15-453

FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY

THE PUMPING LEMMA FOR REGULAR LANGUAGES and REGULAR EXPRESSIONS

TUESDAY Jan 21

WHICH OF THESE ARE REGULAR ?

$B = \{0^n 1^n \mid n \geq 0\}$

C = { w | w has equal number of occurrences of 01 and 10 }

D = { w | w has equal number of 1s and 0s}

THE PUMPING LEMMA

Let L be a regular language with $|L| = \infty$

Then there is a positive integer P s.t.

if $w \in L$ and $|w| \geq P$

then can write $w = xyz$, where:

1. **$|y| > 0$ (y isn't ε)**
2. **$|xy| \leq P$**
3. **For *every* $i \geq 0$, $xy^iz \in L$**

Why is it called the pumping lemma? The word w gets PUMPED into something longer…

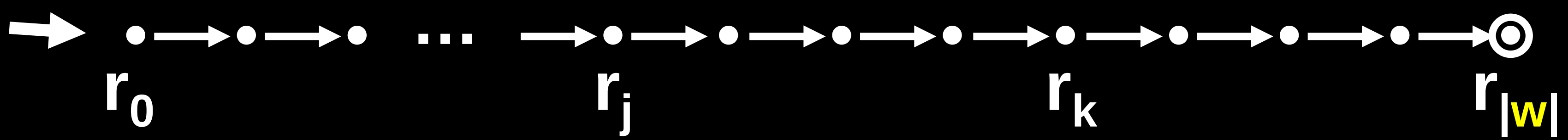
Proof: Let M be a DFA that recognizes L

Let P be the **number of states** in M

Assume $w \in L$ is such that $|w| \geq P$

We show: $w = xyz$

1. $|y| > 0$
2. $|xy| \leq P$
3. $xy^iz \in L$ for *all* $i \geq 0$



There must be j and k such that

$j < k \leq P$, and $r_j = r_k$ (why?) (Note: $k - j > 0$)

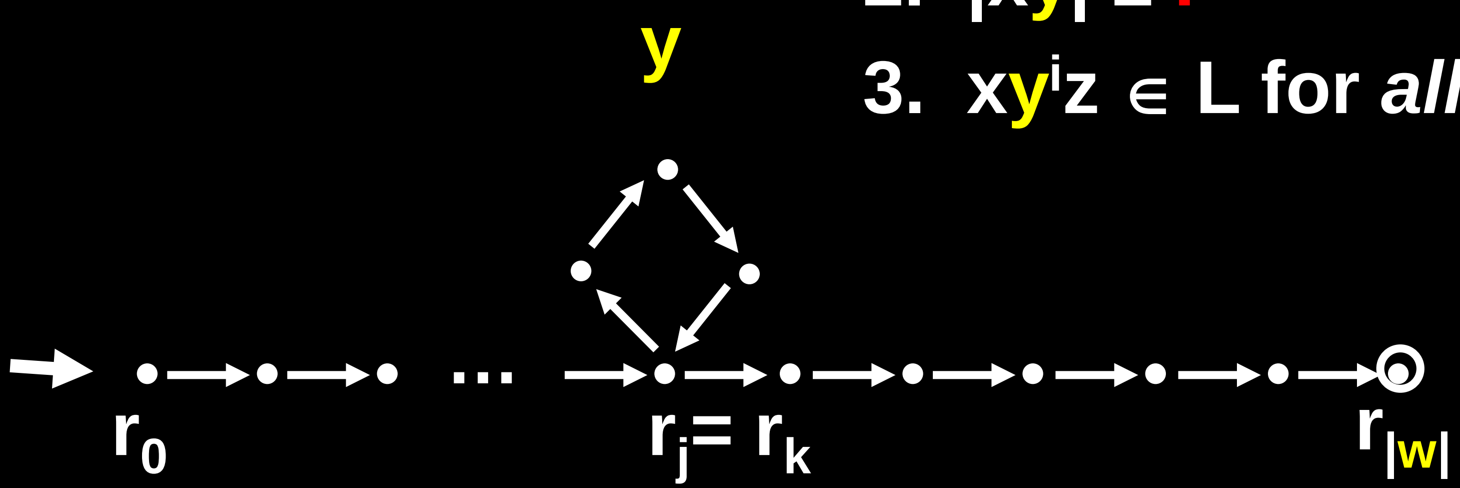
Proof: Let M be a DFA that recognizes L

Let P be the **number of states** in M

Assume $w \in L$ is such that $|w| \geq P$

We show: $w = xyz$

1. $|y| > 0$
2. $|xy| \leq P$
3. $xy^iz \in L$ for *all* $i \geq 0$



There must be j and k such that

$j < k \leq P$, and $r_j = r_k$

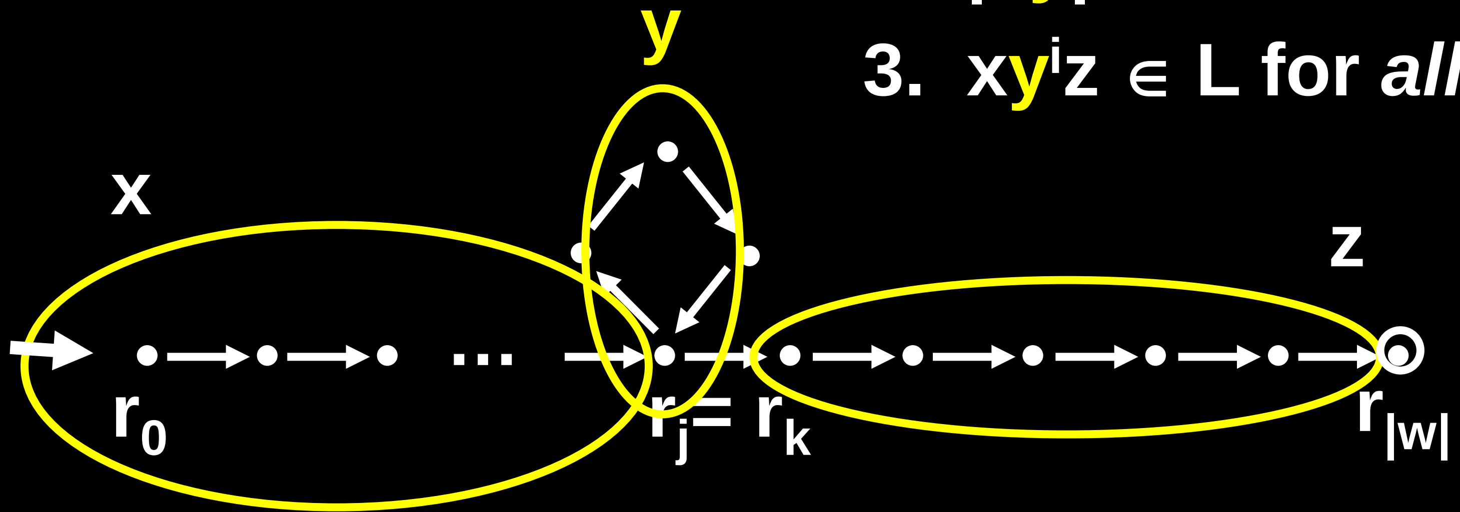
Proof: Let M be a DFA that recognizes L

Let P be the **number of states** in M

Assume w ∈ L is such that |w| ≥ P

We show: w = xyz

1. |y| > 0
2. |xy| ≤ P
3. xy^iz ∈ L for *all* i ≥ 0



There must be j and k such that
j < k ≤ P, and $r_j = r_k$

USING THE PUMPING LEMMA

Let's prove that
B = {0^n1^n | n ≥ 0} is not regular

Assume B is regular. Let w = 0^P1^P

If B is regular, can write w = xyz, |y| > 0, |xy| ≤ P, and for any i ≥ 0, xy^iz is *also* in B

y must be all 0s: Why? **|xy| ≤ P**

xyyz has more 0s than 1s

Contradiction!

USING THE PUMPING LEMMA

D = { w | w has equal number of 1s and 0s}

is not regular

Assume D is regular. Let $w = 0^P1^P$ (w is in D!)

If D is regular, can write w = xyz, |y| > 0, |xy| ≤ P, where for any i ≥ 0, xy^iz is *also* in D

y must be all 0s: Why? **|xy| ≤ P**

xyyz has more 0s than 1s

Contradiction!

WHAT DOES D LOOK LIKE?

D = { w | w has equal number of occurrences of 01 and 10}

WHAT DOES C LOOK LIKE?

C = { w | w has equal number of occurrences of 01 and 10}

= { w | w = 1, w = 0, w = ε or
w starts with a 0 and ends with a 0 or
w starts with a 1 and ends with a 1 }

$$1 \cup 0 \cup \varepsilon \cup 0(0\cup 1)^*0 \cup 1(0\cup 1)^*1$$

REGULAR EXPRESSIONS
(expressions representing languages)

σ is a regexp representing $\{\sigma\}$

ε is a regexp representing $\{\varepsilon\}$

$\emptyset$ is a regexp representing $\emptyset$

If R_1 and R_2 are regular expressions representing L_1 and L_2 then:

(R_1R_2) represents $L_1 \cdot L_2$

$(R_1 \cup R_2)$ represents $L_1 \cup L_2$

$(R_1)^*$ represents L_1^*

PRECEDENCE

$* \quad \cdot \quad \cup$

EXAMPLE

$$R_1{}^*R_2 \cup R_3 = ((R_1{}^*)R_2) \cup R_3$$

{ w | w has exactly a single 1 }

0^*10^*

What language does $\varnothing^*$ represent?

What language does $\varnothing^*$ represent?

$\{\varepsilon\}$

{ w | w has length ≥ 3 and its 3rd symbol is 0 }

{ w | w has length ≥ 3 and its 3rd symbol is 0 }

$$(0 \cup 1)(0 \cup 1)0(0 \cup 1)^*$$

$\{ w \mid \text{every odd position of } w \text{ is a } 1 \}$

$\{ w \mid \text{every odd position of } w \text{ is a } 1 \}$

$$(1(0 \cup 1))^*(1 \cup \boldsymbol{\varepsilon})$$

EQUIVALENCE

L can be represented by a regexp
⇔ L is regular

1. **L can be represented by a regexp**
   ⇒ L is regular

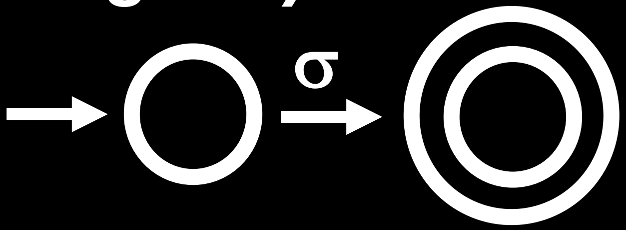
2. **L can be represented by a regexp**
   ⇐
   L is a regular language

1. **Given regular expression R, we show there exists NFA N such that R represents L(N)**

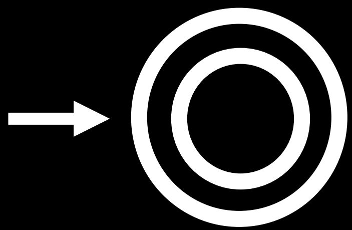
Induction on the *length* of R:

Base Cases (R has length 1):

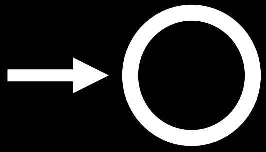
R = σ



R = ε



R = ∅



Inductive Step:

Assume R has length k > 1,
and that every regular expression of length < k represents a regular language

Three possibilities for R:

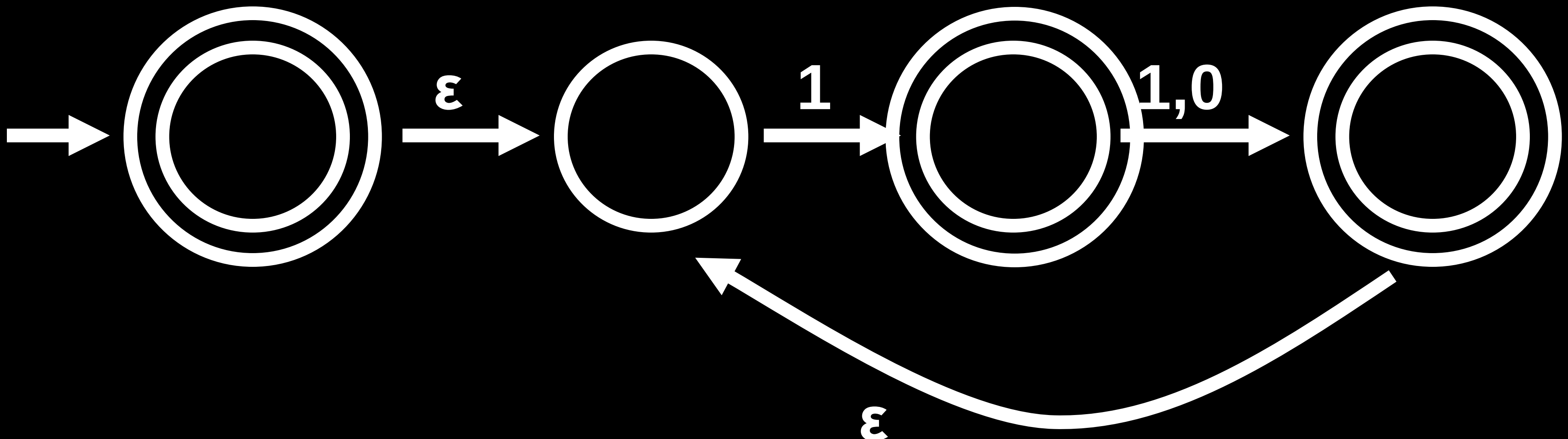
$R = R_1 \cup R_2$ (Union Theorem!)

$R = R_1 R_2$ (Concatenation)

$R = (R_1)^*$ (Star)

Therefore: L can be represented by a regexp
$\Rightarrow$ L is regular

Give an NFA that accepts the language represented by $(1(0 \cup 1))^*$

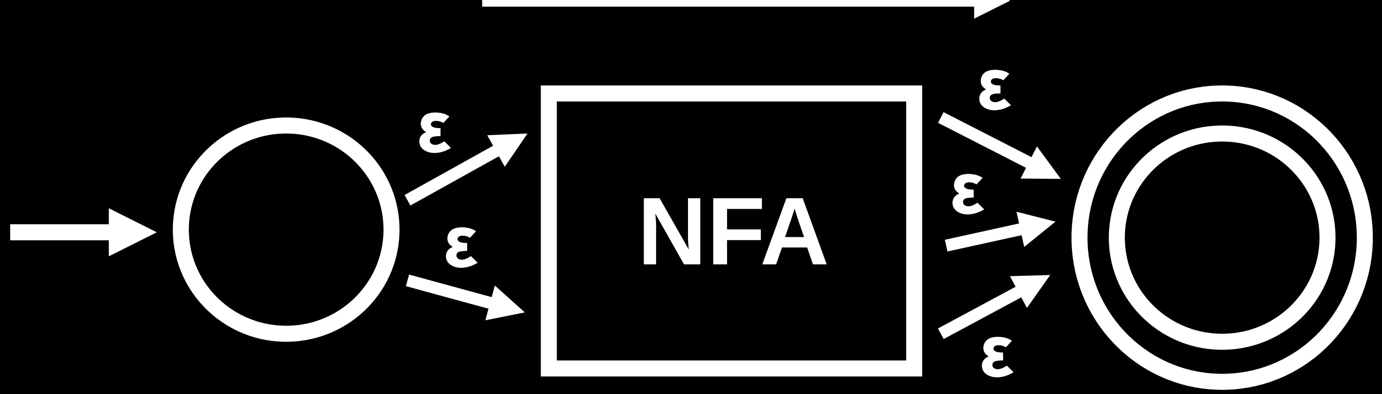


2. **L can be represented by a regexp**

$\Leftarrow$

L is a regular language

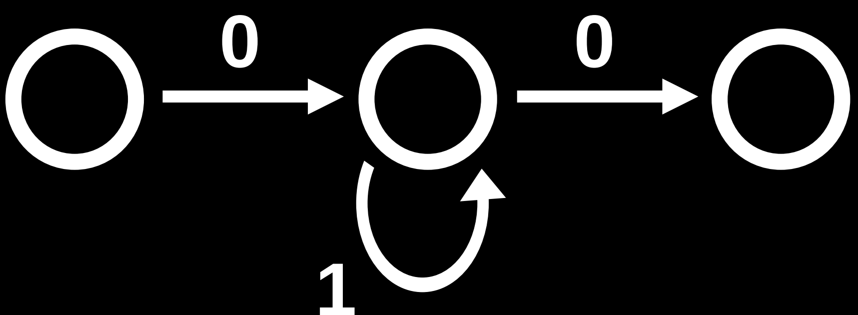
Proof idea: **Transform an NFA for L into a regular expression by removing states and re-labeling arrows with regular expressions**

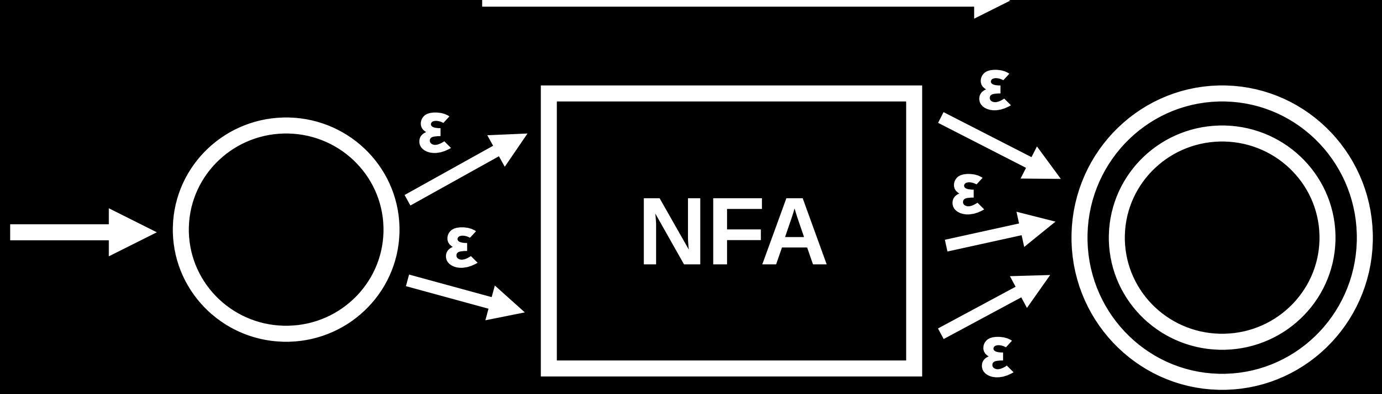


Add unique and distinct start and accept states

While machine has more than 2 states:

Pick an internal state, **rip it out and re-label the arrows with regexps**, to account for the missing state





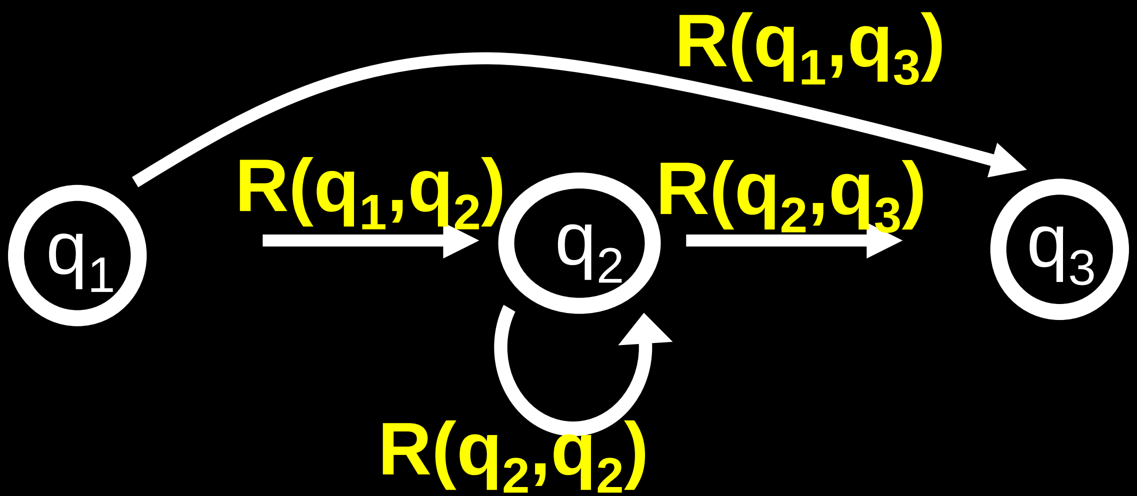
Add unique and distinct start and accept states

While machine has more than 2 states:

Pick an internal state, rip it out and re-label the arrows with regexps, to account for the missing state

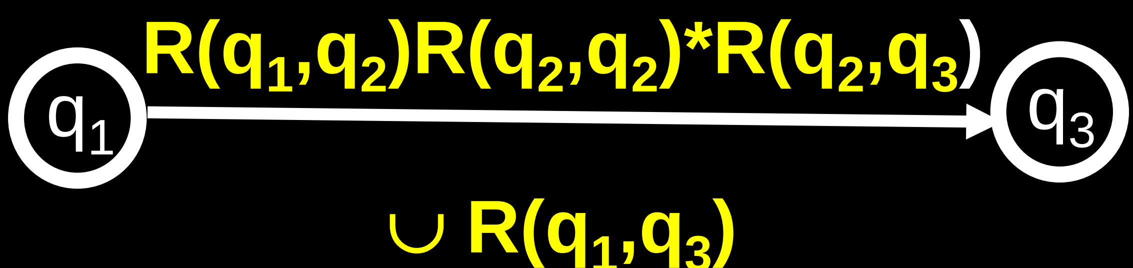
While machine has more than 2 states:

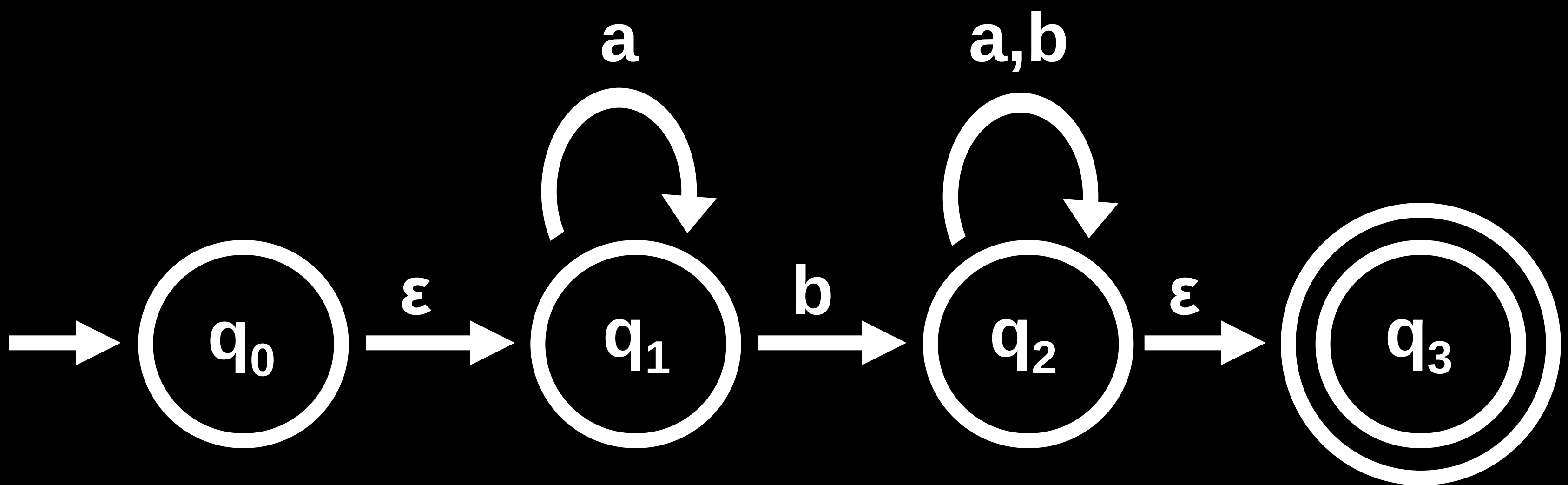
More generally:



While machine has more than 2 states:

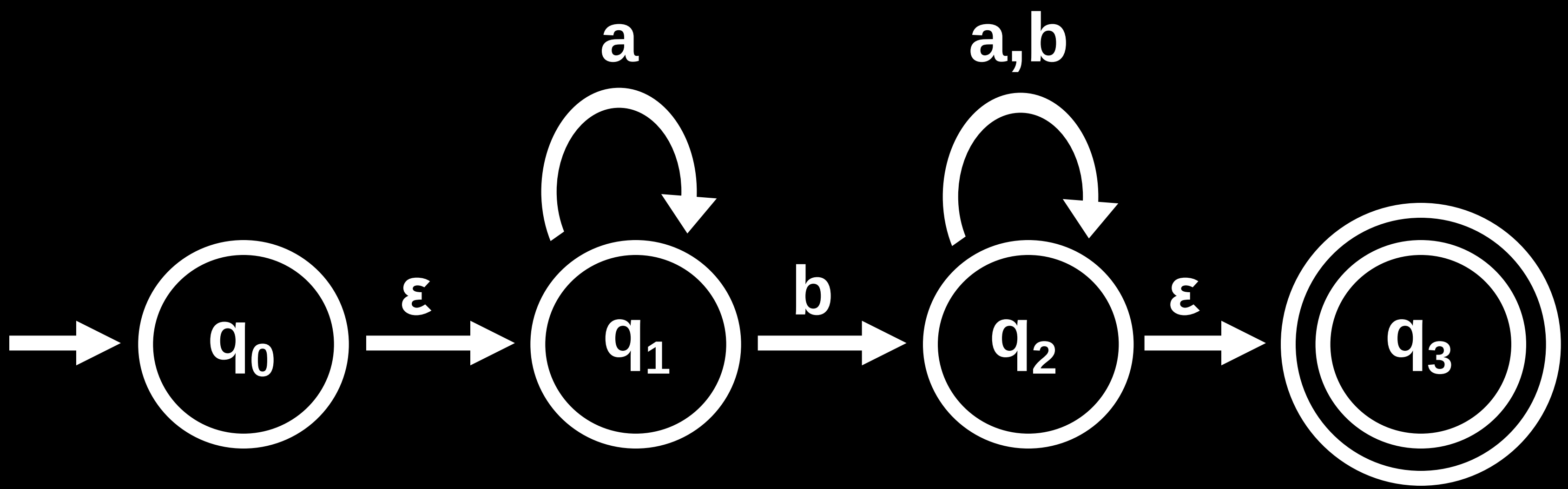
More generally:





$R(q_0,q_3)$ =

represents L(N)



$R(q_0,q_3) = (a^*b)(a \cup b)^*$

represents L(N)

Formally: Add q_{start} and q_{accept} to create G (GNFA)

Run CONVERT(G): (Outputs a regexp)

If #states = 2

return the expression on the arrow going from q_{start} to q_{accept}

Formally: Add q_{start} and q_{accept} to create G (GNFA)

Run CONVERT(G): (Outputs a regexp)

If #states > 2

select $q_{rip} \in Q$ different from q_{start} and q_{accept}

define $Q' = Q - \{q_{rip}\}$

define R′ as:

} Defines: G′ (GNFA)

$$R'(q_i,q_j) = R(q_i,q_{rip})R(q_{rip},q_{rip})^*R(q_{rip},q_j) \cup R(q_i,q_j)$$

(R′ = the regexps for edges in G′)

We note that G and G′ are *equivalent*

return CONVERT(G′)

Claim: CONVERT(G) is *equivalent* to G

Proof by induction on k (number of states in G)

Base Case:

- ✓ **k = 2**

Inductive Step:

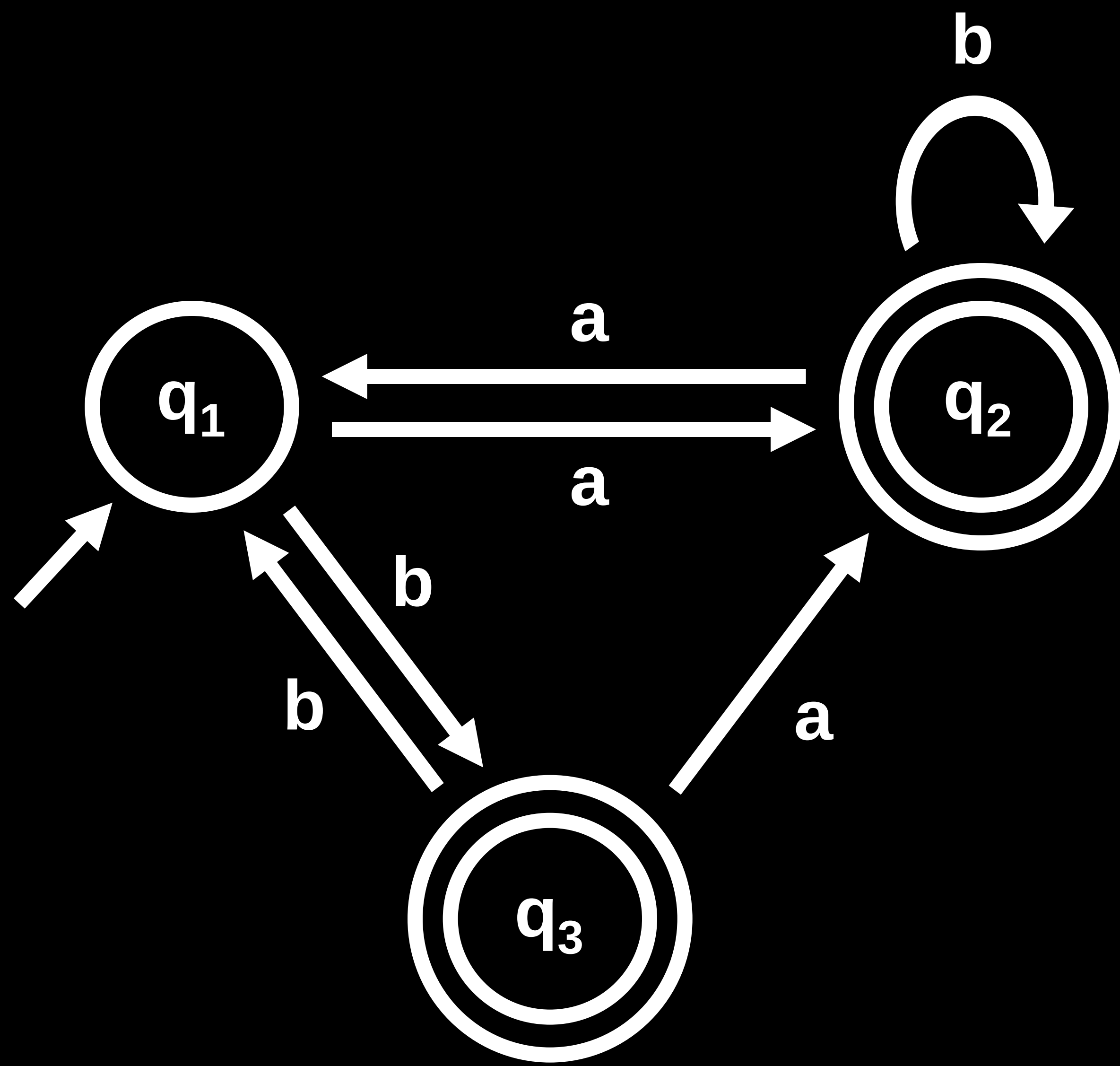
Assume claim is true for k-1 state GNFAs

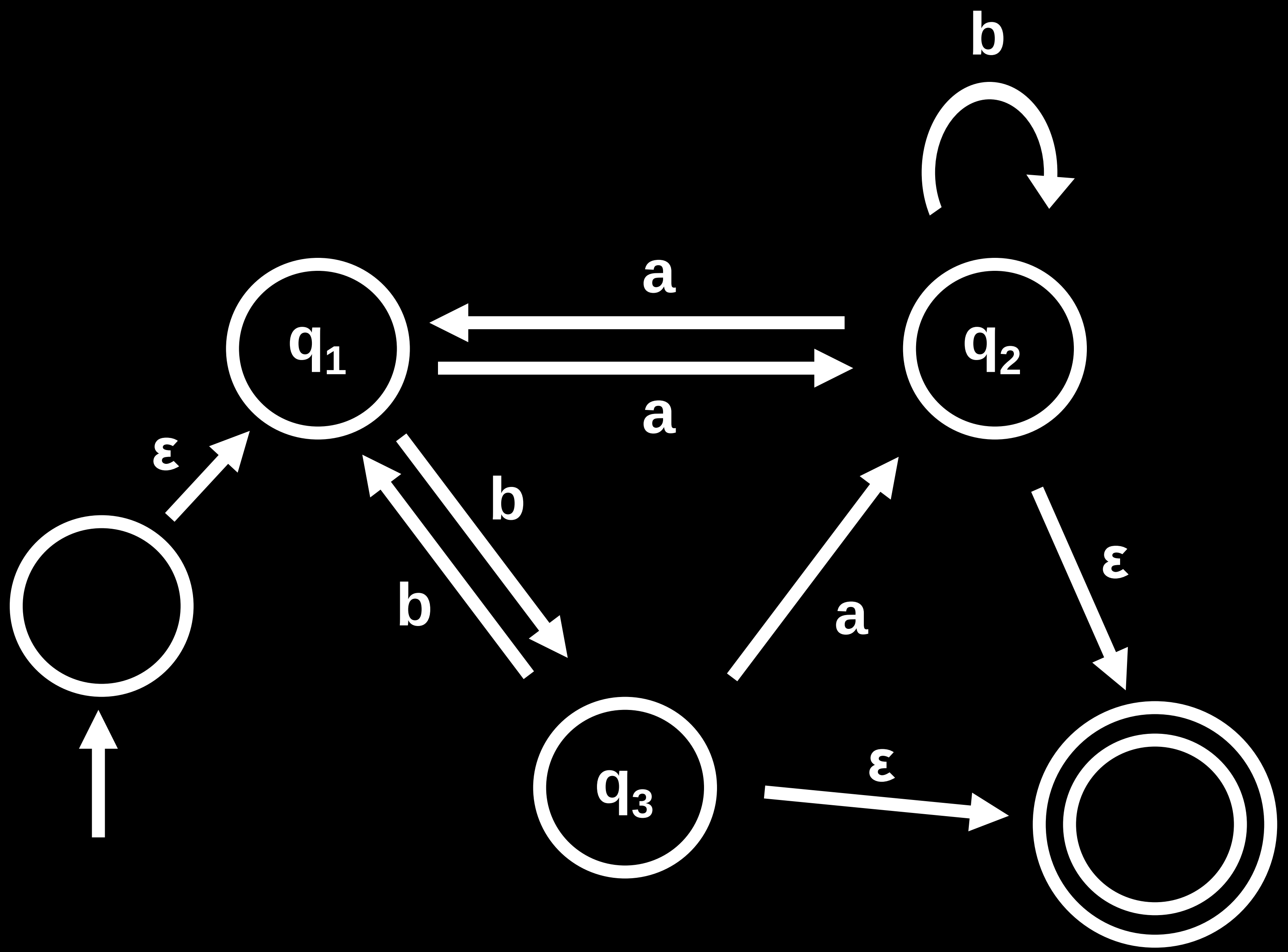
Recall that G and G′ are *equivalent*

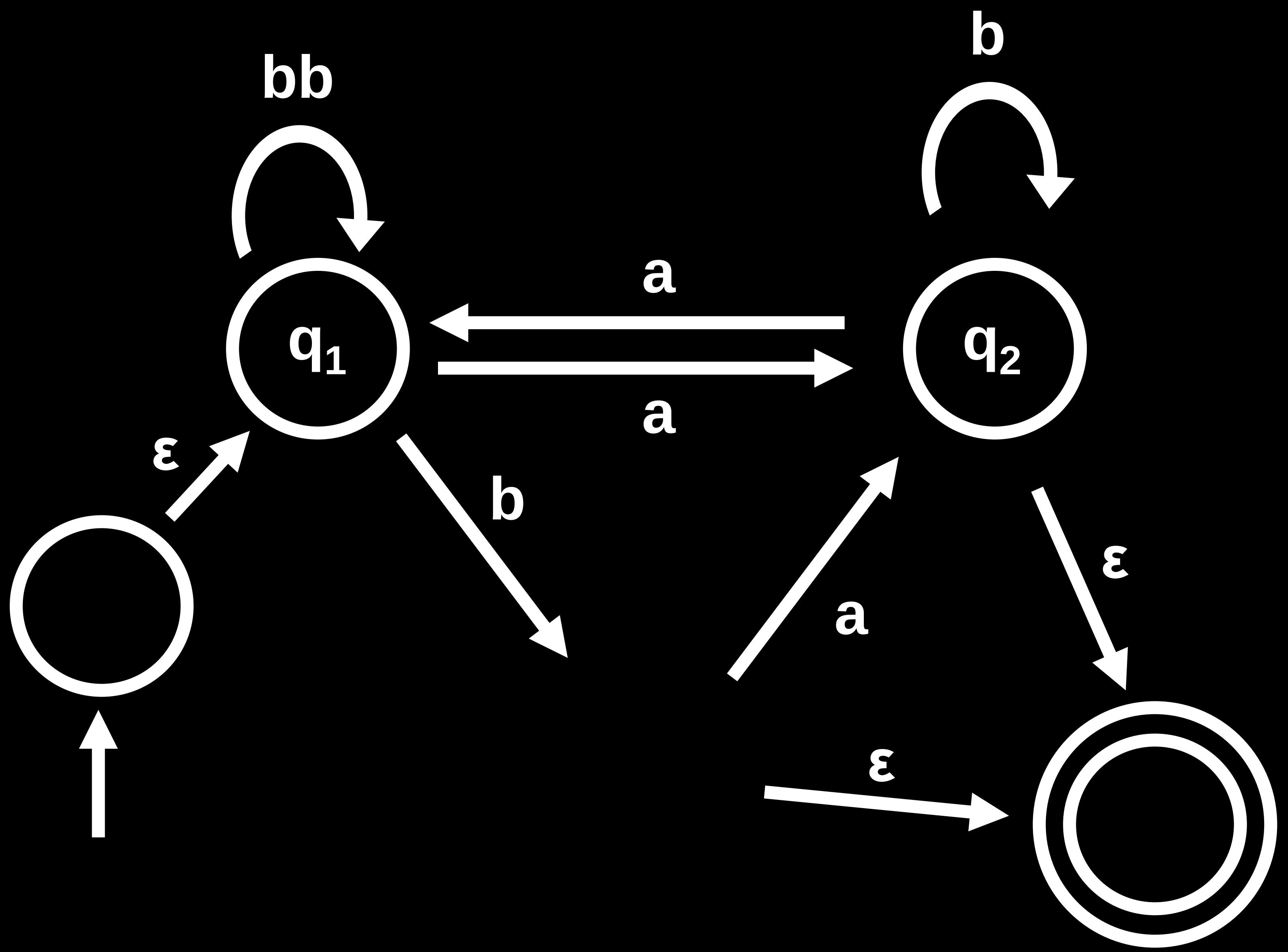
But, by the induction hypothesis, G′ is *equivalent* to CONVERT(G′)

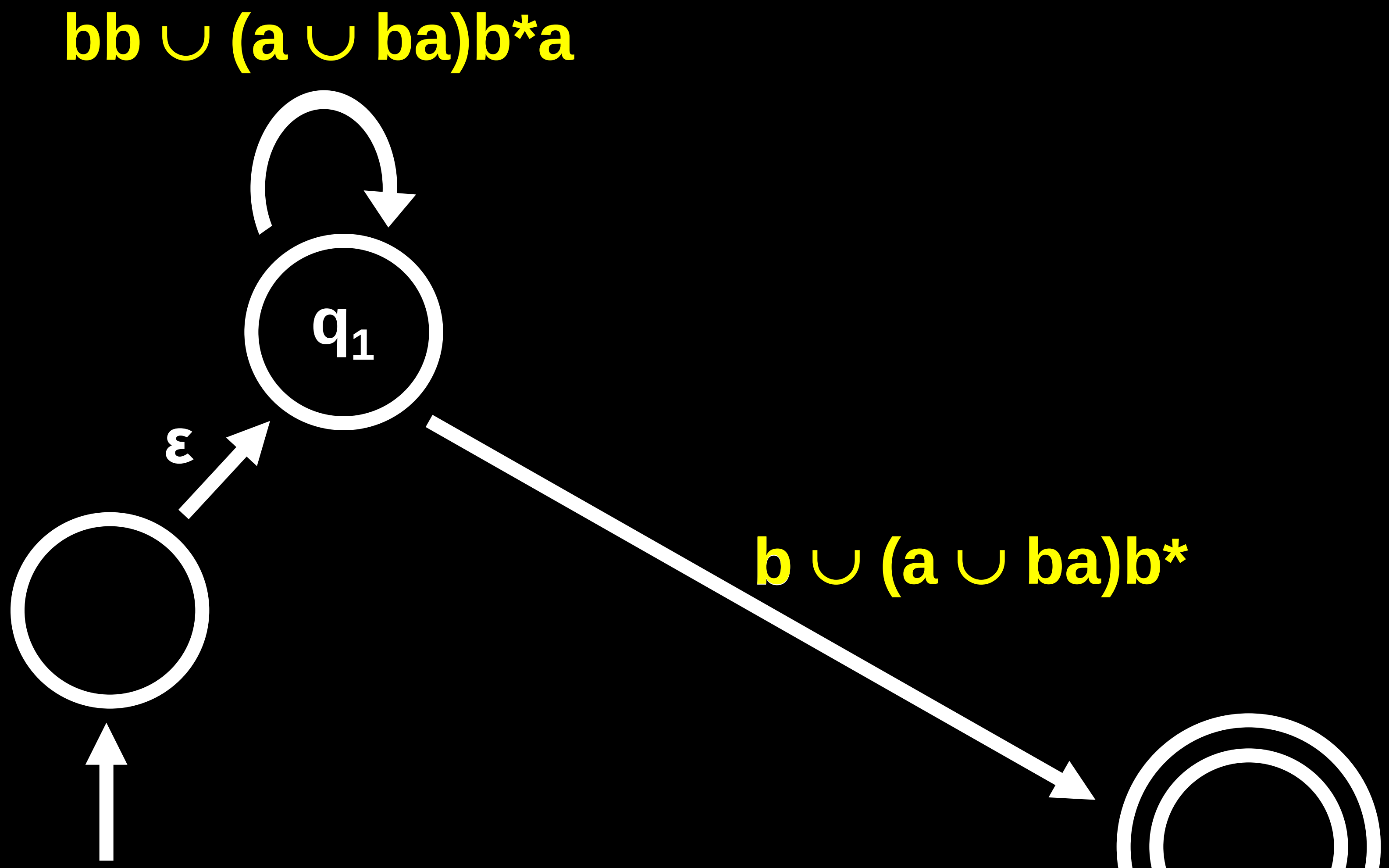
Thus: CONVERT(G′) *equivalent* to CONVERT(G)

QED


b
a
q1
q2
a
b
b
a
q3

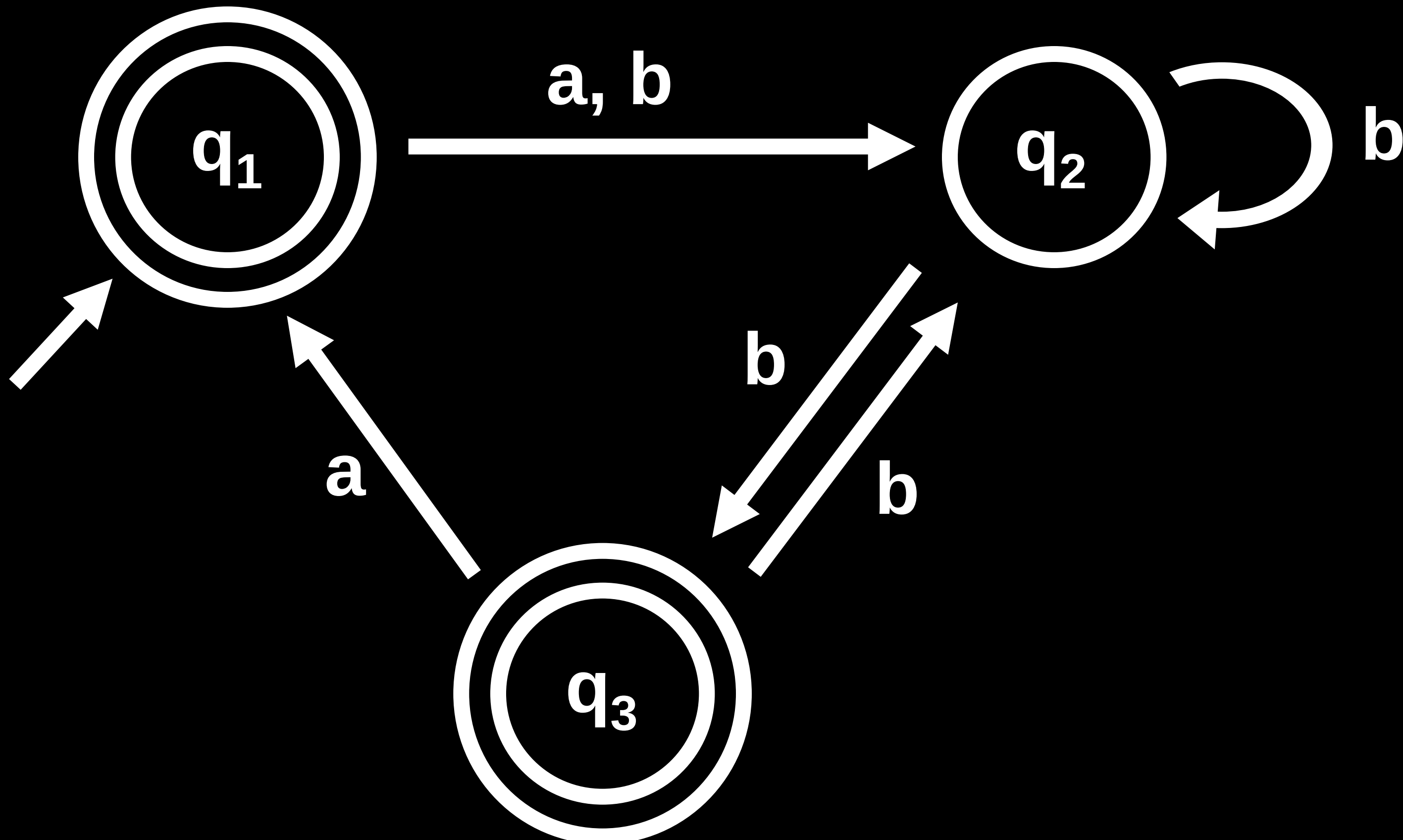

b
a
q1
q2
a
ε
b
b
ε
a
q3
ε

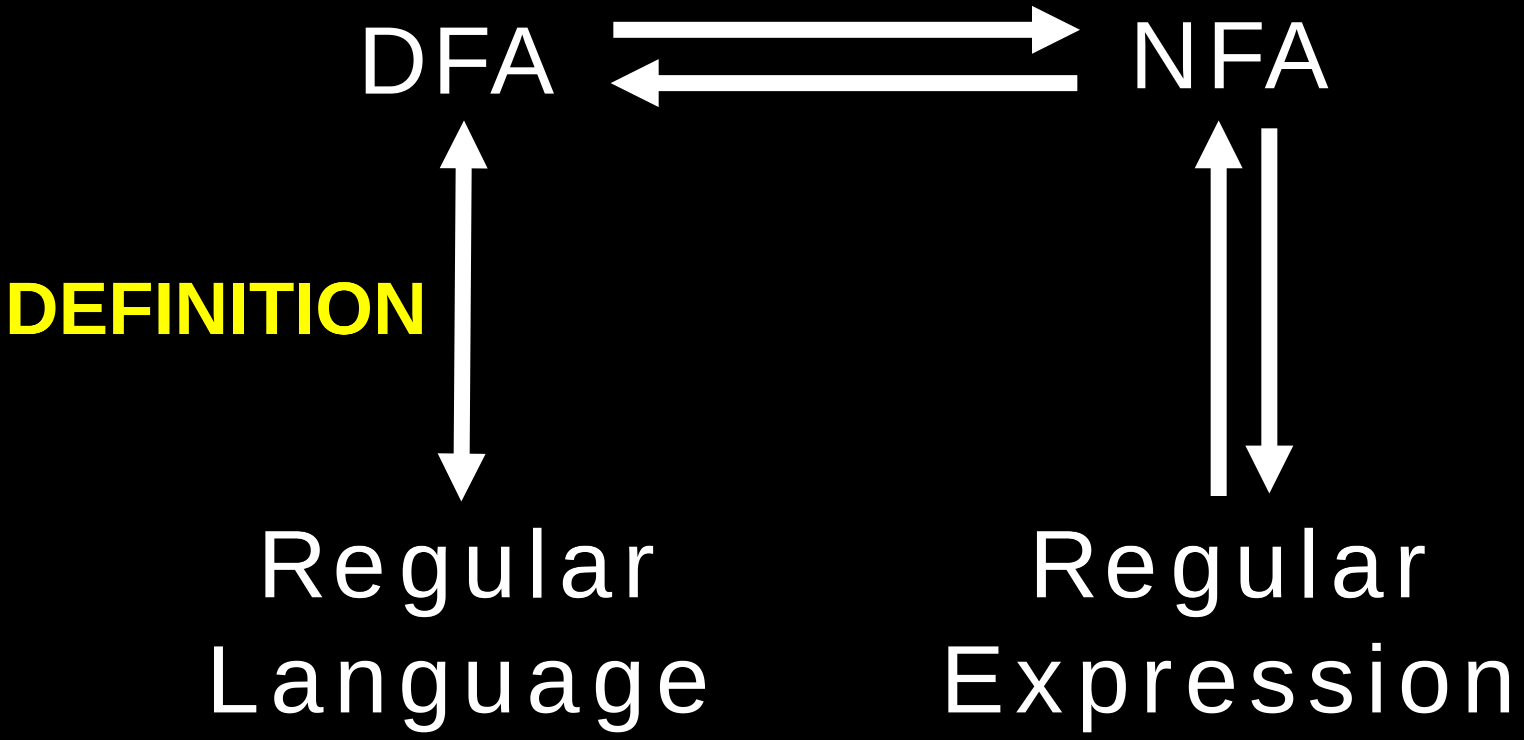

bb
b
a
q1
q2
a
ε
b
ε
a
ε



(bb ∪ (a ∪ ba)b*a)*(b ∪ (a ∪ ba)b*)

Convert the NFA to a regular expression




DFA
NFA
DEFINITION
Regular Language
Regular Expression

WWW.FLAC.WS

Finish Chapter 1 of the book for next time