Lecture 5

Regular Expressions

COT 4420

Theory of Computation

Section 3.1, 3.2

Regular Expressions

- Regular Expressions describe regular languages.

Example:

$(a+b.c)^*$

describes the language:

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

Regular Expressions

- Regular expressions use three operations: Union (+), Concatenation (.) and Kleene Star(*).

1. Ø, λ, a ∈ Σ are all primitive regular expressions.
2. Given regular expressions r_1 and r_2:

r_1+r_2
$r_1.r_2$
r_1*
(r_1)

} Are regular expressions

Regular Expressions

3. A string is a regular expression iff it can be derived from primitive regular expressions and a finite number of applications of defined operators.

Languages Associated with Regular Expressions

$L(\emptyset)$ = empty set

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

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

If r_1 and r_2 are regular expressions:

$L(r_1+r_2) = L(r_1) \cup L(r_2)$

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

$L(r_1^*) = (L(r_1))^*$

Regular Expressions

- Union of languages

  Example: {01,111,10} $\cup$ {00,01} ={01, 111, 10, 00, 01}

- Concatenation of languages L_1 and L_2

  $L_1 . L_2 = \{(x.y) \mid x \in L_1 \text{ and } y \in L_2\}$

  Example: {01,111,10}{00, 01} = {0100, 0101, 11100, 11101, 1000, 1001}.

- Kleene Star L* is the set of strings formed by concatenating zero or more strings from L, in any order.

  $L^* = \{\lambda\} \cup L \cup LL \cup LLL \cup \dots$

  Example: $\{0,10\}^* = \{\lambda$, 0, 10, 00, 010, 100, 1010,...}

Regular Expressions

Example: (a+b).a*

$L((a+b).a^*) = L(a+b).L(a^*)$

$= (L(a) \cup L(b)).(L(a))^*$

$= (\{a\} \cup \{b\})(\{a\})^*$

$= \{a,b\}\{\lambda, a, aa, aaa, \ldots\}$

$= \{a, aa, aaa, \ldots, b, ba, baa, \ldots\}$

Precedence of Operators

- Parentheses can be used whenever needed.

- Oder of precedence from high to low is star * then concatenation . and then union +.

Regular Expression Examples

01	{01}
01 + 0	{01, 0}
0(1+0)	{01, 00}
0*	{λ, 0, 00, 000, ...}
(0+1)*	{λ, 0, 1, 00, 01, 10, 11, 111, 000, 010, ...}
(0+1)*(0+11)	Any string of 0's and 1's that ends with either a 0 or a 11.

Regular Expression Example

- Give a regular expression r such that

  $L(r) = \{ w \in \{0,1\}^*: w$ has at least one pair of consecutive 0s$\}$

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

Regular Expression Example

What is the language described by regular expression $r = (aa)^*(bb)^*b$

$$L(r) = \{ a^{2n}b^{2m}b : n,m \geq 0\}$$

Equivalence of Regular Expressions and Regular Languages

Languages generated by Regular Expressions $=$ Regular languages

Part 1) The set of languages generated by regular expressions is a subset of regular languages.

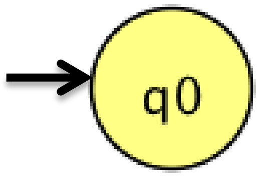
Equivalence of Regular Expressions and Regular Languages – Part 1

Theorem: Let r be a regular expression. There exists some NFA that accepts L(r).

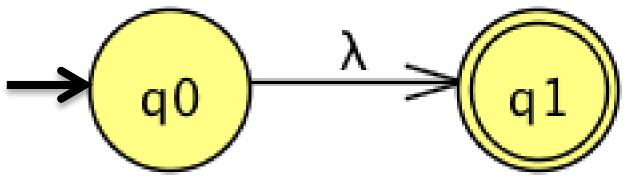
Proof is an induction on the number of operators (+, concatenation, *) in the regular expression.

Convert Regular Expression to NFA

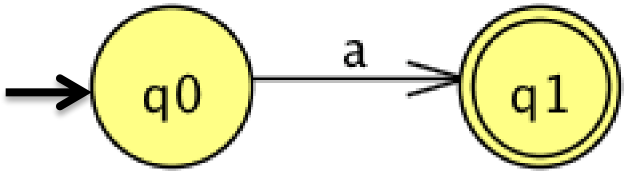
∅



λ

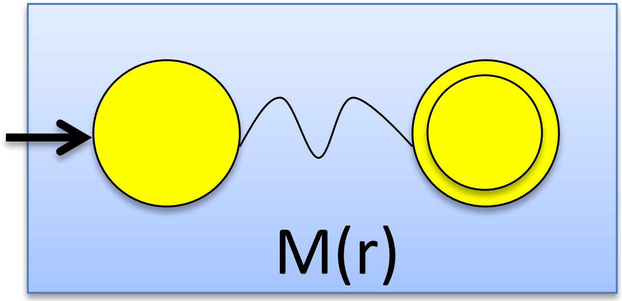


a



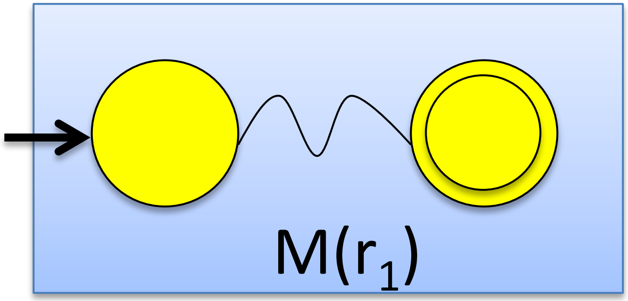
Convert Regular Expression to NFA

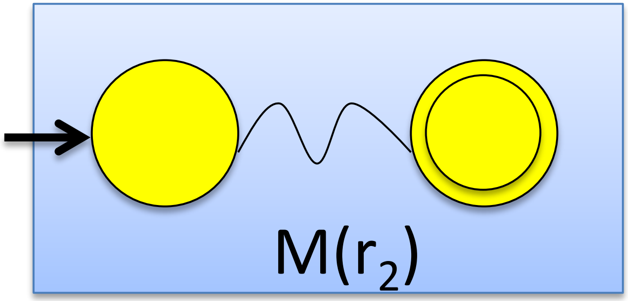
- Suppose this is the representation of an NFA accepting L(r) for regular expression r.



Convert Regular Expression to NFA

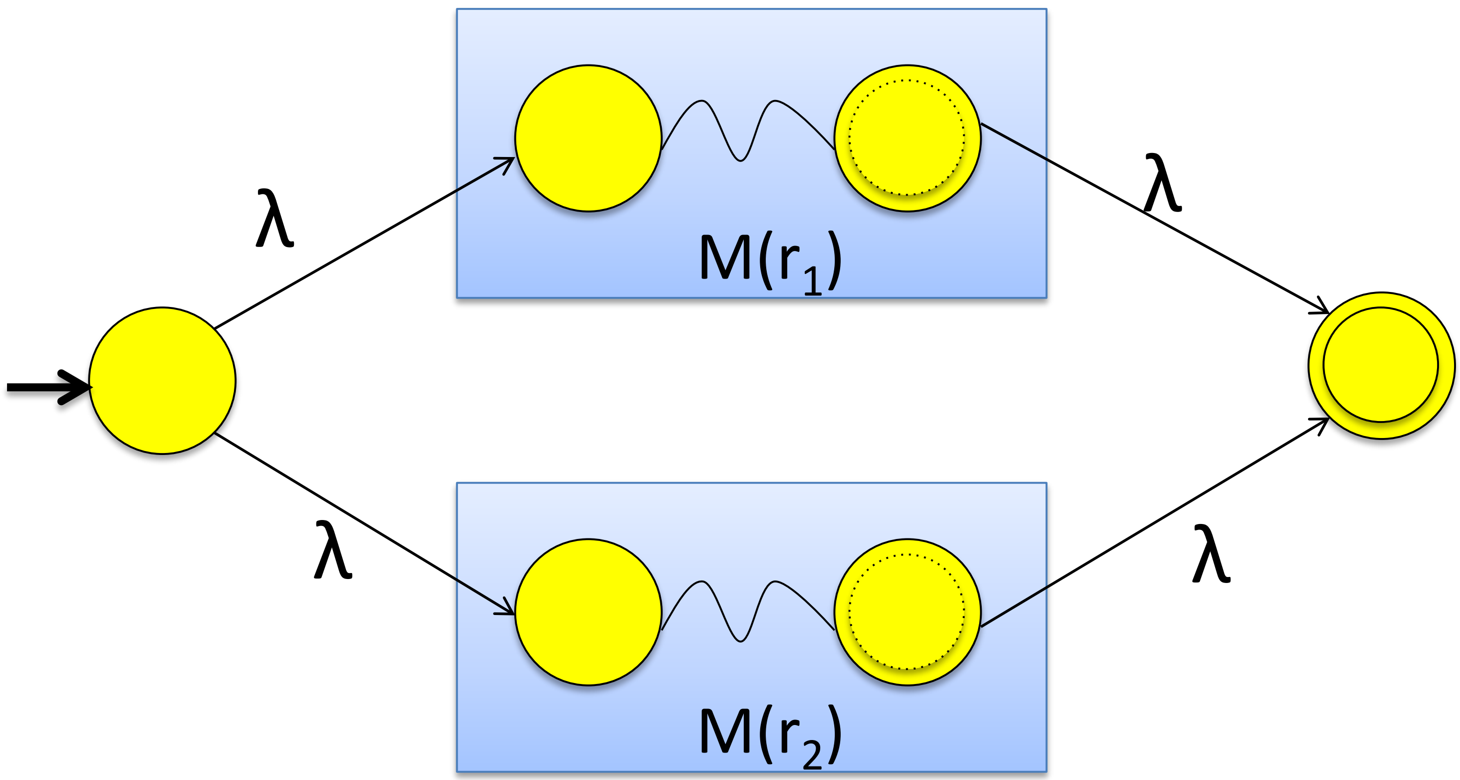
$r_1 + r_2$





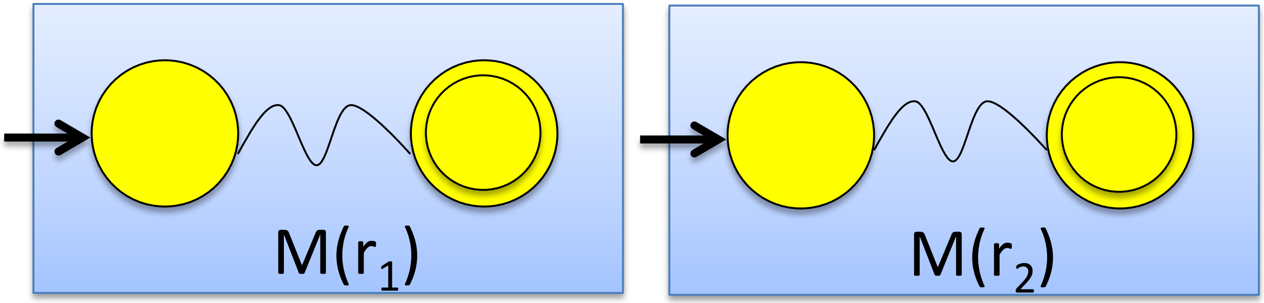
Convert Regular Expression to NFA

r_1+r_2



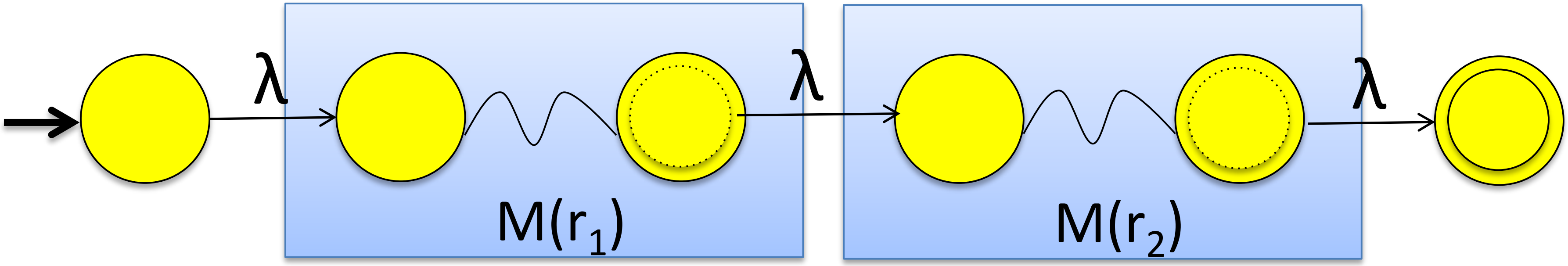
Convert Regular Expression to NFA

$r_1 r_2$



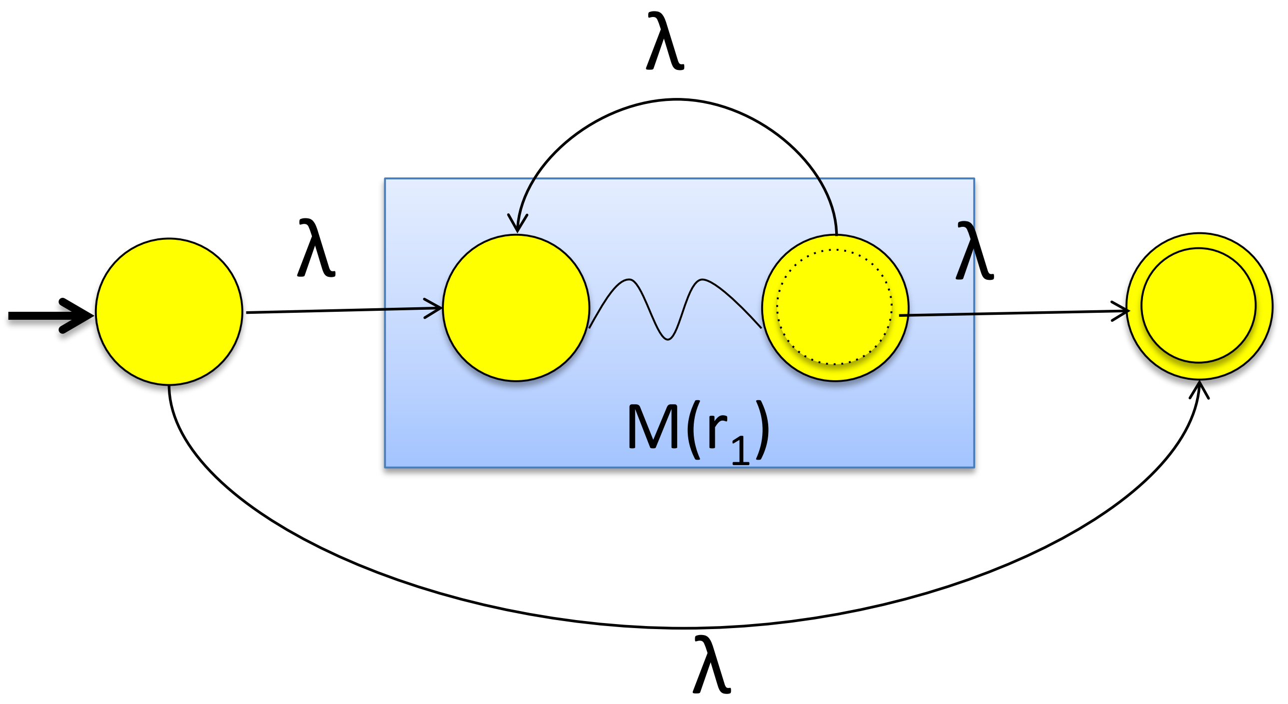
Convert Regular Expression to NFA

$r_1 r_2$



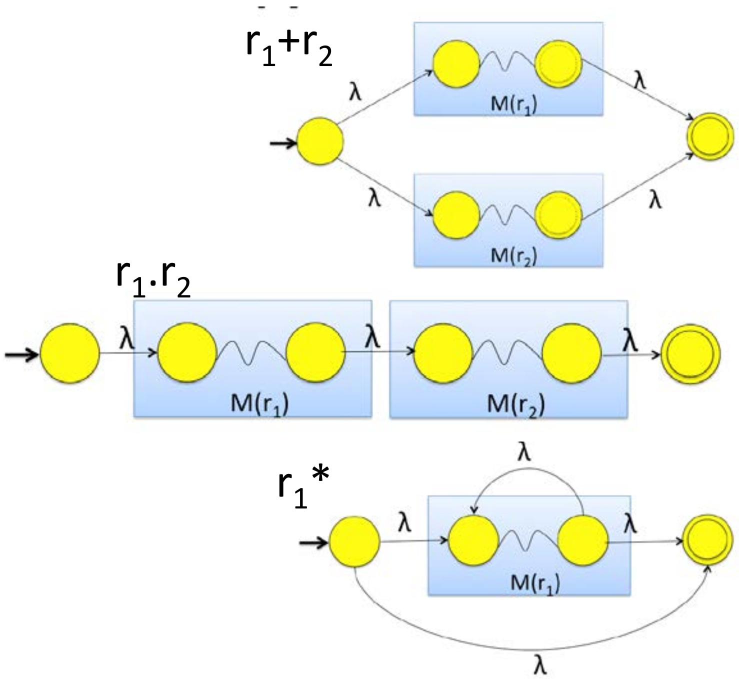
Convert Regular Expression to NFA

r_1*



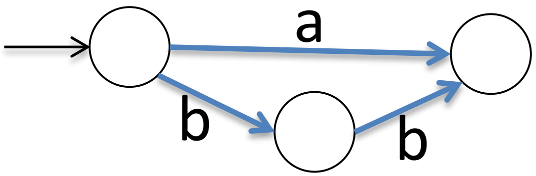
Example

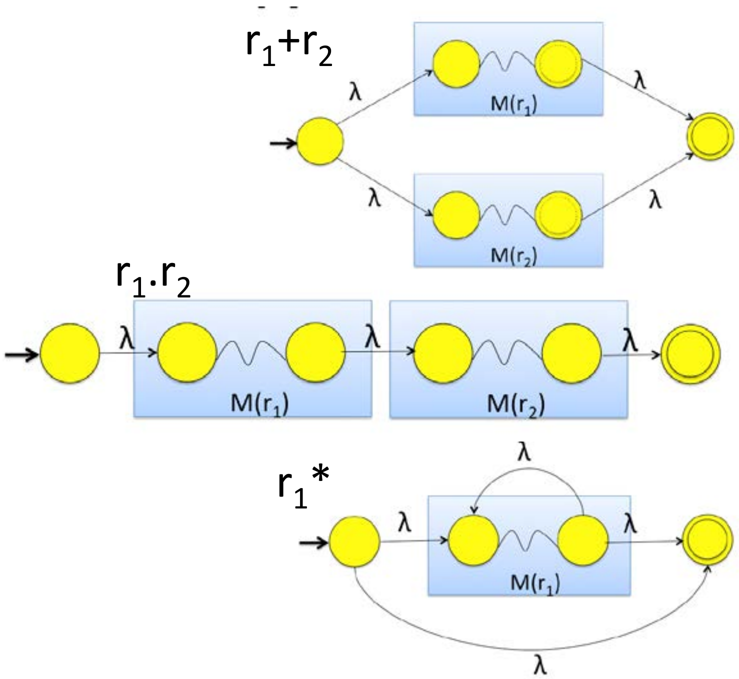
$r = (a+bb)^*(ba^*+\lambda)$



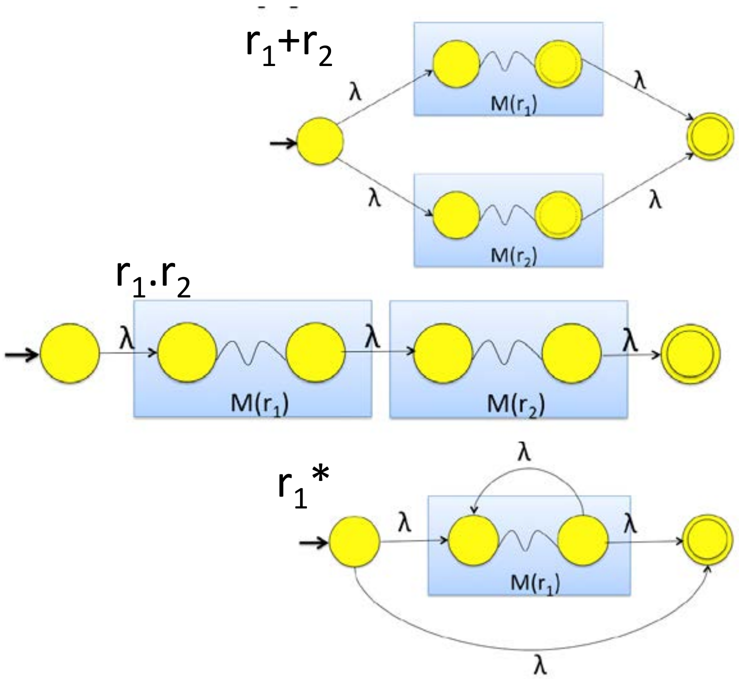
Example

r= (a+bb)*(ba*+λ)

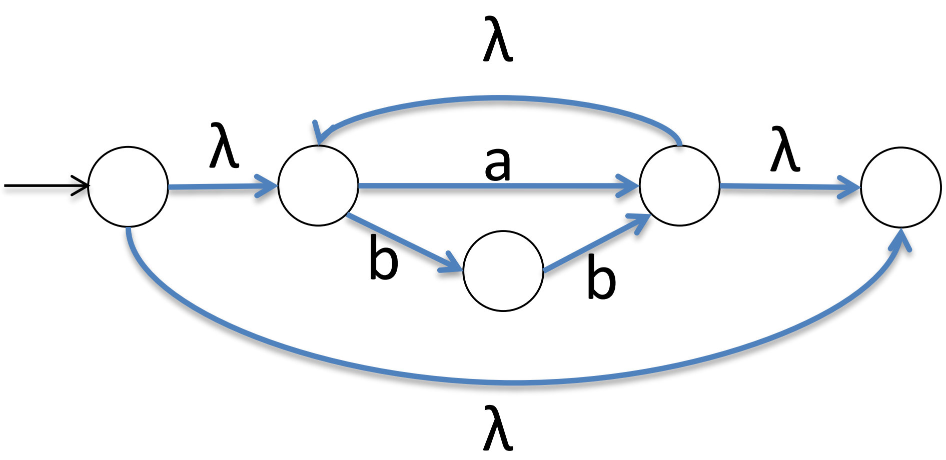




Example

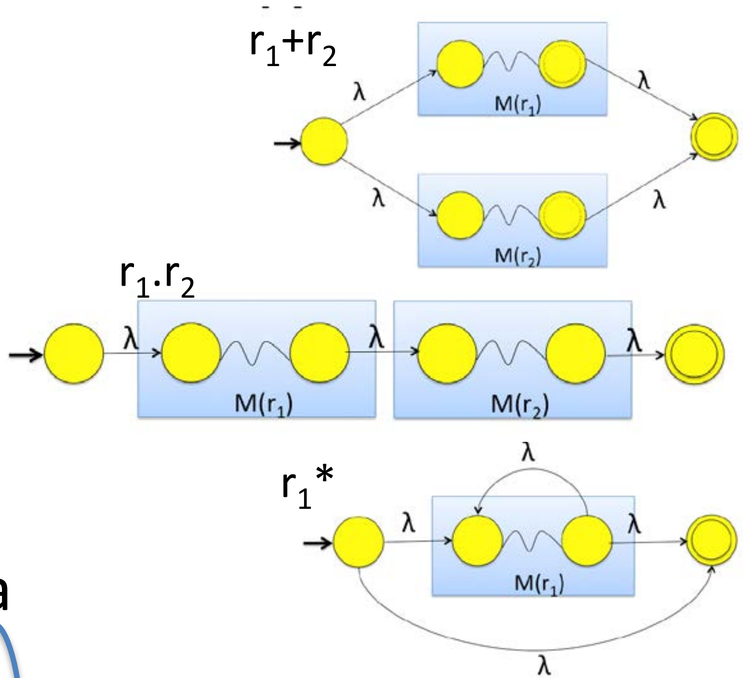


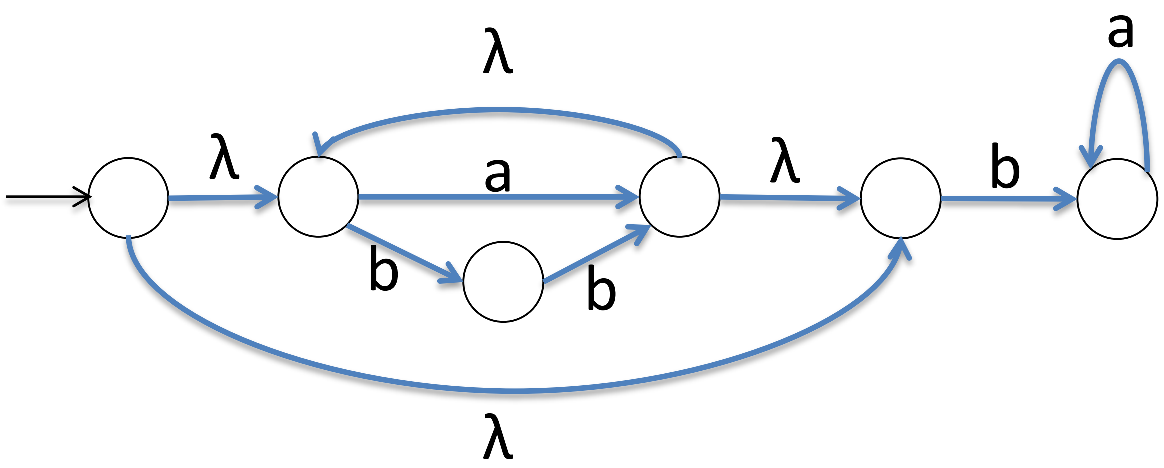
r= (a+bb)*(ba*+λ)



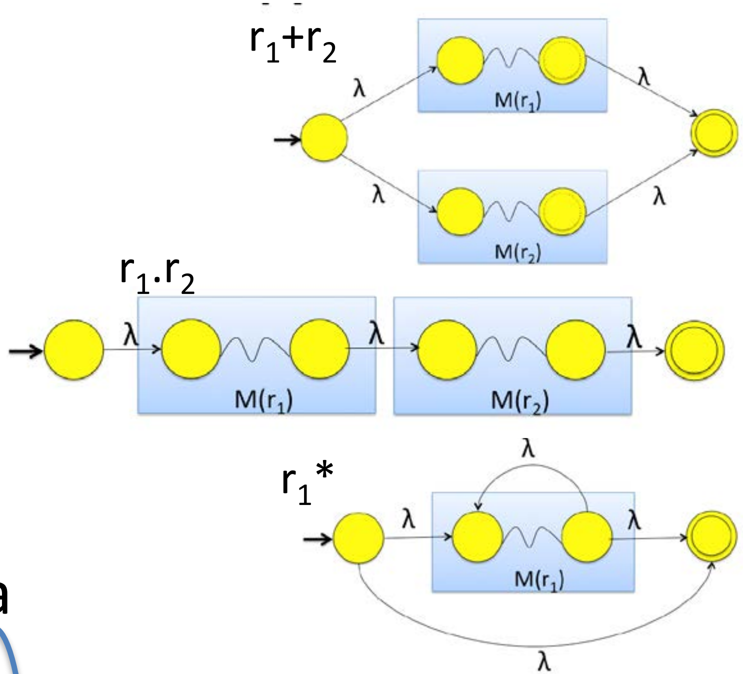
Example

$r = (a+bb)^*(ba^*+\lambda)$

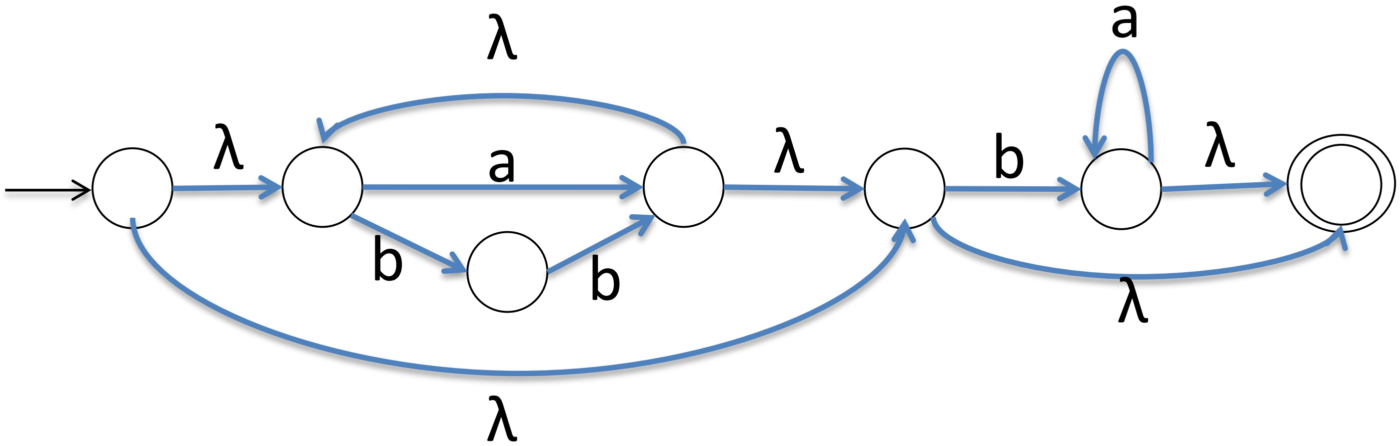




Example



r= (a+bb)*(ba*+λ)



Equivalence of Regular Expressions and Regular Languages

Languages generated by Regular Expressions = Regular languages

Part 2) Regular languages are a subset of the set of languages generated by regular expressions.

- For any regular language L there is a regular expression r with L(r) = L.

Equivalence of Regular Expressions and Regular Languages – Part 2

Theorem: Let L be a regular language accepted by an NFA M. Then there exists a regular expression r such that L = L(r).

- We will convert an NFA that accepts L to a regular expression.

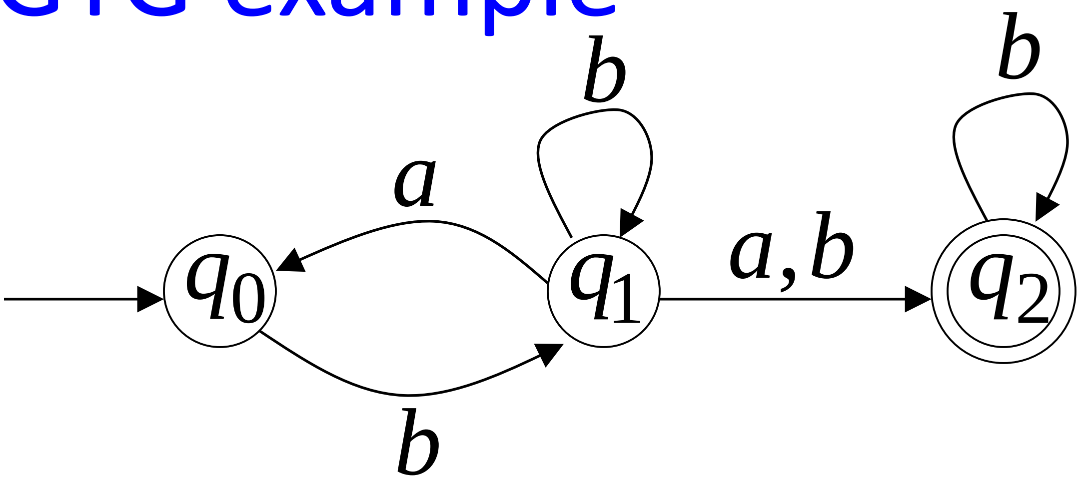
Convert NFA to Regular Expression

- We first construct the equivalent Generalized Transition Graph in which transition labels are regular expressions.

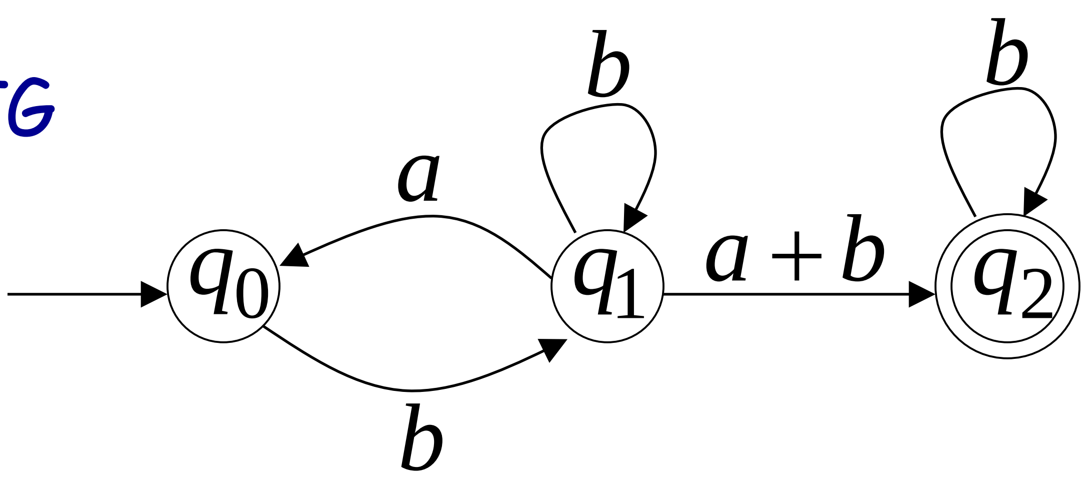
Edge in NFA	Edge in GTG
a	a
a,b,c	a+b+c

GTG example

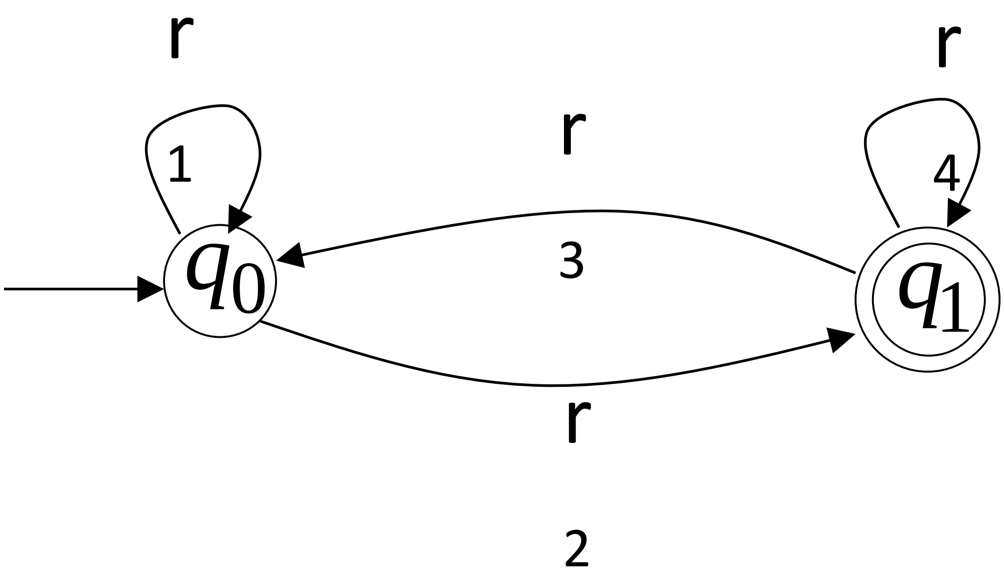
NFA



Corresponding GTG with regular expressions as labels



Example

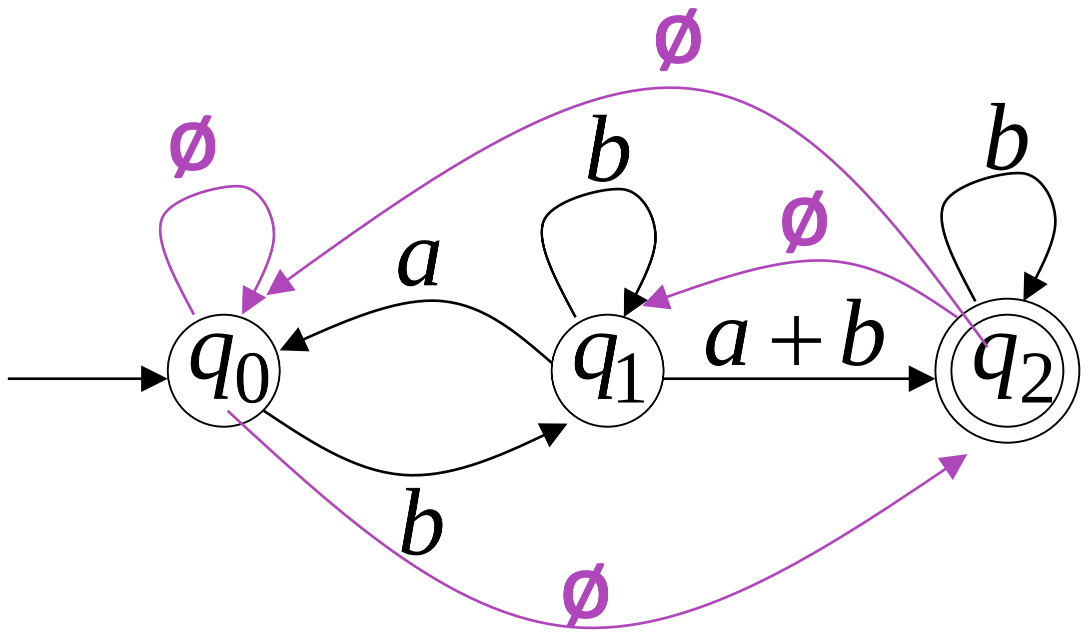


$$r = r_1{}^*r_2(r_4+r_3r_1{}^*r_2)^*$$

Creating Generalized Transition Graph (GTG)

- A complete GTG needs to have edges from every state to every other state.
- A complete GTG of $|V|$ nodes will have $|V|^2$ edges.
- If a GTG has some edges missing, we add the missing edges with label Ø .

Creating Generalized Transition Graph (GTG)

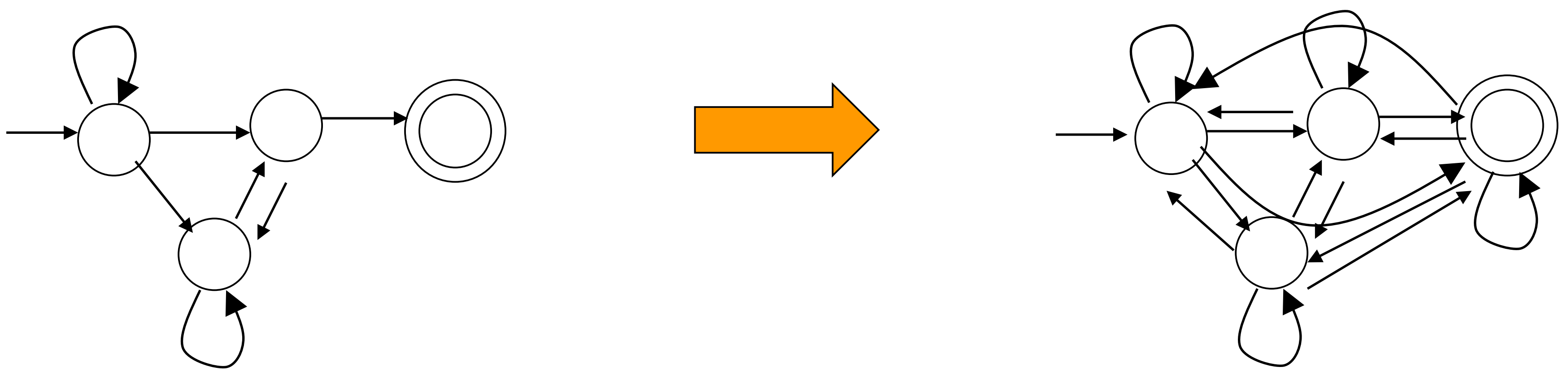


Convert NFA to Regular Expression

STEP 1

Initial NFA graph

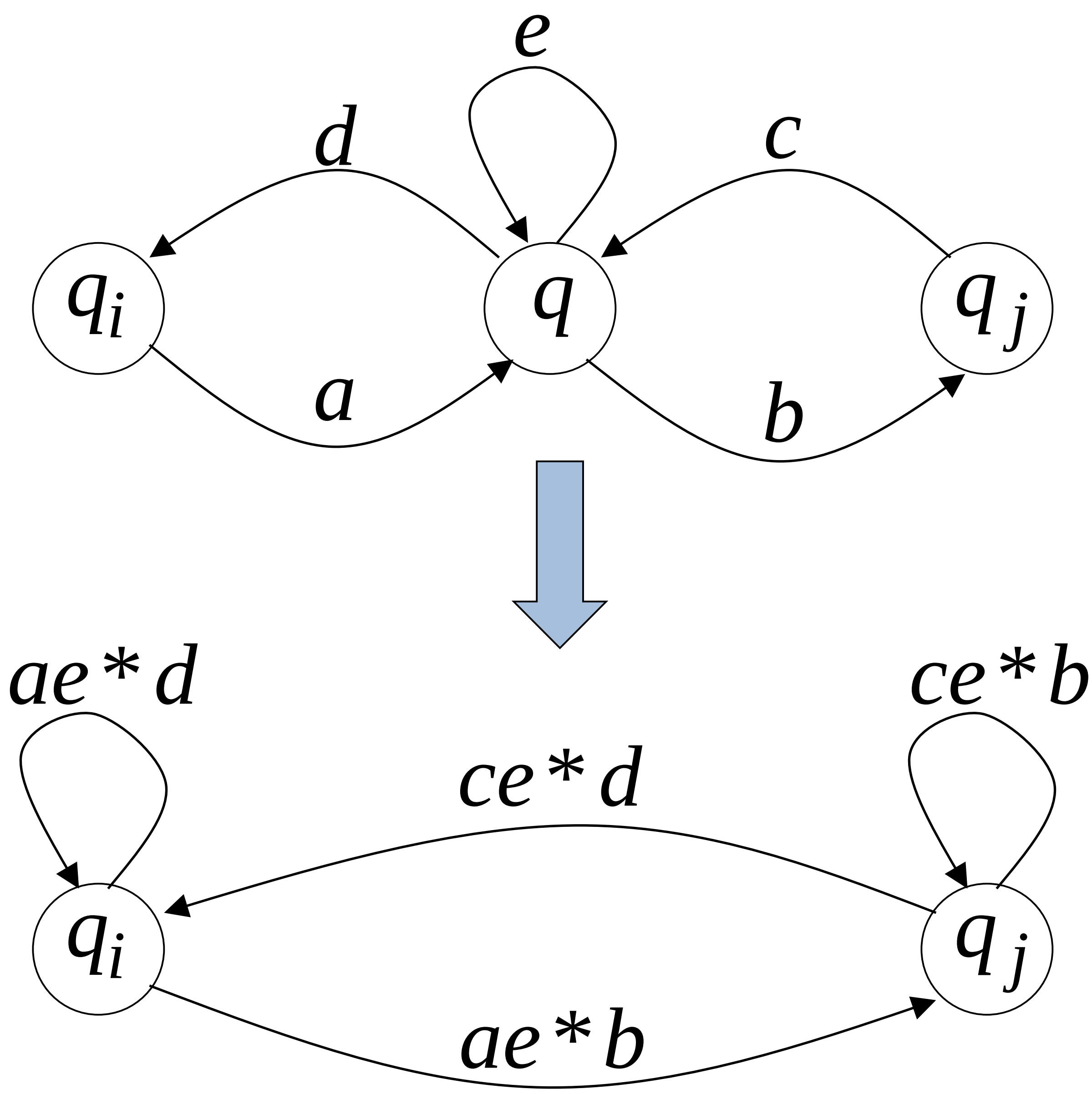
Corresponding GTG with regular expression labels



Convert NFA to Regular Expression

STEP 2

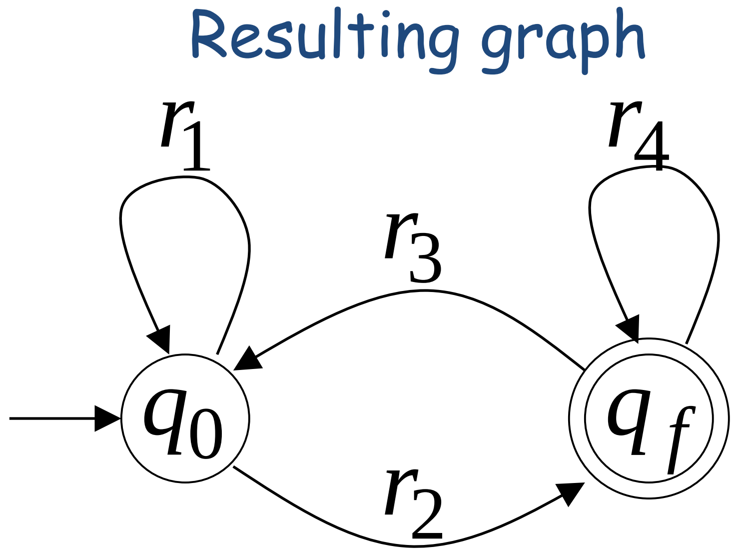
- Keep removing states one at a time from GTG until two states are left only.



Convert NFA to Regular Expression

STEP 3

- When GTG has two states, its associated regular expression is:



$$r = r_1{}^*r_2(r_4+r_3r_1{}^*r_2)^*$$

Convert NFA to Regular Expression

STEP 1: Create GTG

- Start with an NFA with a single final state, distinct from its initial state.
- Convert the NFA into a complete generalized transition graph (with expressions on the edges). r_{ij} is the label of the edge q_i to q_j.

Convert NFA to Regular Expression

STEP 2: removing states

- If the GTG has more than three states, pick an intermediate state q_k to be removed. Introduce new edges for all pairs of states (q_i, q_j), $i \neq k$ and $j \neq k$.

- If you have three states q_i, q_j, and q_k and you want to remove q_k, introduce new edges labeled

$$r_{pq} + r_{pk} r_{kk}^* r_{kq} \qquad \text{for } p=i,j \text{ and } q=i,j$$

Convert NFA to Regular Expression

STEP 2: removing states

- Note that

$$r + \emptyset = r$$

$$r.\emptyset = \emptyset$$

$$\emptyset^* = \lambda$$

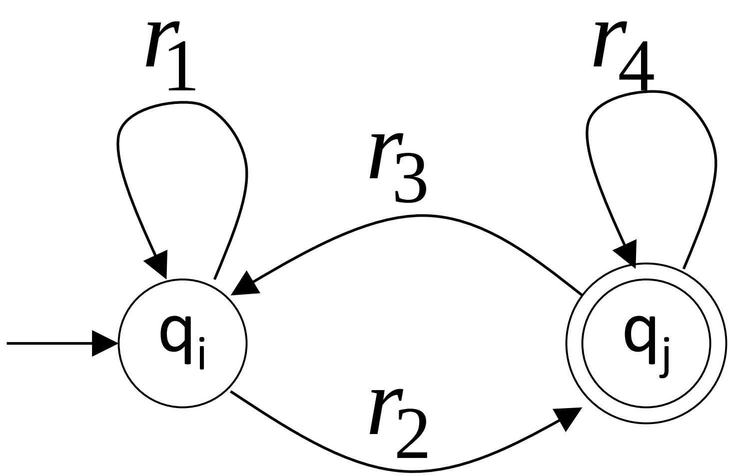
- Then remove node q_k and its associated edges.

Convert NFA to Regular Expression
STEP 3

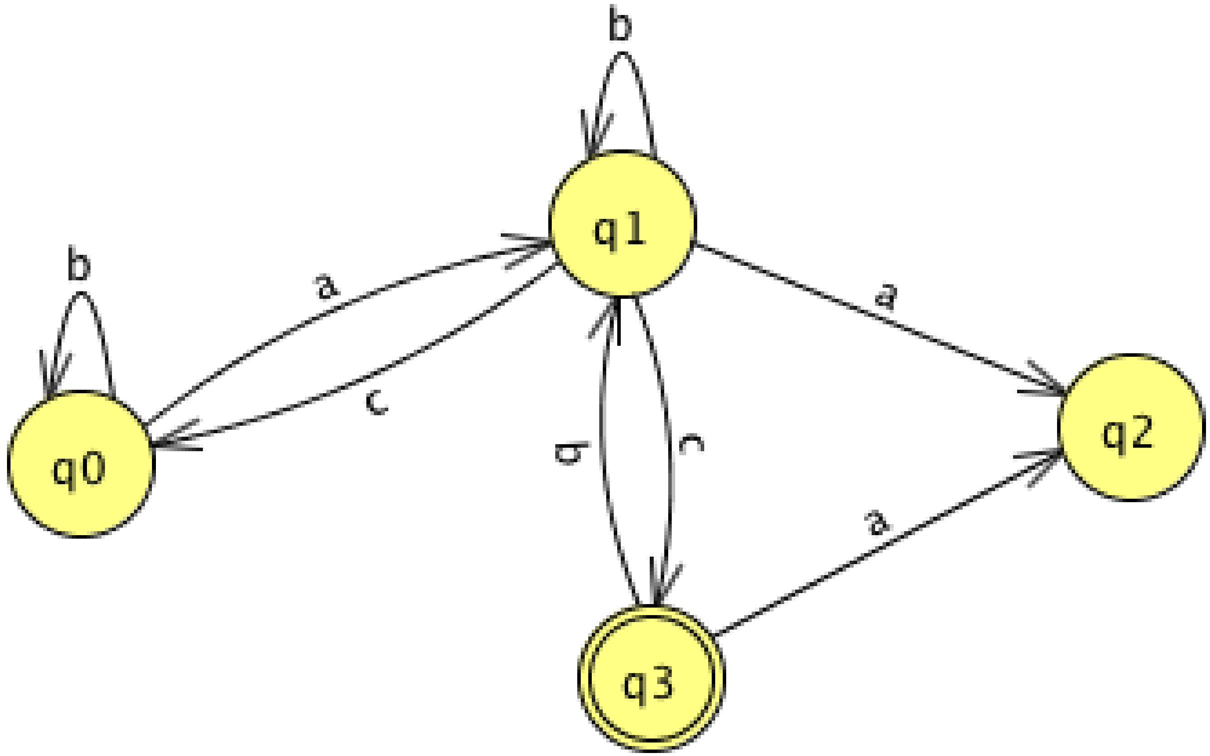
- If the GTG has only two states, with q_i as its initial state and q_j its final state, its associated regular expression is

$$r = r_{ii}^{*}\, r_{ij}\, (r_{jj} + r_{ji}\, r_{ii}^{*}\, r_{ij})^{*}$$

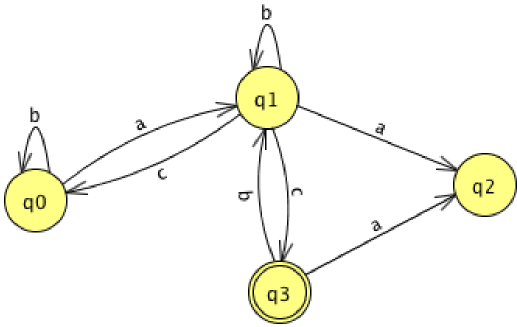
$$r = r_1^{*} r_2 (r_4 + r_3 r_1^{*} r_2)^{*}$$



Convert NFA to Regular Expression Example



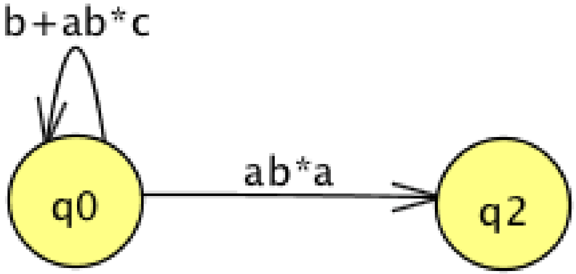
Convert NFA to Regular Expression Example



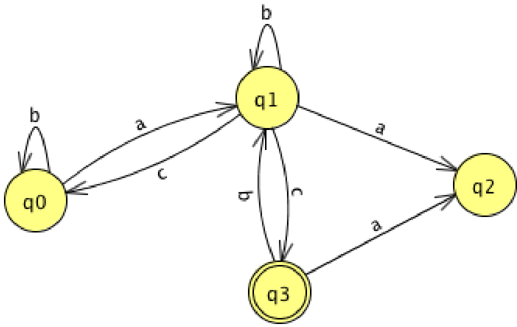
Eliminating q_1

We need to find transitions for all pairs of states (q_i, q_j)

1) (q_0, q_2):
   - p=0, q=0: b+ab*c
   - p=0, q=2: Ø +ab*a = ab*a
   - p=2, q=0: Ø +Ø b*c = Ø +Ø =Ø
   - p=2, q=2: Ø +Ø b*a = Ø + Ø = Ø



Convert NFA to Regular Expression Example

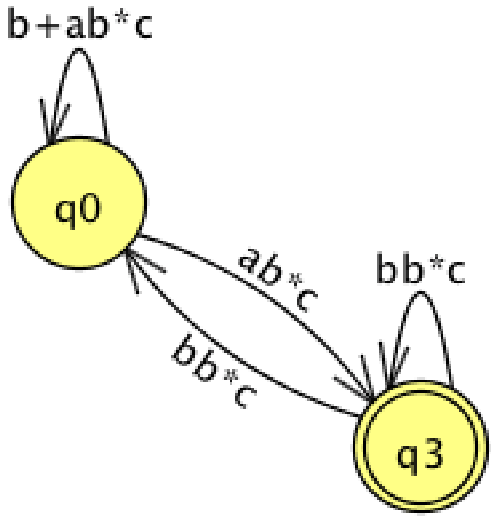


Eliminating q_1

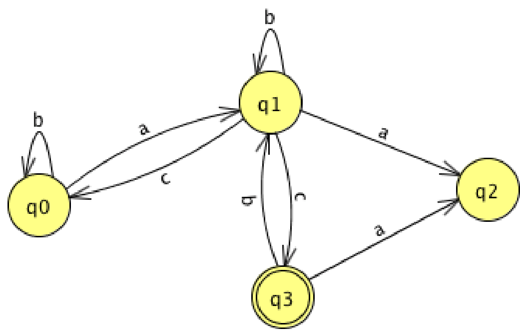
We need to find transitions for all pairs of states (q_i, q_j)

2) (q_0, q_3):

- p=0, q=0: b+ab*c
- p=0, q=3: Ø +ab*c = ab*c
- p=3, q=0: Ø +b b*c = bb*c
- p=3, q=3: Ø +b b*c = bb*c



Convert NFA to Regular Expression Example

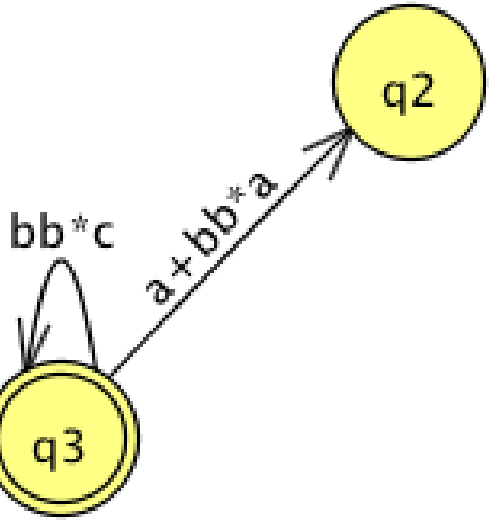


Eliminating q_1

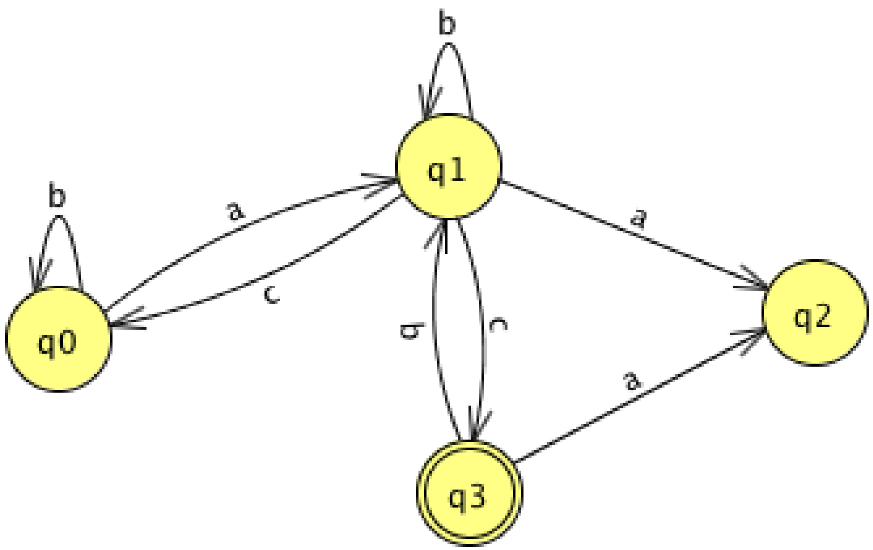
We need to find transitions for all pairs of states (q_i, q_j)

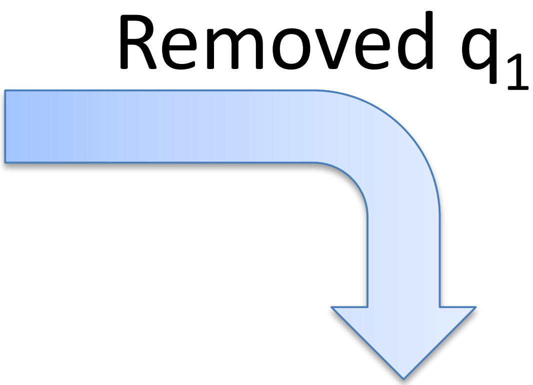
3) (q_2, q_3):

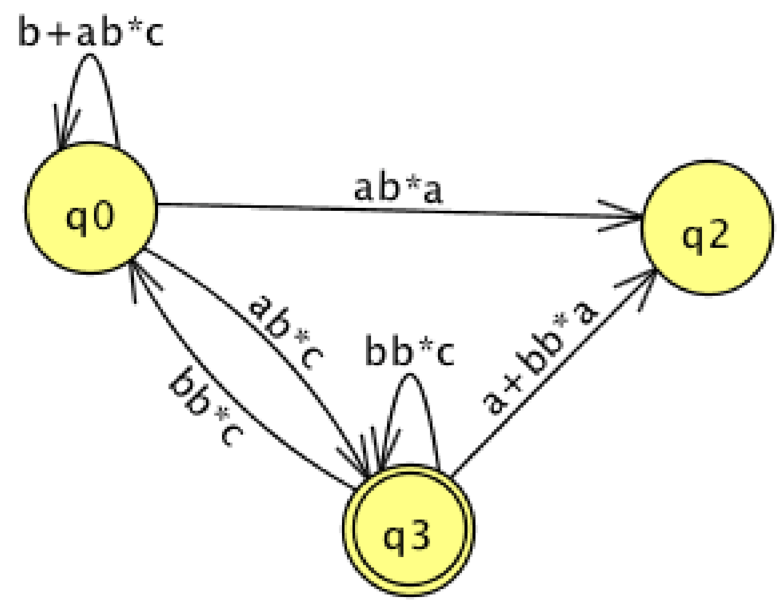
- p=2, q=2: Ø + Øb*a = Ø+Ø=Ø
- p=2, q=3: Ø +Øb*c = Ø+Ø=Ø
- p=3, q=2: a +b b*a
- p=3, q=3: Ø +b b*c = bb*c



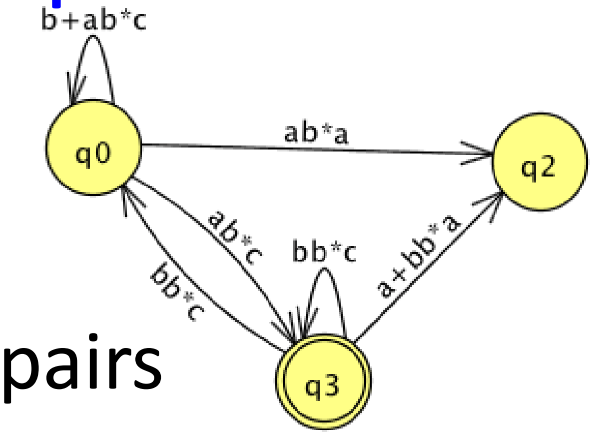
Convert NFA to Regular Expression Example







Convert NFA to Regular Expression Example

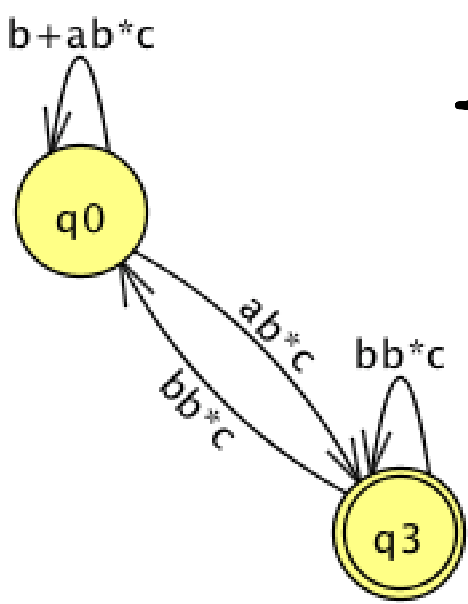


Eliminating q_2

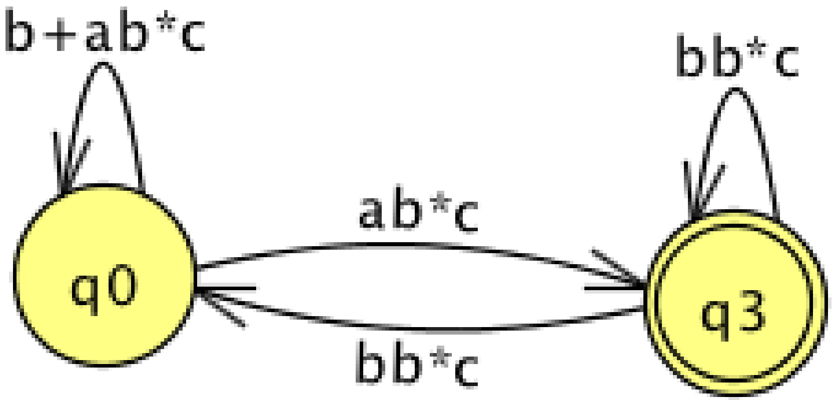
We need to find transitions for all pairs of states (q_i, q_j)

1) (q_0, q_3):

- p=0, q=0: (b+ab*c)+(ab*a)Ø*Ø = b+ab*c
- p=0, q=3: (ab*c)+ab*aØ*Ø = ab*c
- p=3, q=0: (bb*c) + (a+bb*a) Ø*Ø = bb*c
- p=3, q=3: (bb*c)+(a+bb*a) Ø*Ø = bb*c



Convert NFA to Regular Expression Example



$r = r_{ii}^{*}\, r_{ij}\, (r_{jj} + r_{ji}\, r_{ii}^{*}\, r_{ij})^{*}$

(b+ab*c)*(ab*c)((bb*c) + (bb*c)(b+ab*c)*ab*c)*

Summary

- Each of the three types of automata (DFA, NFA, ϵ-NFA) we discussed, and regular expressions as well, define exactly the same set of languages: the regular languages.