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 }
- A ∩ B := { w : w in A and w in B }

are all regular

Big picture

- All languages
- Decidable

  Turing machines
- NP
- P
- Context-free

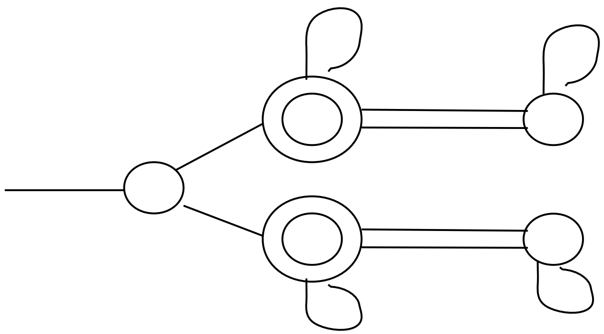
  Context-free grammars, push-down automata
- Regular

  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 ∑ for $U_{a \in \Sigma}$ a So if ∑ = {a,b} then ∑ = 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*, (ΣΣ)*, 01 U 10

Definition Regular expressions RE over Σ 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 $\Sigma = \{ a, b\}$

RE	Language
• ab U ba	?
• a*	
• (a U b)*	
• a*ba*	
• $\Sigma^*b\Sigma^*$	
• $\Sigma^*aab\Sigma^*$	
• $(\Sigma\Sigma)^*$	
• (a*ba*ba*)*	
• a*baba*a$\emptyset$	

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

RE	Language
• ab U ba	{ab, ba}
• a*	
• (a U b)*	
• a*ba*	
• $\Sigma^*b\Sigma^*$	
• $\Sigma^*aab\Sigma^*$	
• $(\Sigma\Sigma)^*$	
• (a*ba*ba*)*	
• a*baba*a$\emptyset$	

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

RE	Language
• ab U ba	{ab, ba}
• a*	$\{\varepsilon$, a, aa, … } = { w : w has only a}
• (a U b)*	
• a*ba*	
• $\Sigma^*b\Sigma^*$	
• $\Sigma^*aab\Sigma^*$	
• $(\Sigma\Sigma)^*$	
• (a*ba*ba*)*	
• a*baba*a$\emptyset$	

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

RE	Language
• ab U ba	{ab, ba}
• a*	$\{\varepsilon$, a, aa, … } = { w : w has only a}
• (a U b)*	all strings
• a*ba*	
• Σ*bΣ*	
• Σ*aabΣ*	
• ($\Sigma\Sigma$)*	
• (a*ba*ba*)*	
• a*baba*a$\varnothing$	

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

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

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*	
• ($\Sigma\Sigma$)*	
• (a*ba*ba*)*	
• a*baba*a∅	

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

RE	Language
• ab U ba	{ab, ba}
• a*	$\{\varepsilon$, 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}
• ($\Sigma\Sigma$)*	
• (a*ba*ba*)*	
• a*baba*a$\emptyset$	

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

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

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*	{w : w contains the string aab}
• (ΣΣ)*	{w : w has even length}
• (a*ba*ba*)*	{w : w contains even number of b}
• a*baba*a∅	

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

RE	Language
• ab U ba	{ab, ba}
• a*	$\{\varepsilon$, 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*)*	{w : w contains even number of b}
• a*baba*a$\emptyset$	$\emptyset$ (anything o $\emptyset$ = $\emptyset$)

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

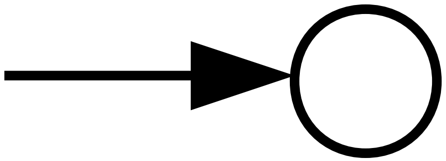
Theorem: For every RE R there is NFA M: L(M) = L(R)

Construction:

- R = ∅ M := ?

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

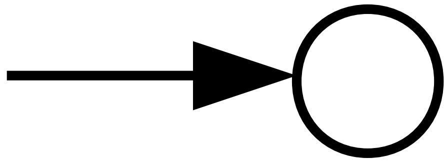
Construction:

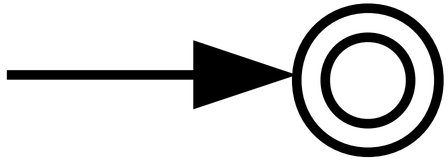
- R = ∅ M := 

- R = ε M := ?

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

Construction:

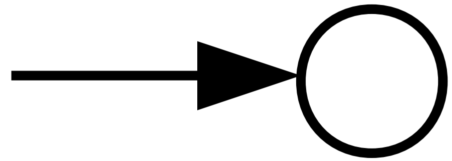
- R = ∅ M := 

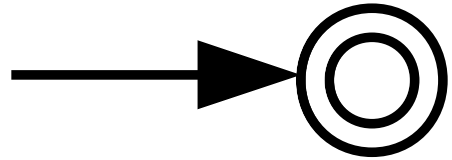
- R = ε M := 

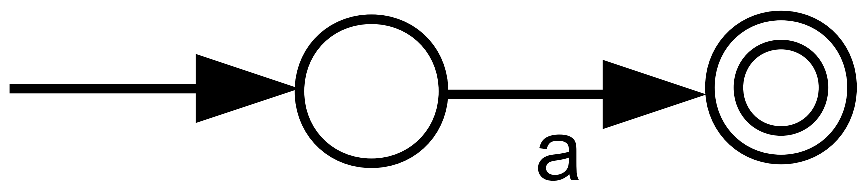
- R = a M := ?

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

Construction:

- R = ∅ M := 

- R = ε M := 

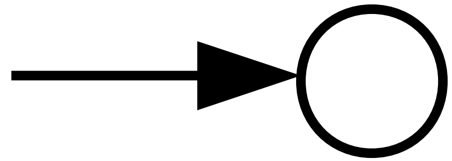
- R = a M := 

- R = R U R' ?

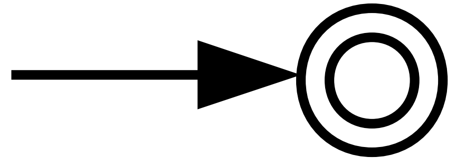
Theorem: For every RE R there is NFA M: L(M) = L(R)

Construction:

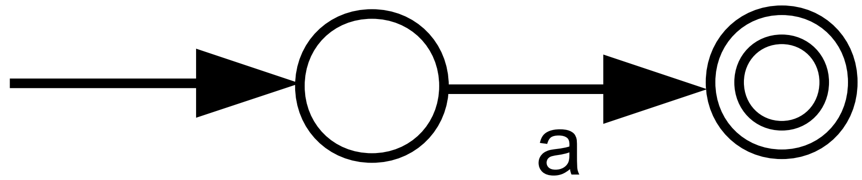
- R = ∅ M :=



- R = ε M :=



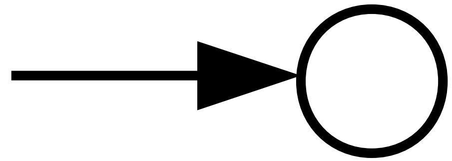
- R = a M :=

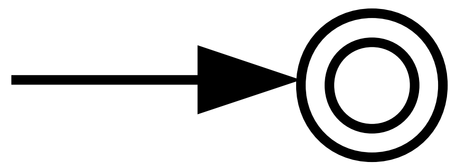


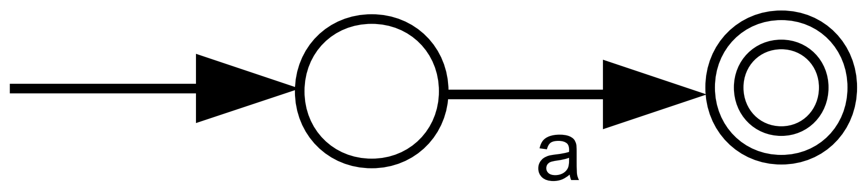
- R = R U R' use construction for A U B seen earlier
- R = R o R' ?

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

Construction:

- R = ∅ M := 

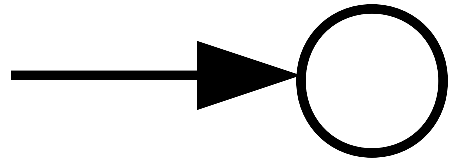
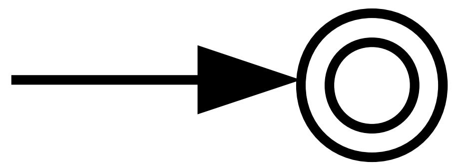
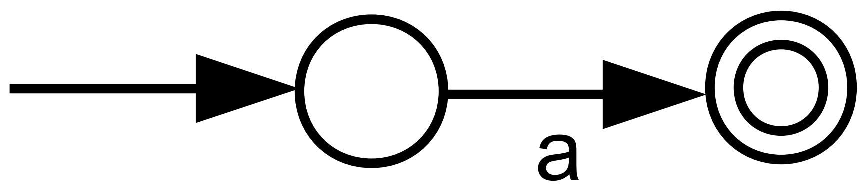
- R = ε M := 

- 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 := 

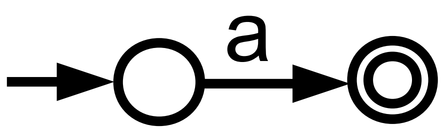
- R = R U R' use construction for A U B seen earlier
- R = R o R' use construction for A o B seen earlier
- R = R* use construction for A* seen earlier

Example: RE → NFA

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

Example: RE → NFA

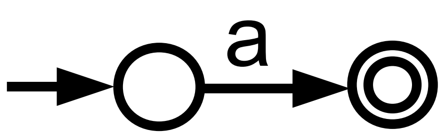
$$RE = (ab \cup a)^*$$

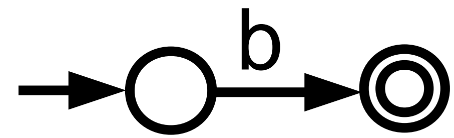
M_a = 

$L(M_a)=L(a)$

Example: RE → NFA

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

M_a = 

M_b = 

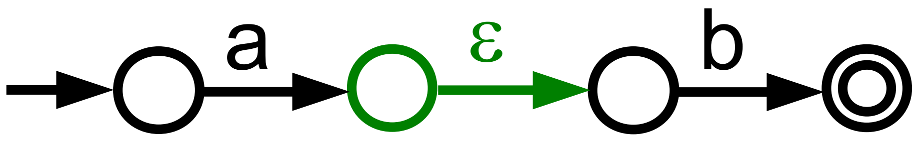
$L(M_a) = L(a)$

$L(M_b) = L(b)$

Example: RE → NFA

RE = (ab U a)*

M_{ab} =

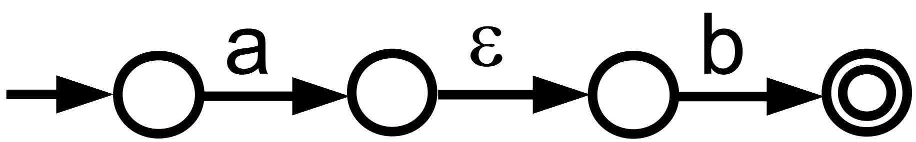


$L(M_{ab})=L(ab)$

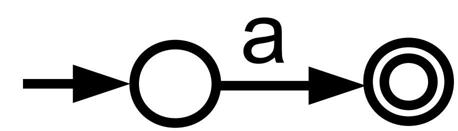
Example: RE → NFA

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

M_{ab} =



M_a =



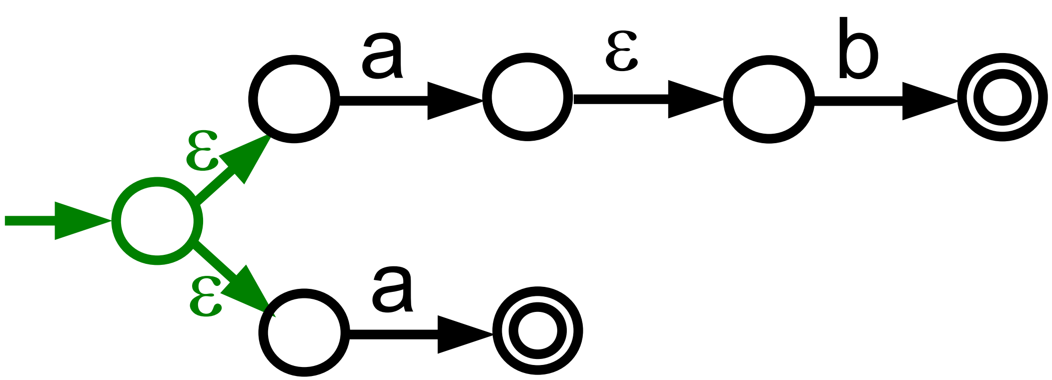
$L(M_{ab}) = L(ab)$

$L(M_a) = L(a)$

Example: RE → NFA

RE = (ab U a)*

$M_{ab\ U\ a}$ =

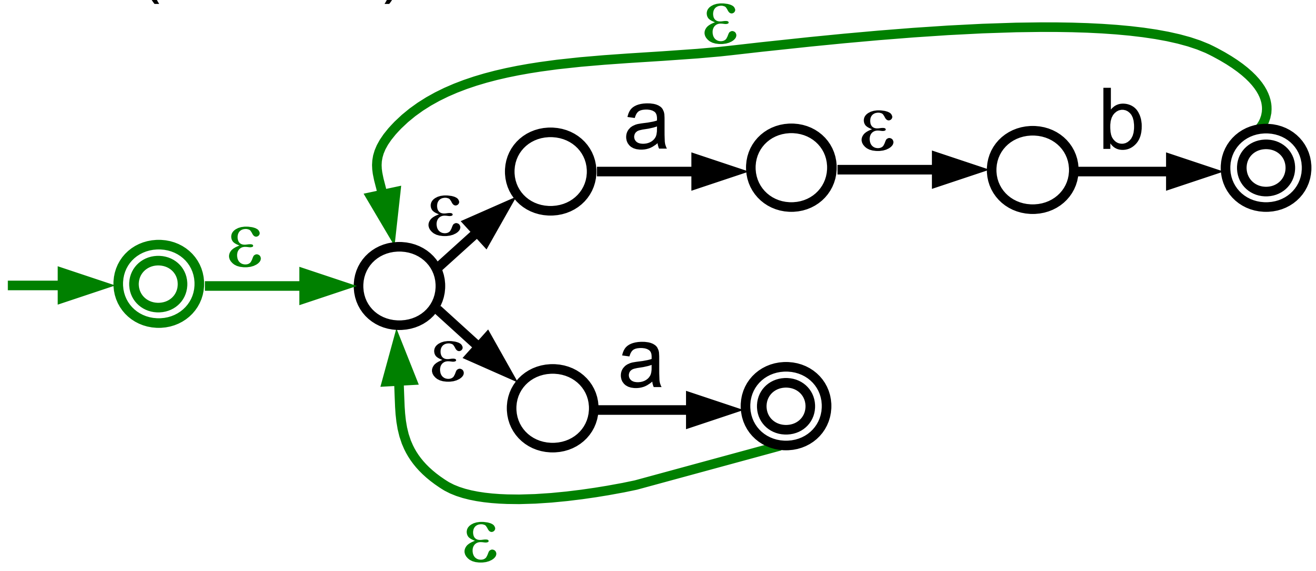


$L(M_{ab\ U\ a}) = L(ab\ U\ a)$

Example: RE → NFA

RE = (ab U a)*

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



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

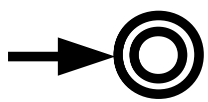
ANOTHER Example: RE → NFA

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

ANOTHER Example: RE → NFA

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

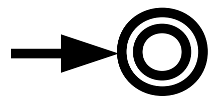
M_ε =

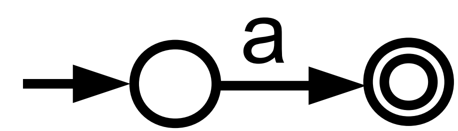


$$L(M_\varepsilon) = L(\varepsilon)$$

ANOTHER Example: RE → NFA

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

M_ε = 

M_a = 

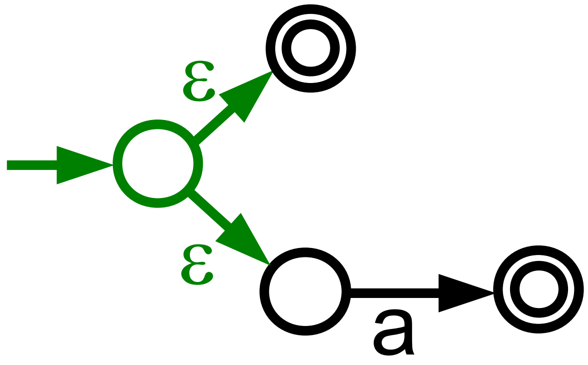
$L(M_\varepsilon)=L(\varepsilon)$

$L(M_a)=L(a)$

ANOTHER Example: RE → NFA

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

$$M_{\varepsilon \cup a} =$$

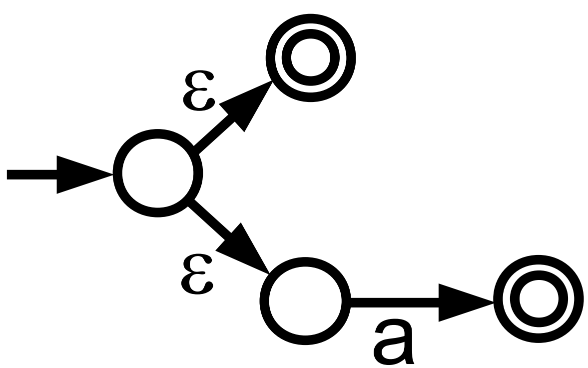


$$L(M_{\varepsilon \cup a}) = L(\varepsilon \cup a)$$

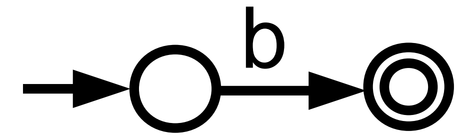
ANOTHER Example: RE → NFA

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

$M_{\varepsilon \cup a}$ =



M_b =



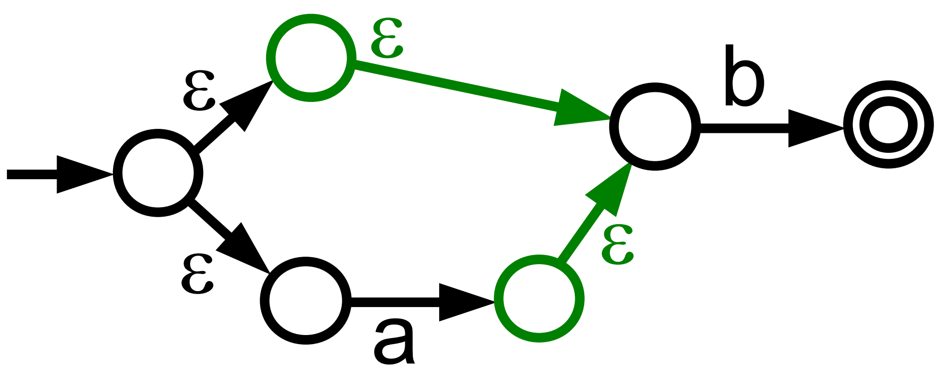
$L(M_b) = L(b)$

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

ANOTHER Example: RE → NFA

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

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

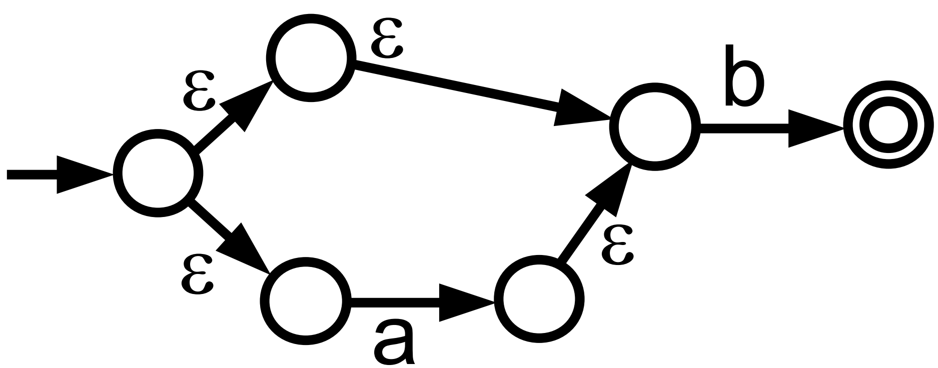


$$L(M_{(\varepsilon \cup a)b}) = L((\varepsilon \cup a)b)$$

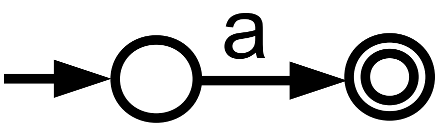
ANOTHER Example: RE → NFA

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

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



$M_a =$



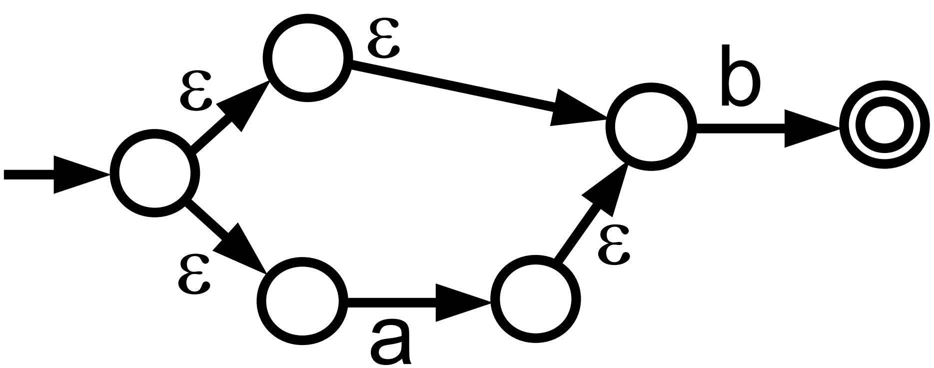
$L(M_a) = L(a)$

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

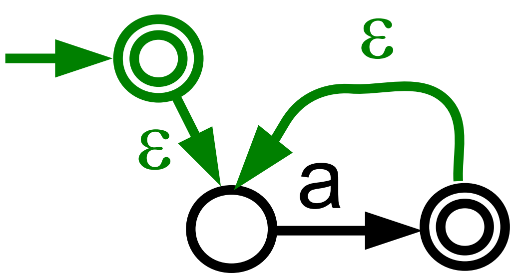
ANOTHER Example: RE → NFA

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

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



$M_{a^*} =$



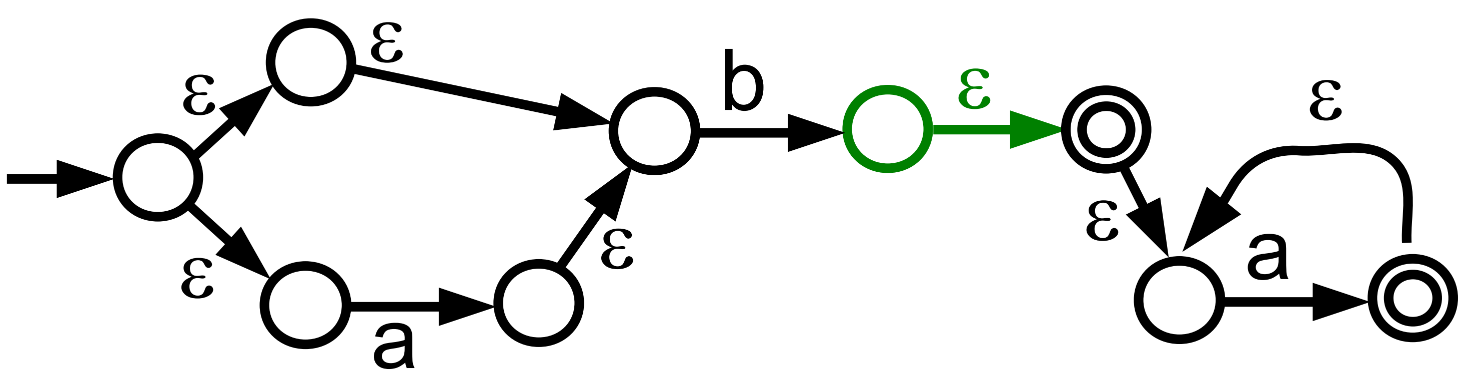
$L(M_{a^*}) = L(a^*)$

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

ANOTHER Example: RE → NFA

RE =(ε U a)ba*

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



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

Recap:

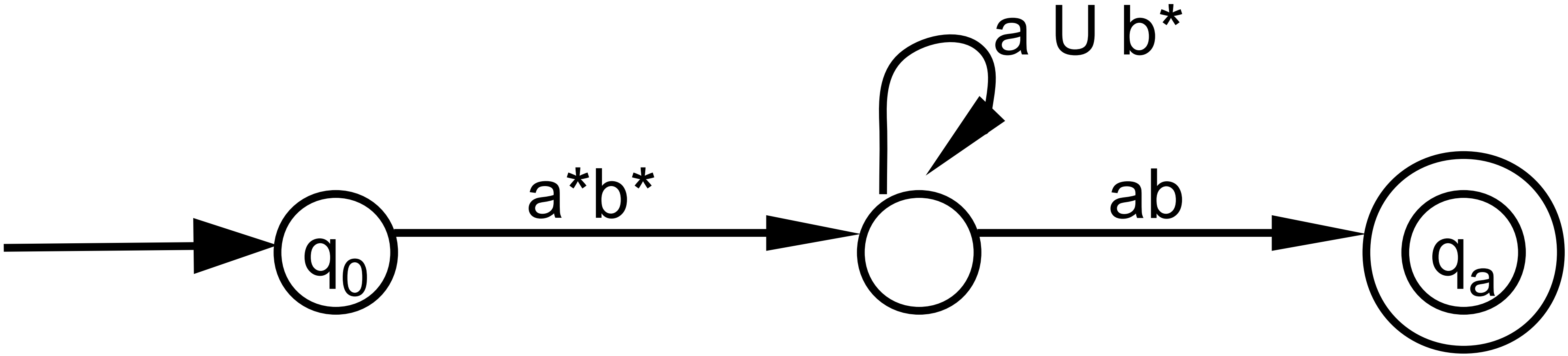
Here "$\Rightarrow$" means "can be converted to"

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)

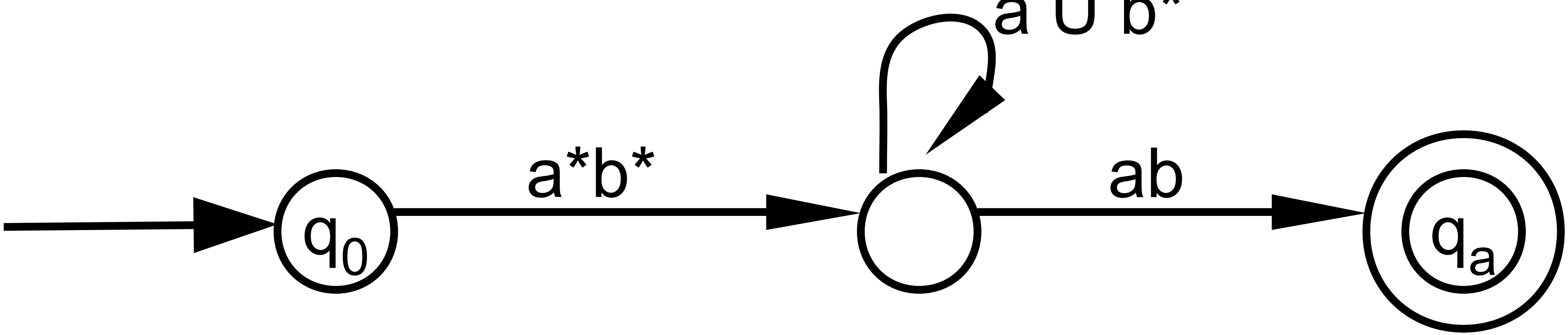


Nondeterministic

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 = ∅

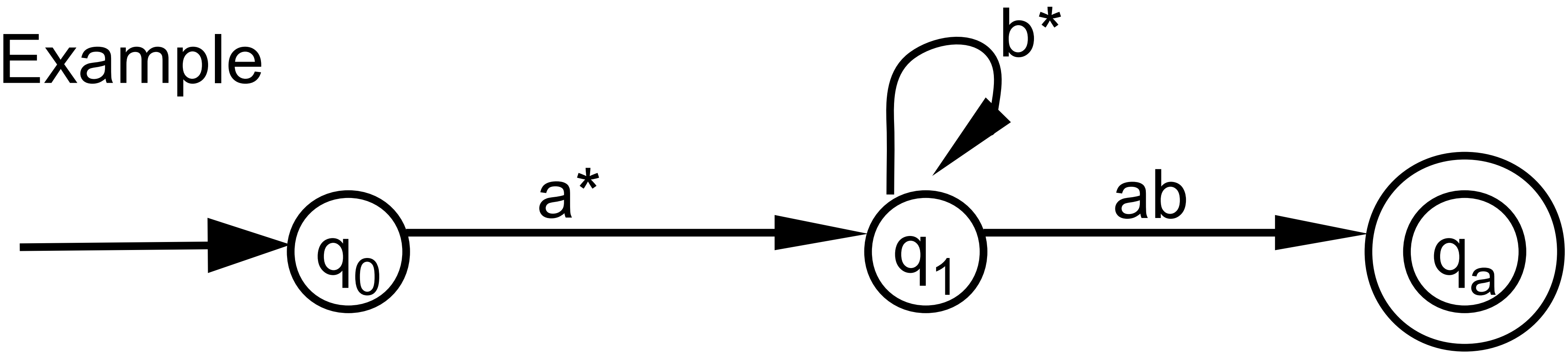
- Definition: A generalized finite automaton (GNFA)
- is a 5-tuple $(Q, \Sigma, \delta, q_0, q_a)$ where
- Q is a finite set of states
- Σ is the input alphabet
- $\delta : (Q - \{q_a\}) \times (Q - \{q_0\}) \rightarrow$ Regular Expressions
- q_0 in Q is the start state
- q_a in Q is the accept state

- **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

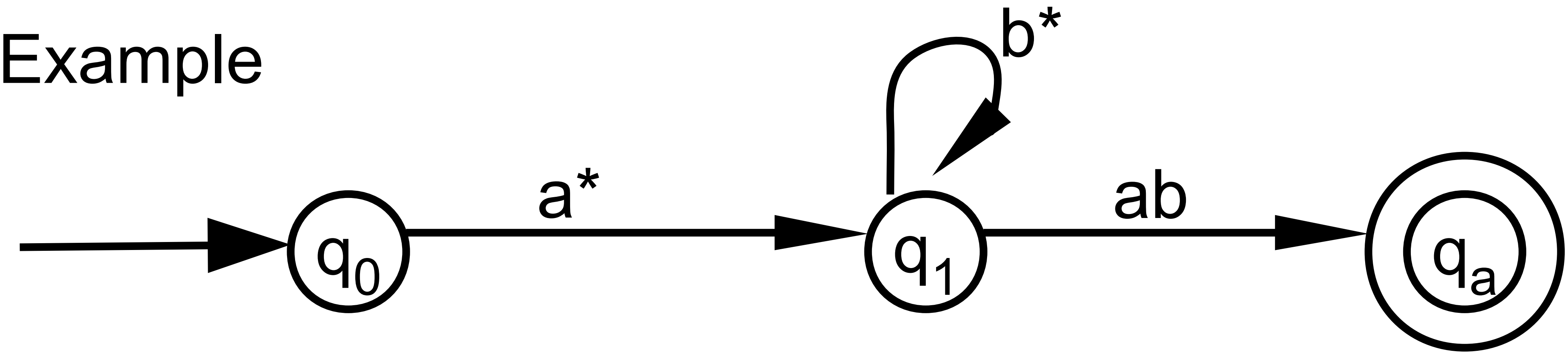
Example



Accepts w = aaabbab

w_1=?

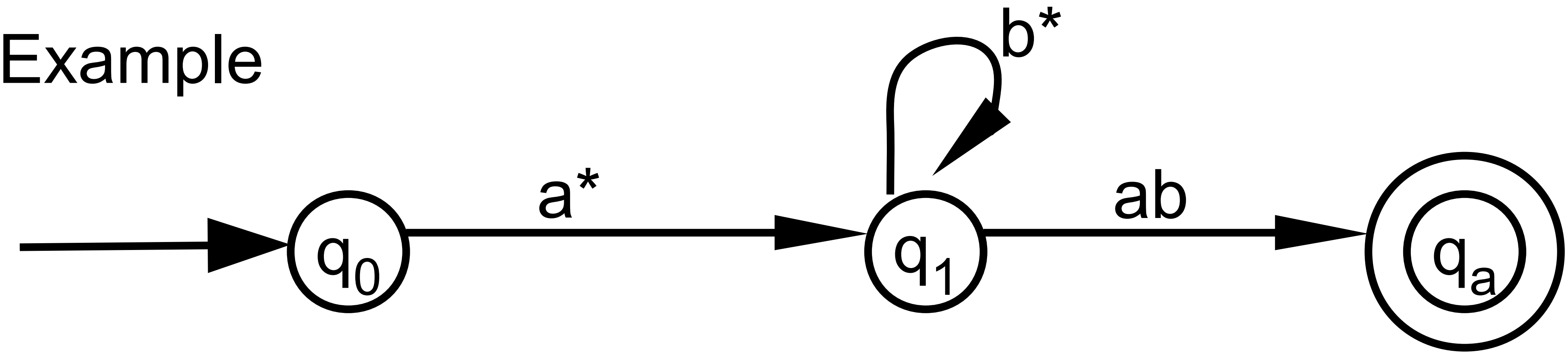
Example



Accepts w = aaabbab

w_1=aaa w_2=?

Example

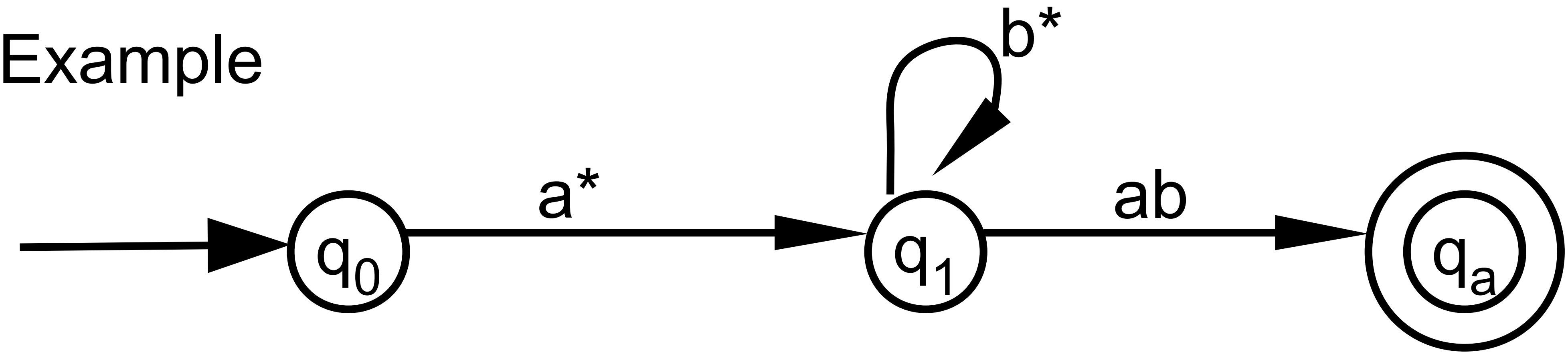


Accepts w = aaabbab

w_1=aaa w_2=bb w_3=ab

r_0=q_0 r_1=?

Example



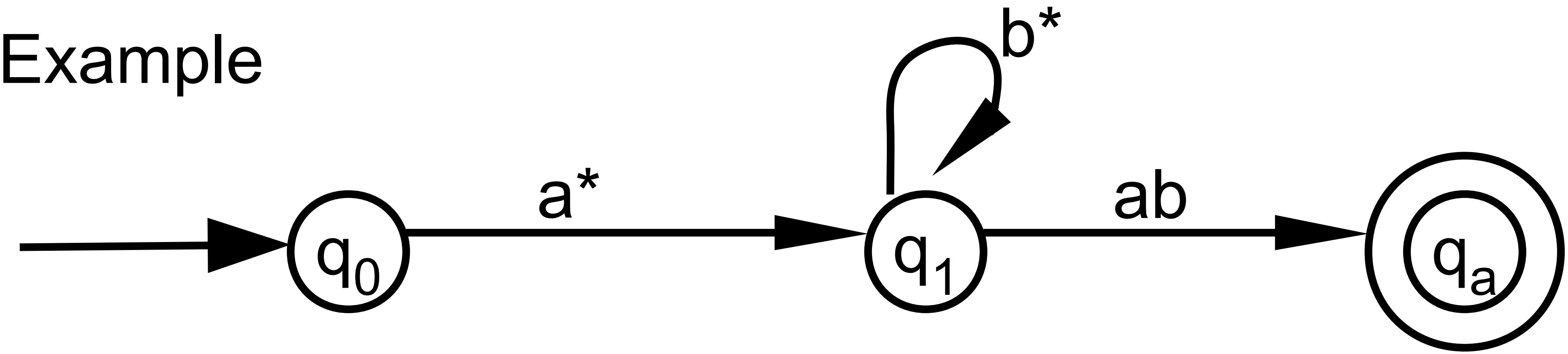
Accepts w = aaabbab

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

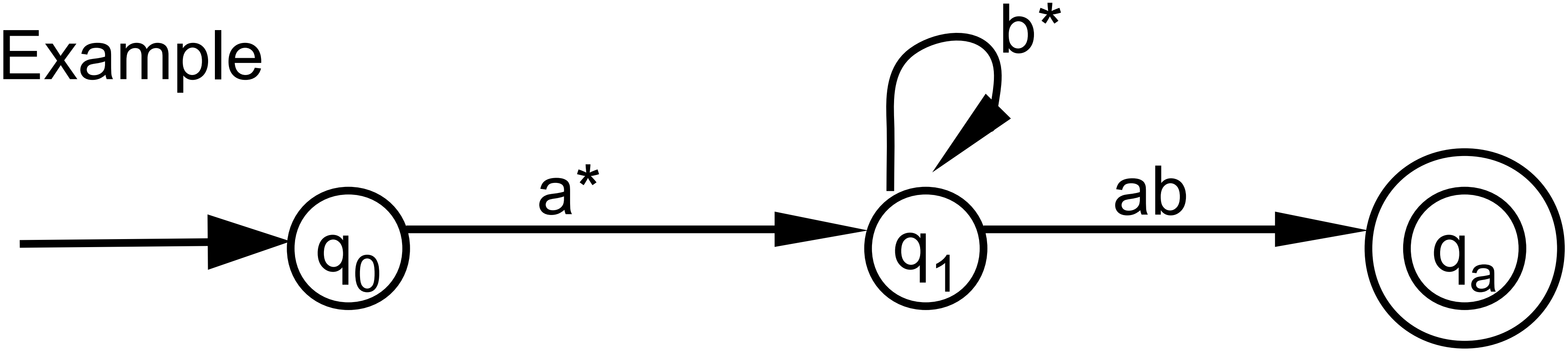
w_1=aaa w_2=bb w_3=ab

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

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

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

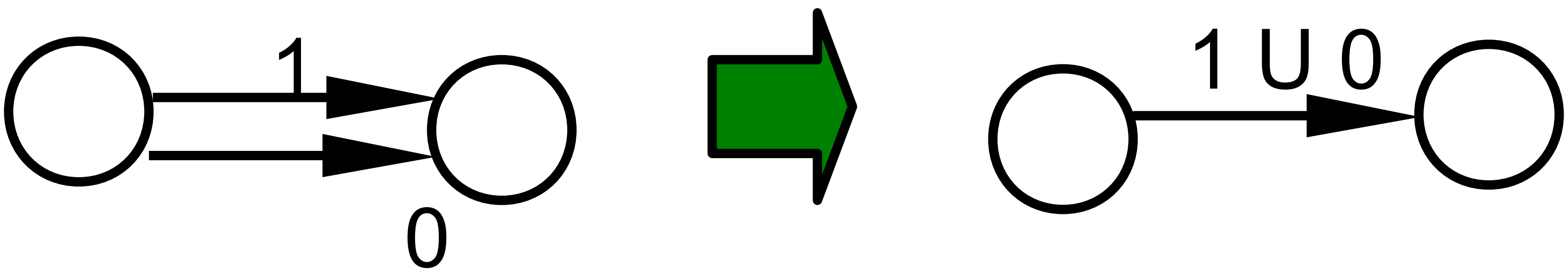
$w_2 = bb \in L(\delta(r_1,r_2)) = L(\delta(q_1,q_1)) = L(b^*)$

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

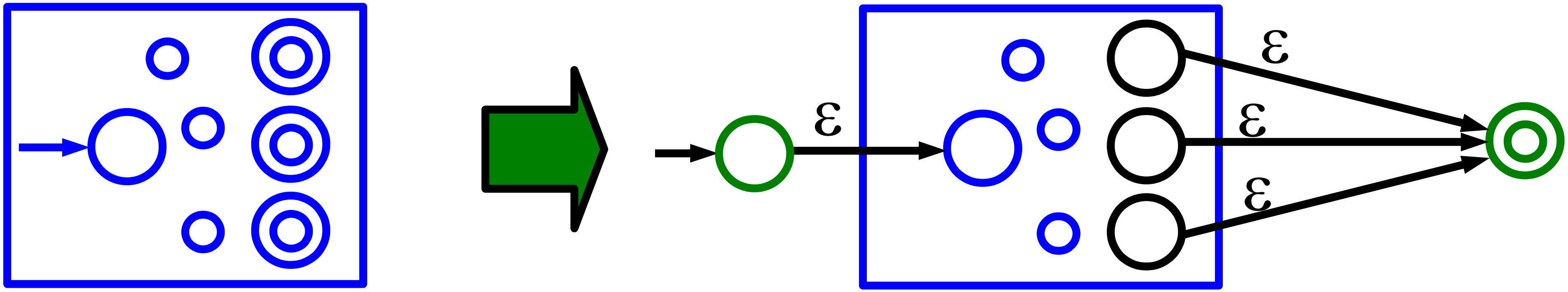
Theorem: ∀ DFA M ∃ GNFA N : L(N) = L(M)

Construction:

To ensure unique transition between each pair:



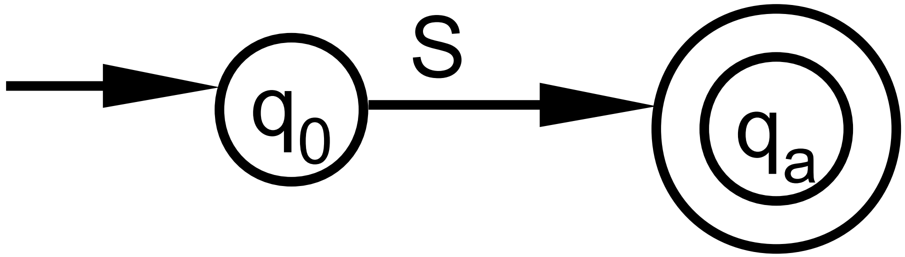
To ensure unique final state, no transitions ingoing start state, no transitions outgoing final state:



Theorem: $\forall$ GNFA N $\exists$ RE R : L(R) = L(N)

Construction:

If N has 2 states, then N =

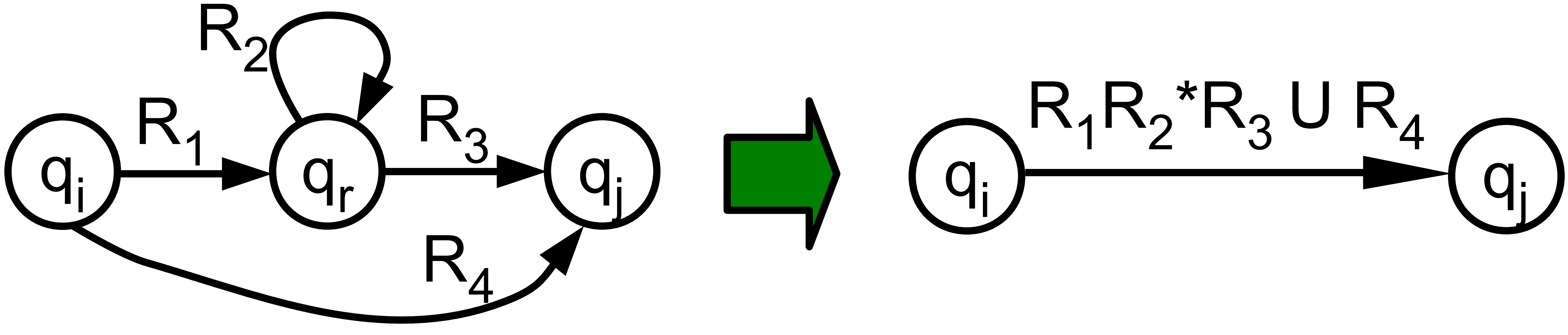


thus R := S

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

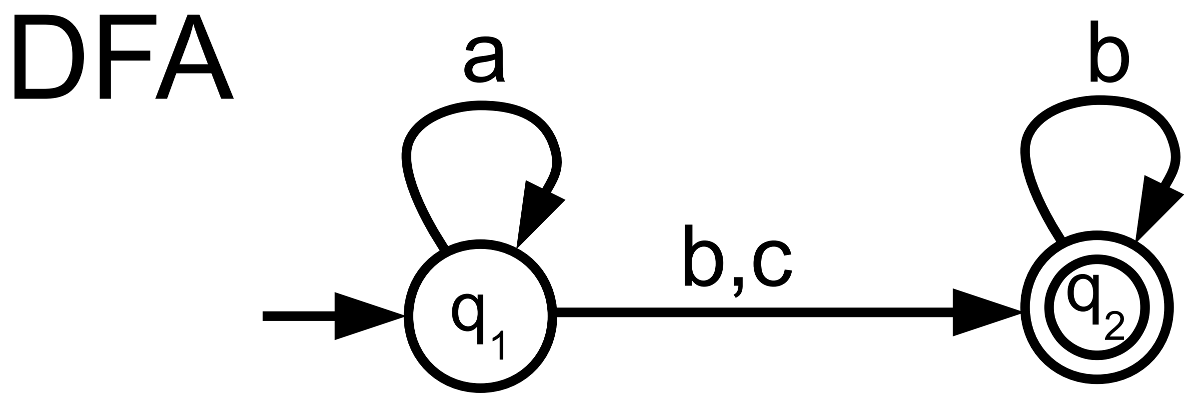
for every ordered pair q_i, q_j (possibly equal)

that are connected through q_r

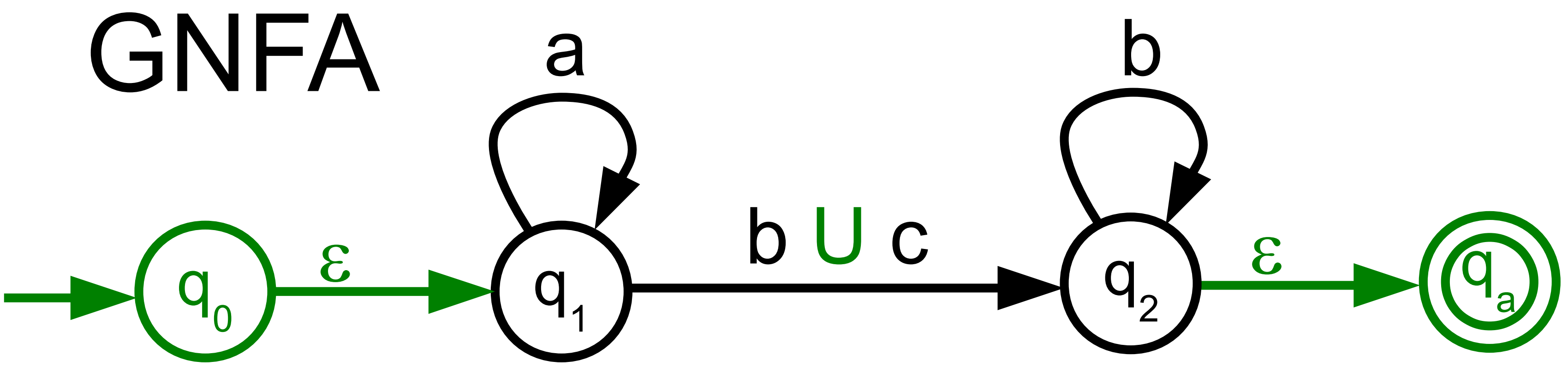


Repeat until 2 states remain

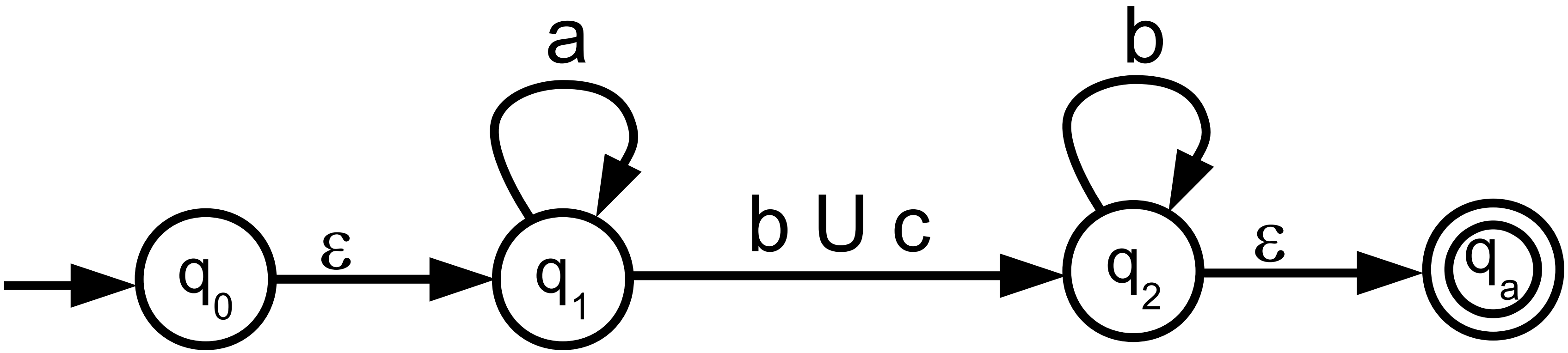
Example: DFA → GNFA → RE



Example: DFA → GNFA → RE

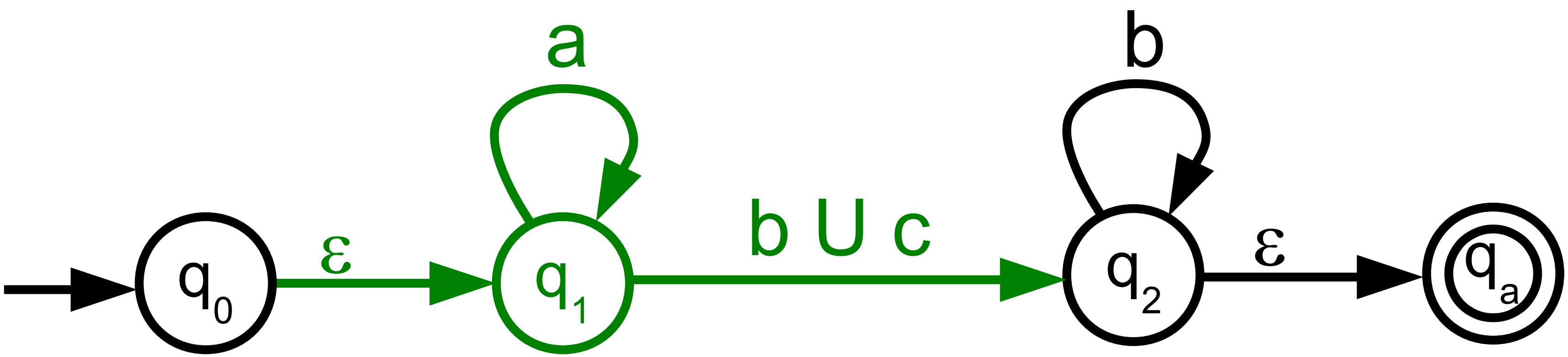


Example: DFA → GNFA → RE



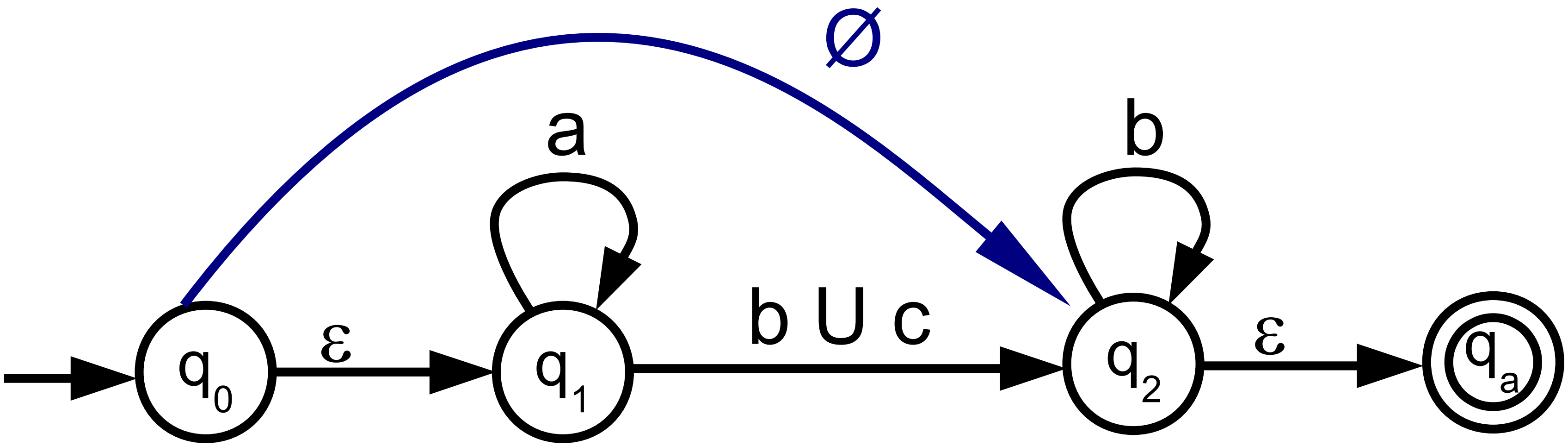
Eliminate q_1: re-draw GNFA with all other states

Example: DFA → GNFA → RE



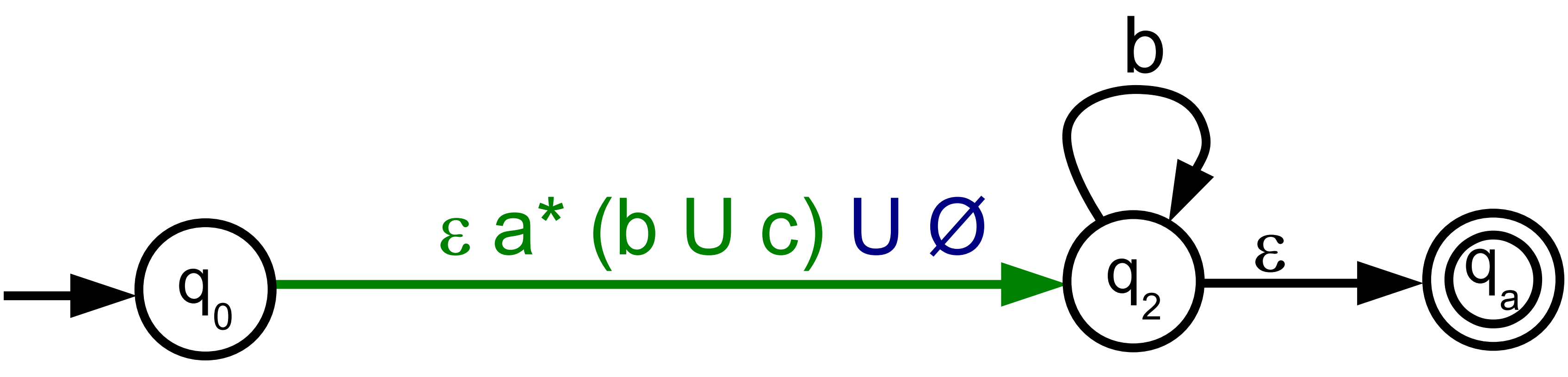
Eliminate q_1: find a path through q_1

Example: DFA → GNFA → RE

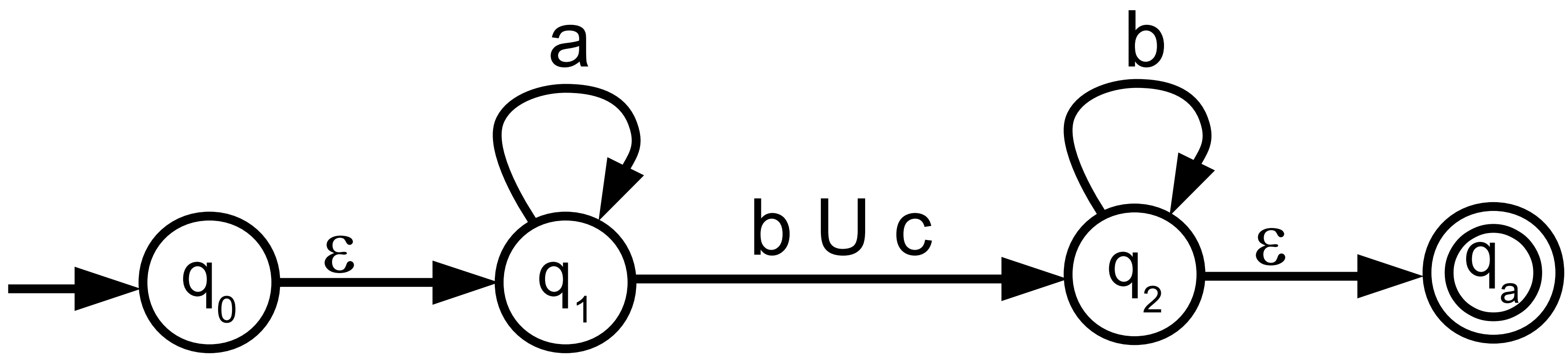


Eliminate q_1: add edge to new GNFA

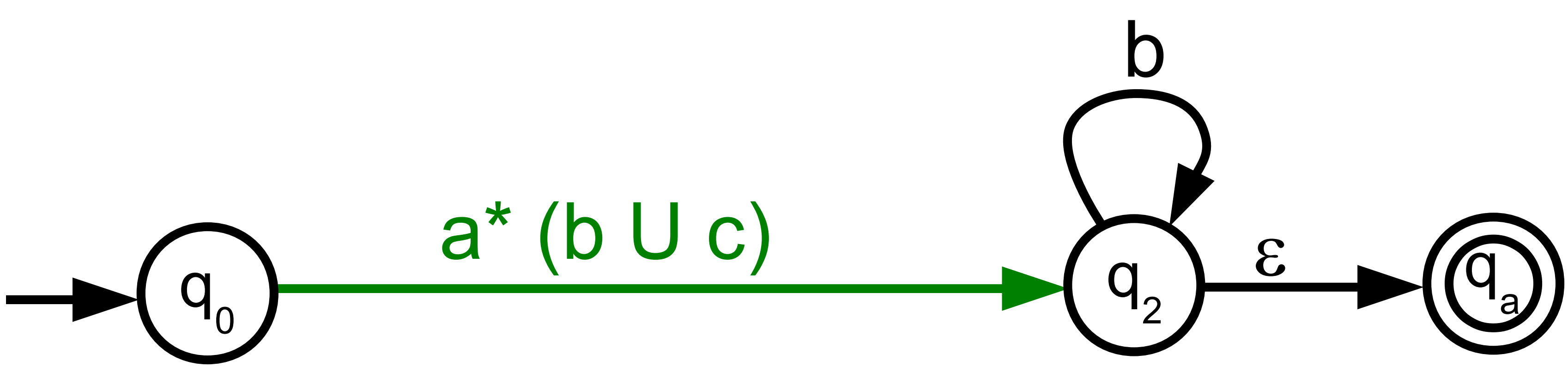
Don't forget: no arrow means label Ø



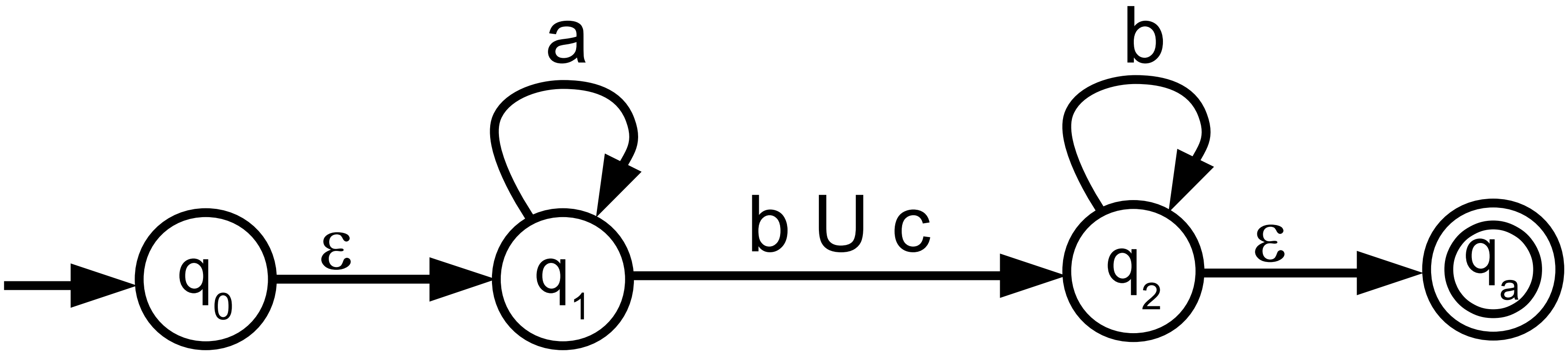
Example: DFA → GNFA → RE



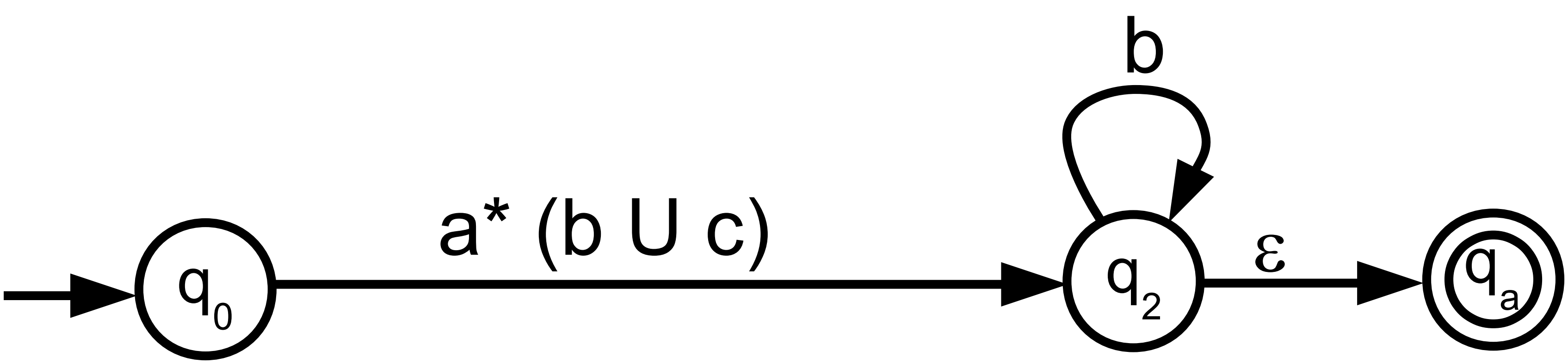
Eliminate q_1: simplify RE on new edge



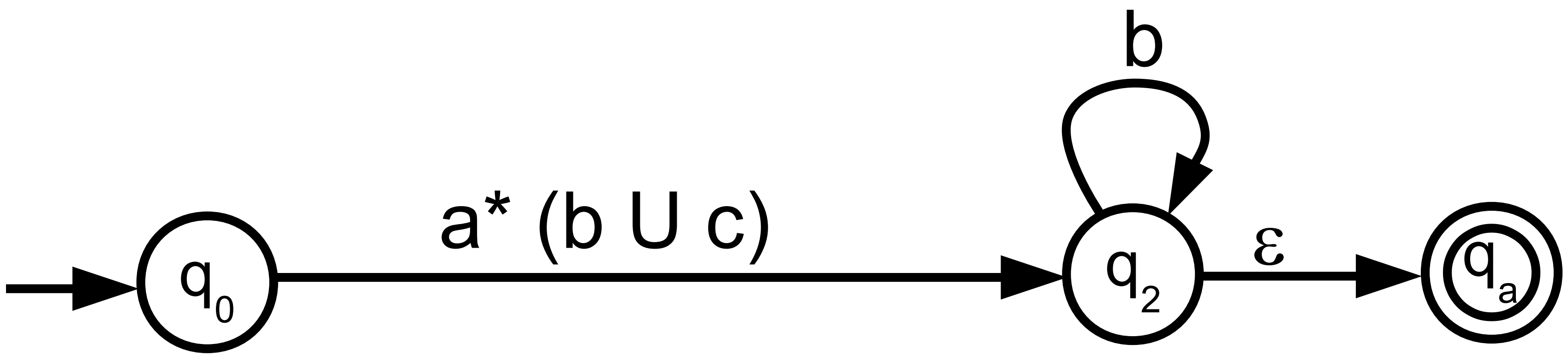
Example: DFA → GNFA → RE



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

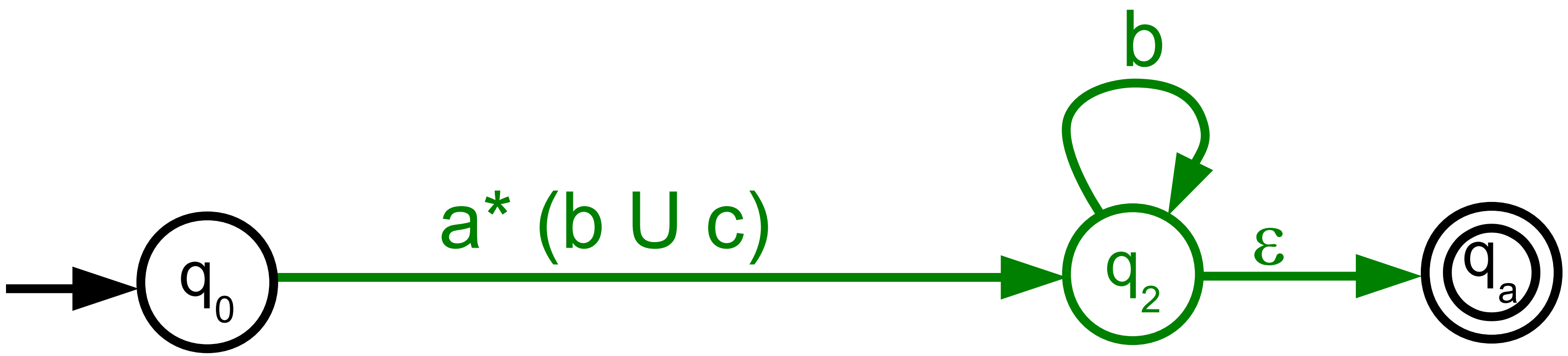


Example: DFA → GNFA → RE



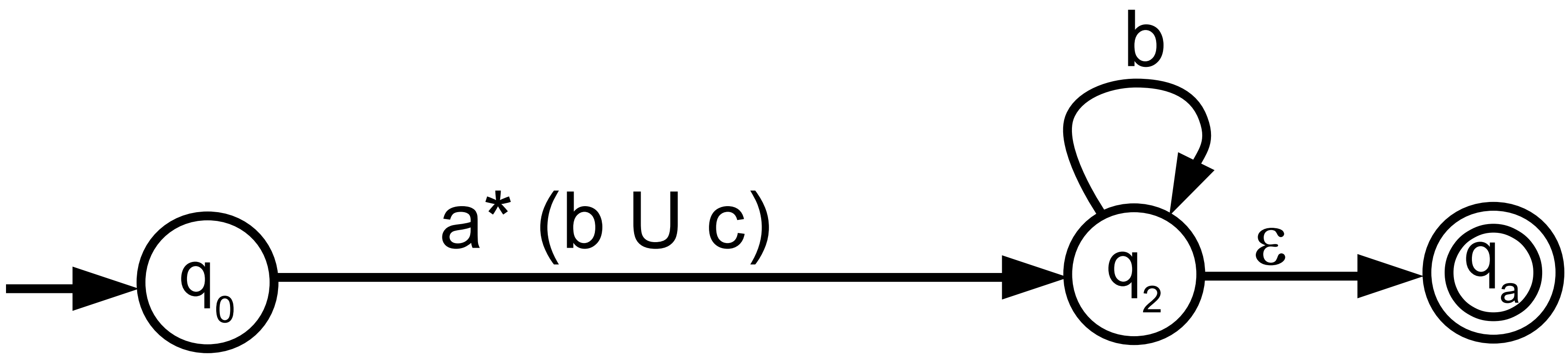
Eliminate q_2: re-draw GNFA with all other states

Example: DFA → GNFA → RE

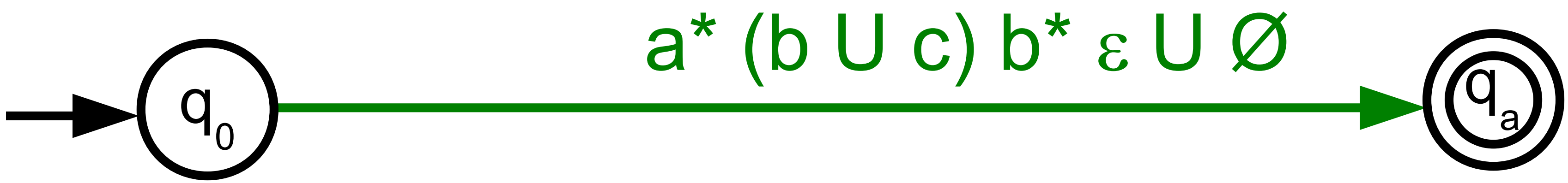


Eliminate q_2: find a path through q_2

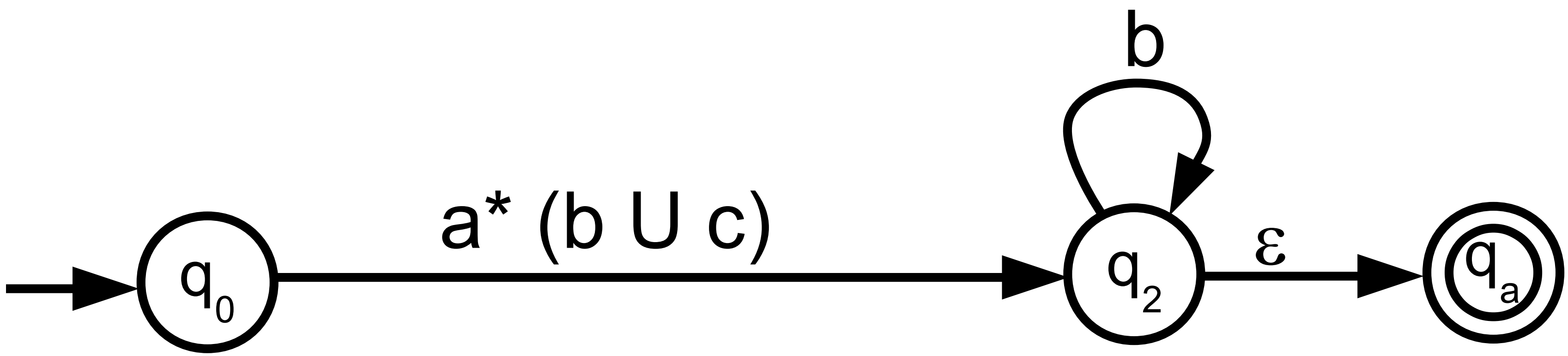
Example: DFA → GNFA → RE



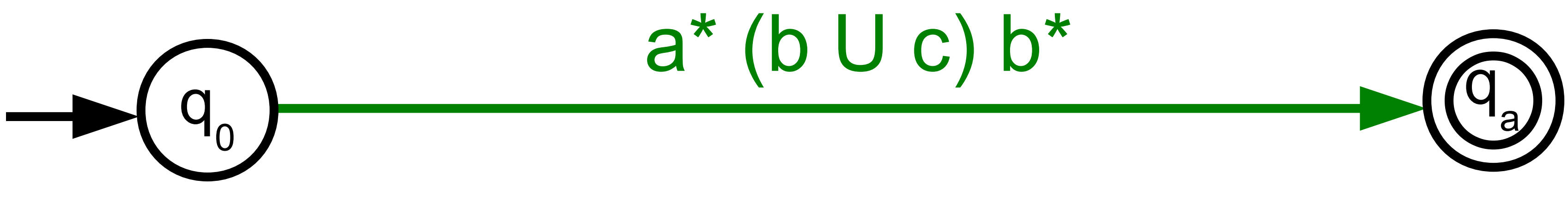
Eliminate q_2: add edge to new GNFA



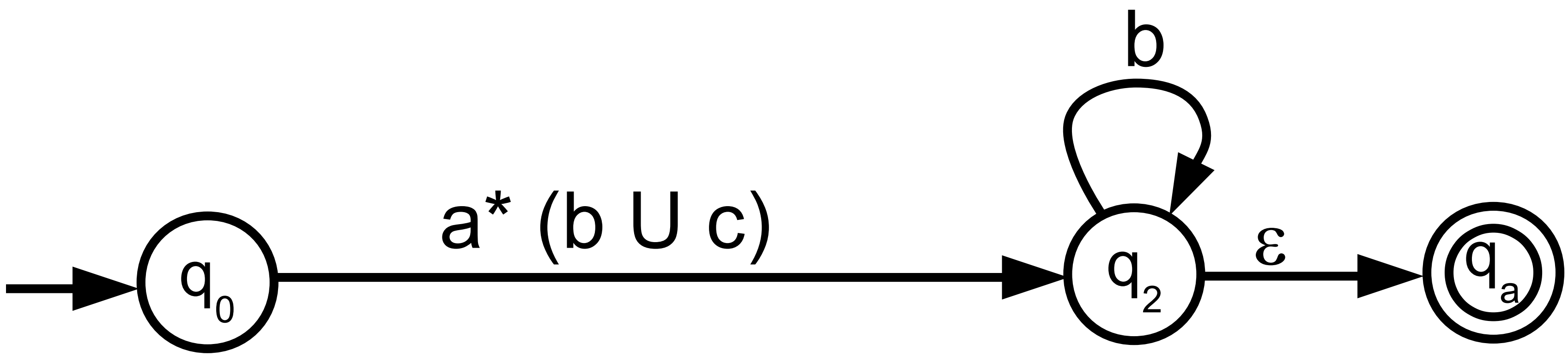
Example: DFA → GNFA → RE



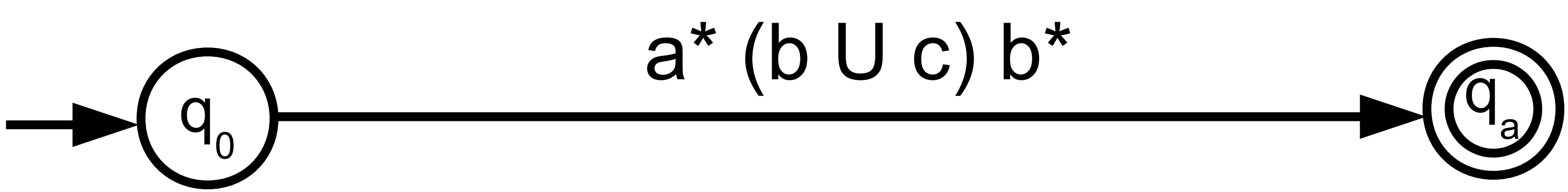
Eliminate q_2: simplify RE on new edge



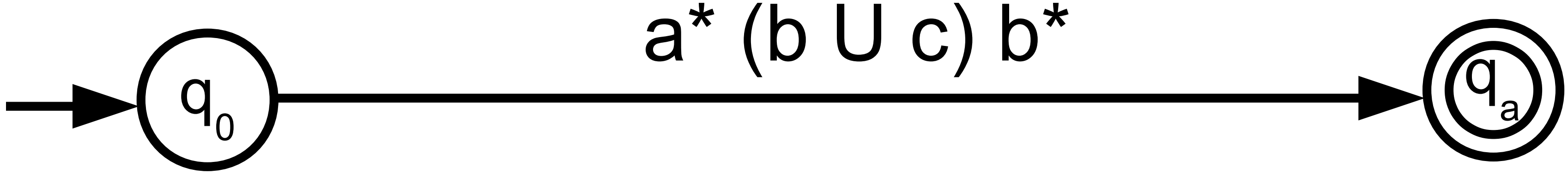
Example: DFA → GNFA → RE



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



Example: DFA → GNFA → RE

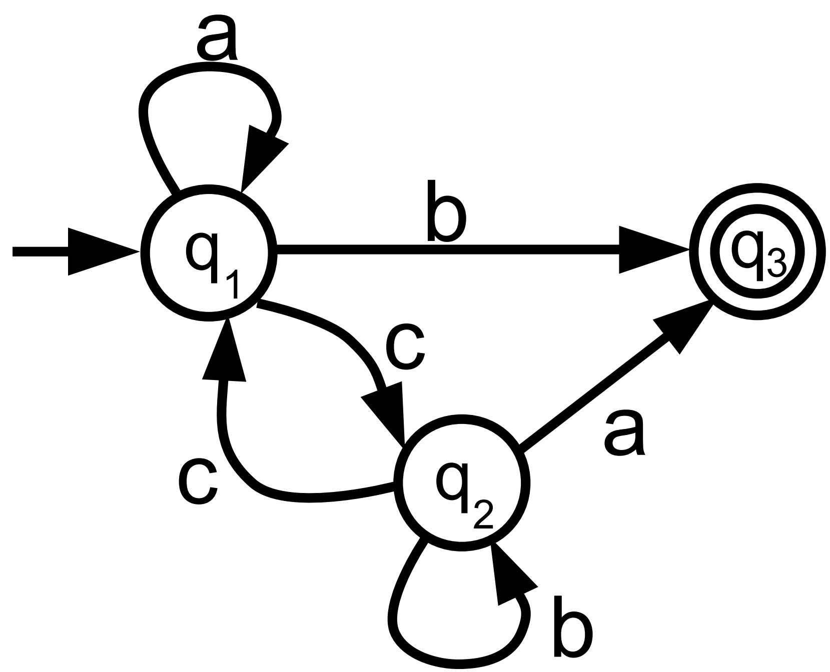


Only two states remain:

RE = a* (b U c) b*

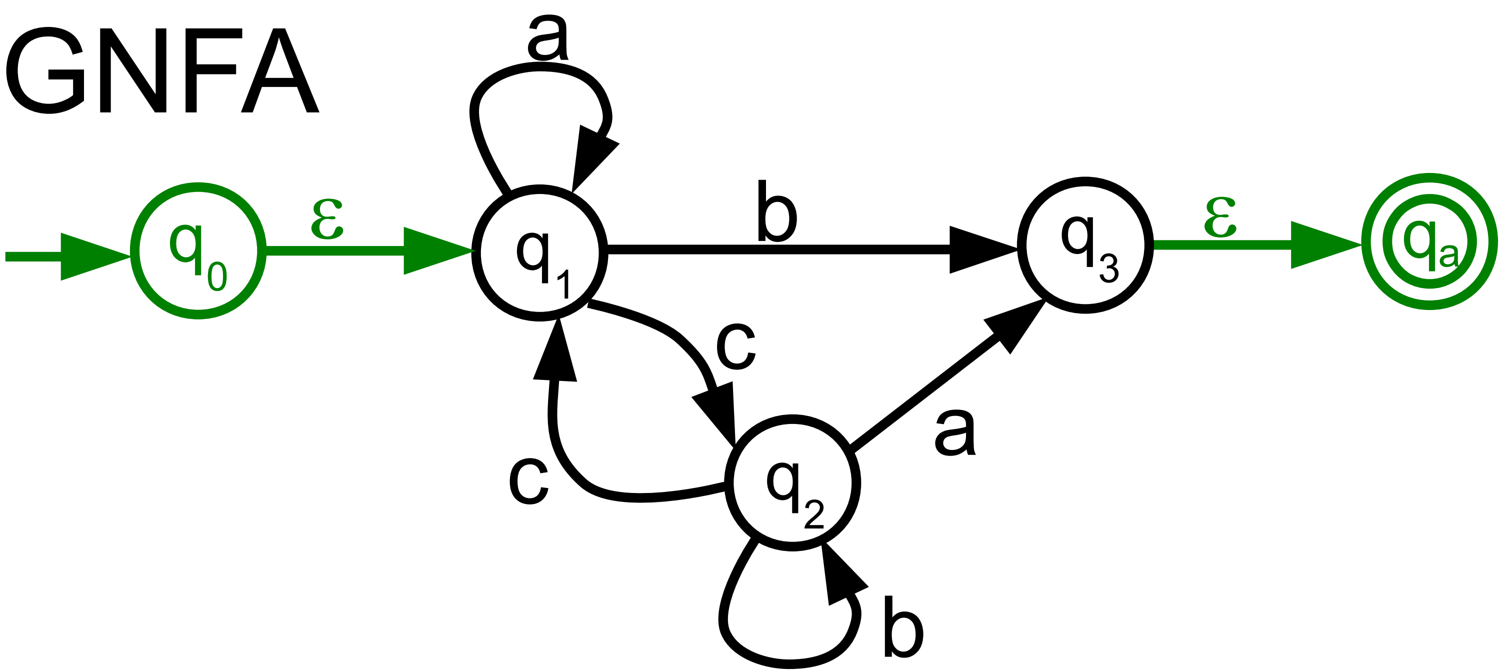
ANOTHER Example: DFA → GNFA → RE

DFA

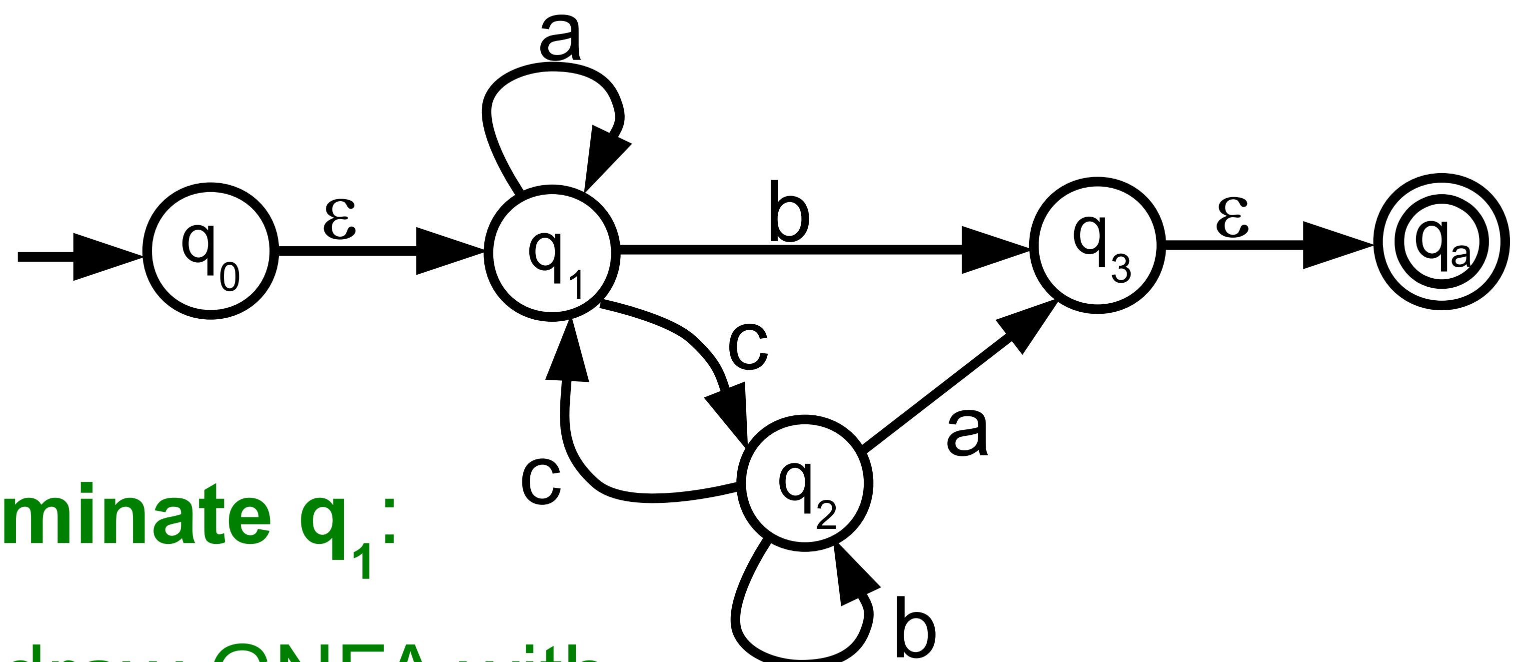


ANOTHER Example: DFA → GNFA → RE

GNFA



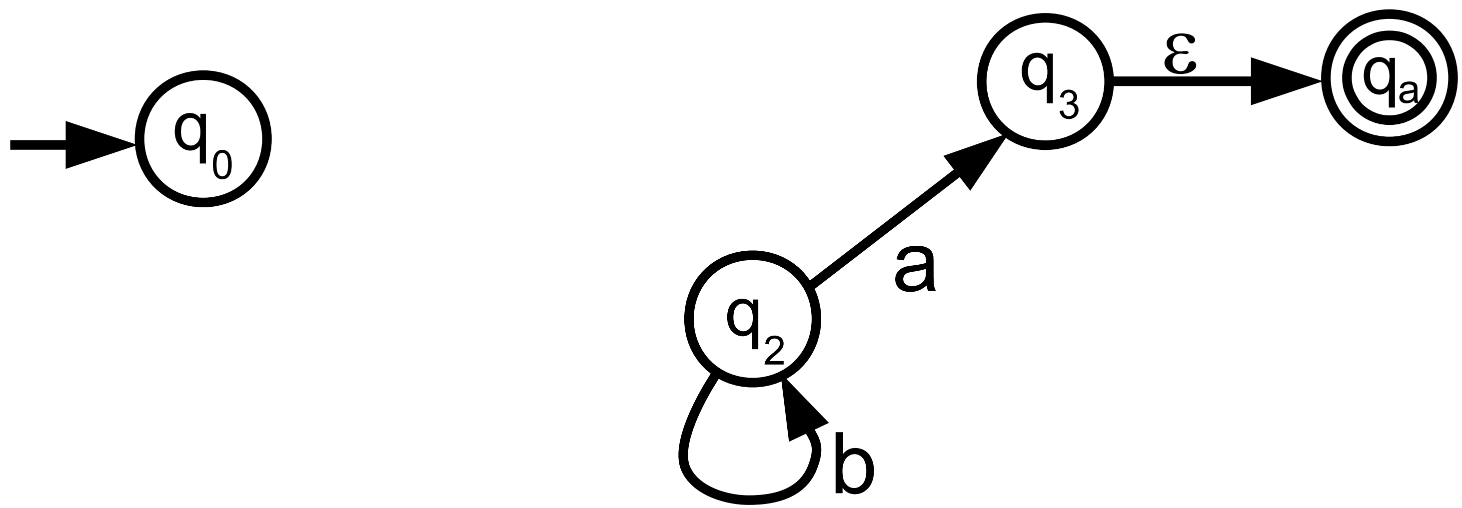
ANOTHER Example: DFA → GNFA → RE



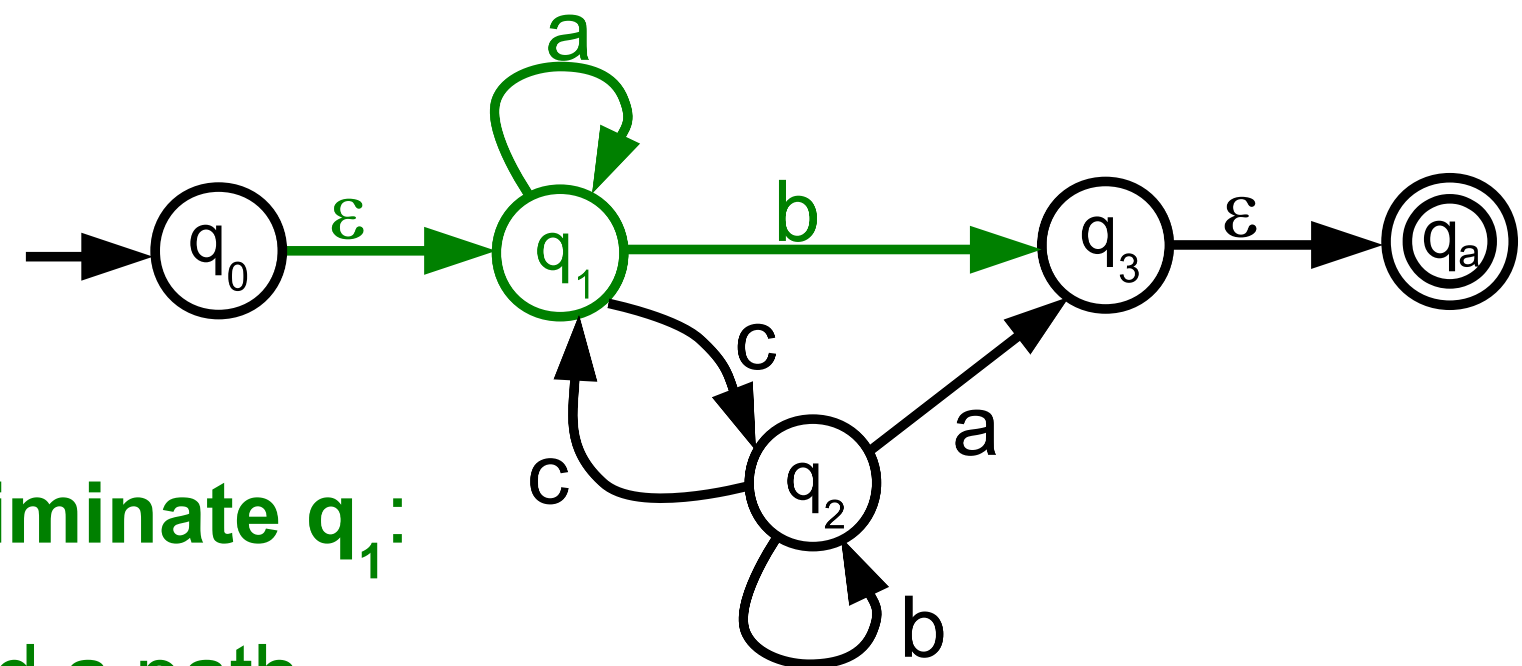
Eliminate q_1:

re-draw GNFA with

all other states

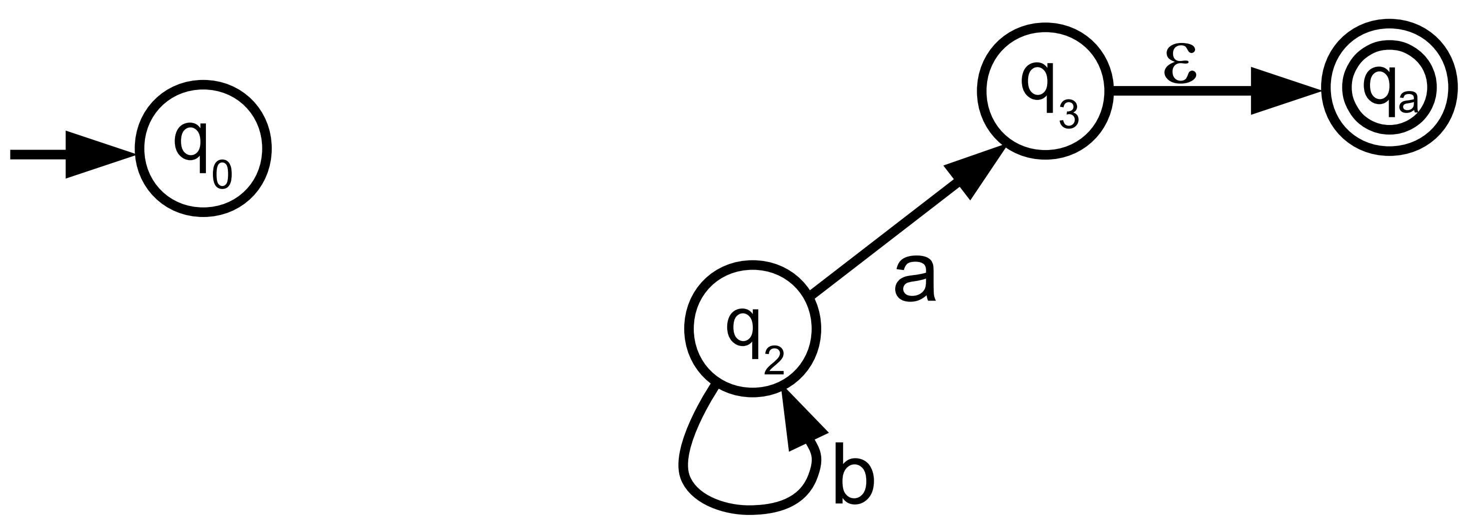


ANOTHER Example: DFA → GNFA → RE

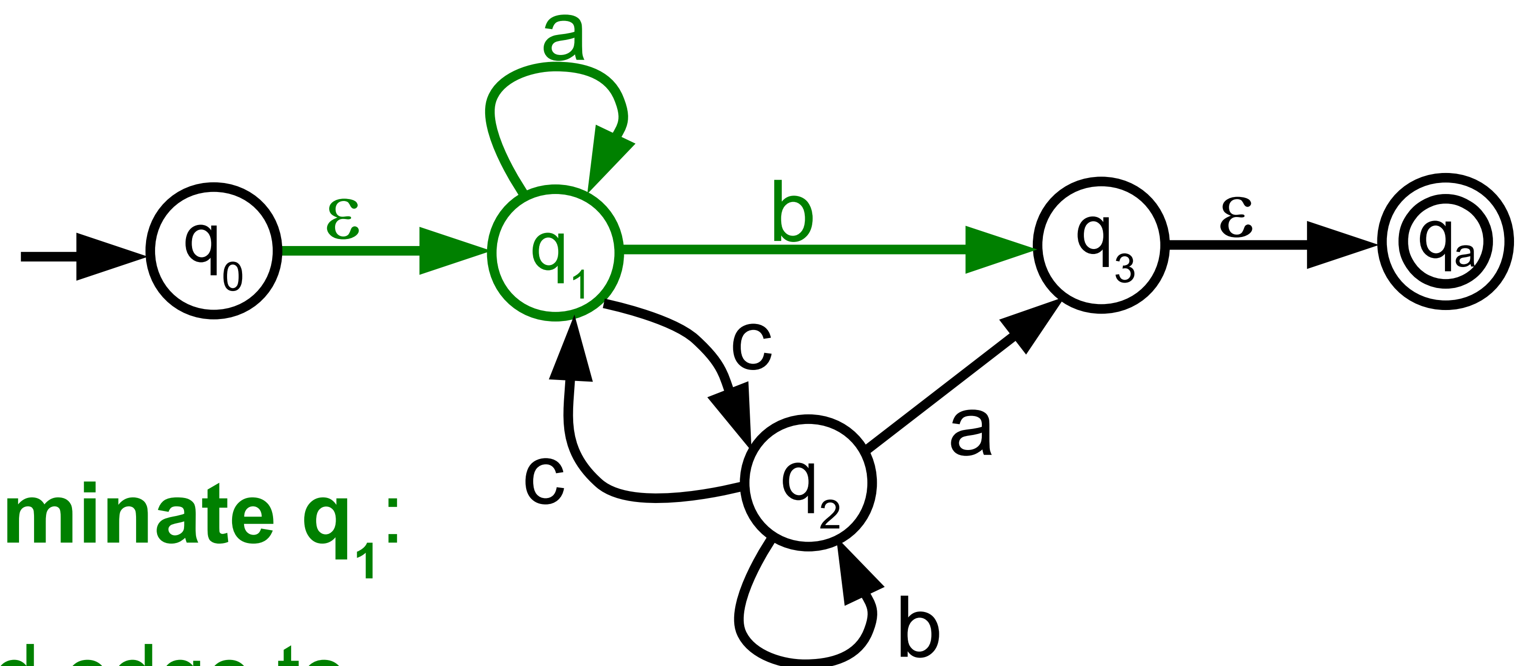


Eliminate q_1:

find a path through q_1



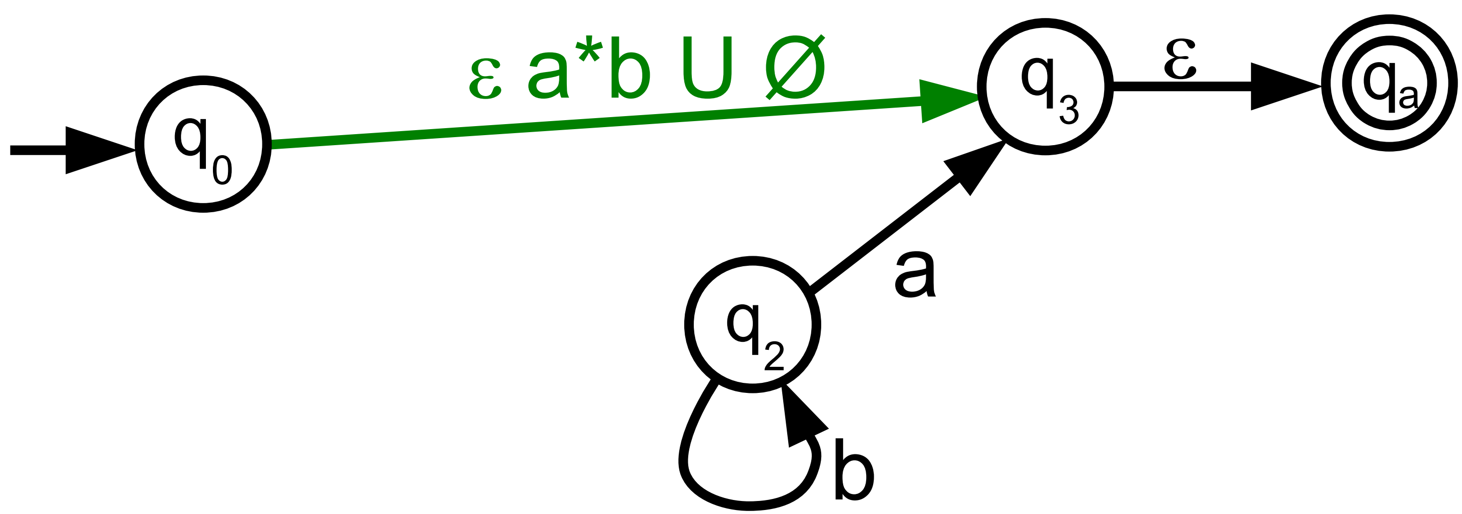
ANOTHER Example: DFA → GNFA → RE



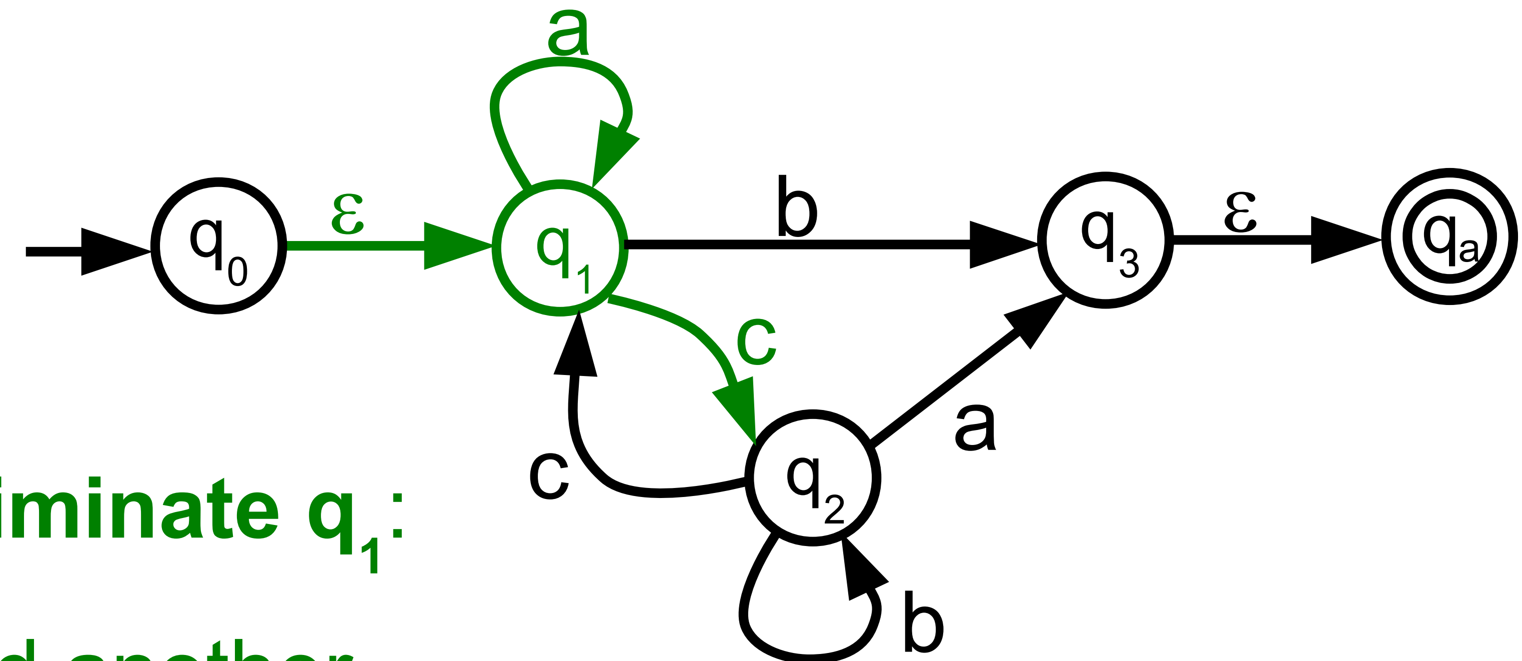
Eliminate q_1:

add edge to

new GNFA

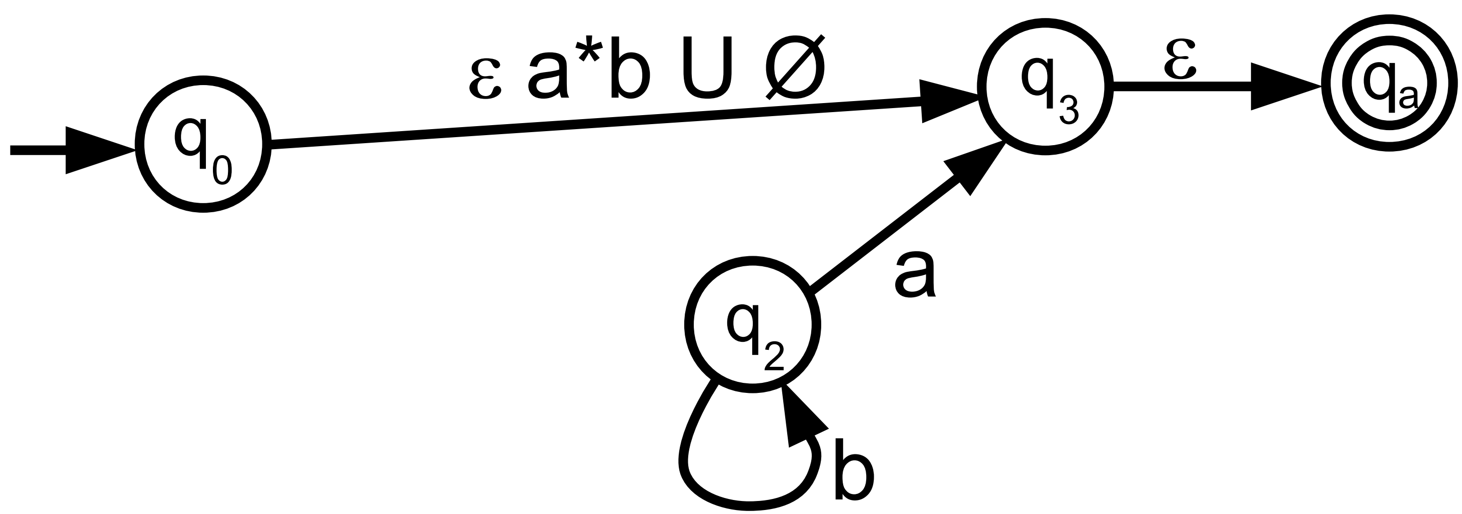


ANOTHER Example: DFA → GNFA → RE

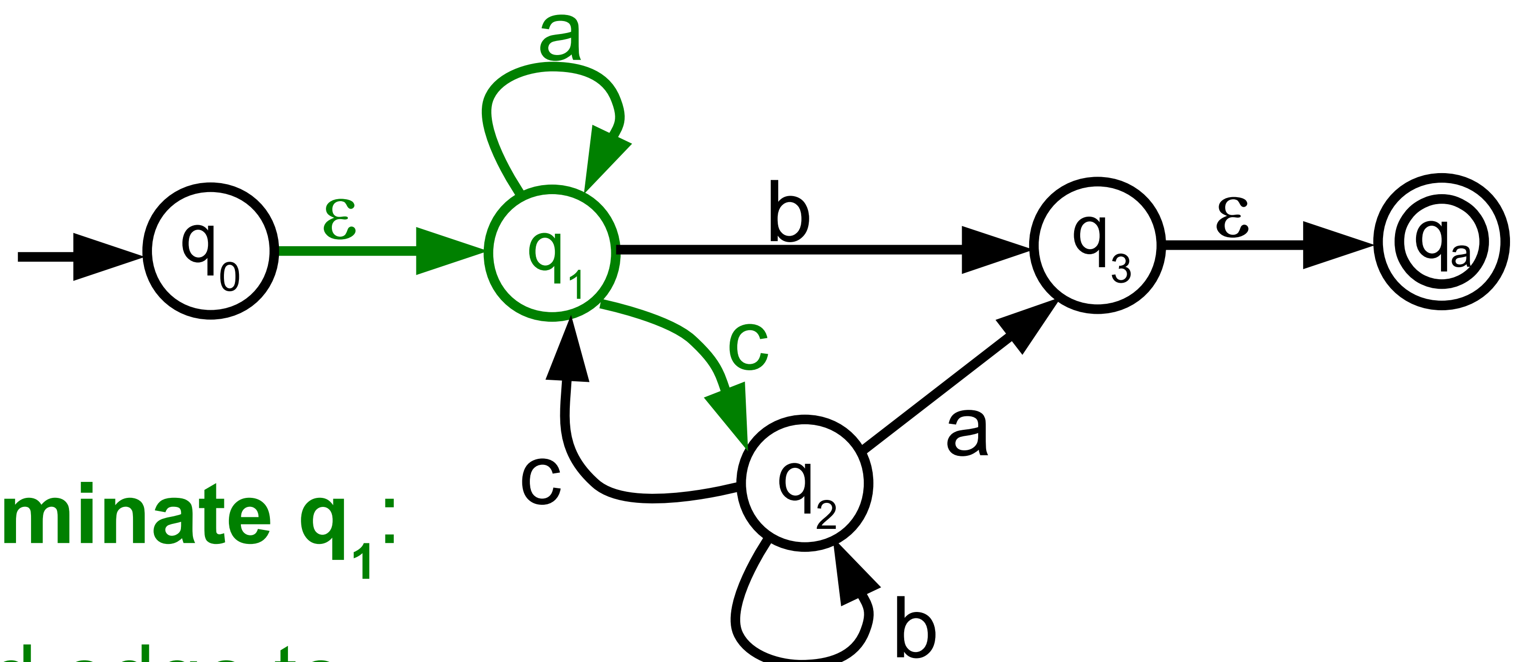


Eliminate q_1:

find another path through q_1

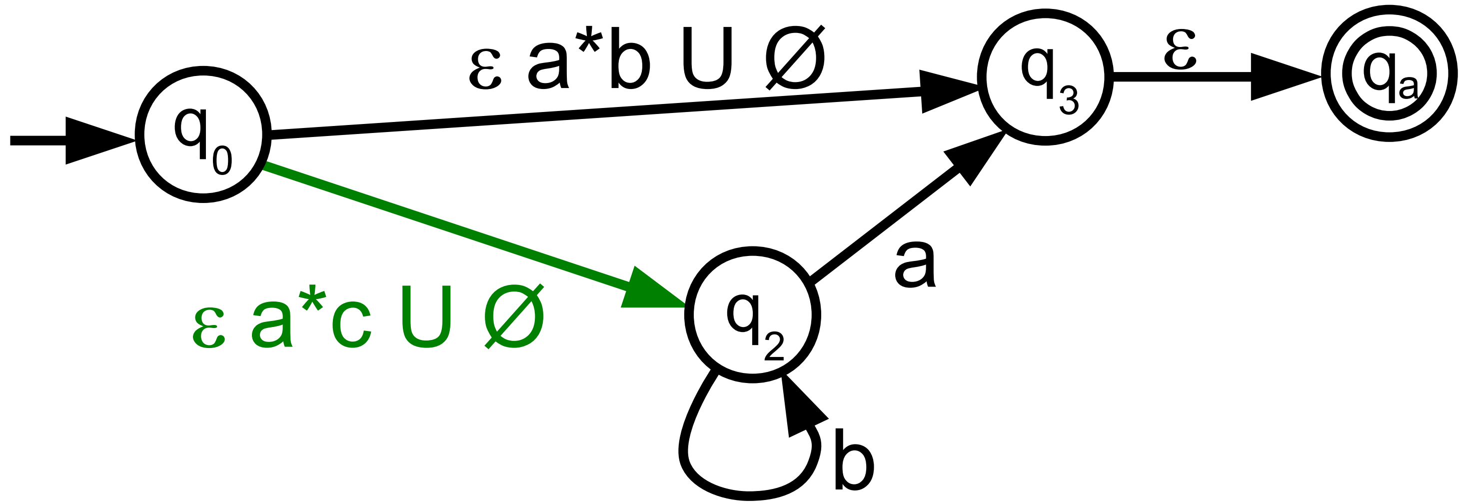


ANOTHER Example: DFA → GNFA → RE

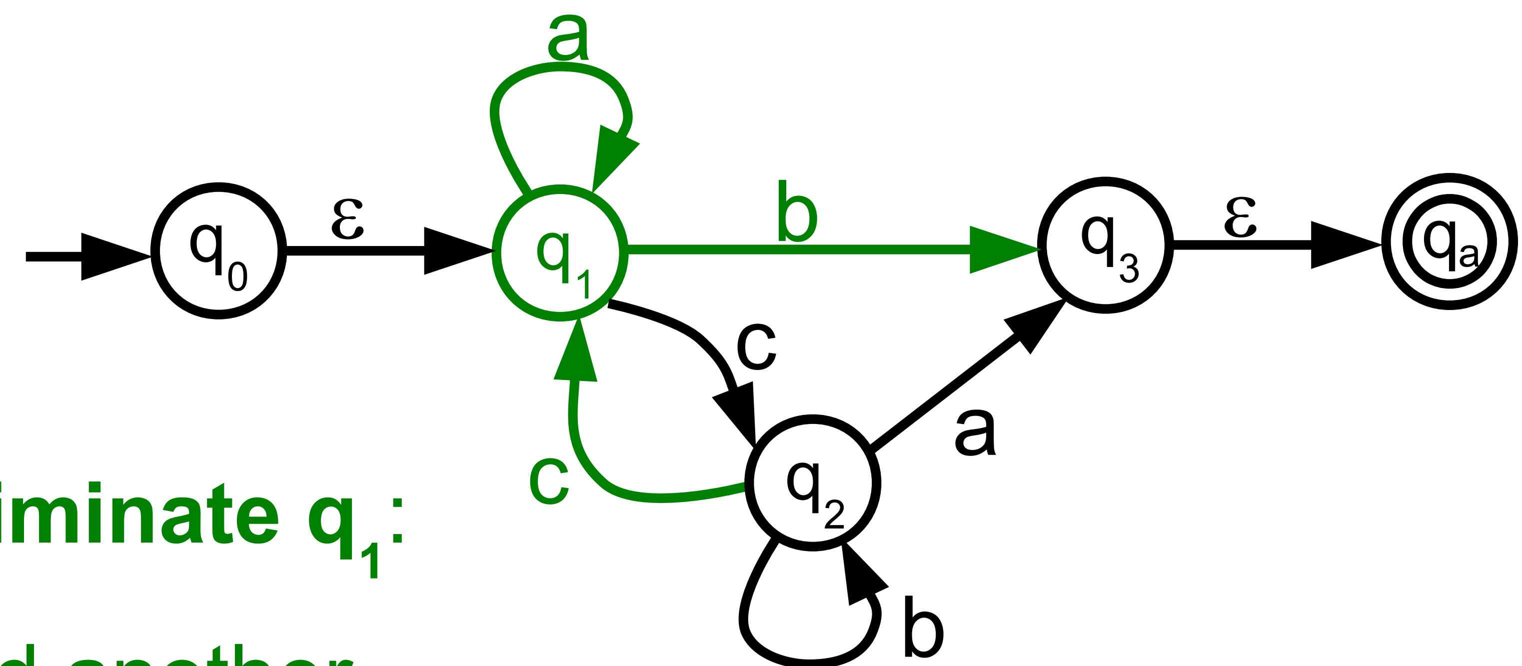


Eliminate q_1:

add edge to new GNFA

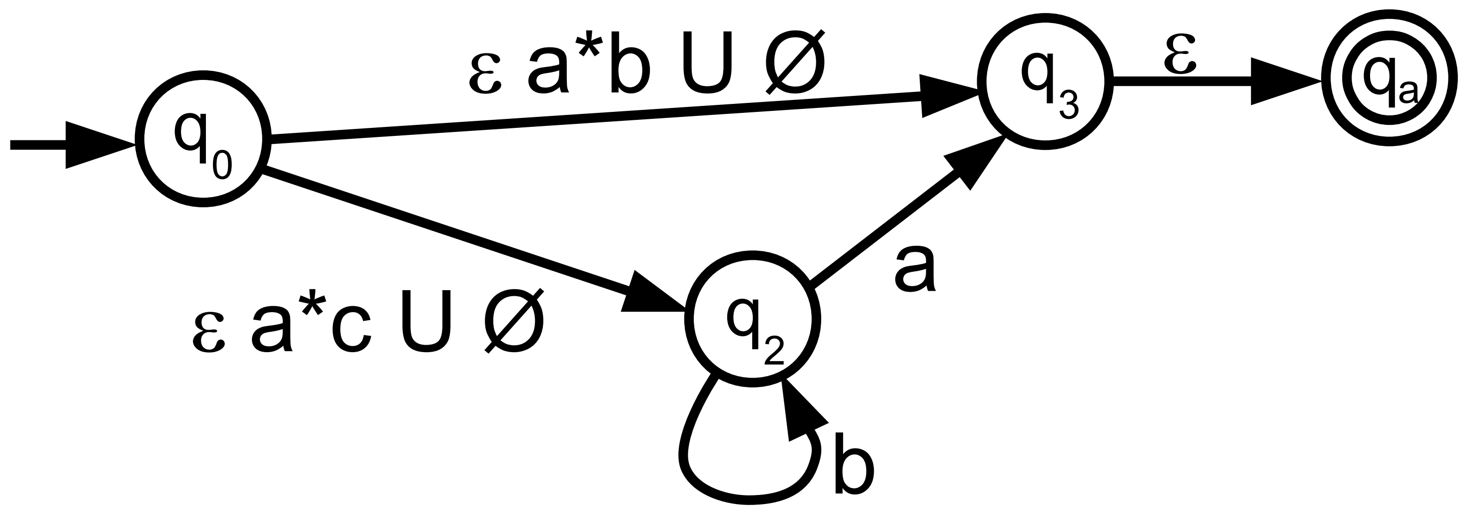


ANOTHER Example: DFA → GNFA → RE

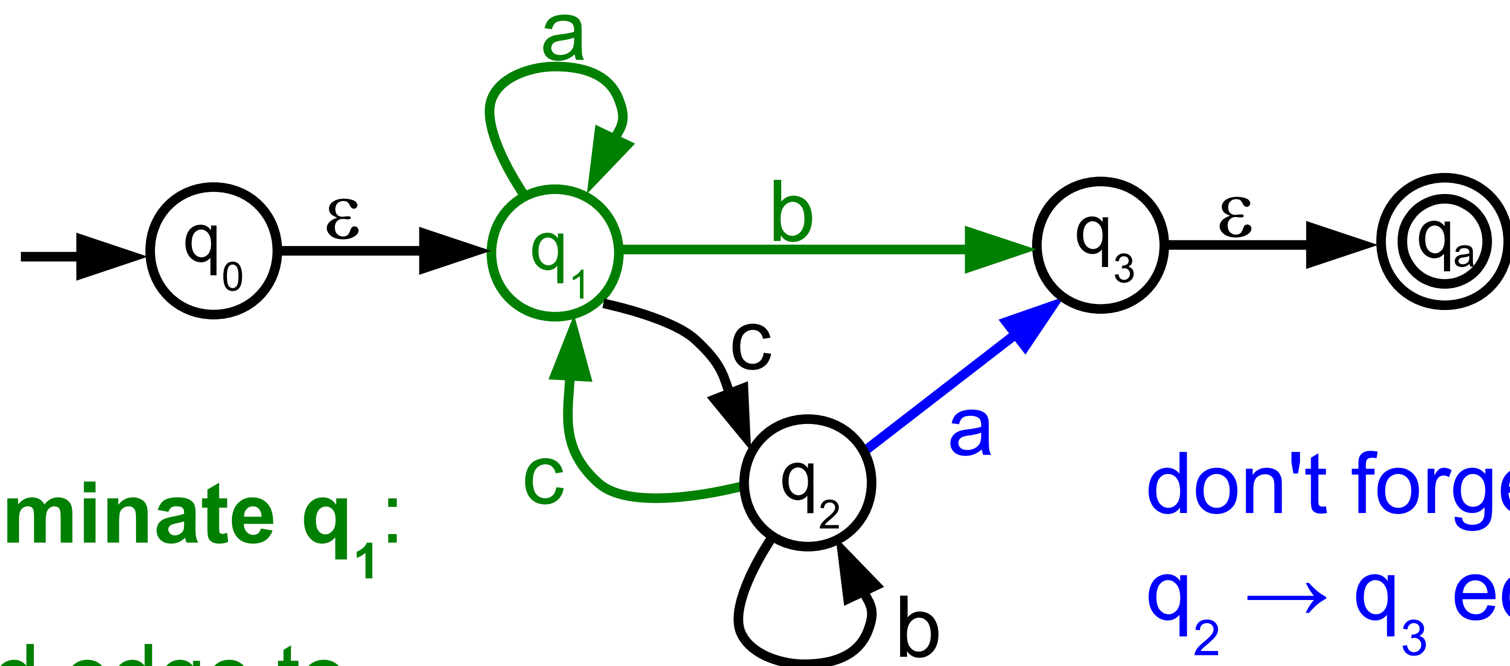


Eliminate q_1:

find another path through q_1



ANOTHER Example: DFA → GNFA → RE

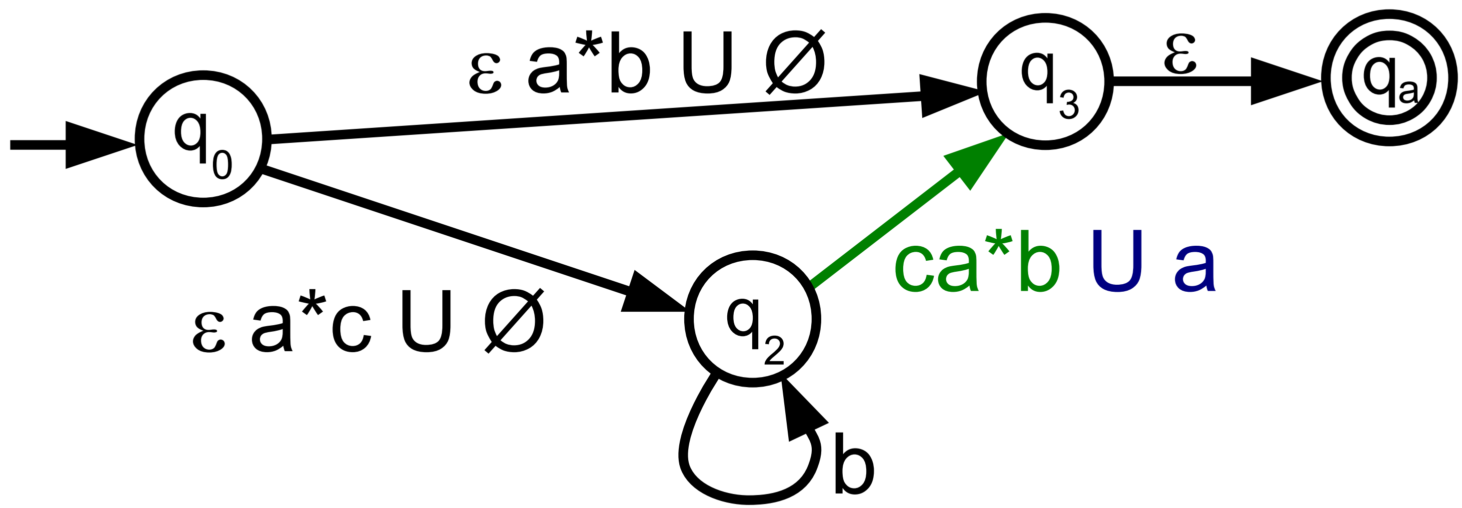


Eliminate q_1:

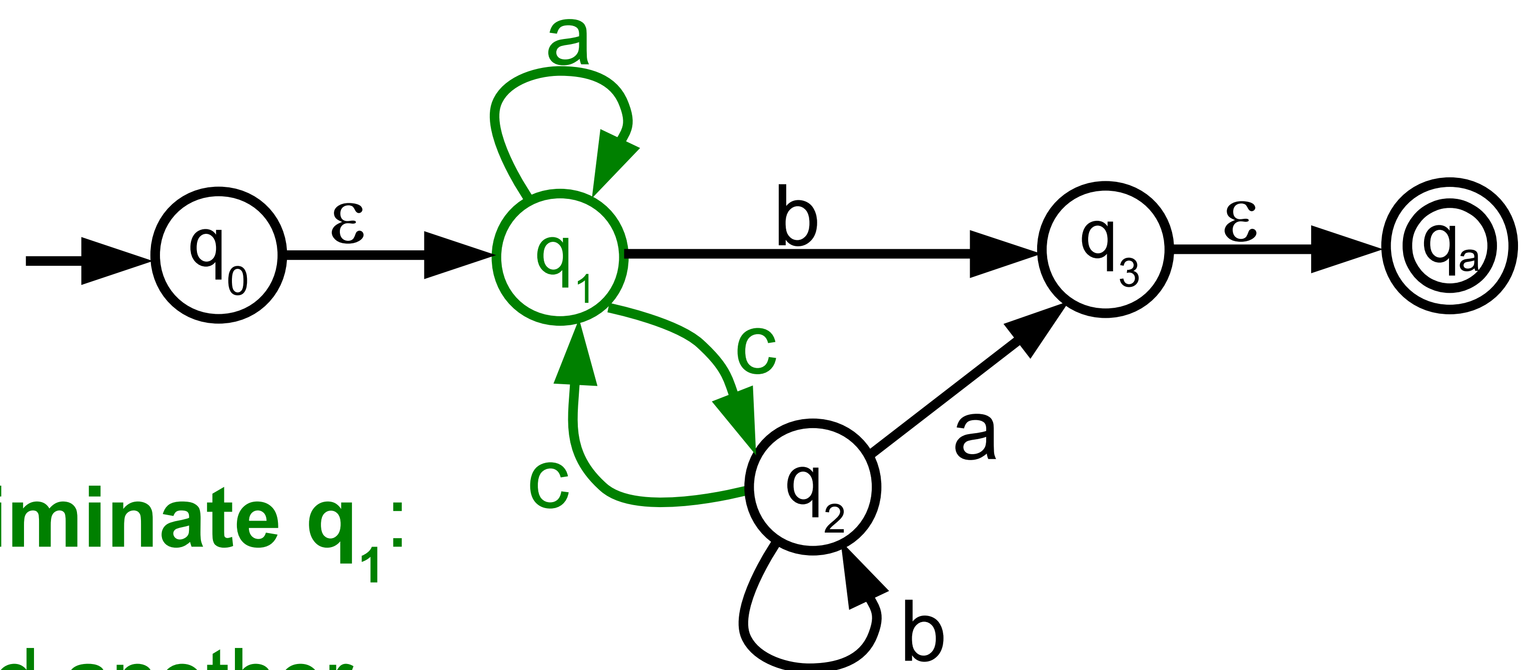
add edge to new GNFA

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

This time is not Ø !

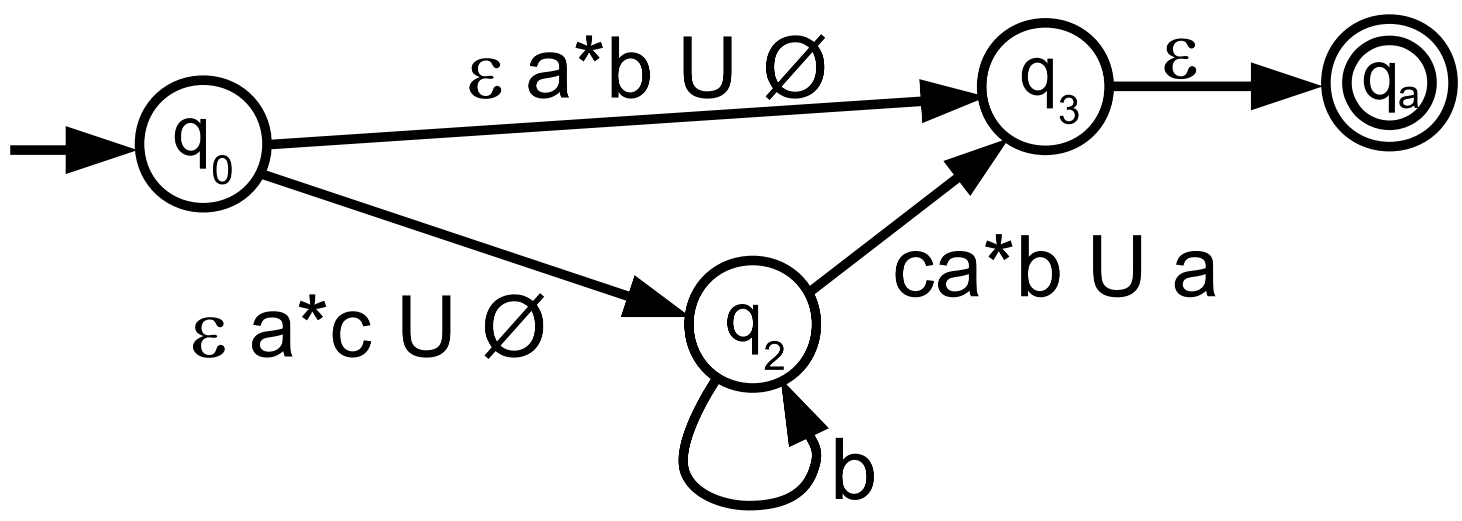


ANOTHER Example: DFA → GNFA → RE

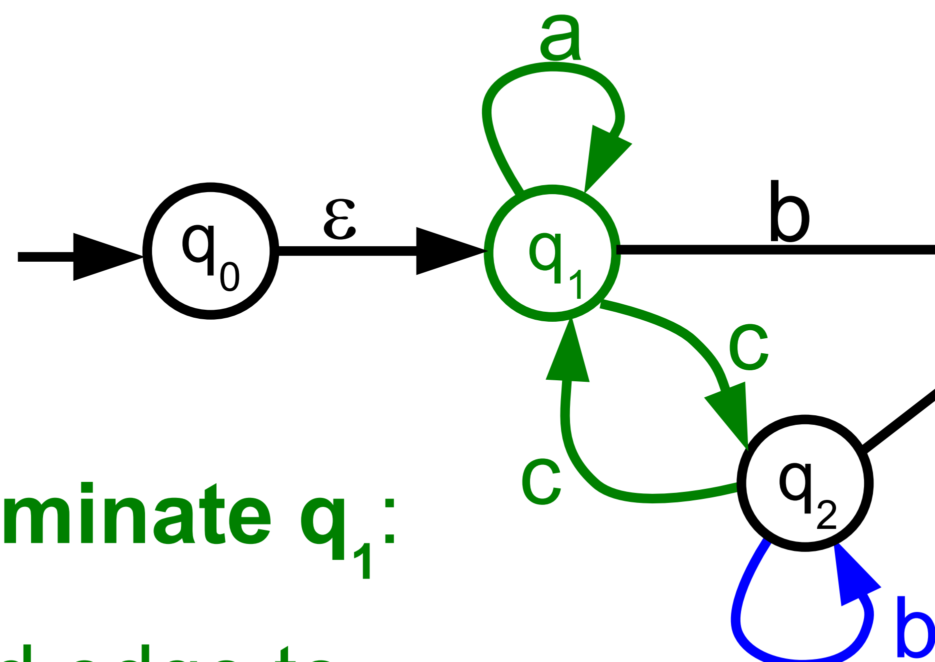


Eliminate q_1:

find another path through q_1



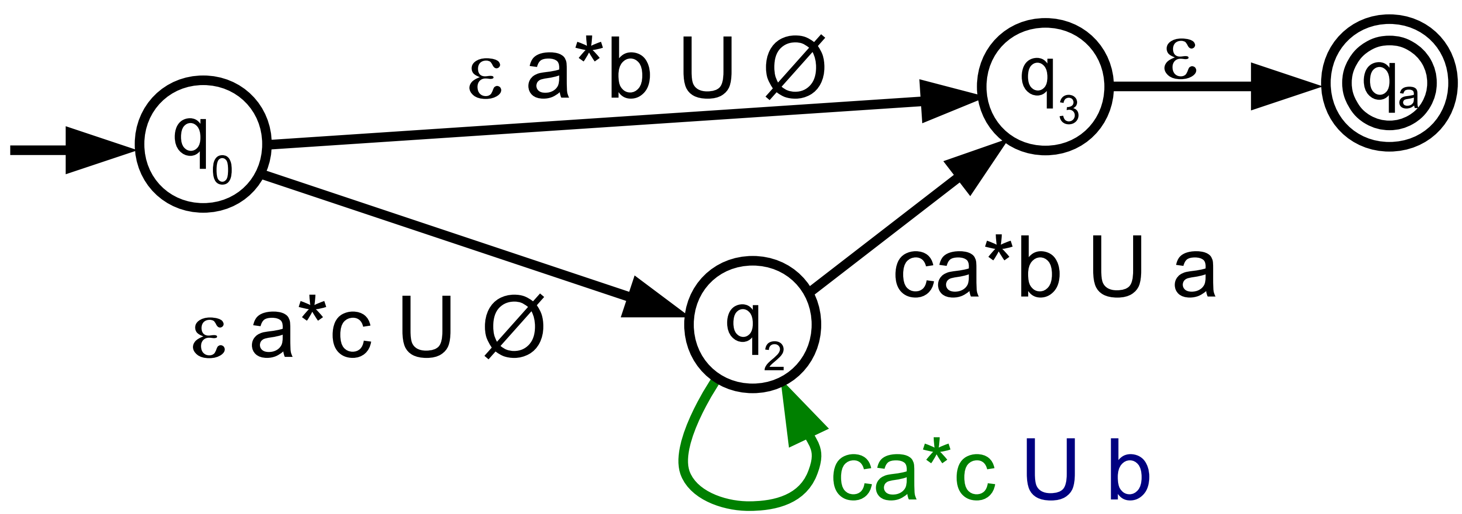
ANOTHER Example: DFA → GNFA → RE



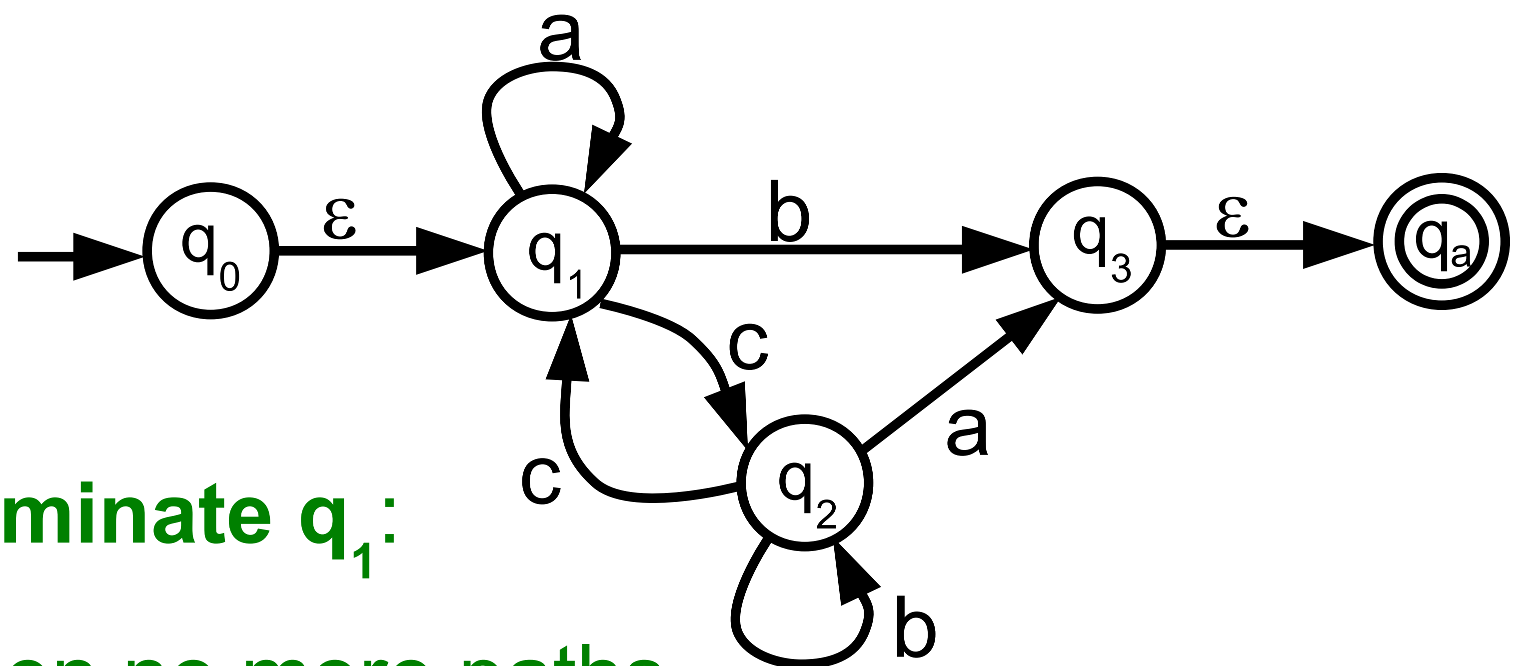
Eliminate q_1:

add edge to new GNFA

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



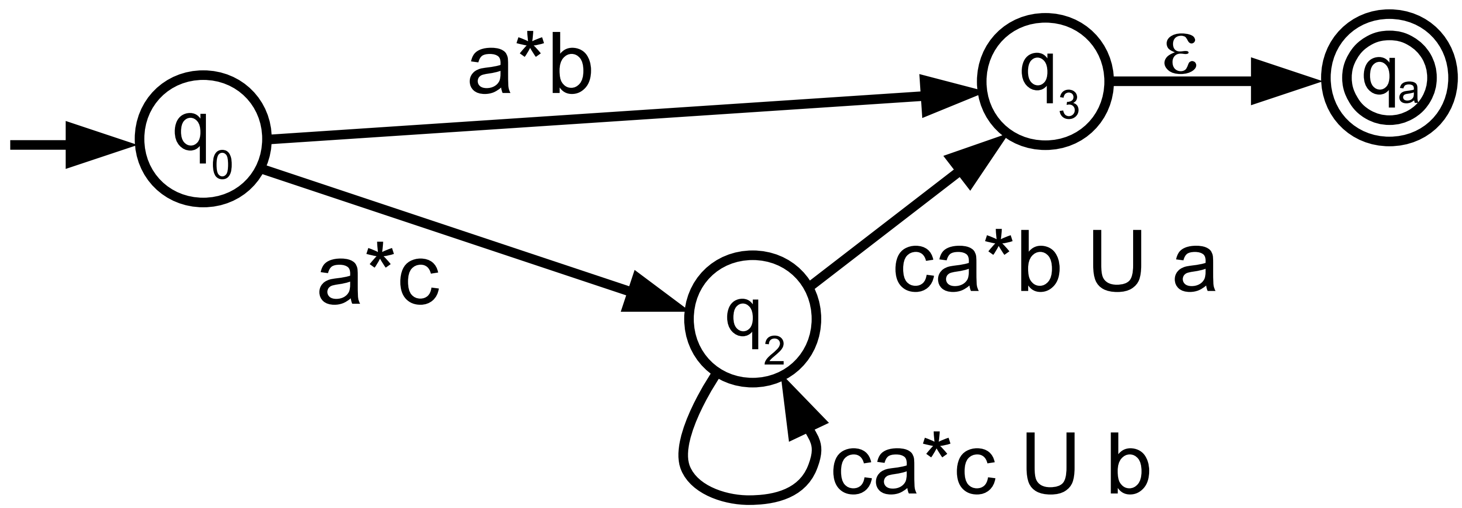
ANOTHER Example: DFA → GNFA → RE



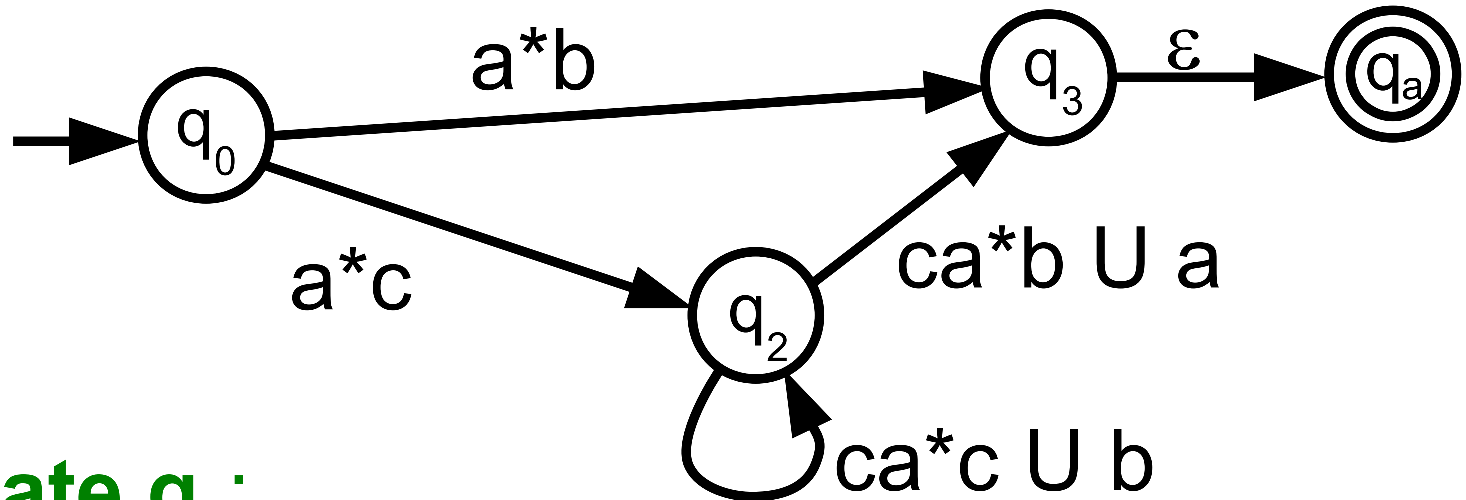
Eliminate q_1:

when no more paths through q_1, start over

(and simplify REs)



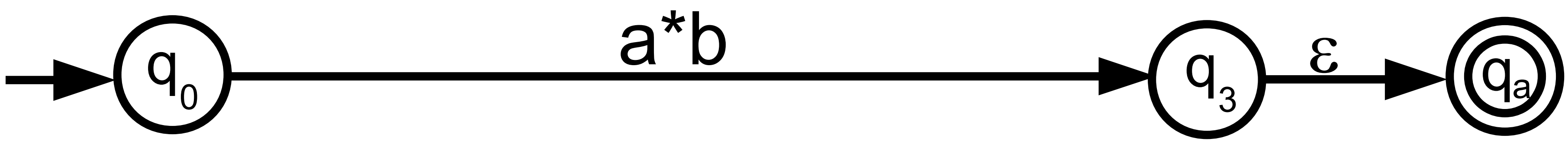
ANOTHER Example: DFA → GNFA → RE



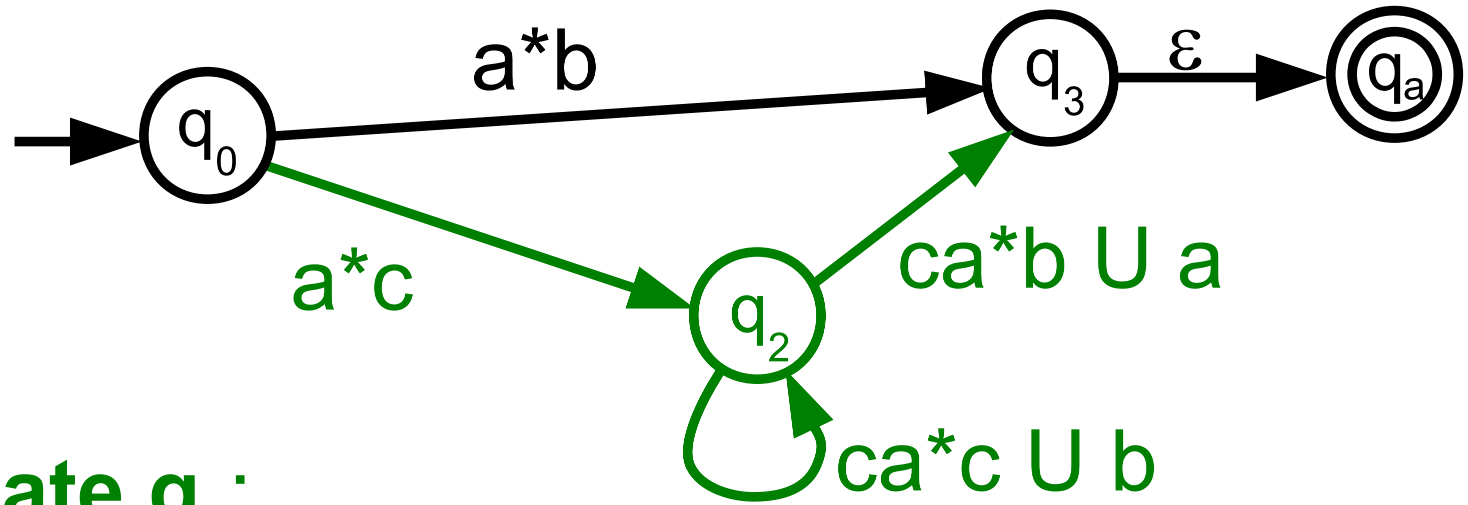
Eliminate q_2:

re-draw GNFA with

all other states

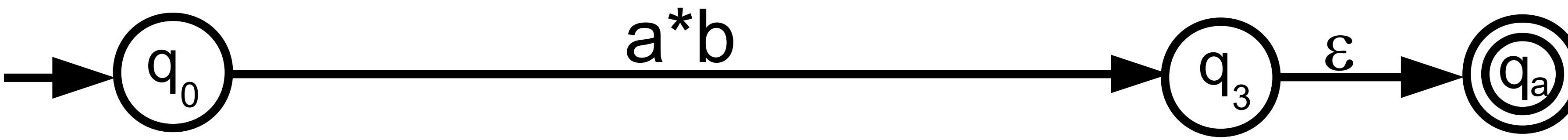


ANOTHER Example: DFA → GNFA → RE

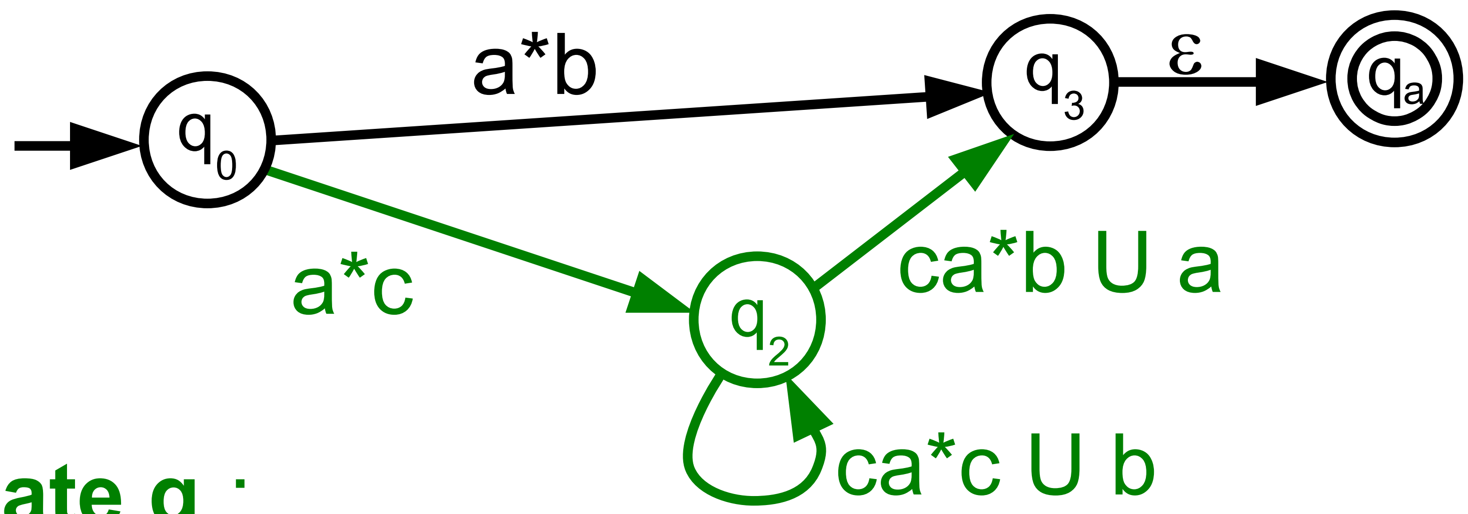


Eliminate q_2:

find a path through q_2



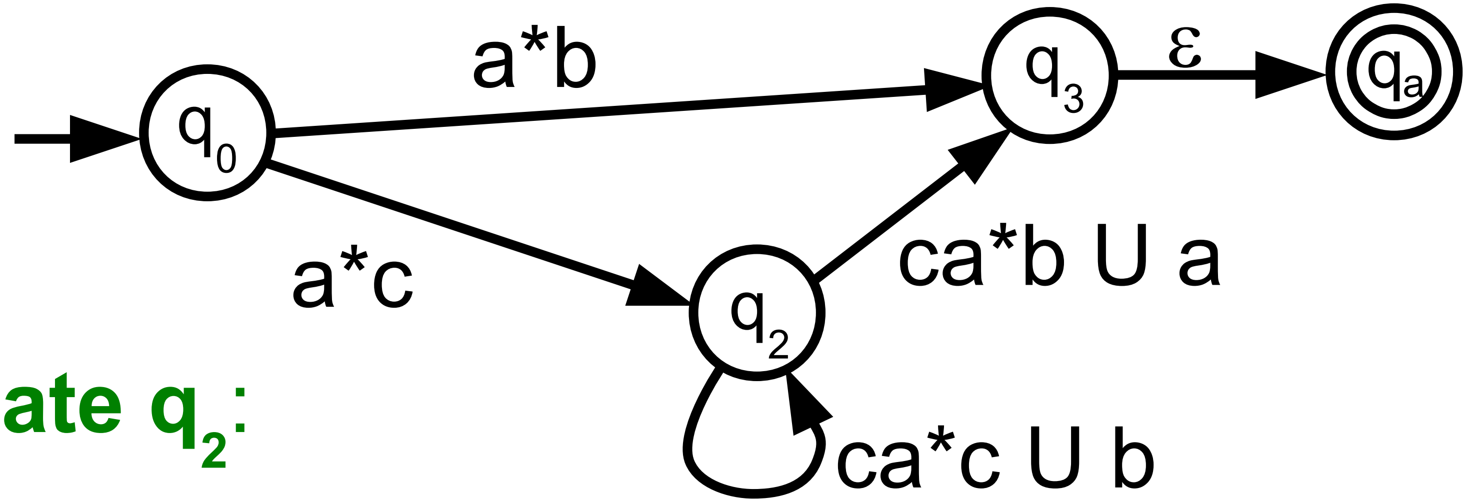
ANOTHER Example: DFA → GNFA → RE



Eliminate q_2:

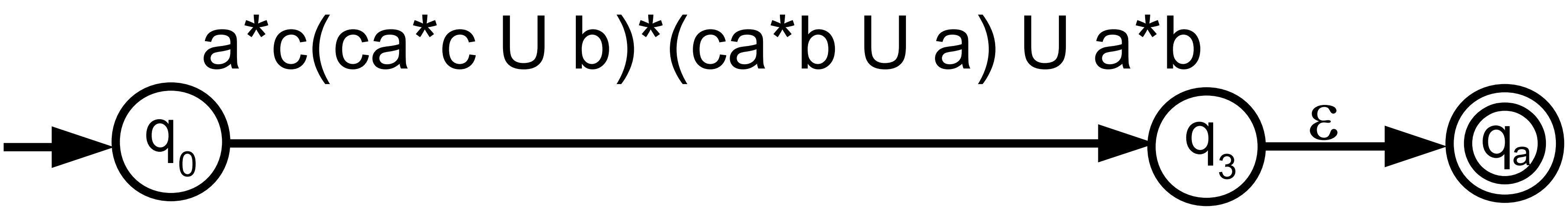
add edge to new GNFA

ANOTHER Example: DFA → GNFA → RE

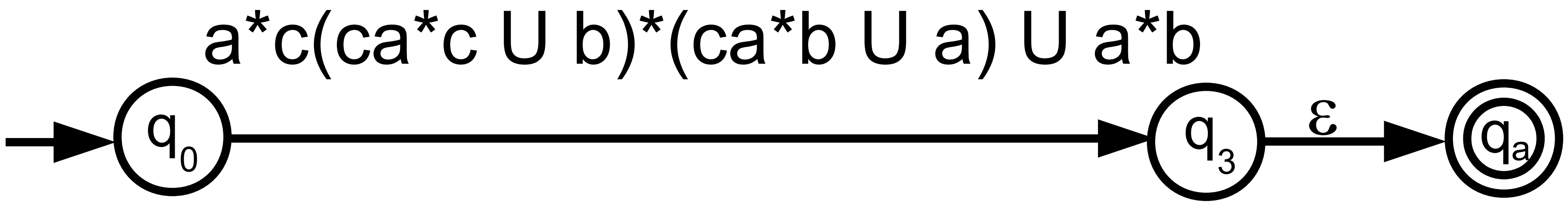


Eliminate q_2:

when no more paths through q_2, start over



ANOTHER Example: DFA → GNFA → RE

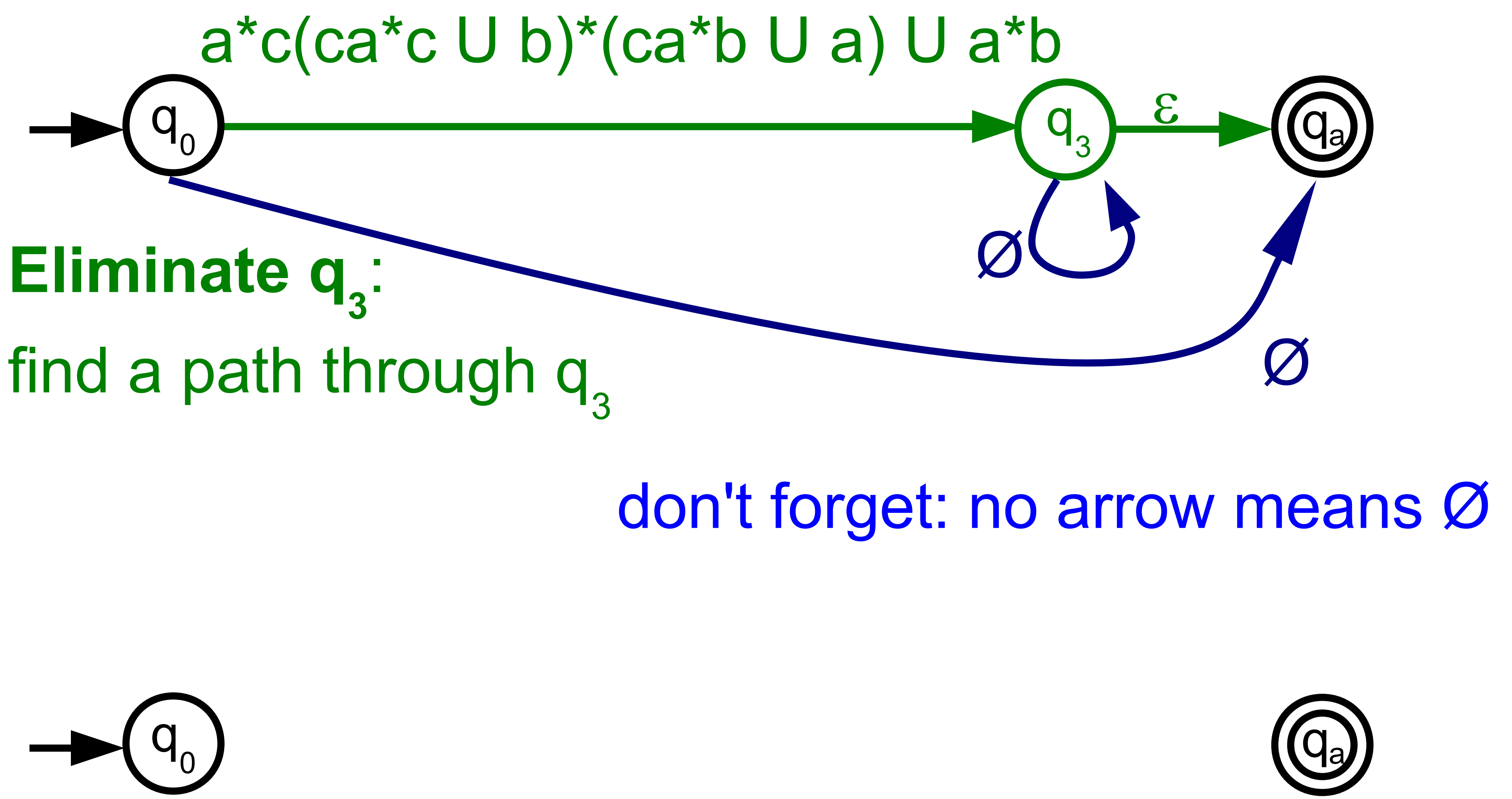


Eliminate q_3:

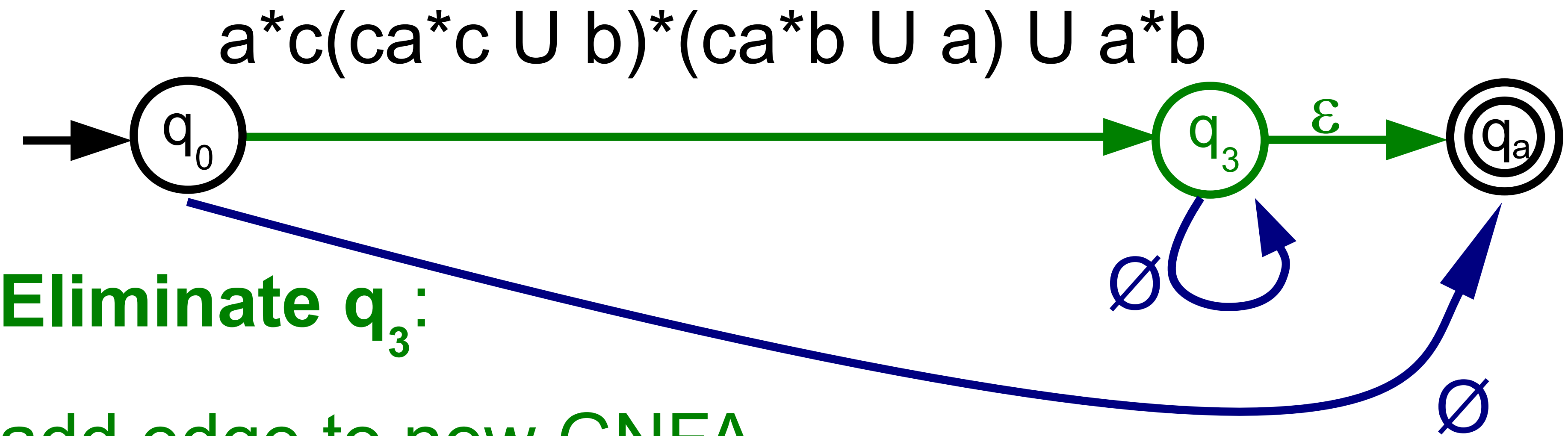
re-draw GNFA with

all other states

ANOTHER Example: DFA → GNFA → RE

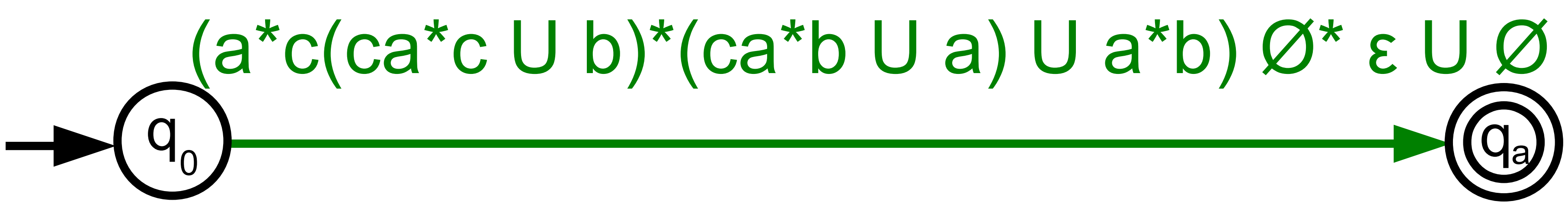


ANOTHER Example: DFA → GNFA → RE

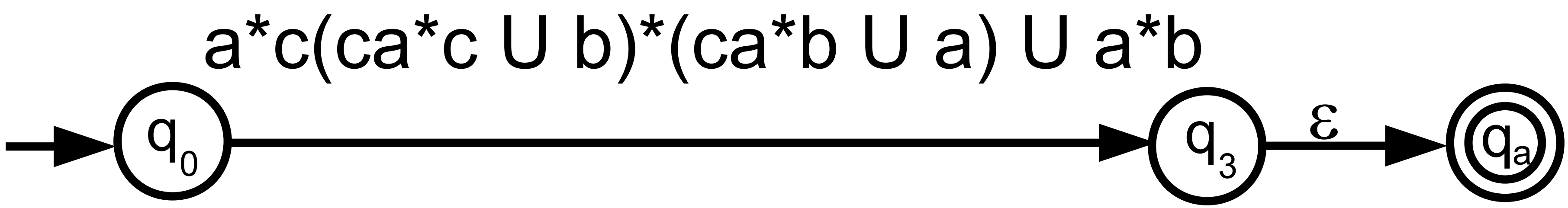


Eliminate q_3:

add edge to new GNFA



ANOTHER Example: DFA → GNFA → RE

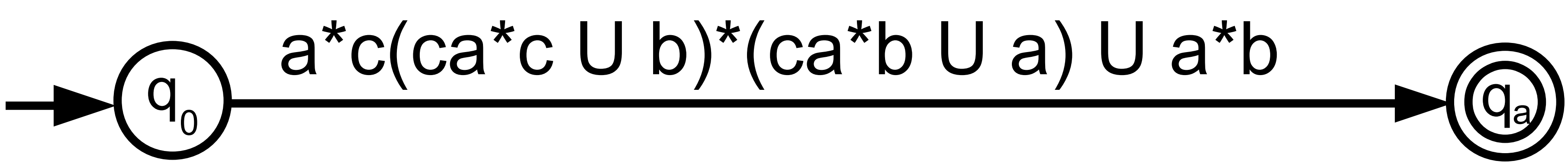


Eliminate q_3:

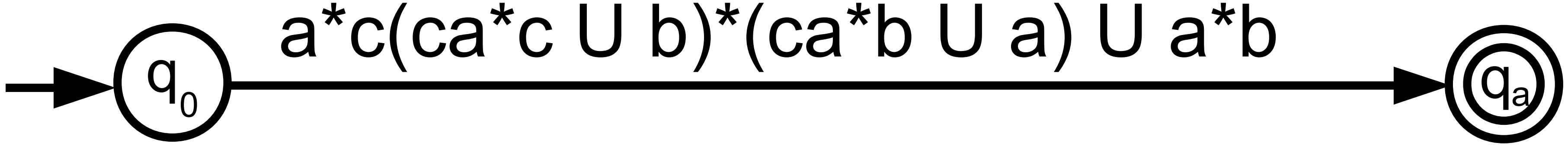
when no more paths through q_3, start over

(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 $\Rightarrow$

$\exists\ p \geq 0$

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

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

$\forall\ i \geq 0 : xy^i z \in L$

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

Pumping lemma:

L regular language $\Rightarrow$

$\exists\ p \geq 0$

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

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

$\forall\ i \geq 0 : xy^i z \in L$

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 $\Rightarrow$

$\exists\ p \geq 0$	A
$\forall\ w \in L,\ \|w\| \geq p$	
$\exists\ x,y,z : w = xyz,\ \|y\| > 0,\ \|xy\| \leq p$	
$\forall\ i \geq 0 : xy^iz \in L$	

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, $\vert w\vert \geq p$	
$\forall$ x,y,z : w = xyz, $\vert y\vert > 0$, $\vert xy\vert \leq p$	
$\exists$ 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)

$\forall\, p \geq 0$ $\exists\, w \in L,\ \lvert w\rvert \geq p$ $\forall\, x,y,z : w = xyz,\ \lvert y\rvert > 0,\ \lvert xy\rvert \leq p$ $\exists\, i \geq 0 : xy^iz \notin L$	$\Rightarrow$ 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 $\lvert y\rvert > 0$, $\lvert xy\rvert \leq p$

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

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

$\forall\ p \geq 0$

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

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

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

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

Same Proof:

Use pumping lemma

Adversary moves p

You move w := ?

$\forall$ 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 ∈ L, |w| ≥ p

$\forall$ x,y,z : w = xyz, |y| > 0, |xy| ≤ 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 $|xy| \le p$ 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, $\|y\| > 0$, $\|xy\| \leq p$
$\exists$ $i \geq 0$: $xy^iz \notin L$

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 $j > k$

$\forall\ p \geq 0$

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

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

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

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

Proof:

Use pumping lemma

Adversary moves p

You move w := ?

$\forall$ 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:

$\forall\ p \geq 0$
$\exists\ w \in L, \lvert w\rvert \geq p$
$\forall\ x,y,z : w = xyz, \lvert y\rvert > 0, \lvert xy\rvert \leq p$
$\exists\ i \geq 0 : xy^iz \notin L$

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

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

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

Theorem: $L := \{ 1^{n^2} : n \geq 0 \}$ 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: $L := \{ 1^{n^2} : n \geq 0 \}$ 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:

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

$\forall\ p \geq 0$

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

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

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

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

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

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

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