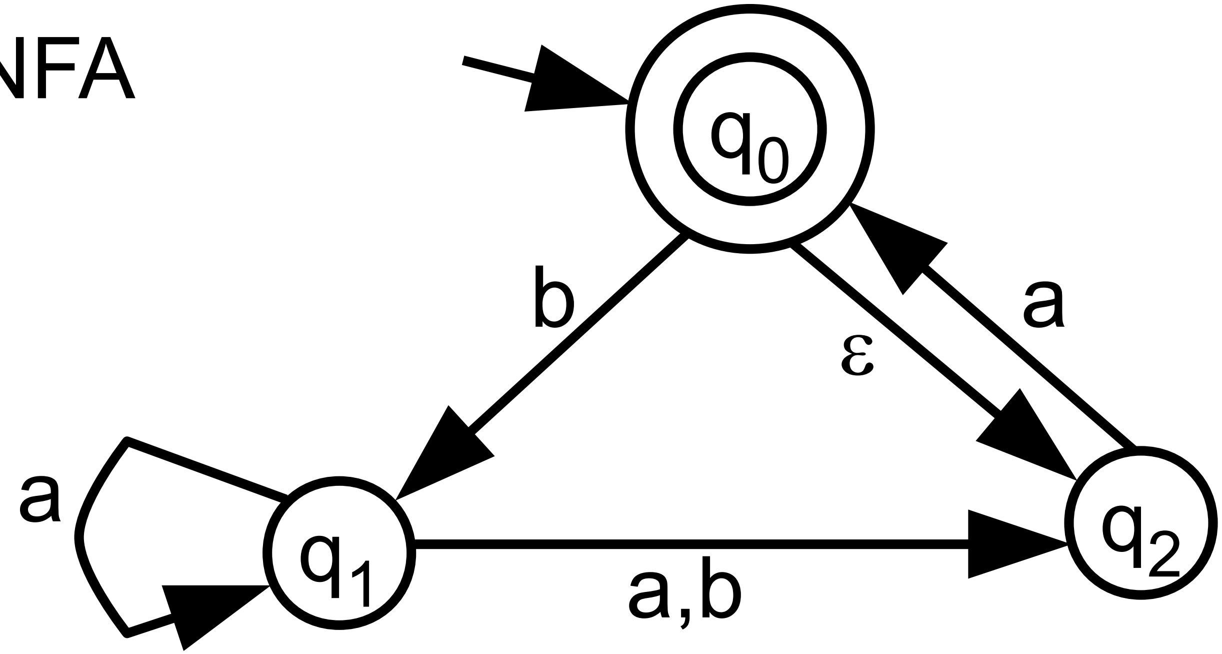
- Next we allow multiple choices, non-deterministic



- We also allow ε-transitions: can follow without reading anything



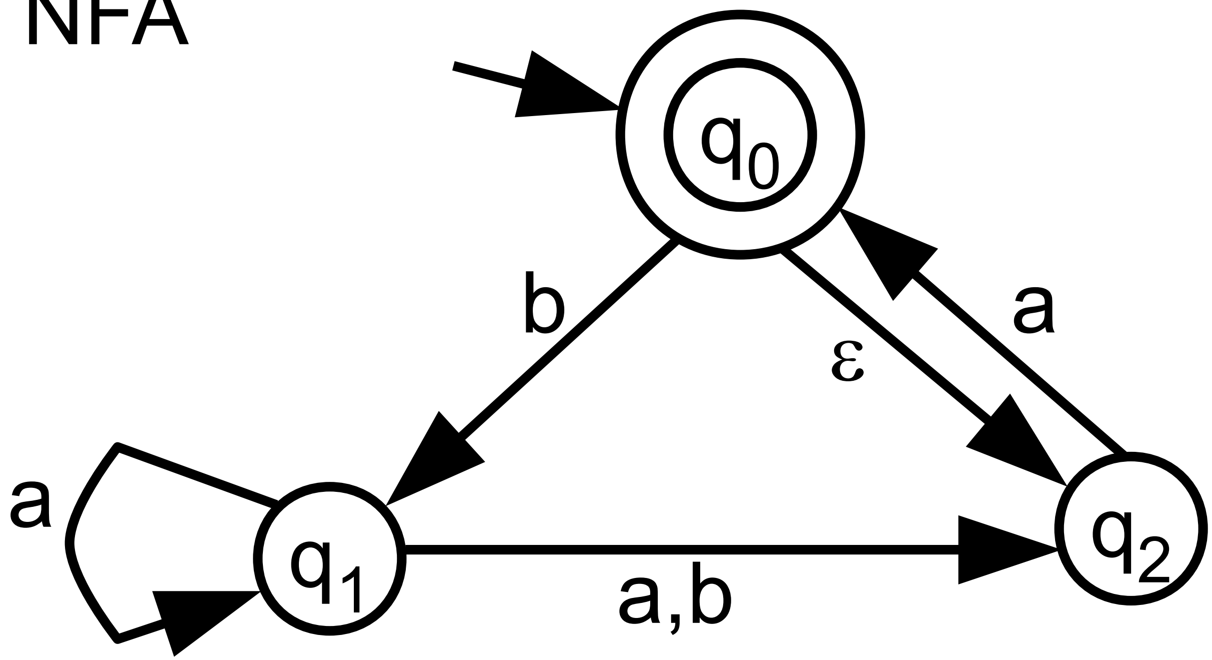
Example of NFA



Intuition of how it computes:

- Accept string w if there is a way to follow transitions that ends in accept state
- Transitions labelled with symbol in $\Sigma = \{a,b\}$ must be matched with input
- ε transitions can be followed without matching

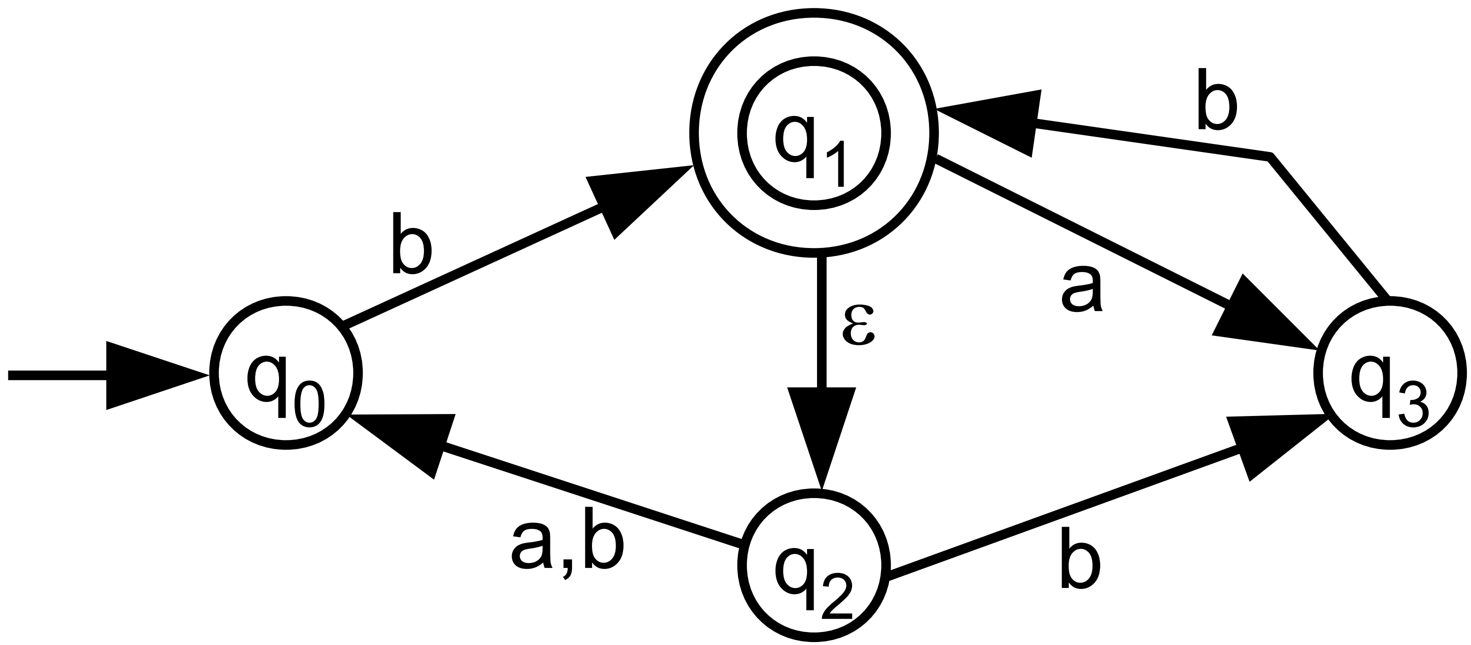
Example of NFA



Example:

- Accept a (first follow ε-transition)
- Accept baaa

ANOTHER Example of NFA



Example:

- Accept bab (two accepting paths, one uses the ε-transition)
- Reject ba (two possible paths, but neither has final state = q_1)

- Definition: A non-deterministic finite automaton (NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states
- Σ is the input alphabet
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \text{Powerset}(Q)$
- q_0 in Q is the start state
- $F \subseteq Q$ is the set of accept states

- Recall: Powerset(Q) = set of all subsets of Q

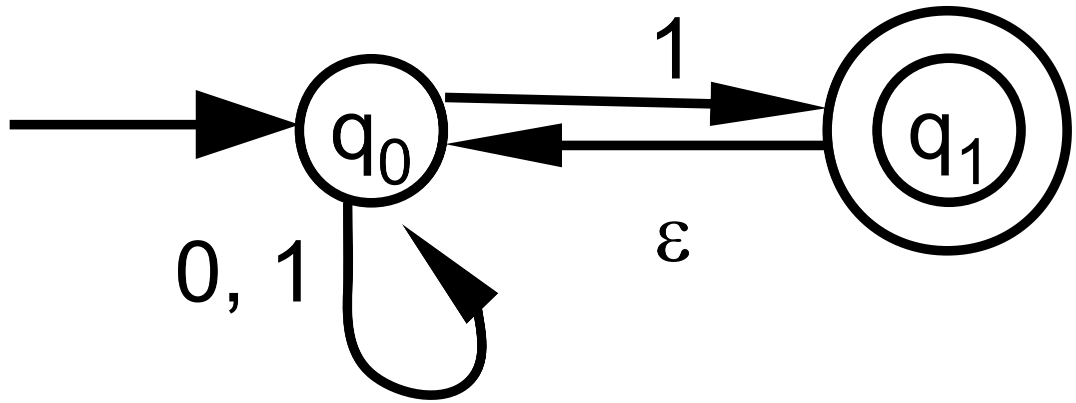
  Example: Powerset({1,2}) = ?

- Definition: A non-deterministic finite automaton (NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

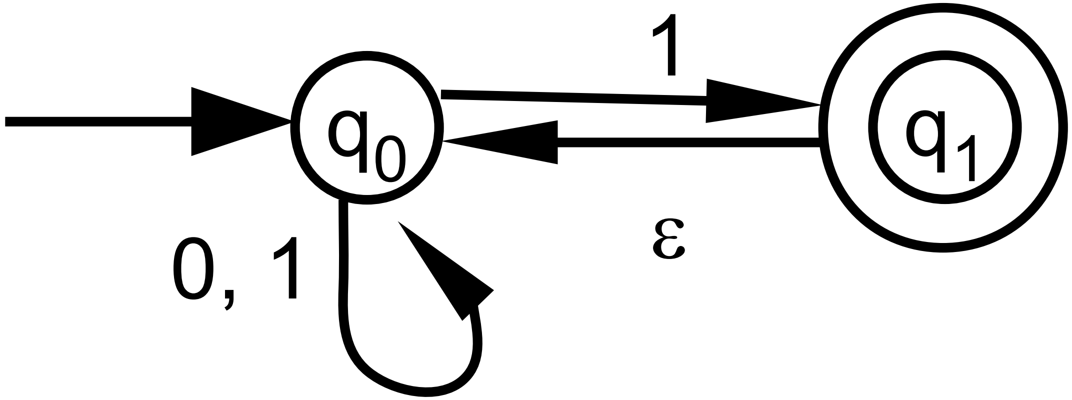
- Q is a finite set of states
- Σ is the input alphabet
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \text{Powerset}(Q)$
- q_0 in Q is the start state
- $F \subseteq Q$ is the set of accept states

- Recall: Powerset(Q) = set of all subsets of Q

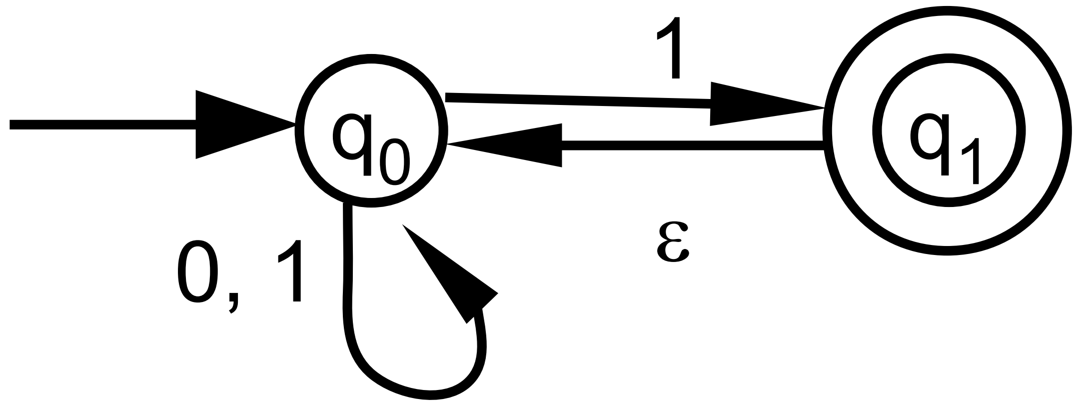
  Example: $\text{Powerset}(\{1,2\}) = \{\emptyset, \{1\}, \{2\}, \{1,2\}\}$



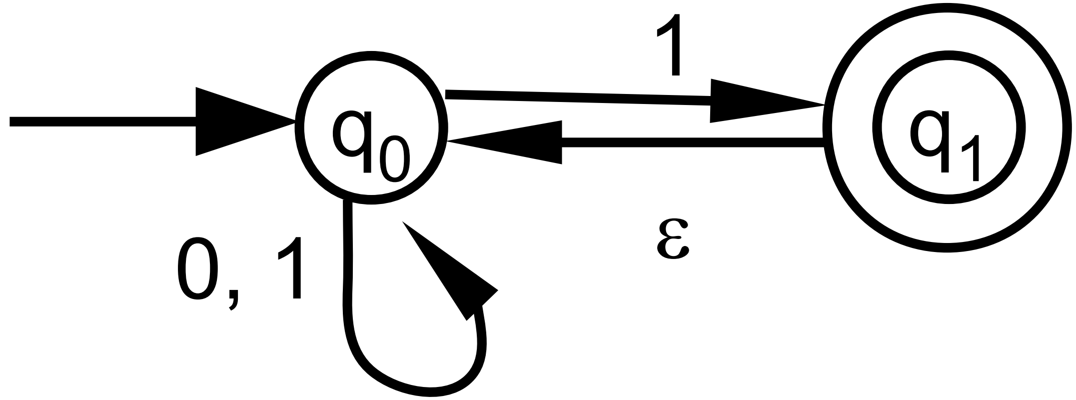
- Example: above NFA is 5-tuple $(Q, \Sigma, \delta, q_0, F)$
- $Q = \{ q_0, q_1\}$
- $\Sigma = \{0,1\}$
- $\delta(q_0 ,0) =$?



- Example: above NFA is 5-tuple $(Q, \Sigma, \delta, q_0, F)$
- $Q = \{ q_0, q_1\}$
- $\Sigma = \{0,1\}$
- $\delta(q_0, 0) = \{q_0\}$ $\delta(q_0, 1) =$?

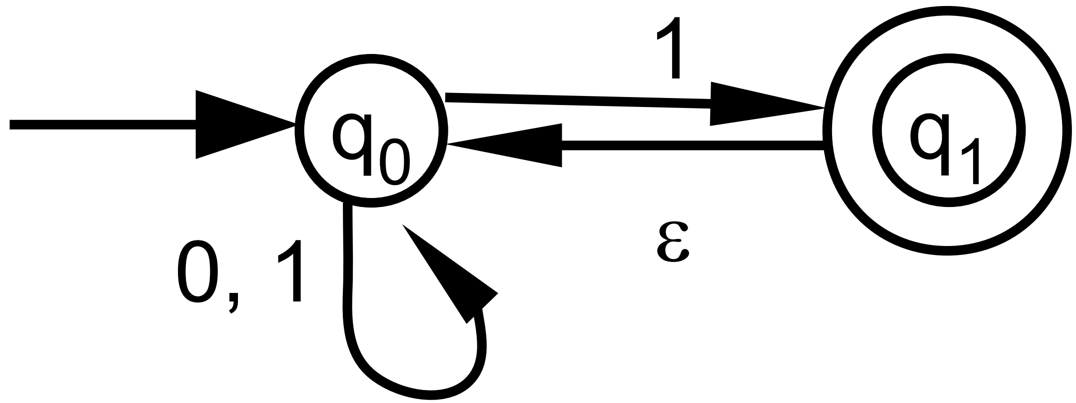


- Example: above NFA is 5-tuple $(Q, \Sigma, \delta, q_0, F)$
- $Q = \{ q_0, q_1 \}$
- $\Sigma = \{0,1\}$
- $\delta(q_0, 0) = \{q_0\}$ $\delta(q_0, 1) = \{q_0, q_1\}$ $\delta(q_0, \varepsilon) =$?



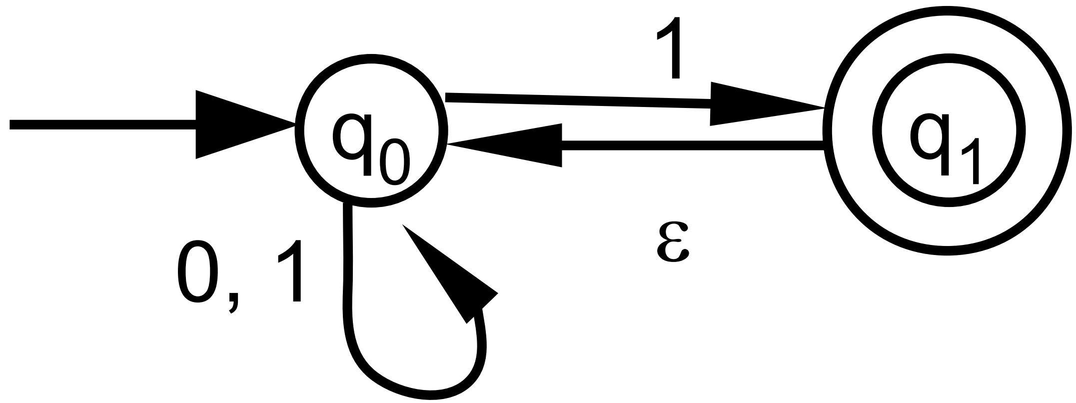
- Example: above NFA is 5-tuple $(Q, \Sigma, \delta, q_0, F)$
- $Q = \{ q_0, q_1\}$
- $\Sigma = \{0,1\}$
- $\delta(q_0 ,0) = \{q_0\}$ $\delta(q_0 ,1) = \{q_0, q_1\}$ $\delta(q_0 ,\varepsilon) = \emptyset$

  $\delta(q_1 ,0) =$?



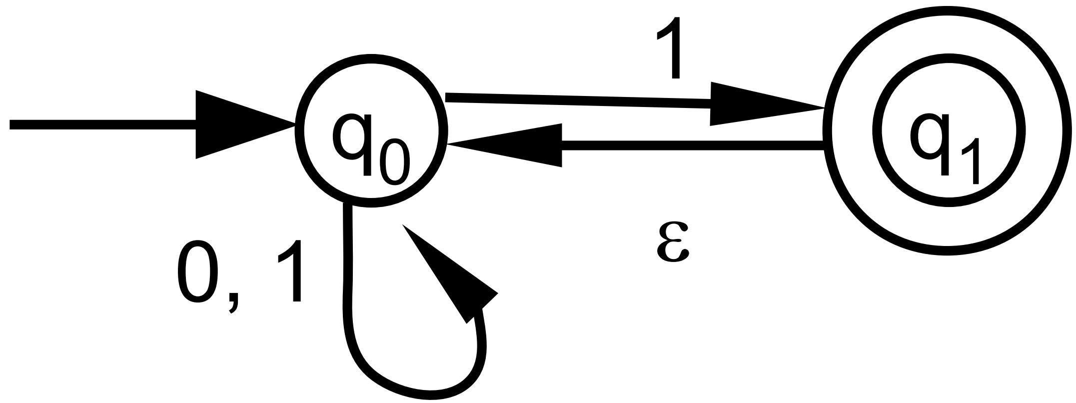
- Example: above NFA is 5-tuple $(Q, \Sigma, \delta, q_0, F)$
- $Q = \{ q_0, q_1 \}$
- $\Sigma = \{0,1\}$
- $\delta(q_0, 0) = \{q_0\}$ $\delta(q_0, 1) = \{q_0, q_1\}$ $\delta(q_0, \varepsilon) = \varnothing$

  $\delta(q_1, 0) = \varnothing$ $\delta(q_1, 1) = ?$



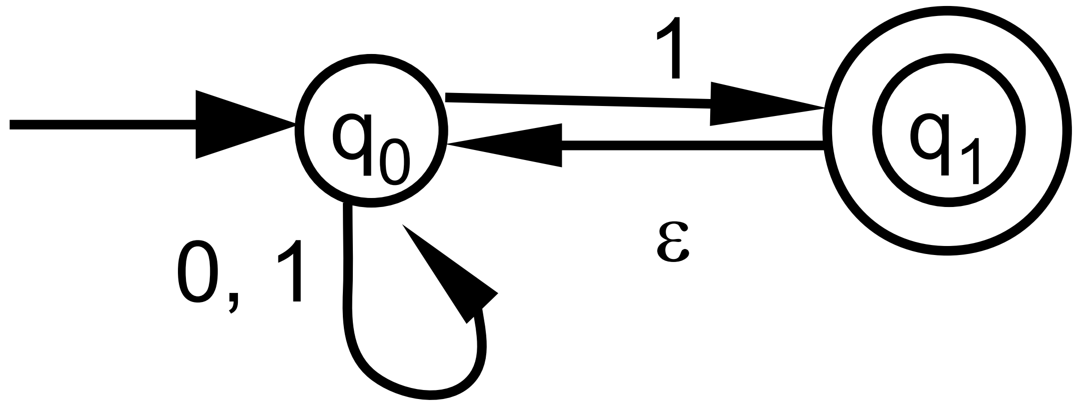
- Example: above NFA is 5-tuple $(Q, \Sigma, \delta, q_0, F)$
- $Q = \{ q_0, q_1 \}$
- $\Sigma = \{0,1\}$
- $\delta(q_0, 0) = \{q_0\}$ $\delta(q_0, 1) = \{q_0, q_1\}$ $\delta(q_0, \varepsilon) = \emptyset$

  $\delta(q_1, 0) = \emptyset$ $\delta(q_1, 1) = \emptyset$ $\delta(q_1, \varepsilon) =$?



- Example: above NFA is 5-tuple $(Q, \Sigma, \delta, q_0, F)$
- $Q = \{ q_0, q_1\}$
- $\Sigma = \{0,1\}$
- $\delta(q_0, 0) = \{q_0\}$ $\delta(q_0, 1) = \{q_0, q_1\}$ $\delta(q_0, \varepsilon) = \emptyset$

  $\delta(q_1, 0) = \emptyset$ $\delta(q_1, 1) = \emptyset$ $\delta(q_1, \varepsilon) = \{q_0\}$
- q_0 in Q is the start state
- F = ?



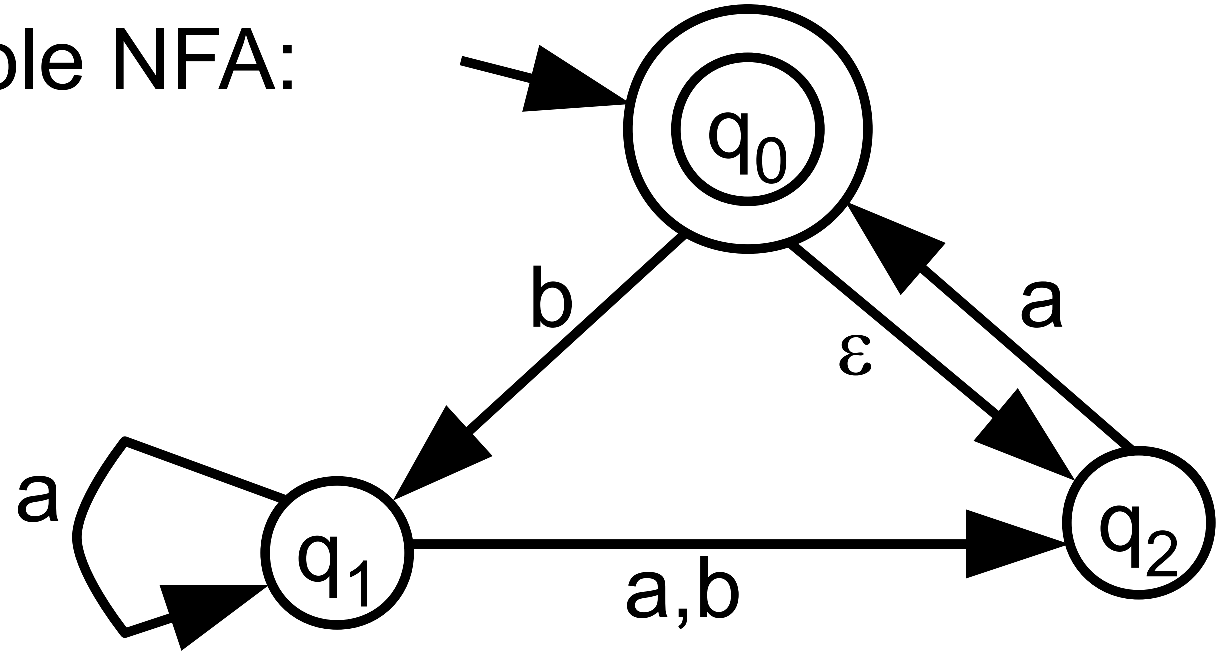
- Example: above NFA is 5-tuple $(Q, \Sigma, \delta, q_0, F)$
- $Q = \{ q_0, q_1\}$
- $\Sigma = \{0,1\}$
- $\delta(q_0, 0) = \{q_0\}$ $\delta(q_0, 1) = \{q_0, q_1\}$ $\delta(q_0, \varepsilon) = \varnothing$

  $\delta(q_1, 0) = \varnothing$ $\delta(q_1, 1) = \varnothing$ $\delta(q_1, \varepsilon) = \{q_0\}$
- q_0 in Q is the start state
- $F = \{ q_1\} \subseteq Q$ is the set of accept states

- Definition: A NFA $(Q, \Sigma, \delta, q_0, F)$ accepts a string w if
  $\exists$ integer k, $\exists$ k strings w_1 , w_2 , …, w_k such that
- $w = w_1 w_2 \ldots w_k$ where $\forall\ 1 \leq i \leq k$, $w_i \in \Sigma \cup \{\varepsilon\}$
  (the symbols of w, or ε)
- $\exists$ sequence of k+1 states $r_0, r_1, .., r_k$ in Q such that:
- $r_0 = q_0$
- $r_{i+1} \in \delta(r_i, w_{i+1})\ \forall\ 0 \leq i < k$
- r_k is in F

- Differences with DFA are in green

Back to first example NFA:



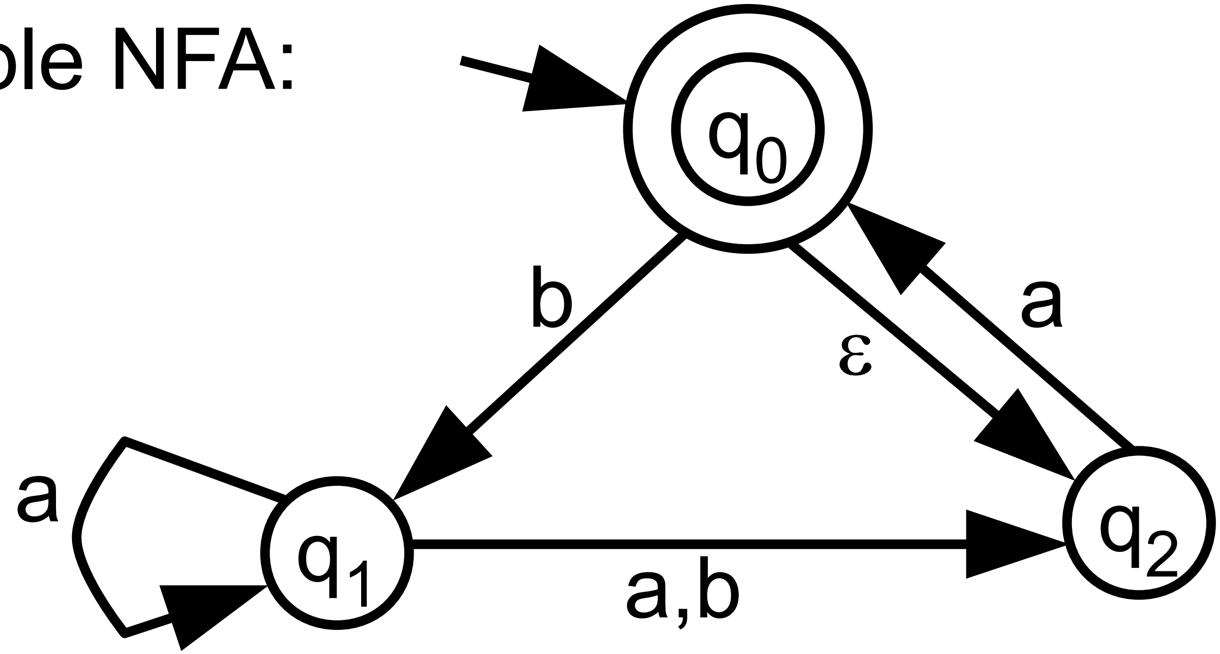
Accepts w = baaa

$w_1 = b$, $w_2 = a$, $w_3 = a$, $w_4 = \varepsilon$, $w_5 = a$

Accepting sequence of 5+1 = 6 states:

$r_0 = ?$

Back to first example NFA:



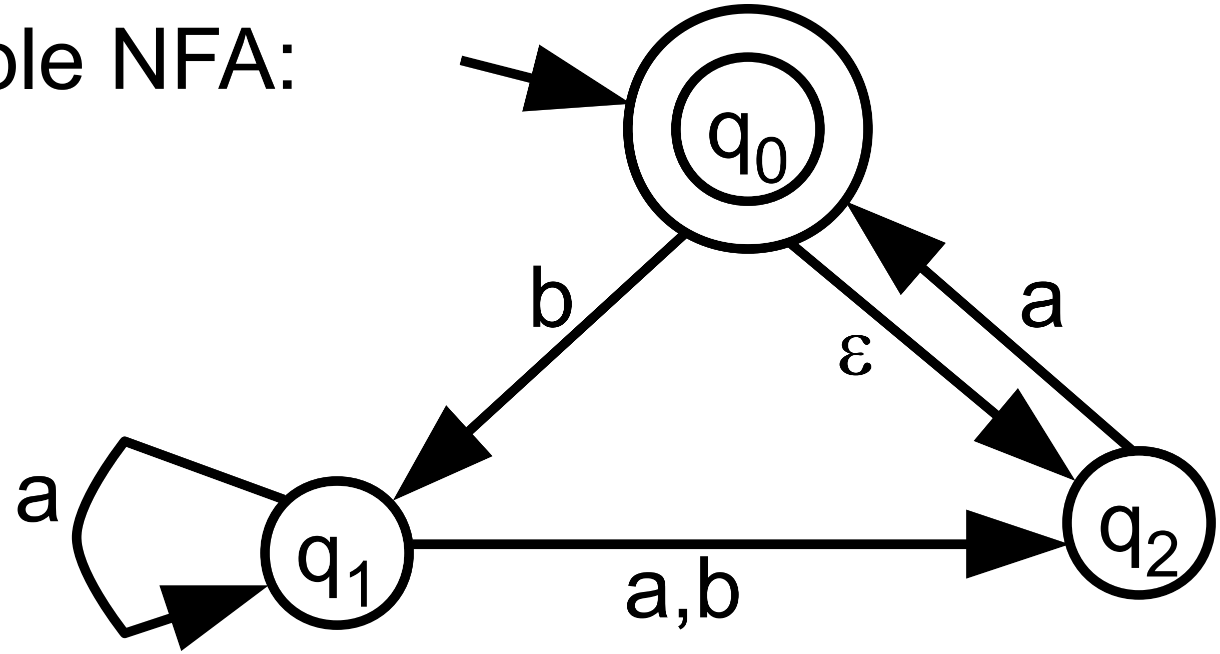
Accepts w = baaa

$w_1 = b$, $w_2 = a$, $w_3 = a$, $w_4 = \varepsilon$, $w_5 = a$

Accepting sequence of 5+1 = 6 states:

$r_0 = q_0$, $r_1 =$?

Back to first example NFA:



Accepts w = baaa

$w_1 = b$, $w_2 = a$, $w_3 = a$, $w_4 = \varepsilon$, $w_5 = a$

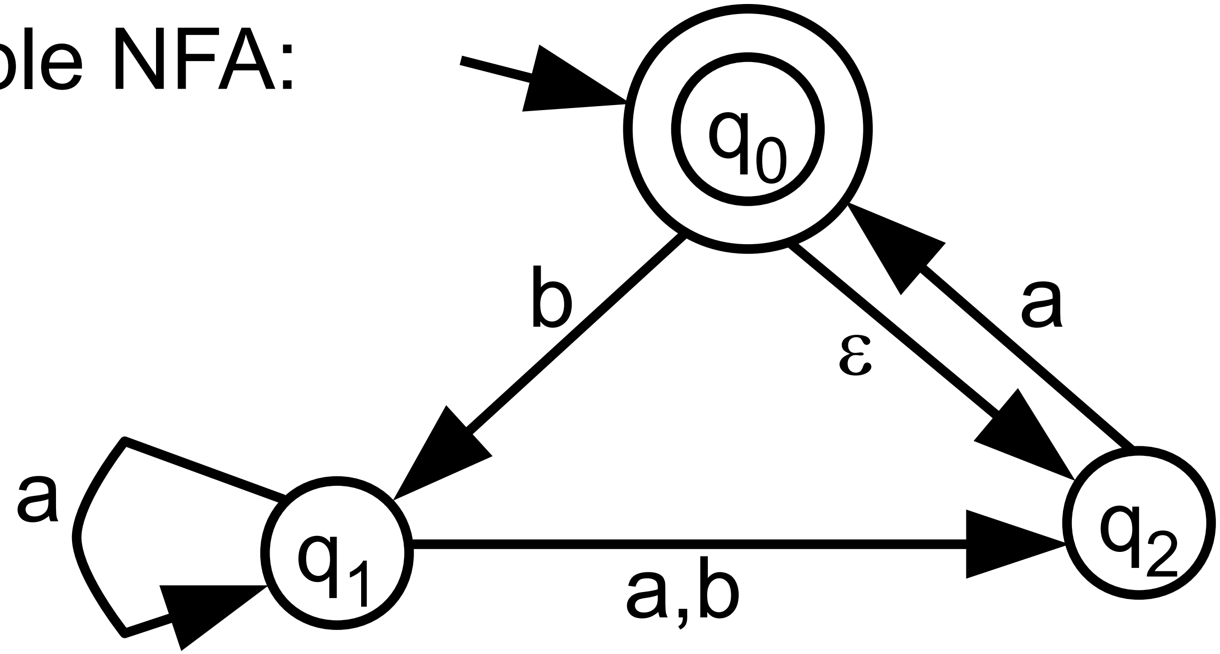
Accepting sequence of 5+1 = 6 states:

$r_0 = q_0$, $r_1 = q_1$, $r_2 =$?

Transitions:

$r_1 \in \delta(r_0, b) = \{q_1\}$

Back to first example NFA:



Accepts w = baaa

$w_1 = b$, $w_2 = a$, $w_3 = a$, $w_4 = \varepsilon$, $w_5 = a$

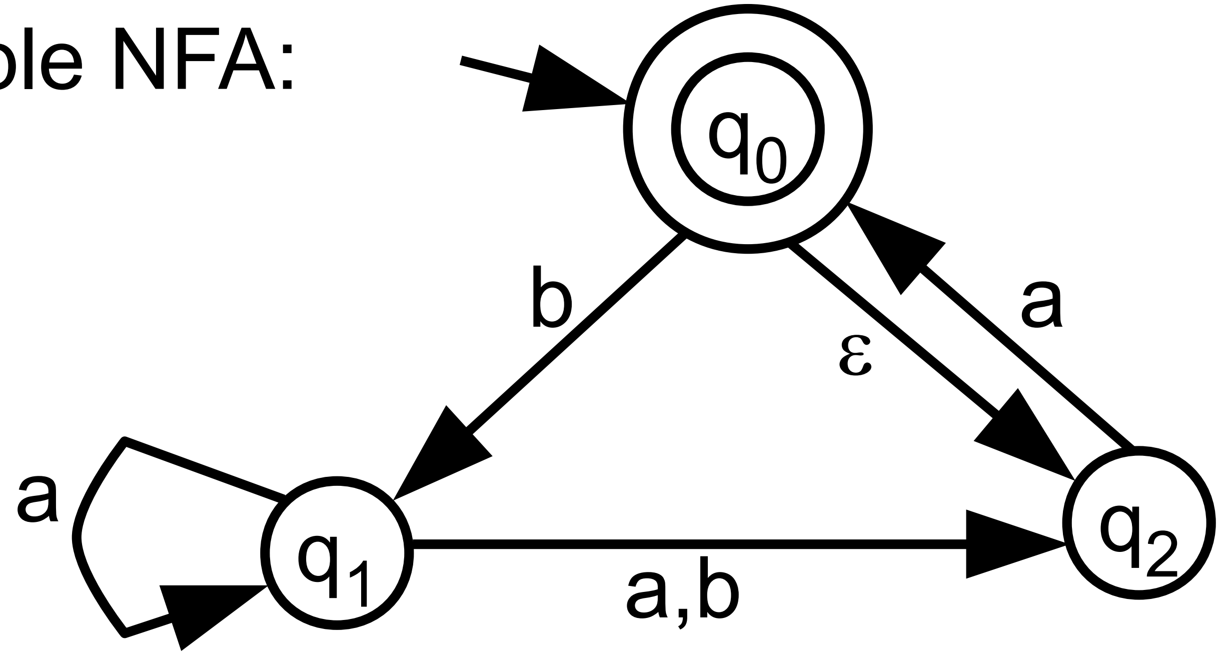
Accepting sequence of 5+1 = 6 states:

$r_0 = q_0$, $r_1 = q_1$, $r_2 = q_2$, $r_3 =$?

Transitions:

$r_1 \in \delta(r_0,b) = \{q_1\}$ $r_2 \in \delta(r_1,a) = \{q_1,q_2\}$

Back to first example NFA:



Accepts w = baaa

$w_1 = b, \quad w_2 = a, \quad w_3 = a, \quad w_4 = \varepsilon, \quad w_5 = a$

Accepting sequence of 5+1 = 6 states:

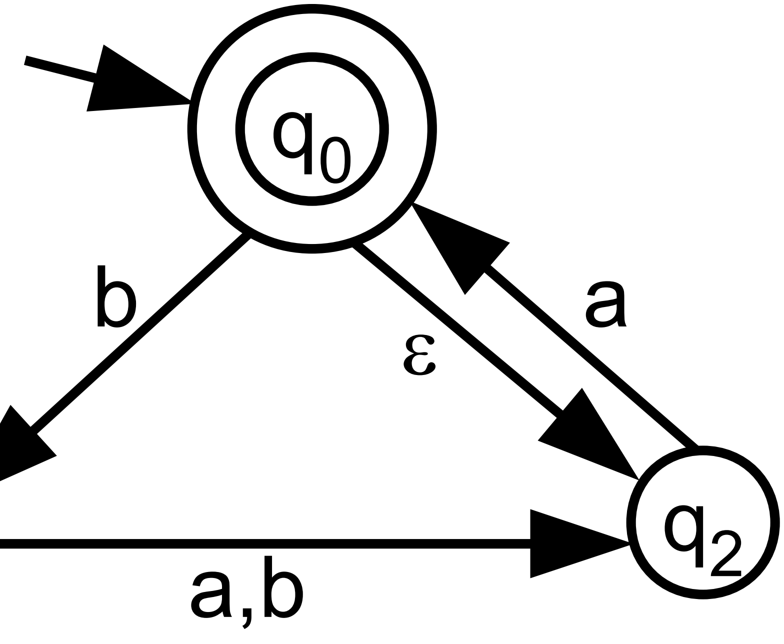
$r_0 = q_0, \quad r_1 = q_1, \quad r_2 = q_2, \quad r_3 = q_0, \quad r_4 =$?

Transitions:

$r_1 \in \delta(r_0,b) = \{q_1\} \quad r_2 \in \delta(r_1,a) = \{q_1,q_2\}$

$r_3 \in \delta(r_2,a) = \{q_0\}$

Back to first example NFA:



Accepts w = baaa

$w_1 = b, \quad w_2 = a, \quad w_3 = a, \quad w_4 = \varepsilon, \quad w_5 = a$

Accepting sequence of 5+1 = 6 states:

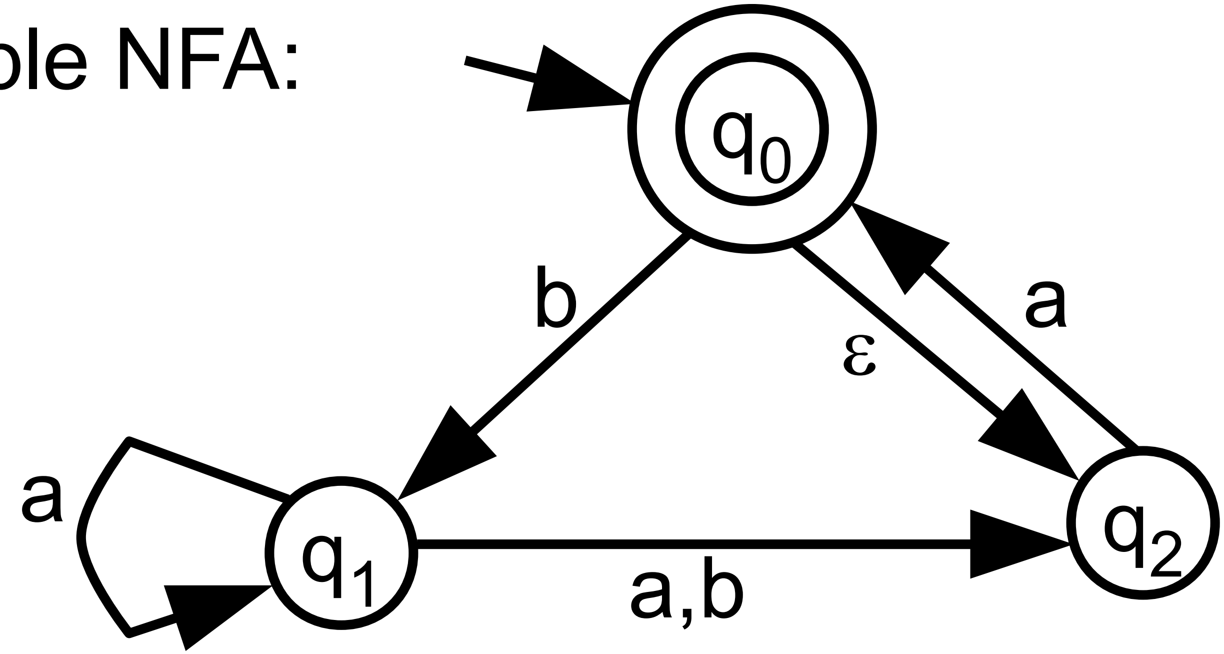
$r_0 = q_0, \quad r_1 = q_1, \quad r_2 = q_2, \quad r_3 = q_0, \quad r_4 = q_2, \quad r_5 =$?

Transitions:

$r_1 \in \delta(r_0,b) = \{q_1\} \quad r_2 \in \delta(r_1,a) = \{q_1,q_2\}$

$r_3 \in \delta(r_2,a) = \{q_0\} \quad r_4 \in \delta(r_3,\varepsilon) = \{q_2\}$

Back to first example NFA:



Accepts w = baaa

$w_1 = b, \quad w_2 = a, \quad w_3 = a, \quad w_4 = \varepsilon, \quad w_5 = a$

Accepting sequence of 5+1 = 6 states:

$r_0 = q_0, \quad r_1 = q_1, \quad r_2 = q_2, \quad r_3 = q_0, \quad r_4 = q_2, \quad r_5 = q_0$

Transitions:

$r_1 \in \delta(r_0, b) = \{q_1\} \quad r_2 \in \delta(r_1, a) = \{q_1, q_2\}$

$r_3 \in \delta(r_2, a) = \{q_0\} \quad r_4 \in \delta(r_3, \varepsilon) = \{q_2\} \quad r_5 \in \delta(r_4, a) = \{q_0\}$

- NFA are at least as powerful as DFA, because DFA are a special case of NFA

- Are NFA more powerful than DFA?

- Surprisingly, they are not:

- Theorem:

For every NFA N there is DFA M : L(M) = L(N)

- Theorem:

For every NFA N there is DFA M : L(M) = L(N)

- Construction without ε transitions
- Given NFA N $(Q, \Sigma, \delta, q, F)$
- Construct DFA M $(Q', \Sigma, \delta', q', F')$ where:
- $Q' :=$ Powerset(Q)
- $q' = \{q\}$
- $F' = \{ S : S \in Q'$ and S contains an element of F$\}$
- $\delta'(S, a) := \bigcup_{s \in S} \delta(s,a)$

$$= \{ t : t \in \delta(s,a) \text{ for some } s \in S \}$$

- It remains to deal with ε transitions

- Definition: Let S be a set of states.

E(S) := { q : q can be reached from some state s in S traveling along 0 or more ε transitions }

- We think of following ε transitions at beginning, or right after reading an input symbol in Σ

- Theorem:

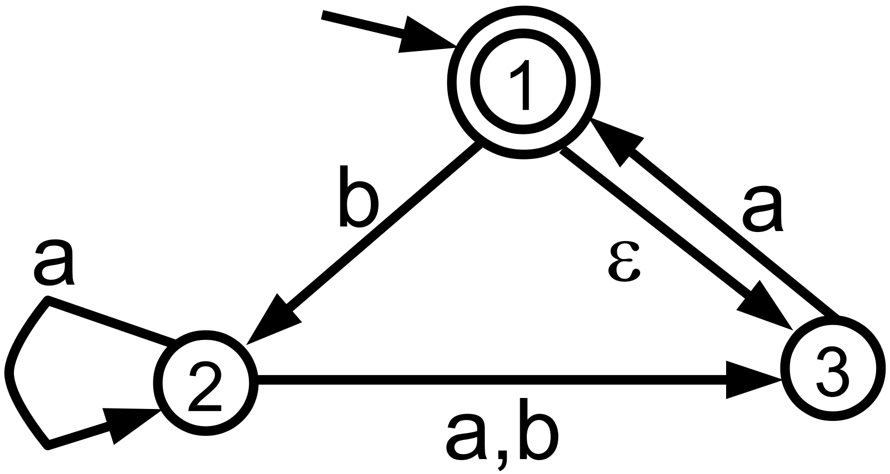
For every NFA N there is DFA M : L(M) = L(N)

- Construction including ε transitions
- Given NFA N $(Q, \Sigma, \delta, q, F)$
- Construct DFA M $(Q', \Sigma, \delta', q', F')$ where:
- $Q' := \text{Powerset}(Q)$
- $q' = E(\{q\})$
- $F' = \{ S : S \in Q' \text{ and } S \text{ contains an element of } F\}$
- $\delta'(S, a) := E(\bigcup_{s \in S} \delta(s,a))$

$$= \{ t : t \in E(\delta (s,a)) \text{ for some } s \in S \}$$

Example: NFA → DFA conversion

NFA

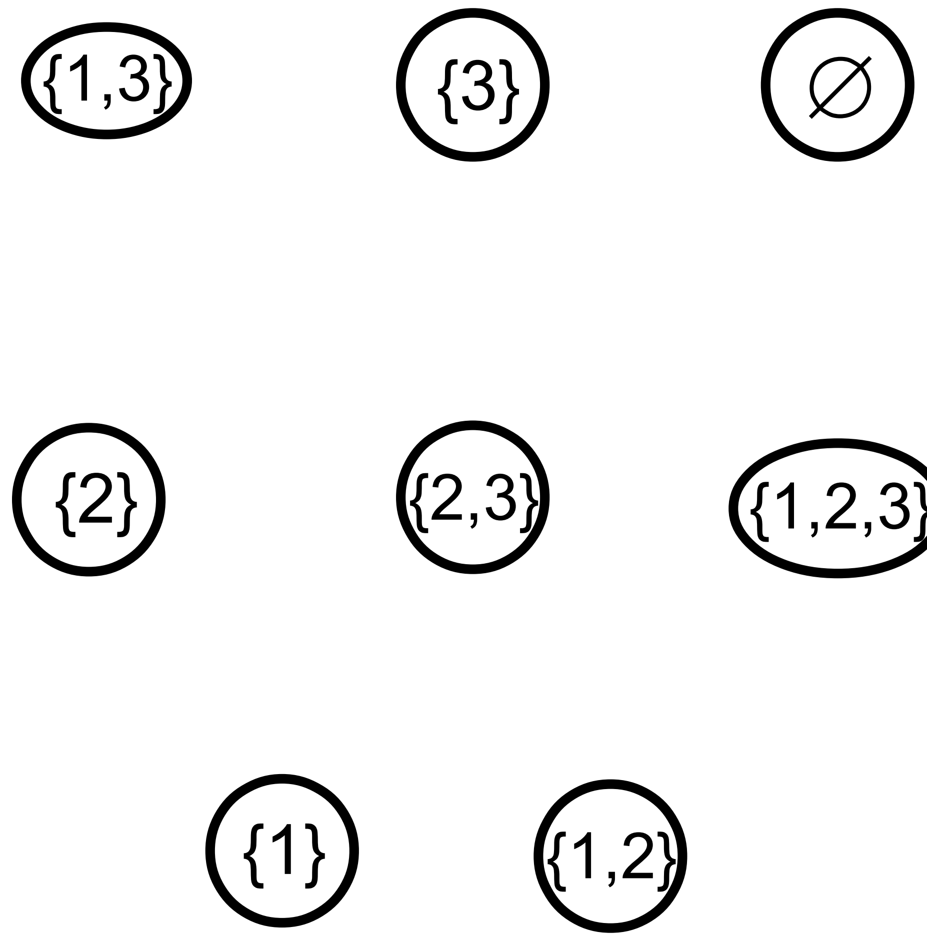


$$Q_{DFA} = \text{Powerset}(Q_{NFA})$$

$$= \text{Powerset}(\{1,2,3\})$$

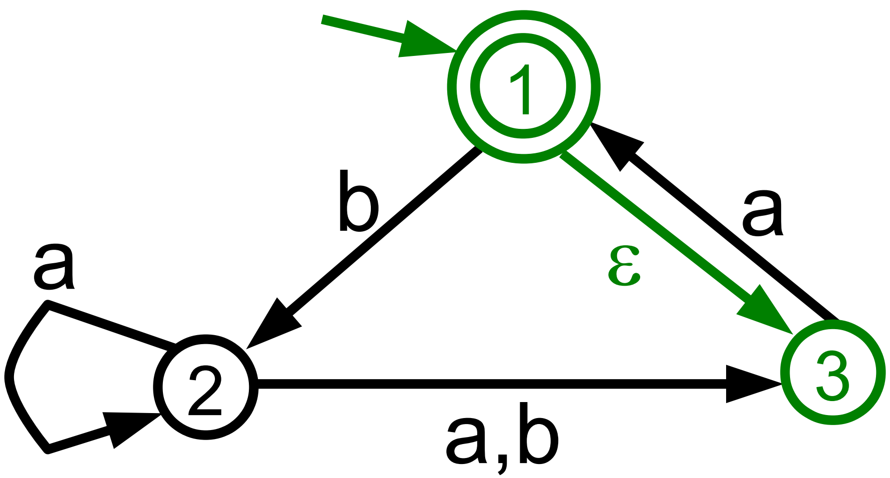
$$= \{\varnothing,\{1\},\{2\},\{3\},\{1,2\}\ldots\}$$

DFA



Example: NFA → DFA conversion

NFA

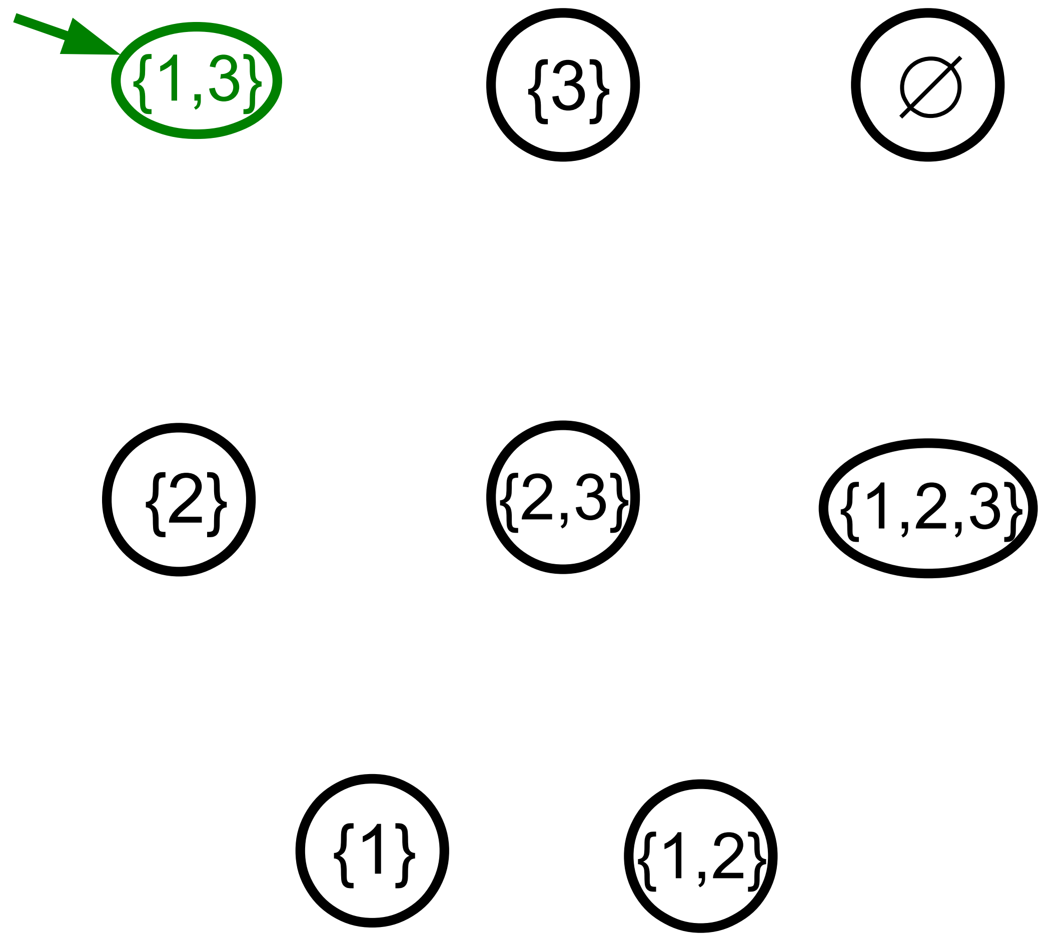


$$q_{DFA} = E(\{q_{NFA}\})$$

$$= E(\{1\})$$

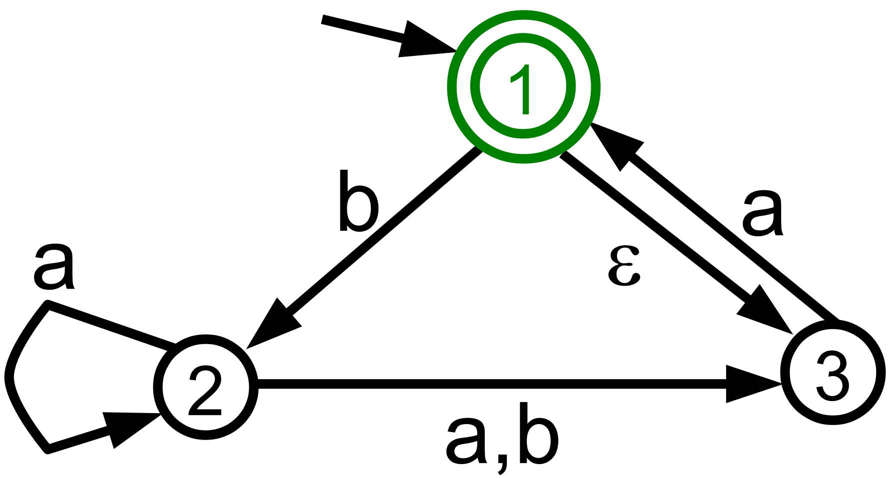
$$= \{1,3\}$$

DFA

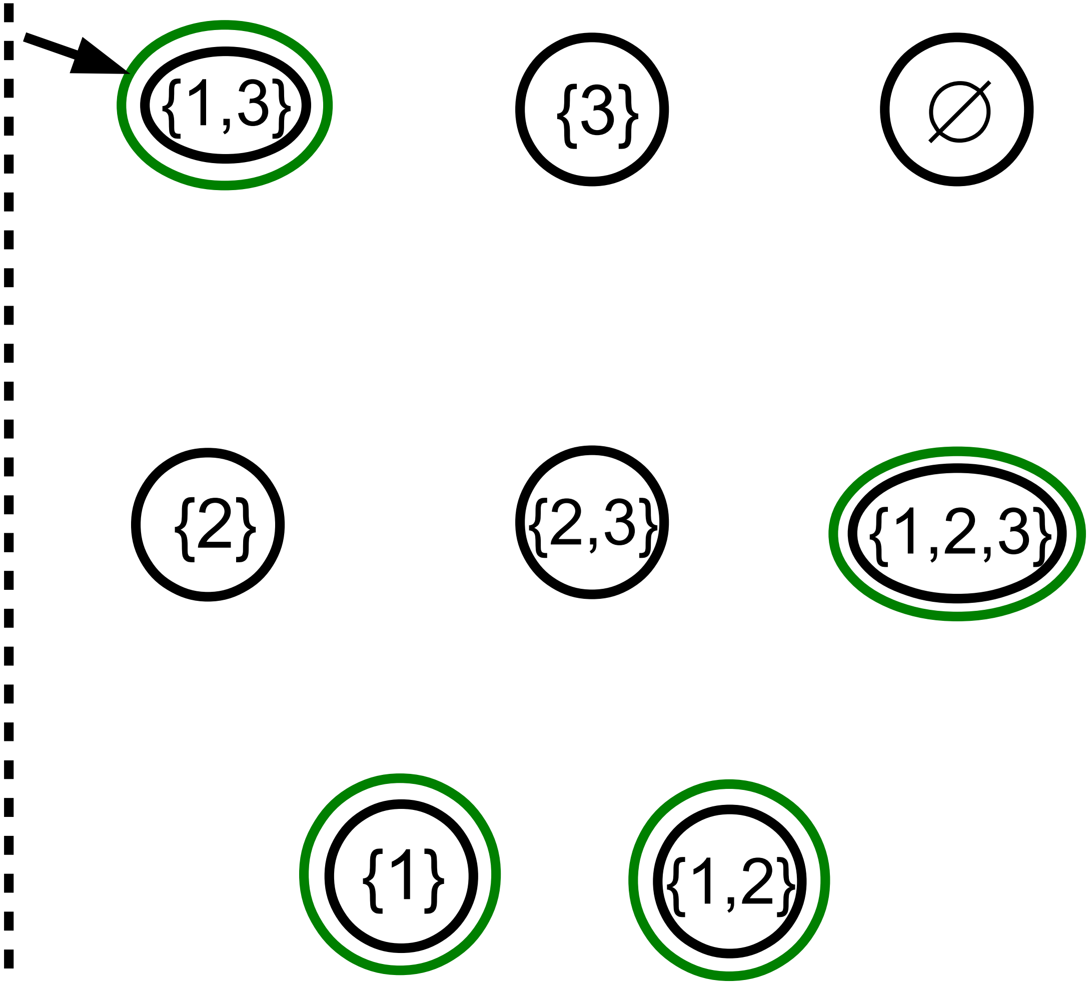


Example: NFA → DFA conversion

NFA



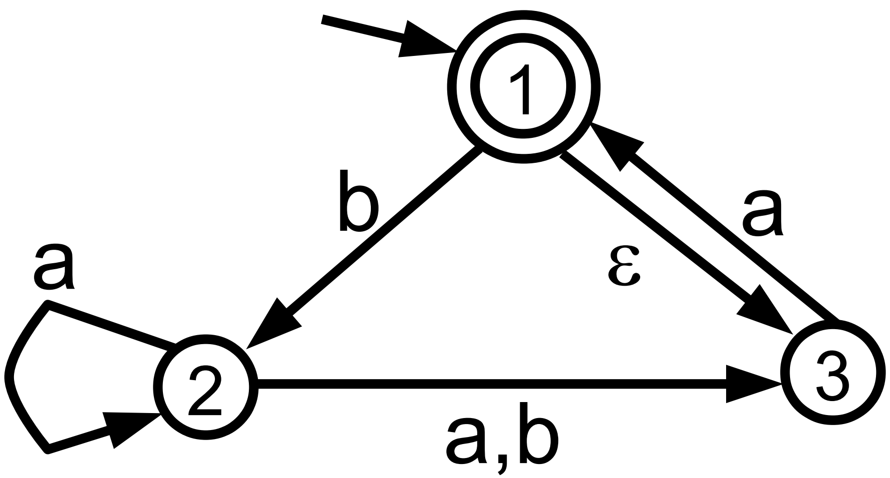
DFA



$F_{DFA} = \{S : S \text{ contains an element of } F_{NFA}\}$

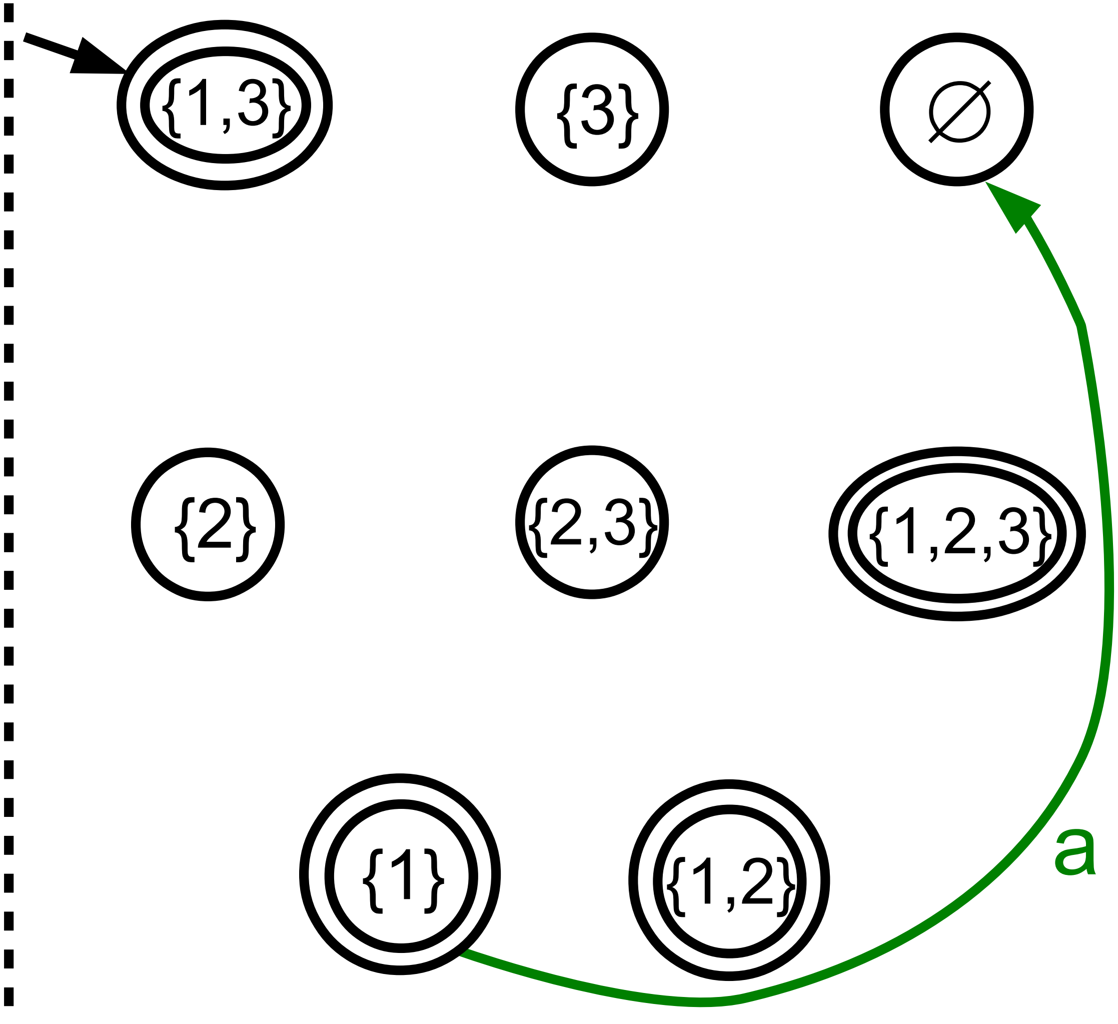
Example: NFA → DFA conversion

NFA

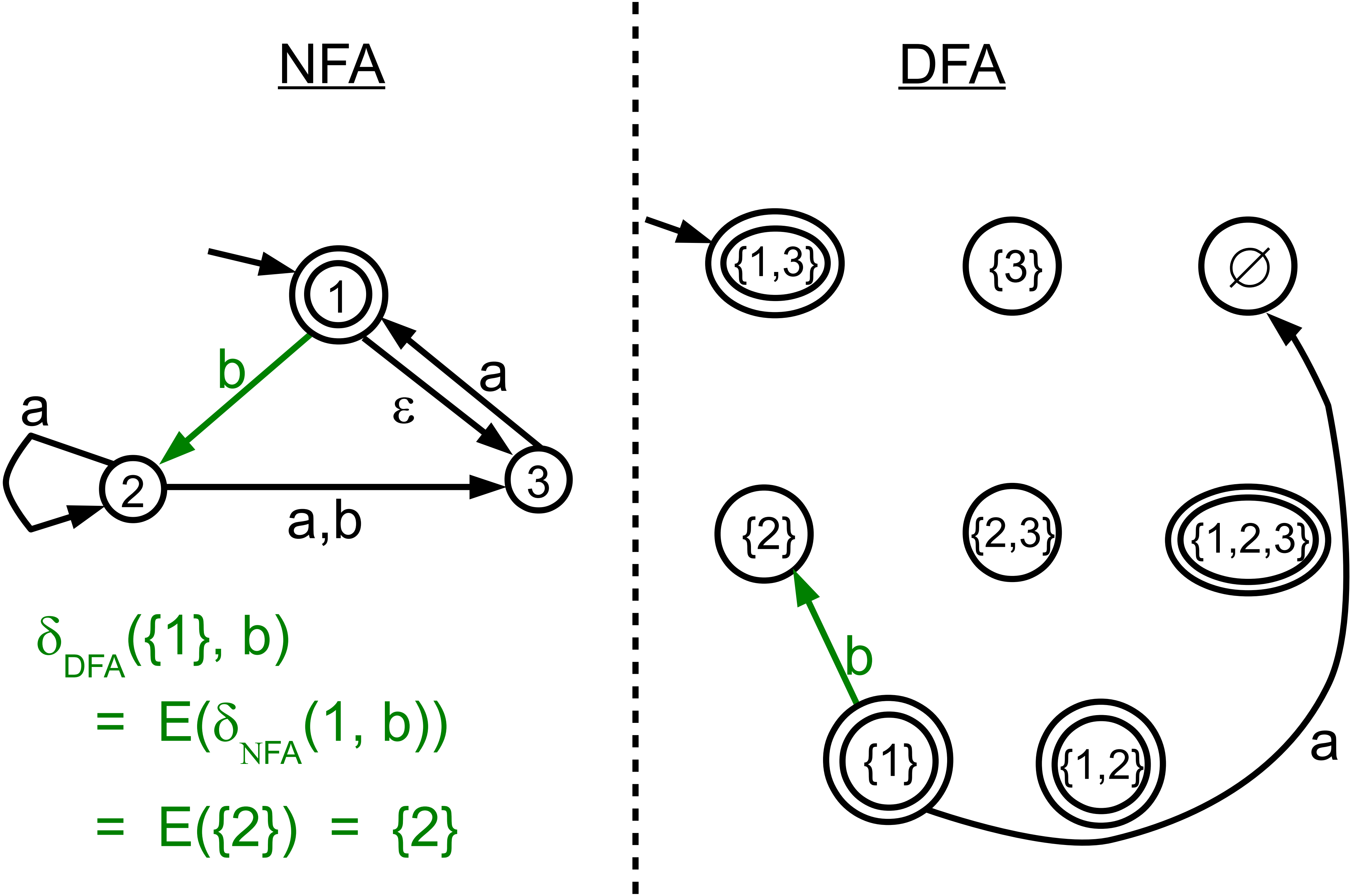


$$\delta_{DFA}(\{1\}, a)$$
$$= E(\delta_{NFA}(1, a))$$
$$= E(\varnothing) = \varnothing$$

DFA



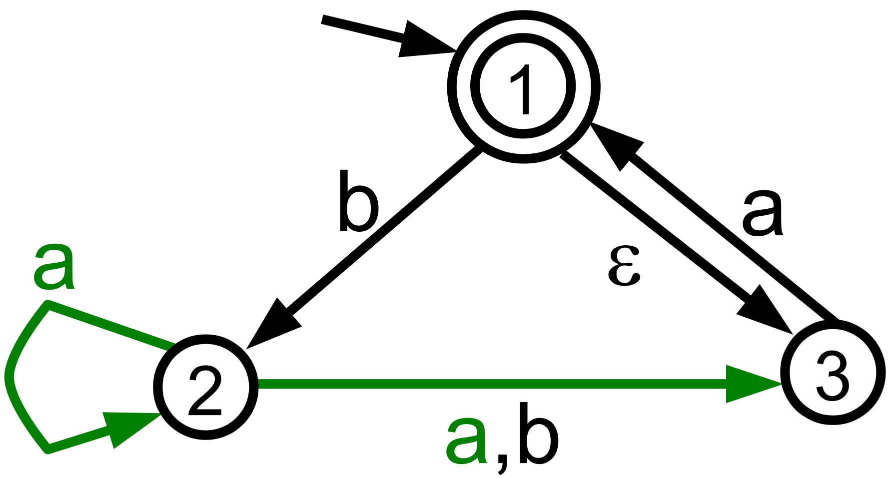
Example: NFA → DFA conversion



$$\delta_{DFA}(\{1\}, b)$$
$$= E(\delta_{NFA}(1, b))$$
$$= E(\{2\}) = \{2\}$$

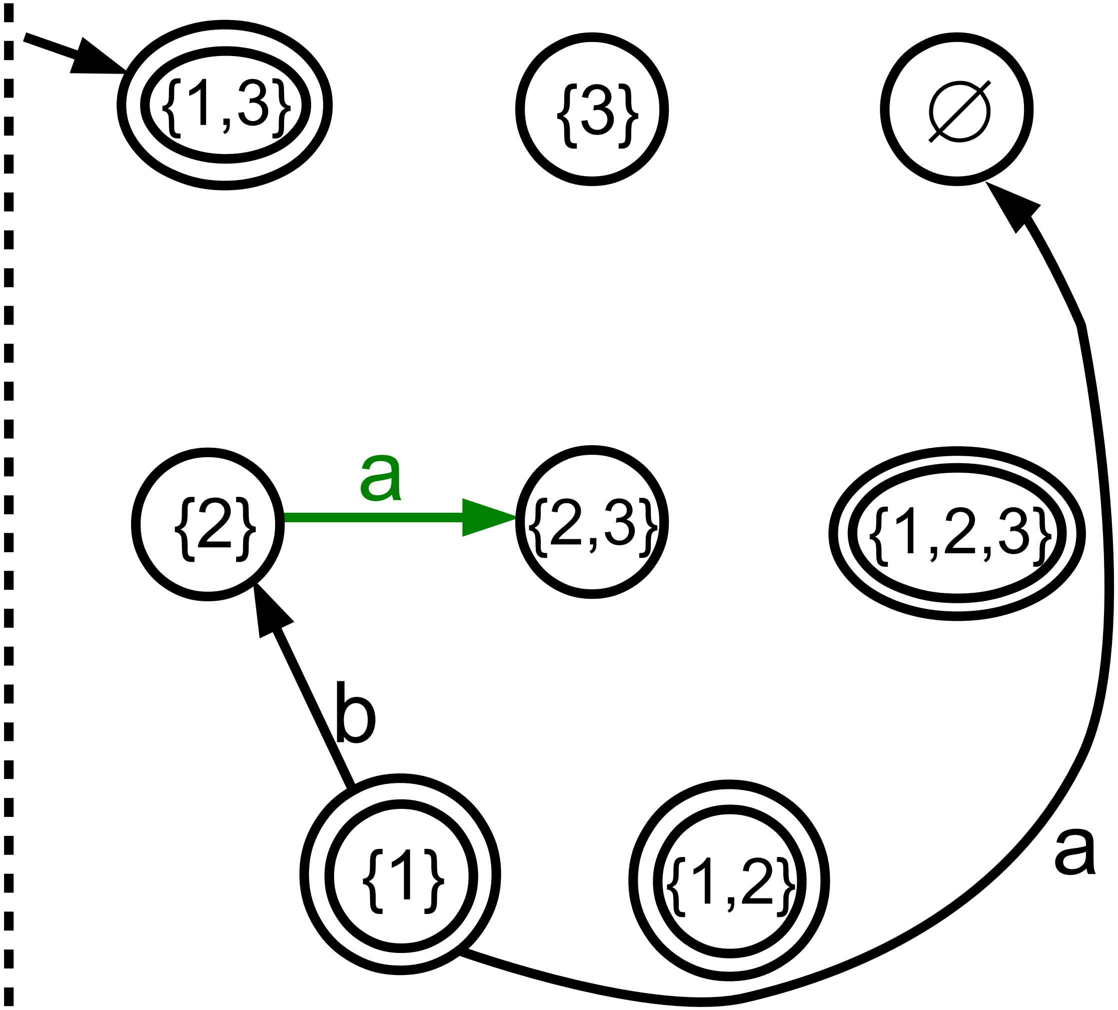
Example: NFA → DFA conversion

NFA

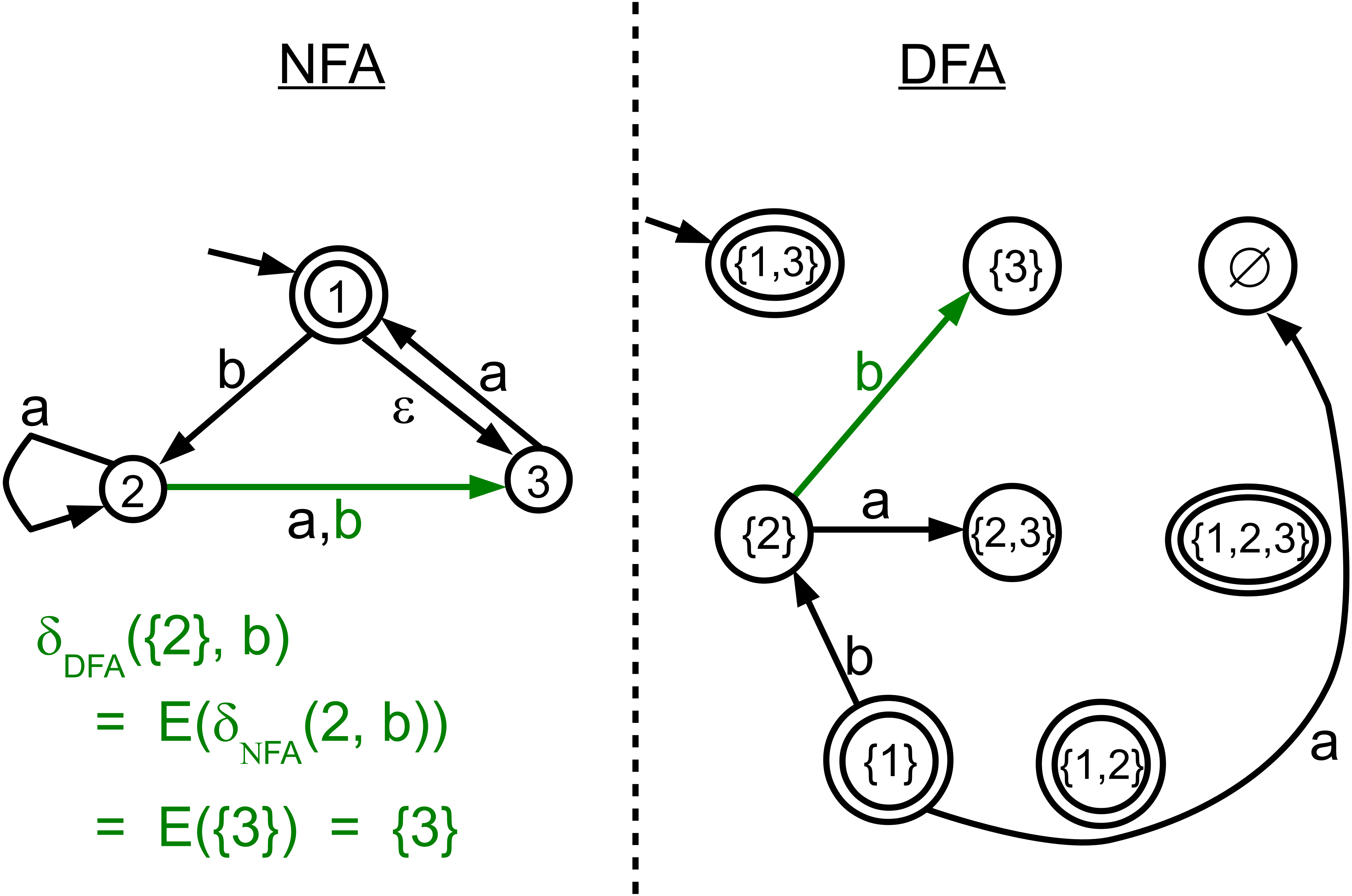


$$\delta_{DFA}(\{2\}, a)$$
$$= E(\delta_{NFA}(2, a))$$
$$= E(\{2,3\}) = \{2,3\}$$

DFA



Example: NFA → DFA conversion

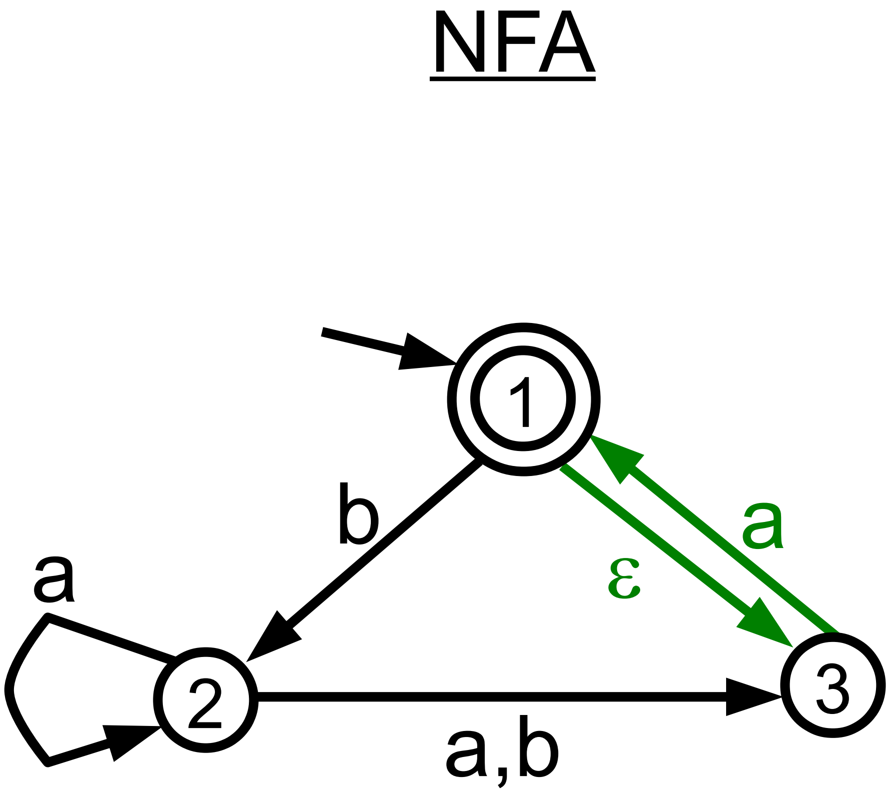


$\delta_{DFA}(\{2\}, b)$

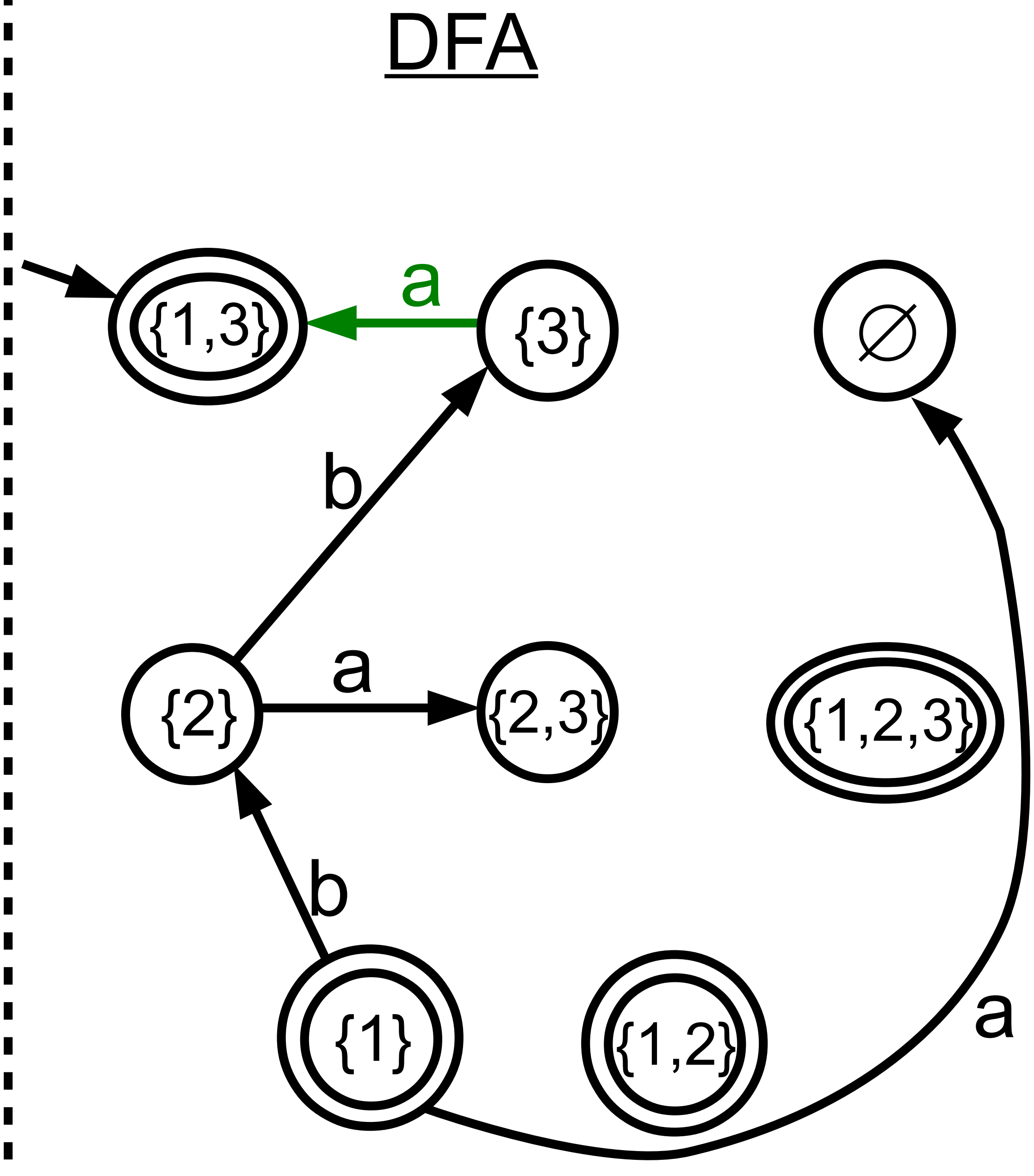
$= E(\delta_{NFA}(2, b))$

$= E(\{3\}) = \{3\}$

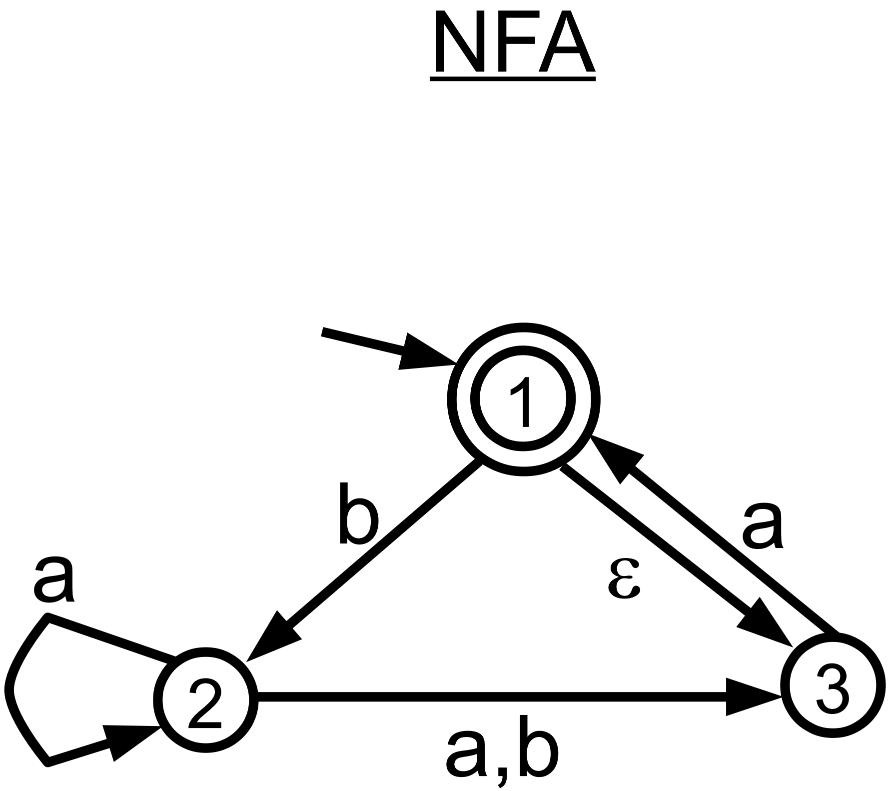
Example: NFA → DFA conversion

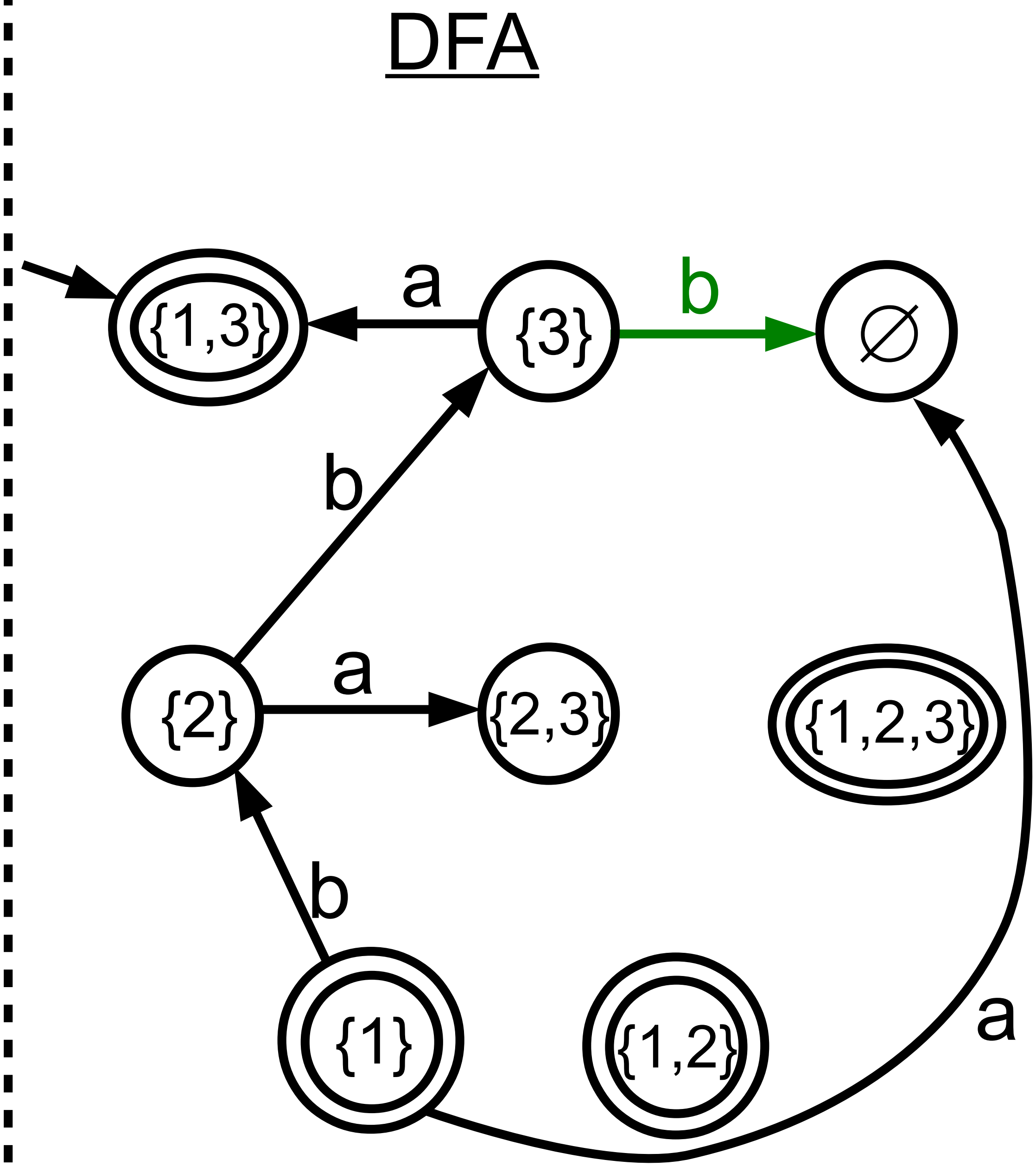


$$\delta_{DFA}(\{3\}, a)$$
$$= E(\delta_{NFA}(3, a))$$
$$= E(\{1\}) = \{1,3\}$$



Example: NFA → DFA conversion

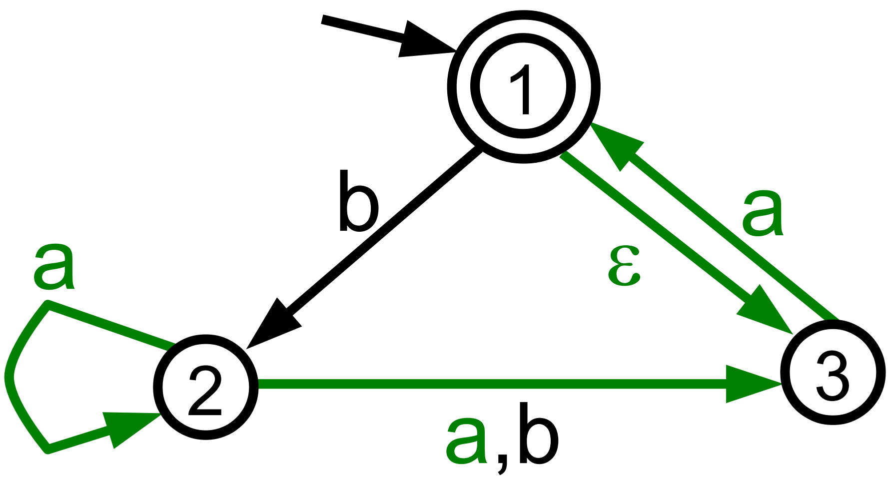




$$\delta_{DFA}(\{3\}, b)$$
$$= E(\delta_{NFA}(3, b))$$
$$= E(\varnothing) = \varnothing$$

Example: NFA → DFA conversion

NFA

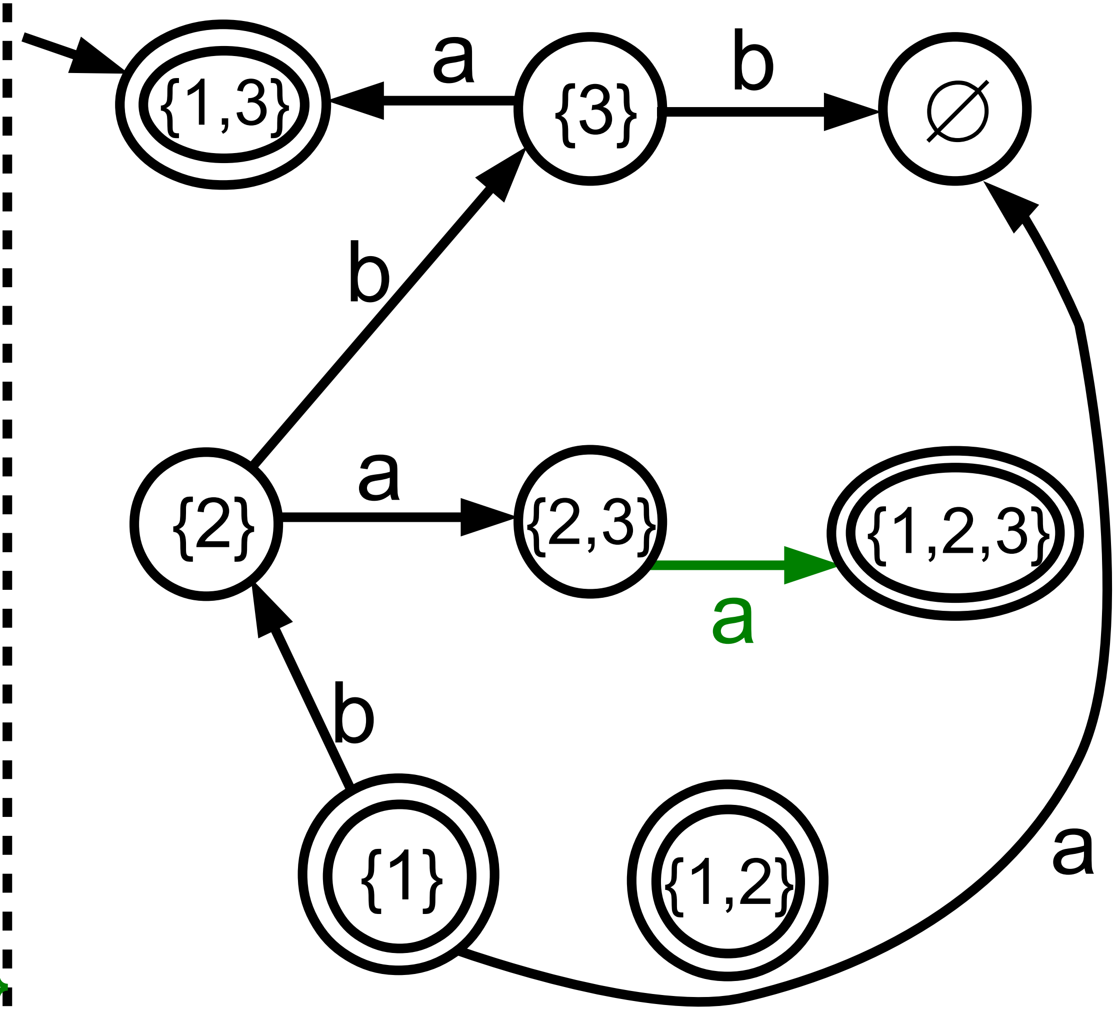


$$\delta_{DFA}(\{2,3\}, a)$$

$$= E(\delta_{NFA}(2,a) \cup \delta_{NFA}(3,a))$$

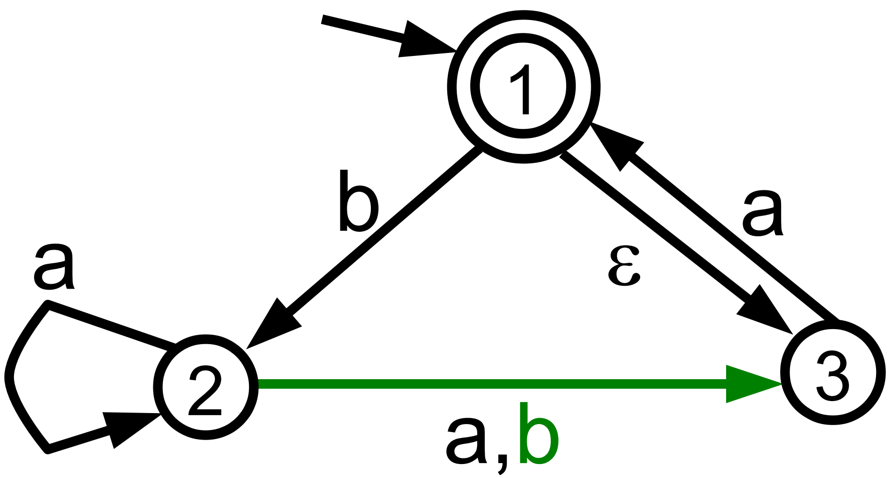
$$= E(\{2,3\} \cup \{1\}) = \{1,2,3\}$$

DFA



Example: NFA → DFA conversion

NFA

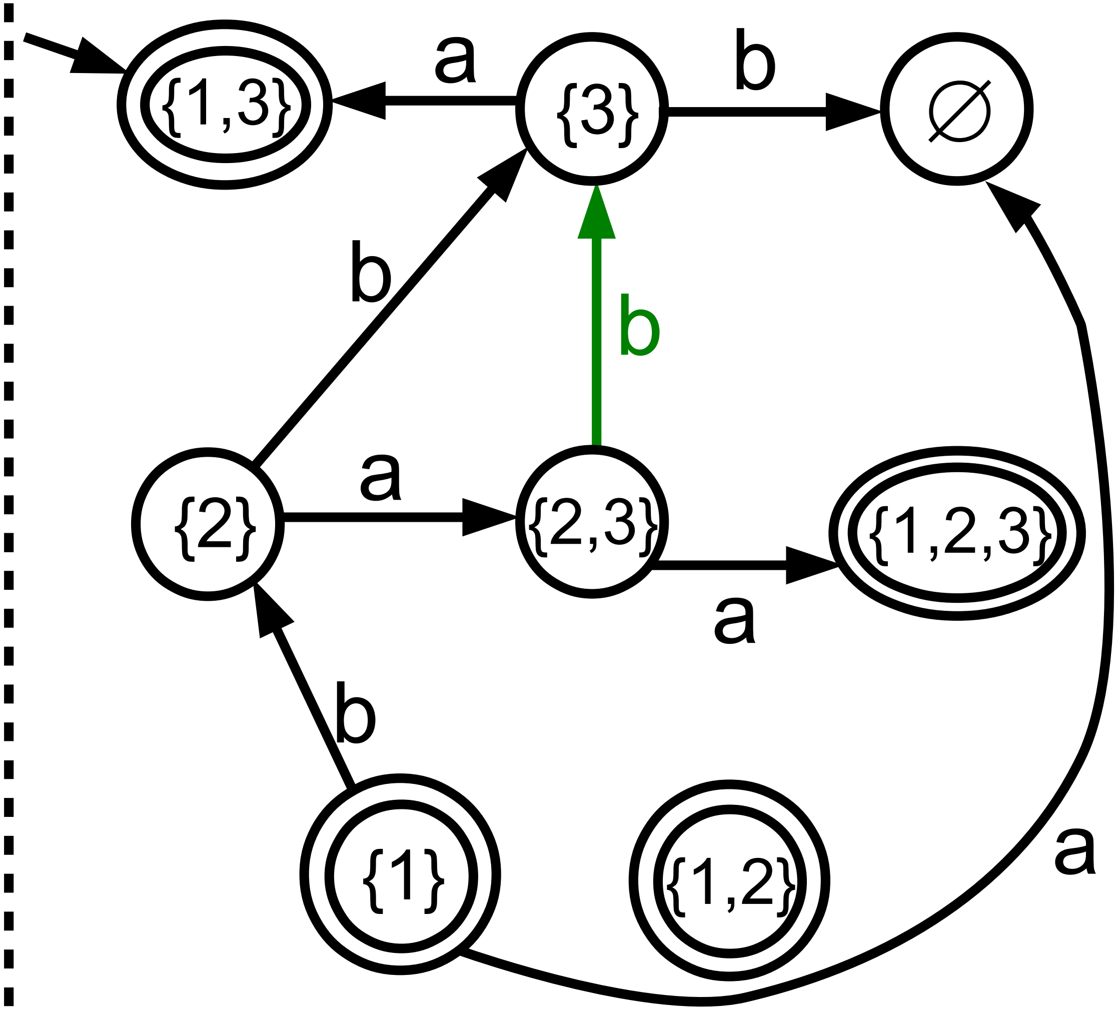


$\delta_{DFA}(\{2,3\}, b)$

$= E(\delta_{NFA}(2,b) \cup \delta_{NFA}(3,b))$

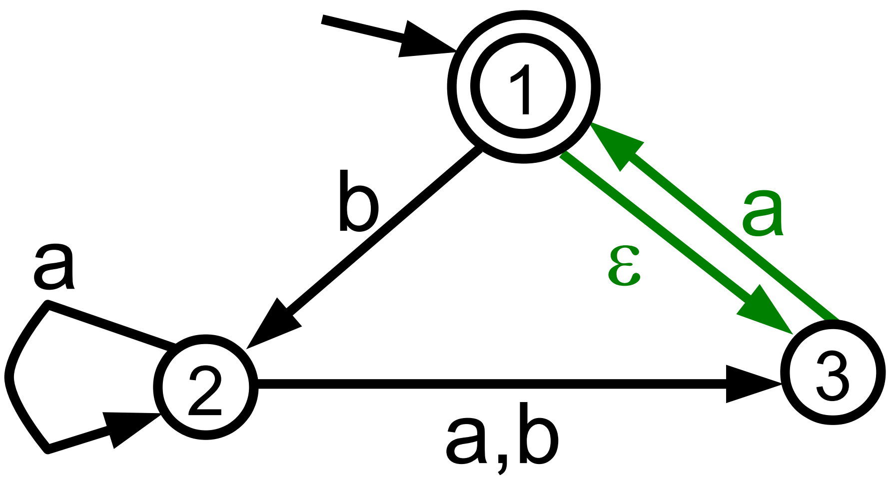
$= E(\{3\} \cup \varnothing) = \{3\}$

DFA

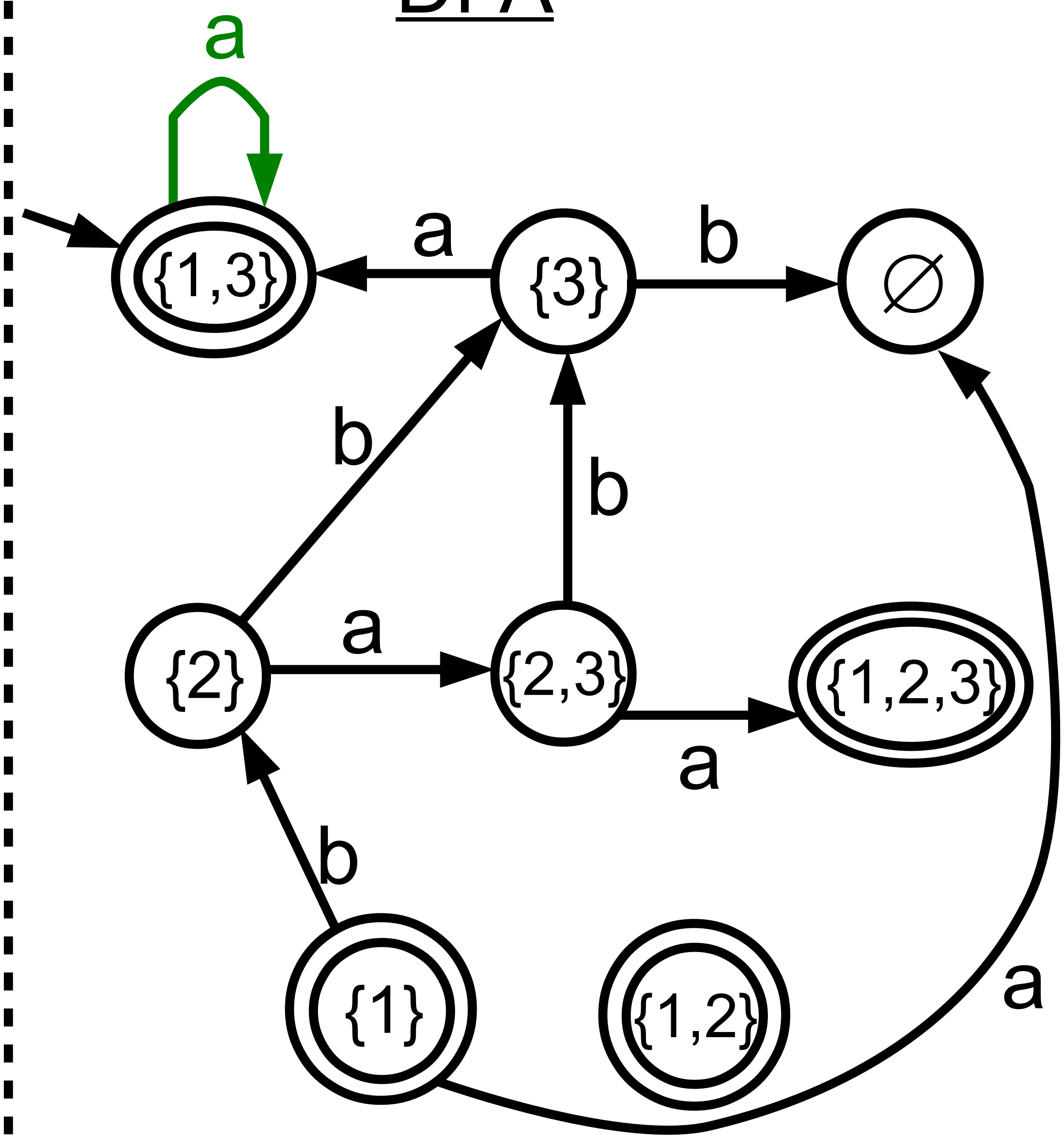


Example: NFA → DFA conversion

NFA



DFA



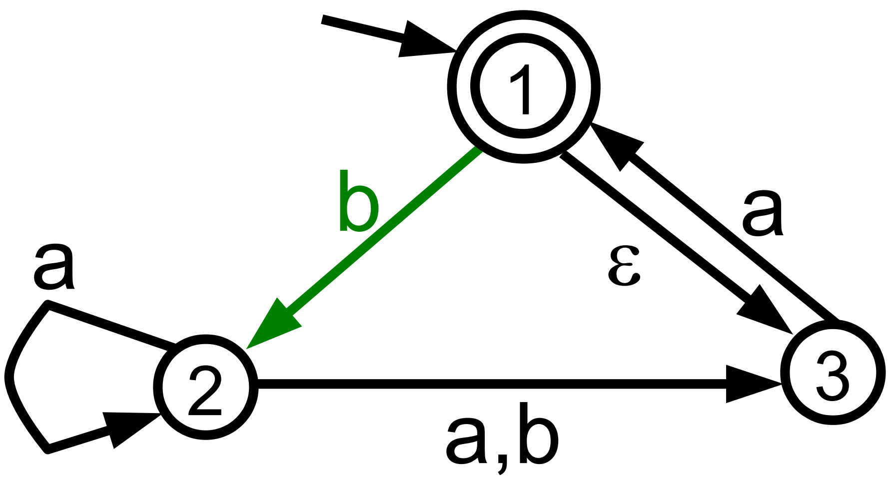
$$\delta_{DFA}(\{1,3\}, a)$$

$$= E(\delta_{NFA}(1,a) \cup \delta_{NFA}(3,a))$$

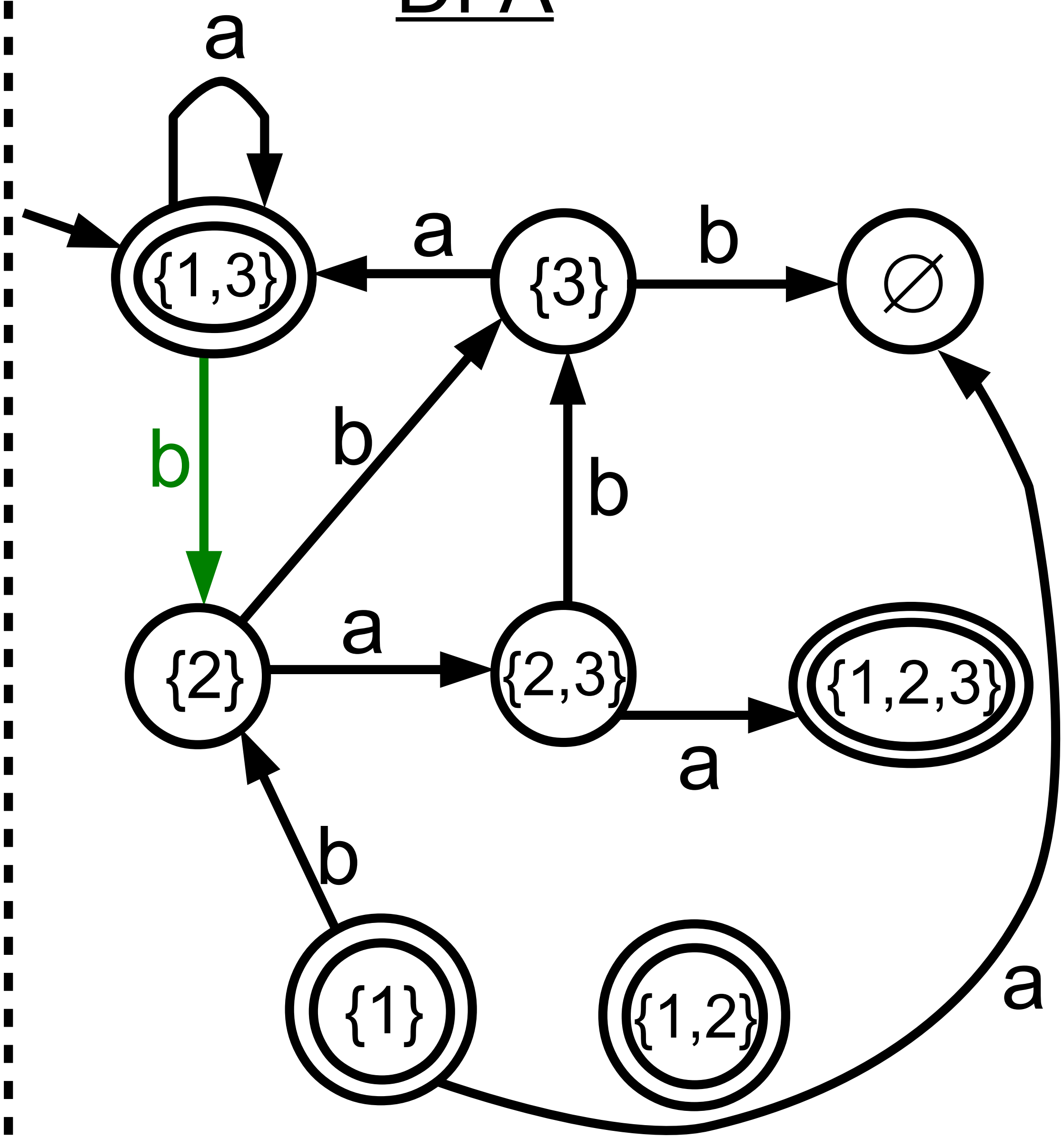
$$= E(\varnothing \cup \{1\}) = \{1,3\}$$

Example: NFA → DFA conversion

NFA



DFA



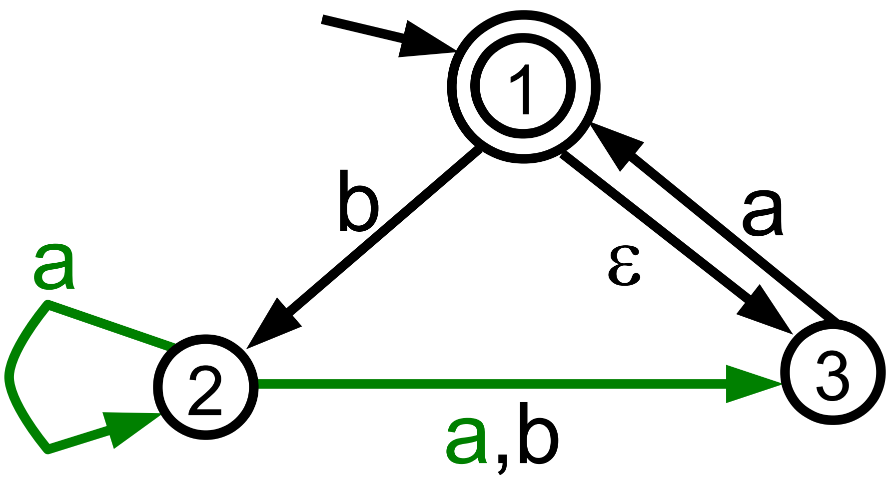
$$\delta_{DFA}(\{1,3\}, b)$$

$$= E(\delta_{NFA}(1,b) \cup \delta_{NFA}(3,b))$$

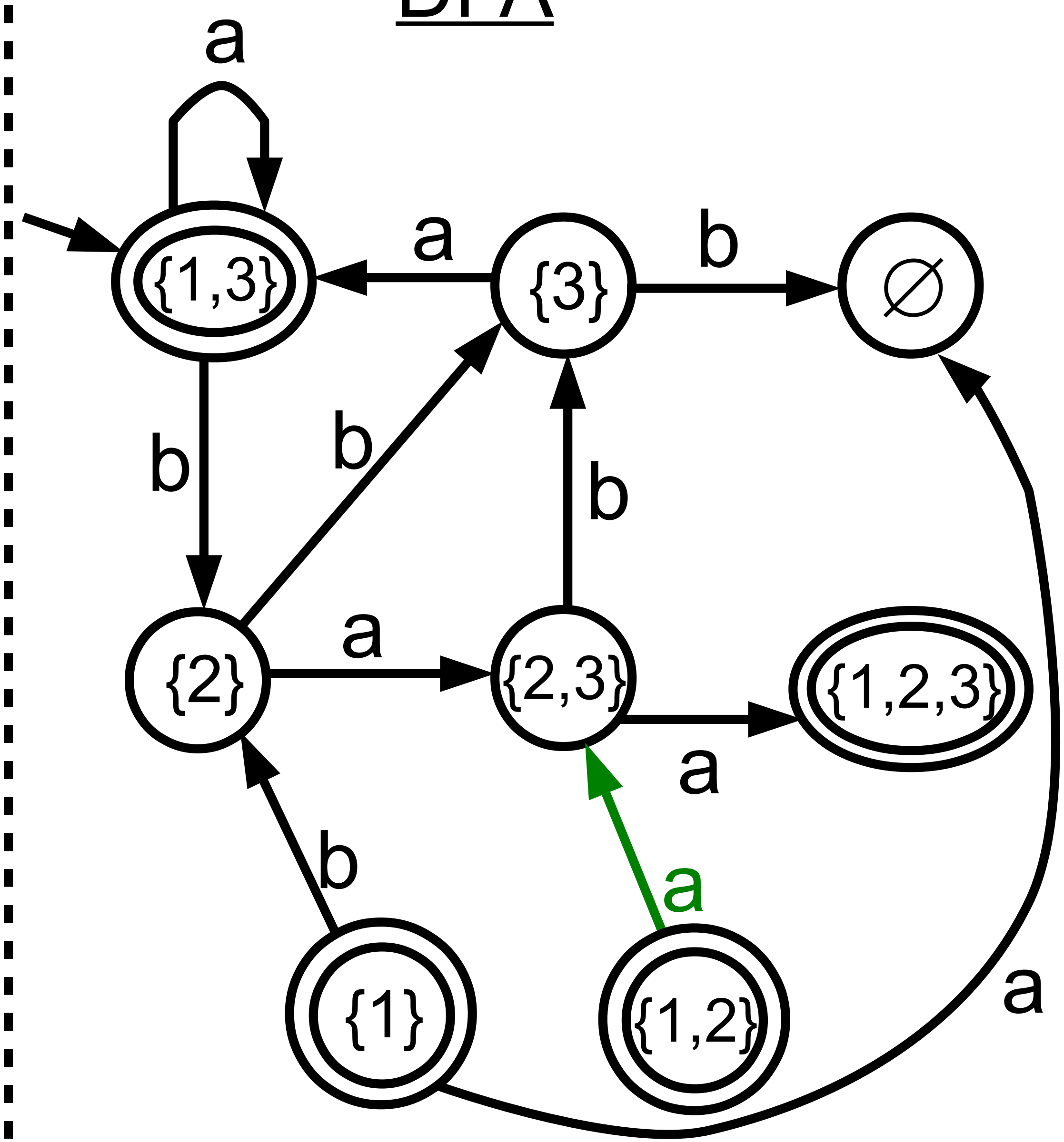
$$= E(\{2\} \cup \varnothing) = \{2\}$$

Example: NFA → DFA conversion

NFA



DFA



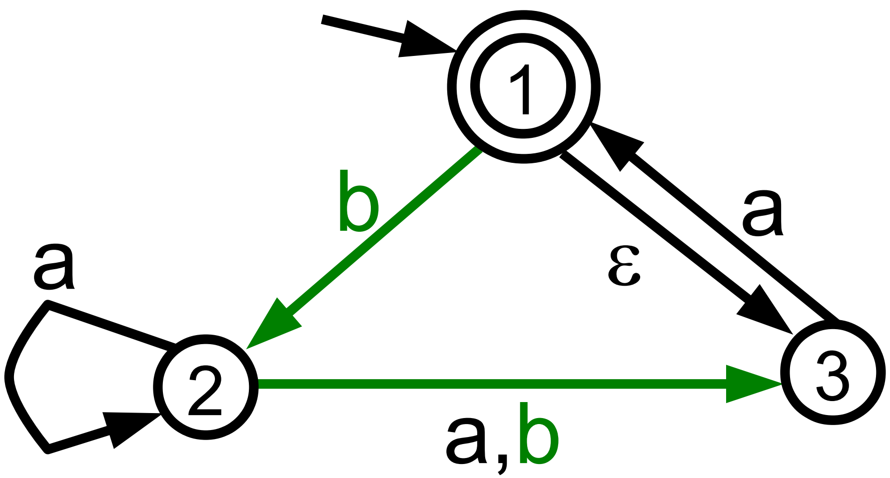
$$\delta_{DFA}(\{1,2\}, a)$$

$$= E(\delta_{NFA}(1,a) \cup \delta_{NFA}(2,a))$$

$$= E(\varnothing \cup \{2,3\}) = \{2,3\}$$

Example: NFA → DFA conversion

NFA

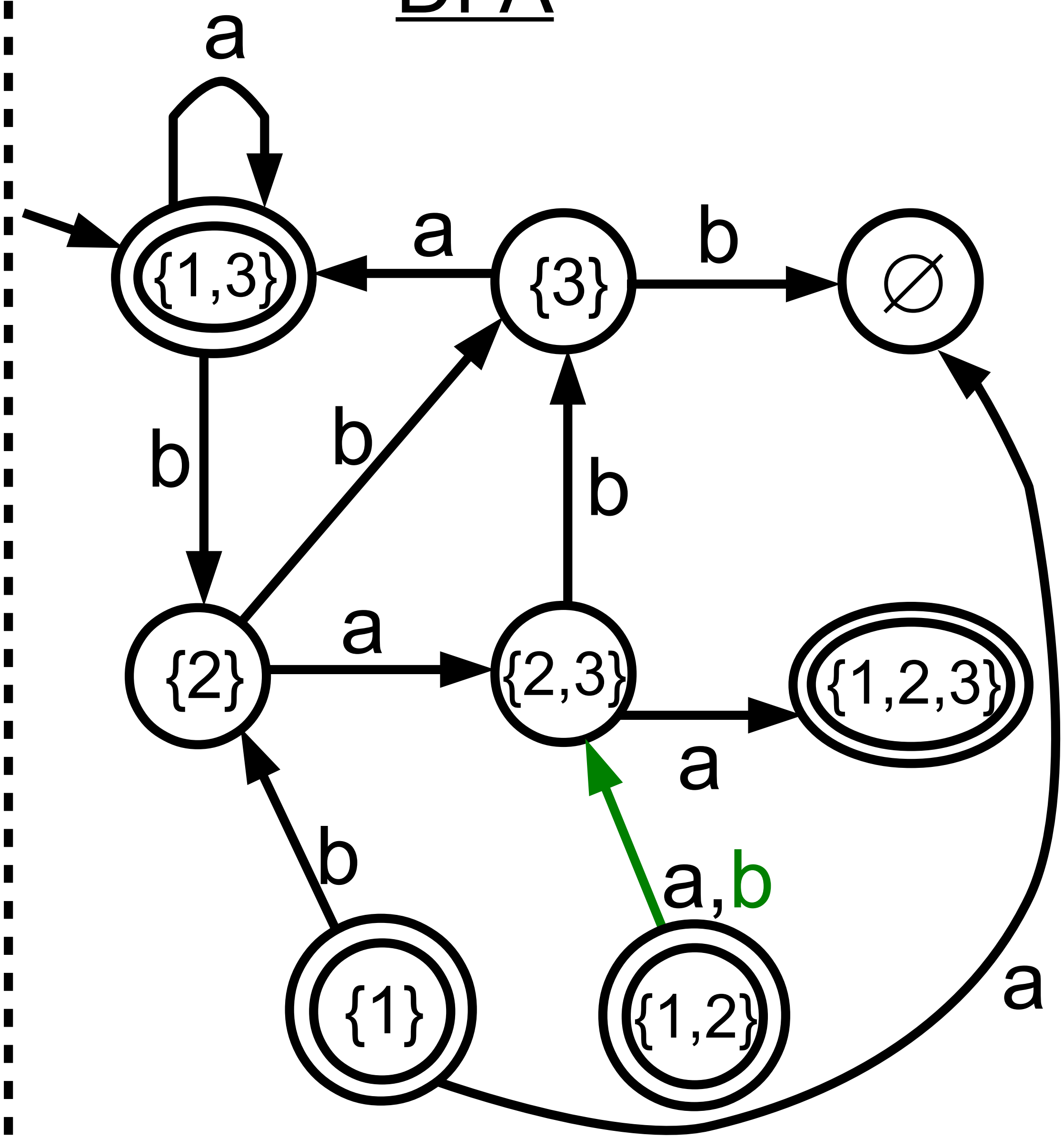


$\delta_{DFA}(\{1,2\}, b)$

$= E(\delta_{NFA}(1,b) \cup \delta_{NFA}(2,b))$

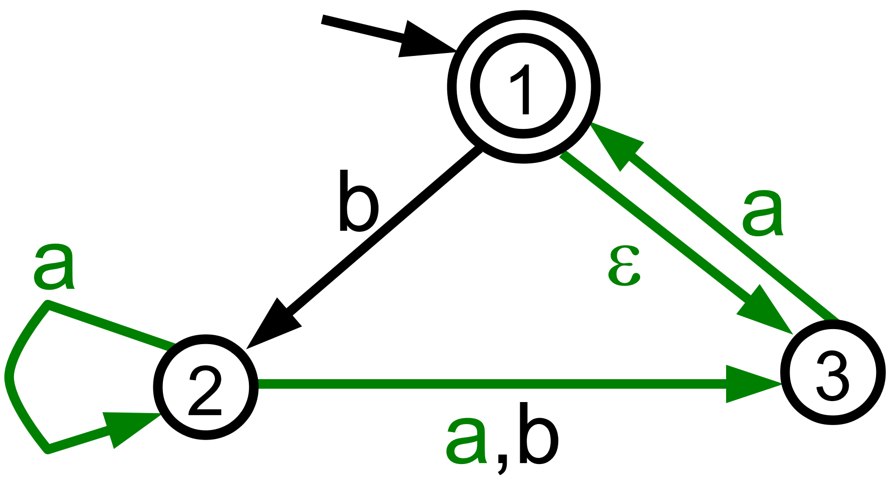
$= E(\{2\} \cup \{3\}) = \{2,3\}$

DFA

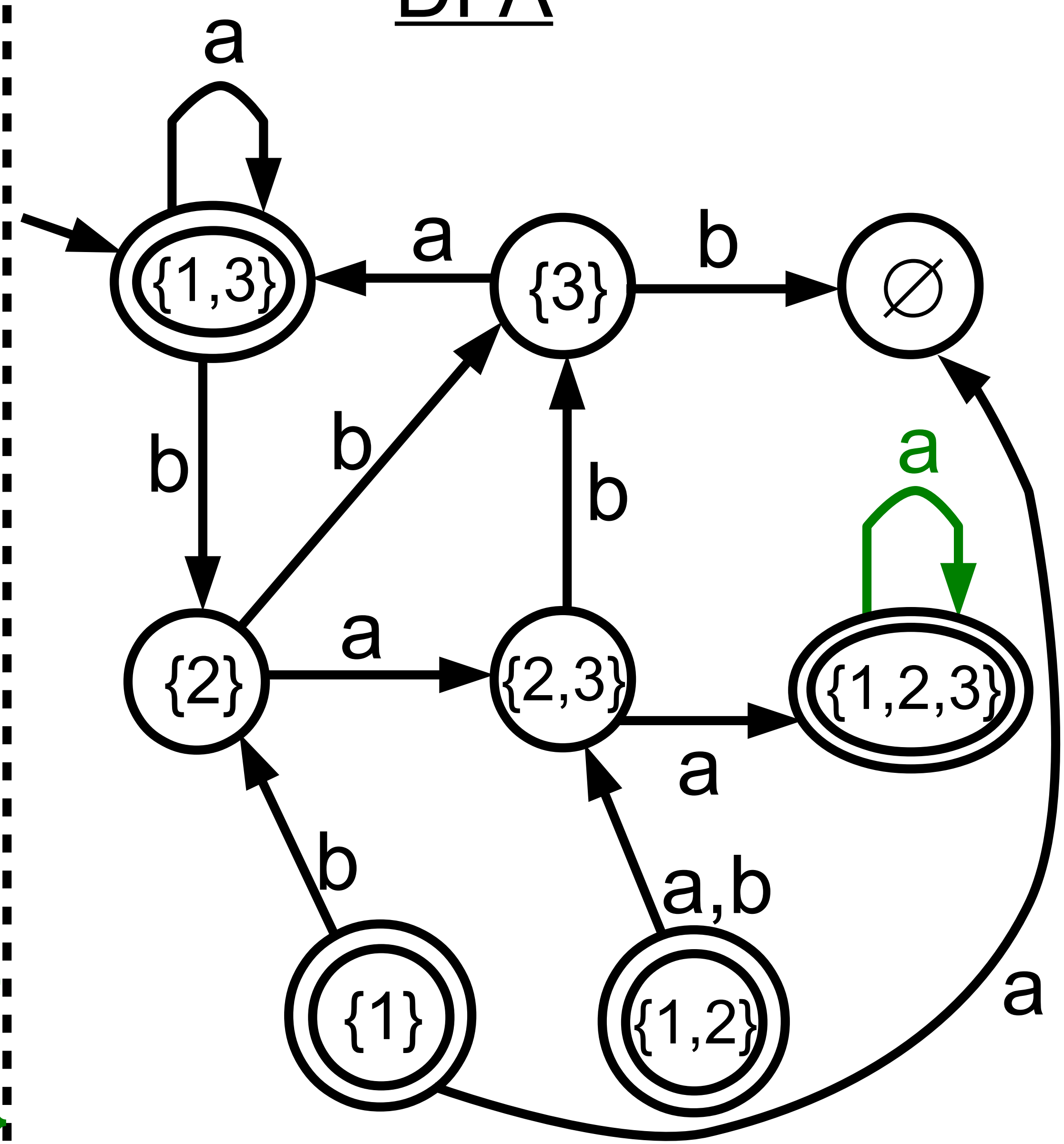


Example: NFA → DFA conversion

NFA



DFA



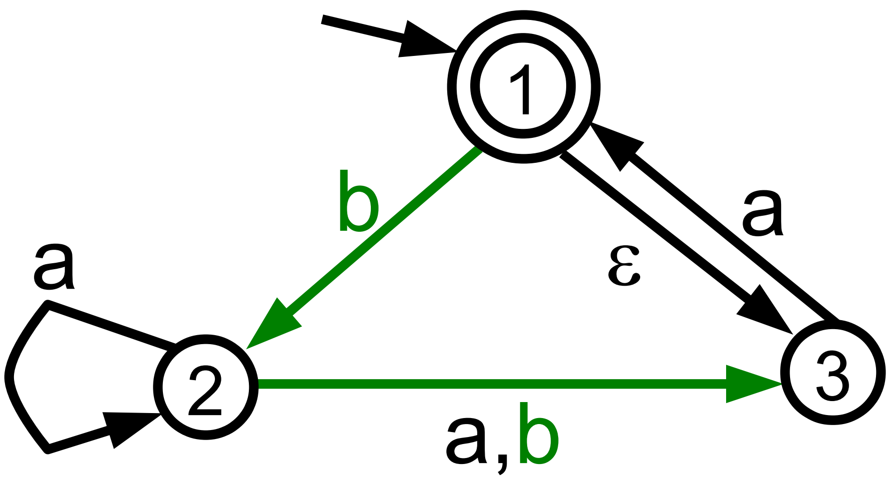
$$\delta_{DFA}(\{1,2,3\}, a)$$

$$=E(\delta_{NFA}(1,a) \cup \delta_{NFA}(2,a) \cup \delta_{NFA}(3,a))$$

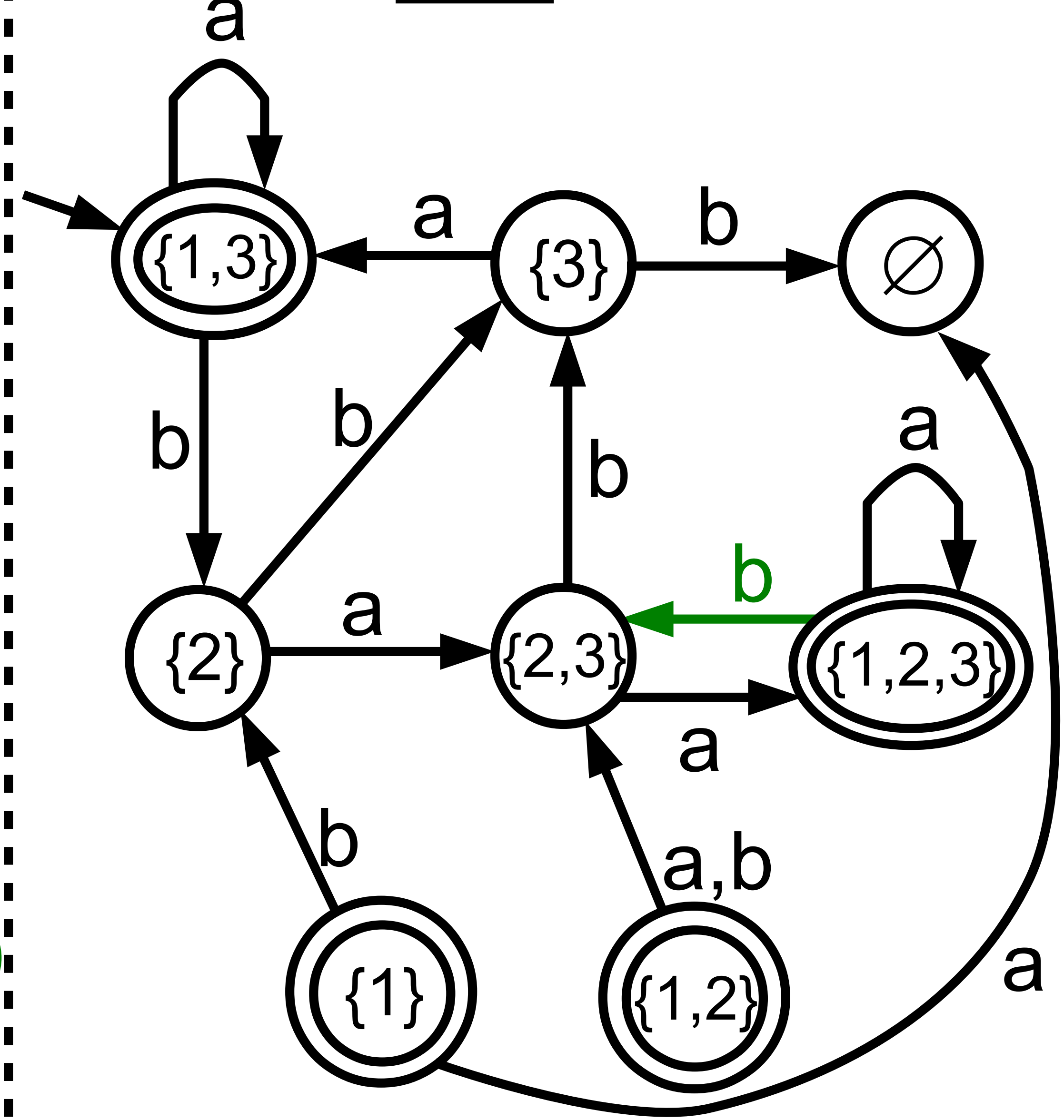
$$=E(\emptyset \cup \{2,3\} \cup \{1\}) = \{1,2,3\}$$

Example: NFA → DFA conversion

NFA



DFA

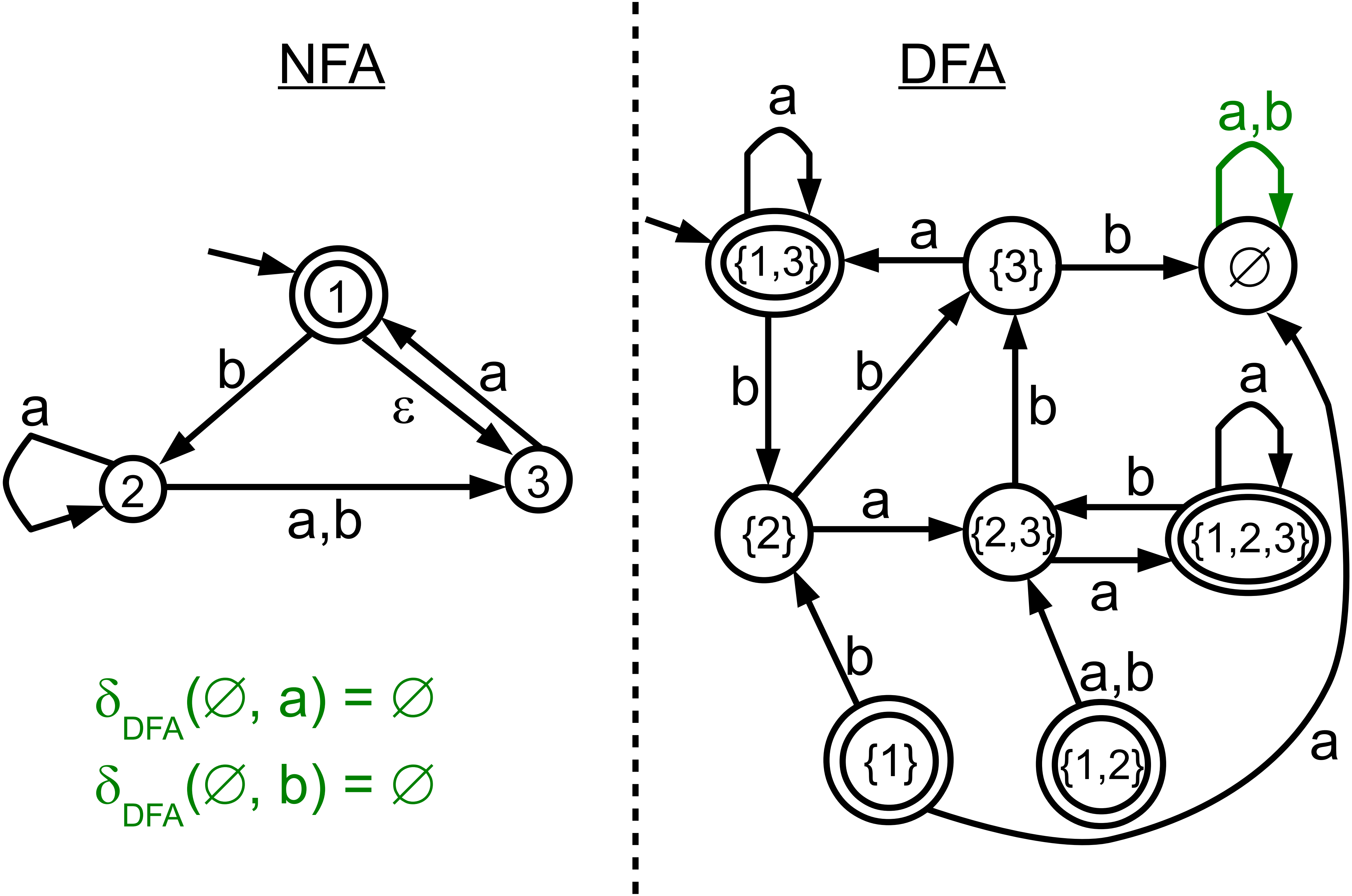


$$\delta_{DFA}(\{1,2,3\}, b)$$

$$=E(\delta_{NFA}(1,b) \cup \delta_{NFA}(2,b) \cup \delta_{NFA}(3,b))$$

$$=E(\{2\} \cup \{3\} \cup \emptyset) = \{2,3\}$$

Example: NFA → DFA conversion

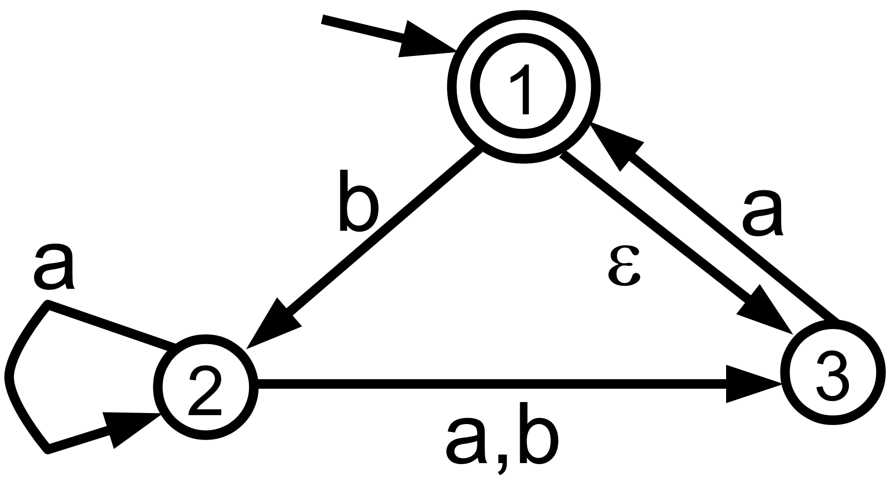


$\delta_{DFA}(\varnothing, a) = \varnothing$

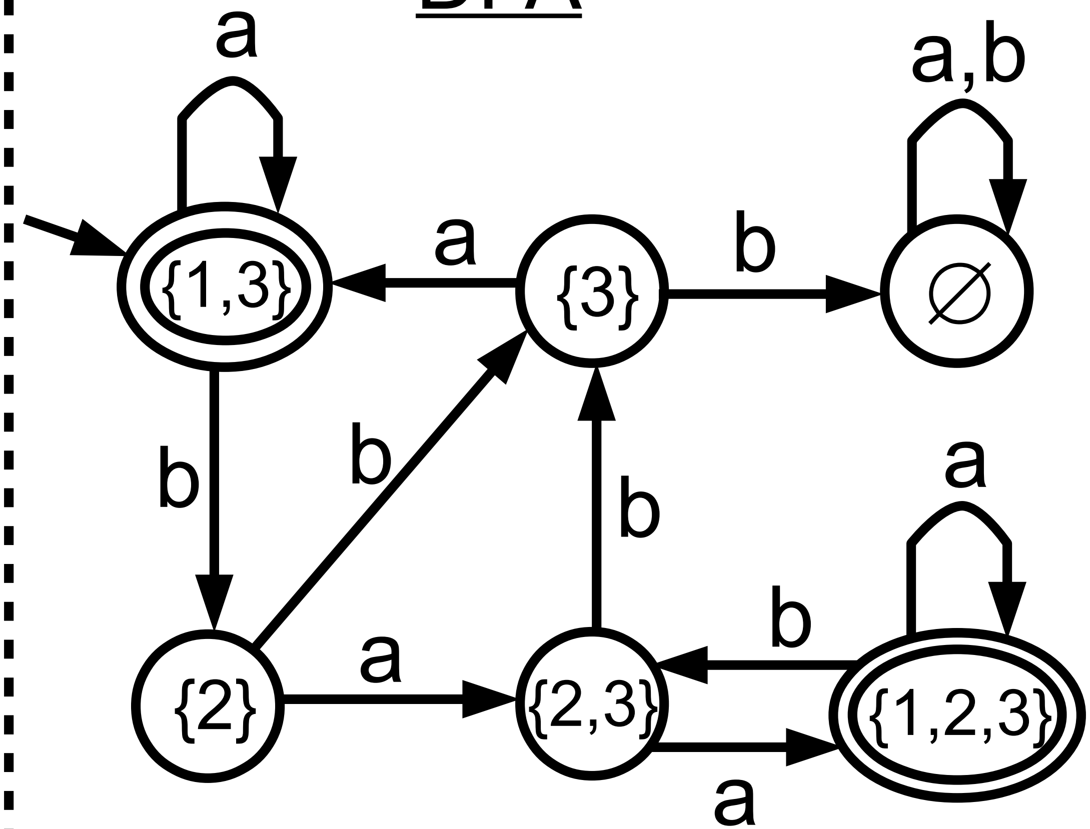
$\delta_{DFA}(\varnothing, b) = \varnothing$

Example: NFA → DFA conversion

NFA

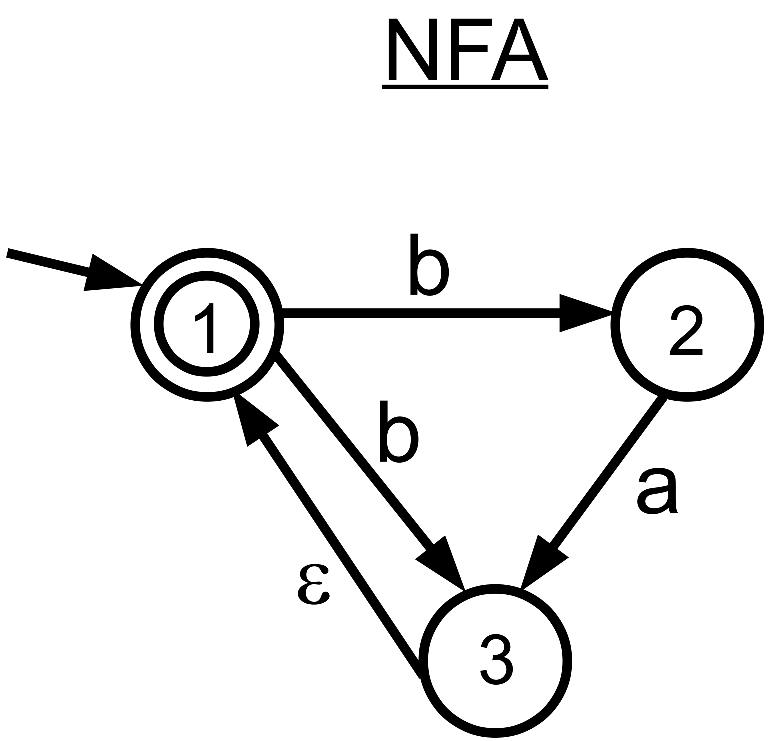


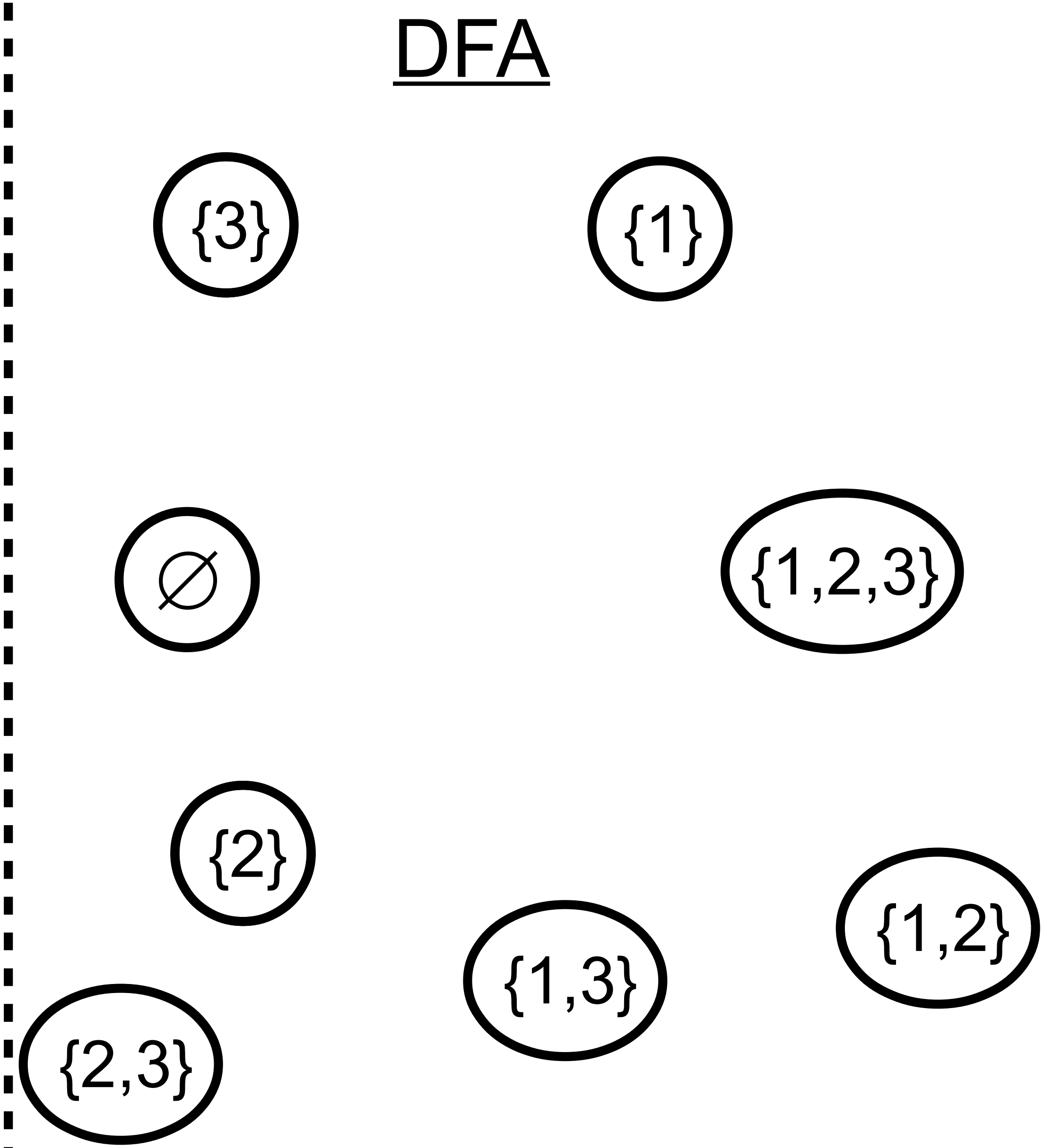
DFA



We can delete the unreachable states.

ANOTHER Example: NFA → DFA conversion





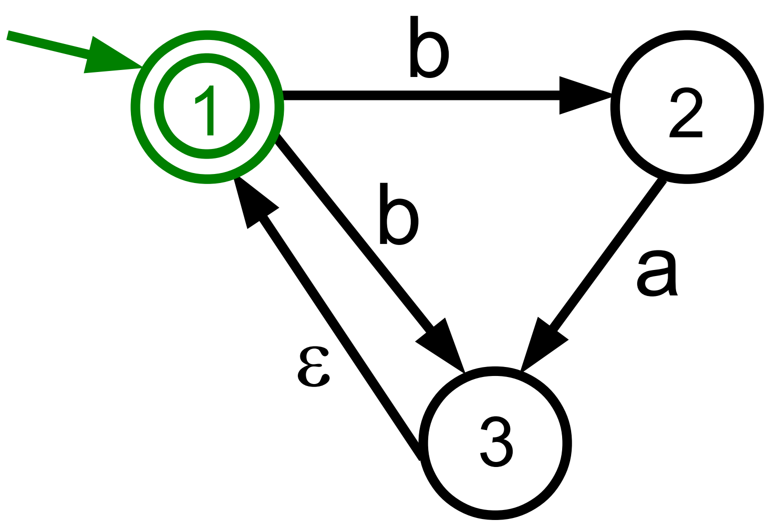
Q_{DFA} = Powerset(Q_{NFA})

= Powerset({1,2,3})

= {∅,{1},{2},{3},{1,2}...}

ANOTHER Example: NFA → DFA conversion

NFA

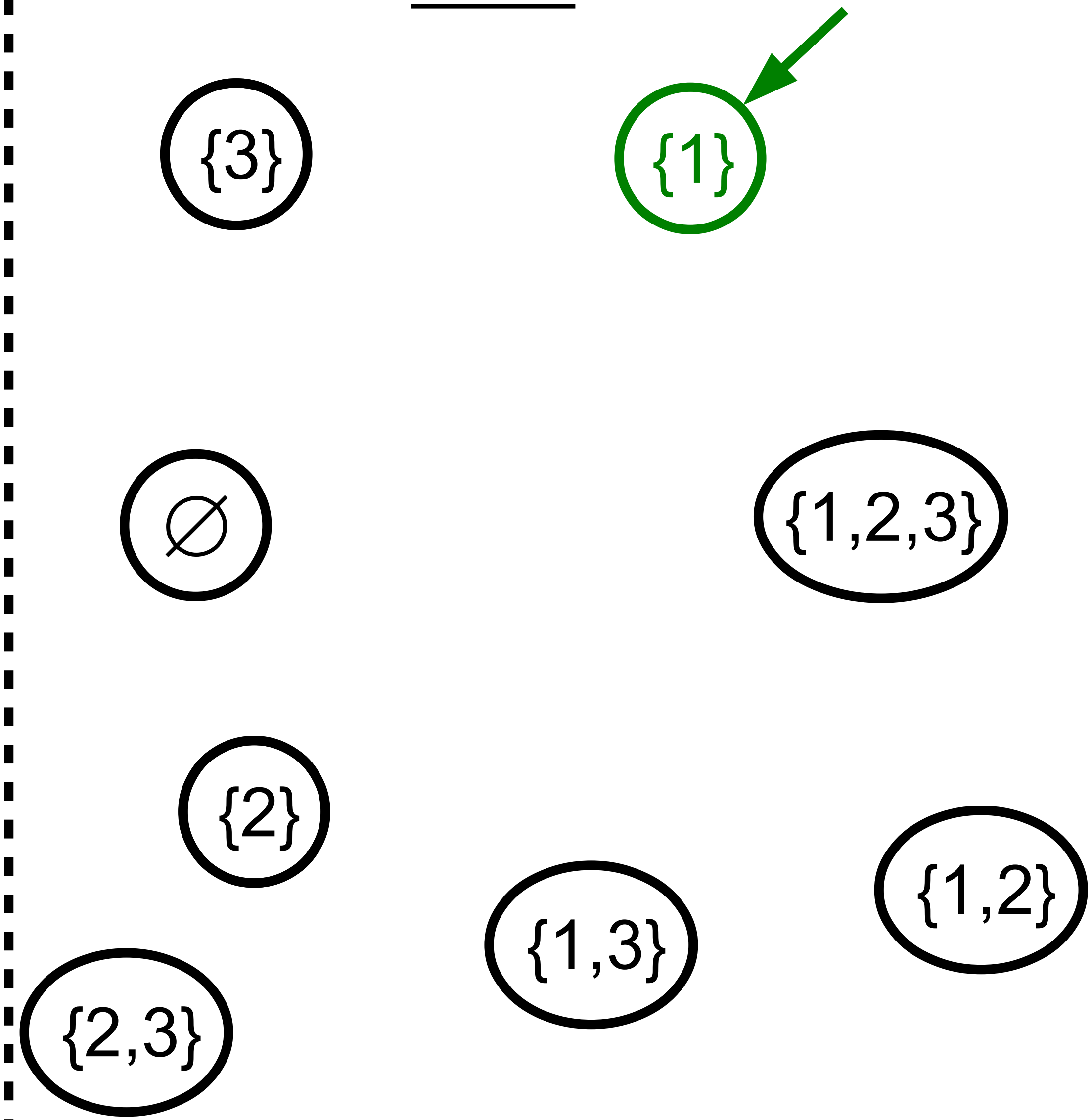


$$q_{DFA} = E(\{q_{NFA}\})$$

$$= E(\{1\})$$

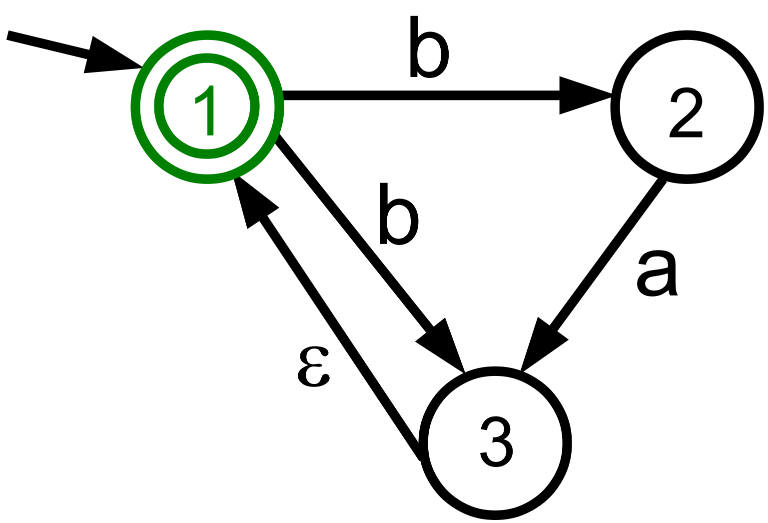
$$= \{1\}$$

DFA

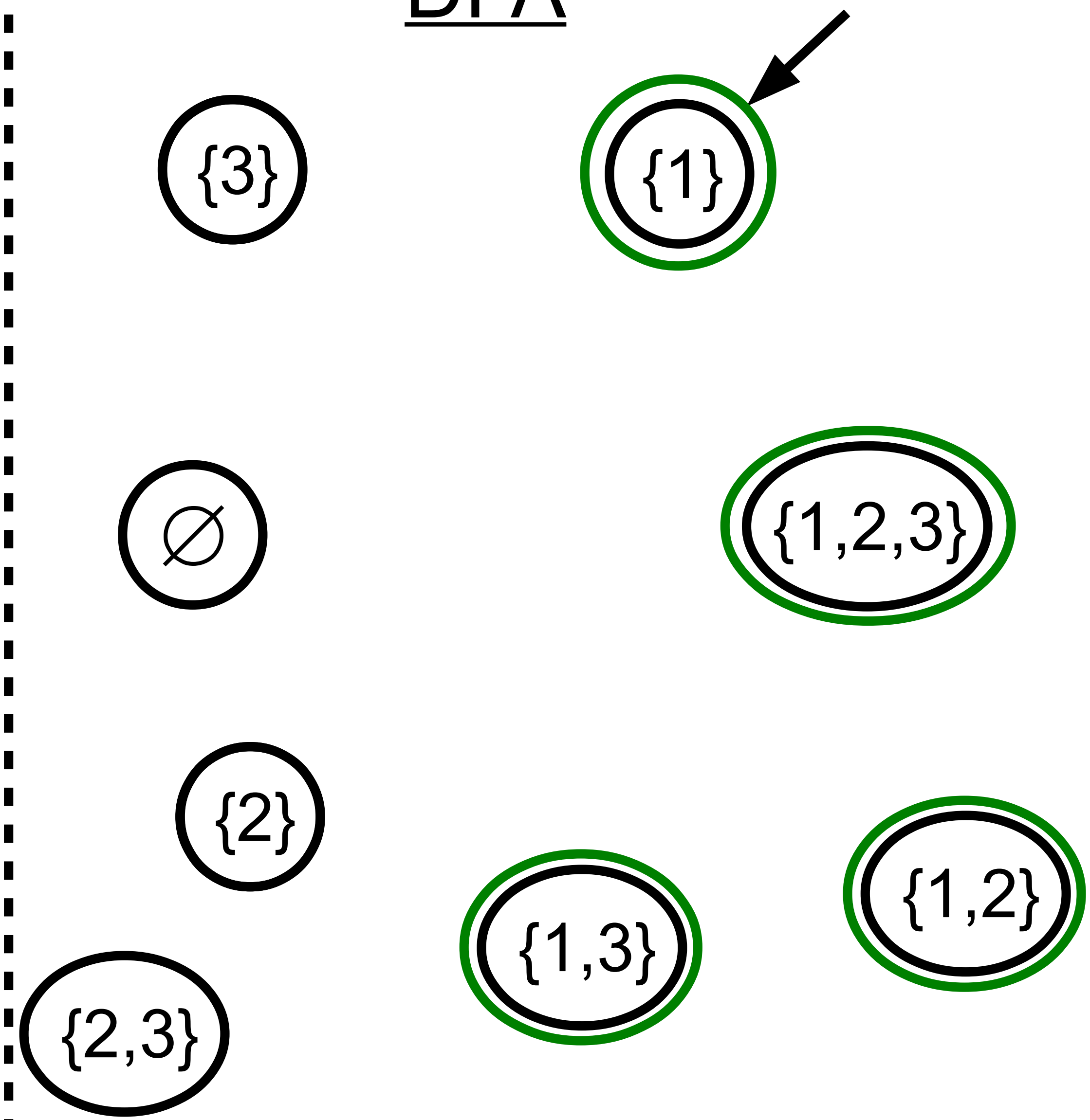


ANOTHER Example: NFA → DFA conversion

NFA

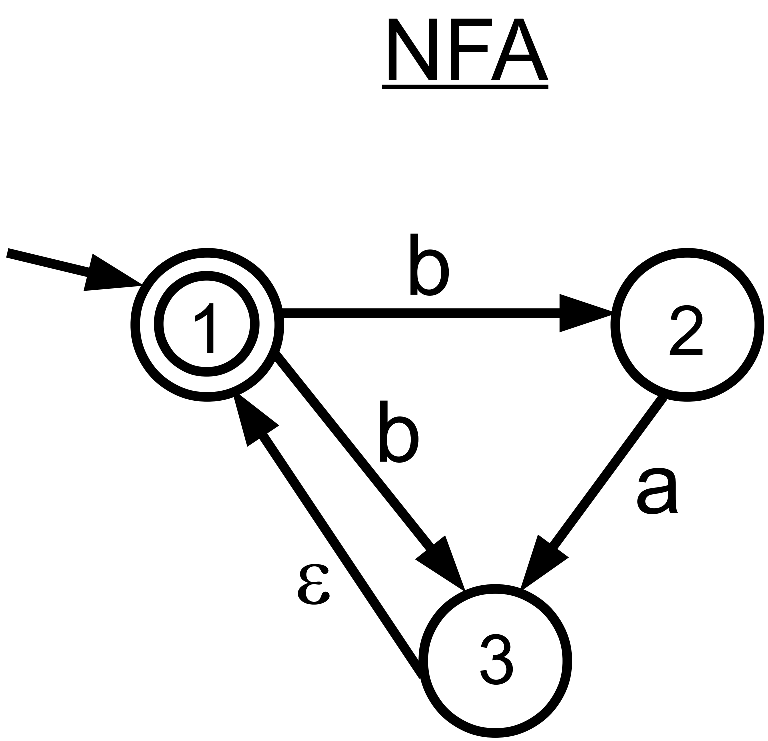


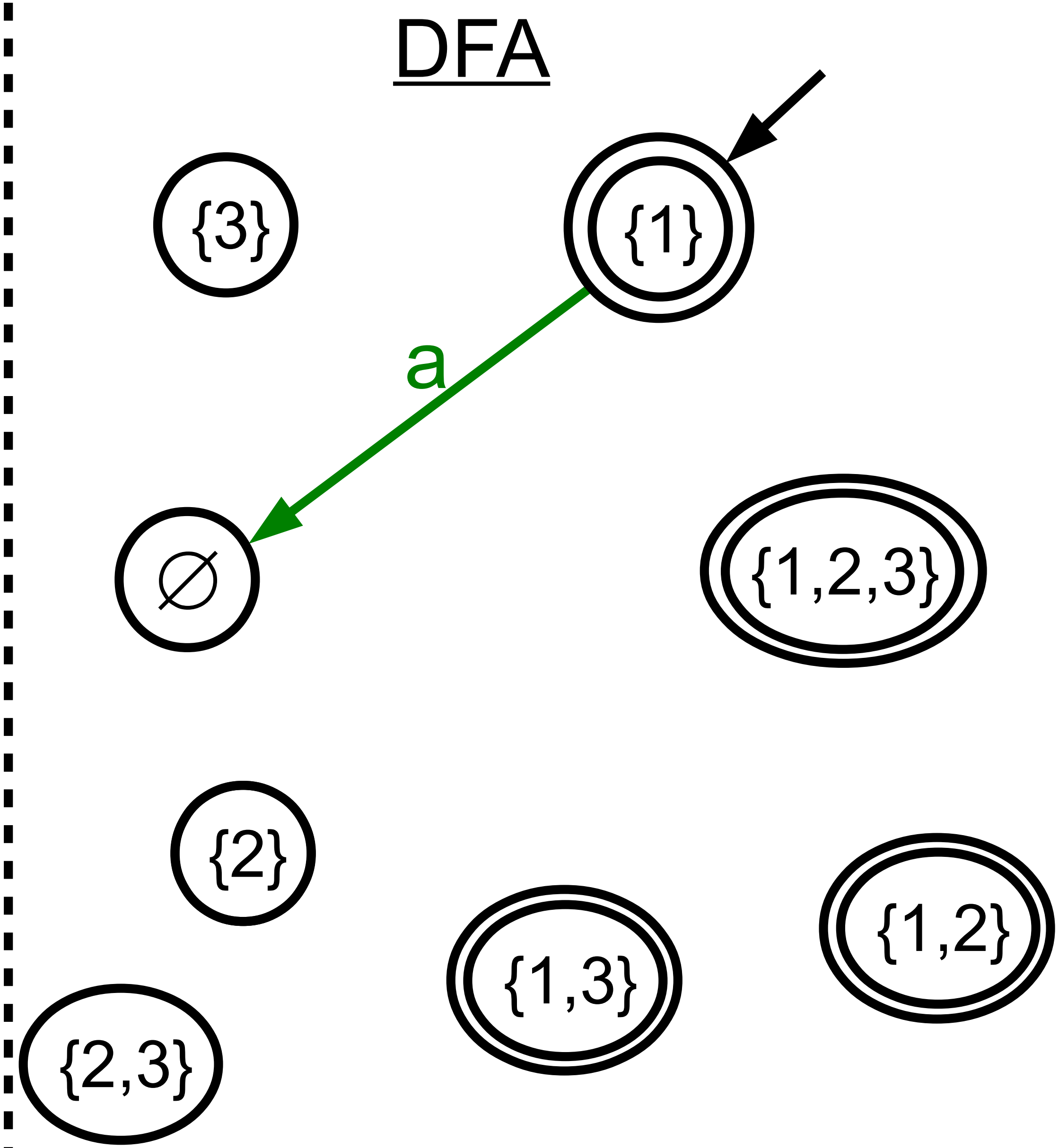
DFA



F_{DFA} = {S : S contains an element of F_{NFA}}

ANOTHER Example: NFA → DFA conversion

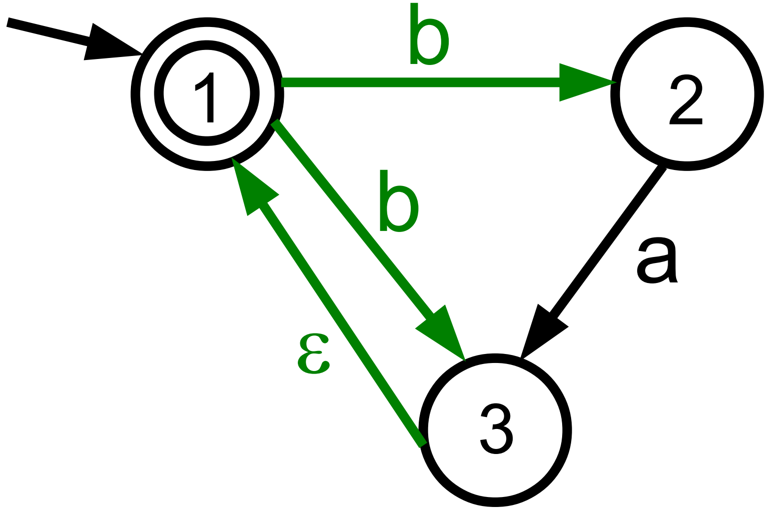




$$\delta_{DFA}(\{1\}, a) = E(\delta_{NFA}(1, a)) = E(\varnothing) = \varnothing$$

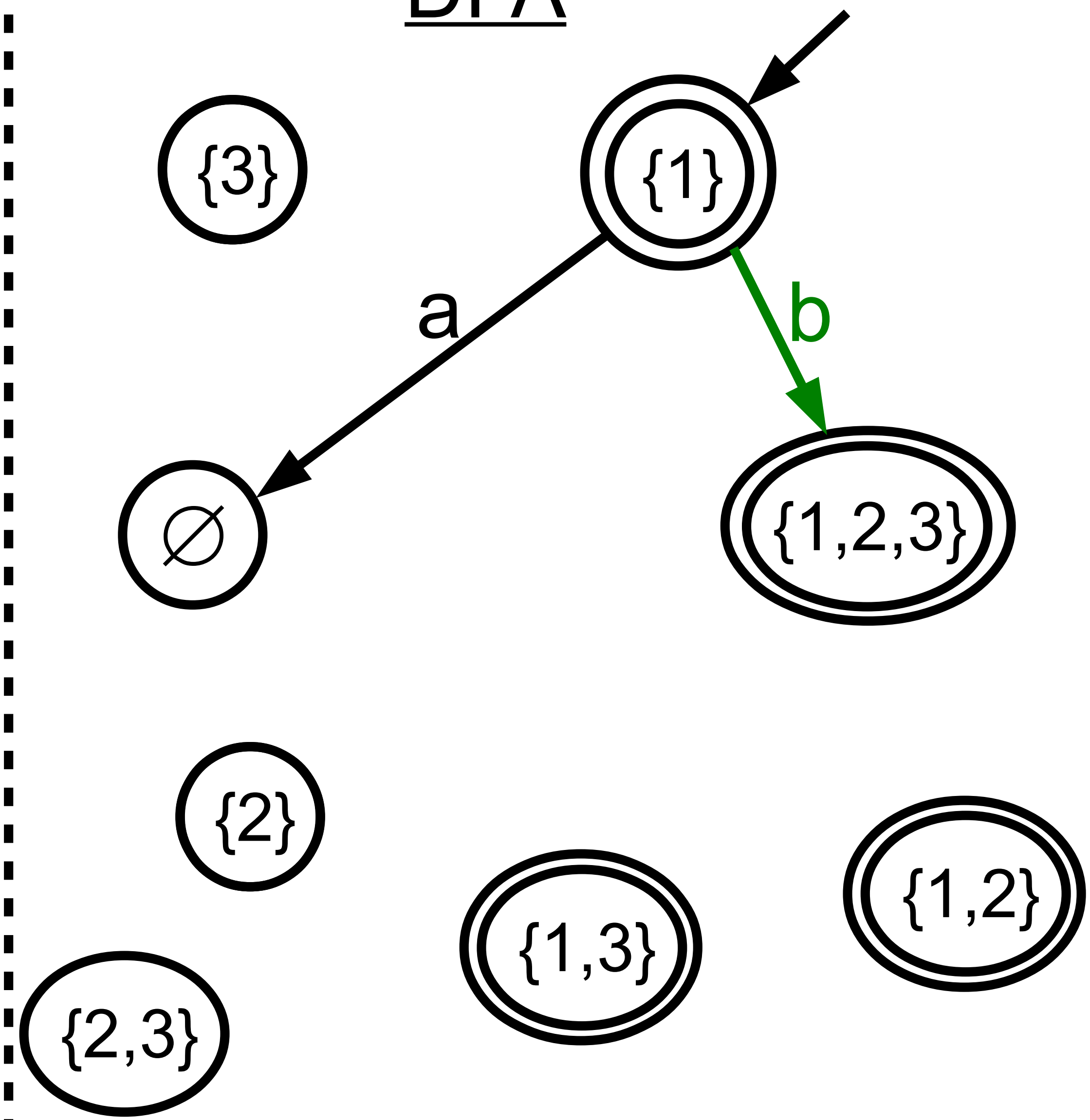
ANOTHER Example: NFA → DFA conversion

NFA



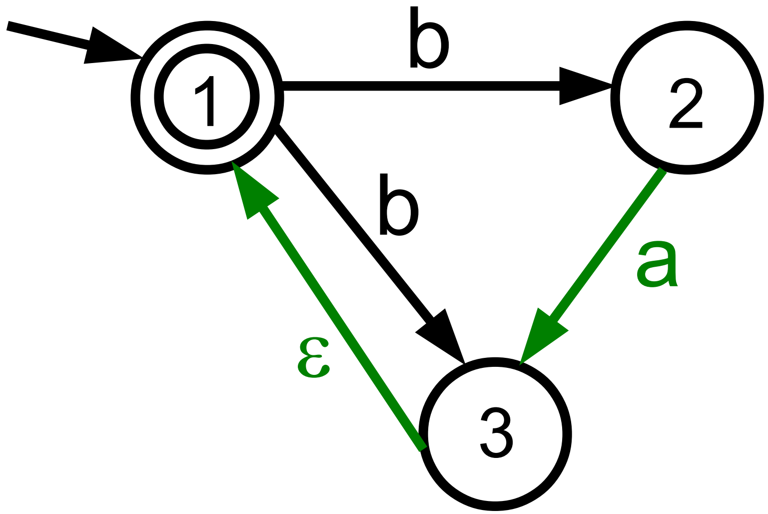
$$\delta_{DFA}(\{1\}, b)$$
$$= E(\delta_{NFA}(1, b))$$
$$= E(\{2,3\}) = \{1,2,3\}$$

DFA

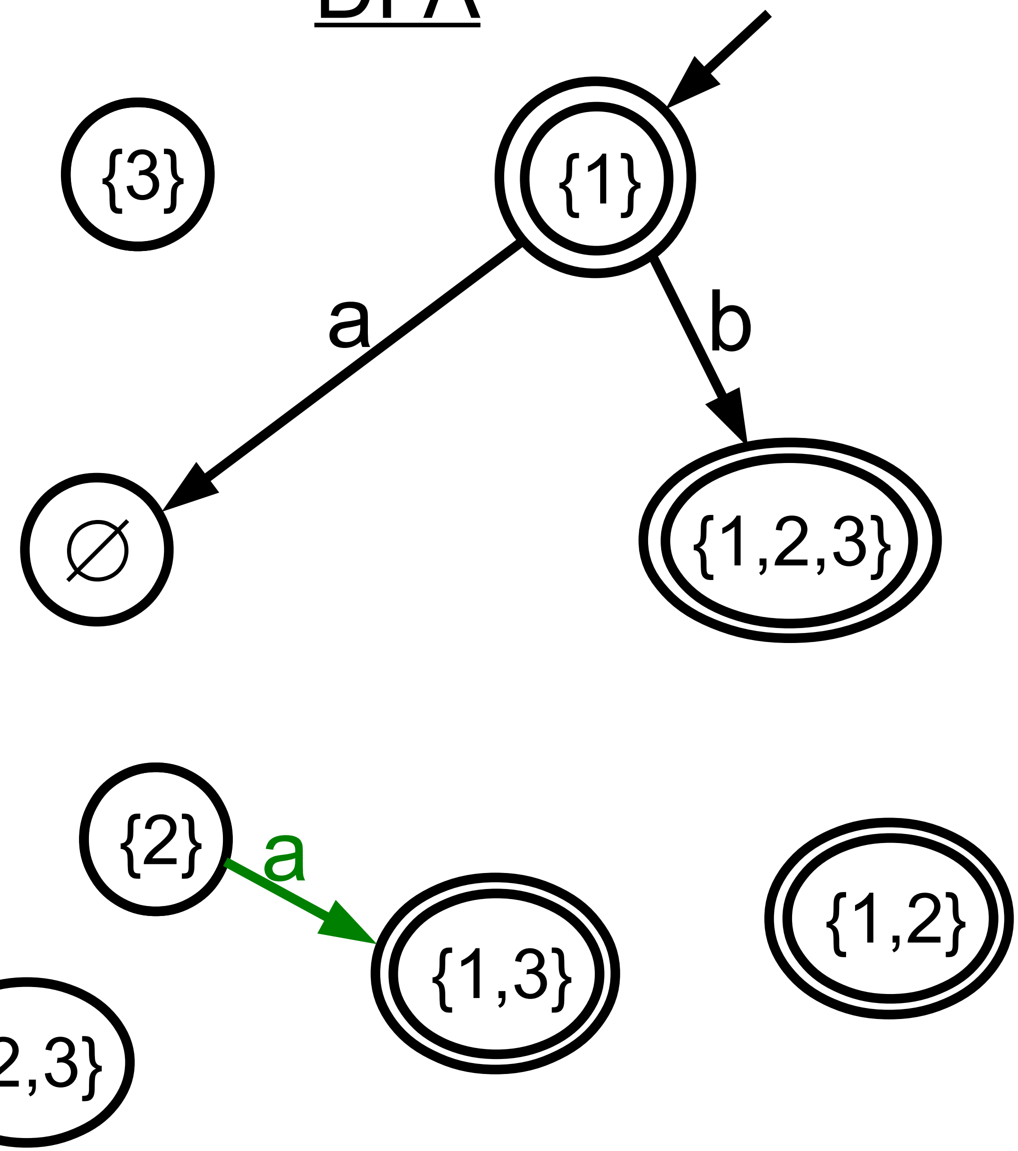


ANOTHER Example: NFA → DFA conversion

NFA

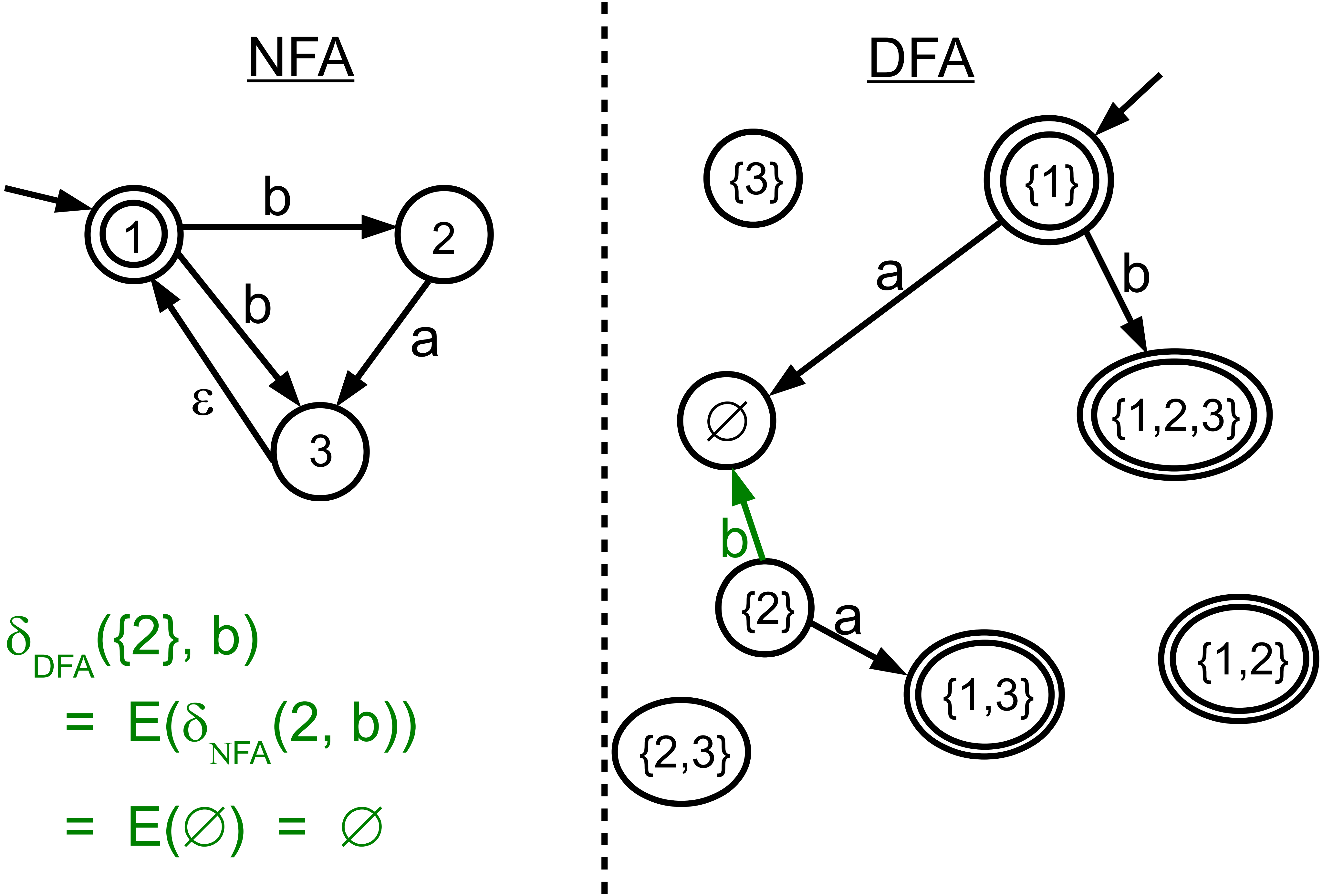


DFA



$$\delta_{DFA}(\{2\}, a) = E(\delta_{NFA}(2, a)) = E(\{3\}) = \{1,3\}$$

ANOTHER Example: NFA → DFA conversion



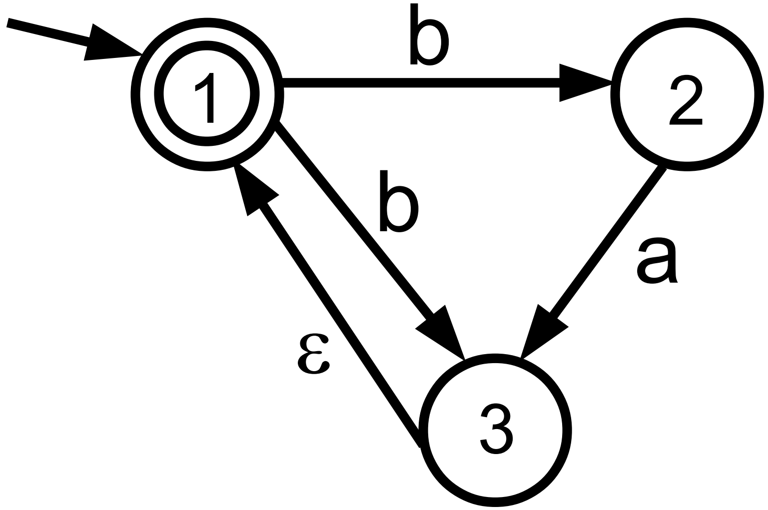
$\delta_{DFA}(\{2\}, b)$

$= E(\delta_{NFA}(2, b))$

$= E(\varnothing) = \varnothing$

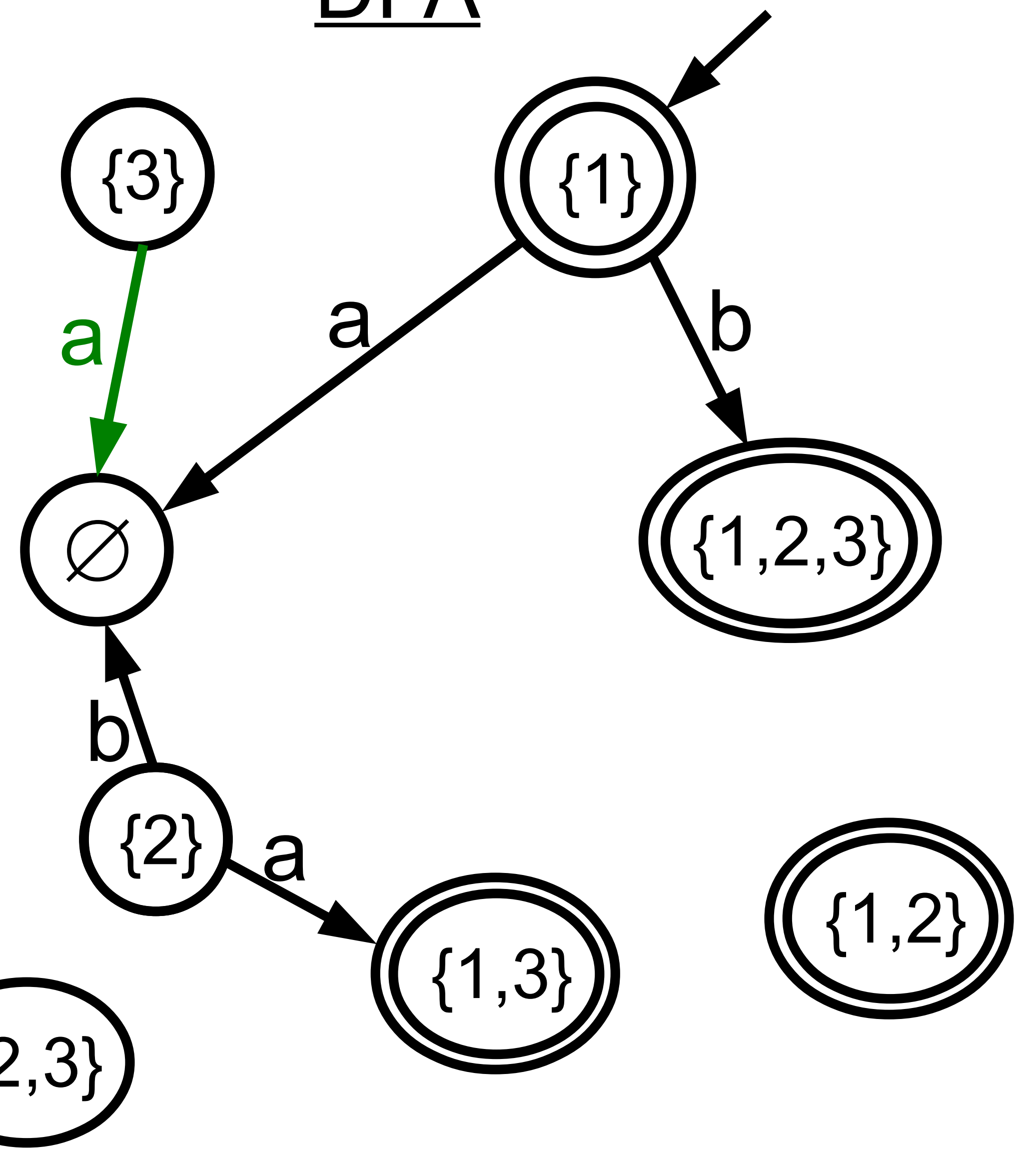
ANOTHER Example: NFA → DFA conversion

NFA

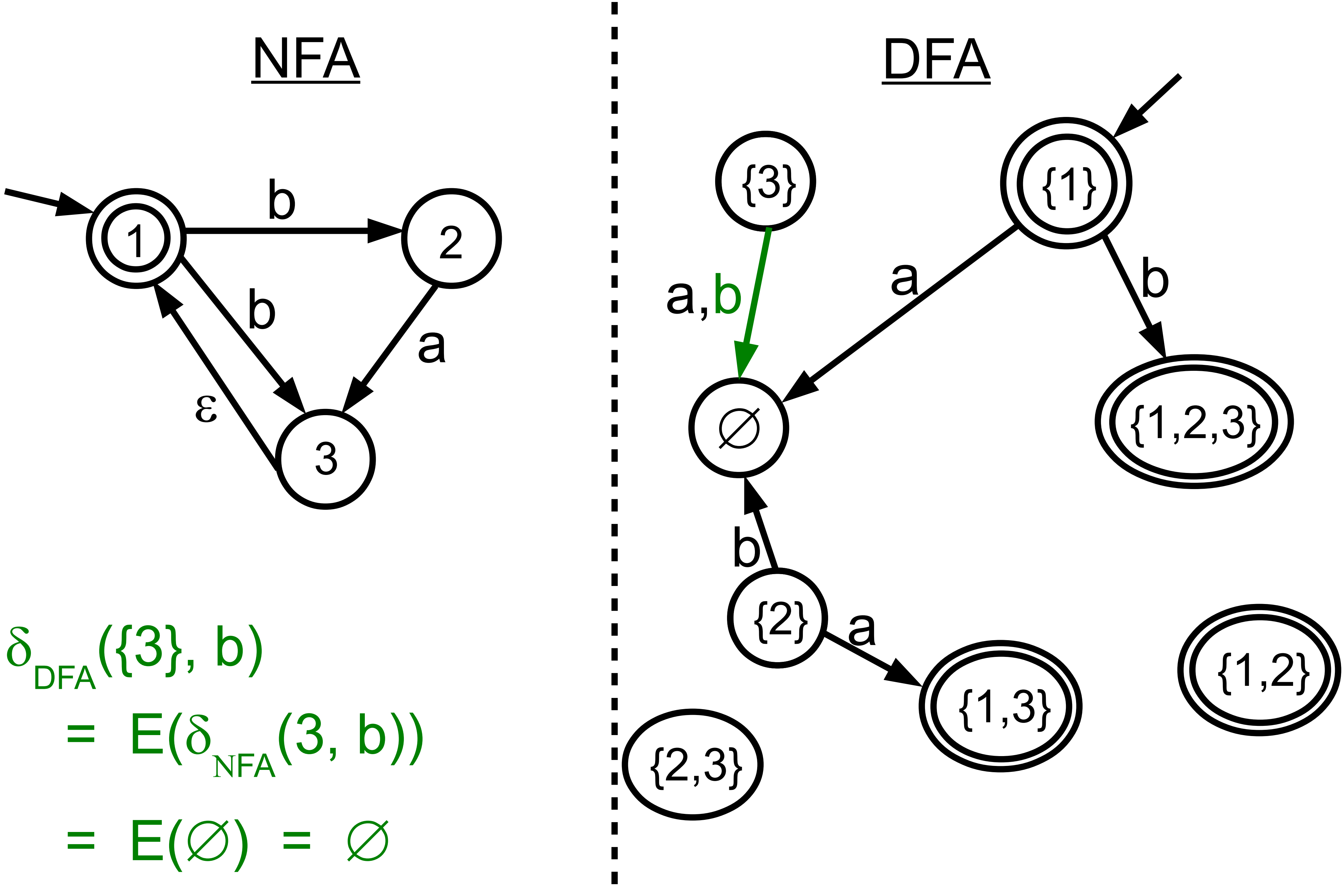


$$\delta_{DFA}(\{3\}, a)$$
$$= E(\delta_{NFA}(3, a))$$
$$= E(\varnothing) = \varnothing$$

DFA



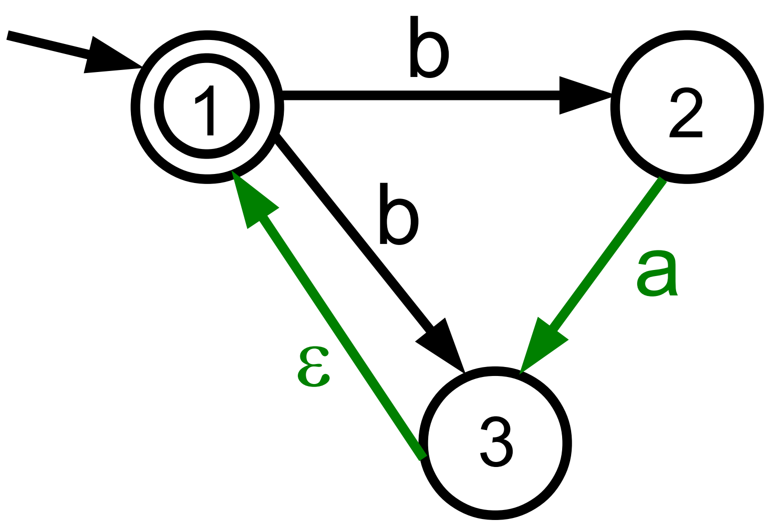
ANOTHER Example: NFA → DFA conversion



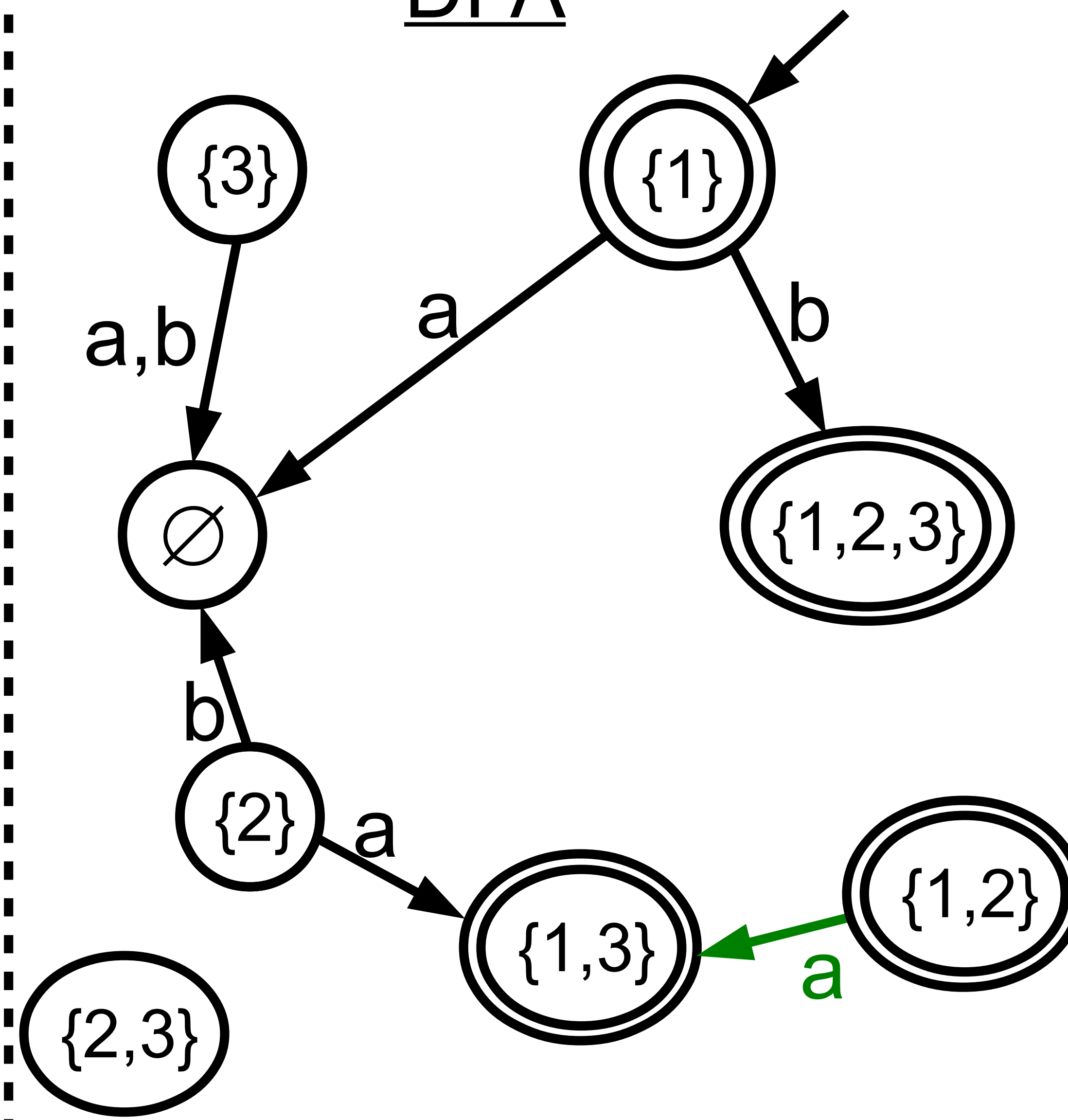
$$\delta_{DFA}(\{3\}, b)$$
$$= E(\delta_{NFA}(3, b))$$
$$= E(\varnothing) = \varnothing$$

ANOTHER Example: NFA → DFA conversion

NFA



DFA

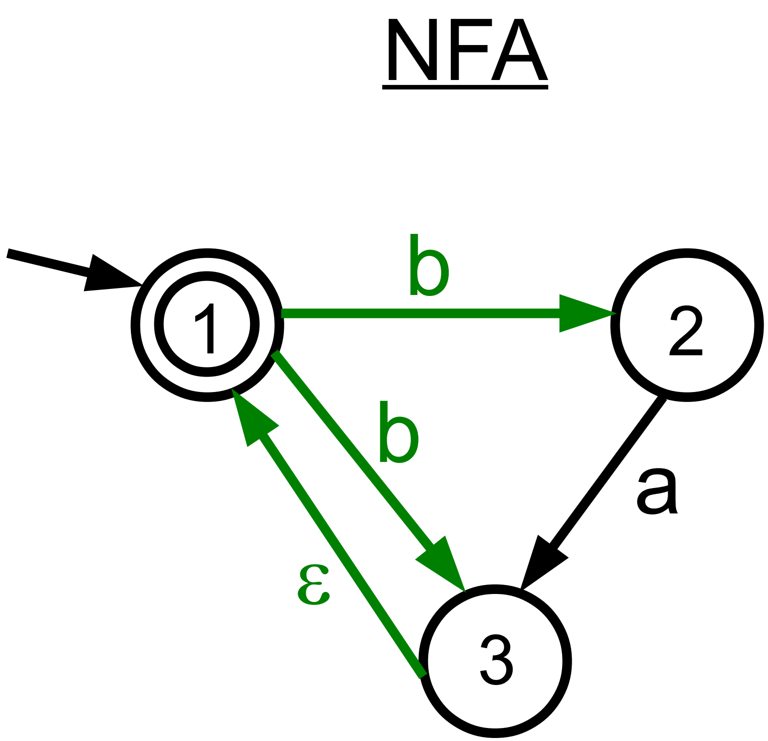


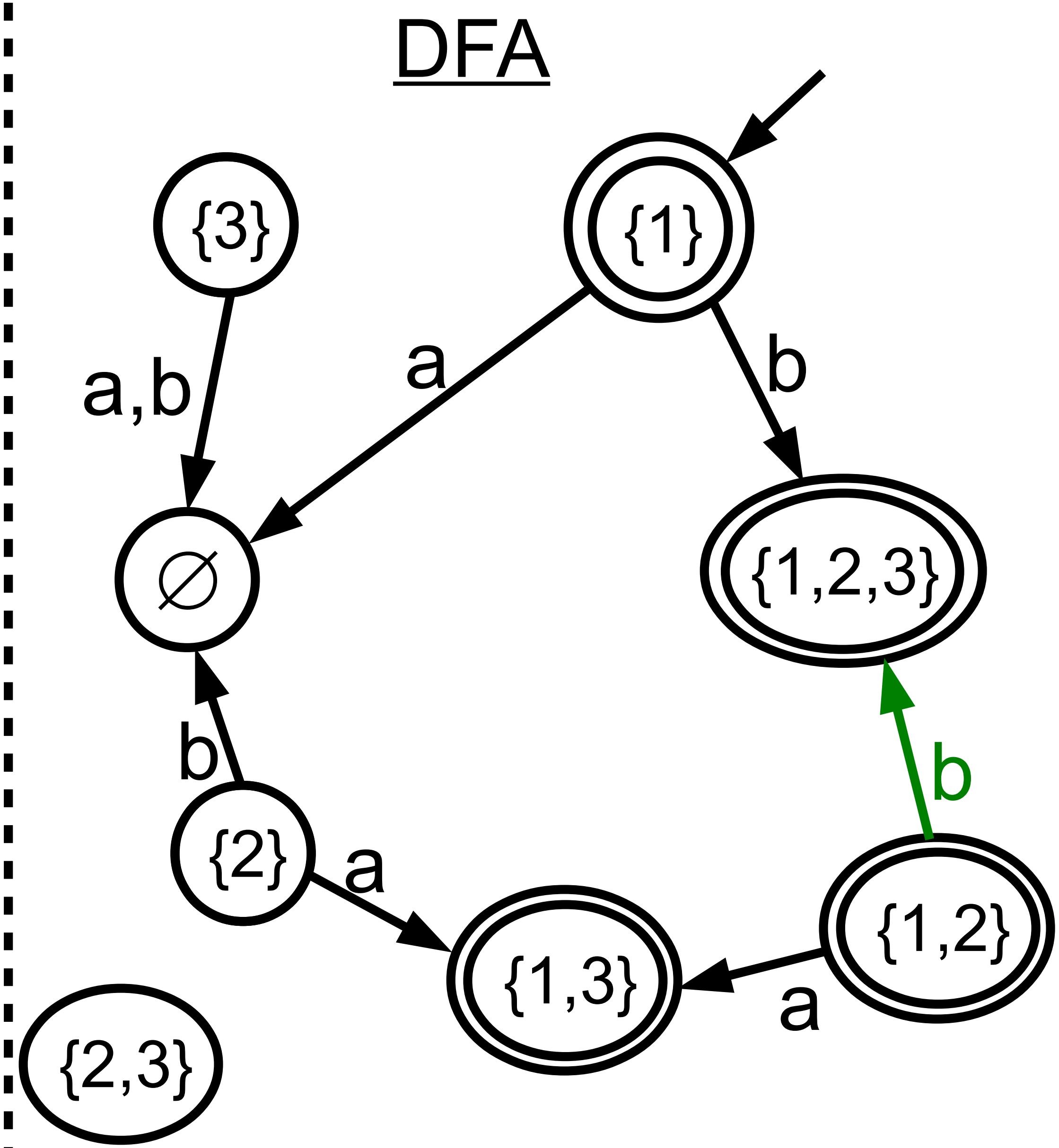
$\delta_{DFA}(\{1,2\}, a)$

$= E(\delta_{NFA}(1,a) \cup \delta_{NFA}(2,a))$

$= E(\varnothing \cup \{3\}) = \{1,3\}$

ANOTHER Example: NFA → DFA conversion

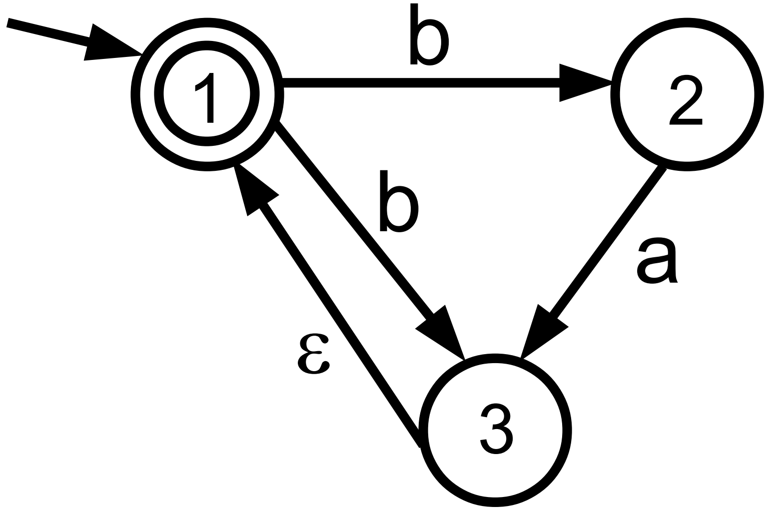




$$\delta_{DFA}(\{1,2\}, b)$$
$$= E(\delta_{NFA}(1,b) \cup \delta_{NFA}(2,b))$$
$$= E(\{2,3\} \cup \emptyset) = \{1,2,3\}$$

ANOTHER Example: NFA → DFA conversion

NFA

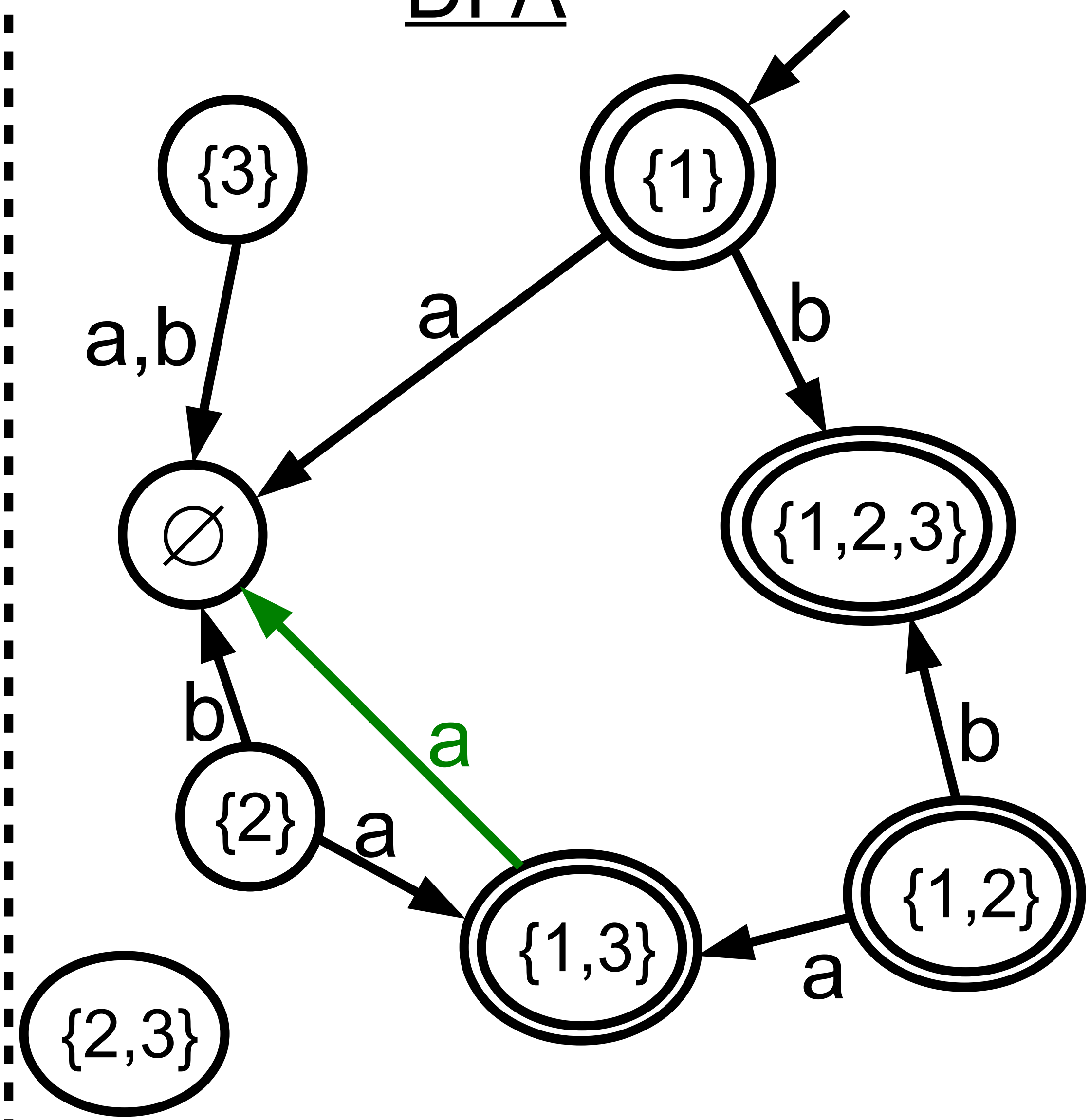


$\delta_{DFA}(\{1,3\}, a)$

$= E(\delta_{NFA}(1,a) \cup \delta_{NFA}(3,a))$

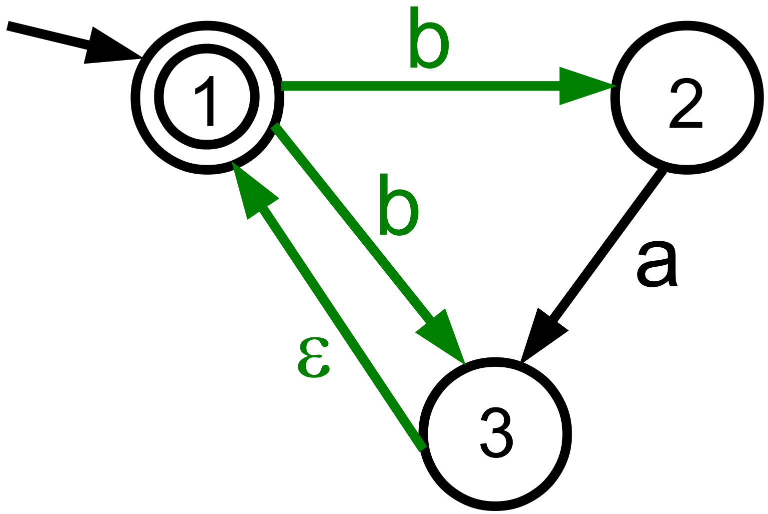
$= E(\varnothing \cup \varnothing) = \varnothing$

DFA

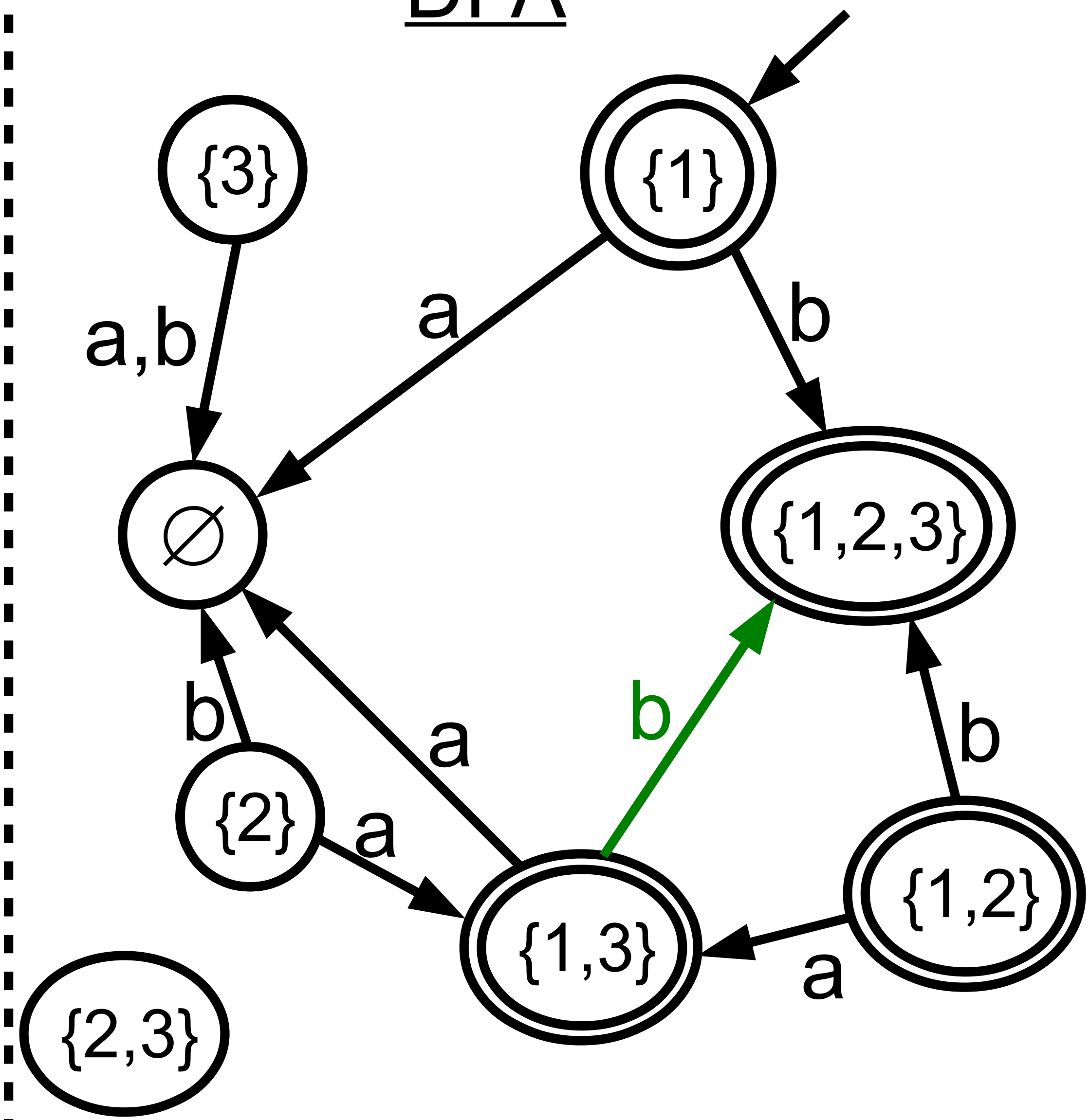


ANOTHER Example: NFA → DFA conversion

NFA



DFA



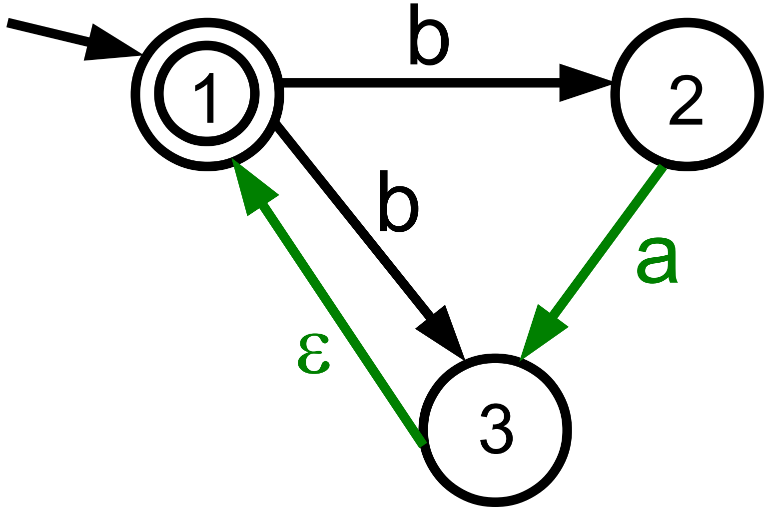
$$\delta_{DFA}(\{1,3\}, b)$$

$$= E(\delta_{NFA}(1,b) \cup \delta_{NFA}(3,b))$$

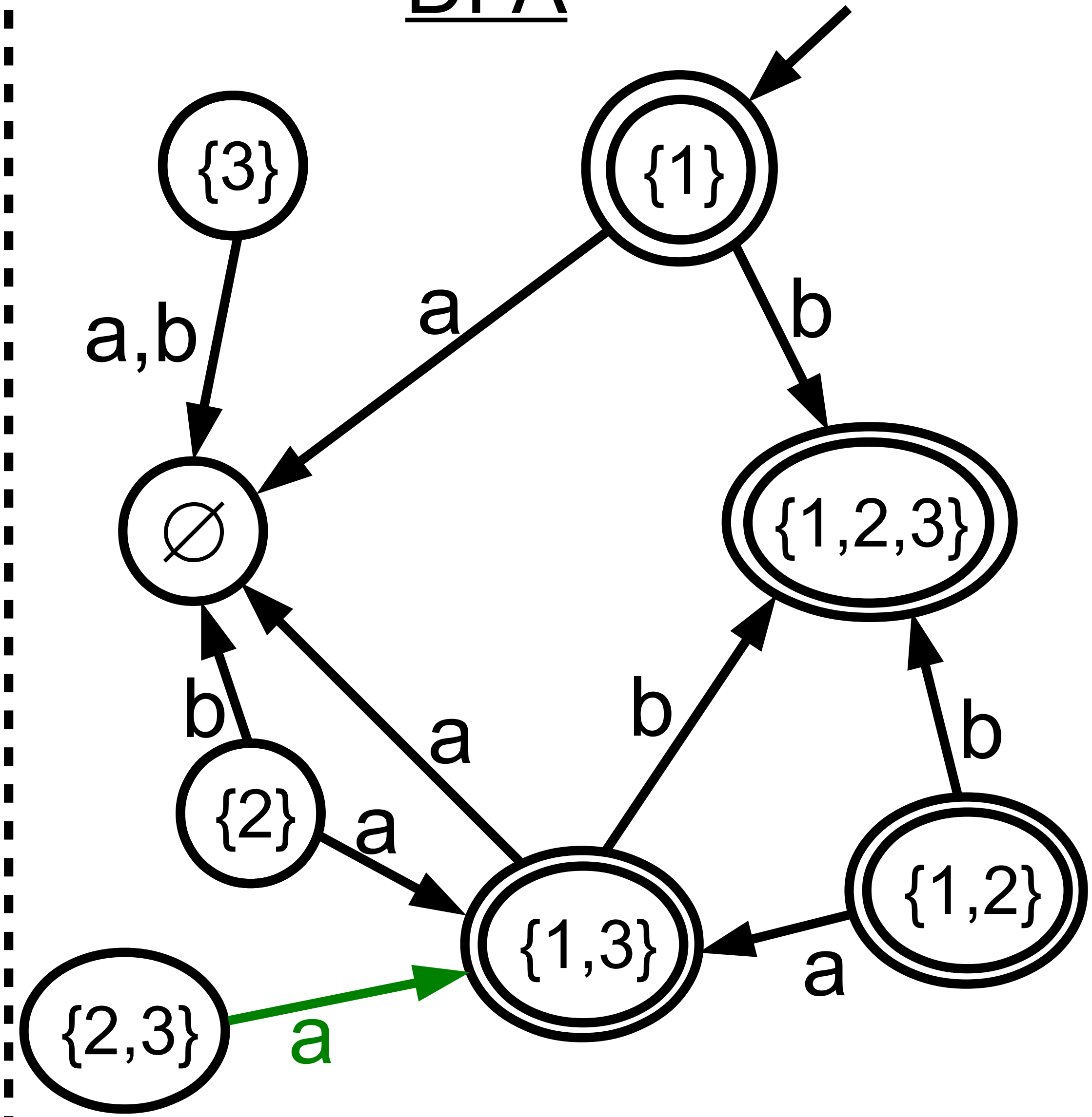
$$= E(\{2,3\} \cup \emptyset) = \{1,2,3\}$$

ANOTHER Example: NFA → DFA conversion

NFA



DFA



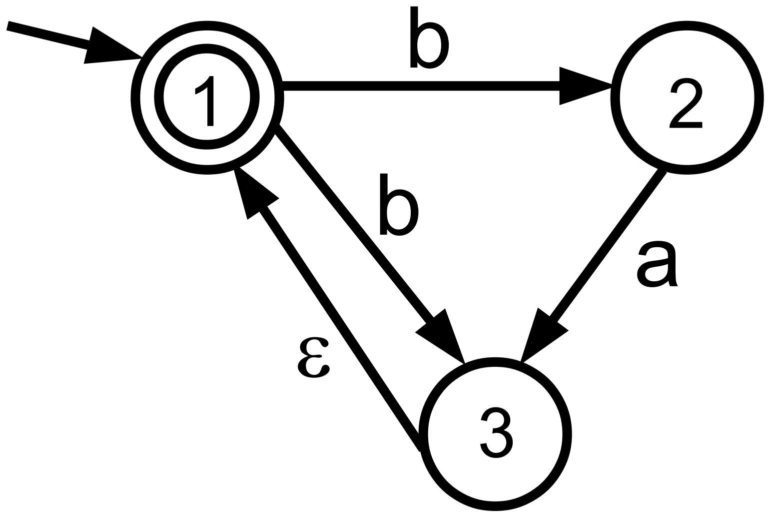
$\delta_{DFA}(\{2,3\}, a)$

$= E(\delta_{NFA}(2,a) \cup \delta_{NFA}(3,a))$

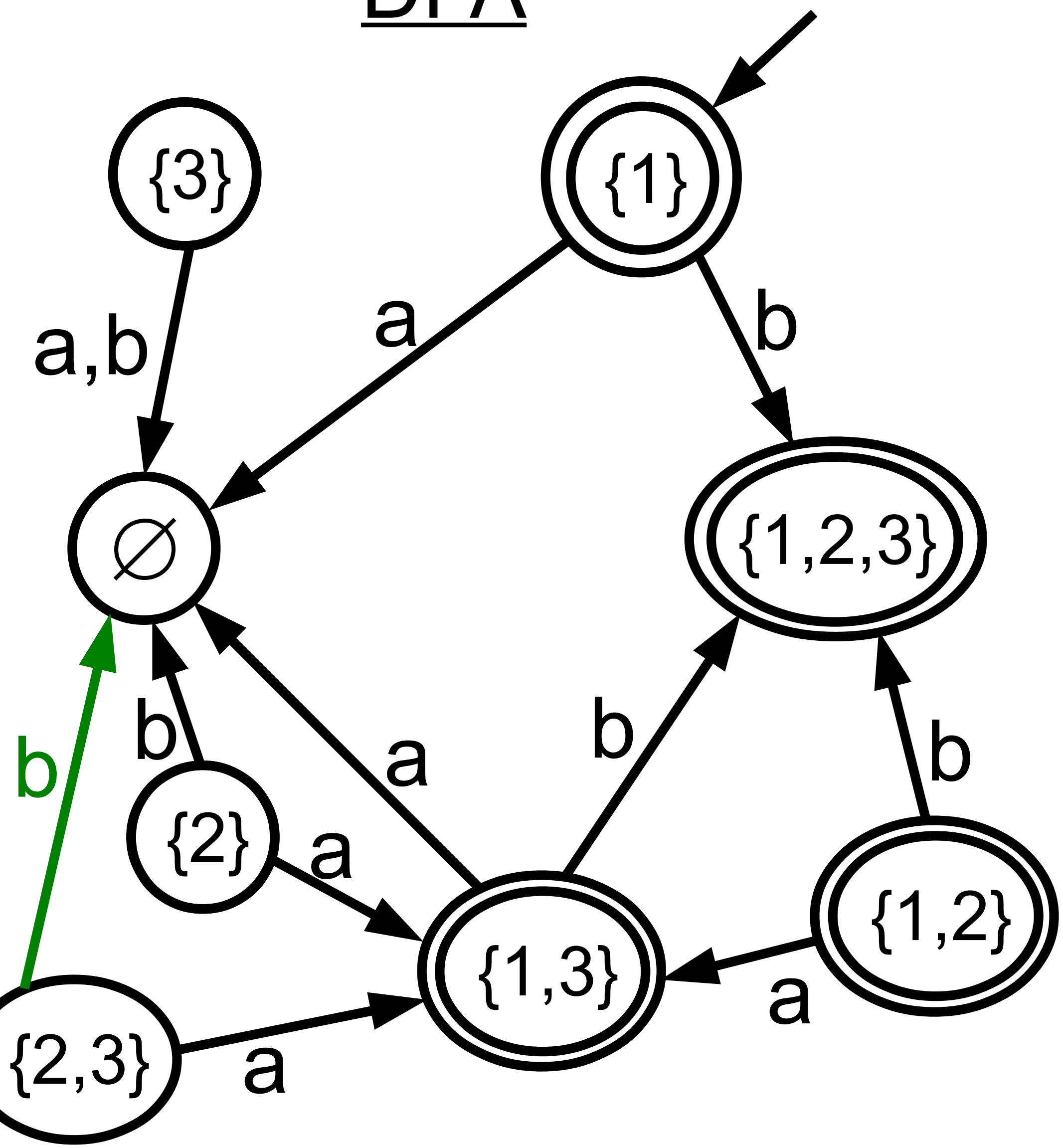
$= E(\{3\} \cup \emptyset) = \{1,3\}$

ANOTHER Example: NFA → DFA conversion

NFA



DFA



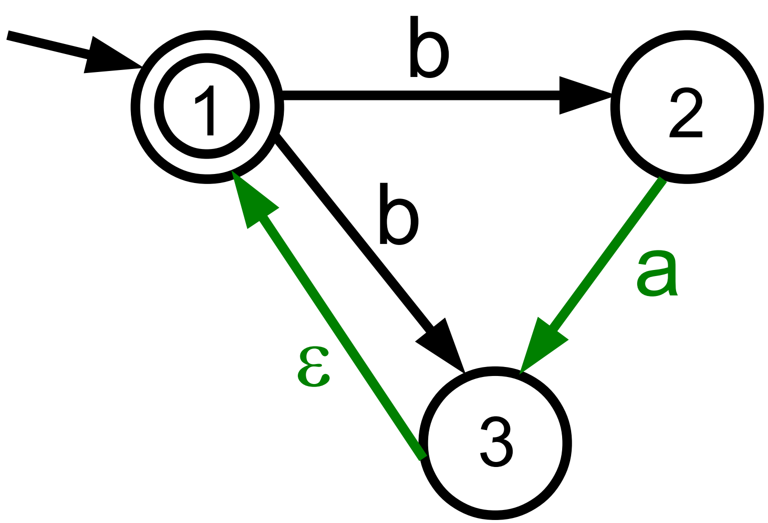
$$\delta_{DFA}(\{2,3\}, b)$$

$$= E(\delta_{NFA}(2,b) \cup \delta_{NFA}(3,b))$$

$$= E(\varnothing \cup \varnothing) = \varnothing$$

ANOTHER Example: NFA → DFA conversion

NFA

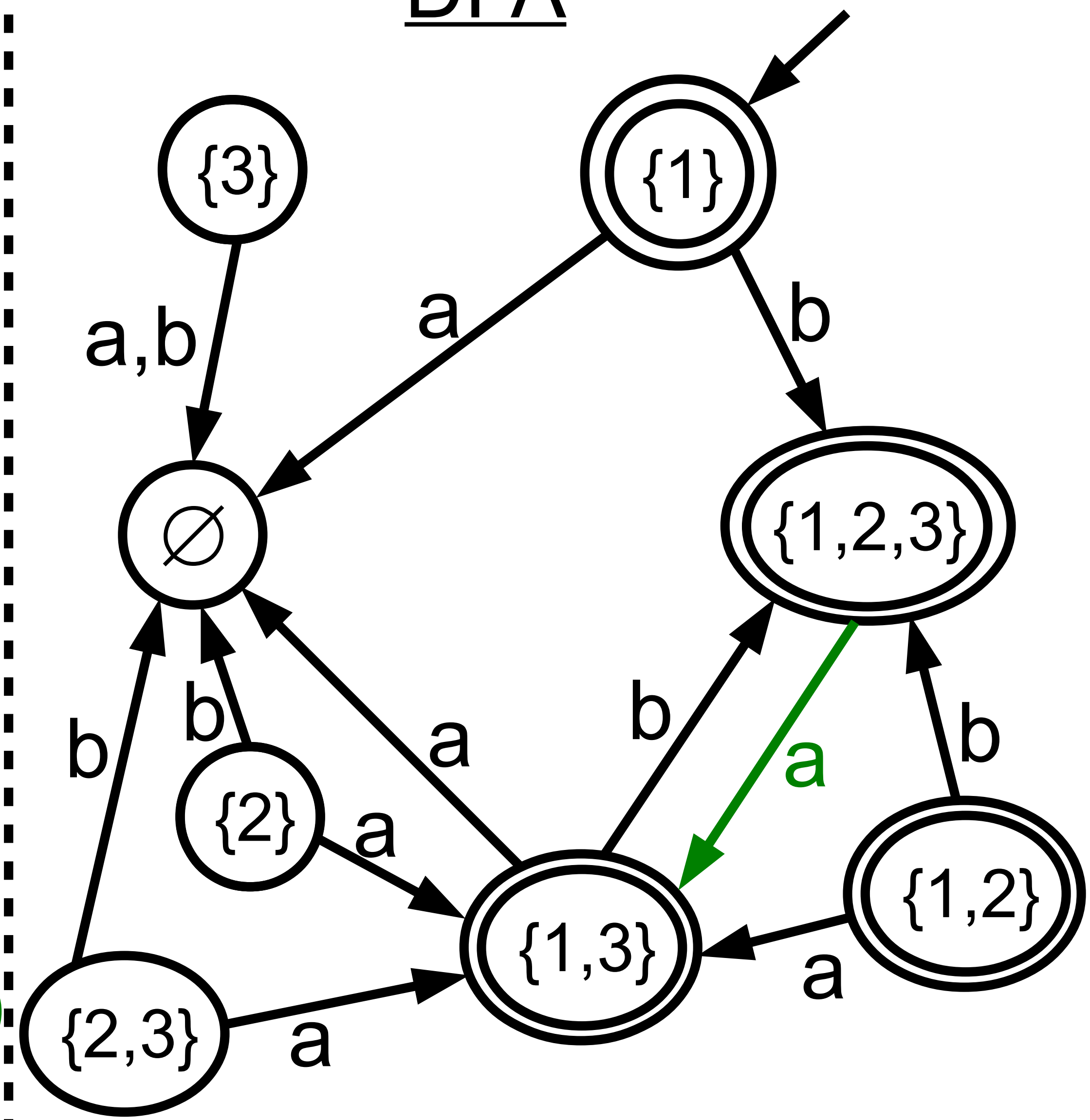


$$\delta_{DFA}(\{1,2,3\}, a)$$

$$=E(\delta_{NFA}(1,a) \cup \delta_{NFA}(2,a) \cup \delta_{NFA}(3,a))$$

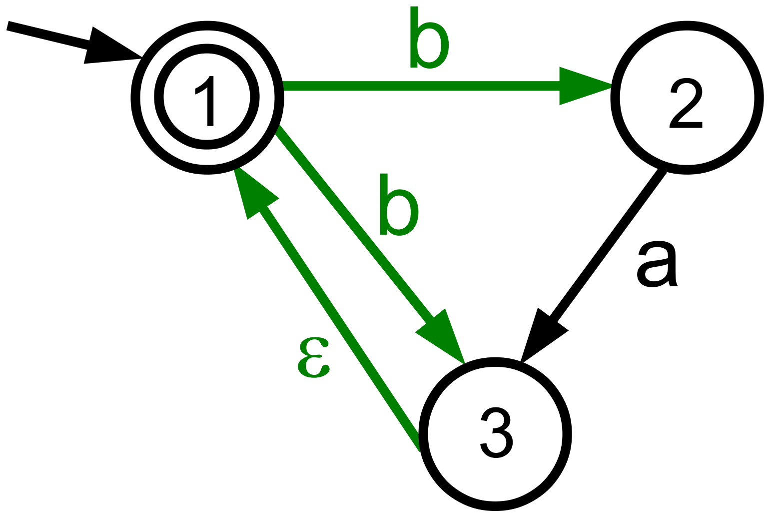
$$=E(\emptyset \cup \{3\} \cup \emptyset) = \{1,3\}$$

DFA

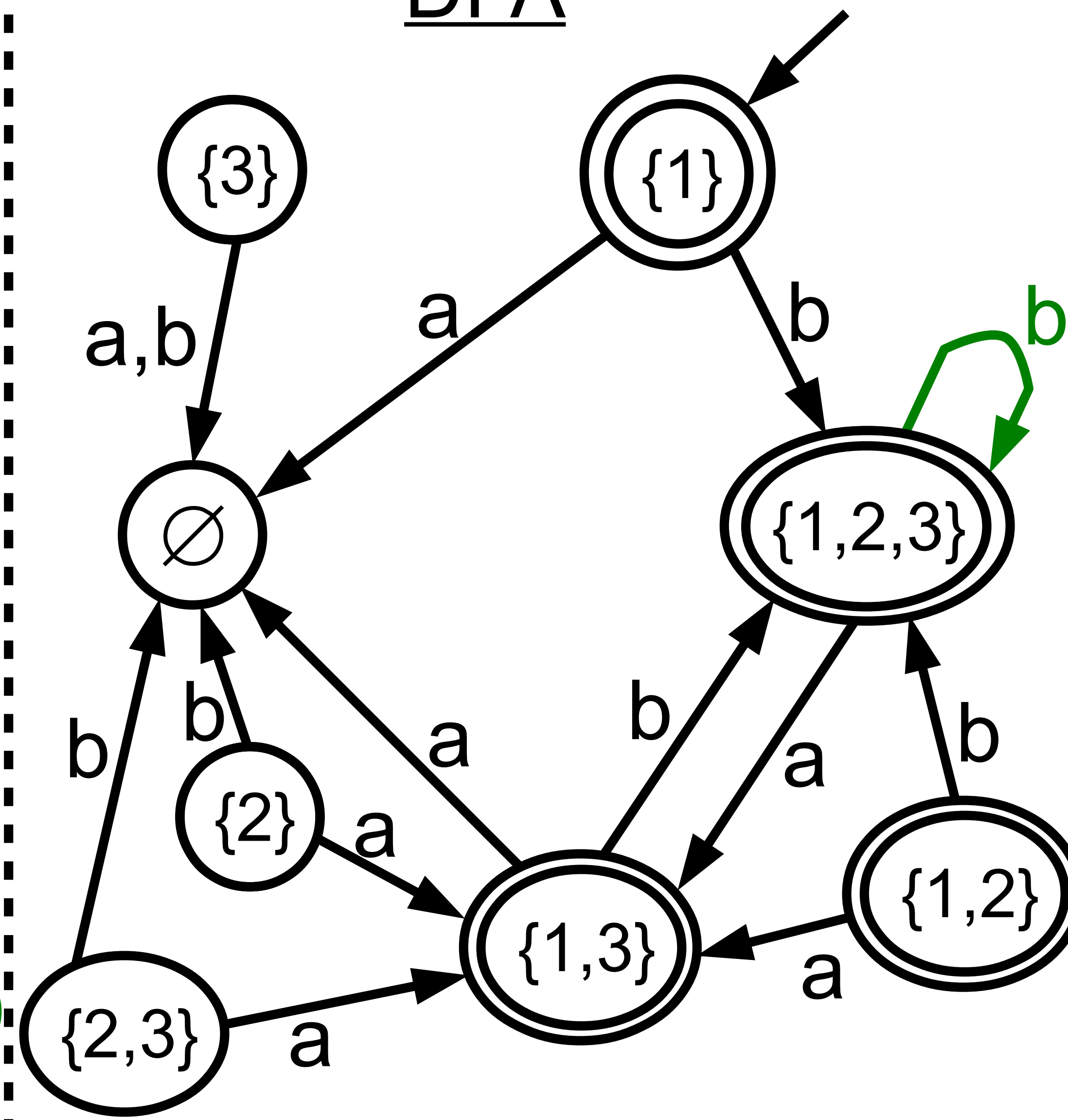


ANOTHER Example: NFA → DFA conversion

NFA



DFA



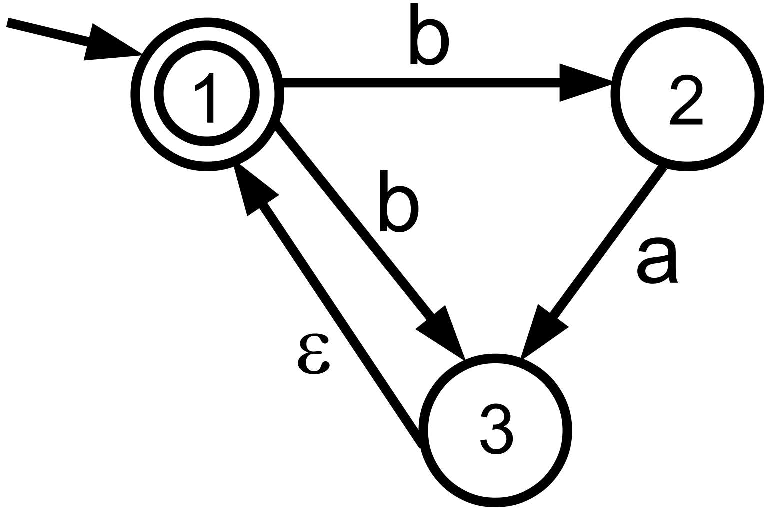
$$\delta_{DFA}(\{1,2,3\}, b)$$

$$=E(\delta_{NFA}(1,b) \cup \delta_{NFA}(2,b) \cup \delta_{NFA}(3,b))$$

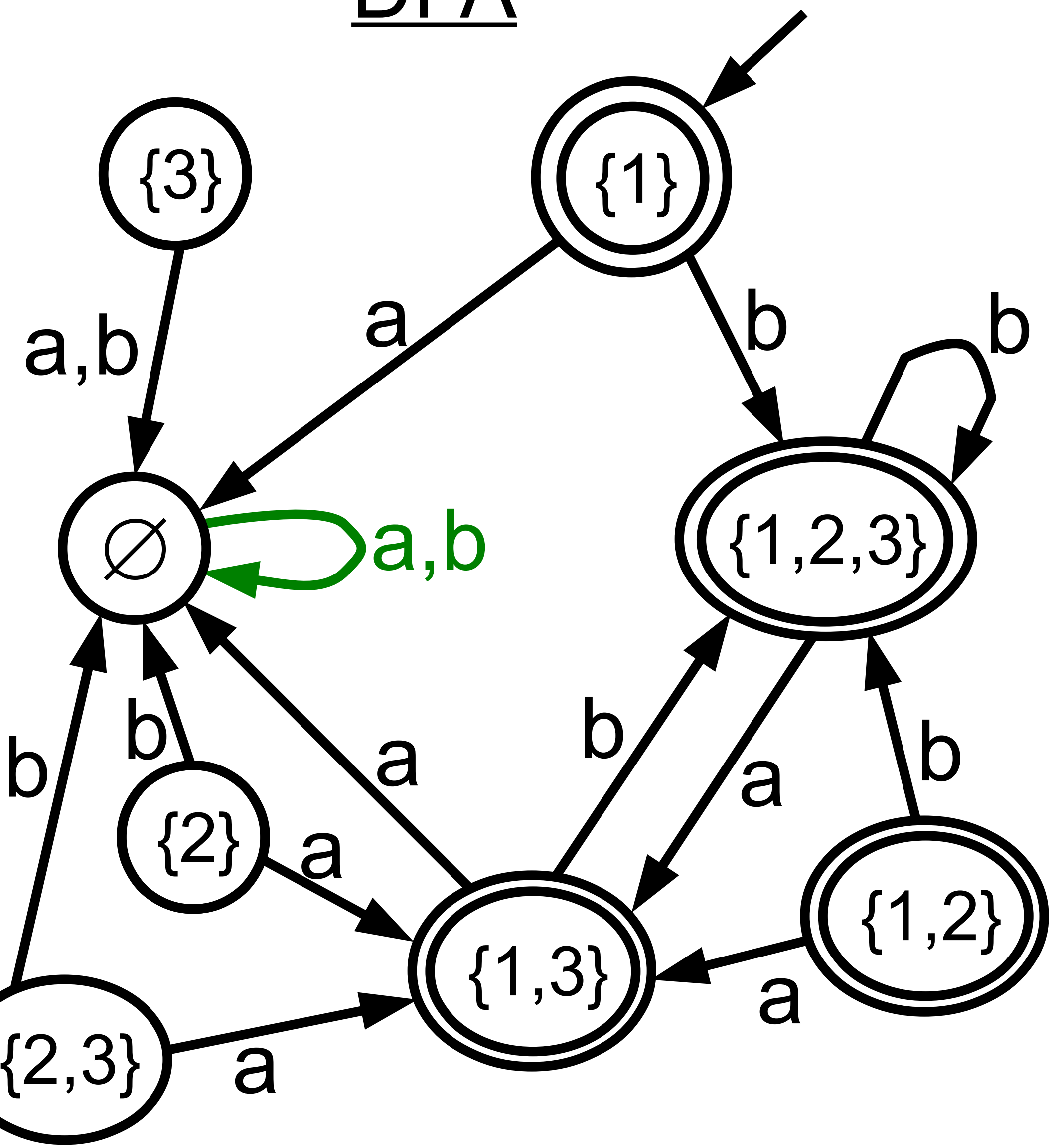
$$=E(\{2,3\} \cup \varnothing \cup \varnothing) = \{1,2,3\}$$

ANOTHER Example: NFA → DFA conversion

NFA



DFA

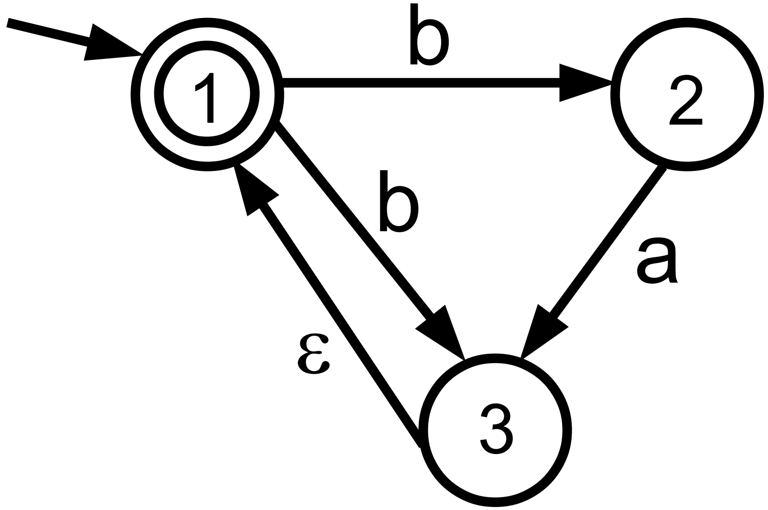


$$\delta_{DFA}(\varnothing, a) = \varnothing$$

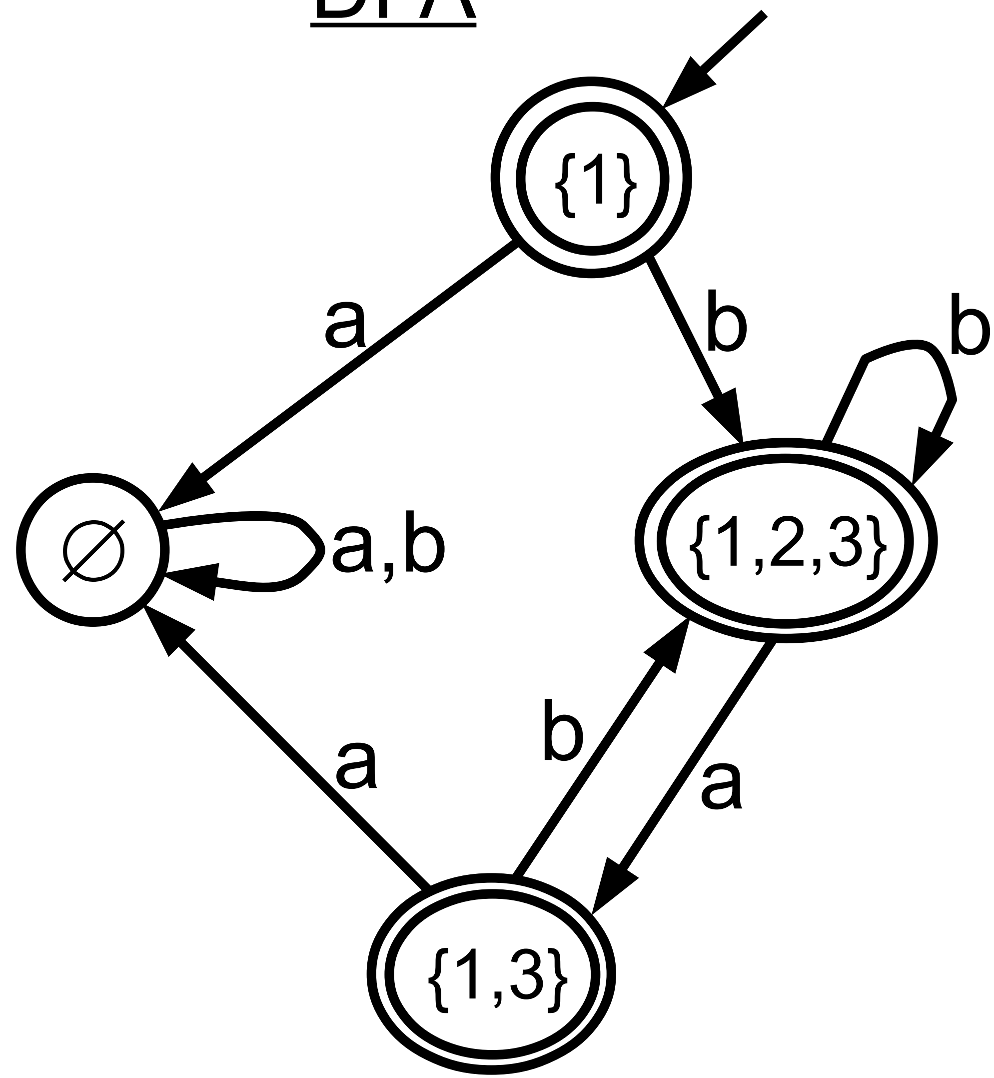
$$\delta_{DFA}(\varnothing, b) = \varnothing$$

ANOTHER Example: NFA → DFA conversion

NFA



DFA



We can delete the unreachable states.

Summary: NFA and DFA recognize the same languages

We now return to the question:

- Suppose A, B are regular languages, what about
- not A := { w : w is not in A } REGULAR
- A U B := { w : w in A or w in B } REGULAR
- A o B := { w_1 w_2 : w_1 in A and w_2 in B }
- A* := { w_1 w_2 … w_k : $k \geq 0$, w_i in A for every i }

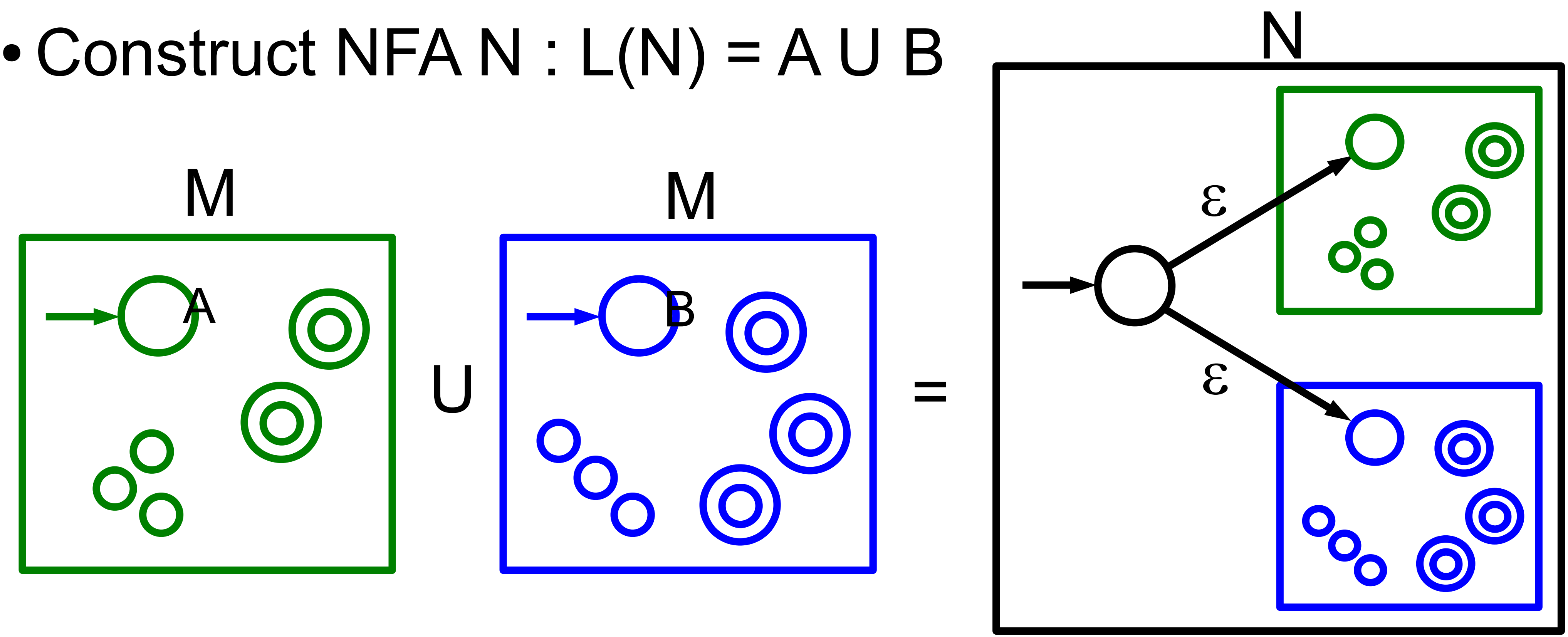
Theorem: If A, B are regular languages, then so is

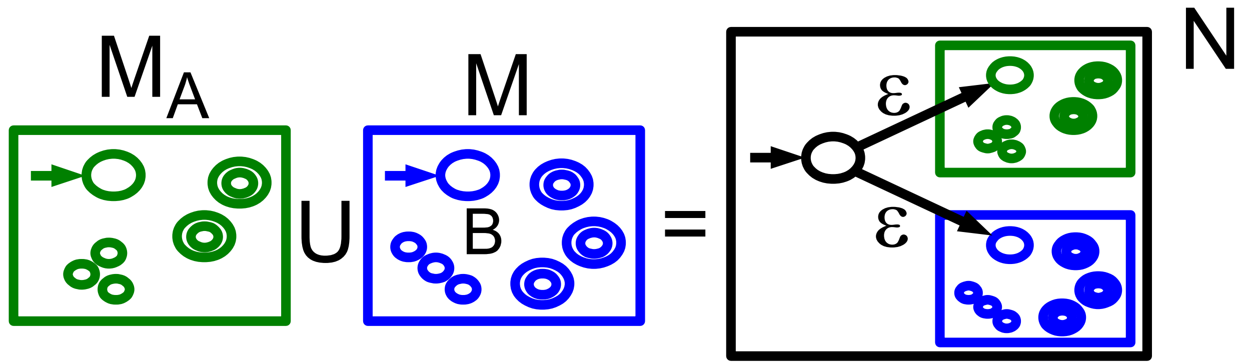
A U B := { w : w in A or w in B }

- Proof idea: Given DFA M_A : $L(M_A) = A$,

  DFA M_B : $L(M_B) = B$,

- Construct NFA N : L(N) = A U B





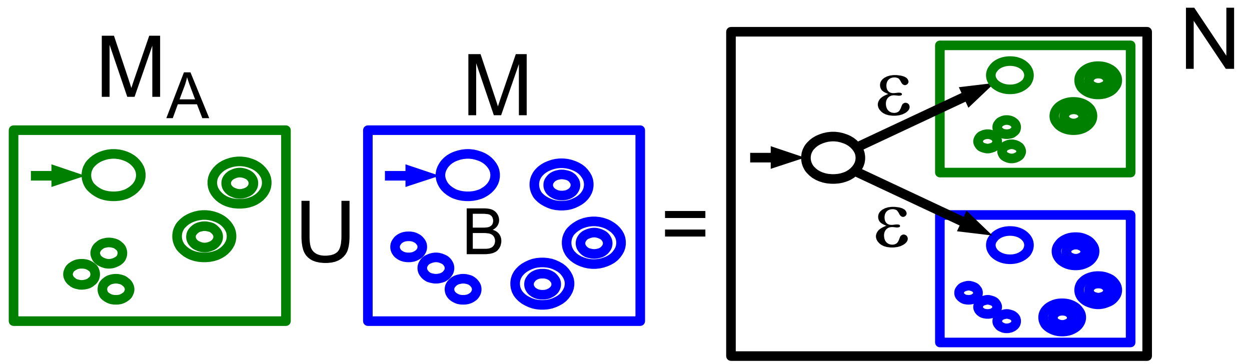
Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A) : L(M_A) = A$,

  DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B) : L(M_B) = B$,

- Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:

- $Q :=$?

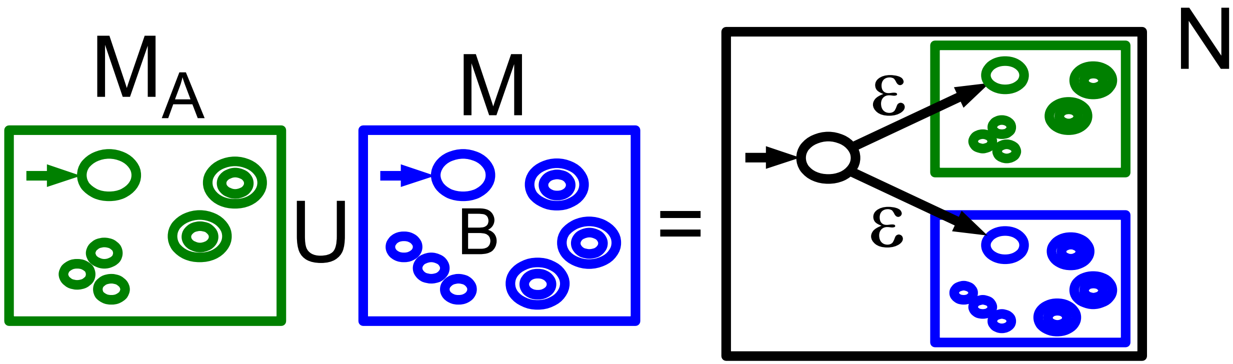


Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A) : L(M_A) = A$,

  DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B) : L(M_B) = B$,

- Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:
- $Q := \{q\} \cup Q_A \cup Q_B$, $F :=$?

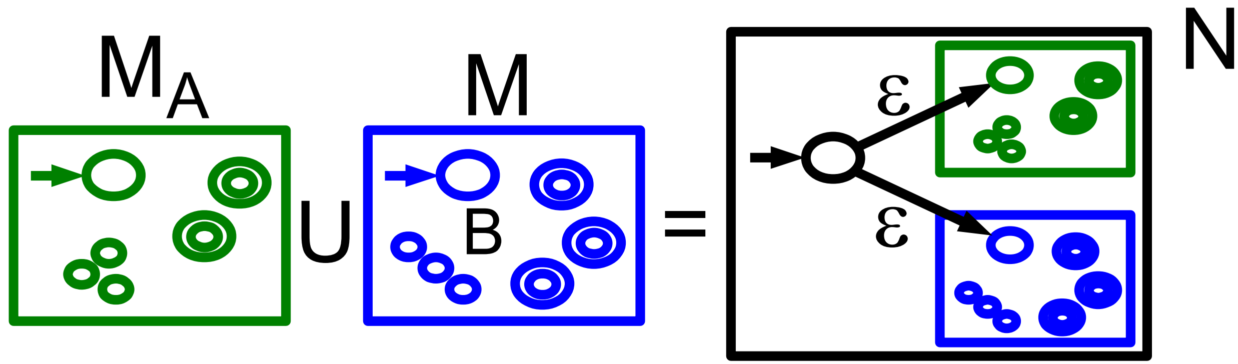


Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A) : L(M_A) = A$,

  DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B) : L(M_B) = B$,

- Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:
- $Q := \{q\} \cup Q_A \cup Q_B$, $F := F_A \cup F_B$
- $\delta(r,x) := \{ \delta_A(r,x) \}$ if r in Q_A and $x \neq \varepsilon$
- $\delta(r,x) :=$? if r in Q_B and $x \neq \varepsilon$

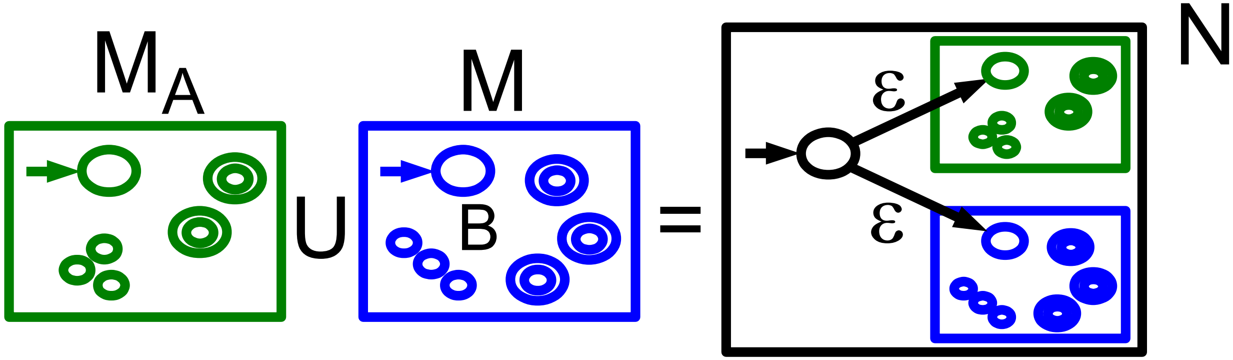


Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A) : L(M_A) = A$,

  DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B) : L(M_B) = B$,

- Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:
- $Q := \{q\} \cup Q_A \cup Q_B$, $F := F_A \cup F_B$
- $\delta(r,x) := \{ \delta_A(r,x) \}$ if r in Q_A and $x \neq \varepsilon$
- $\delta(r,x) := \{ \delta_B(r,x) \}$ if r in Q_B and $x \neq \varepsilon$
- $\delta(q,\varepsilon) :=$?



Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A) : L(M_A) = A$,

  DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B) : L(M_B) = B$,

- Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:
- $Q := \{q\} \cup Q_A \cup Q_B$, $F := F_A \cup F_B$
- $\delta(r,x) := \{ \delta_A(r,x) \}$ if r in Q_A and $x \neq \varepsilon$
- $\delta(r,x) := \{ \delta_B(r,x) \}$ if r in Q_B and $x \neq \varepsilon$
- $\delta(q,\varepsilon) := \{q_A, q_B\}$
- We have $L(N) = A \cup B$

Example

Is L = {w in {0,1}* : |w| is divisible by 3 OR

w starts with a 1} regular?

Example

Is $L = \{w \text{ in } \{0,1\}^* : |w| \text{ is divisible by 3 OR } w \text{ starts with a } 1\}$ regular?

OR is like U, so try to write $L = L_1 \cup L_2$

where L_1, L_2 are regular

Example

Is $L = \{w \text{ in } \{0,1\}^* : |w| \text{ is divisible by 3 OR}$
$w \text{ starts with a 1}\}$ regular?

OR is like U, so try to write $L = L_1 \cup L_2$
where L_1, L_2 are regular

$L_1 = \{w : |w| \text{ is div. by } 3\}$ $L_2 = \{w : w \text{ starts with a } 1\}$

Example

Is $L = \{w \text{ in } \{0,1\}^* : |w| \text{ is divisible by 3 OR}$

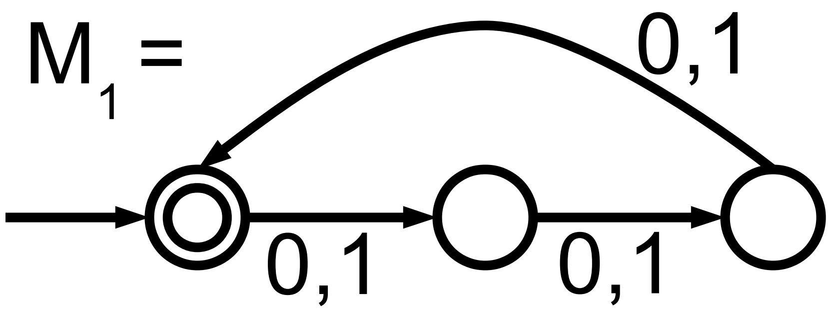
$w \text{ starts with a 1}\}$ regular?

OR is like U, so try to write $L = L_1 \cup L_2$

where L_1, L_2 are regular

$L_1 = \{w : |w| \text{ is div. by 3}\}$ $L_2 = \{w : w \text{ starts with a 1}\}$

$M_1 =$



$L(M_1) = L_1$

Example

Is $L = \{w \text{ in } \{0,1\}^* : |w| \text{ is divisible by 3 OR}$

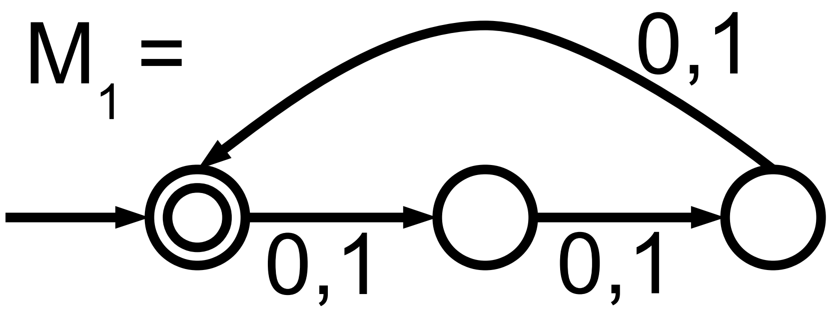
$w \text{ starts with a } 1\}$ regular?

OR is like U, so try to write $L = L_1 \cup L_2$

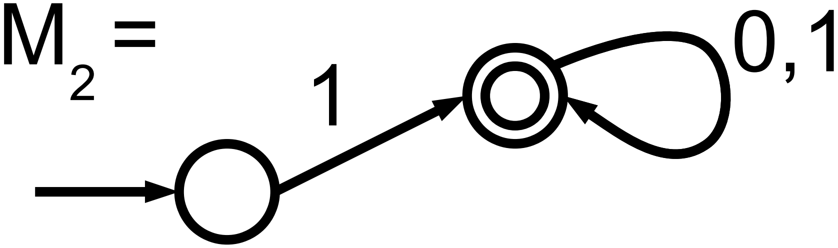
where L_1, L_2 are regular

$L_1 = \{w : |w| \text{ is div. by } 3\}$ $L_2 = \{w : w \text{ starts with a } 1\}$

$M_1 =$



$M_2 =$



$L(M_1) = L_1$ $L(M_2) = L_2$

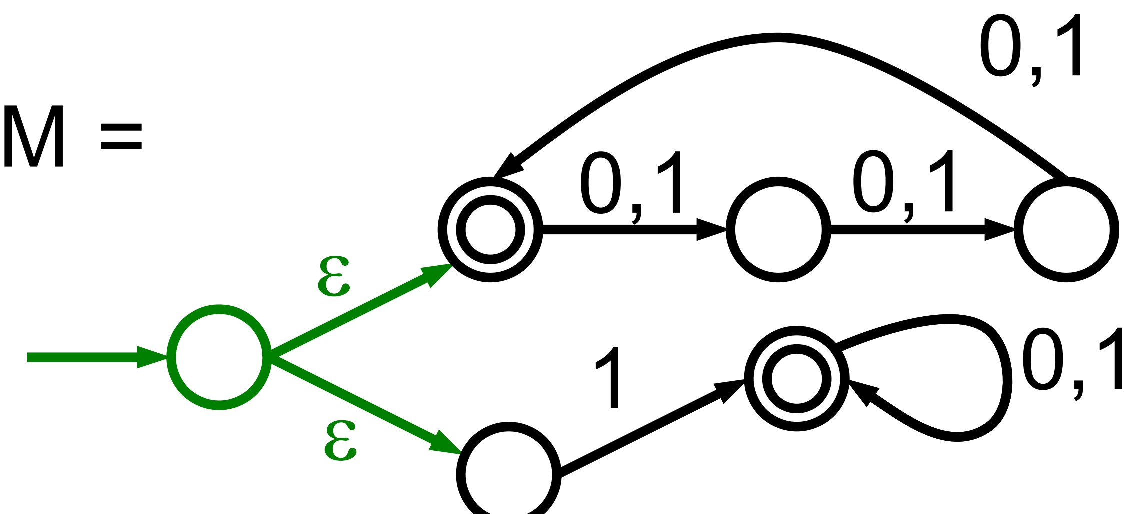
Example

Is $L = \{w \text{ in } \{0,1\}^* : |w| \text{ is divisible by 3 OR}$
$w \text{ starts with a 1}\}$ regular?

OR is like U, so try to write $L = L_1 \cup L_2$
where L_1, L_2 are regular

$L_1 = \{w : |w| \text{ is div. by 3}\}$ $L_2 = \{w : w \text{ starts with a 1}\}$

M =



$L(M) = L(M_1) \cup L(M_2)$
$= L_1 \cup L_2$
$= L$

$\Rightarrow$ L is regular.

We now return to the question:

- Suppose A, B are regular languages, then
- not A := { w : w is not in A } REGULAR
- A U B := { w : w in A or w in B } REGULAR
- A o B := { w_1 w_2 : w_1 in A and w_2 in B }
- A* := { w_1 w_2 … w_k : $k \geq 0$, w_i in A for every i }

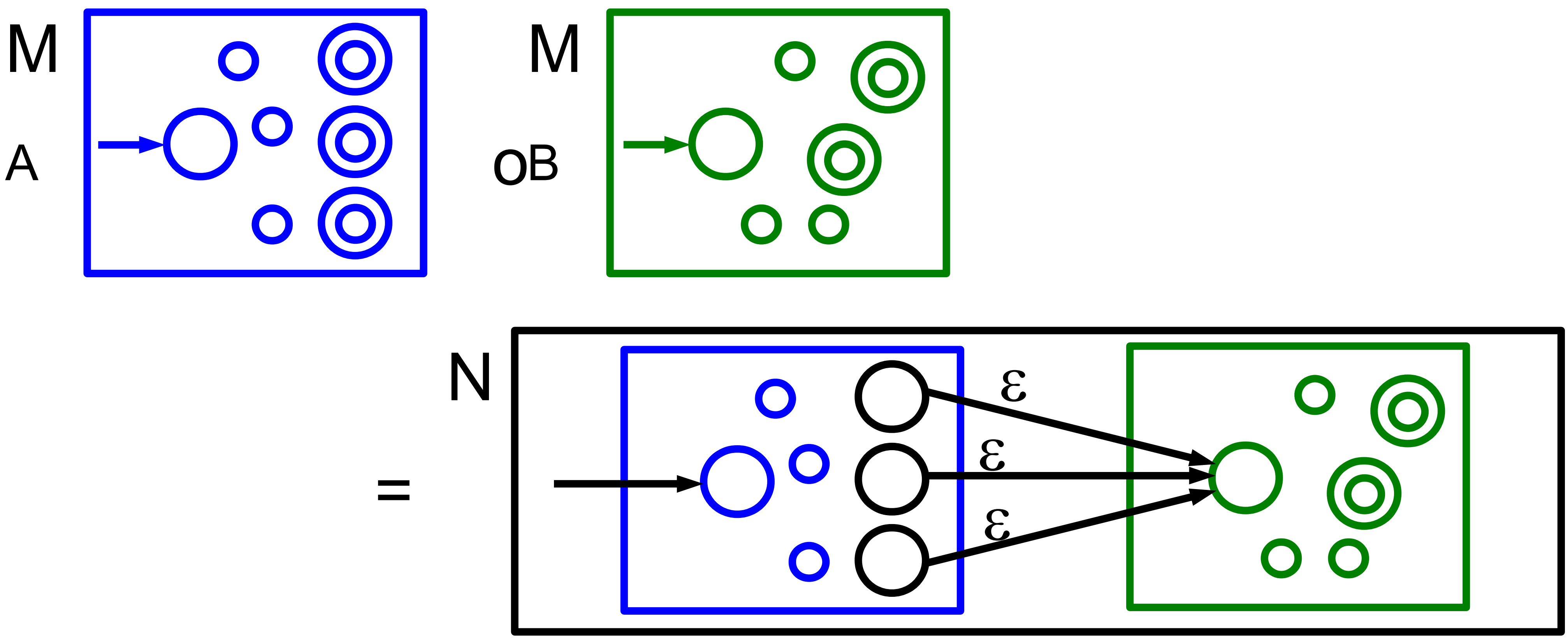
Theorem: If A, B are regular languages, then so is

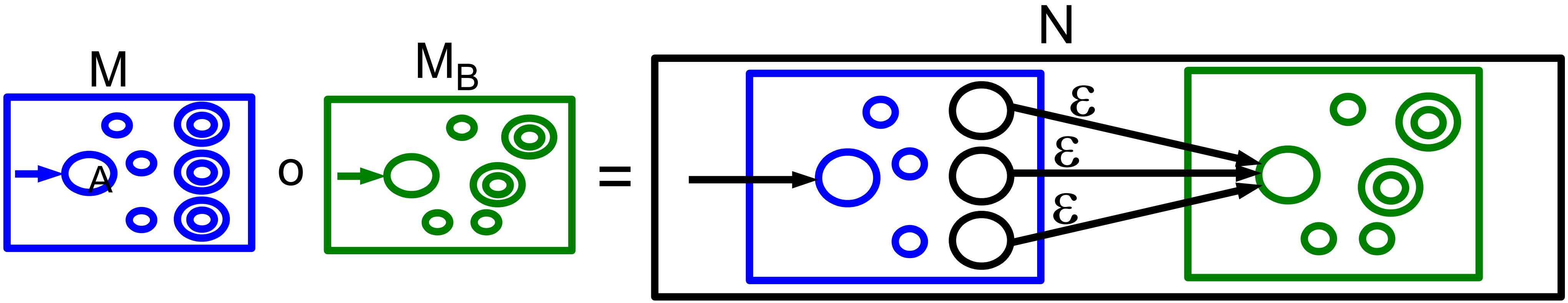
A o B := { w : w = xy for some

x in A and y in B }.

- Proof idea: Given DFAs M_A, M_B for A, B

  construct NFA N : L(N) = A o B.





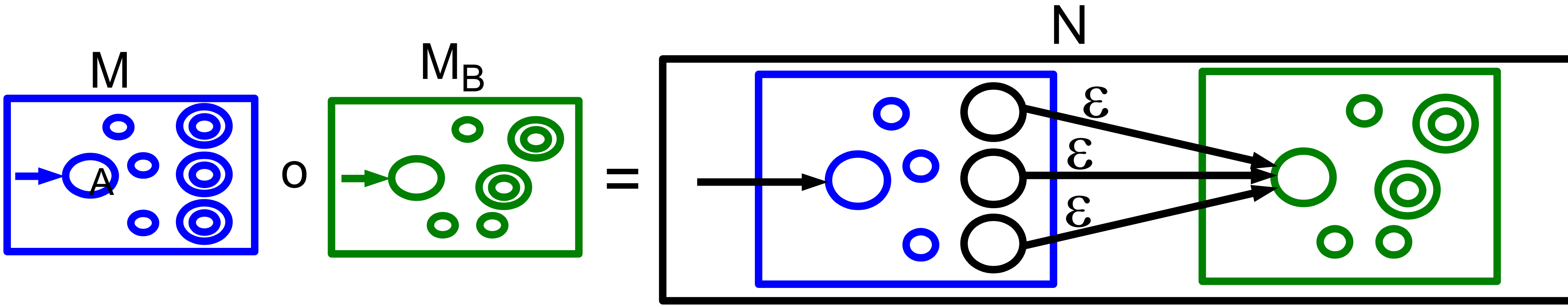
Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A) : L(M_A) = A$,

  DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B) : L(M_B) = B$,

- Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:

- $Q :=$?

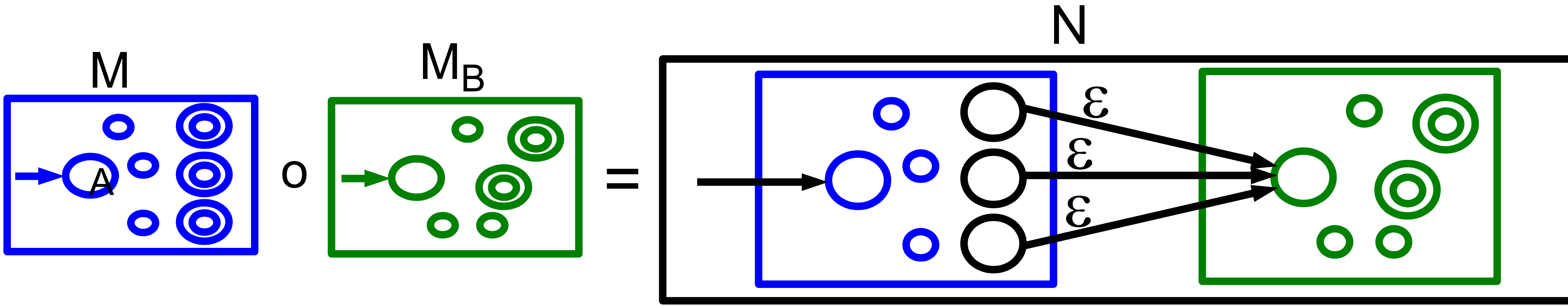


Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A) : L(M_A) = A$,

  DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B) : L(M_B) = B$,

- Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:
- $Q := Q_A \cup Q_B$, $q :=$?

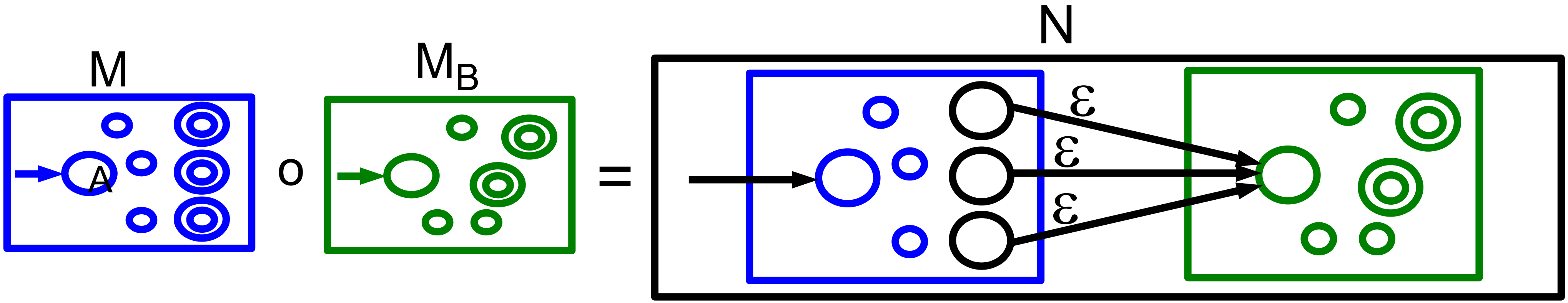


Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A) : L(M_A) = A$,

  DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B) : L(M_B) = B$,

- Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:
- $Q := Q_A \cup Q_B$, $q := q_A$, $F :=$?

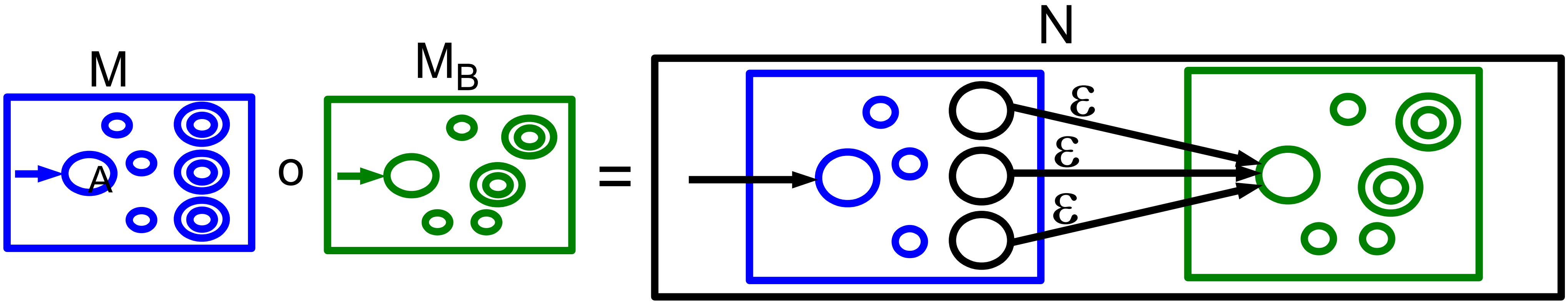


Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A) : L(M_A) = A$,

  DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B) : L(M_B) = B$,

- Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:
- $Q := Q_A \cup Q_B$, $q := q_A$, $F := F_B$
- $\delta(r,x) :=$? if r in Q_A and $x \neq \varepsilon$

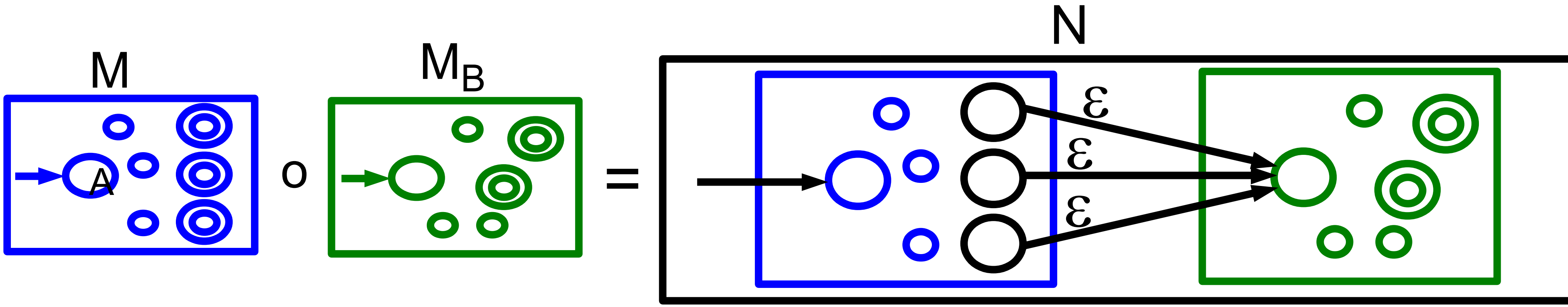


Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A) : L(M_A) = A$,

  DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B) : L(M_B) = B$,

- Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:
- $Q := Q_A \cup Q_B$, $q := q_A$, $F := F_B$
- $\delta(r,x) := \{ \delta_A(r,x) \}$ if r in Q_A and $x \neq \varepsilon$
- $\delta(r,\varepsilon) :=$? if r in F_A

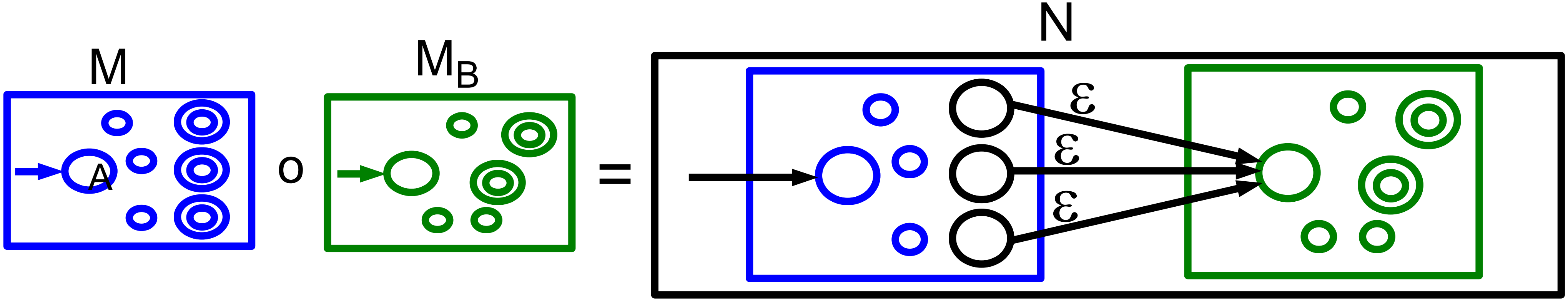


Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A) : L(M_A) = A,$

  DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B) : L(M_B) = B,$

- Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:
- $Q := Q_A \cup Q_B$, $q := q_A$, $F := F_B$
- $\delta(r,x) := \{ \delta_A(r,x) \}$ if r in Q_A and $x \neq \varepsilon$
- $\delta(r,\varepsilon) := \{ q_B \}$ if r in F_A
- $\delta(r,x) :=$? if r in Q_B and $x \neq \varepsilon$



Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A) : L(M_A) = A$,

  DFA $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B) : L(M_B) = B$,

- Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:
- $Q := Q_A \cup Q_B$, $q := q_A$, $F := F_B$
- $\delta(r,x) := \{ \delta_A(r,x) \}$ if r in Q_A and $x \neq \varepsilon$
- $\delta(r,\varepsilon) := \{ q_B \}$ if r in F_A
- $\delta(r,x) := \{ \delta_B(r,x) \}$ if r in Q_B and $x \neq \varepsilon$
- We have $L(N) = A \circ B$

Example

Is $L = \{w \text{ in } \{0,1\}^* : w \text{ contains a 1 after a 0}\}$ regular?

Note: $L = \{01, 0001001, 111001, \ldots\}$

Example

Is $L = \{w \text{ in } \{0,1\}^* : w \text{ contains a 1 after a 0}\}$

regular?

Let $L_0 = \{w : w \text{ contains a } 0\}$

$L_1 = \{w : w \text{ contains a } 1\}$. Then $L = L_0 \circ L_1$.

Example

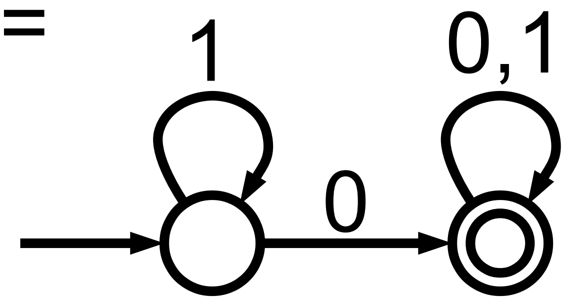
Is $L = \{w \text{ in } \{0,1\}^* : w \text{ contains a 1 after a 0}\}$

regular?

Let $L_0 = \{w : w \text{ contains a 0}\}$

$L_1 = \{w : w \text{ contains a 1}\}$. Then $L = L_0 \circ L_1$.

$M_0 =$



$L(M_0) = L_0$

Example

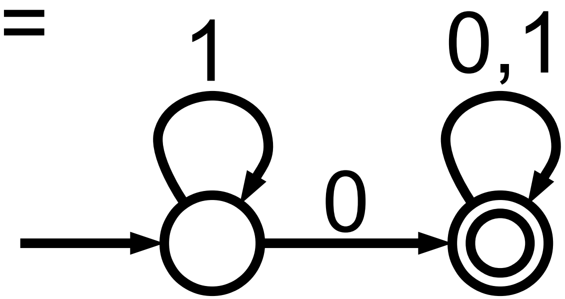
Is $L = \{w \text{ in } \{0,1\}^* : w \text{ contains a 1 after a 0}\}$

regular?

Let $L_0 = \{w : w \text{ contains a } 0\}$

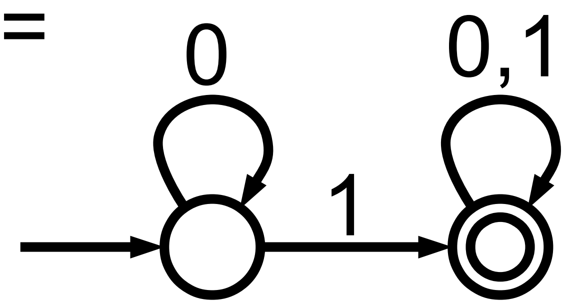
$L_1 = \{w : w \text{ contains a } 1\}$. Then $L = L_0 \circ L_1$.

$M_0 =$



$L(M_0) = L_0$

$M_1 =$



$L(M_1) = L_1$

Example

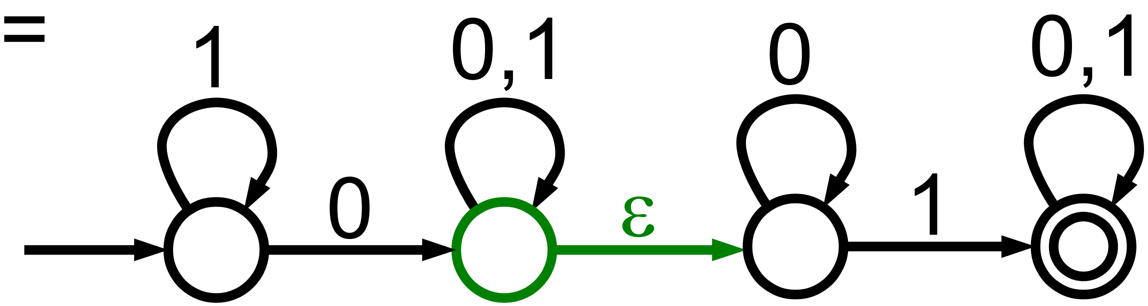
Is $L = \{w \text{ in } \{0,1\}^* : w \text{ contains a 1 after a 0}\}$

regular?

Let $L_0 = \{w : w \text{ contains a } 0\}$

$L_1 = \{w : w \text{ contains a } 1\}$. Then $L = L_0 \circ L_1$.

M =



$$L(M) = L(M_0) \circ L(M_1) = L_0 \circ L_1 = L$$

$\Rightarrow$ L is regular.

We now return to the question:

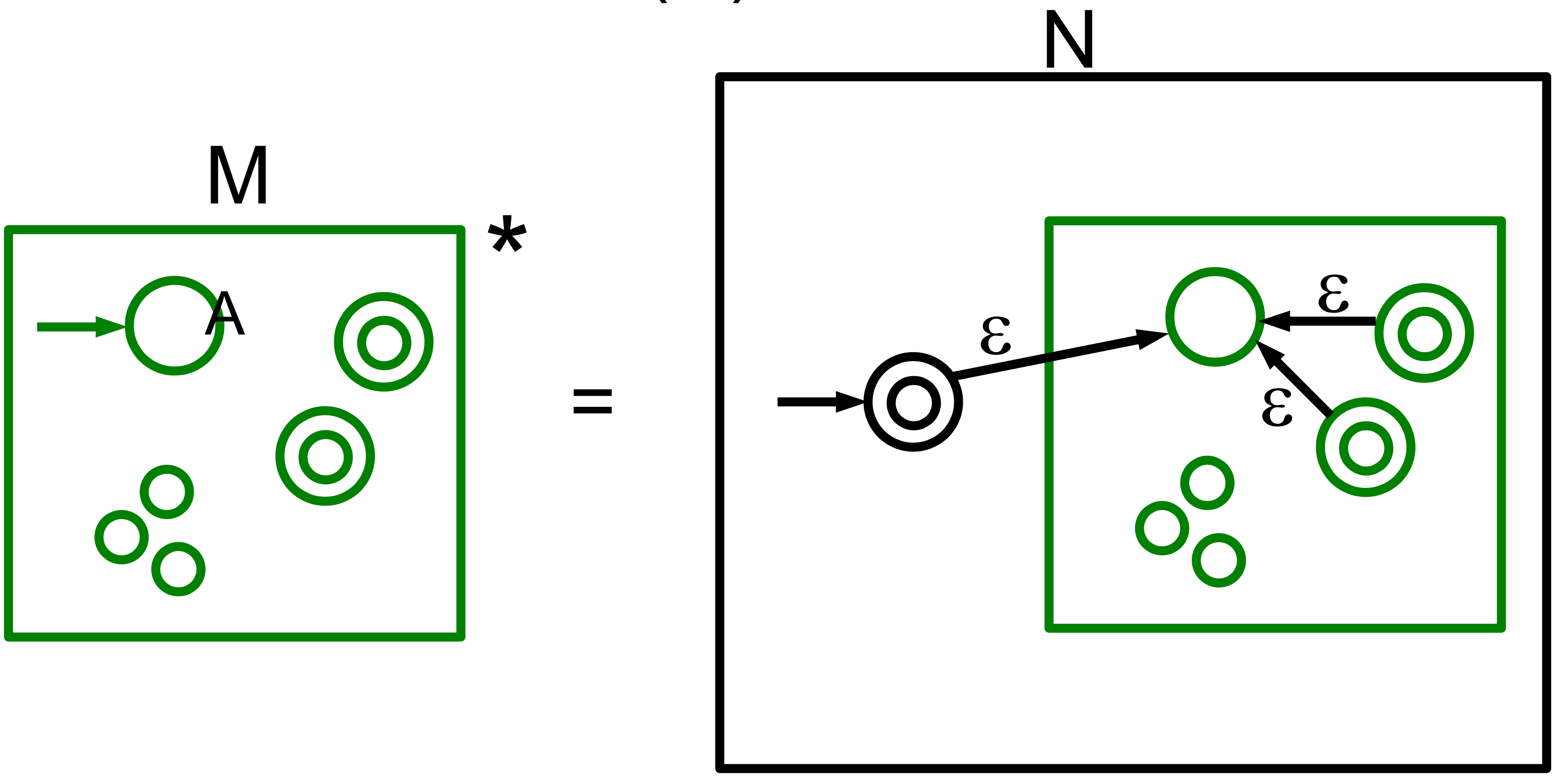
- Suppose A, B are regular languages, then
- not A := { w : w is not in A } REGULAR
- A U B := { w : w in A or w in B } REGULAR
- A o B := { w_1 w_2 : $w_1 \in$ A and $w_2 \in$ B } REGULAR
- A* := { w_1 w_2 … w_k : $k \geq 0$, w_i in A for every i }

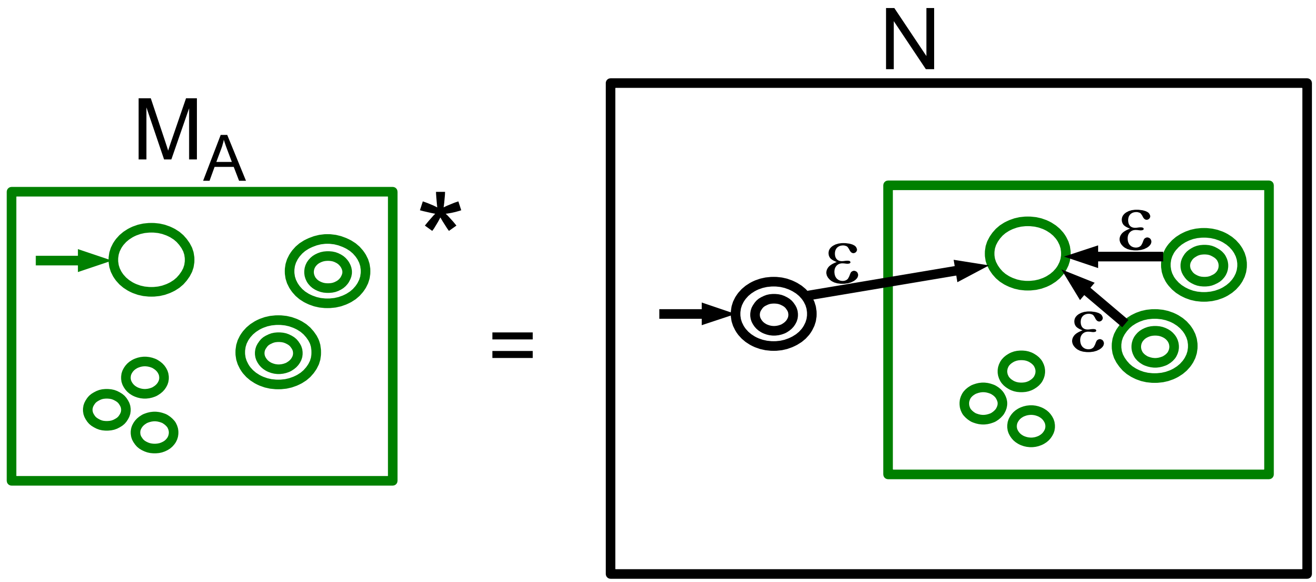
Theorem: If A is a regular language, then so is

$A^* := \{ w : w = w_1 \ldots w_k, \ w_i \text{ in } A \text{ for } i=1,\ldots,k \}$

- Proof idea: Given DFA M_A : $L(M_A) = A$,

Construct NFA N : L(N) = A*



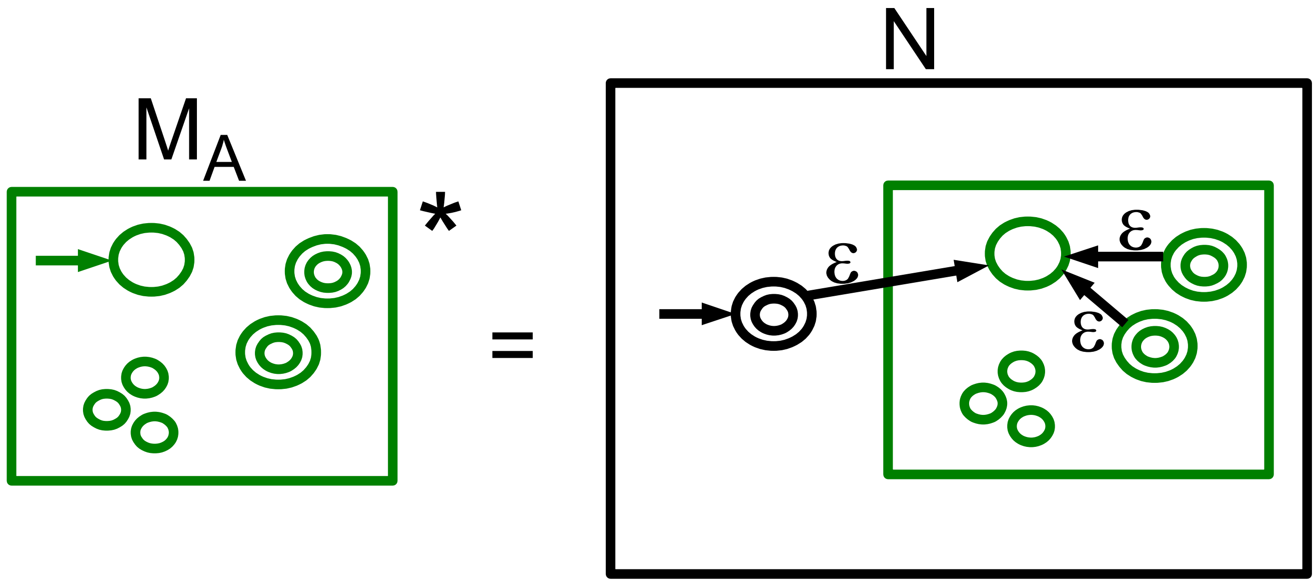


Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$: $L(M_A) = A$,

  Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:

- $Q :=$?

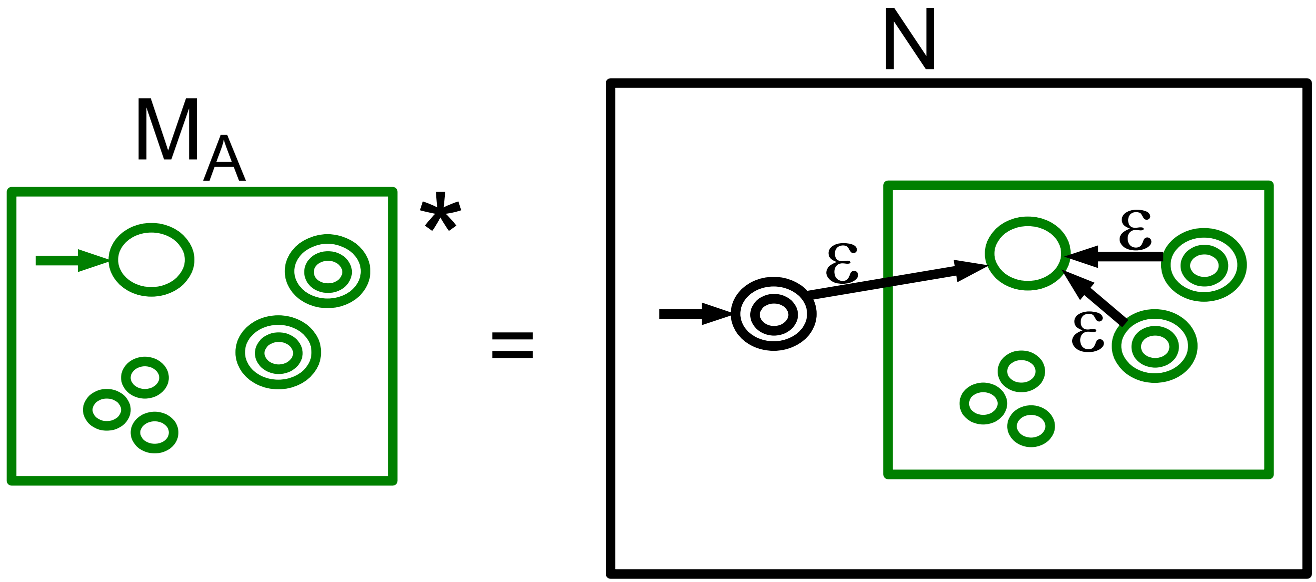


Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A) : L(M_A) = A$,

Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:

- $Q := \{q\} \cup Q_A$, $F :=$?

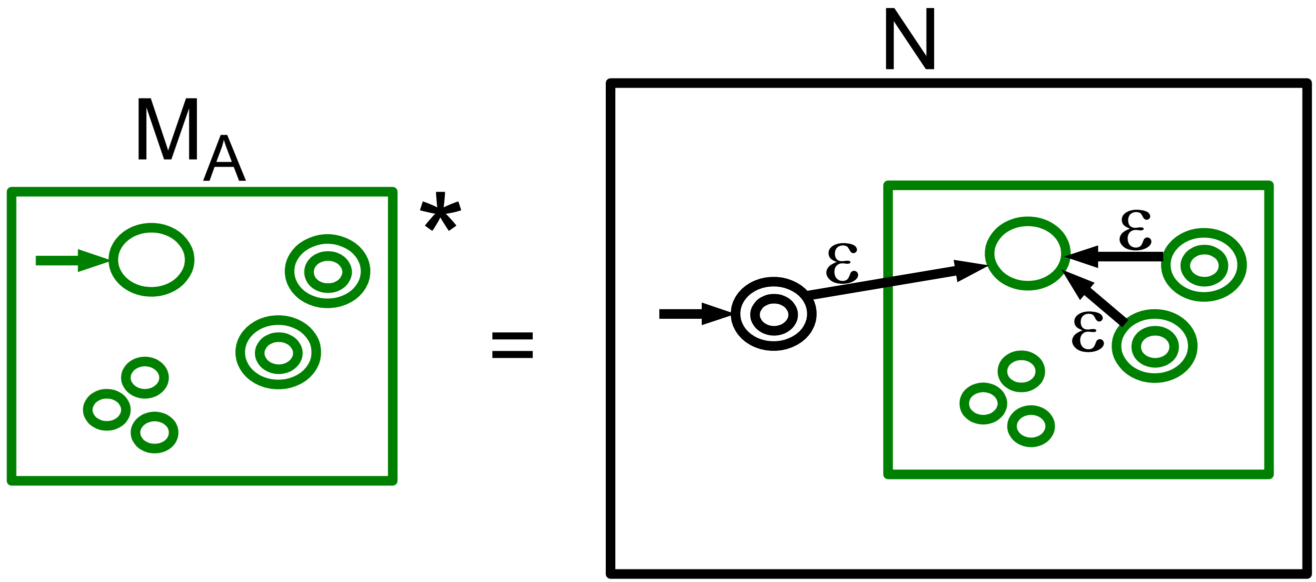


Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A) : L(M_A) = A$,

Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:

- $Q := \{q\} \cup Q_A$, $F := \{q\} \cup F_A$
- $\delta(r,x) :=$?   if r in Q_A and $x \neq \varepsilon$

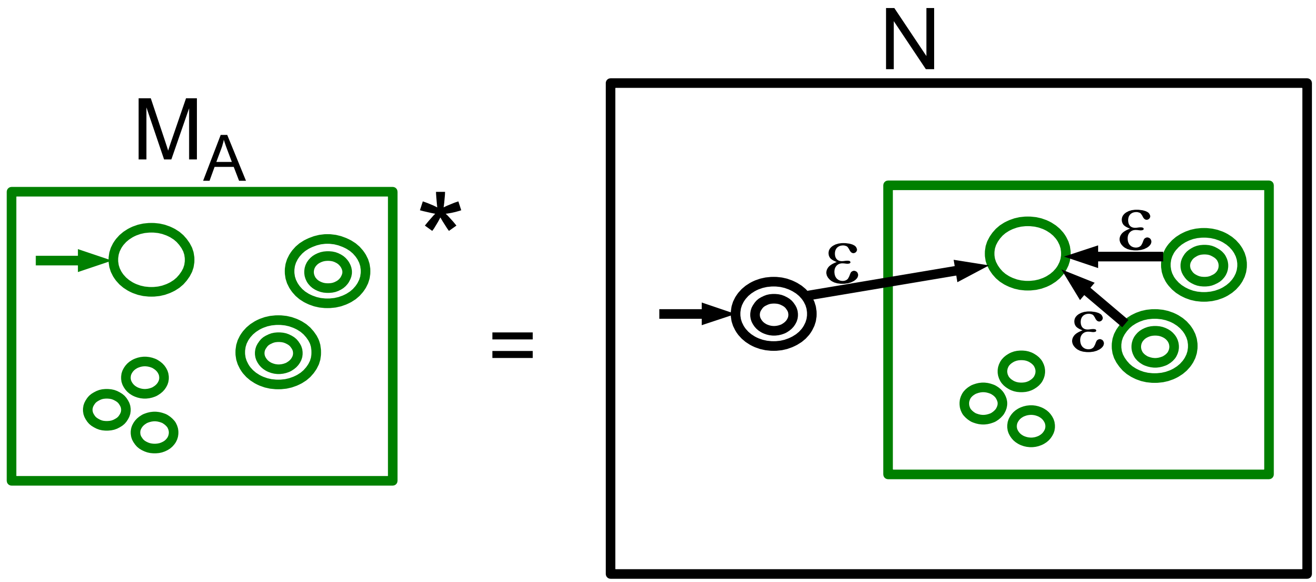


Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$: $L(M_A) = A$,

Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:

- $Q := \{q\} \cup Q_A$, $F := \{q\} \cup F_A$
- $\delta(r,x) := \{ \delta_A(r,x) \}$ if r in Q_A and $x \neq \varepsilon$
- $\delta(r,\varepsilon) :=$? if r in $\{q\} \cup F_A$



Construction:

- Given DFA $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$: $L(M_A) = A$,

Construct NFA $N = (Q, \Sigma, \delta, q, F)$ where:

- $Q := \{q\} \cup Q_A$, $F := \{q\} \cup F_A$
- $\delta(r,x) := \{ \delta_A(r,x) \}$ if r in Q_A and $x \neq \varepsilon$
- $\delta(r,\varepsilon) := \{ q_A \}$ if r in $\{q\} \cup F_A$
- We have $L(N) = A^*$

Example

Is $L = \{w \text{ in } \{0,1\}^* : w \text{ has even length}\}$

regular?

Example

Is $L = \{w \text{ in } \{0,1\}^* : w \text{ has even length}\}$

regular?

Let $L_0 = \{w : w \text{ has length} = 2\}$. Then $L = L_0^*$.

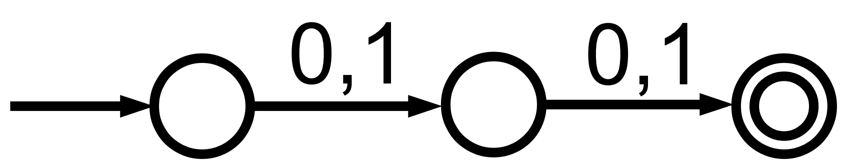
Example

Is $L = \{w \text{ in } \{0,1\}^* : w \text{ has even length}\}$

regular?

Let $L_0 = \{w : w \text{ has length} = 2\}$. Then $L = L_0^*$.

$M_0 =$



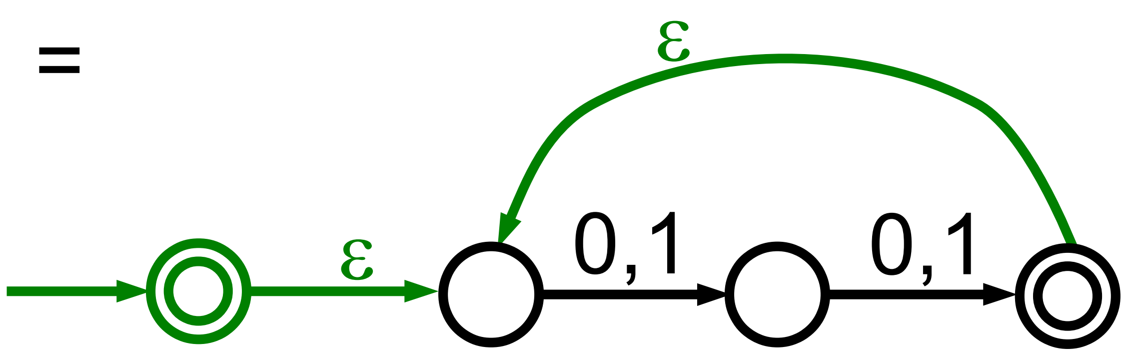
$L(M_0) = L_0$

Example

Is $L = \{w \text{ in } \{0,1\}^* : w \text{ has even length}\}$ regular?

Let $L_0 = \{w : w \text{ has length} = 2\}$. Then $L = L_0{}^*$.

M =



$$L(M) = L(M_0)^* = L_0{}^* = L$$

$\Rightarrow$ L is regular.

We now return to the question:

- Suppose A, B are regular languages, then
- not A := { w : w is not in A }
- A U B := { w : w in A or w in B }
- A o B := { w_1 w_2 : w_1 in A and w_2 in B }
- A* := { w_1 w_2 … w_k : $k \geq 0$, w_i in A for every i }

are all regular!

We now return to the question:

- Suppose A, B are regular languages, then
- not A := { w : w is not in A }
- A U B := { w : w in A or w in B }
- A o B := { w_1 w_2 : w_1 in A and w_2 in B }
- A* := { w_1 w_2 … w_k : $k \geq 0$, w_i in A for every i }

What about $A \cap B$:= { w : w in A and w in B } ?

We now return to the question:

- Suppose A, B are regular languages, then
- not A := { w : w is not in A }
- A U B := { w : w in A or w in B }
- A o B := { w_1 w_2 : w_1 in A and w_2 in B }
- A* := { w_1 w_2 … w_k : $k \geq 0$, w_i in A for every i }

De Morgan's laws: A ∩ B = not ((not A) U (not B))

By above, (not A) is regular, (not B) is regular,

(not A) U (not B) is regular,

not ((not A) U (not B)) = A ∩ B regular

We now return to the question:

- Suppose A, B are regular languages, then
- not A := { w : w is not in A }
- A U B := { w : w in A or w in B }
- A o B := { $w_1 w_2$: w_1 in A and w_2 in B }
- A* := { $w_1 w_2 \dots w_k$: $k \geq 0$, w_i in A for every i }
- A ∩ B := { w : w in A and w in B }

are all regular

Big picture

- All languages
- Decidable

  Turing machines
- NP
- P
- Context-free

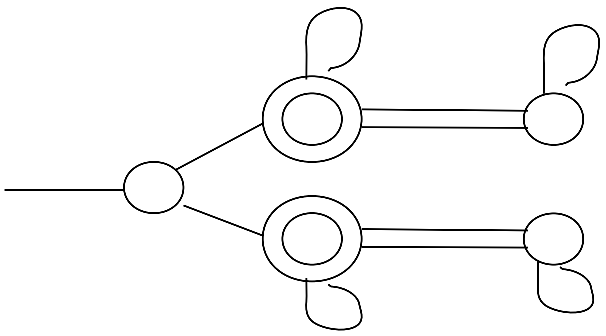
  Context-free grammars, push-down automata
- Regular

  Automata, non-deterministic automata,

  regular expressions

How to specify a regular language?

Write a picture → complicated



Write down formal definition → complicated

$\delta(q_0, 0) = q_0, \ldots$

Use symbols from Σ and operations *, o, U → good

$(\{0\}^* \cup \{1\}) \circ \{001\}$

Regular expressions: anything you can write with

$\varnothing$, ε , symbols from Σ, and operations *, o, U

Conventions:

- Write a instead of {a}
- Write AB for A o B
- Write $\sum$ for $U_{a \in \sum}$ a   So if $\sum$ = {a,b} then $\sum$ = a U b
- Operation * has precedence over o, and o over U

so 1 U 01* means 1U(0(1)*)

Example: 110, 0*, Σ*, Σ*001Σ*, ($\Sigma\Sigma$)*, 01 U 10

Definition Regular expressions RE over ∑ are:

$\emptyset$	
ε	
a	if a in Σ
R R'	if R, R' are RE
R U R'	if R, R' are RE
R*	if R is RE

Definition The language described by RE:

$L(\emptyset) = \emptyset$

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

$L(a) = \{a\}$ if a in Σ

$L(R\,R') = L(R) \circ L(R')$

$L(R \cup R') = L(R) \cup L(R')$

$L(R^*) = L(R)^*$

Example ∑ = { a, b}

RE	Language
ab U ba	?
a*	
(a U b)*	
a*ba*	
∑*b∑*	
∑*aab∑*	
(∑∑)*	
(a*ba*ba*)*	
a*baba*a Ø	

Example $\Sigma = \{ a, b\}$

RE	Language
• ab U ba	{ab, ba}
• a*	
• (a U b)*	
• a*ba*	
• ∑*b∑*	
• ∑*aab∑*	
• (∑∑)*	
• (a*ba*ba*)*	
• a*baba*a Ø	

Example ∑ = { a, b}

RE	Language
• ab U ba	{ab, ba}
• a*	{ε, a, aa, … } = { w : w has only a}
• (a U b)*	
• a*ba*	
• ∑*b∑*	
• ∑*aab∑*	
• (∑∑)*	
• (a*ba*ba*)*	
• a*baba*a Ø	

Example $\Sigma = \{ a, b\}$

RE	Language
• ab U ba	{ab, ba}
• a*	{ε, a, aa, … } = { w : w has only a}
• (a U b)*	all strings
• a*ba*	
• ∑*b∑*	
• ∑*aab∑*	
• (∑∑)*	
• (a*ba*ba*)*	
• a*baba*a Ø	

Example ∑ = { a, b}

RE	Language
• ab U ba	{ab, ba}
• a*	{ε, a, aa, … } = { w : w has only a}
• (a U b)*	all strings
• a*ba*	{w : w has exactly one b}
• ∑*b∑*	
• ∑*aab∑*	
• (∑∑)*	
• (a*ba*ba*)*	
• a*baba*a Ø	

Example $\Sigma = \{ a, b\}$

RE	Language
• ab U ba	{ab, ba}
• a*	{ε, a, aa, … } = { w : w has only a}
• (a U b)*	all strings
• a*ba*	{w : w has exactly one b}
• Σ*bΣ*	{w : w has at least one b}
• Σ*aabΣ*	
• ($\Sigma\Sigma$)*	
• (a*ba*ba*)*	
• a*baba*a Ø	

Example ∑ = { a, b}

RE	Language
• ab U ba	{ab, ba}
• a*	{ε, a, aa, … } = { w : w has only a}
• (a U b)*	all strings
• a*ba*	{w : w has exactly one b}
• ∑*b∑*	{w : w has at least one b}
• ∑*aab∑*	{w : w contains the string aab}
• (∑∑)*	
• (a*ba*ba*)*	
• a*baba*a Ø	

Example Σ = { a, b}

RE	Language
ab U ba	{ab, ba}
a*	{ε, a, aa, … } = { w : w has only a}
(a U b)*	all strings
a*ba*	{w : w has exactly one b}
Σ*bΣ*	{w : w has at least one b}
Σ*aabΣ*	{w : w contains the string aab}
($\Sigma\Sigma$)*	{w : w has even length}
(a*ba*ba*)*	
a*baba*a Ø	

Example $\sum = \{a, b\}$

RE	Language
• ab U ba	{ab, ba}
• a*	{ε, a, aa, … } = { w : w has only a}
• (a U b)*	all strings
• a*ba*	{w : w has exactly one b}
• ∑*b∑*	{w : w has at least one b}
• ∑*aab∑*	{w : w contains the string aab}
• (∑∑)*	{w : w has even length}
• (a*ba*ba*)*	{w : w contains even number of b}
• a*baba*a Ø	

Example $\sum = \{ a, b\}$

RE	Language
• ab U ba	{ab, ba}
• a*	{ε, a, aa, … } = { w : w has only a}
• (a U b)*	all strings
• a*ba*	{w : w has exactly one b}
• ∑*b∑*	{w : w has at least one b}
• ∑*aab∑*	{w : w contains the string aab}
• (∑∑)*	{w : w has even length}
• (a*ba*ba*)*	{w : w contains even number of b}
• a*baba*a Ø	Ø (anything o Ø = Ø)

Theorem: For every RE R there is NFA M: L(M) = L(R)

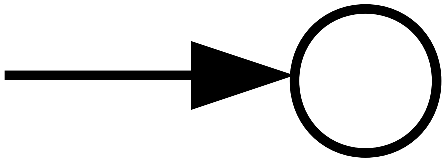
Theorem: For every RE R there is NFA M: L(M) = L(R)

Construction:

- $R = \varnothing$ $\quad\quad\quad$ $M := ?$

Theorem: For every RE R there is NFA M: L(M) = L(R)

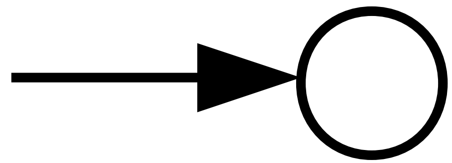
Construction:

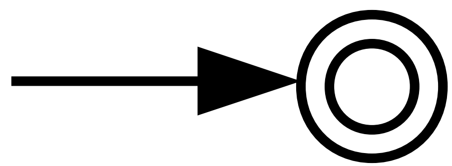
- R = $\varnothing$ M := 

- R = ε M := ?

Theorem: For every RE R there is NFA M: L(M) = L(R)

Construction:

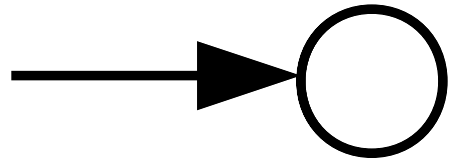
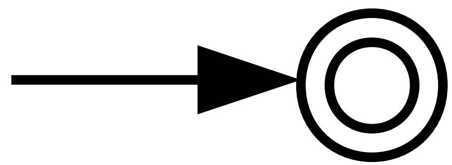
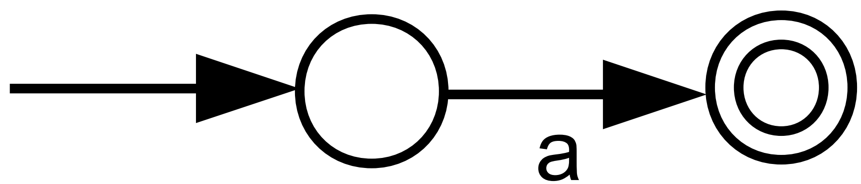
- R = ∅ M := 

- R = ε M := 

- R = a M := ?

Theorem: For every RE R there is NFA M: $L(M) = L(R)$

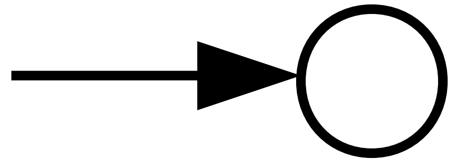
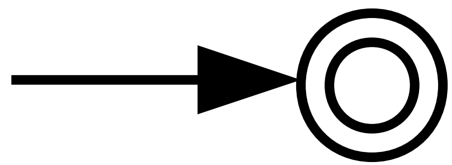
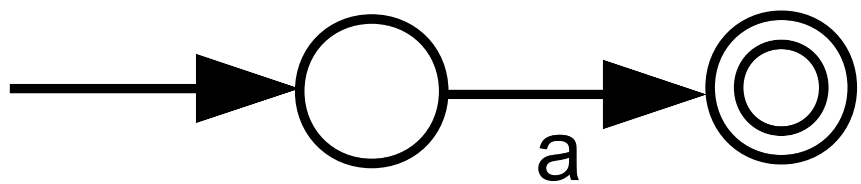
Construction:

- $R = \varnothing$ M := 
- $R = \varepsilon$ M := 
- $R = a$ M := 

- $R = R \cup R'$?

Theorem: For every RE R there is NFA M: L(M) = L(R)

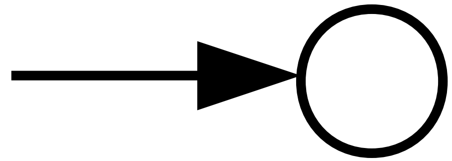
Construction:

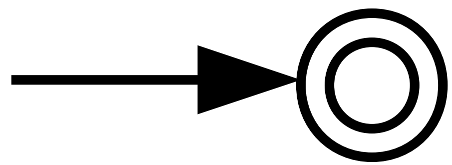
- R = ∅ M := 
- R = ε M := 
- R = a M := 

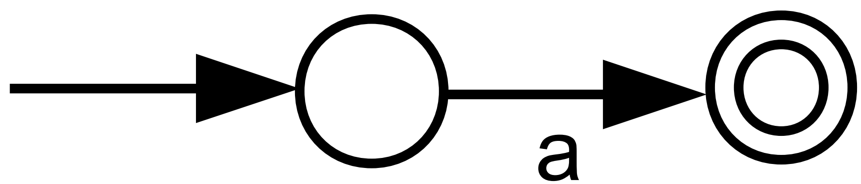
- R = R U R' use construction for A U B seen earlier
- R = R o R' ?

Theorem: For every RE R there is NFA M: L(M) = L(R)

Construction:

- R = ∅ M := 

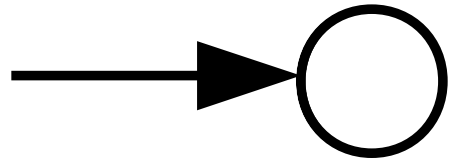
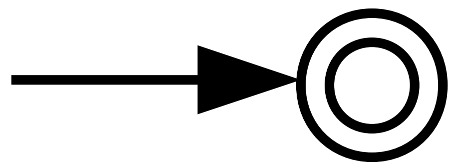
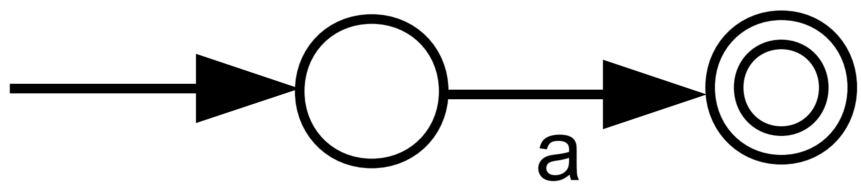
- R = ε M := 

- R = a M := 

- R = R U R' use construction for A U B seen earlier
- R = R o R' use construction for A o B seen earlier
- R = R* ?

Theorem: For every RE R there is NFA M: L(M) = L(R)

Construction:

- R = ∅ M := 
- R = ε M := 
- R = a M := 

- R = R U R' use construction for A U B seen earlier
- R = R o R' use construction for A o B seen earlier
- R = R* use construction for A* seen earlier

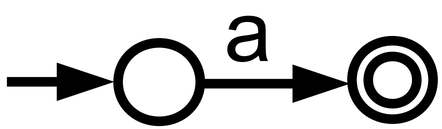
Example: RE → NFA

$$RE = (ab \cup a)^*$$

Example: RE → NFA

$$RE = (ab \cup a)^*$$

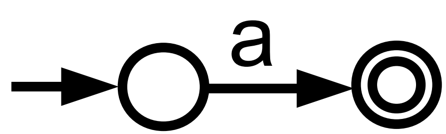
M_a =

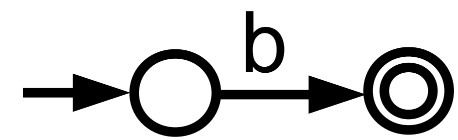


$$L(M_a)=L(a)$$

Example: RE $\rightarrow$ NFA

RE = (ab U a)*

M_a = 

M_b = 

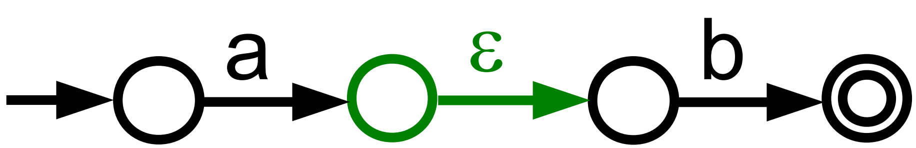
$L(M_a)=L(a)$

$L(M_b)=L(b)$

Example: RE → NFA

RE = (ab U a)*

M_{ab} =

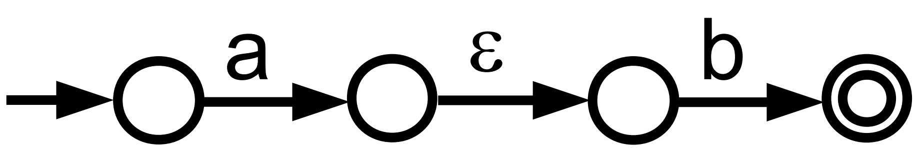


$L(M_{ab})=L(ab)$

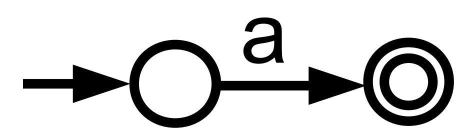
Example: RE → NFA

$$RE = (ab \cup a)^*$$

M_{ab} =



M_a =



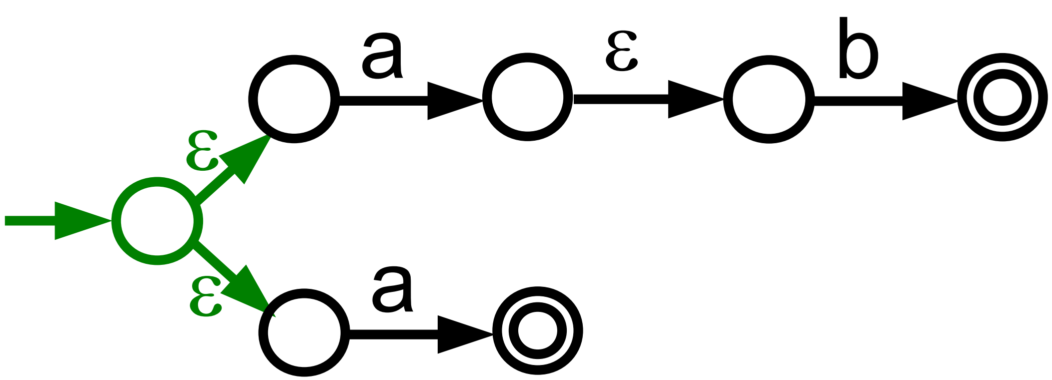
$L(M_{ab})=L(ab)$

$L(M_a)=L(a)$

Example: RE → NFA

$$RE = (ab\ U\ a)^*$$

$$M_{ab\ U\ a} =$$

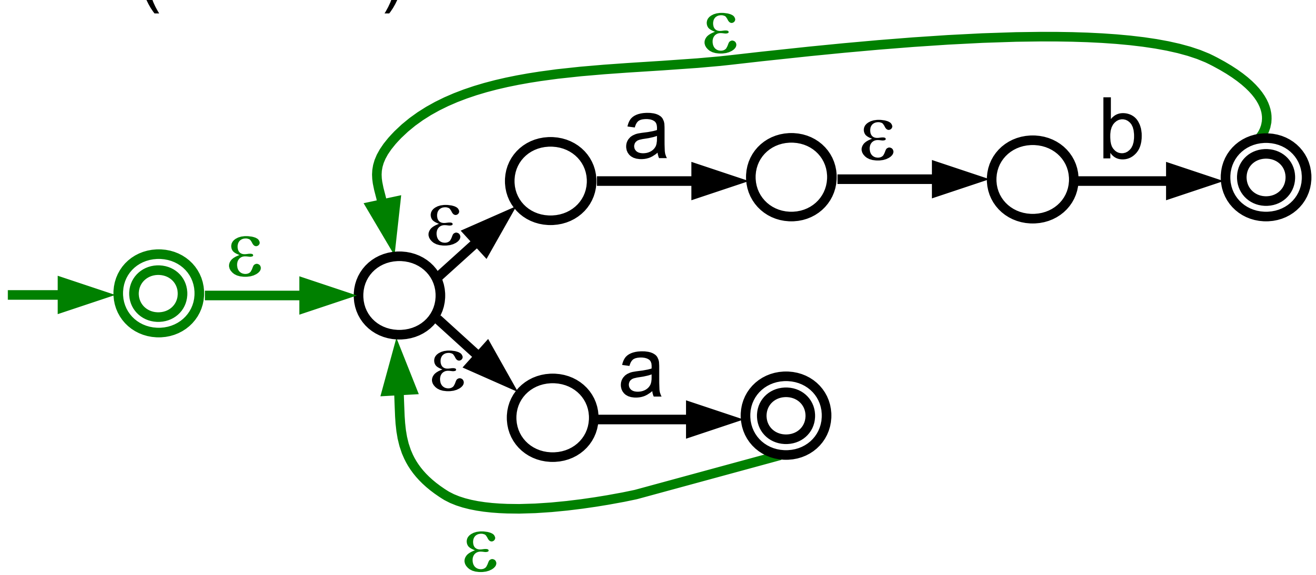


$$L(M_{ab\ U\ a}) = L(ab\ U\ a)$$

Example: RE → NFA

$$RE = (ab \cup a)^*$$

$$M_{(ab \cup a)^*} =$$



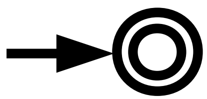
$$L(M_{(ab \cup a)^*}) = L((ab \cup a)^*) = L(RE)$$

ANOTHER Example: RE $\rightarrow$ NFA

$$RE = (\varepsilon \cup a)ba^*$$

ANOTHER Example: RE $\rightarrow$ NFA

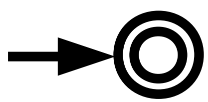
$$RE = (\varepsilon \cup a)ba^*$$

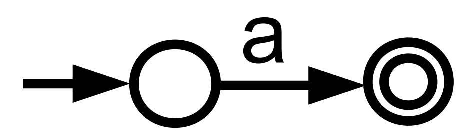
$M_\varepsilon =$ 

$$L(M_\varepsilon) = L(\varepsilon)$$

ANOTHER Example: RE → NFA

$$RE = (\varepsilon \cup a)ba^*$$

$M_\varepsilon =$ 

$M_a =$ 

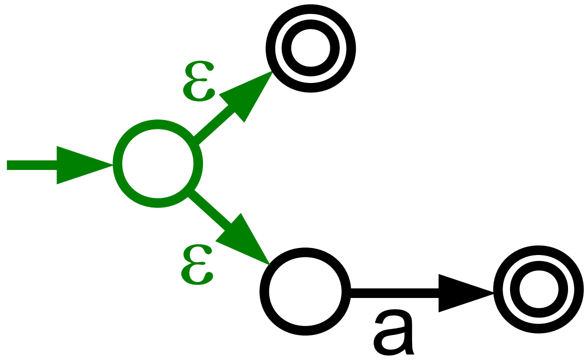
$L(M_\varepsilon) = L(\varepsilon)$

$L(M_a) = L(a)$

ANOTHER Example: RE → NFA

$$RE = (\varepsilon \cup a)ba^*$$

$M_{\varepsilon \cup a} =$

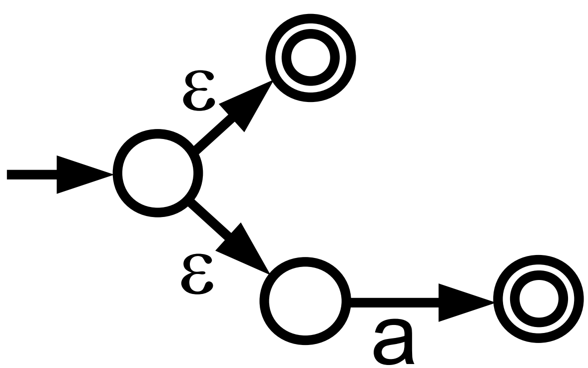


$$L(M_{\varepsilon \cup a}) = L(\varepsilon \cup a)$$

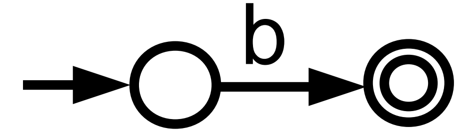
ANOTHER Example: RE → NFA

$$RE = (\varepsilon \cup a)ba^*$$

$M_{\varepsilon \cup a} =$



$M_b =$



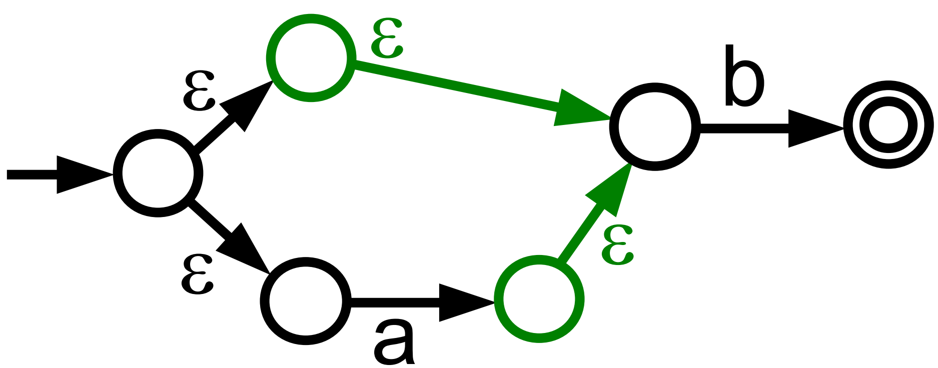
$$L(M_b) = L(b)$$

$$L(M_{\varepsilon \cup a}) = L(\varepsilon \cup a)$$

ANOTHER Example: RE → NFA

$$RE = (\varepsilon \cup a)ba^*$$

$$M_{(\varepsilon \cup a)b} =$$

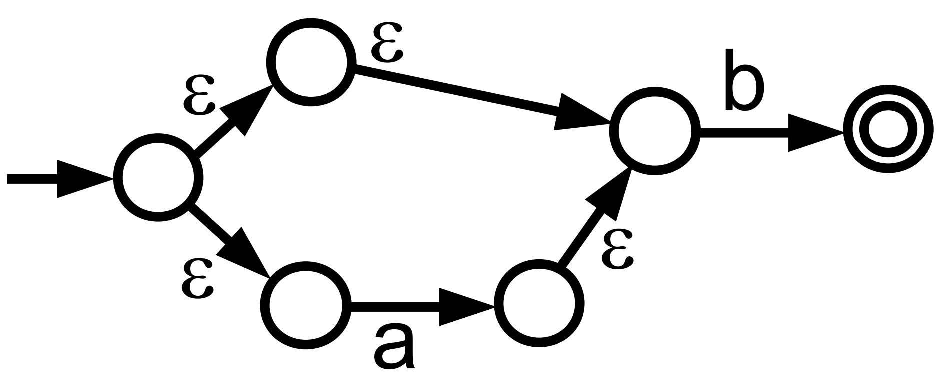


$$L(M_{(\varepsilon \cup a)b}) = L((\varepsilon \cup a)b)$$

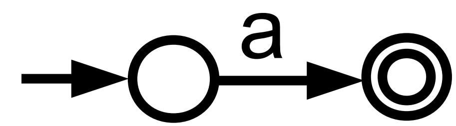
ANOTHER Example: RE → NFA

$$RE = (\varepsilon \cup a)ba^*$$

$M_{(\varepsilon \cup a)b} =$



$M_a =$



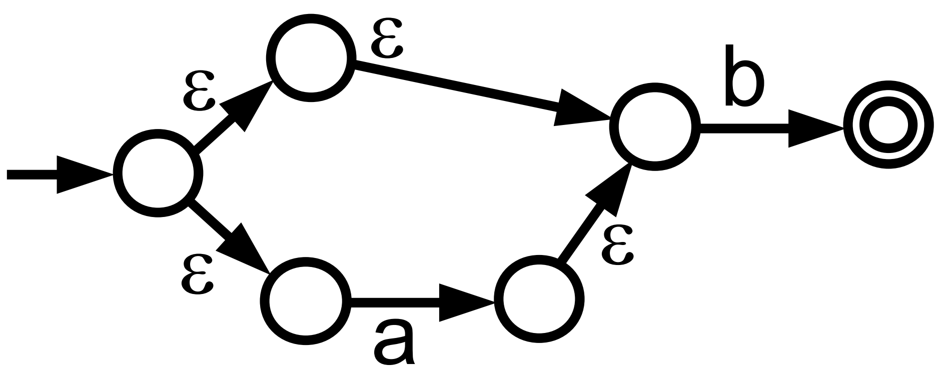
$L(M_a) = L(a)$

$L(M_{(\varepsilon \cup a)b}) = L((\varepsilon \cup a)b)$

ANOTHER Example: RE → NFA

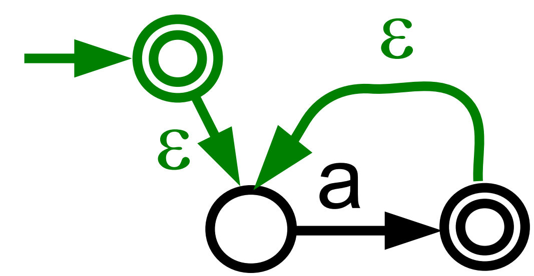
$$RE = (\varepsilon \cup a)ba^*$$

$M_{(\varepsilon \cup a)b} =$



$$L(M_{(\varepsilon \cup a)b}) = L((\varepsilon \cup a)b)$$

$M_{a^*} =$

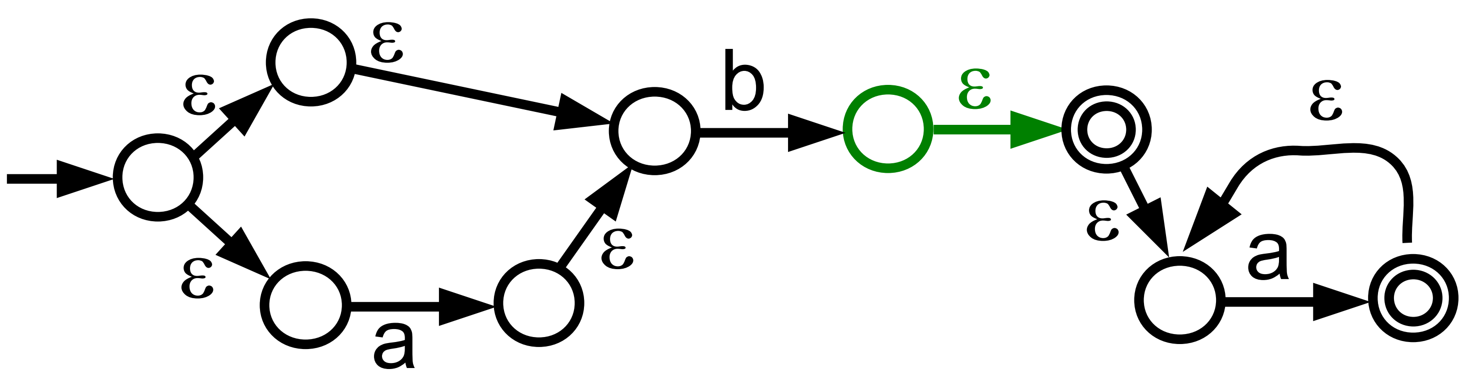


$$L(M_{a^*}) = L(a^*)$$

ANOTHER Example: RE → NFA

$$RE = (\varepsilon \cup a)ba^*$$

$$M_{(\varepsilon \cup a)ba^*} =$$



$$L(M_{(\varepsilon \cup a)ba^*}) = L((\varepsilon \cup a)ba^*) = L(RE)$$

Recap:

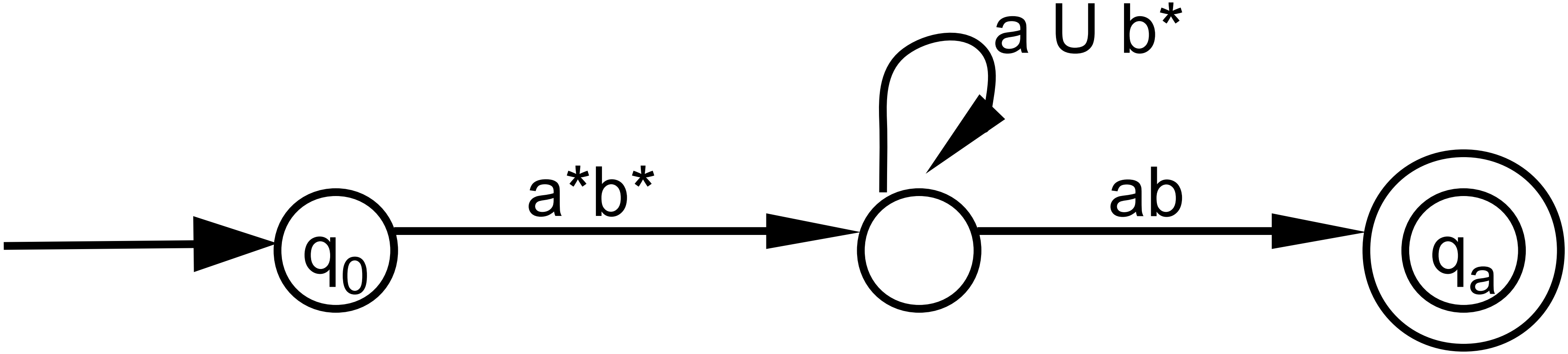
Here “$\Rightarrow$” means “can be converted to”

We have seen: RE $\Rightarrow$ NFA $\Leftrightarrow$ DFA

Next we see: DFA $\Rightarrow$ RE

In two steps: DFA $\Rightarrow$ Generalized NFA $\Rightarrow$ RE

Generalized NFA (GNFA)

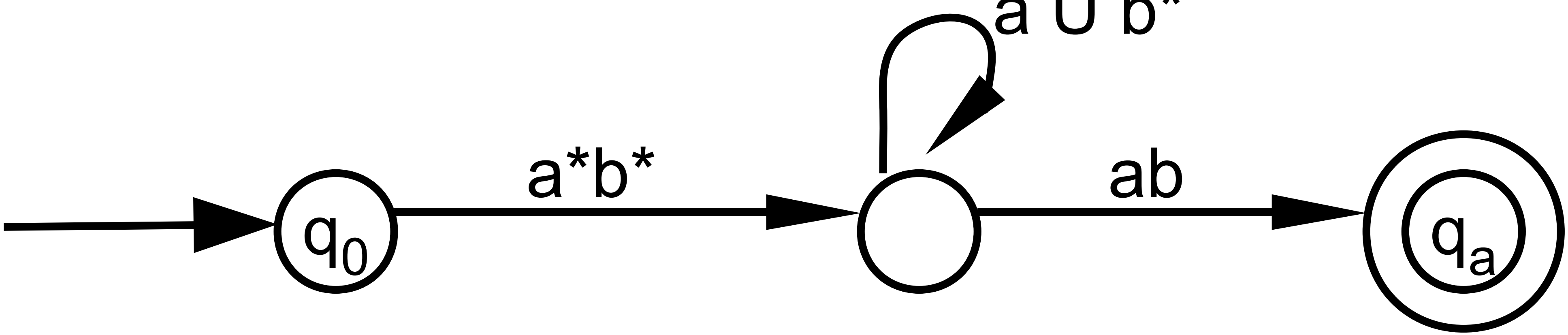


Nondeterministic

Transitions labelled by RE

Read blocks of input symbols at a time

Generalized NFA (GNFA)



Convention:

Unique final state

Exactly one transition between each pair of states

except nothing going into start state

nothing going out of final state

If arrow not shown in picture, label = ∅

- Definition: A generalized finite automaton (GNFA)
- is a 5-tuple $(Q, \Sigma, \delta, q_0, q_a)$ where
- Q is a finite set of states
- Σ is the input alphabet
- $\delta : (Q - \{q_a\}) \times (Q - \{q_0\}) \rightarrow$ Regular Expressions
- q_0 in Q is the start state
- q_a in Q is the accept state

- Definition: GNFA $(Q, \Sigma, \delta, q_0, q_a)$ accepts a string w if
- $\exists$ integer k, $\exists$ k strings $w_1, w_2, \ldots, w_k \in \Sigma^*$

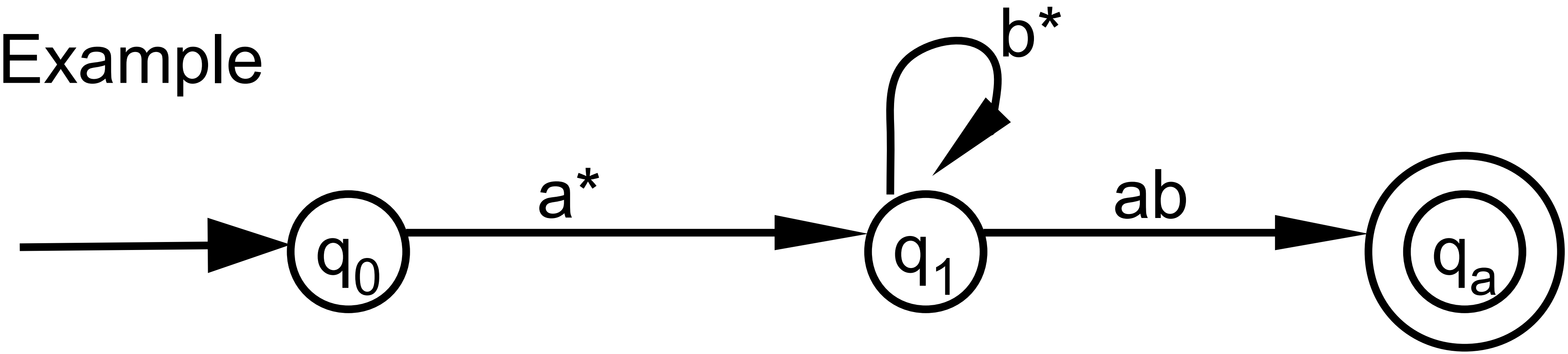
  such that $w = w_1 w_2 \ldots w_k$

  (divide w in k strings)

- $\exists$ sequence of k+1 states $r_0, r_1, .., r_k$ in Q such that:
- $r_0 = q_0$
- $w_{i+1} \in L(\delta(r_i, r_{i+1})) \; \forall\, 0 \leq i < k$
- $r_k = q_a$

- Differences with NFA are in green

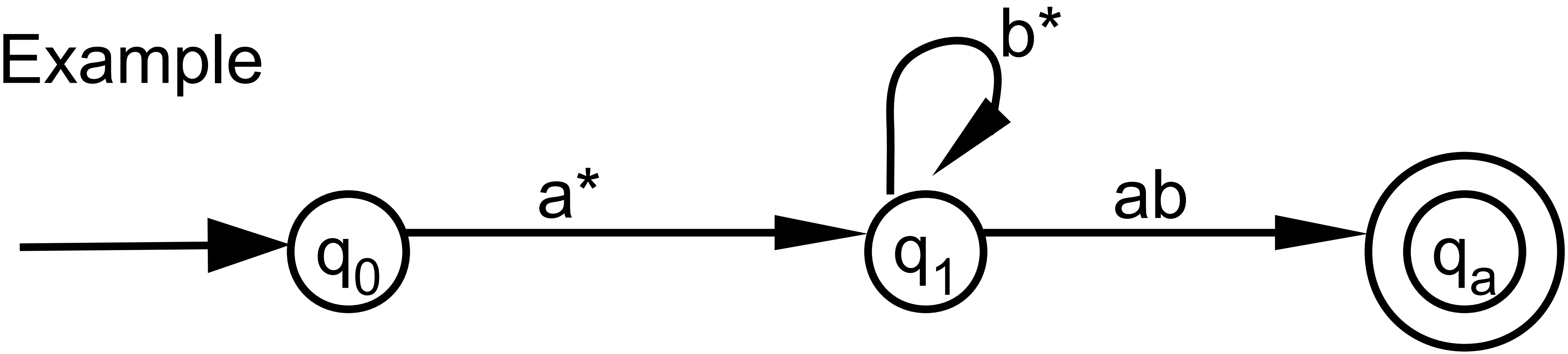
Example



Accepts w = aaabbbab

w_1=?

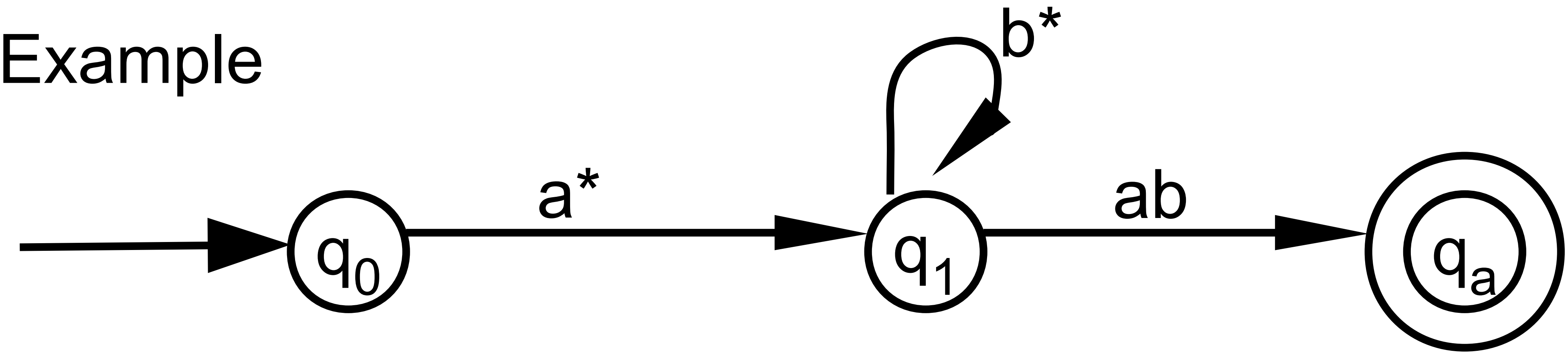
Example



Accepts w = aaabbab

w_1=aaa w_2=?

Example

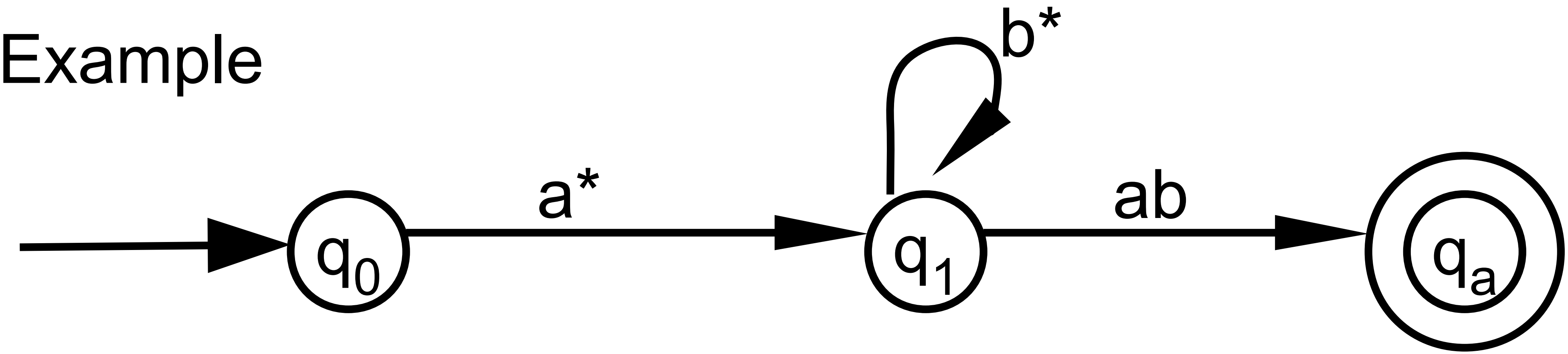


Accepts w = aaabbab

w_1=aaa w_2=bb w_3=ab

$r_0=q_0$ r_1=?

Example



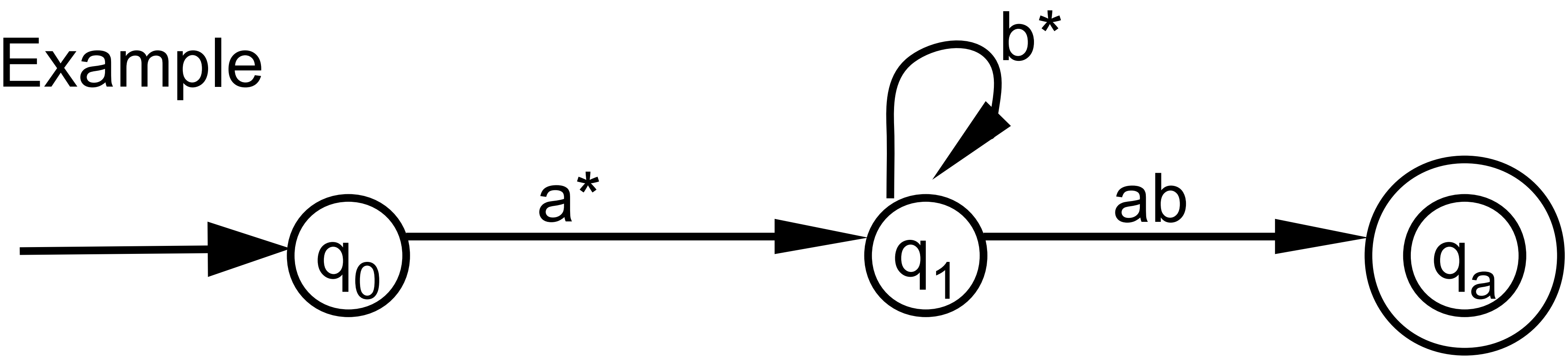
Accepts w = aaabbab

w_1=aaa w_2=bb w_3=ab

$r_0=q_0$ $r_1=q_1$ $r_2=$?

$w_1 = aaa \in L(\delta(r_0,r_1)) = L(\delta(q_0,q_1)) = L(a^*)$

Example



Accepts w = aaabbab

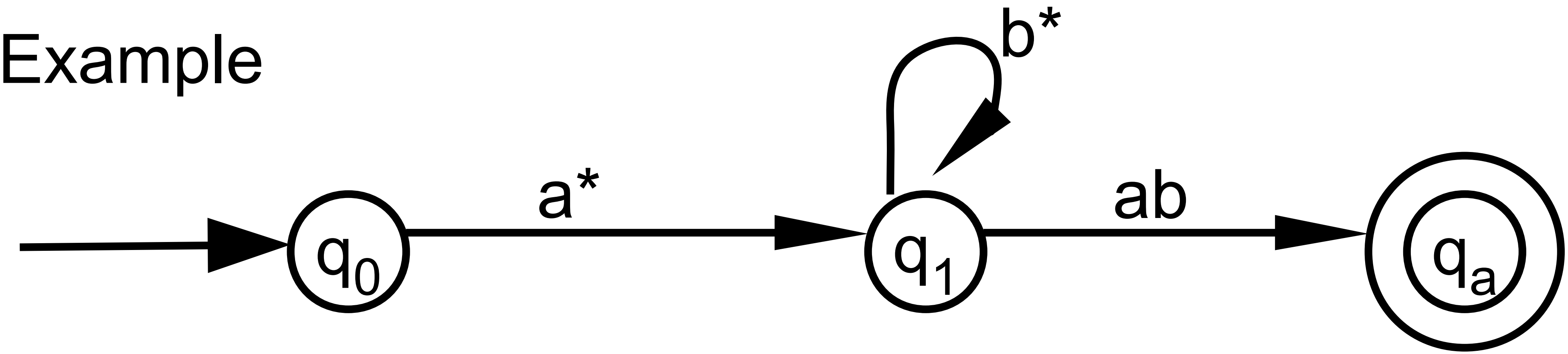
w_1=aaa w_2=bb w_3=ab

$r_0=q_0$ $r_1=q_1$ $r_2=q_1$ r_3 = ?

$w_1 = aaa \in L(\delta(r_0,r_1)) = L(\delta(q_0,q_1)) = L(a^*)$

$w_2 = bb \in L(\delta(r_1,r_2)) = L(\delta(q_1,q_1)) = L(b^*)$

Example



Accepts w = aaabbab

w_1=aaa w_2=bb w_3=ab

$r_0=q_0$ $r_1=q_1$ $r_2=q_1$ $r_3 = q_a$

$w_1 = aaa \in L(\delta(r_0,r_1)) = L(\delta(q_0,q_1)) = L(a^*)$

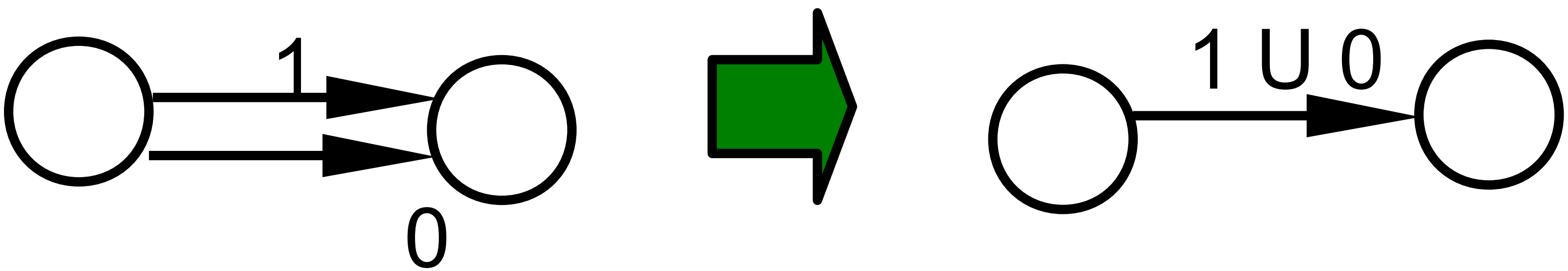
$w_2 = bb \in L(\delta(r_1,r_2)) = L(\delta(q_1,q_1)) = L(b^*)$

$w_3 = ab \in L(\delta(r_2,r_3)) = L(\delta(q_1,q_a)) = L(ab)$

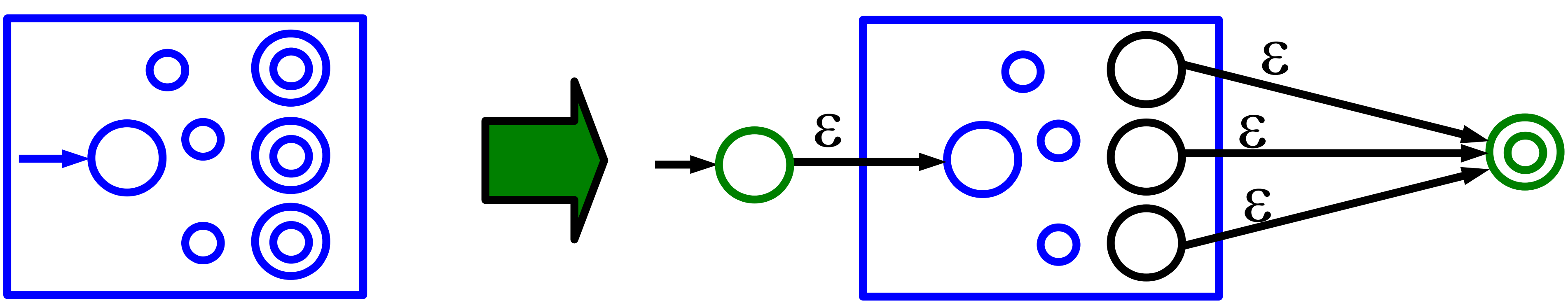
Theorem: ∀ DFA M ∃ GNFA N : L(N) = L(M)

Construction:

To ensure unique transition between each pair:



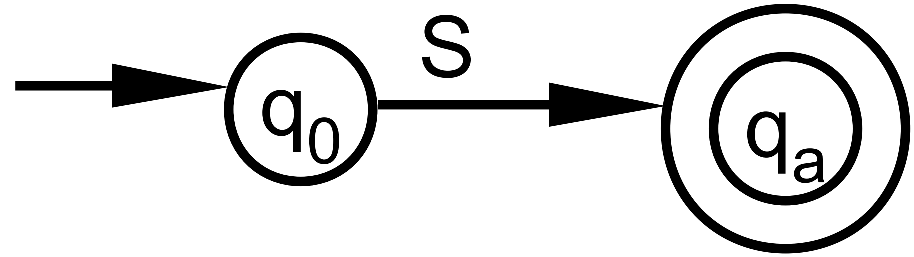
To ensure unique final state, no transitions ingoing start state, no transitions outgoing final state:



Theorem: ∀ GNFA N ∃ RE R : L(R) = L(N)

Construction:

If N has 2 states, then N =

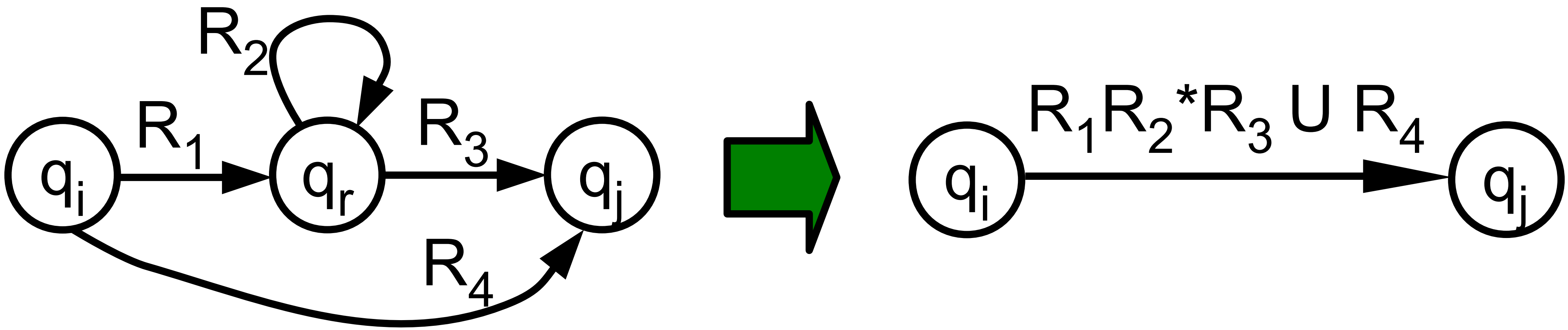


thus R := S

If N has > 2 states, eliminate some state $q_r \neq q_0, q_a$:

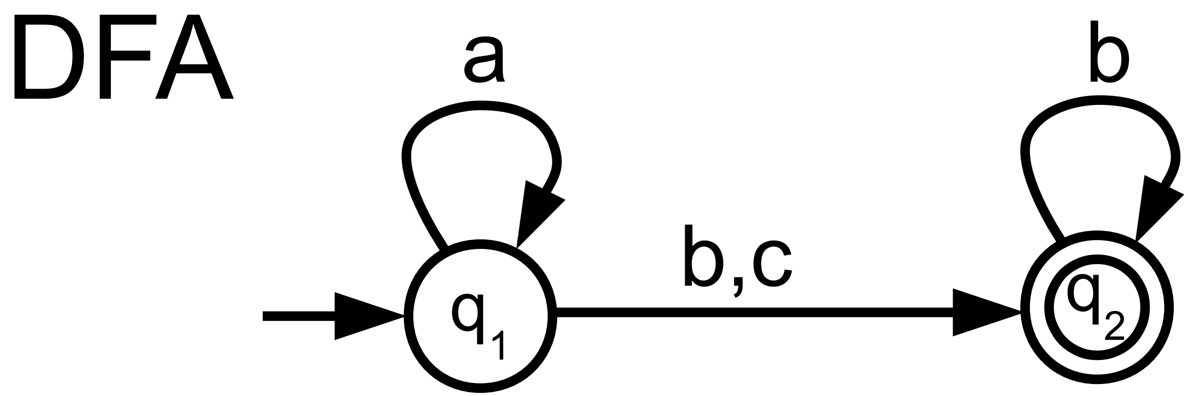
for every ordered pair q_i, q_j (possibly equal)

that are connected through q_r

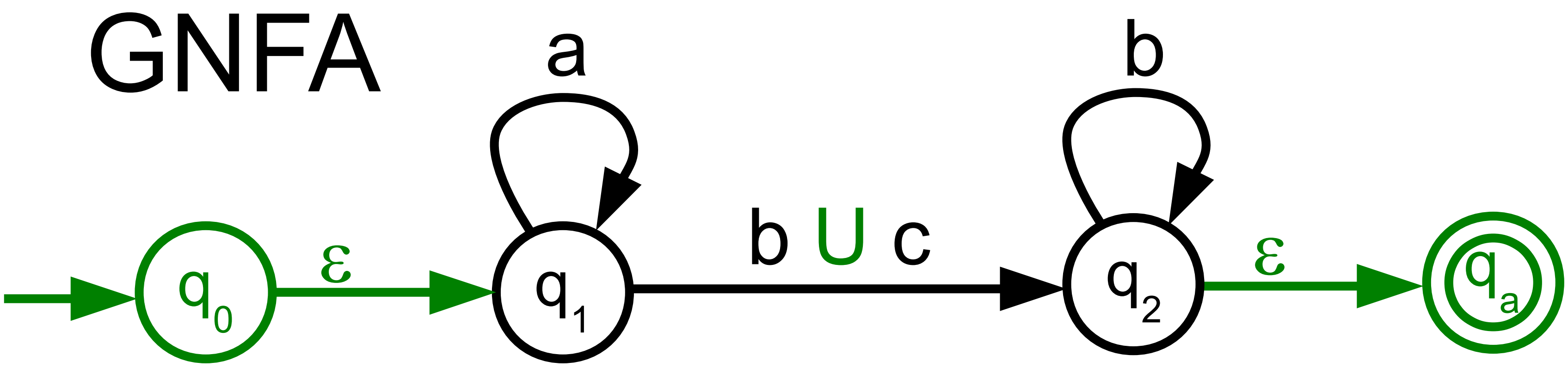


Repeat until 2 states remain

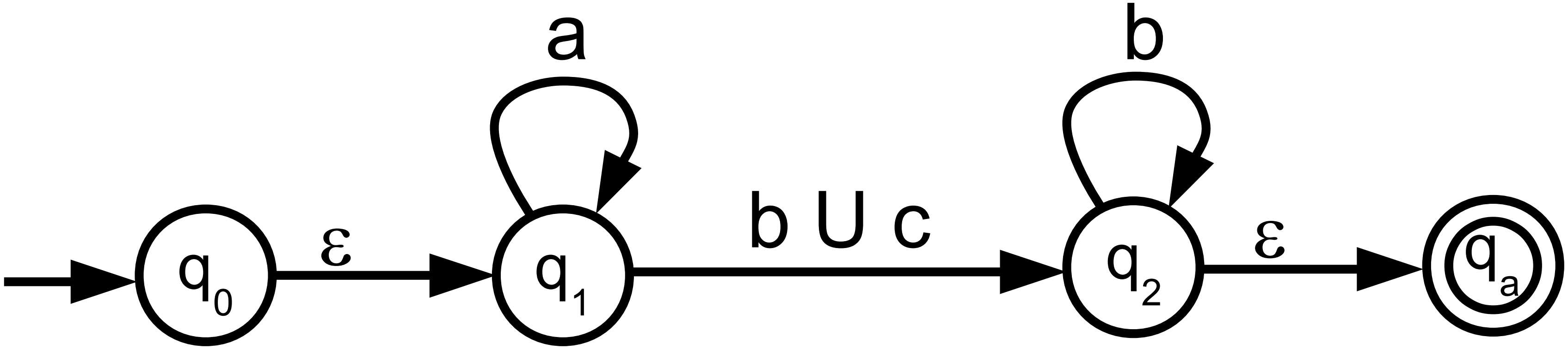
Example: DFA → GNFA → RE



Example: DFA → GNFA → RE

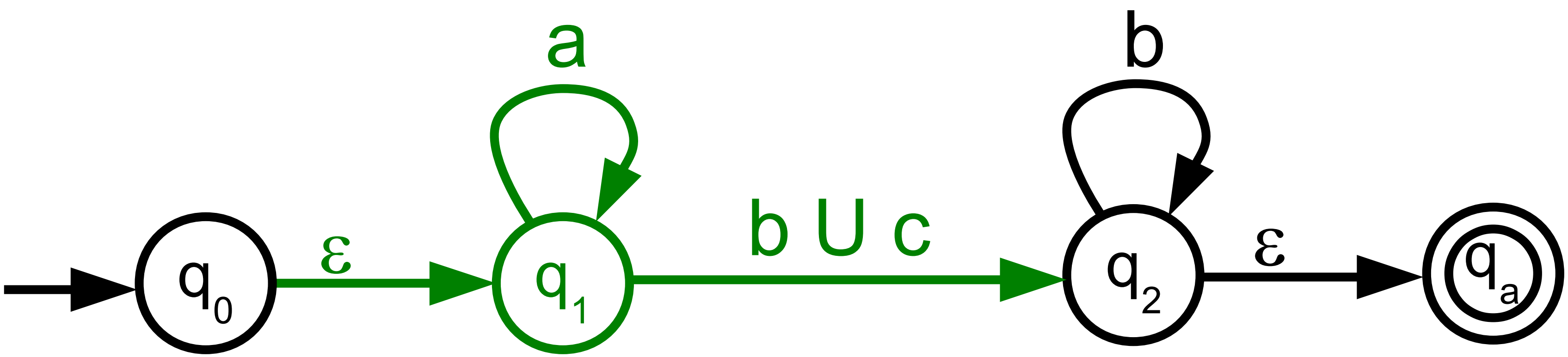


Example: DFA → GNFA → RE



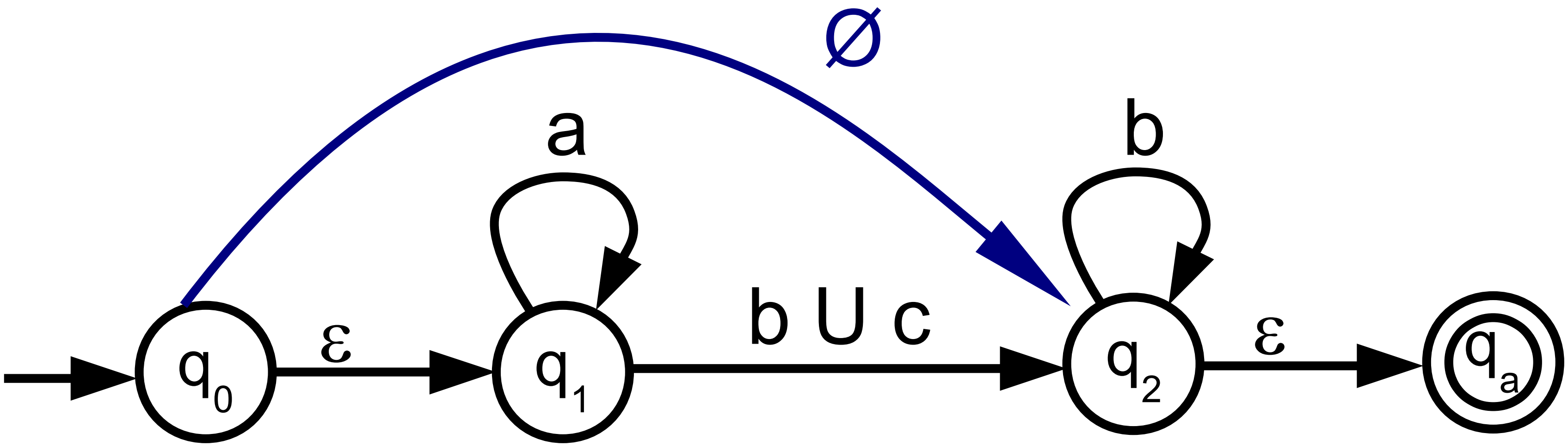
Eliminate q_1: re-draw GNFA with all other states

Example: DFA → GNFA → RE



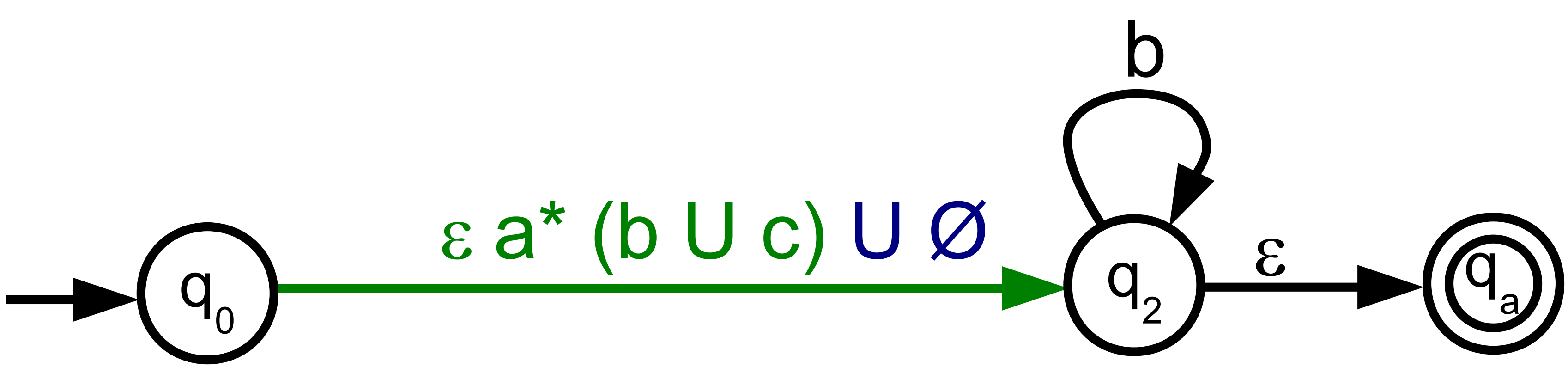
Eliminate q_1: find a path through q_1

Example: DFA → GNFA → RE

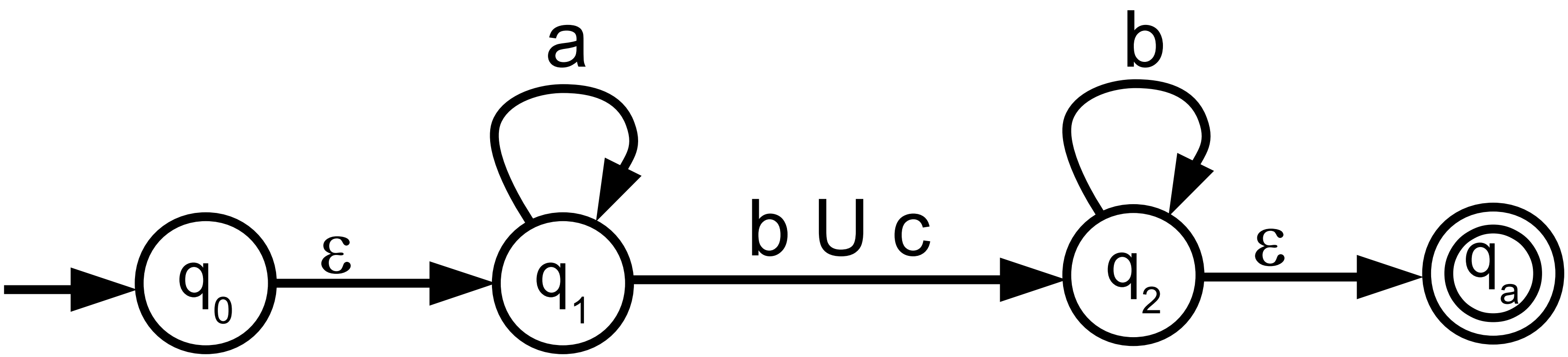


Eliminate q_1: add edge to new GNFA

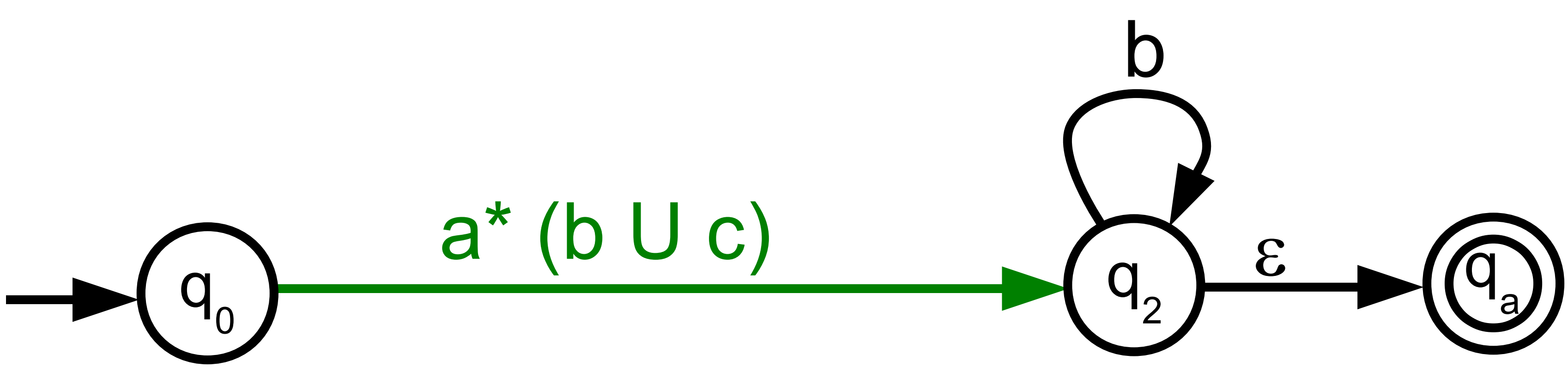
Don't forget: no arrow means label Ø



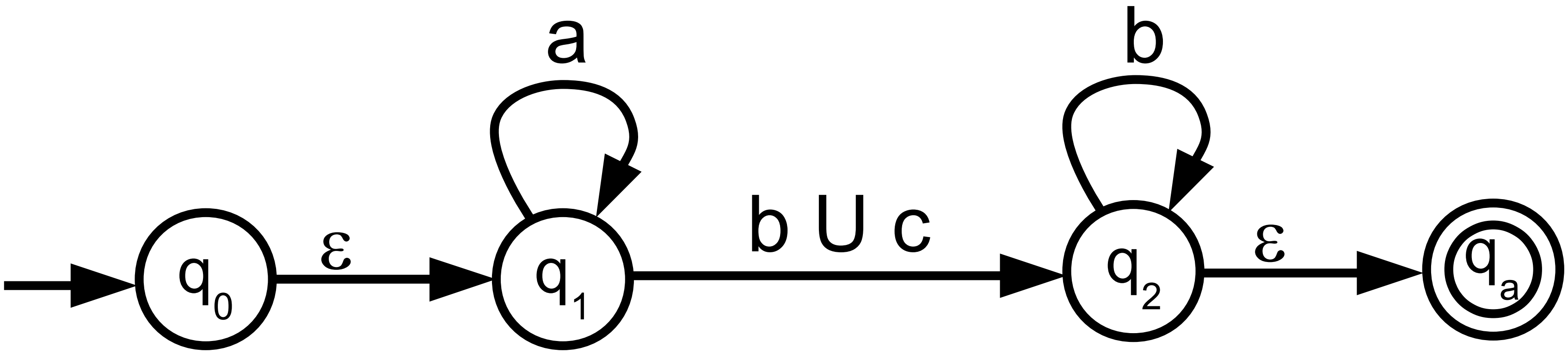
Example: DFA → GNFA → RE



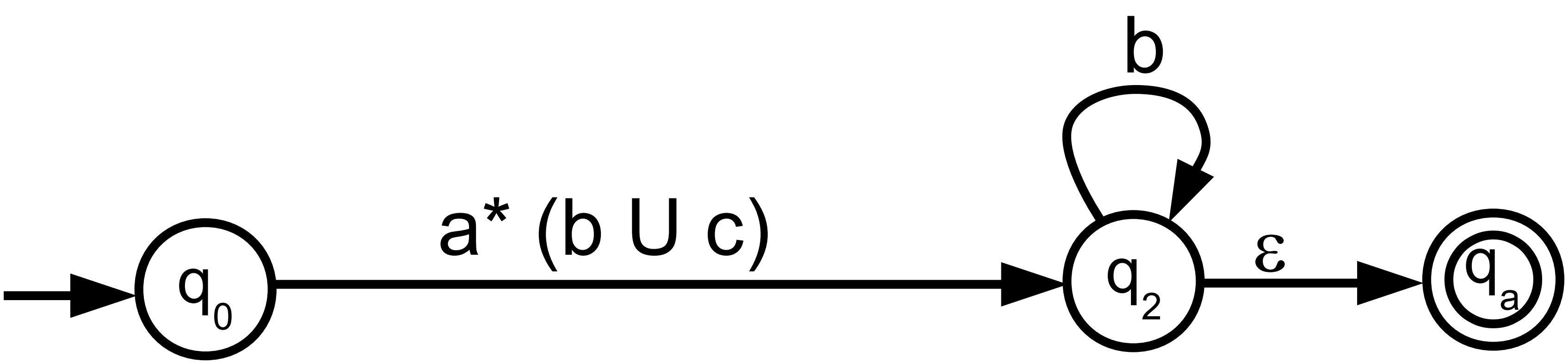
Eliminate q_1: simplify RE on new edge



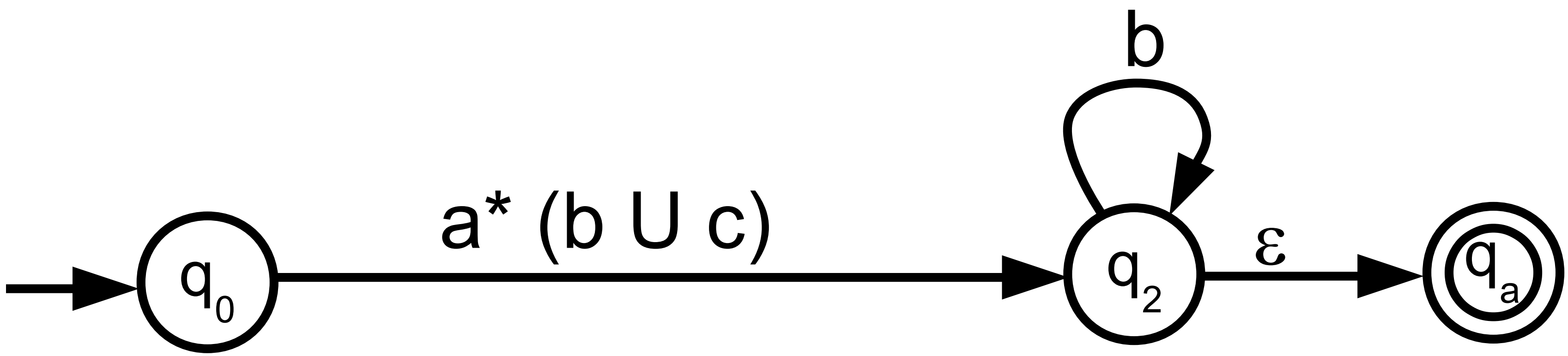
Example: DFA → GNFA → RE



Eliminate q_1: if no more paths through q_1, start over

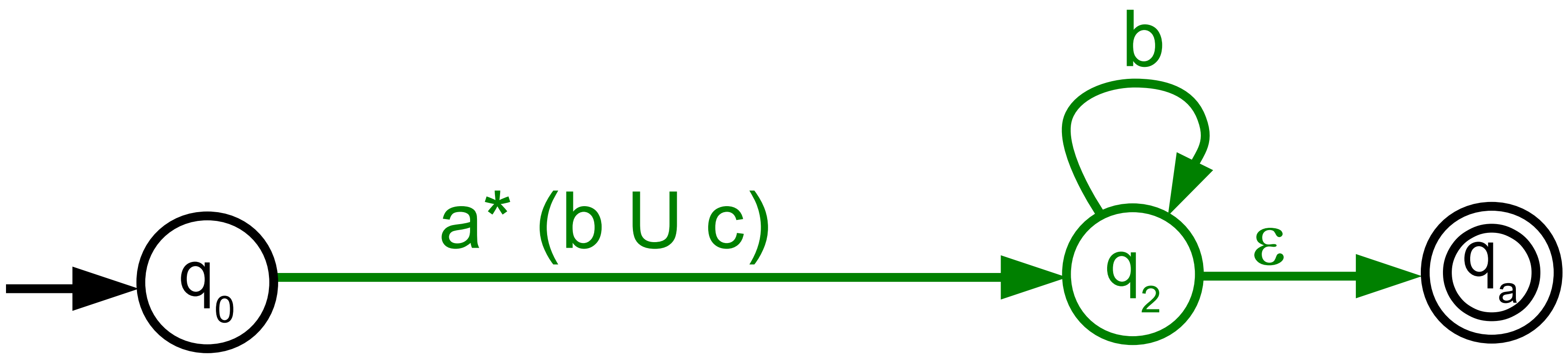


Example: DFA → GNFA → RE



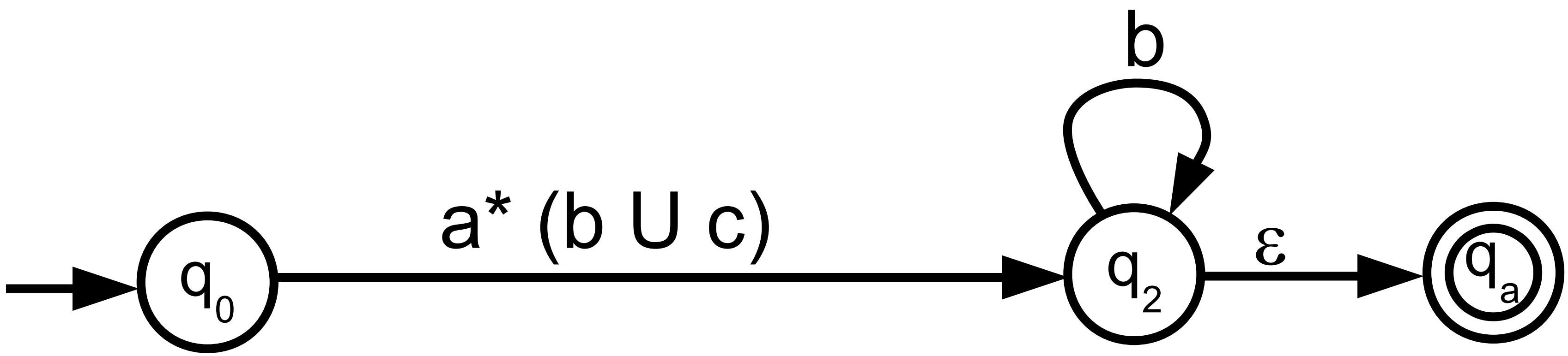
Eliminate q_2: re-draw GNFA with all other states

Example: DFA → GNFA → RE

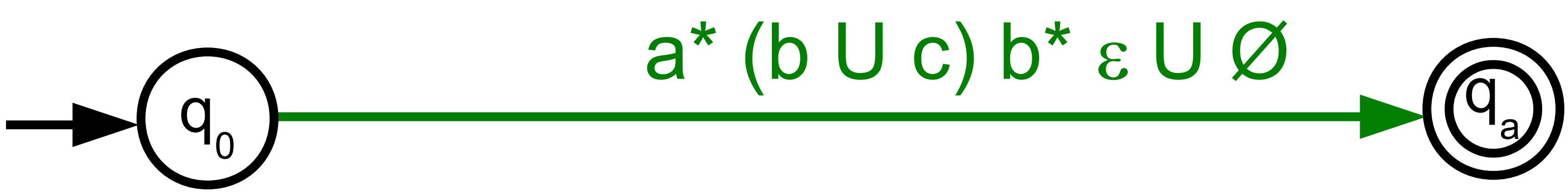


Eliminate q_2: find a path through q_2

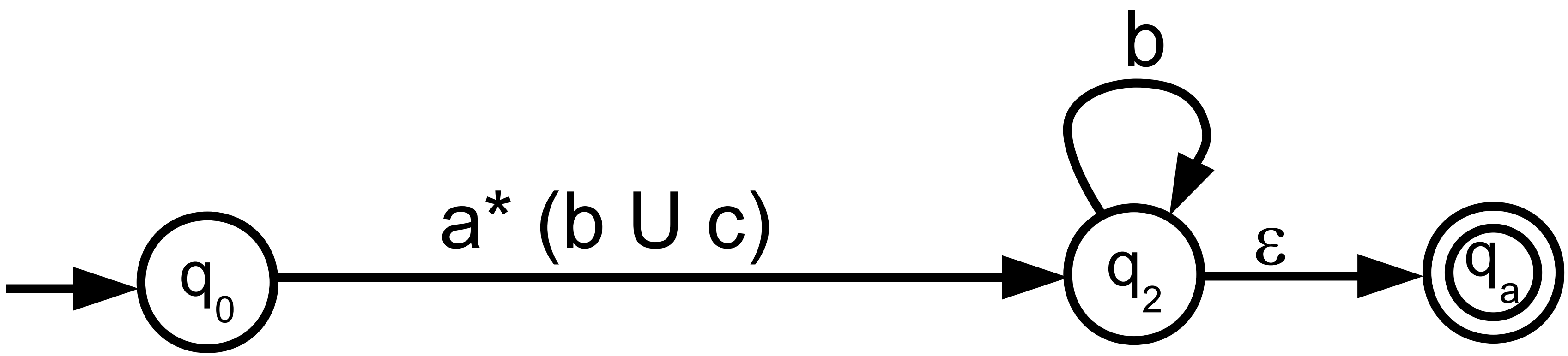
Example: DFA → GNFA → RE



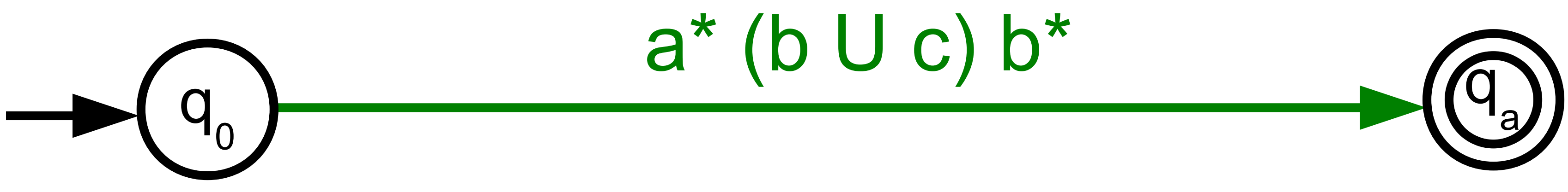
Eliminate q_2: add edge to new GNFA



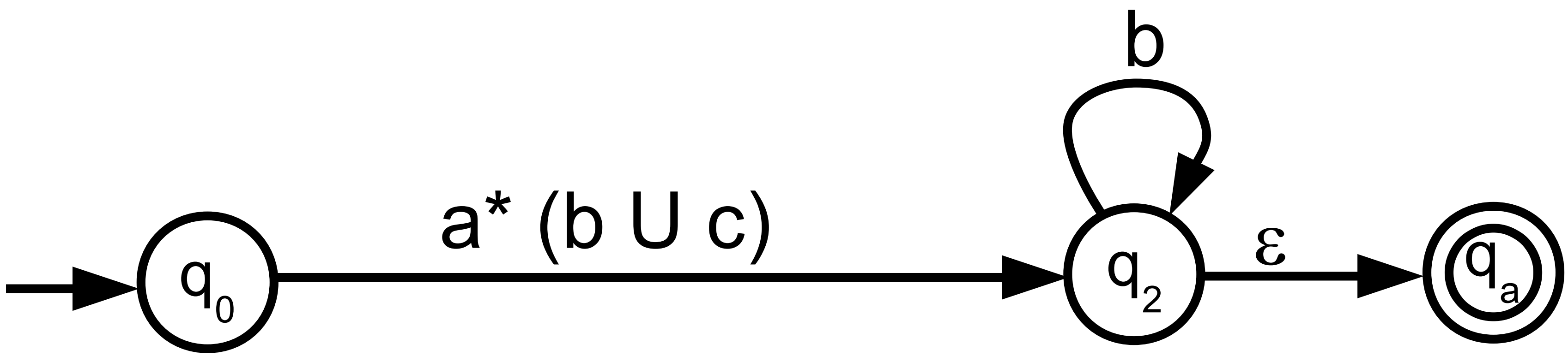
Example: DFA → GNFA → RE



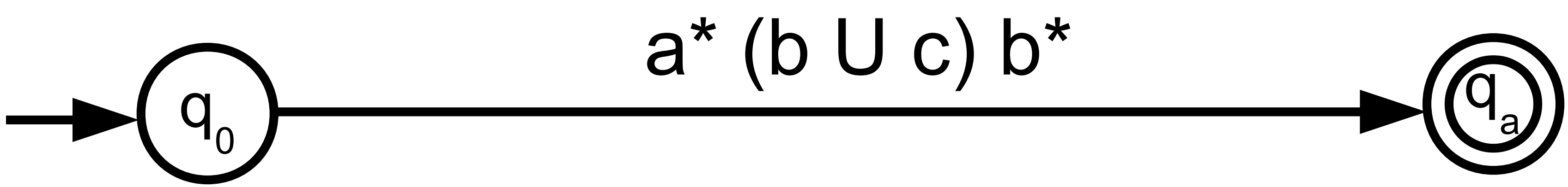
Eliminate q_2: simplify RE on new edge



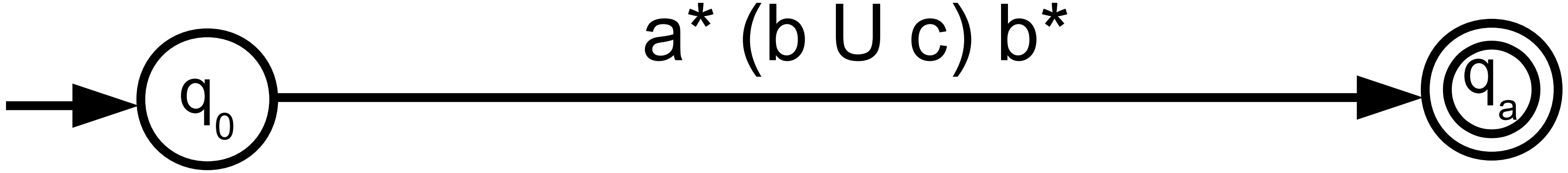
Example: DFA → GNFA → RE



Eliminate q_2: if no more paths through q_2, start over



Example: DFA → GNFA → RE

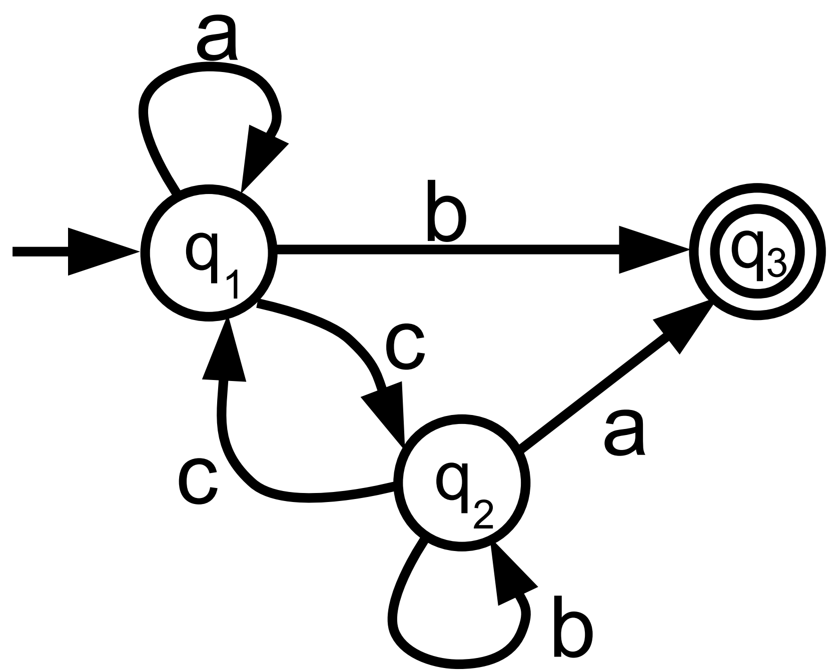


Only two states remain:

RE = a* (b U c) b*

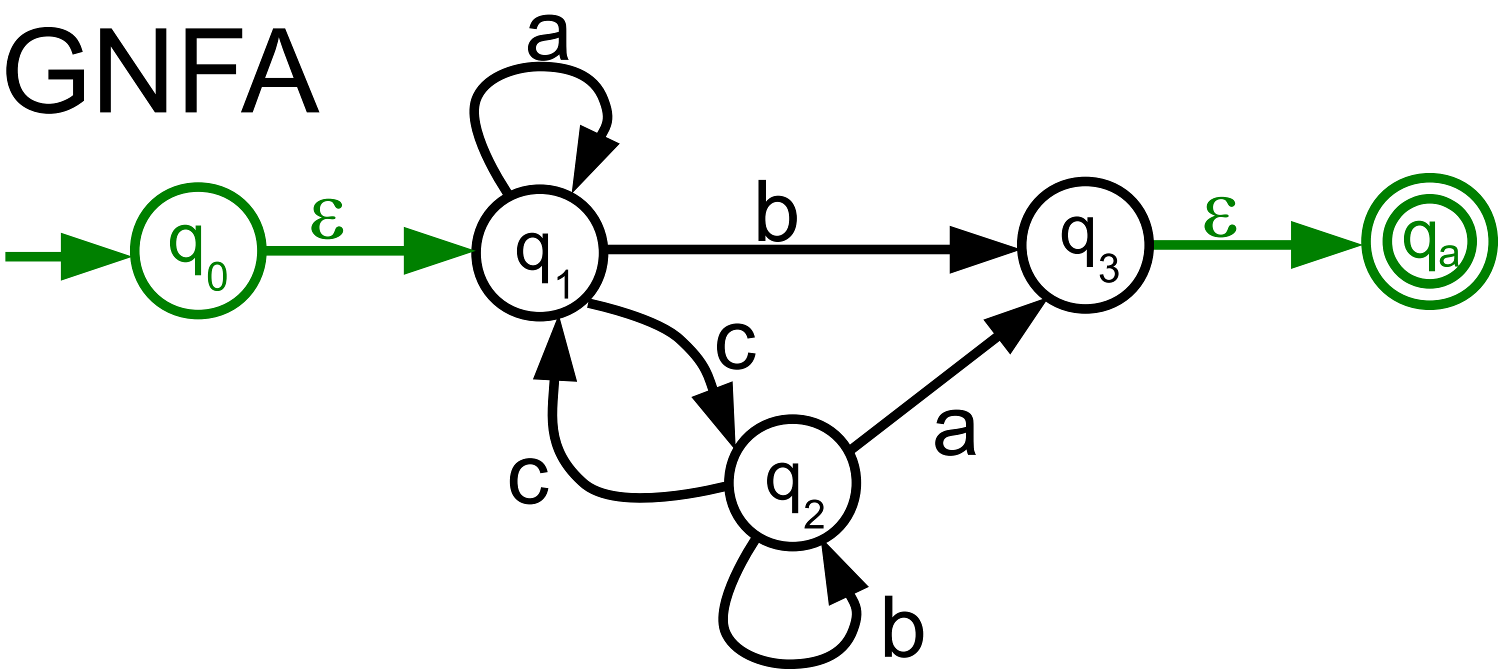
ANOTHER Example: DFA → GNFA → RE

DFA

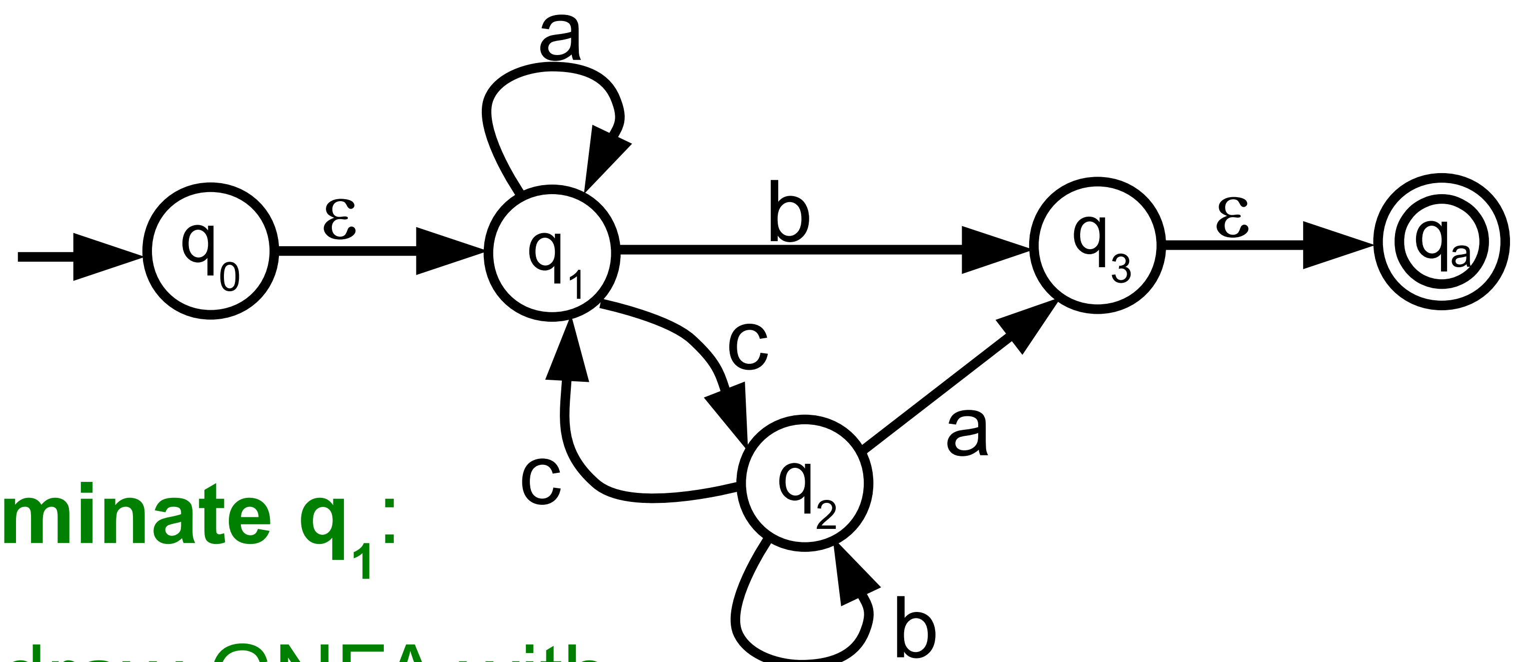


ANOTHER Example: DFA → GNFA → RE

GNFA



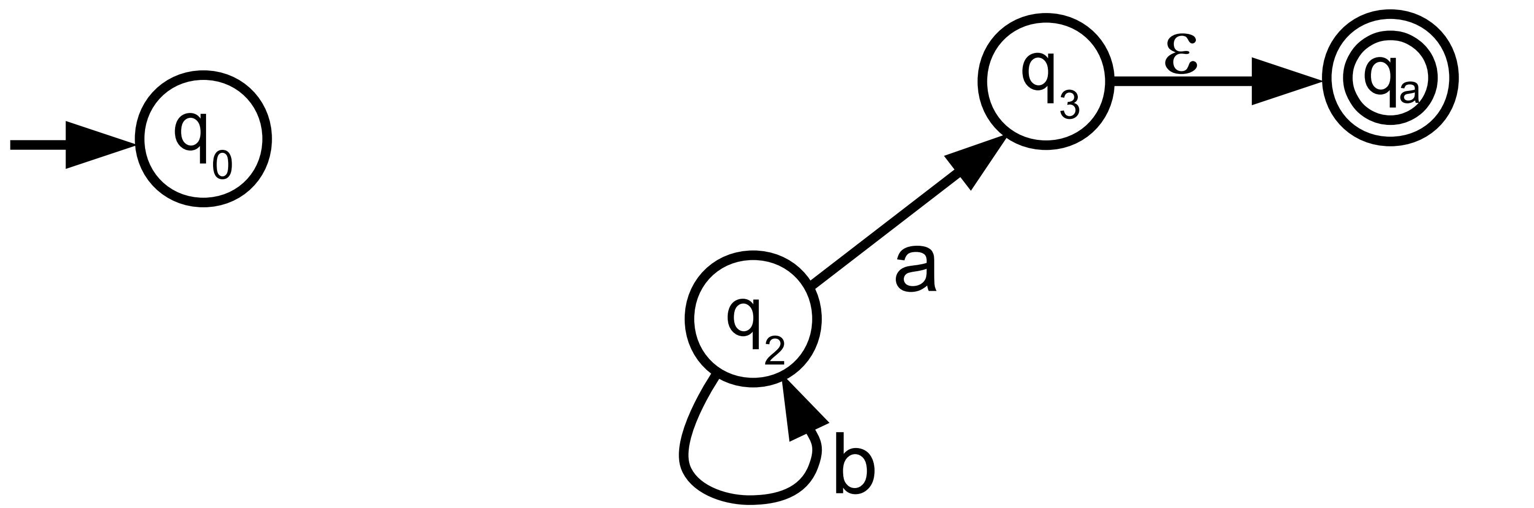
ANOTHER Example: DFA → GNFA → RE



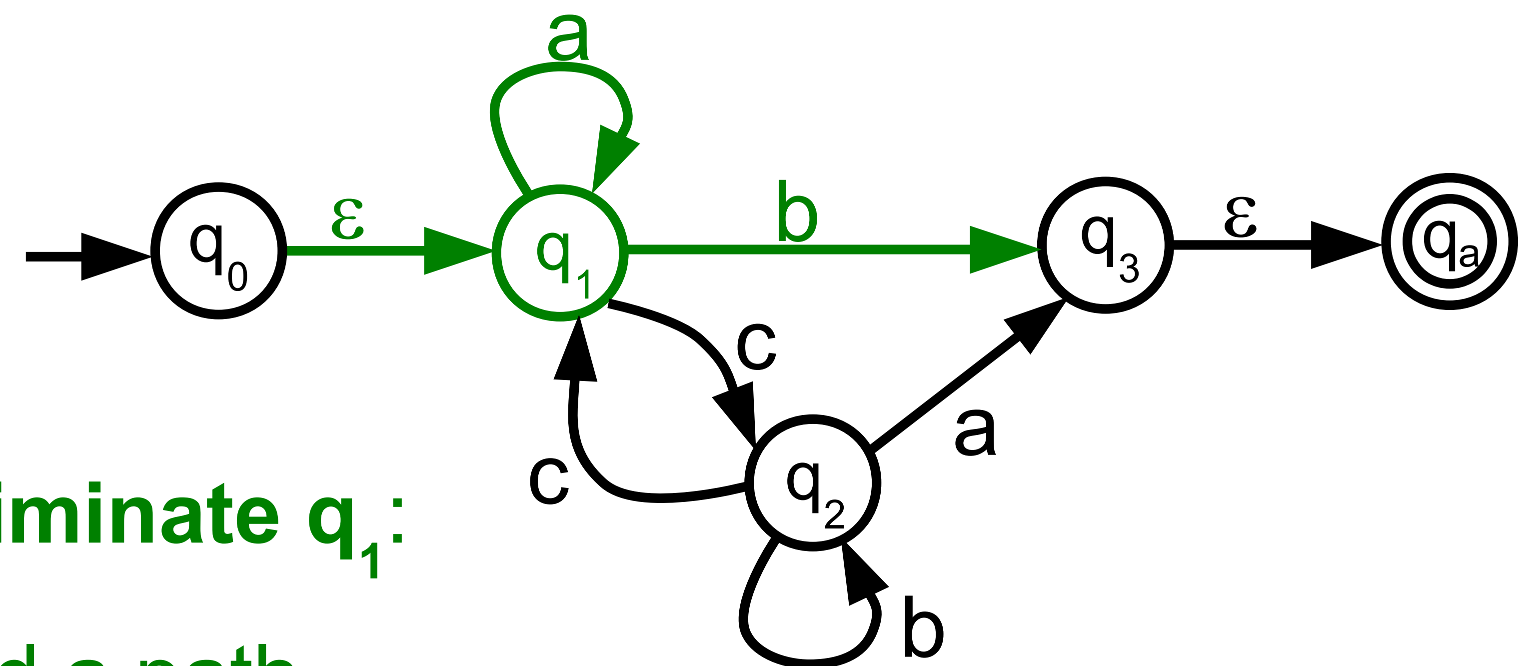
Eliminate q_1:

re-draw GNFA with

all other states

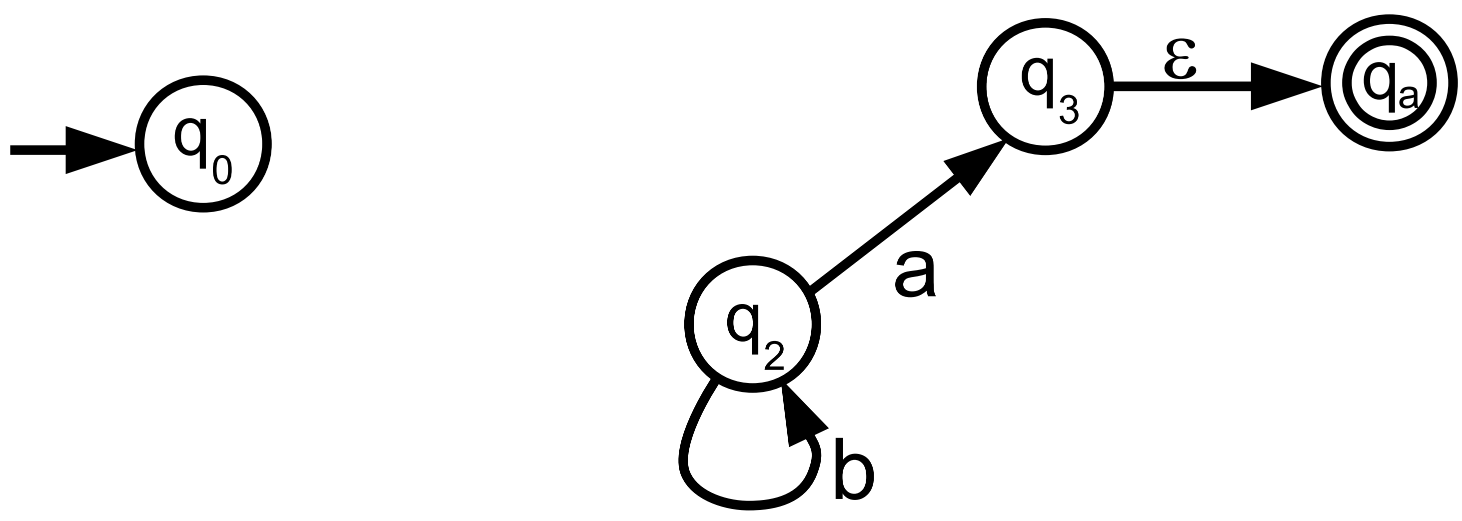


ANOTHER Example: DFA → GNFA → RE

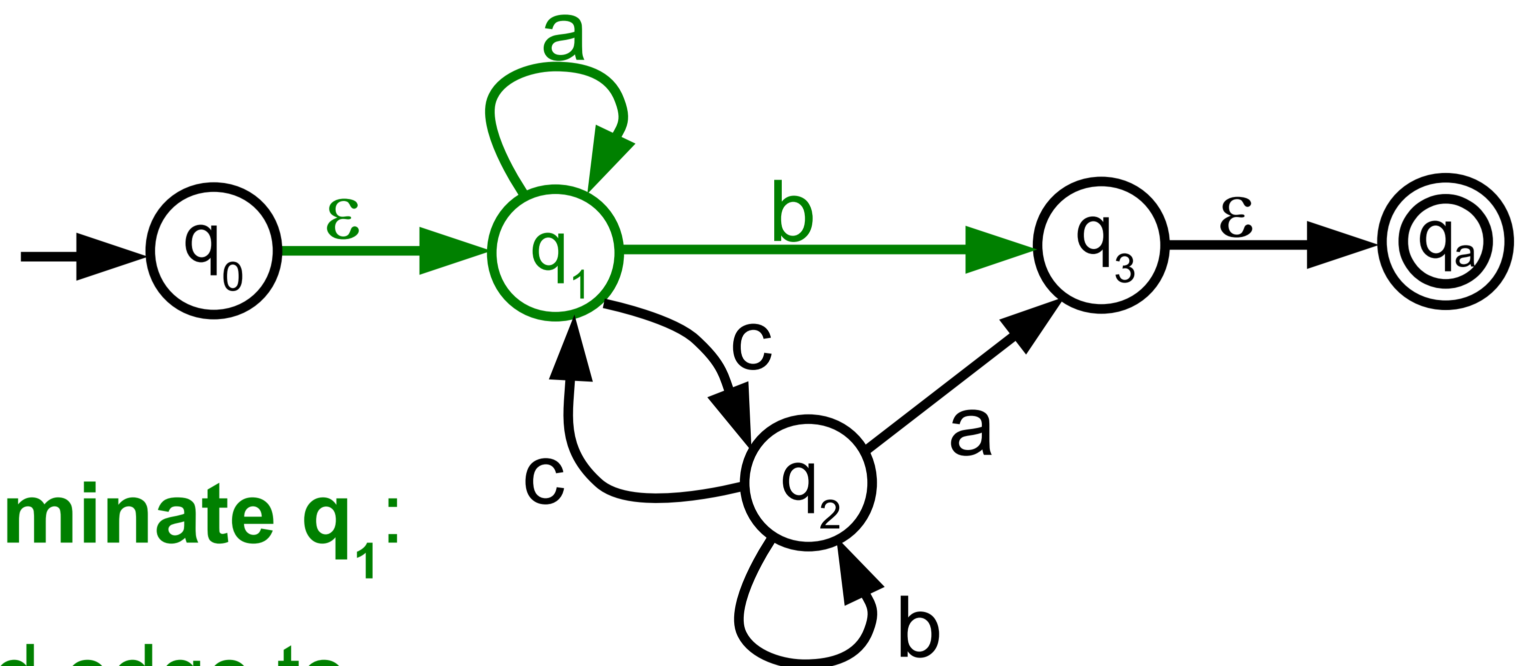


Eliminate q_1:

find a path through q_1



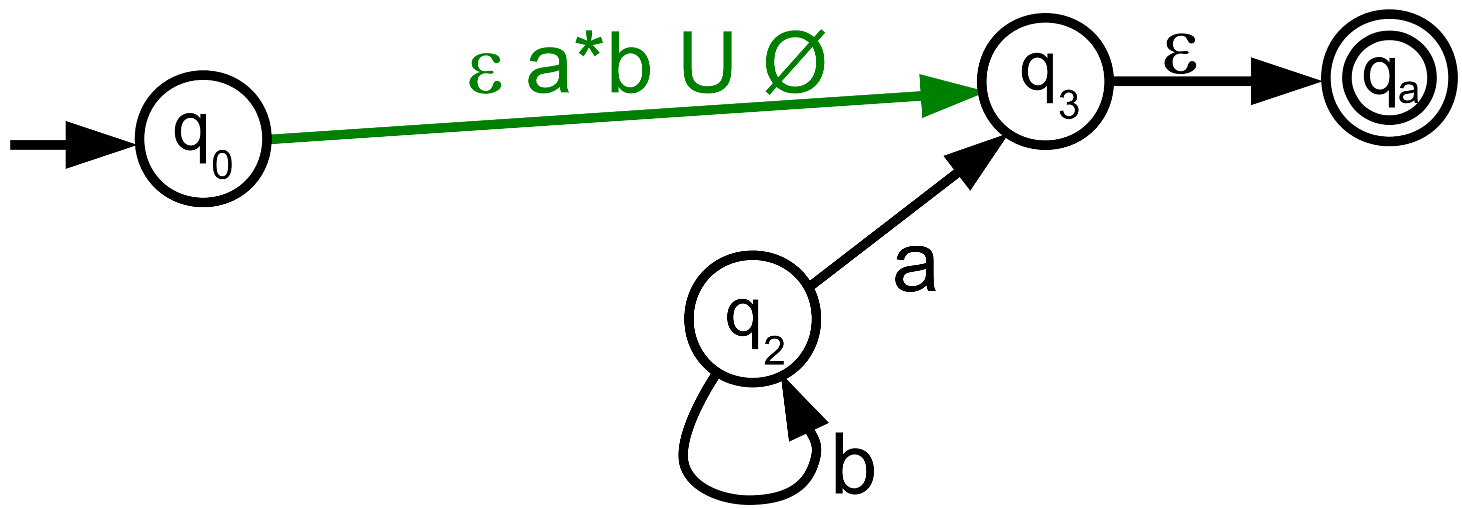
ANOTHER Example: DFA → GNFA → RE



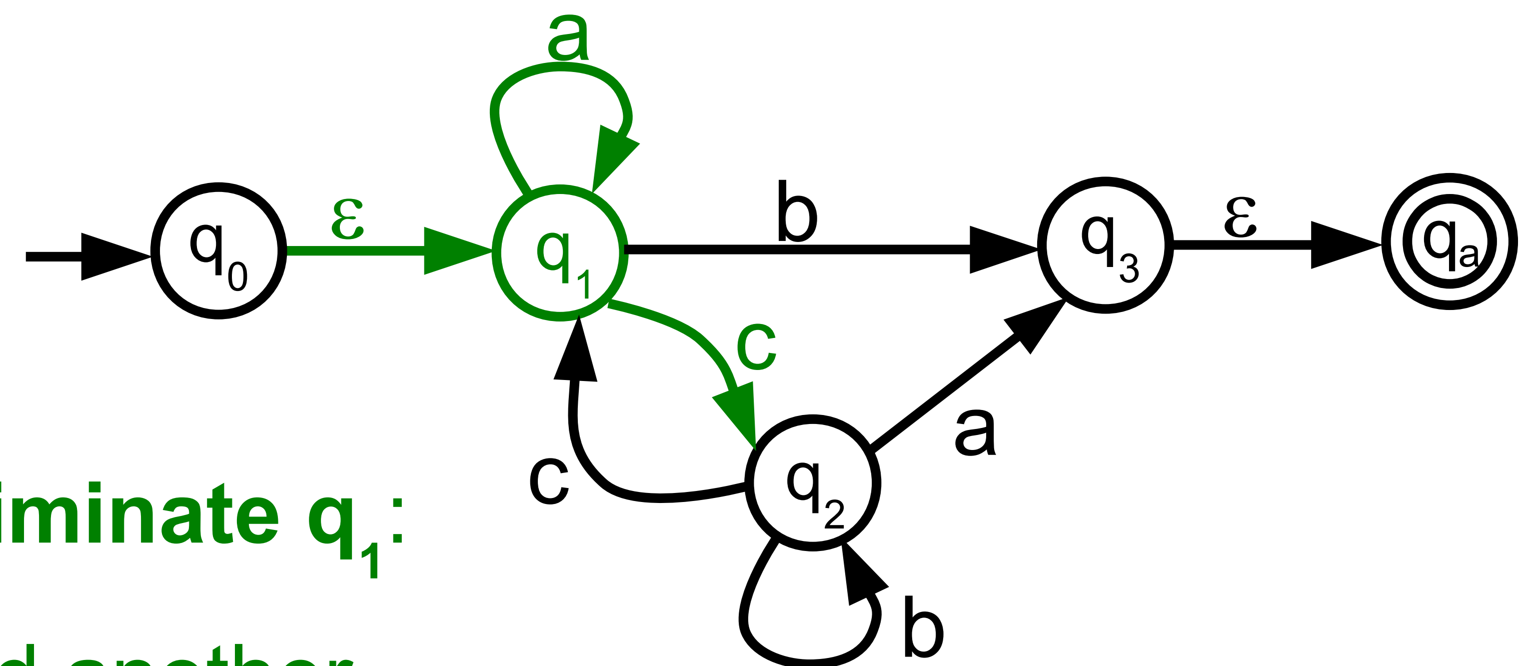
Eliminate q_1:

add edge to

new GNFA

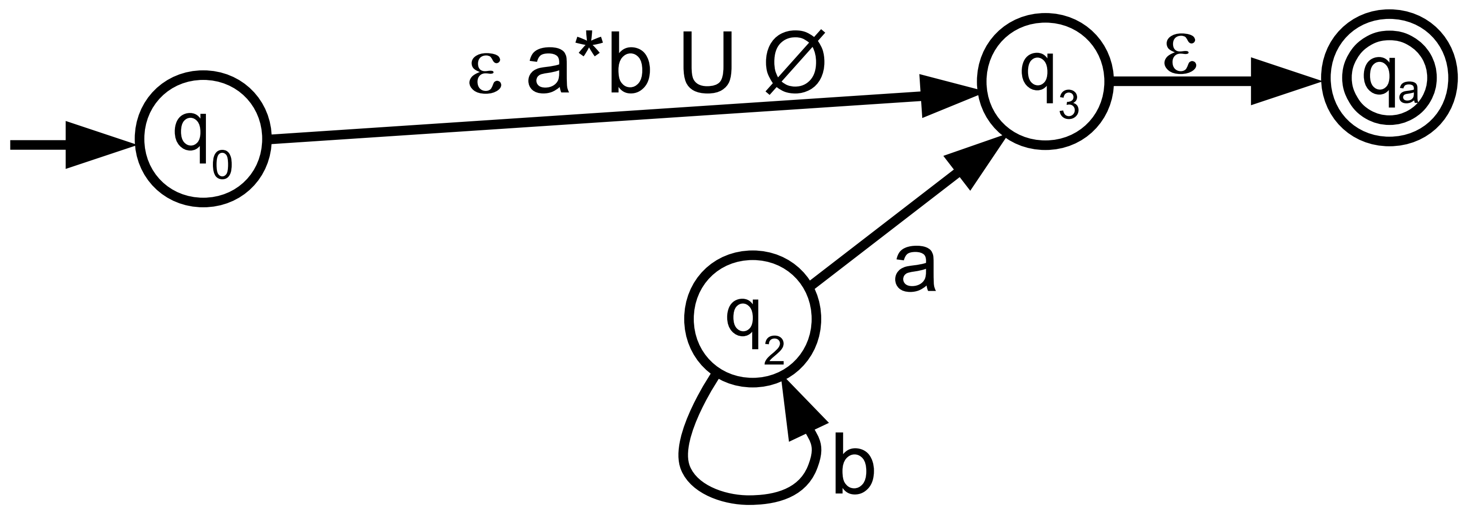


ANOTHER Example: DFA → GNFA → RE

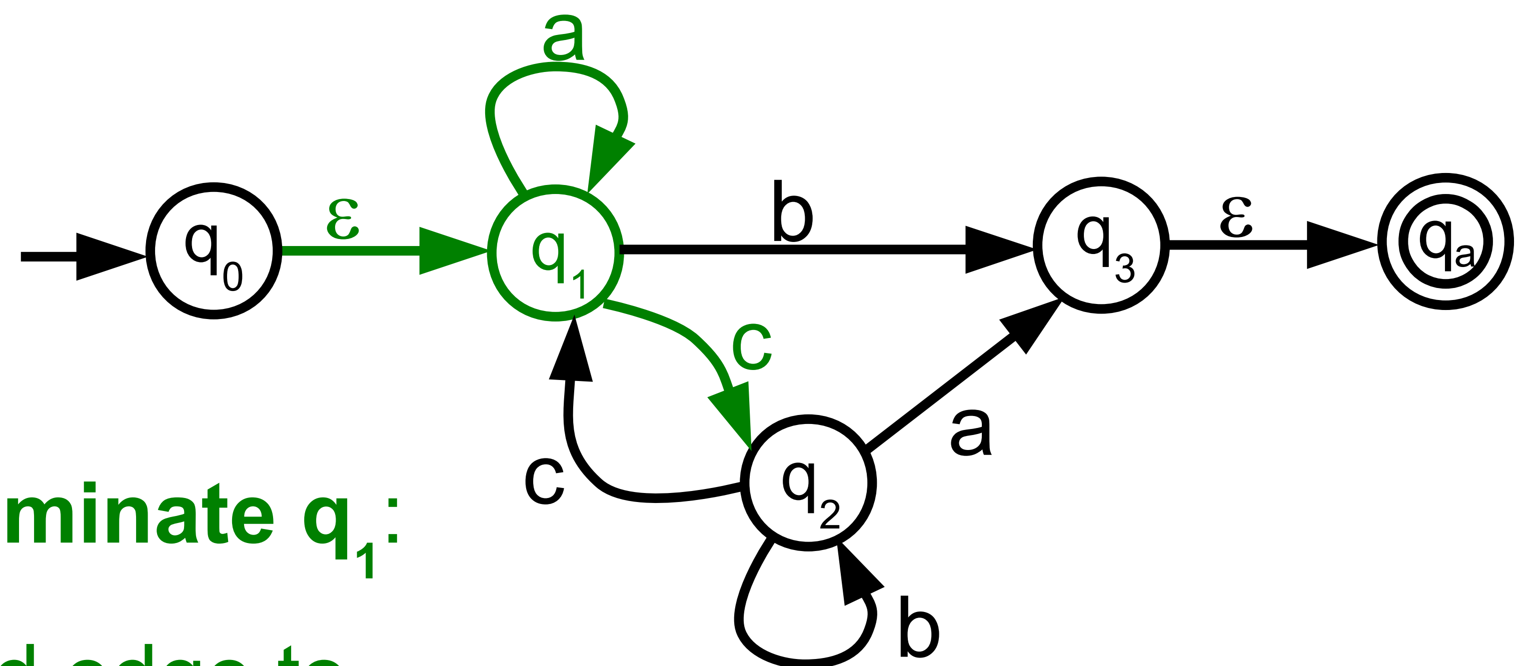


Eliminate q_1:

find another path through q_1

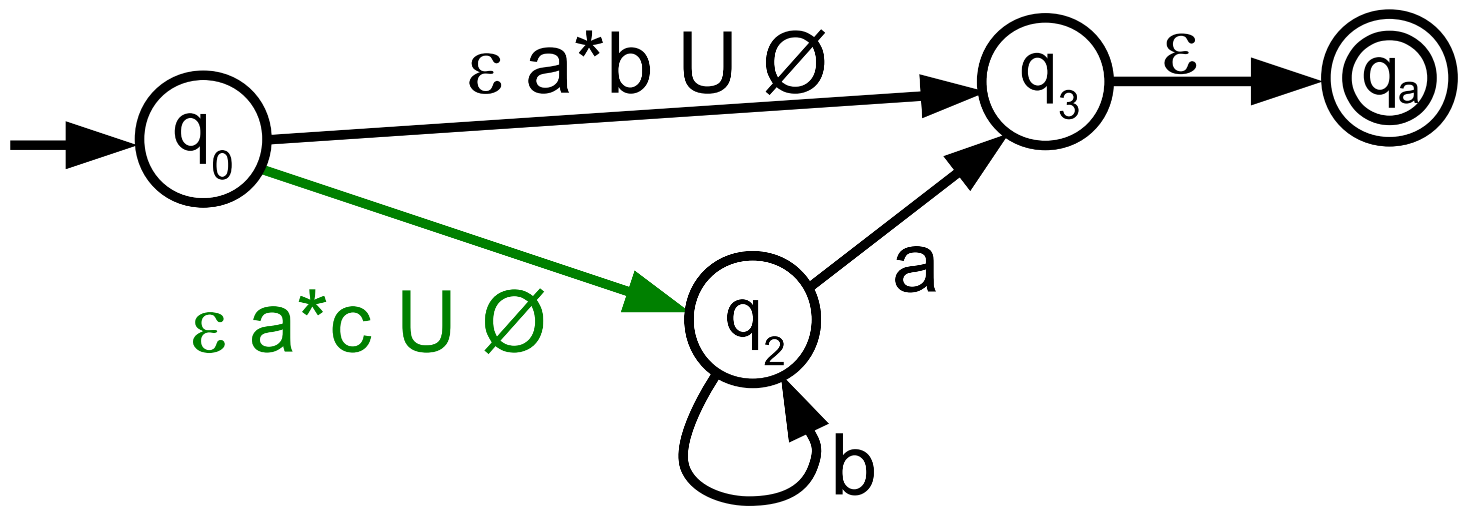


ANOTHER Example: DFA → GNFA → RE

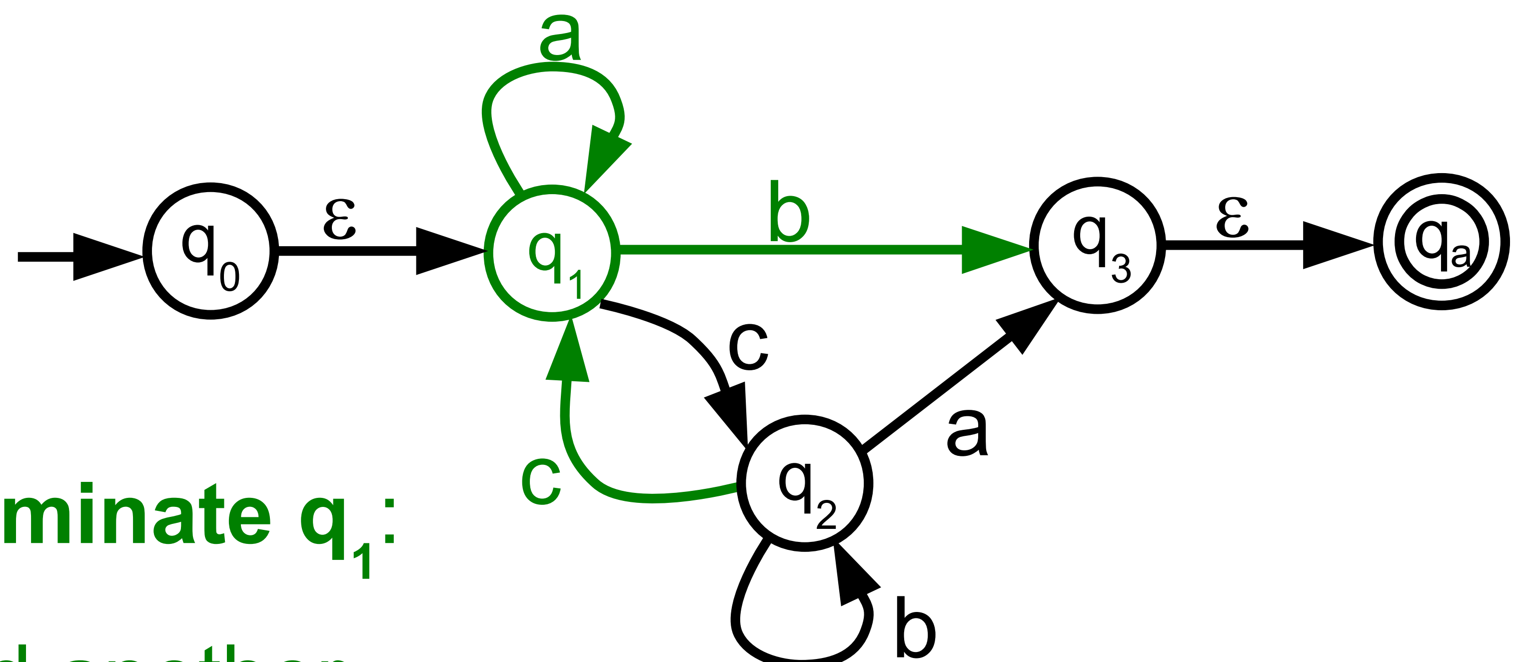


Eliminate q_1:

add edge to new GNFA

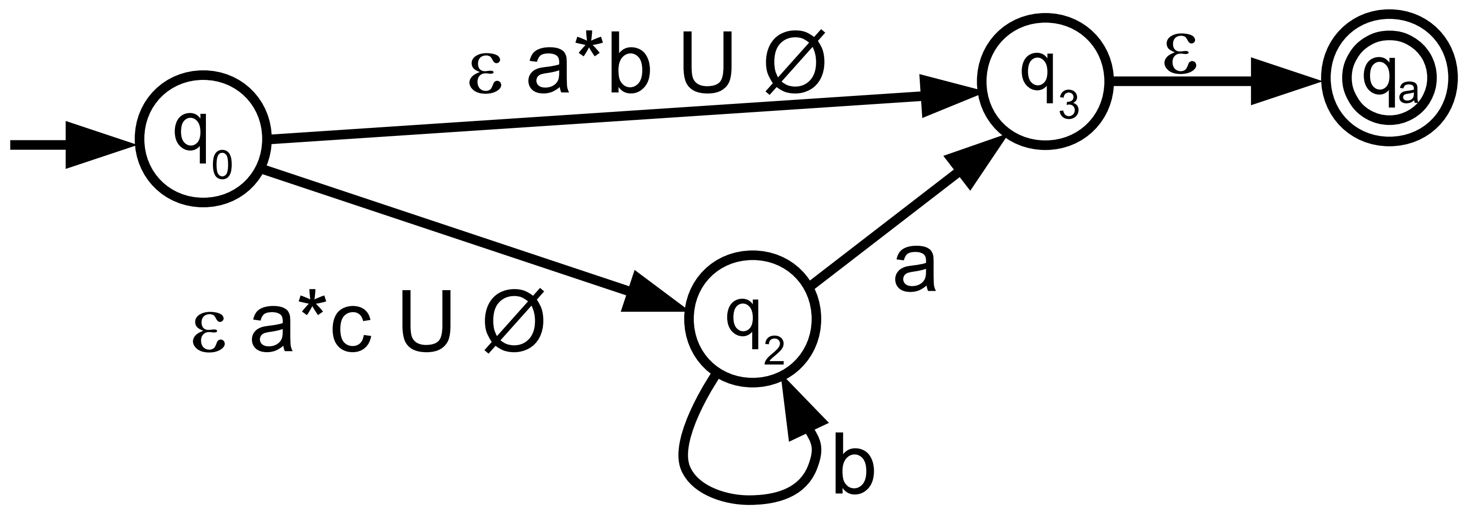


ANOTHER Example: DFA → GNFA → RE

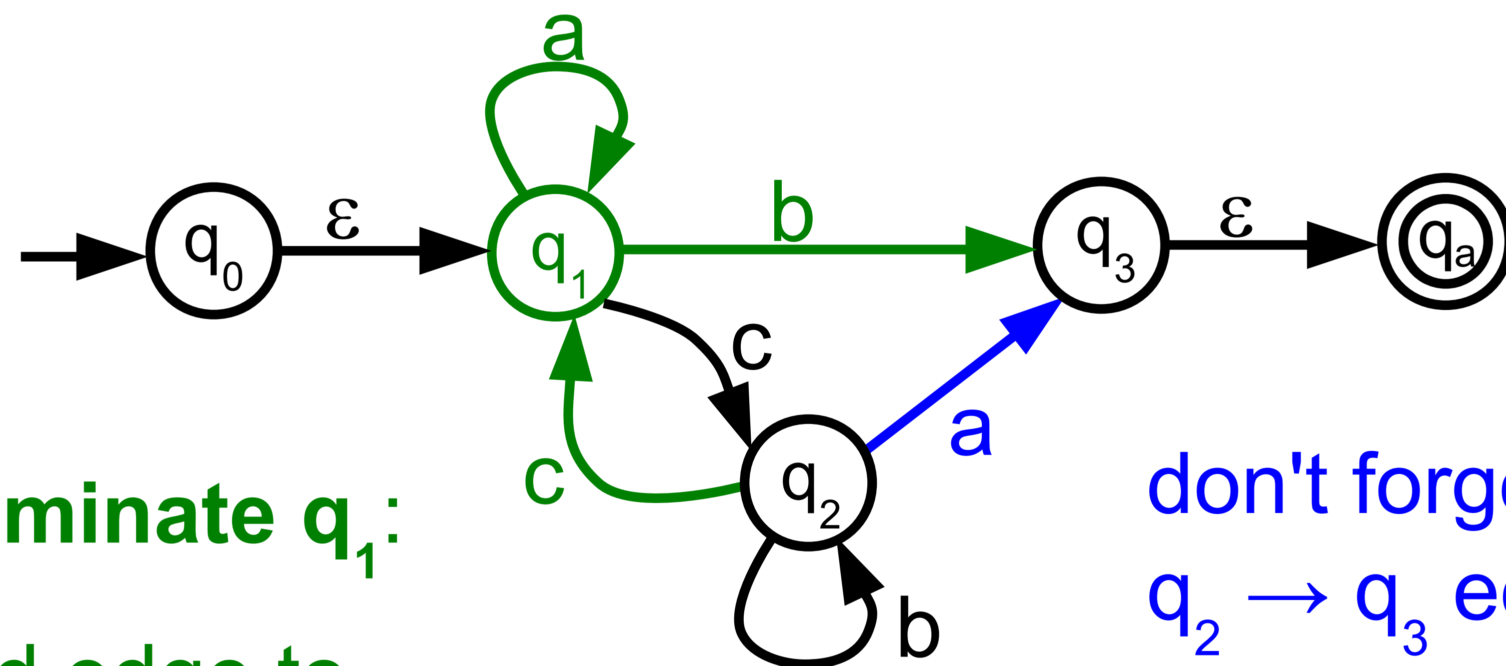


Eliminate q_1:

find another path through q_1



ANOTHER Example: DFA → GNFA → RE

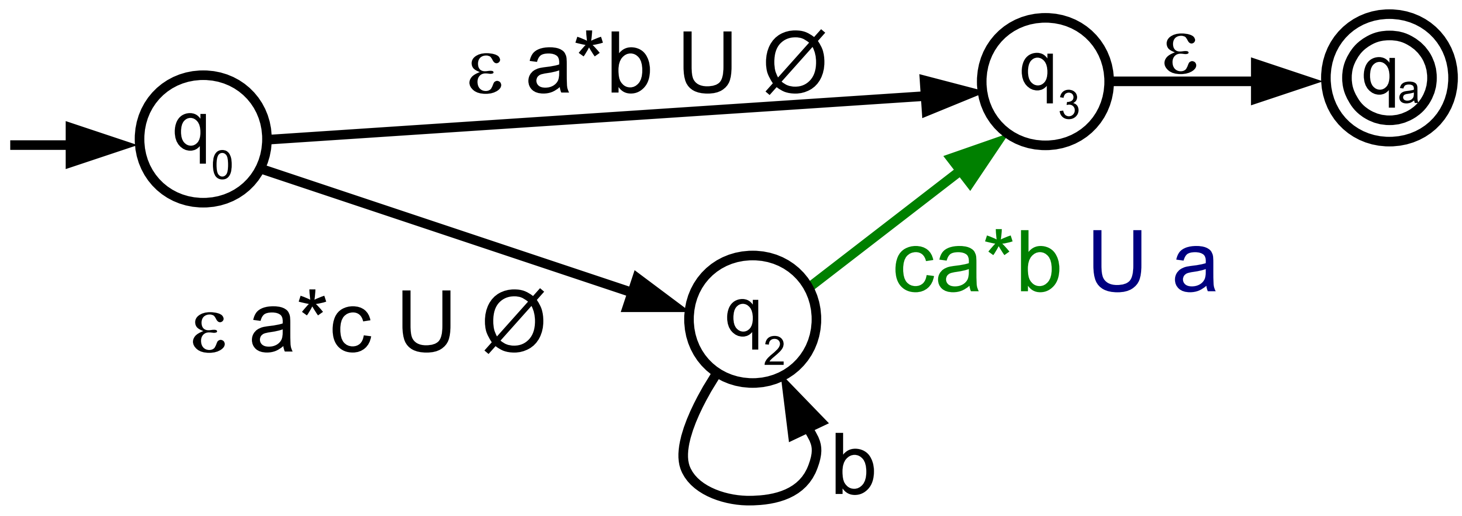


Eliminate q_1:

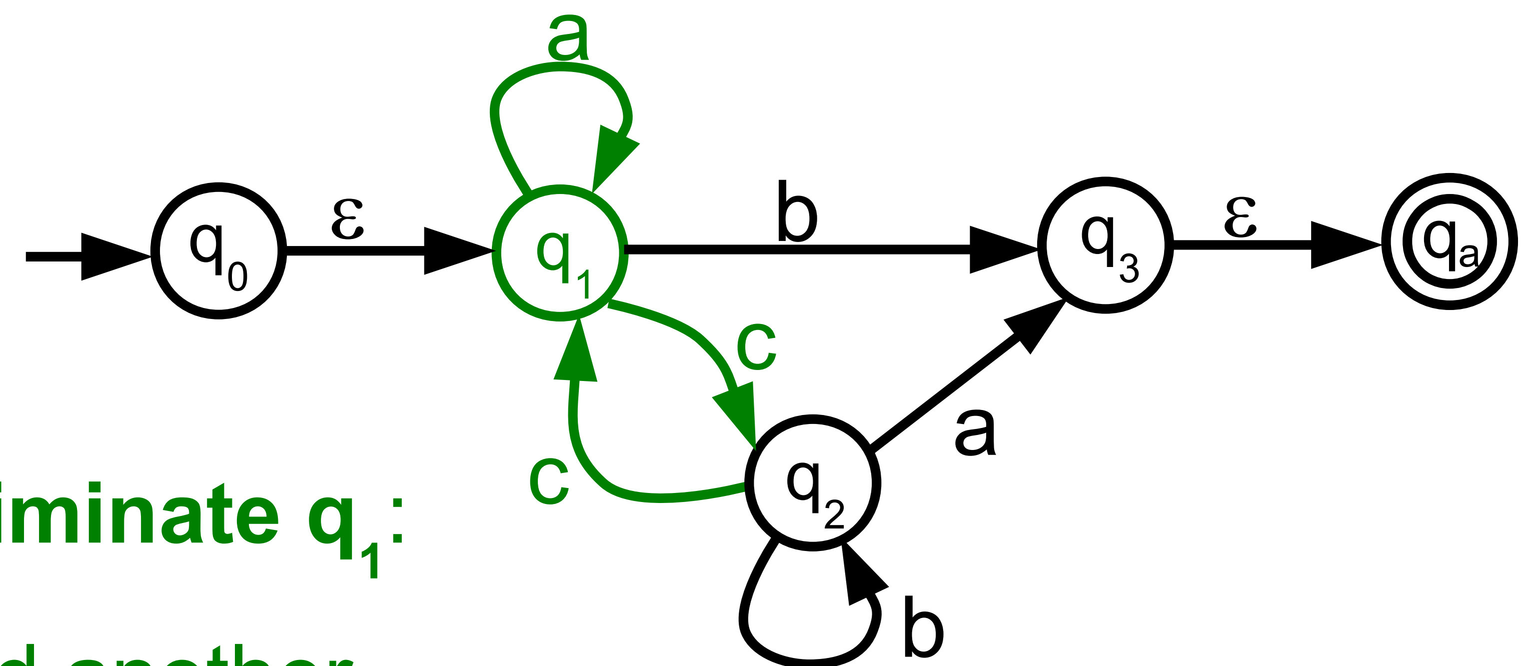
add edge to new GNFA

don't forget current $q_2 \rightarrow q_3$ edge!

This time is not Ø !

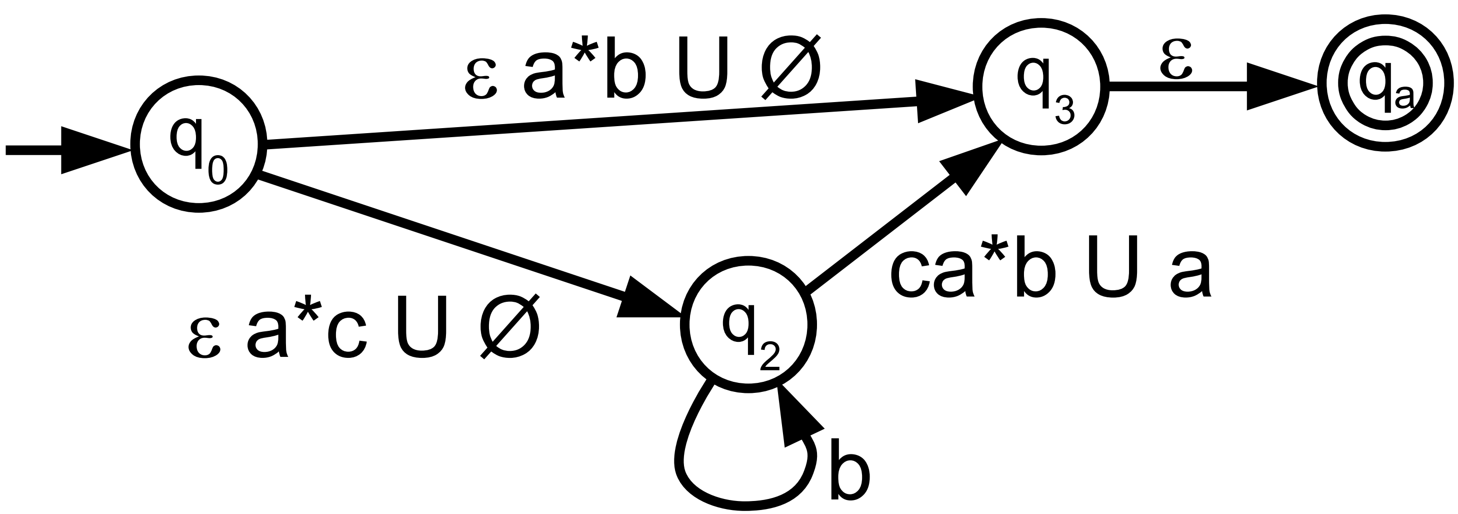


ANOTHER Example: DFA → GNFA → RE

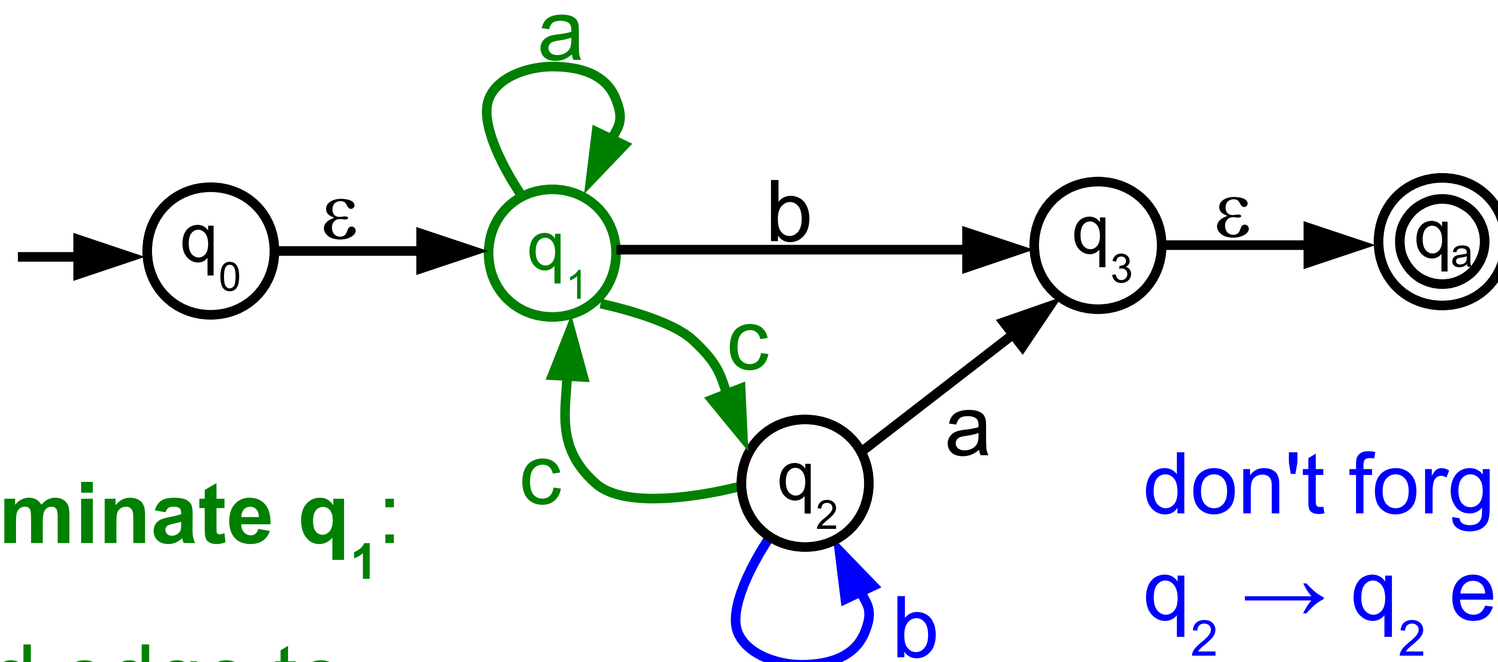


Eliminate q_1:

find another path through q_1



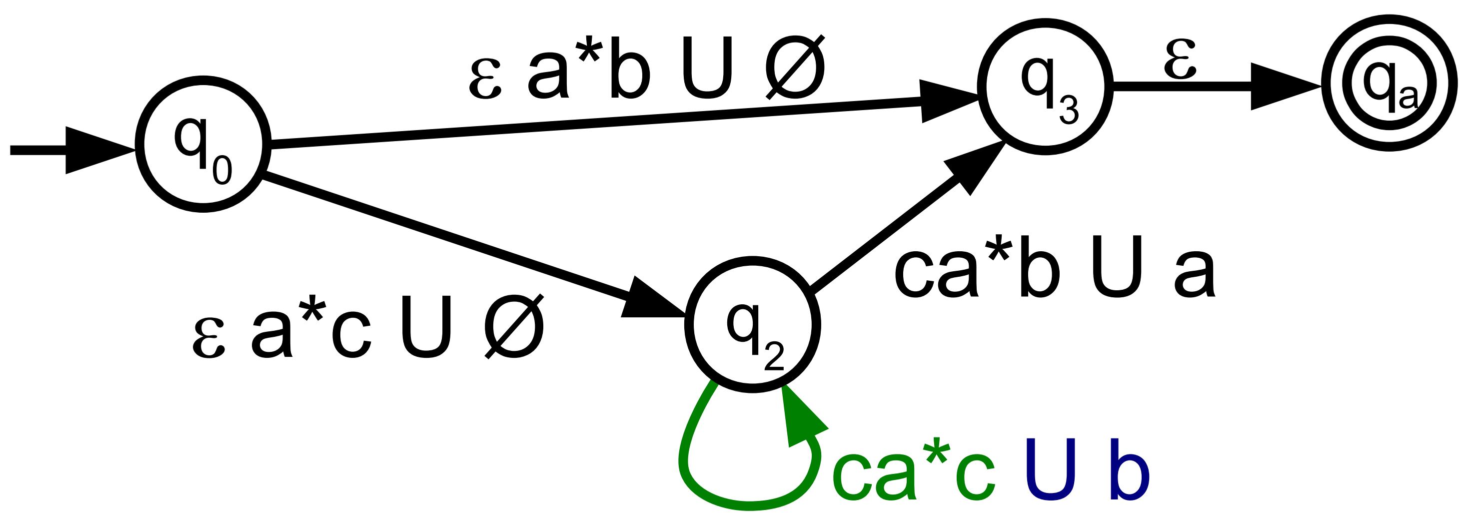
ANOTHER Example: DFA → GNFA → RE



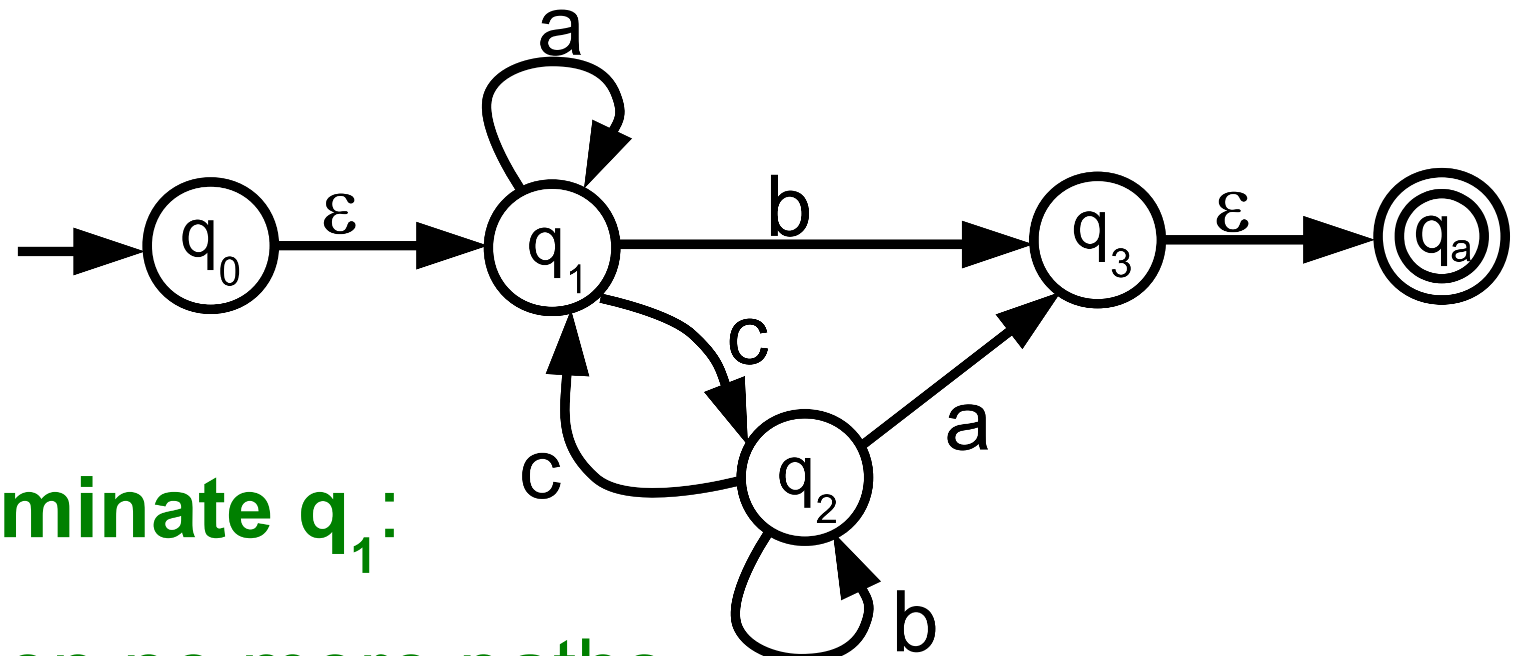
Eliminate q_1:

add edge to new GNFA

don't forget current $q_2 \to q_2$ edge!



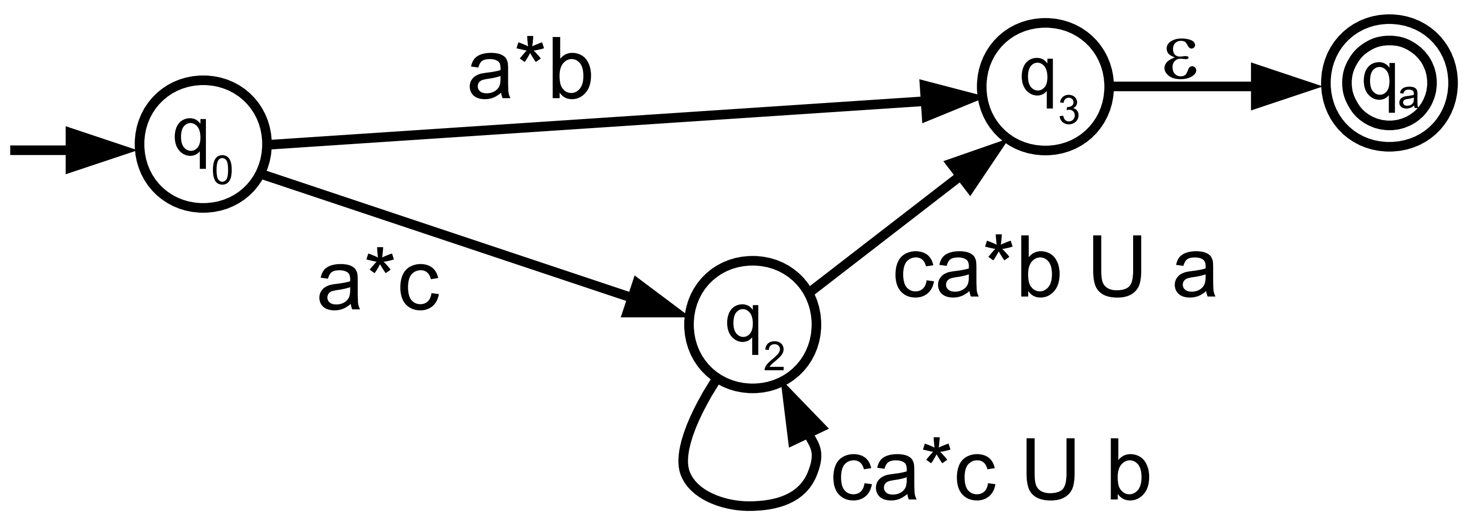
ANOTHER Example: DFA → GNFA → RE



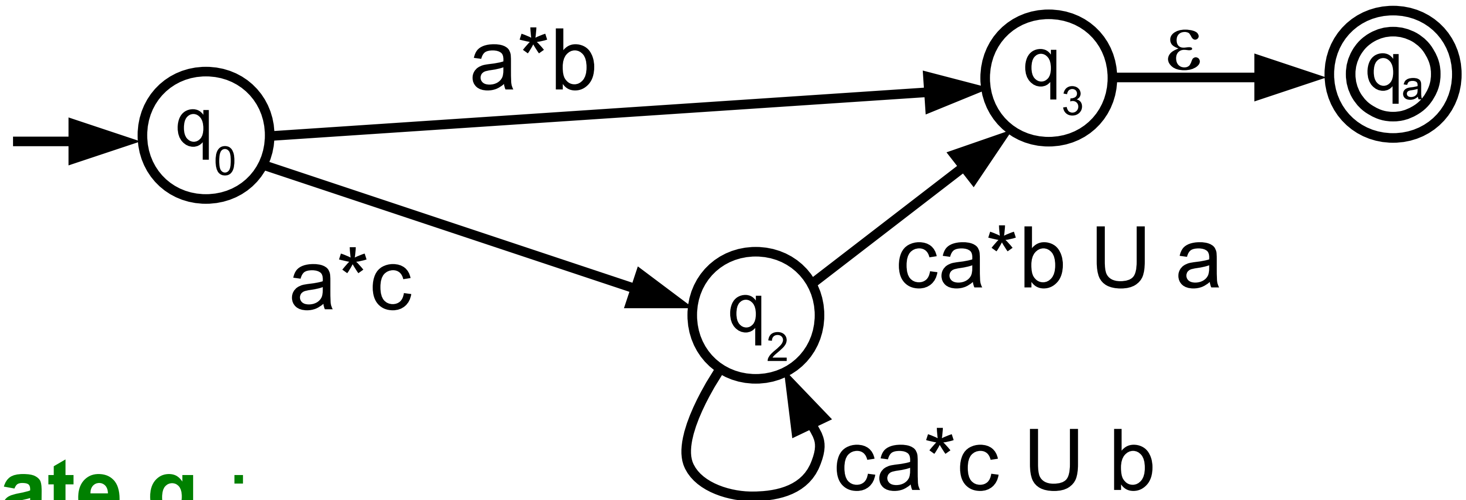
Eliminate q_1:

when no more paths through q_1, start over

(and simplify REs)



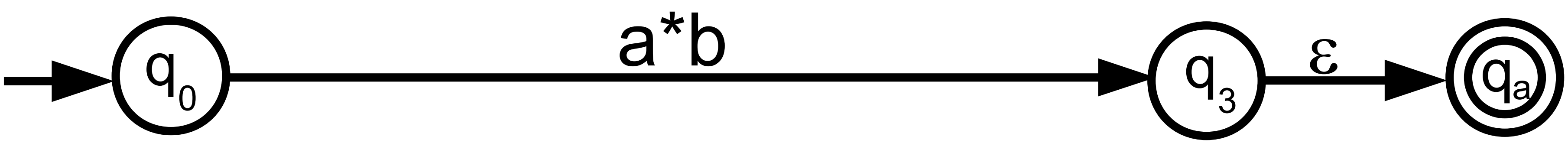
ANOTHER Example: DFA → GNFA → RE



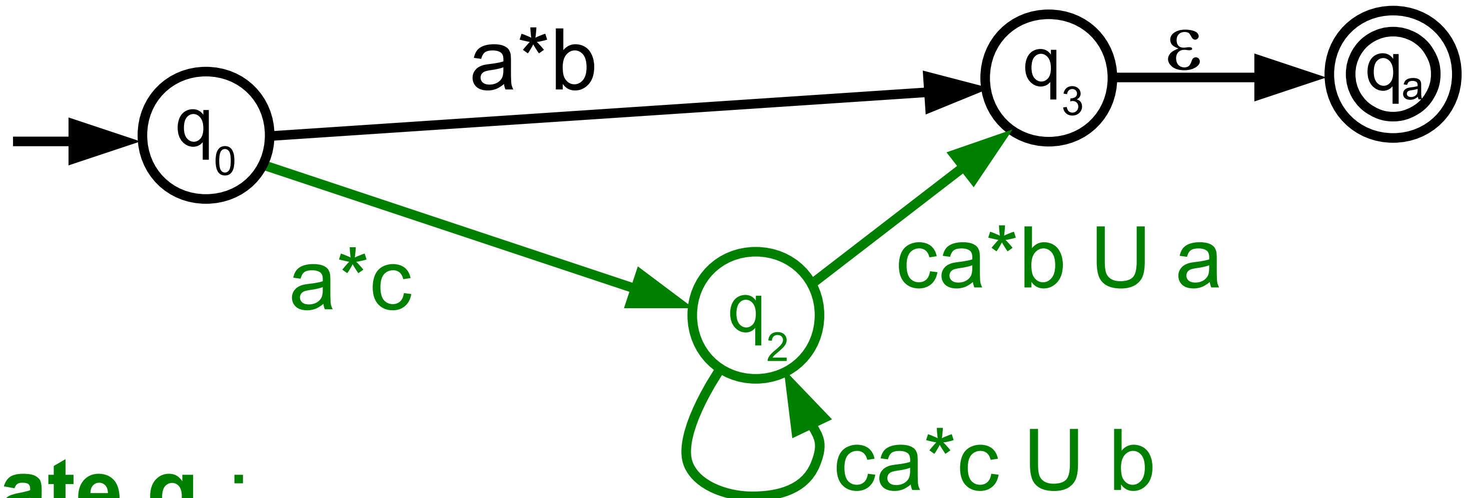
Eliminate q_2:

re-draw GNFA with

all other states

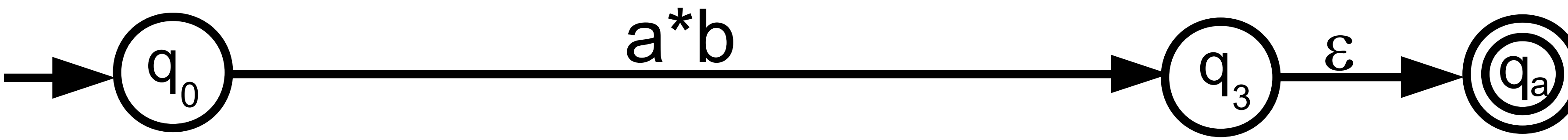


ANOTHER Example: DFA → GNFA → RE

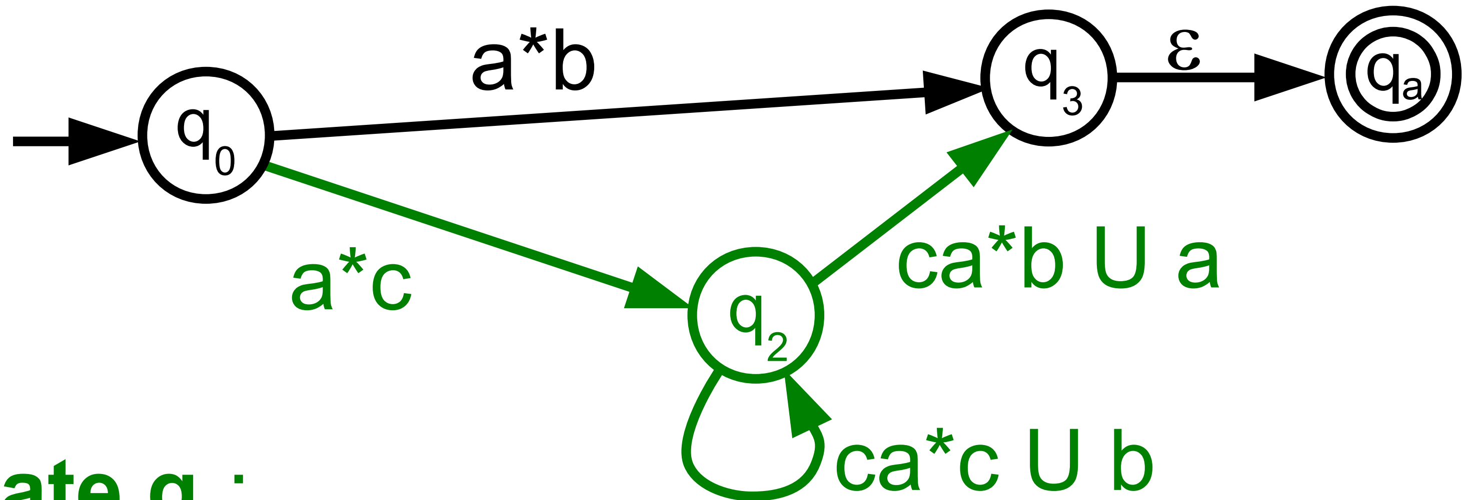


Eliminate q_2:

find a path through q_2

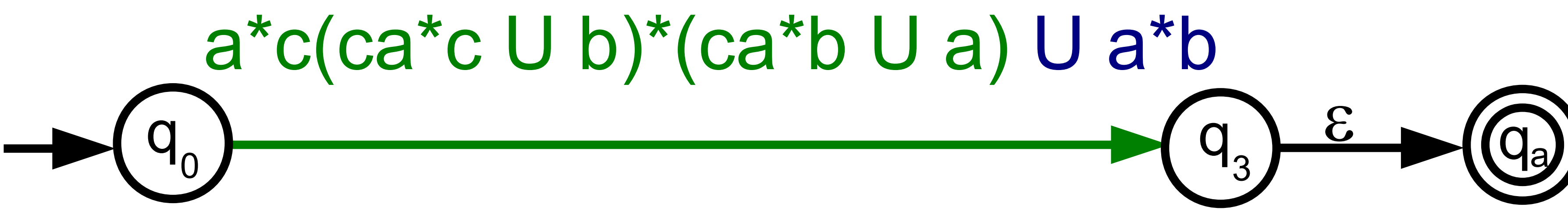


ANOTHER Example: DFA → GNFA → RE

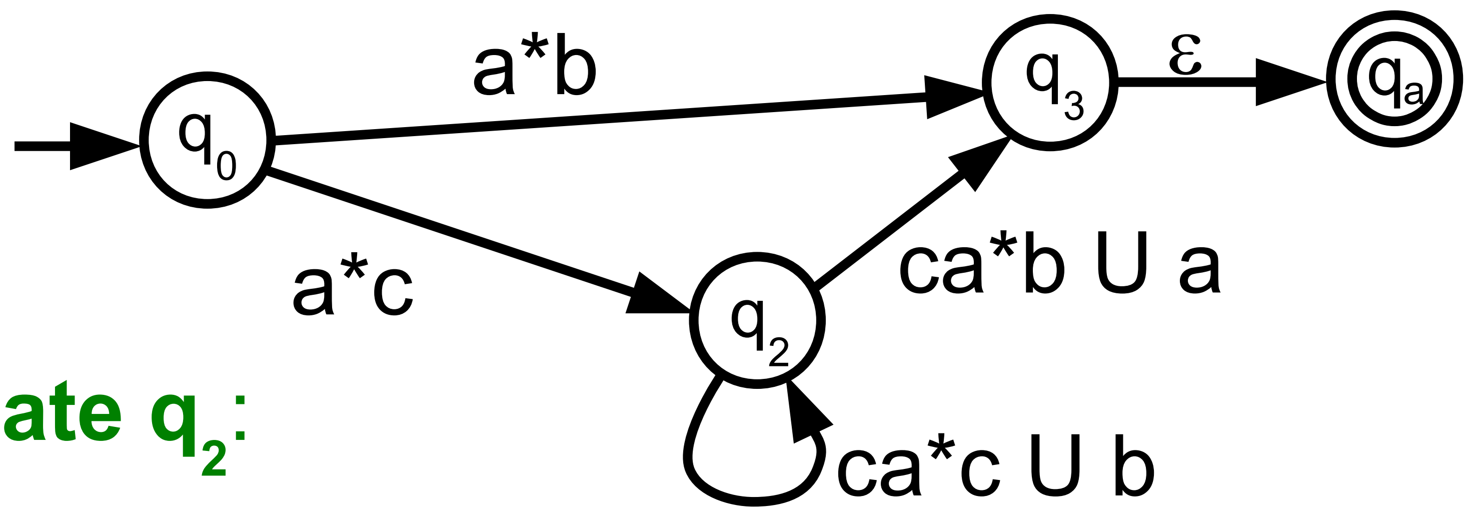


Eliminate q_2:

add edge to new GNFA

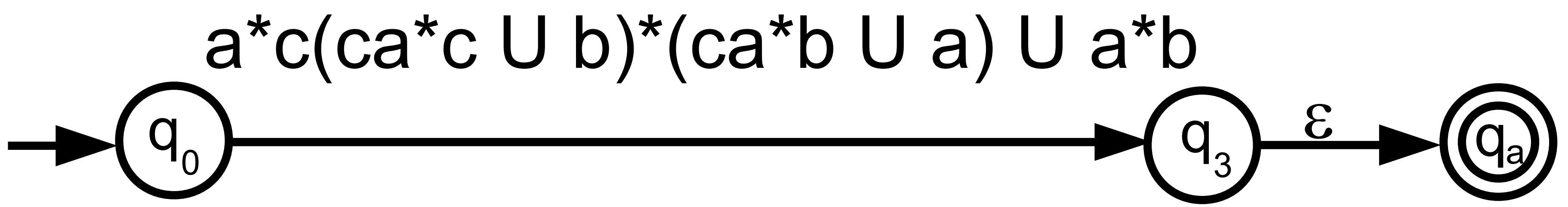


ANOTHER Example: DFA → GNFA → RE

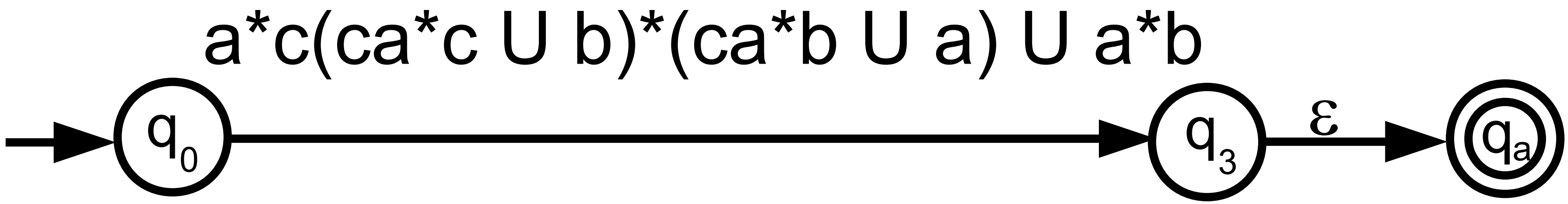


Eliminate q_2:

when no more paths through q_2, start over



ANOTHER Example: DFA → GNFA → RE

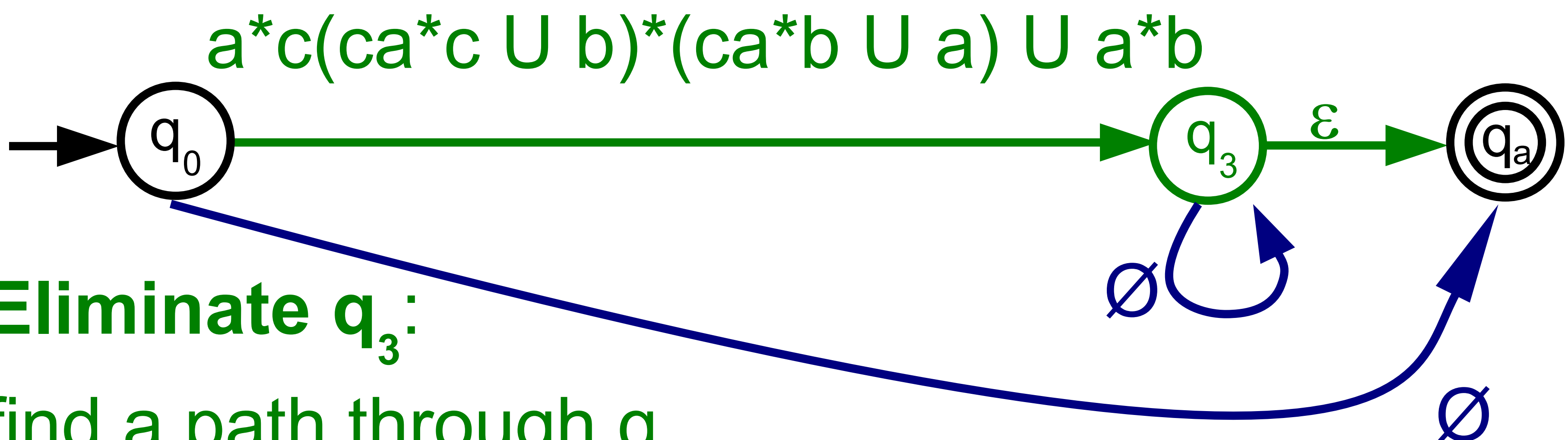


Eliminate q_3:

re-draw GNFA with

all other states

ANOTHER Example: DFA → GNFA → RE

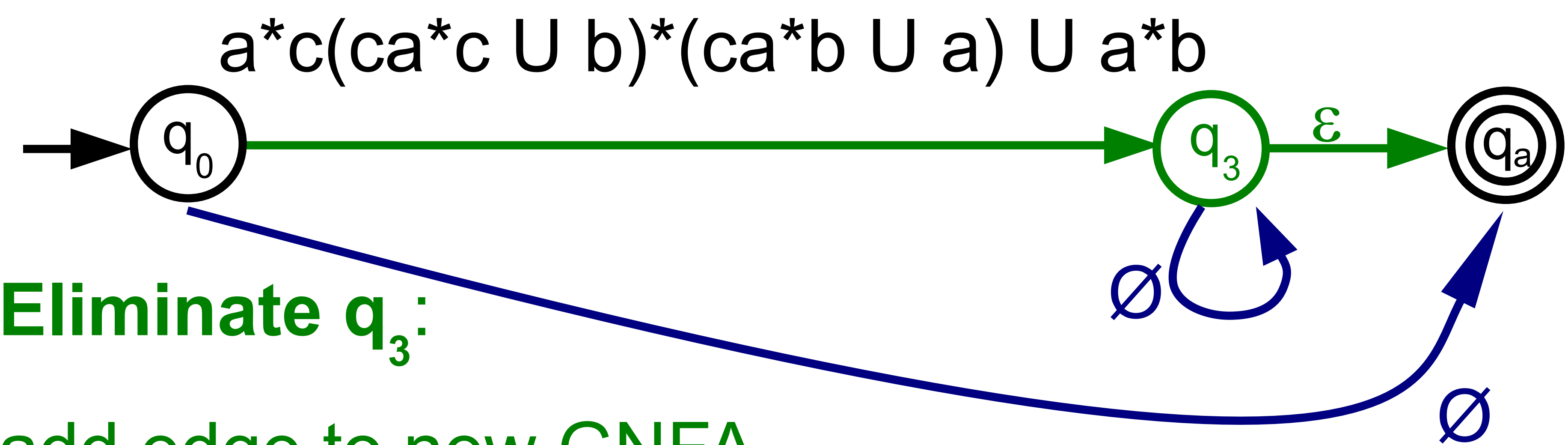


Eliminate q_3:

find a path through q_3

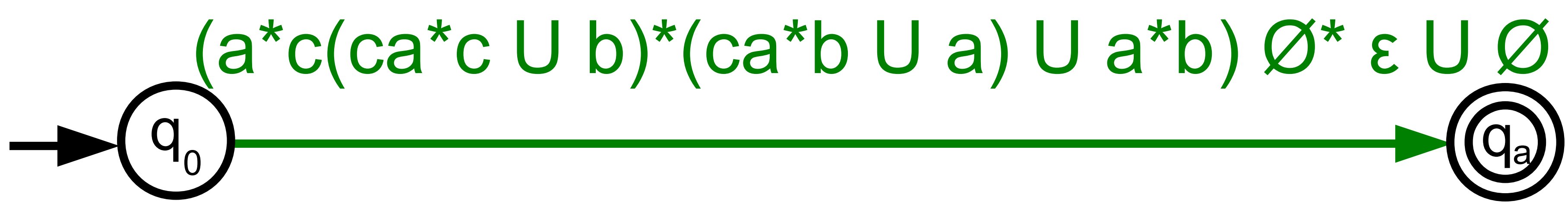
don't forget: no arrow means ∅

ANOTHER Example: DFA → GNFA → RE

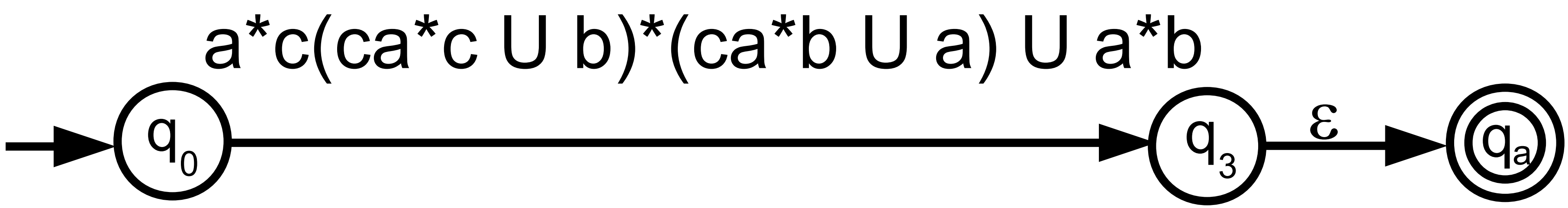


Eliminate q_3:

add edge to new GNFA



ANOTHER Example: DFA → GNFA → RE

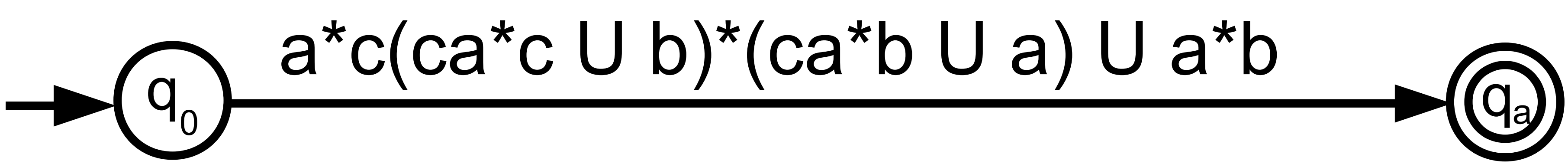


Eliminate q_3:

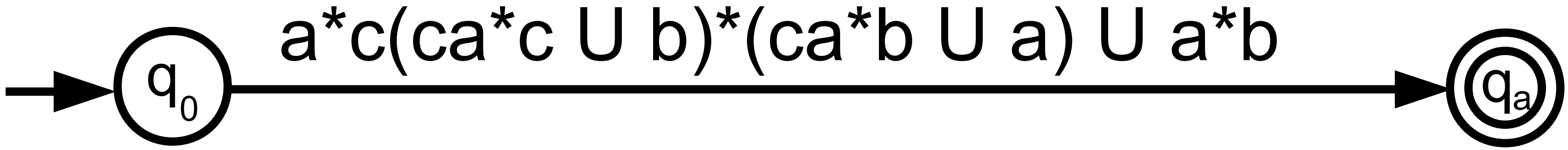
when no more paths through q_3, start over

(and simplify REs)

don't forget: Ø*= ε



ANOTHER Example: DFA → GNFA → RE



Only two states remain:

RE = a*c(ca*c U b)*(ca*b U a) U a*b

Recap:

Here "$\Rightarrow$" means "can be converted to"

$$\text{RE} \Leftrightarrow \text{DFA} \Leftrightarrow \text{NFA}$$

Any of the three recognize exactly

the regular languages (initially defined using DFA)

These conversions are used every time you enter an RE, for example for pattern matching using *grep*

- The RE is converted to an NFA
- Then the NFA is converted to a DFA
- The DFA representation is used to pattern-match

Optimizations have been devised,
but this is still the general approach.

What language is NOT regular?

Is $\{ 0^n 1^n : n \geq 0 \} = \{\varepsilon, 01, 0011, 000111, \ldots \}$ regular?

Pumping lemma:

L regular language $\Rightarrow$

$$\boxed{\begin{array}{l} \exists\ p \geq 0 \\ \forall\ w \in L,\ |w| \geq p \\ \exists\ x,y,z : w = xyz,\ |y| > 0,\ |xy| \leq p \\ \forall\ i \geq 0 : xy^i z \in L \end{array}}$$

Recall $y^0 = \varepsilon$, $y^1 = y$, $y^2 = yy$, $y^3 = yyy$, ...

Pumping lemma:

L regular language $\Rightarrow$ $\boxed{\begin{array}{l} \exists\ p \geq 0 \\ \forall\ w \in L,\ |w| \geq p \\ \exists\ x,y,z : w = xyz,\ |y| > 0,\ |xy| \leq p \\ \forall\ i \geq 0 : xy^iz \in L \end{array}}$

We will not see the proof. But here's the idea:

p := |Q| for DFA recognizing L

If $w \in L$, $|w| \geq p$, then during computation

2 states must be the same $q \in Q$

y = portion of w that brings back to q

can repeat y and still accept string

Pumping lemma:

L regular language $\Rightarrow$ $\boxed{\begin{array}{l} \exists\, p \geq 0 \qquad\qquad\qquad\qquad\qquad\qquad A \\ \forall\, w \in L,\ |w| \geq p \\ \exists\, x,y,z : w = xyz,\ |y| > 0,\ |xy| \leq p \\ \forall\, i \geq 0 : xy^{i}z \in L \end{array}}$

Useful to prove L NOT regular. Use contrapositive:

L regular language $\Rightarrow$ A

same as

(not A) $\Rightarrow$ L not regular

Pumping lemma (contrapositive)

$\forall\ p \geq 0$ not A

$\exists\ w \in L,\ |w| \geq p$

$\forall\ x,y,z : w = xyz,\ |y| > 0,\ |xy| \leq p$

$\exists\ i \geq 0 : xy^iz \notin L$

$\Rightarrow$ L not regular

To prove L not regular it is enough to prove not A

Not A is the stuff in the box.

Proving something like

$\forall$ bla $\exists$ bla $\forall$ bla $\exists$ bla bla

means winning a game

Theory is all about winning games!

Example NAME THE BIGGEST NUMBER GAME

- Two players:

  You, Adversary.

- Rules:

  First Adversary says a number.

  Then You say a number.

  You win if your number is bigger.

Can you win this game?

Example NAME THE BIGGEST NUMBER GAME

- Two players:

  You, Adversary.

- Rules:

  First Adversary says a number.

  Then You say a number.

  You win if your number is bigger.

You have winning strategy:

if adversary says x, you say x+1

Example NAME THE BIGGEST NUMBER GAME

- Two players:

You, Adversary. $\exists$, $\forall$

- Rules:

First Adversary says a number. $\forall x \exists y : y > x$

Then You say a number.

You win if your number is bigger.

You have winning strategy: Claim is true

if adversary says x, you say x+1

Another example:

Theorem: $\forall$ NFA N $\exists$ DFA M : L(M) = L(N)

We already saw a winning strategy for this game

What is it?

Another example:

Theorem: $\forall$ NFA N $\exists$ DFA M : L(M) = L(N)

We already saw a winning strategy for this game

The power set construction.

Games with more moves:

Chess, Checkers, Tic-Tac-Toe

You can win if

$\forall$ move of the Adversary

$\exists$ move You can make

$\forall$ move of the Adversary

$\exists$ move You can make

…

: You checkmate

Pumping lemma (contrapositive)

$\forall\ p \geq 0$

$\exists\ w \in L,\ |w| \geq p$

$\forall\ x,y,z : w = xyz,\ |y| > 0,\ |xy| \leq p$

$\exists\ i \geq 0 : xy^iz \notin L$

$\Rightarrow$ L not regular

Rules of the game:

Adversary picks p,

You pick $w \in L$ of length $\geq p$,

Adversary decomposes w in xyz, where $|y| > 0$, $|xy| \leq p$

You pick $i \geq 0$

Finally, you win if $xy^iz \notin L$

Theorem: $L := \{0^n 1^n : n \geq 0\}$ is not regular

$\forall\ p \geq 0$

$\exists\ w \in L, |w| \geq p$

$\forall\ x,y,z : w = xyz, |y| > 0, |xy| \leq p$

$\exists\ i \geq 0 : xy^iz \notin L$

Proof:

Use pumping lemma

Adversary moves p

You move $w := 0^p 1^p$

Adversary moves x,y,z

You move $i := 2$

You must show $xyyz \notin L$:

Since $|xy| \leq p$ and $w = xyz = 0^p 1^p$, y only has 0

So $xyyz = 0^{p + |y|} 1^p$

Since $|y| > 0$, this is not of the form $0^n 1^n$ DONE

Theorem: L := {w : w has as many 0 as 1} not regular

Same Proof:

Use pumping lemma

Adversary moves p

You move w := ?

$\forall$ p $\geq$0

$\exists$ w $\in$ L, |w| $\geq$ p

$\forall$ x,y,z : w = xyz, |y| > 0, |xy| $\leq$ p

$\exists$ i $\geq$ 0 : $xy^iz \notin$ L

Theorem: L := {w : w has as many 0 as 1} not regular

Same Proof:

Use pumping lemma

Adversary moves p

You move w := $0^p\ 1^p$

Adversary moves x,y,z

You move i := ?

$\forall$ p ≥ 0

$\exists$ w $\in$ L, $|w| \geq p$

$\forall$ x,y,z : w = xyz, $|y| > 0$, $|xy| \leq p$

$\exists$ i ≥ 0 : $xy^iz \notin L$

Theorem: L := {w : w has as many 0 as 1} not regular

Same Proof:

Use pumping lemma

Adversary moves p

You move w := $0^p 1^p$

Adversary moves x,y,z

You move i := 2

You must show xyyz $\notin$ L:

Since $|xy| \leq p$ and $w = xyz = 0^p 1^p$, y only has 0

So xyyz = ?

$\forall\ p \geq 0$

$\exists\ w \in L,\ |w| \geq p$

$\forall\ x,y,z : w = xyz,\ |y| > 0,\ |xy| \leq p$

$\exists\ i \geq 0 : xy^iz \notin L$

Theorem: L := {w : w has as many 0 as 1} not regular

Same Proof:

$\forall\ p \geq 0$

$\exists\ w \in L, |w| \geq p$

$\forall\ x,y,z : w = xyz, |y| > 0, |xy| \leq p$

$\exists\ i \geq 0 : xy^i z \notin L$

Use pumping lemma

Adversary moves p

You move $w := 0^p\ 1^p$

Adversary moves x,y,z

You move i := 2

You must show $xyyz \notin L$:

Since $|xy| \leq p$ and $w = xyz = 0^p\ 1^p$, y only has 0

So $xyyz = 0^{p + |y|}\ 1^p$

Since $|y| > 0$, not as many 0 as 1 DONE

Theorem: $L := \{0^j 1^k : j > k\}$ is not regular

Proof:

Use pumping lemma

Adversary moves p

You move w := ?

$\forall$ $p \geq 0$

$\exists$ $w \in L$, $|w| \geq p$

$\forall$ $x,y,z : w = xyz$, $|y| > 0$, $|xy| \leq p$

$\exists$ $i \geq 0 : xy^iz \notin L$

Theorem: $L := \{0^j 1^k : j > k\}$ is not regular

Proof:

Use pumping lemma

Adversary moves p

You move $w := 0^{p+1} 1^p$

Adversary moves x,y,z

You move $i :=$?

$\forall\ p \geq 0$

$\exists\ w \in L, |w| \geq p$

$\forall\ x,y,z : w = xyz, |y| > 0, |xy| \leq p$

$\exists\ i \geq 0 : xy^iz \notin L$

Theorem: $L := \{0^j 1^k : j > k\}$ is not regular

Proof:

Use pumping lemma

Adversary moves p

You move $w := 0^{p+1} 1^p$

Adversary moves x,y,z

You move $i := 0$

You must show $xz \notin L$:

Since $|xy| \leq p$ and $w = xyz = 0^{p+1} 1^p$, y only has 0

So $xz = 0^{p+1-|y|} 1^p$

Since $|y| > 0$, this is not of the form $0^j 1^k$ with $j > k$

$$\boxed{\begin{array}{l} \forall\ p \geq 0 \\ \exists\ w \in L, |w| \geq p \\ \forall\ x,y,z : w = xyz, |y| > 0, |xy| \leq p \\ \exists\ i \geq 0 : xy^iz \notin L \end{array}}$$

Theorem: $L := \{uu : u \in \{0,1\}^*\}$ is not regular

Proof:

Use pumping lemma

Adversary moves p

You move w := ?

$\forall\ p \geq 0$

$\exists\ w \in L, |w| \geq p$

$\forall\ x,y,z : w = xyz, |y| > 0, |xy| \leq p$

$\exists\ i \geq 0 : xy^iz \notin L$

Theorem: $L := \{uu : u \in \{0,1\}^*\}$ is not regular

Proof:

Use pumping lemma

Adversary moves p

You move $w := 0^p 1\ 0^p\ 1$

Adversary moves x,y,z

You move $i :=$?

$\forall\ p \geq 0$

$\exists\ w \in L, |w| \geq p$

$\forall\ x,y,z : w = xyz, |y| > 0, |xy| \leq p$

$\exists\ i \geq 0 : xy^iz \notin L$

Theorem: $L := \{uu : u \in \{0,1\}^* \}$ is not regular

Proof:

$\forall\ p \geq 0$

$\exists\ w \in L,\ |w| \geq p$

$\forall\ x,y,z : w = xyz,\ |y| > 0,\ |xy| \leq p$

$\exists\ i \geq 0 : xy^iz \notin L$

Use pumping lemma

Adversary moves p

You move $w := 0^p\, 1\, 0^p\, 1$

Adversary moves x,y,z

You move $i := 2$

You must show $xyyz \notin L$:

Since $|xy| \leq p$ and $w = xyz = 0^p\, 1\, 0^p\, 1$, y only has 0

So $xyyz = 0^{p + |y|}\, 1\, 0^p\, 1$

Since $|y| > 0$, first half of xyyz only 0, so $xyyz \notin L$

Theorem: L := { 1^{n^2} : n ≥ 0 } is not regular

Proof:

Use pumping lemma

Adversary moves p

You move w := ?

$\forall$ p ≥0

$\exists$ w ∈ L, |w| ≥ p

$\forall$ x,y,z : w = xyz, |y| > 0, |xy| ≤ p

$\exists$ i ≥ 0 : $xy^iz \notin$ L

Theorem: $L := \{ 1^{n^2} : n \geq 0 \}$ is not regular

Proof:

Use pumping lemma

Adversary moves p

You move $w := 1^{p^2}$

Adversary moves x,y,z

You move $i :=$?

$\forall\ p \geq 0$

$\exists\ w \in L, |w| \geq p$

$\forall\ x,y,z : w = xyz, |y| > 0, |xy| \leq p$

$\exists\ i \geq 0 : xy^iz \notin L$

Theorem: $L := \{ 1^{n^2} : n \geq 0 \}$ is not regular

Proof:

Use pumping lemma

Adversary moves p

You move $w := 1^{p^2}$

Adversary moves x,y,z

You move $i := 2$

You must show $xyyz \notin L$:

Since $|xy| \leq p$, $|xyyz| \leq$?

$\forall\ p \geq 0$

$\exists\ w \in L, |w| \geq p$

$\forall\ x,y,z : w = xyz, |y| > 0, |xy| \leq p$

$\exists\ i \geq 0 : xy^iz \notin L$

Theorem: $L := \{ 1^{n^2} : n \geq 0 \}$ is not regular

Proof:

$\forall\ p \geq 0$

$\exists\ w \in L, |w| \geq p$

$\forall\ x,y,z : w = xyz, |y| > 0, |xy| \leq p$

$\exists\ i \geq 0 : xy^i z \notin L$

Use pumping lemma

Adversary moves p

You move $w := 1^{p^2}$

Adversary moves x,y,z

You move $i := 2$

You must show $xyyz \notin L$:

Since $|xy| \leq p$, $|xyyz| \leq p^2 + p < (p+1)^2$

Since $|y| > 0$, $|xyyz| >$?

Theorem: $L := \{ 1^{n^2} : n \geq 0 \}$ is not regular

Proof:

Use pumping lemma

Adversary moves p

You move $w := 1^{p^2}$

Adversary moves x,y,z

You move $i := 2$

You must show $xyyz \notin L$:

Since $|xy| \leq p$, $|xyyz| \leq p^2 + p < (p+1)^2$

Since $|y| > 0$, $|xyyz| > p^2$

So $|xyyz|$ cannot be … what ?

$\forall\ p \geq 0$

$\exists\ w \in L, |w| \geq p$

$\forall\ x,y,z : w = xyz, |y| > 0, |xy| \leq p$

$\exists\ i \geq 0 : xy^iz \notin L$

Theorem: $L := \{ 1^{n^2} : n \geq 0 \}$ is not regular

$\forall\ p \geq 0$

$\exists\ w \in L, |w| \geq p$

$\forall\ x,y,z : w = xyz, |y| > 0, |xy| \leq p$

$\exists\ i \geq 0 : xy^iz \notin L$

Proof:

Use pumping lemma

Adversary moves p

You move $w := 1^{p^2}$

Adversary moves x,y,z

You move $i := 2$

You must show $xyyz \notin L$:

Since $|xy| \leq p$, $|xyyz| \leq p^2 + p < (p+1)^2$

Since $|y| > 0$, $|xyyz| > p^2$

So $|xyyz|$ cannot be a square. $xyyz \notin L$

Big picture

- All languages
- Decidable

  Turing machines
- NP
- P
- Context-free

  Context-free grammars, push-down automata
- Regular

  Automata, non-deterministic automata,

  regular expressions