# Theory of Computation

## ***Introduced By:***

## ***Prof.*** Safia Abbas

Text Books:

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

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

# Agenda

- Regular Expression Presentation
- Regular Expression equivalence
- Regular Expression and Regular Language
- Closure properties of Regular Expression
- Regular Expression and FSM

# Example

- Regular expression

$$r = (0+1)^* 00 (0+1)^*$$

$L(r)$ = { all strings with at least two consecutive 0 }

# Example

$L$ = {all strings with no two consecutive 0 }

$$r_1 = (1+01)^*(0+\lambda)$$

$$r_2 = (1^*011^*)^*(0+\lambda)+1^*(0+\lambda)$$

$L(r_1) = L(r_2) = L$ ⟹ $r_1$ and $r_2$ are equivalent regular expr.

# Equivalent Regular Expressions

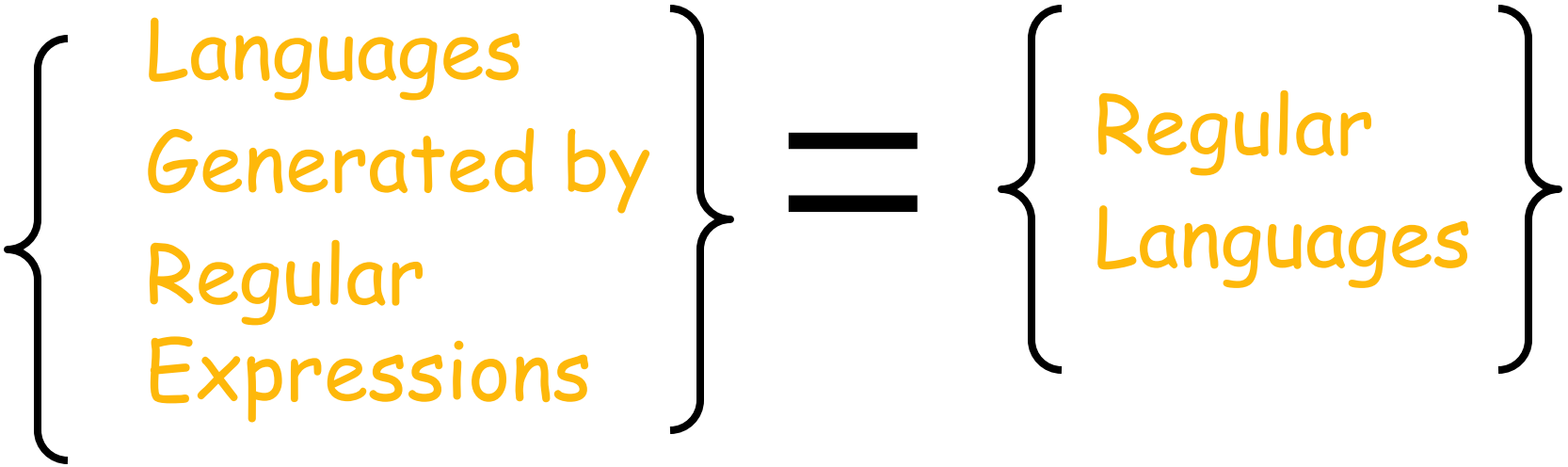
- Definition:

Regular expressions $r_1$ and $r_2$

are **equivalent** if

$$L(r_1) = L(r_2)$$

# Regular Expression & Regular Language



# Definition

For primitive regular expressions:

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

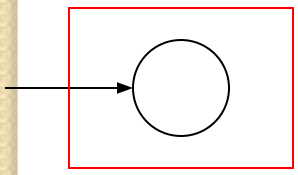
$$L(\lambda) = \{\lambda\}$$

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

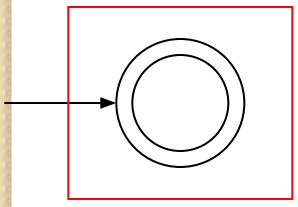
# Regular Expression and FA

- Primitive Regular Expressions:

$$\emptyset, \quad \lambda, \quad \alpha$$

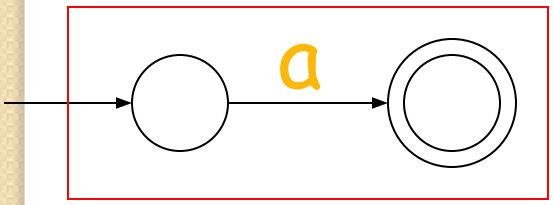


$$L(M_1) = \emptyset = L(\emptyset)$$



$$L(M_2) = \{\lambda\} = L(\lambda)$$

regular languages




$$L(M_3) = \{a\} = L(a)$$

# Closure properties or R.E.

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

Given regular expressions $r_1$ and $r_2$

$r_1 + r_2$

$r_1 \cdot r_2$

$r_1 *$

$(r_1)$

Are regular expressions!!!

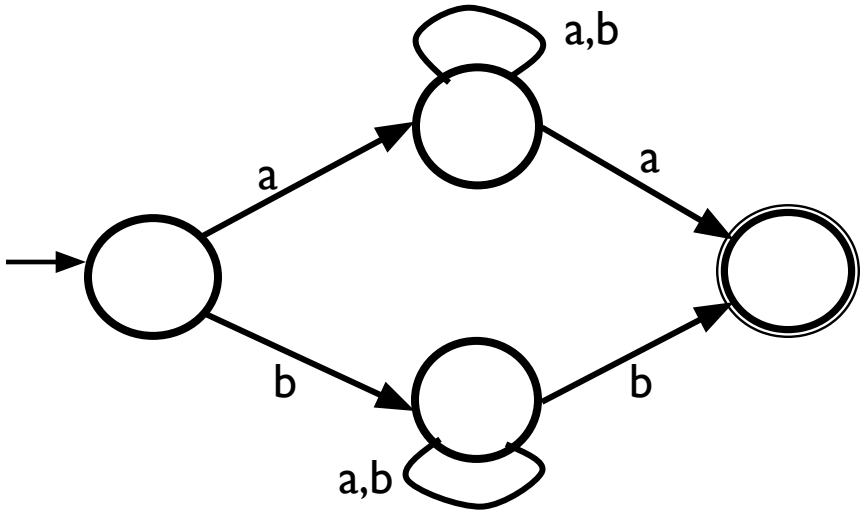
# Regular Expression and FA

Every language that can be defined by a regular expression can also be defined by a finite automaton.

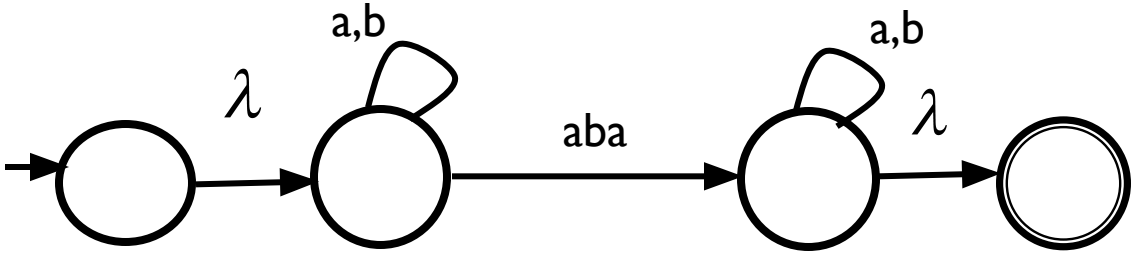
So we will see how to convert Regular expression to a FA, then how to convert FA to Regular expression.

# Example

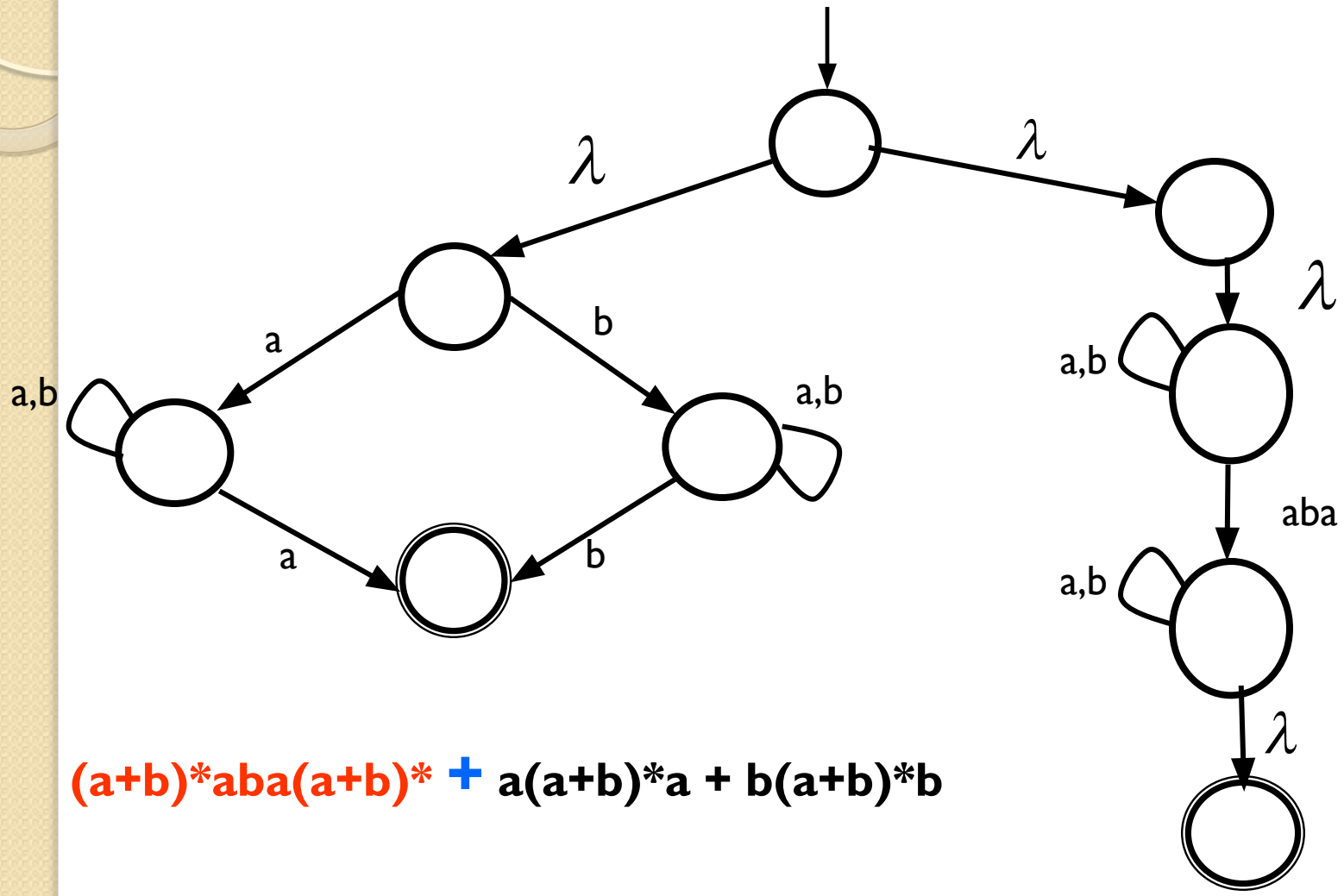
Words that begin and end with the same letter. a(a+b)*a + b(a+b)*b




Words that contain aba. (a+b)*aba(a+b)*

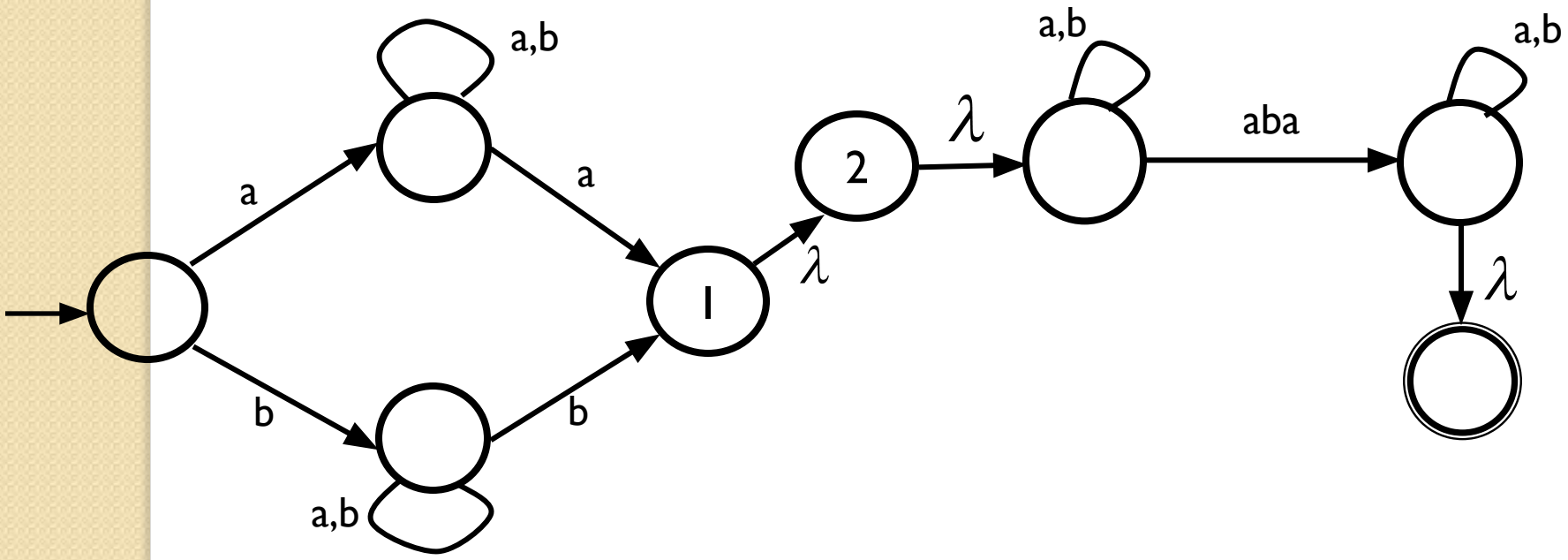


# Example



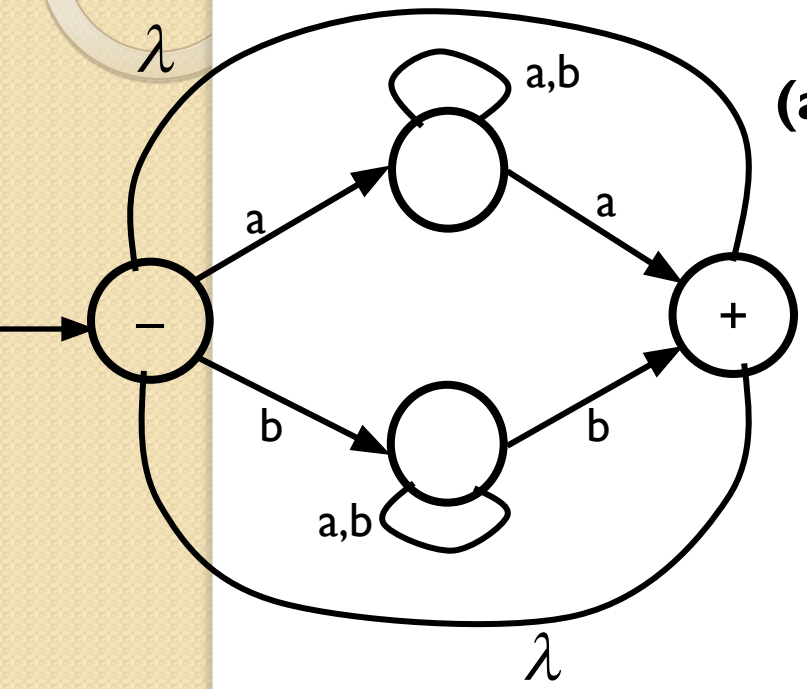

**(a+b)*aba(a+b)* + a(a+b)*a + b(a+b)*b**

# Example

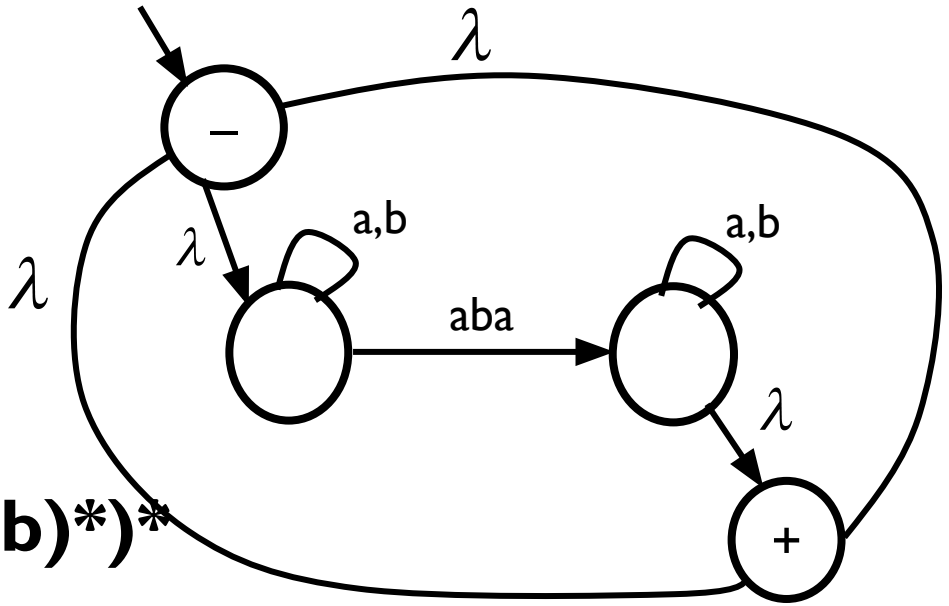



**(a(a+b)*a + b(a+b)*b)((a+b)*aba(a+b)*)**

# Example

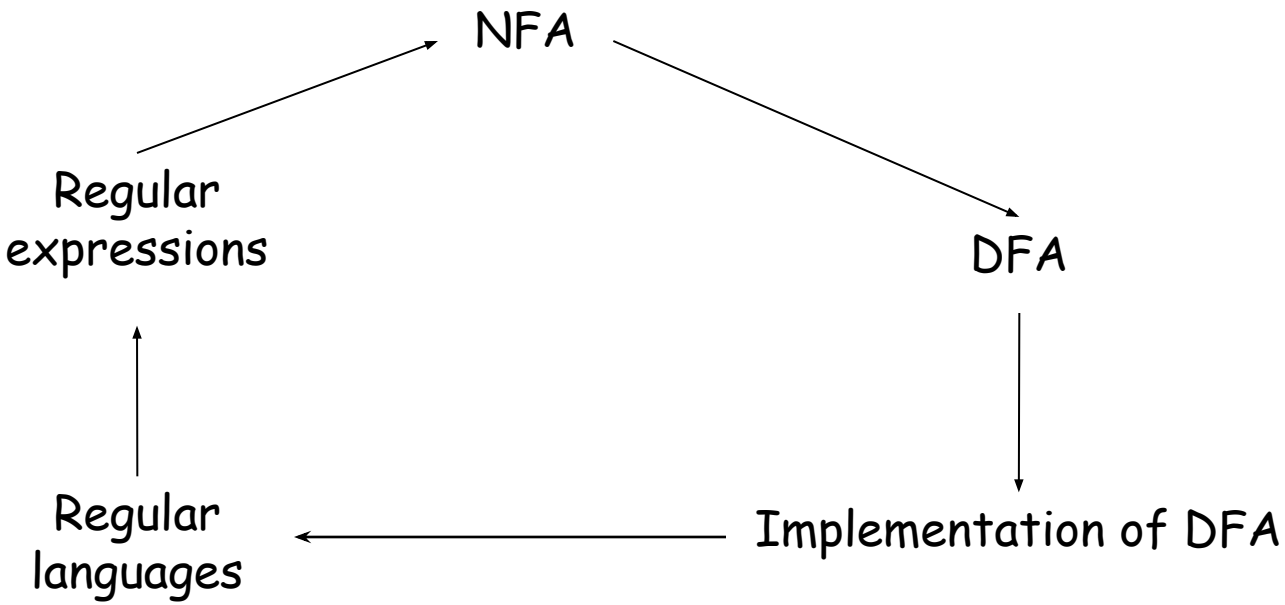



**(a(a+b)*a + b(a+b)*b)***




**((a+b)*aba(a+b)*)***

# Regular Expressions to Finite Automata



# Question

- Describe the language of the following regular expressions, and construct the NFA for r
- r = b* + ab + c*b
- r = (a*c). (ad)

# Answer for the language only:

- For r = b* + ab + c*b
- the answer is
  - L={L1 ∪ L2 ∪ L3} L1={$b^n$: n>=0} L2={ab} L3={$c^n$b : n>=0}.

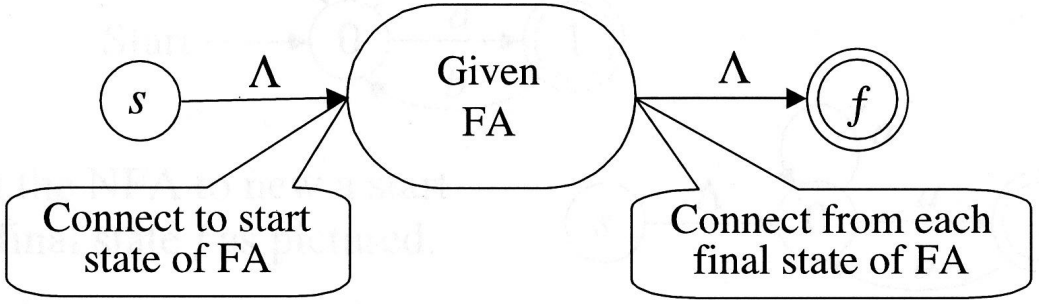
- For r = (a*c). (ad)
- the answer is
  - L={L1.L2} L1={$a^n$c : n>=0} L2={ad}

# Regular Expression and FA

Every language that can be defined by a regular expression can also be defined by a finite automaton.

So we were able to see how to convert Regular expression to a FA, then we will see how to convert FA to Regular expression.

# Algorithm

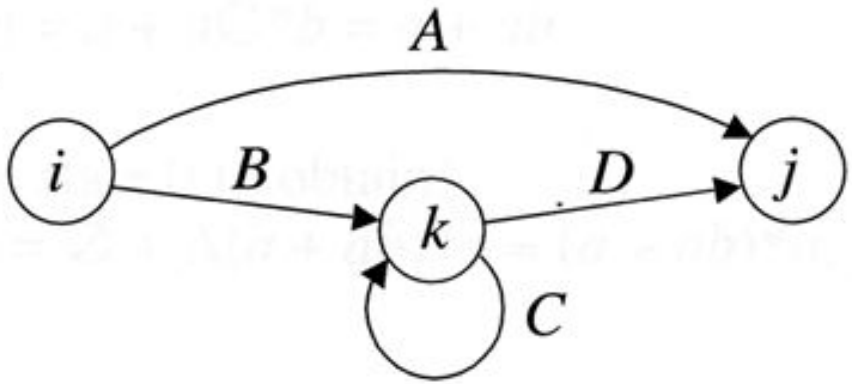



- Algorithm: Transform a Finite Automaton into a Regular Expression
- Connect a new start state s to the start state of the FA and connect each final state of the FA to a new final state f as shown in the figure.
- If needed, combine all multiple edges between the same two nodes into one edge with label the sum of the labels on the multiple edges. If there is no edge between two states, assume there is an Φ-edge.
- Now eliminate each state k of the FA by constructing a new edge (i, j) for each pair of edges (i, k) and (k, j) where $i \neq k$ and $j \neq k$. The new label new(i, j) is defined in terms of the old labels by the formula

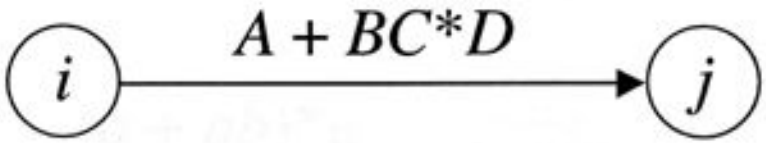
  new(i, j) = old(i, j) + old(i, k)old(k, k)*old(k, j)

# Example

*Example.*

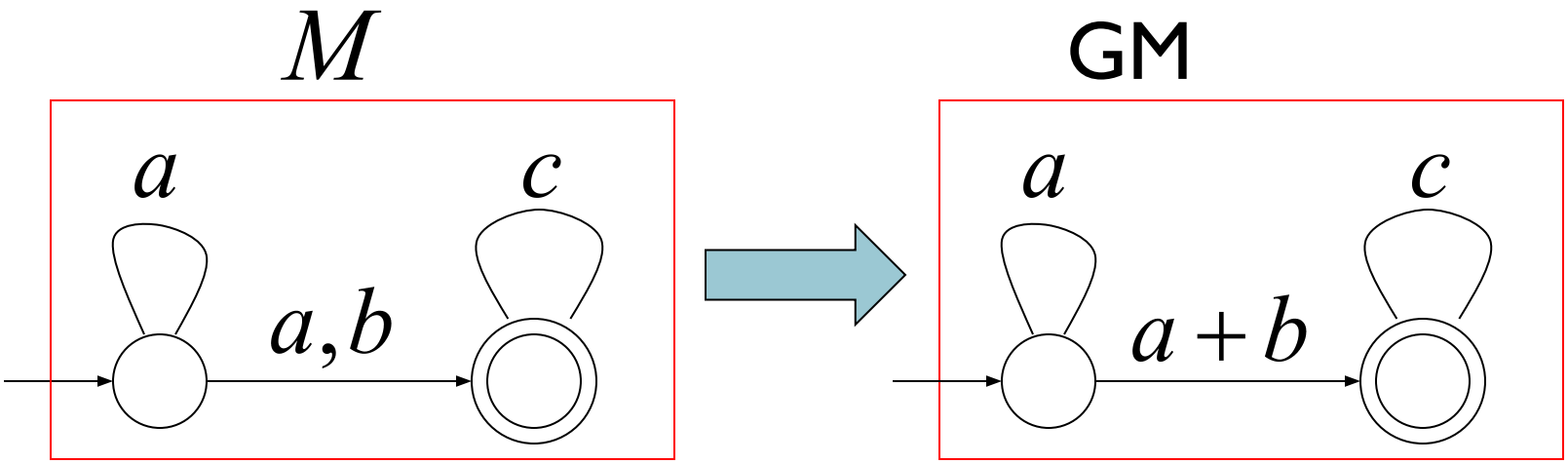



becomes



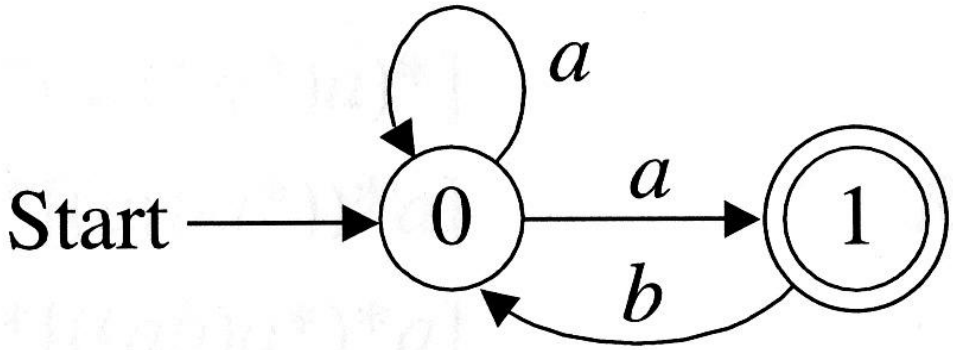
# Example:

- From $M$ construct the equivalent Generalized Transition Graph



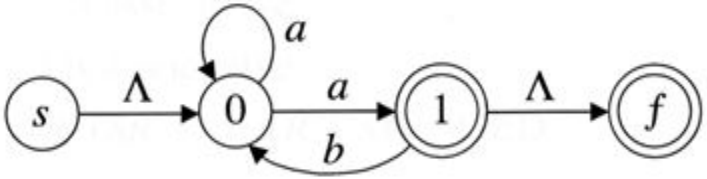
# Example

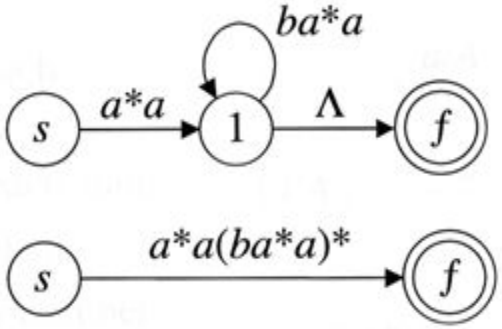
- Use the algorithm to transform the following NFA into a regular expression.



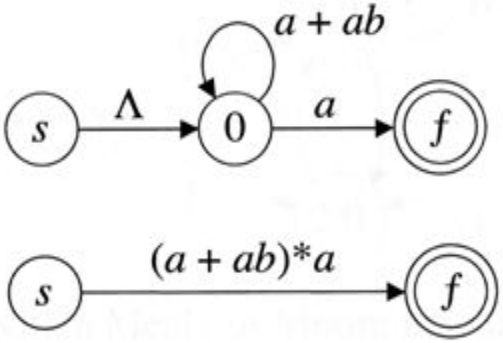
# Solutions

- First Solution: Eliminate state 0 to obtain:
  new(s, 1) = Φ + λa*a = a*a.
  new(1, 1) = Φ + ba*a = ba*a.
- Eliminate state 1 to obtain:
  new(s, f) = Φ + a*a(ba*a)* λ = a*a(ba*a)*.




- Second Solution: Eliminate state 1 to obtain:
  new(0, f) = Φ + aΦ* λ = a.
  new(0, 0) = a + aΦ*b = a + ab.
- Eliminate state 0 to obtain:
  new(s, f) = Φ + λ(a + ab)*a = (a + ab)*a.



# Example

- Use regular algebra to show the following equality from the previous example.

  a*a(ba*a)* = (a + ab)*a.

*Proof:* $a^*a(ba^*a)^* = a^*[a((ba^*)a)^*]$ · is associative

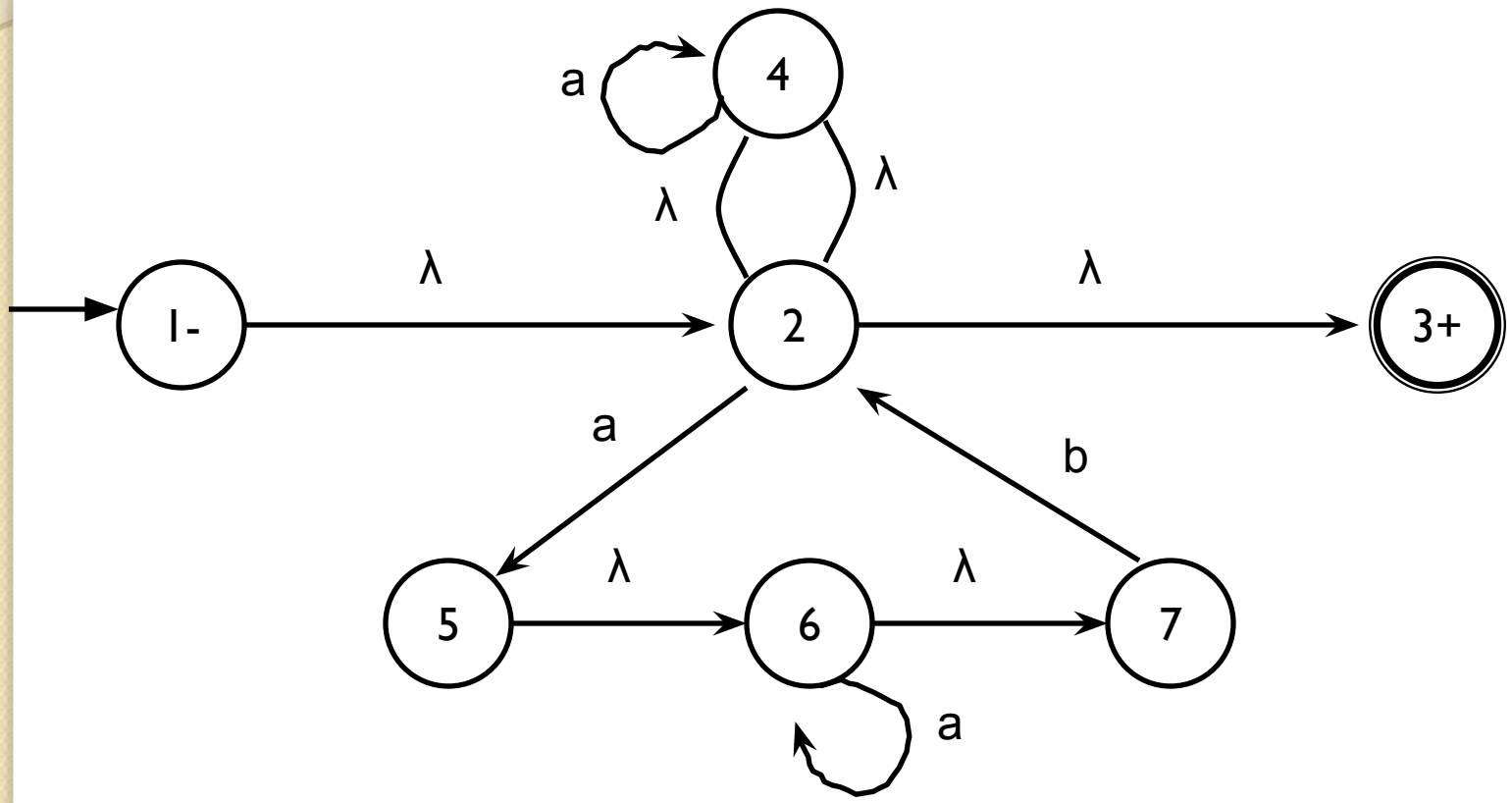
$= a^*[(a(ba^*))^*a]$ $R(SR)^* = (RS)^*R$

$= a^*[((ab)a^*)^*a]$ · is associative

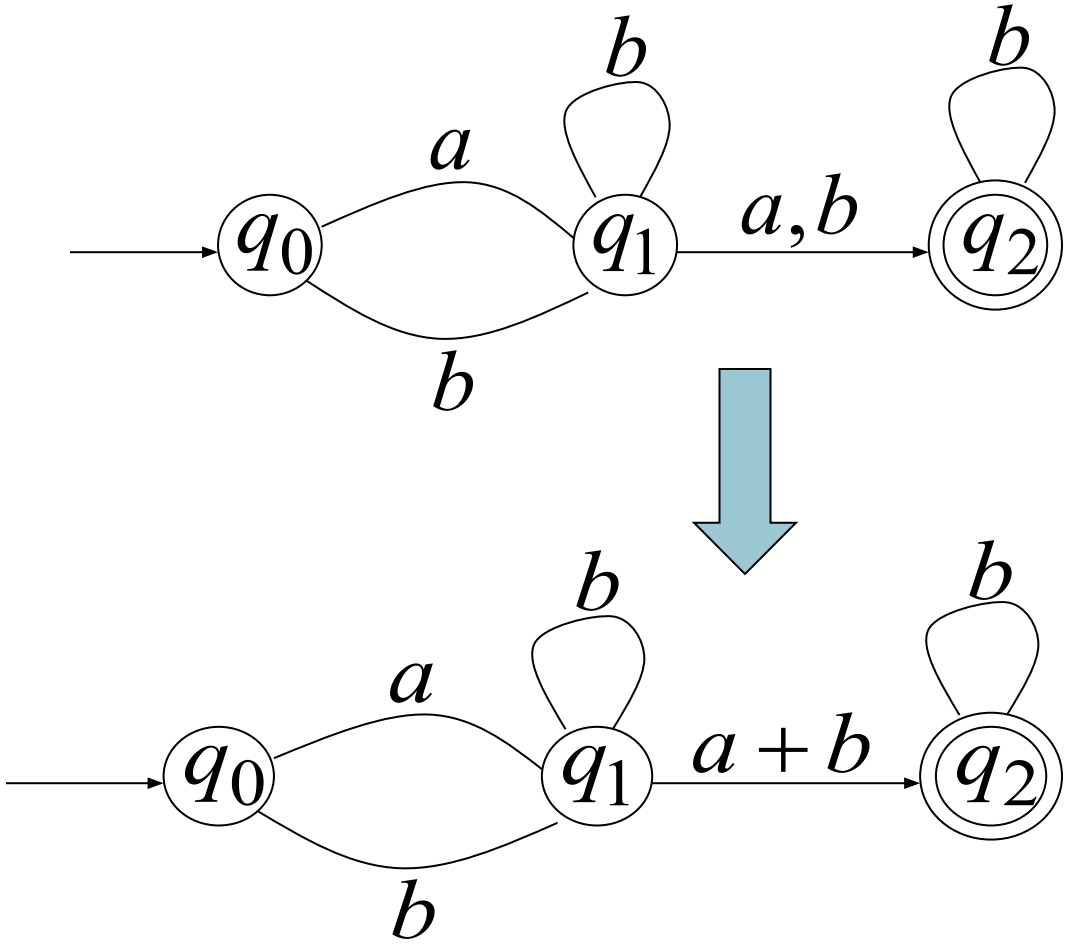
$= [a^*((ab)a^*)^*]a$ · is associative

$= (a + ab)^*a$ $R^*(SR^*)^* = (R + S)^*$. QED.

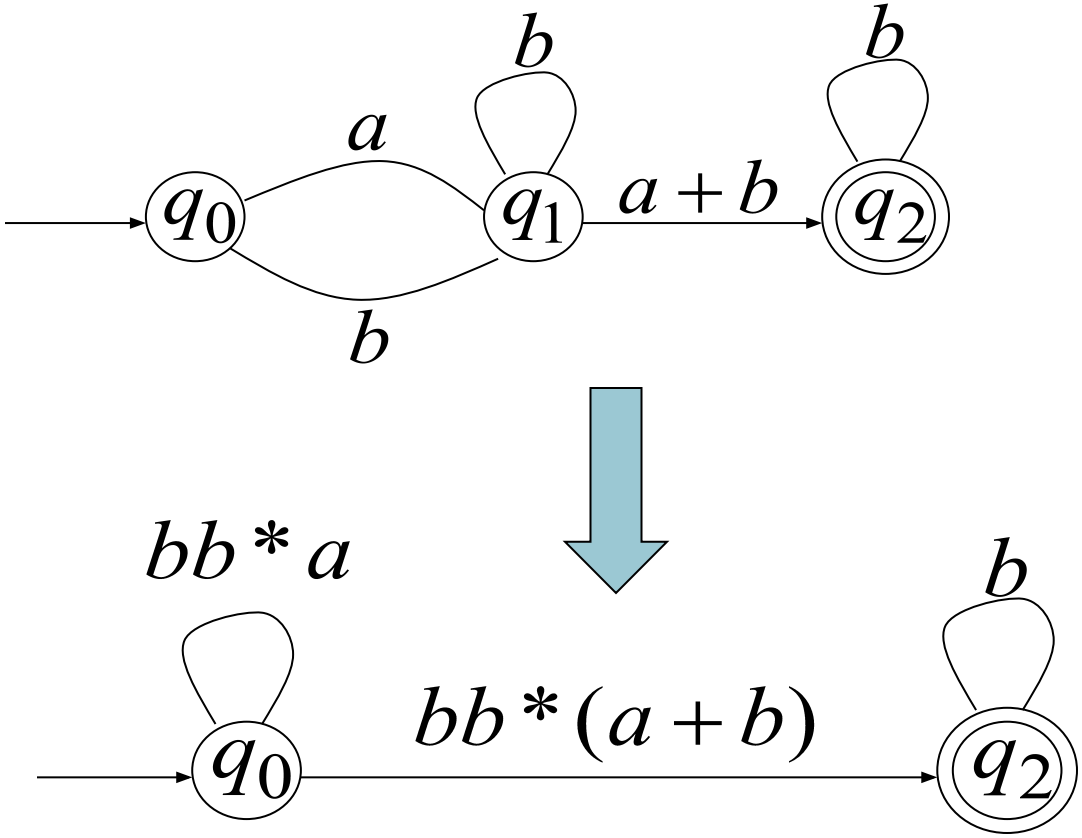
# (a* + aa*b)*



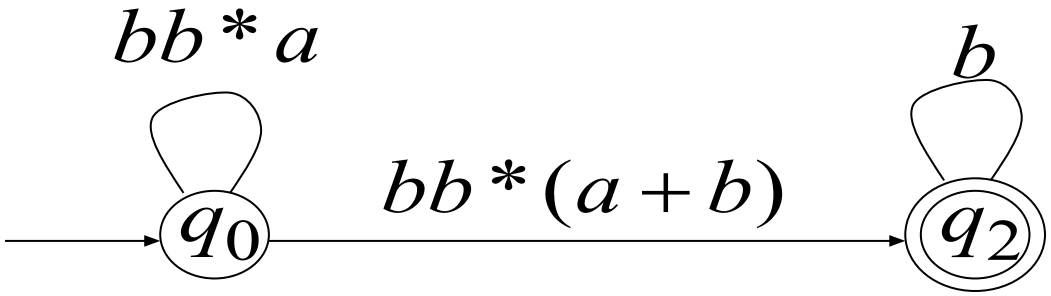
**Example:**



- Reducing the states:



- Resulting Regular Expression:

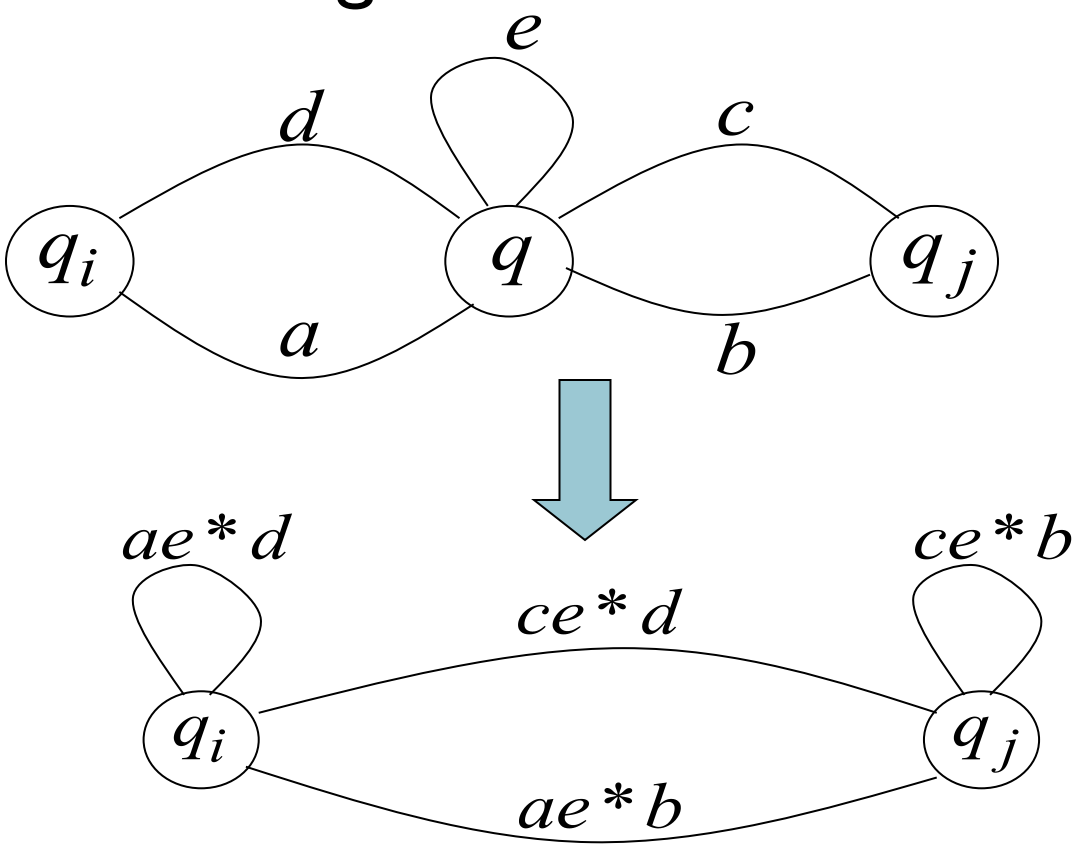



$$r = (bb^*a)^*bb^*(a+b)b^*$$

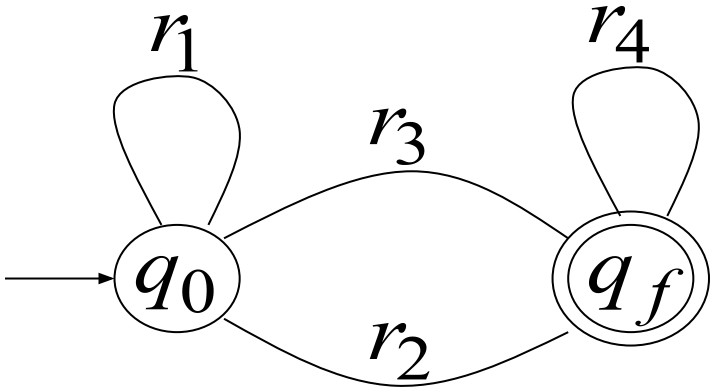
$$L(r) = L(M) = L$$

# In General

- Removing states:



- Obtaining the final regular expression:



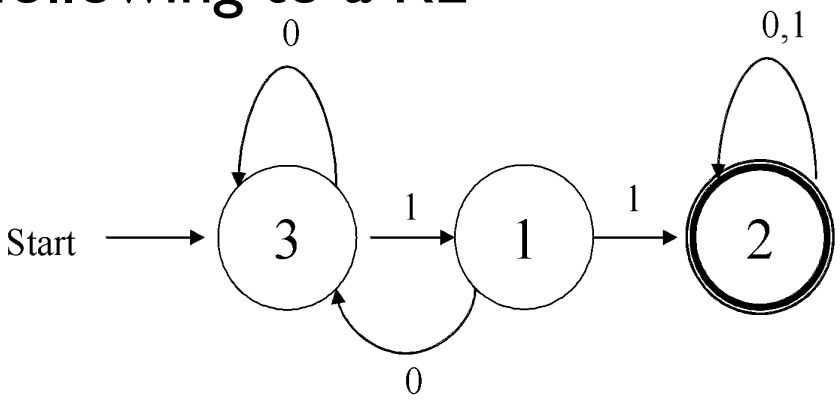

$$r = r_1^* r_2 (r_4 + r_3 r_1^* r_2)^*$$

$$L(r) = L(M) = L$$

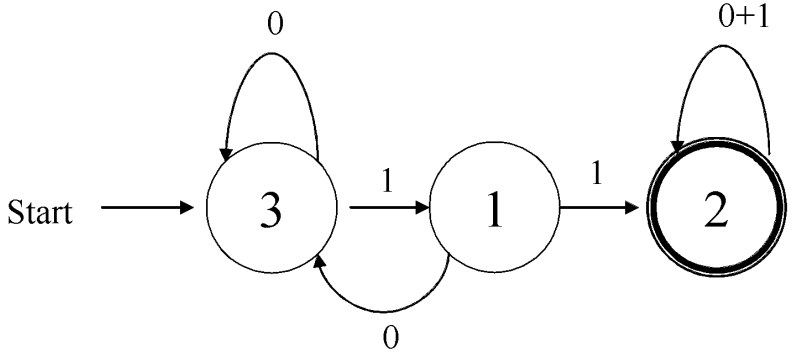
q0,qf

# DFA□RE Example

- Convert the following to a RE

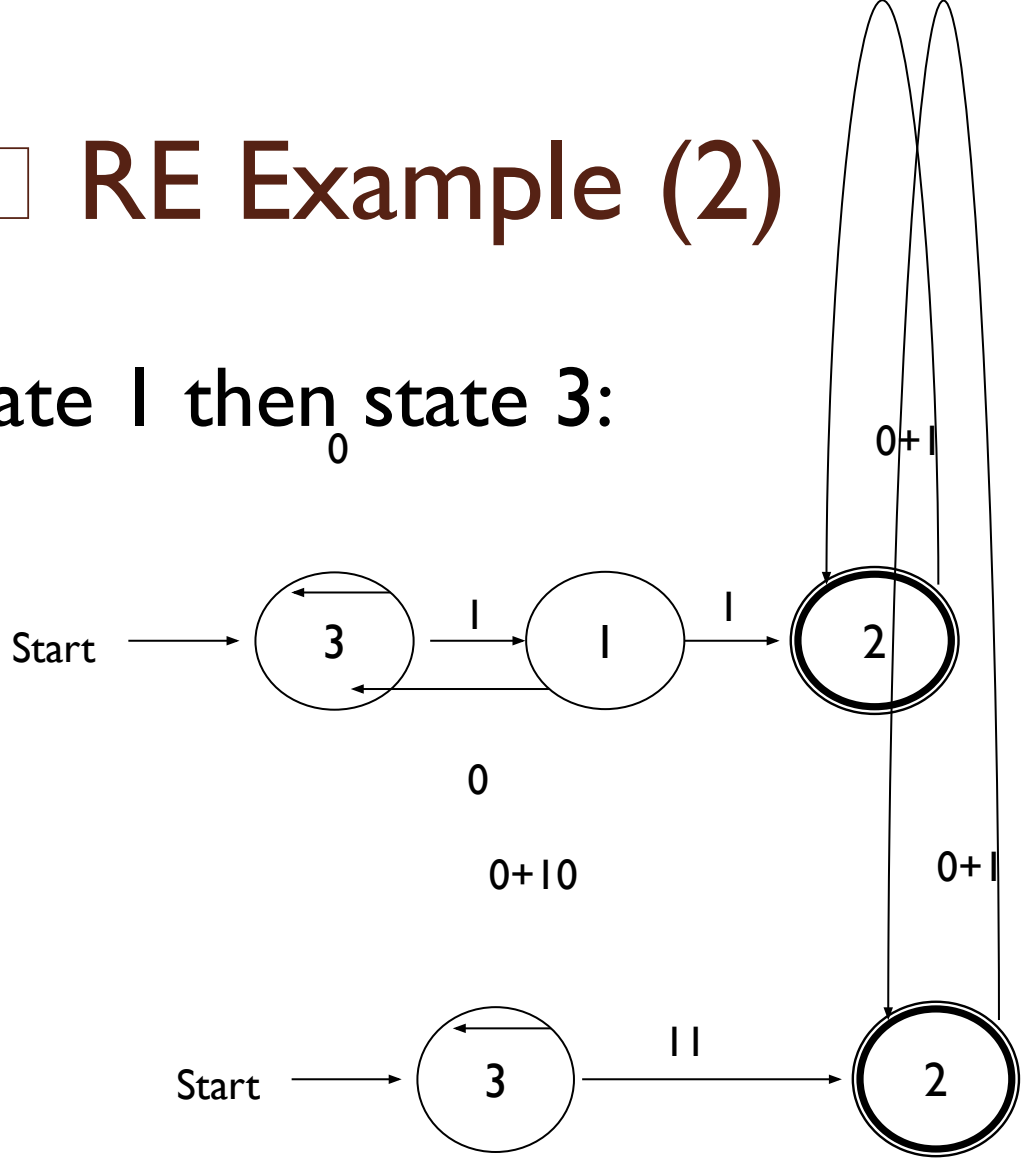



- First convert the edges to RE's:



# DFA  RE Example (2)

- Eliminate State 1 then state 3:



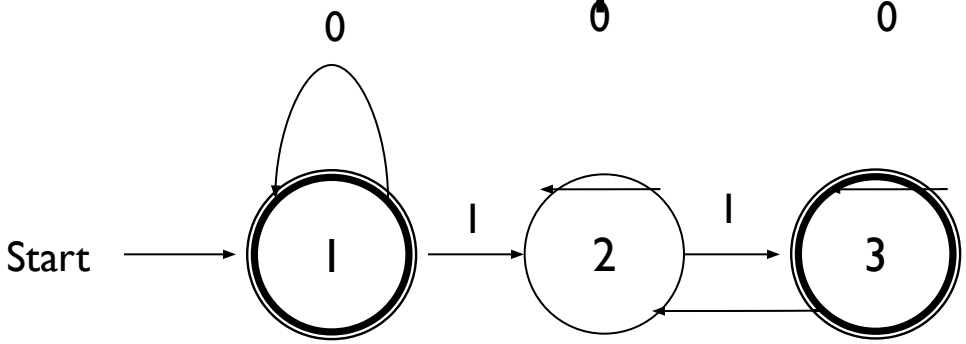

- To:



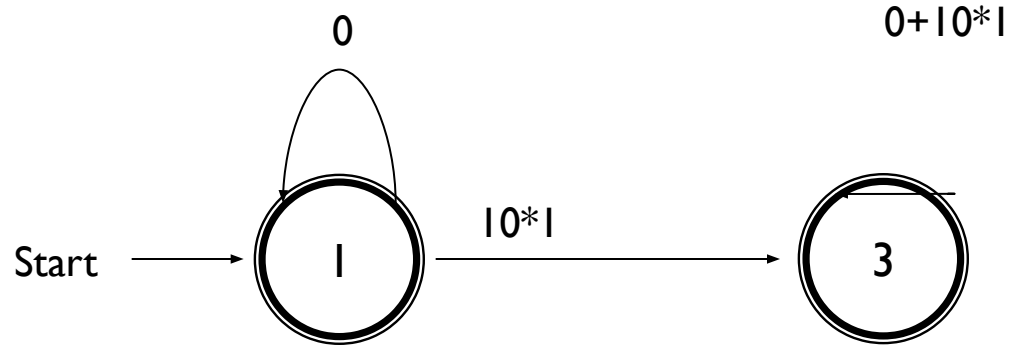

Answer: (0+10)*11(0+1)*

# Second Example

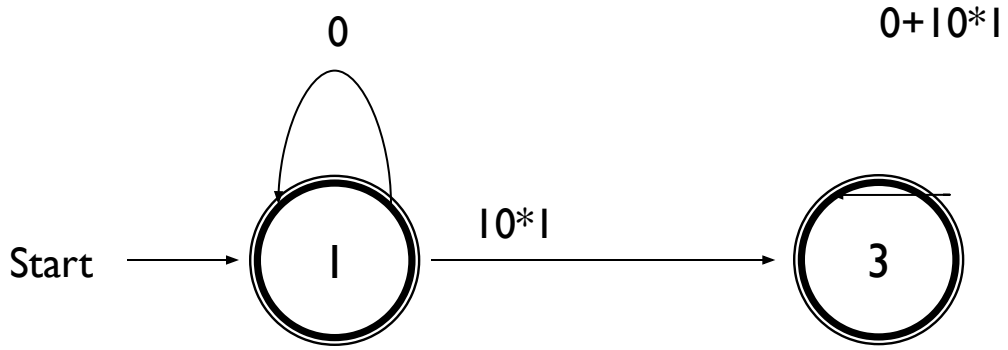
- Automata that accepts even number of 1's



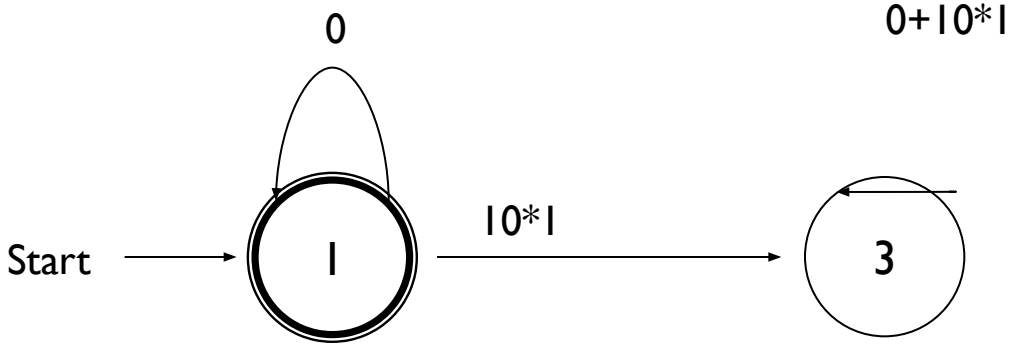

- Eliminate state 2:



# Second Example (2)

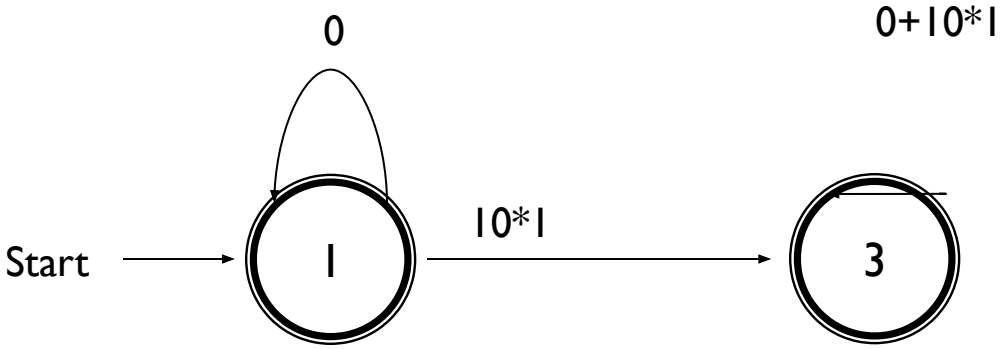



- Two accepting states, turn off state 3 first

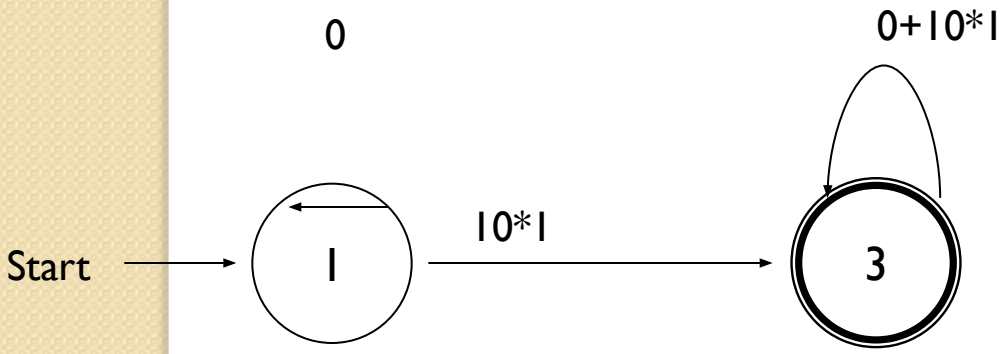



This is just 0*; can ignore going to state 3 since we would "die"

# Second Example (3)




- Turn off state 1 second:




This is just 0*10*1(0+10*1)*

Combine from previous slide to get
0* + 0*10*1(0+10*1)*

# Thanks for Listening