Regular Expressions

Regular Expressions

Regular expressions
describe regular languages

Example: $(a+b\cdot c)^*$

describes the language

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

Recursive Definition

Primitive regular expressions: $\varnothing, \quad \lambda, \quad a$

Given regular expressions r_1 and r_2

$$\left.\begin{array}{l} r_1 + r_2 \\ r_1 \cdot r_2 \\ r_1 {}^* \\ (r_1) \end{array}\right\} \text{Are regular expressions}$$

Languages of Regular Expressions

$L(r)$: language of regular expression r

Example

$$L((a+b\cdot c)^*) = \{\lambda, a, bc, aa, abc, bca, \ldots\}$$

Definition

For primitive regular expressions:

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

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

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

Definition (continued)

For regular expressions r_1 and r_2

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

$$L(r_1 \cdot r_2) = L(r_1)\, L(r_2)$$

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

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

Example

Regular expression: $(a+b)\cdot a^*$

$$
\begin{aligned}
L((a+b)\cdot a^*) &= L((a+b))\,L(a^*) \\
&= L(a+b)\,L(a^*) \\
&= (L(a)\cup L(b))\,(L(a))^* \\
&= (\{a\}\cup\{b\})\,(\{a\})^* \\
&= \{a,b\}\,\{\lambda,a,aa,aaa,\ldots\} \\
&= \{a,aa,aaa,\ldots,b,ba,baa,\ldots\}
\end{aligned}
$$

Example

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

$$L(r) =$$

$$L(r) = \{a, bb, aa, abb, ba, bbb, \ldots\}$$

Example

Regular expression for zero or more repetitions of the symbols 0 and 1

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

Example

Regular expression for even binary numbers

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

Example

Regular expression for the language which accepts strings that begin with atleast 2 a's and end with an even number of b's (with nothing else in between)

$$r = aa(a)^*(bb)^*$$

Regular expression for the language which accepts strings containing no more than two 0's

1. Having no 0 at all.
2. Having exactly one 0.
3. Having exactly two 0s.

$$1^* + 1^*.0.1^* + 1^*.0.1^*.0.1^*$$

$$1^*(\lambda + 0.1^*(\lambda + 0.1^*))$$

Regular expression for the language which accepts strings whose length is a multiple of 3 over the alphabet = {a,b}

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

Regular expression for the language which accepts strings that contains atleast one a and atleast one b over the alphabet ={a,b,c}

a string that contains at least one b can be represented by

$$(a+b+c)^*.b.(a+b+c)^*$$

$$(a+b+c)^*.a.(a+b+c)^*$$

$$(a+b+c)^*.a.(a+b+c)^*.b.(a+b+c)^*$$

$$(a+b+c)^*.a.(a+b+c)^*.b.(a+b+c)^* + (a+b+c)^*.b.(a+b+c)^*.a.(a+b+c)^*$$

$L = \{ uvw \mid u, w \text{ belong to } \Sigma^* \text{ and } |v| = 2 \}$

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

Example

Regular expression $r = (aa)^*(bb)^*b$

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

Example

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

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

Example

Regular expression $r = (1 + 01)^*(0 + \lambda)$

$L(r)$ = { all strings without two consecutive 0 }

Equivalent Regular Expressions

Definition:

Regular expressions r_1 and r_2

are **equivalent** if $L(r_1) = L(r_2)$

Example

L = { all strings without two consecutive 0 }

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

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

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

Find a regular expression for the set $\{a^n b^m : n \geq 3, m \text{ is even}\}$.

$aaa(a^*)(bb)^*$

Find a regular expression for the set $\{a^n b^m : (n + m) \text{ is even}\}$.

$$\left[(aa)^*(bb)^* + (aa)^*a(bb)^*b \right]$$

Kleen's Theorem

1. **For any Regular Expression *r* that represents Language L(r), there is a Finite Automata that accepts same language.**

-It states that any language accepted by a finite automaton is regular.

2. **Every regular language that can be defined by a FA can be defined by a regular expression**

- Any language accepted by a finite automaton is regular.

Equivalence of FA and RE

Finite Automata and Regular Expressions are equivalent. To show this:

- Show we can express a RE as an ε-NFA. Since the ε-NFA can be converted to a DFA and the DFA to an NFA, then RE will be equivalent to all the automata we have described.
- Show we can express a DFA as an equivalent RE

From a Regular Expression to an NFA

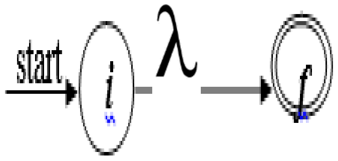
Now give an algorithm to construct an NFA from a regular expression. The algorithm is syntax-directed in that it uses the syntactic structure of the regular expression to guide the construction process.

Algorithm: (Thompson's construction):

Input: a regular expression R over alphabet ∑.

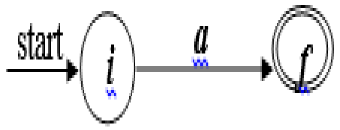
Output: NFA N accepting L(R).

1- For λ construct the NFA

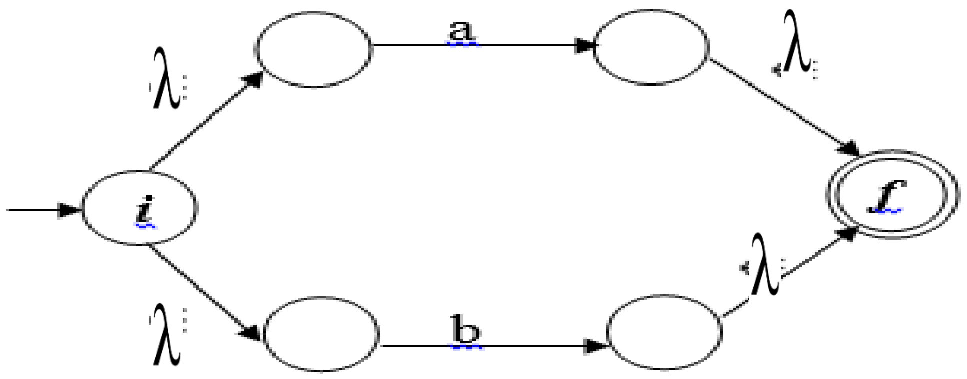


From a Regular Expression to an NFA

2- For ***a*** in ∑, construct the ***NFA***

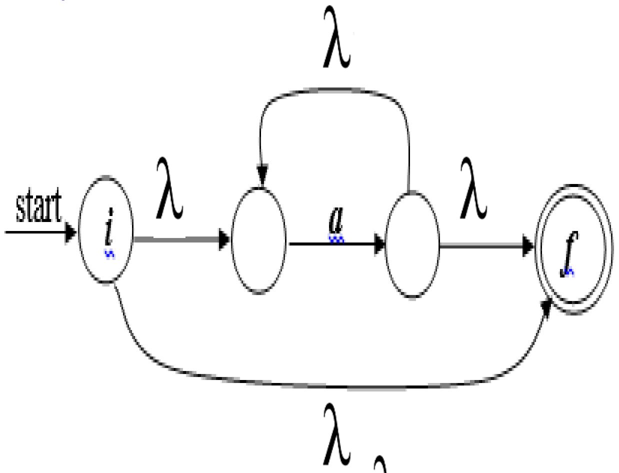


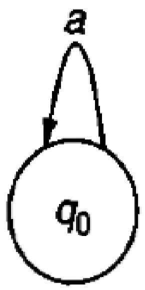
3- For the regular expression ***a | b*** construct the following composite NFA ***N(a | b)***.



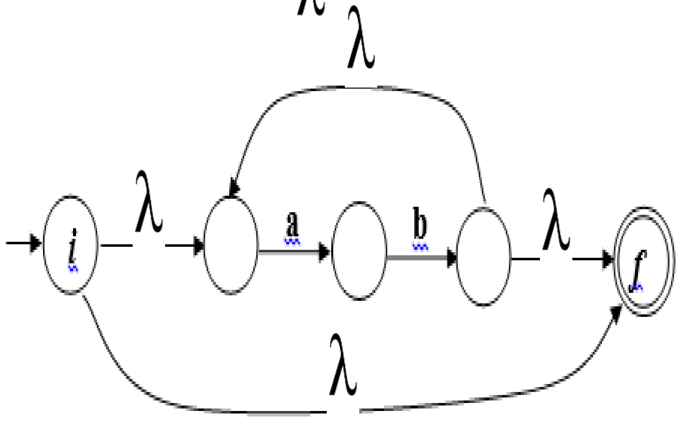
From a Regular Expression to an NFA

4-For the regular expression ***a**** construct the following composite NFA ***N(a*)***.



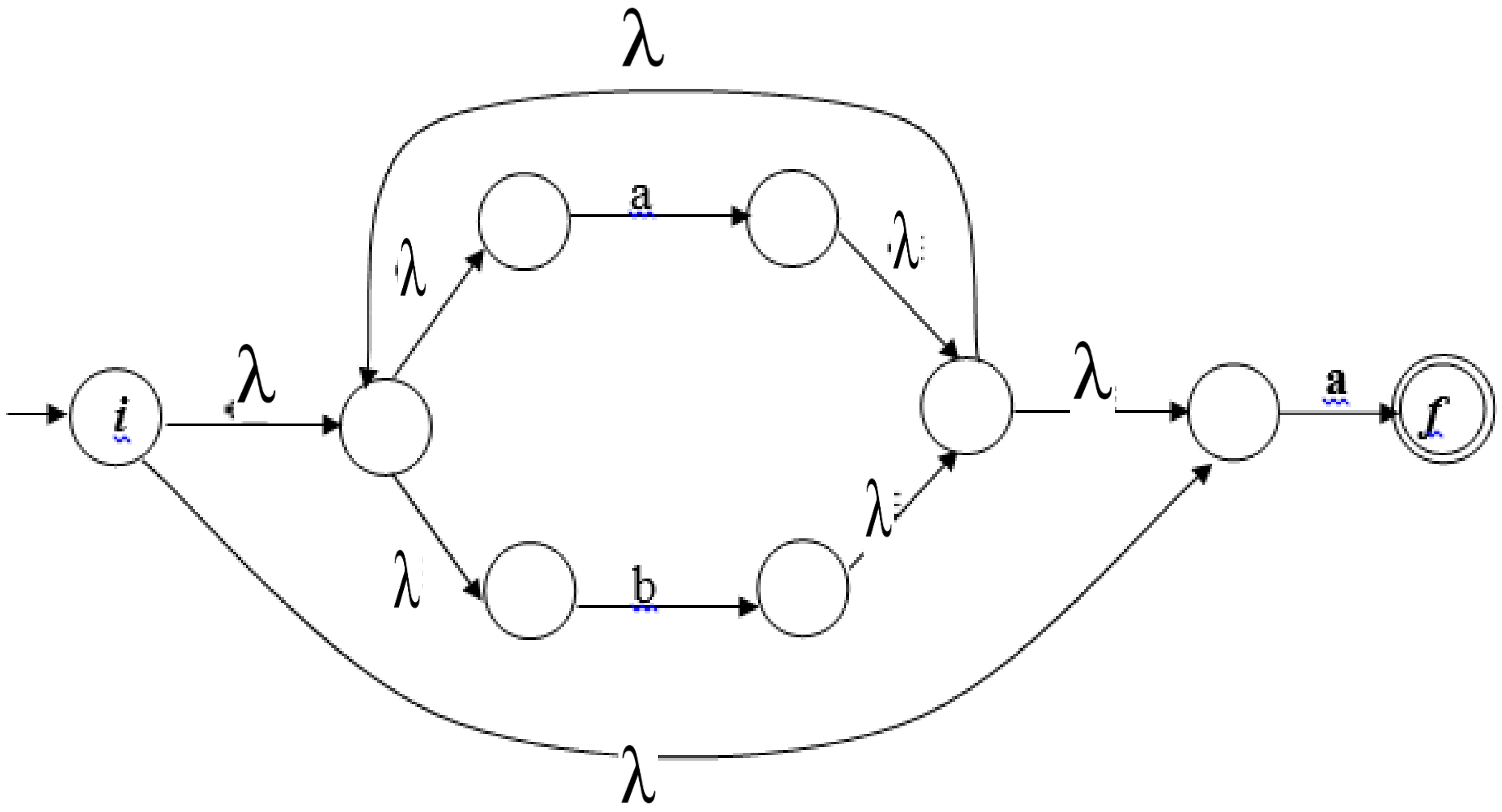


RE = ***(ab)****



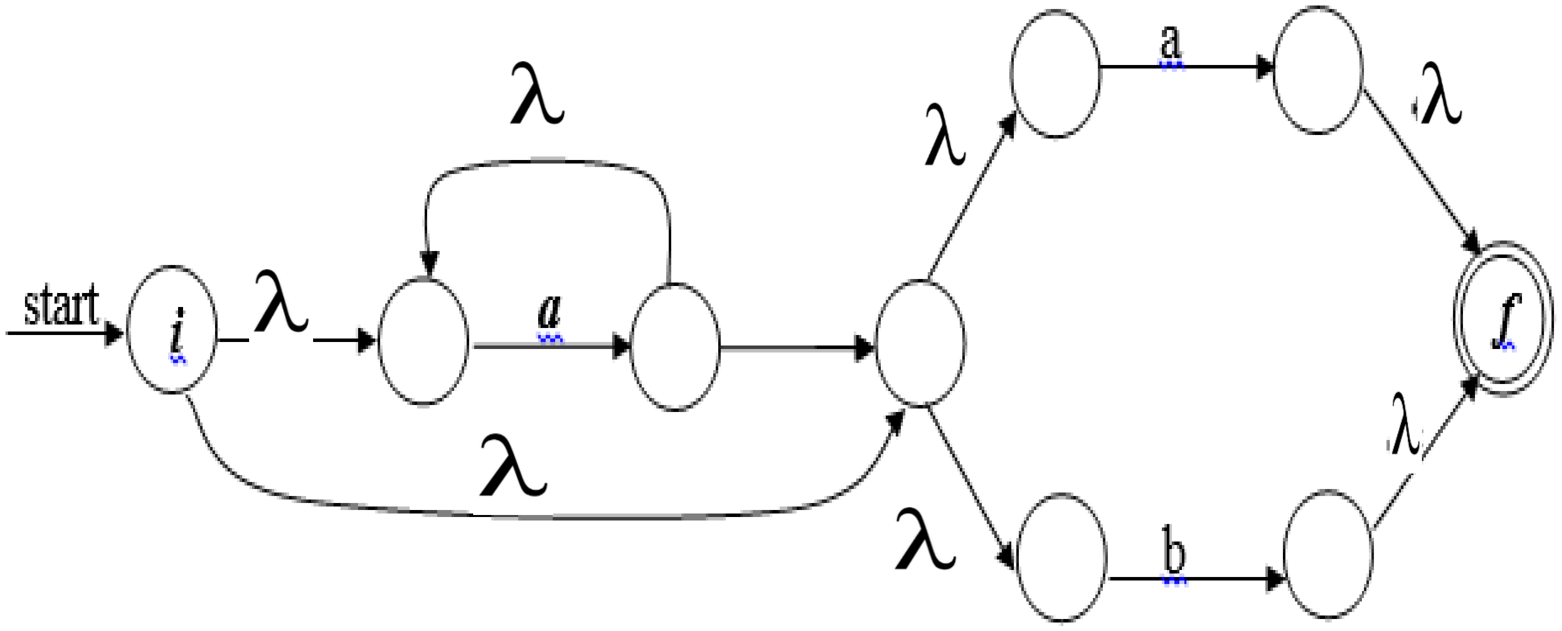
From a Regular Expression to an NFA

RE = (a | b)*a

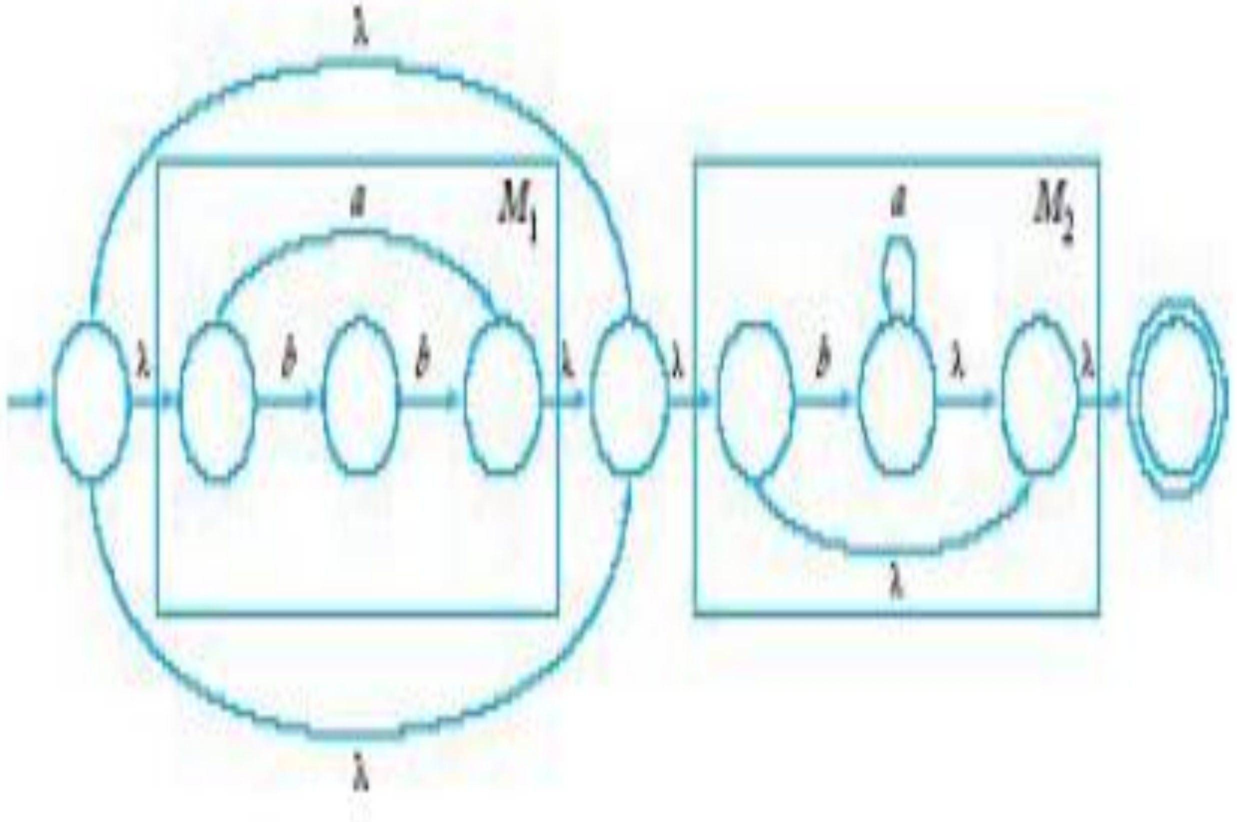


From a Regular Expression to an NFA

RE = *a* (a | b)*

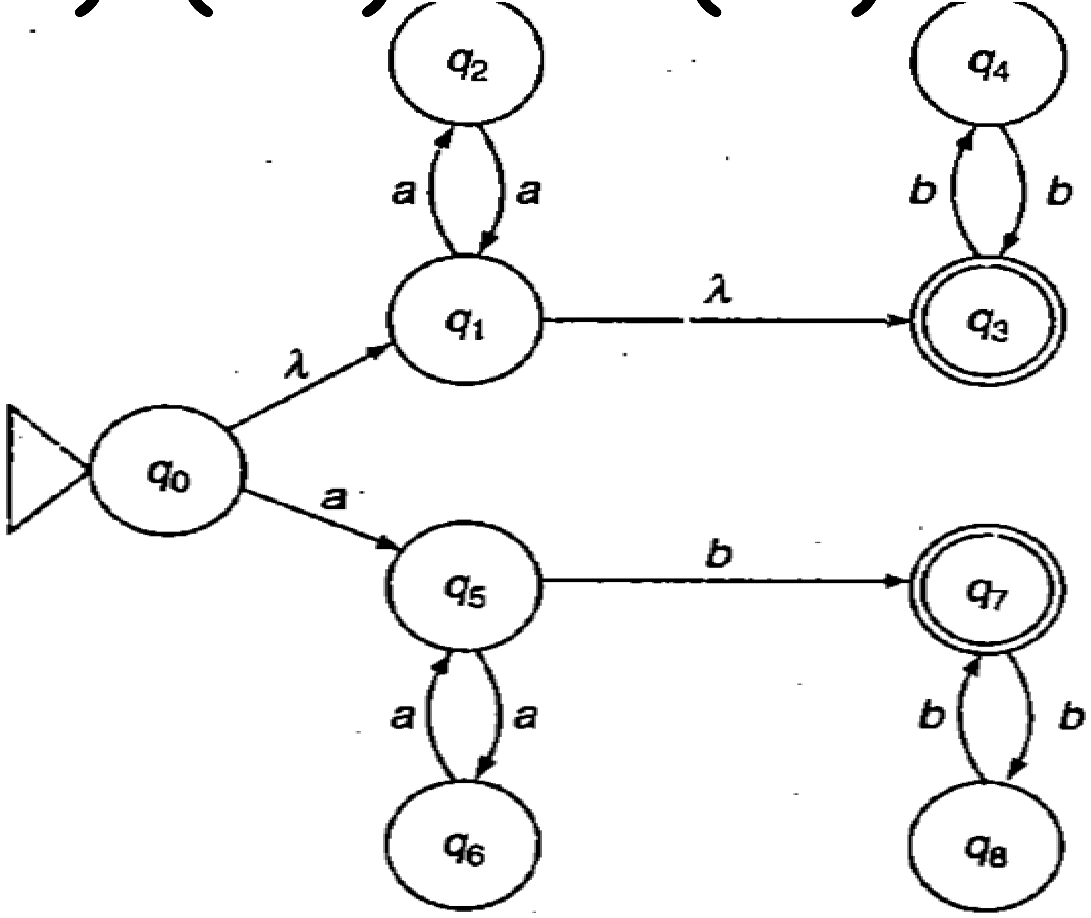


$r = (a + bb)^* (ba^* + \lambda)$

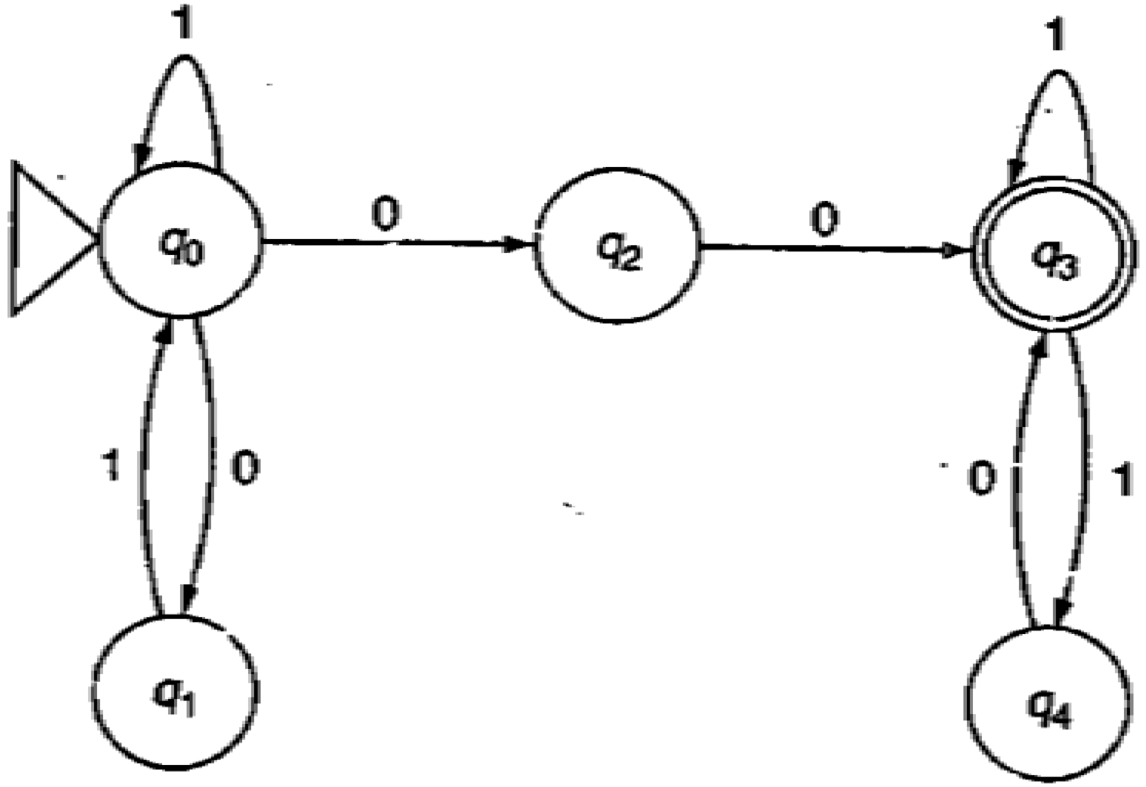


Regular Expressions

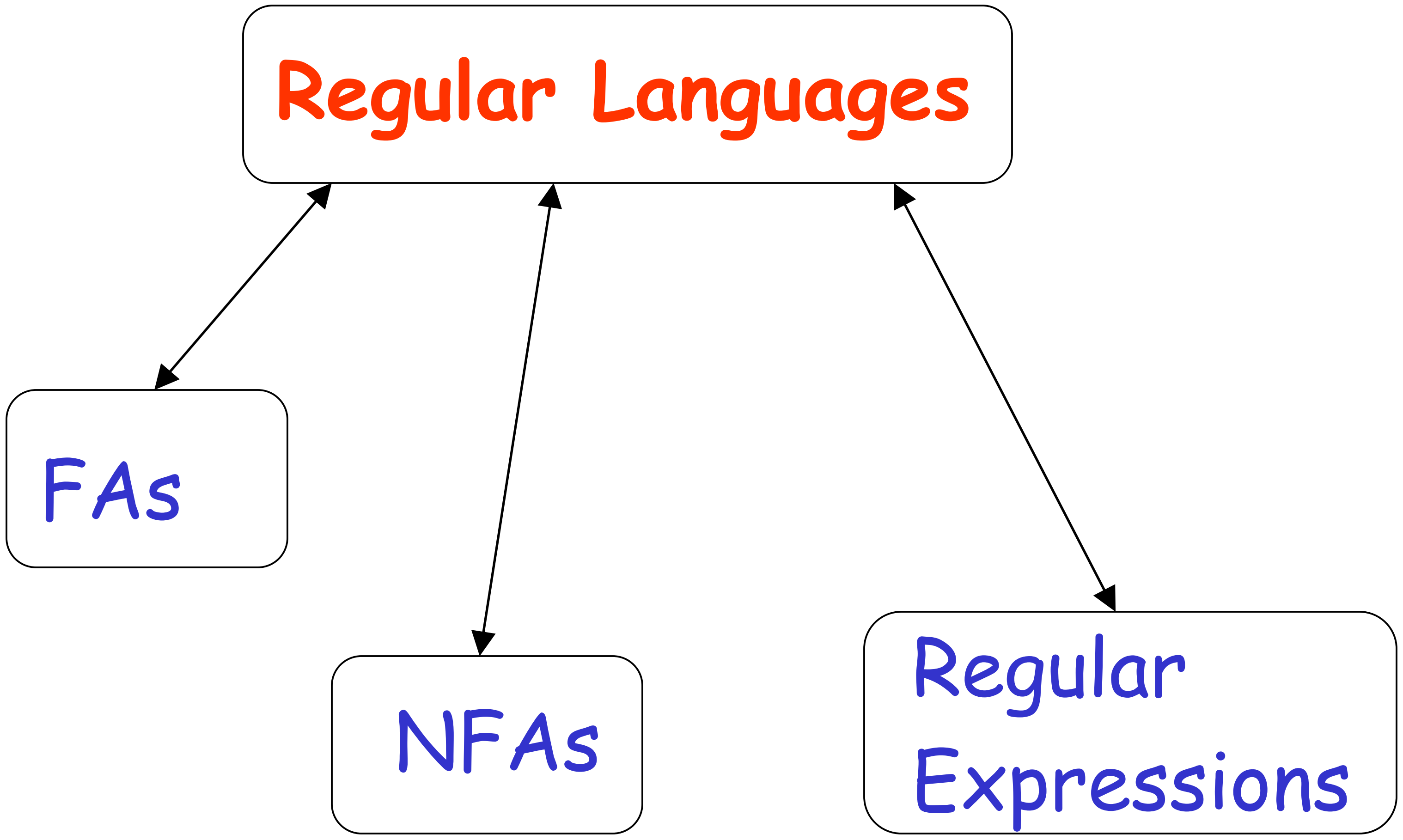
(aa)*(bb)* + a (aa)* b (bb)*



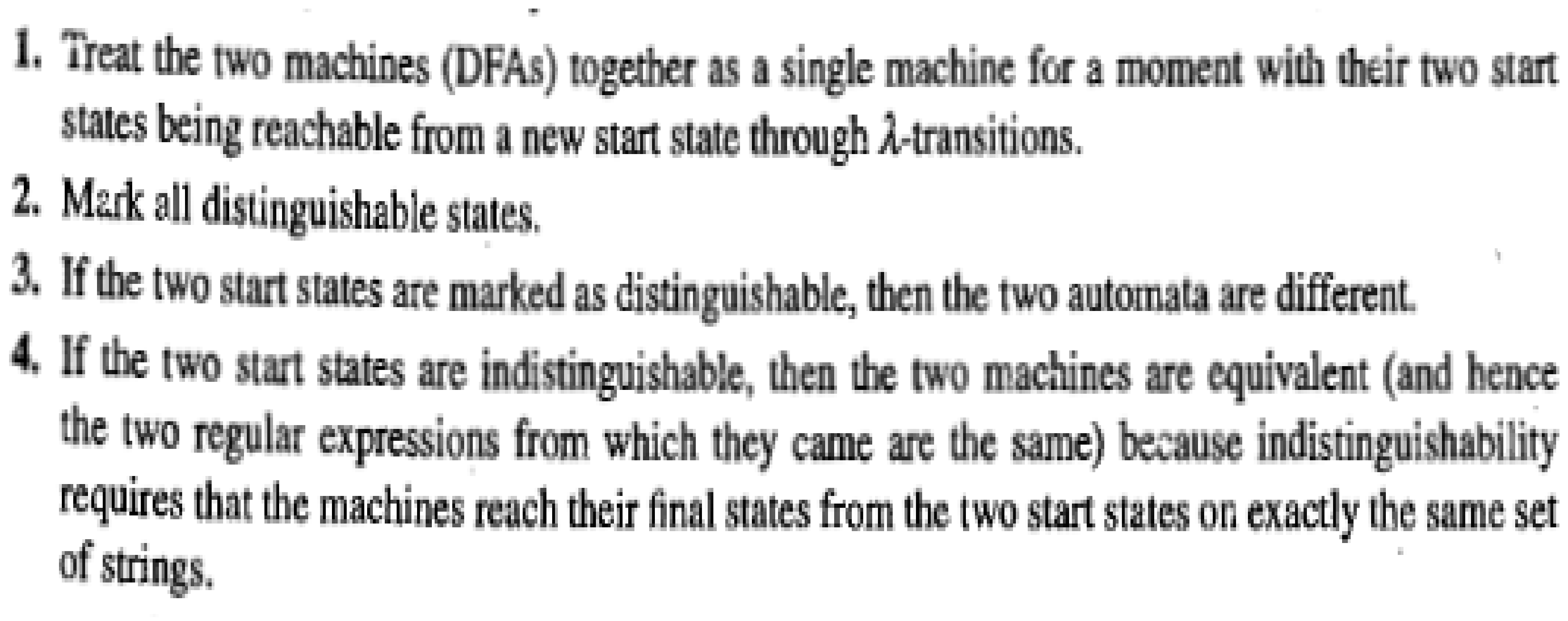
$(1 + (01))^* 00 (1+(10))^*$



Standard Representations of Regular Languages



EQUIVALENCE OF REGULAR EXPRESSIONS



1. Treat the two machines (DFAs) together as a single machine for a moment with their two start states being reachable from a new start state through λ-transitions.
2. Mark all distinguishable states.
3. If the two start states are marked as distinguishable, then the two automata are different.
4. If the two start states are indistinguishable, then the two machines are equivalent (and hence the two regular expressions from which they came are the same) because indistinguishability requires that the machines reach their final states from the two start states on exactly the same set of strings.

Consider the two regular expressions:

$$0^*10^*1(0+1)^*$$

and $$((00)^*1+0(00)^*1)((00)^*1+0(00)^*1)(0+1)^*$$

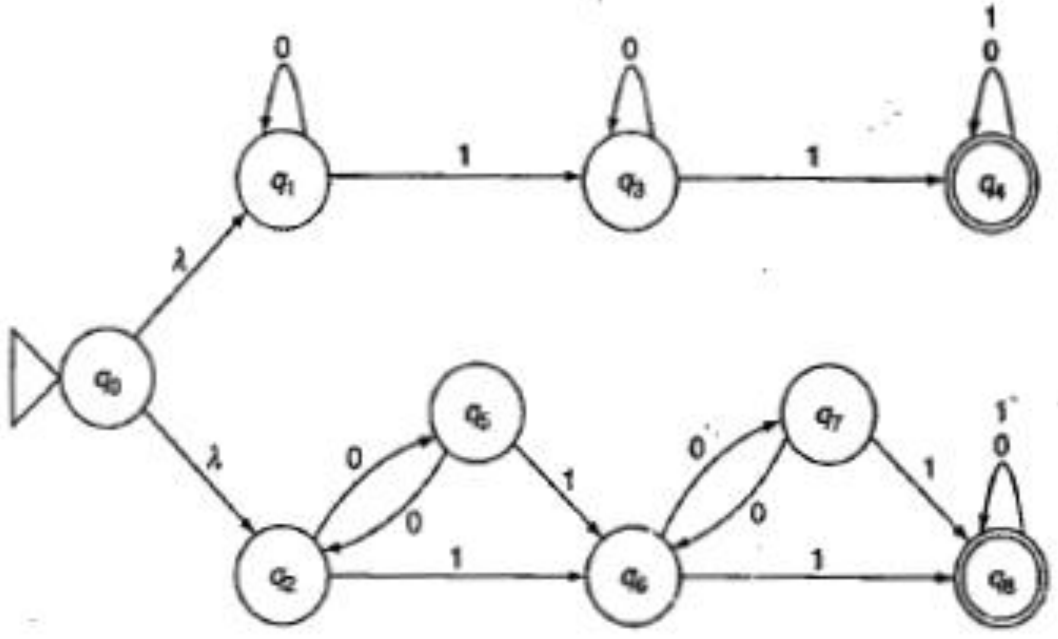
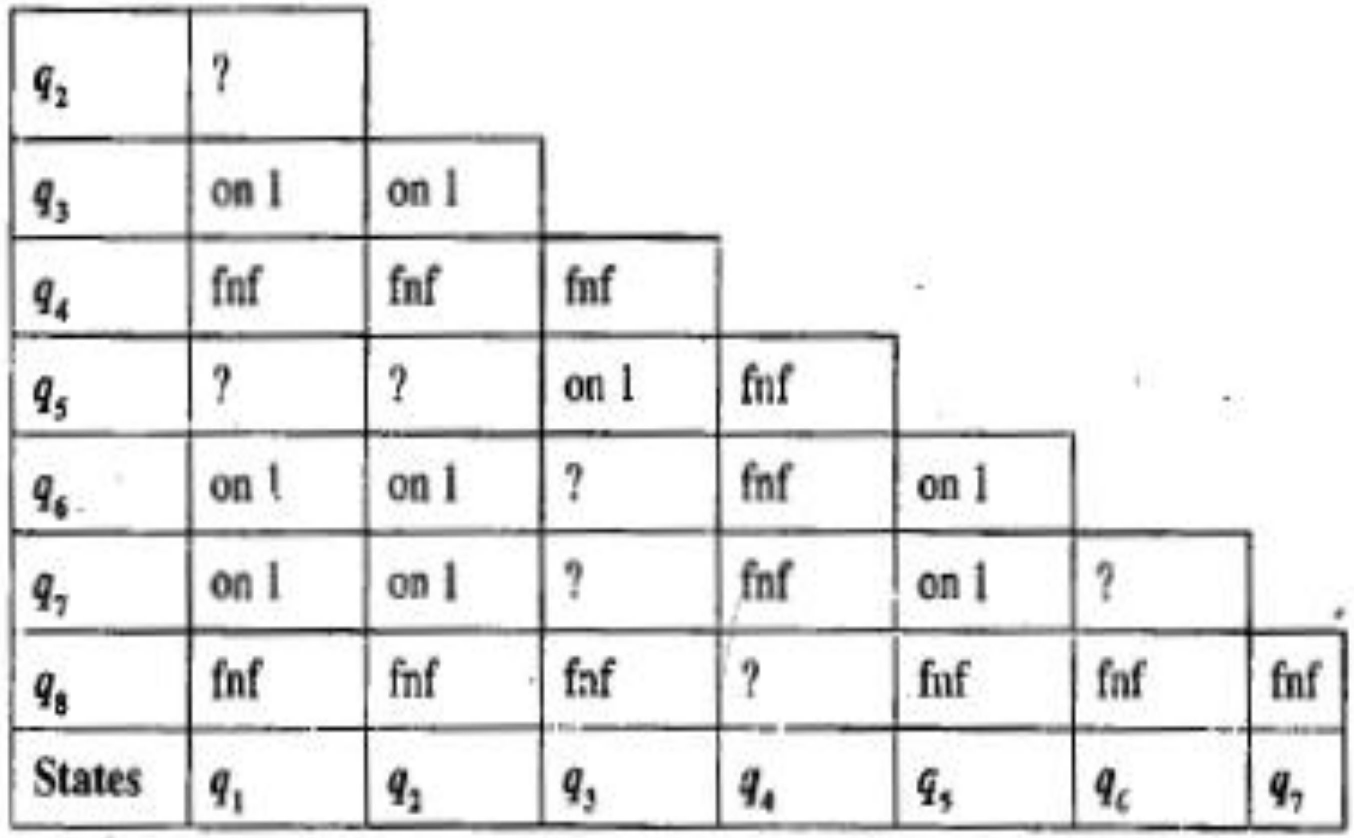


FIGURE 4.9 Combining automata to compare two regular expressions.

BY IGNORING NEW START STATE q0

TABLE 4.3 Marking Distinguishable States for Comparing Regular Expressions



q_2	?						
q_3	on 1	on 1					
q_4	fnf	fnf	fnf				
q_5	?	?	on 1	fnf			
q_6	on 1	on 1	?	fnf	on 1		
q_7	on 1	on 1	?	fnf	on 1	?	
q_8	fnf	fnf	fnf	?	fnf	fnf	fnf
States	q_1	q_2	q_3	q_4	q_5	q_6	q_7

Regular Expressions
and
Regular Languages

Theorem

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{Generated by} \\ \text{Regular Expressions} \end{array} \right\} = \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

We will show:

$$\left\{\begin{array}{l}\text{Languages} \\ \text{Generated by} \\ \text{Regular Expressions}\end{array}\right\} \subseteq \left\{\begin{array}{l}\text{Regular} \\ \text{Languages}\end{array}\right\}$$

$$\left\{\begin{array}{l}\text{Languages} \\ \text{Generated by} \\ \text{Regular Expressions}\end{array}\right\} \supseteq \left\{\begin{array}{l}\text{Regular} \\ \text{Languages}\end{array}\right\}$$

Proof - Part 1

$$\left\{\begin{array}{l}\text{Languages}\\ \text{Generated by}\\ \text{Regular Expressions}\end{array}\right\} \subseteq \left\{\begin{array}{l}\text{Regular}\\ \text{Languages}\end{array}\right\}$$

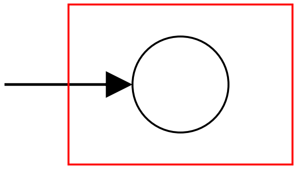
For any regular expression r
the language $L(r)$ is regular

Proof by induction on the size of r

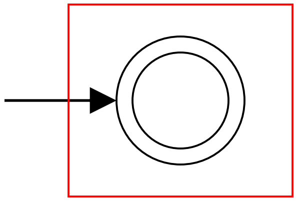
Induction Basis

Primitive Regular Expressions: $\varnothing, \quad \lambda, \quad \alpha$

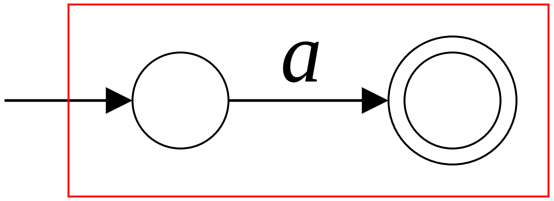
NFAs



$$L(M_1) = \varnothing = L(\varnothing)$$



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



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

regular languages

Inductive Hypothesis

Assume
for regular expressions r_1 and r_2
that
$L(r_1)$ and $L(r_2)$ are regular languages

Inductive Step

We will prove:

$$\left.\begin{array}{l} L(r_1 + r_2) \\ \\ L(r_1 \cdot r_2) \\ \\ L(r_1 *) \\ \\ L((r_1)) \end{array}\right\} \text{Are regular Languages}$$

By definition of regular expressions:

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

$$L(r_1 \cdot r_2) = L(r_1)\, L(r_2)$$

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

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

By inductive hypothesis we know:

$L(r_1)$ and $L(r_2)$ are regular languages

We also know:

Regular languages are closed under:

Union	$L(r_1) \cup L(r_2)$
Concatenation	$L(r_1)\, L(r_2)$
Star	$(L(r_1))^*$

Therefore:

$$\left.\begin{array}{l} L(r_1 + r_2) = L(r_1) \cup L(r_2) \\ \\ L(r_1 \cdot r_2) = L(r_1)\,L(r_2) \\ \\ L(r_1 *) = (L(r_1))* \end{array}\right\} \text{Are regular languages}$$

And trivially:

$L((r_1))$ is a regular language

Proof - Part 2

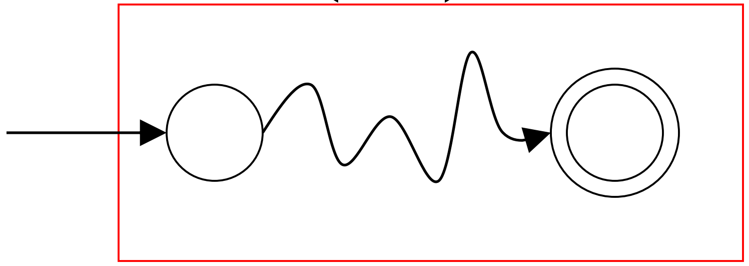
$$\left\{\begin{array}{l}\text{Languages}\\ \text{Generated by}\\ \text{Regular Expressions}\end{array}\right\} \supseteq \left\{\begin{array}{l}\text{Regular}\\ \text{Languages}\end{array}\right\}$$

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

Proof by construction of regular expression

Since L is regular take the NFA M that accepts it

$$L(M) = L$$



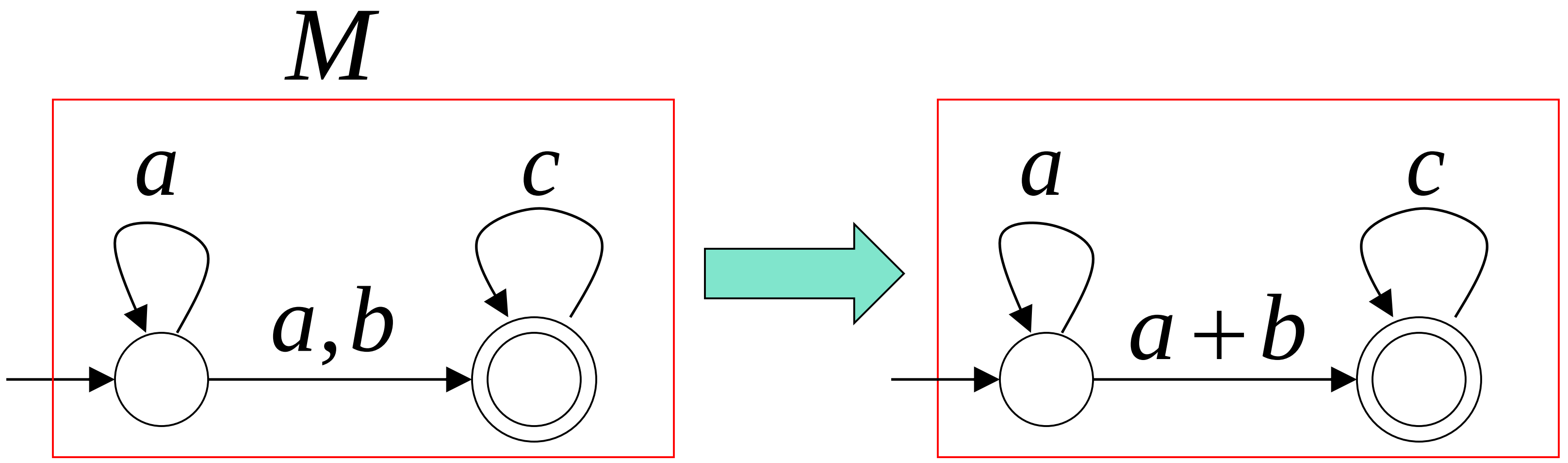
Single final state

From M construct the equivalent

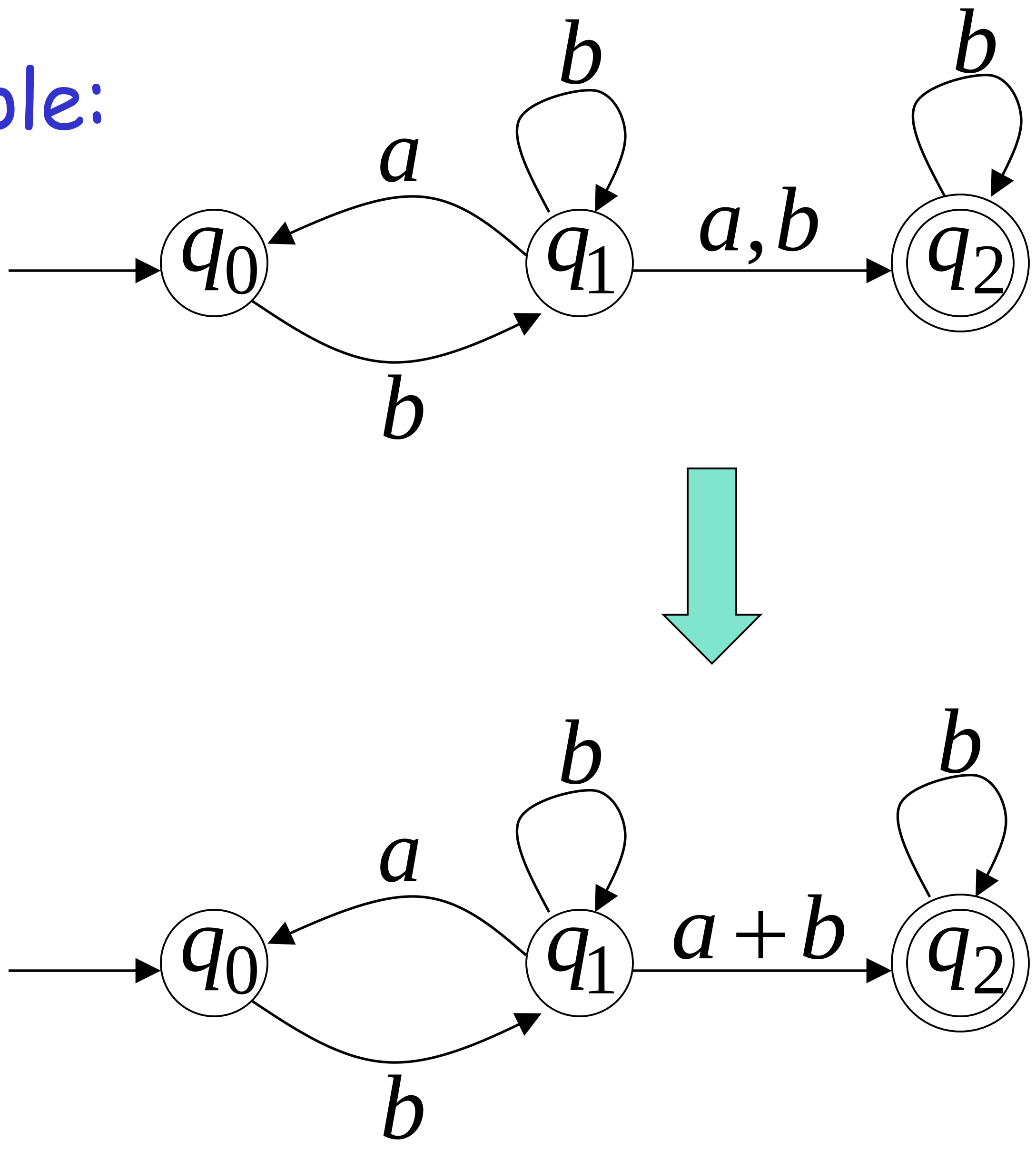
Generalized Transition Graph

in which transition labels are regular expressions

Example:



Another Example:



Reducing the states:

$$\begin{tikzcd}
{} \arrow[r] & q_0 \arrow[rr, bend right, "b"'] & & q_1 \arrow[ll, bend right, "a"'] \arrow[loop above, "b"] \arrow[rr, "a+b"] & & \boxed{q_2} \arrow[loop above, "b"]
\end{tikzcd}$$

$$\Downarrow$$

$$\begin{tikzcd}
{} \arrow[r] & q_0 \arrow[loop above, "bb^*a"] \arrow[rrr, "bb^*(a+b)"] & & & \boxed{q_2} \arrow[loop above, "b"]
\end{tikzcd}$$

Resulting Regular Expression:

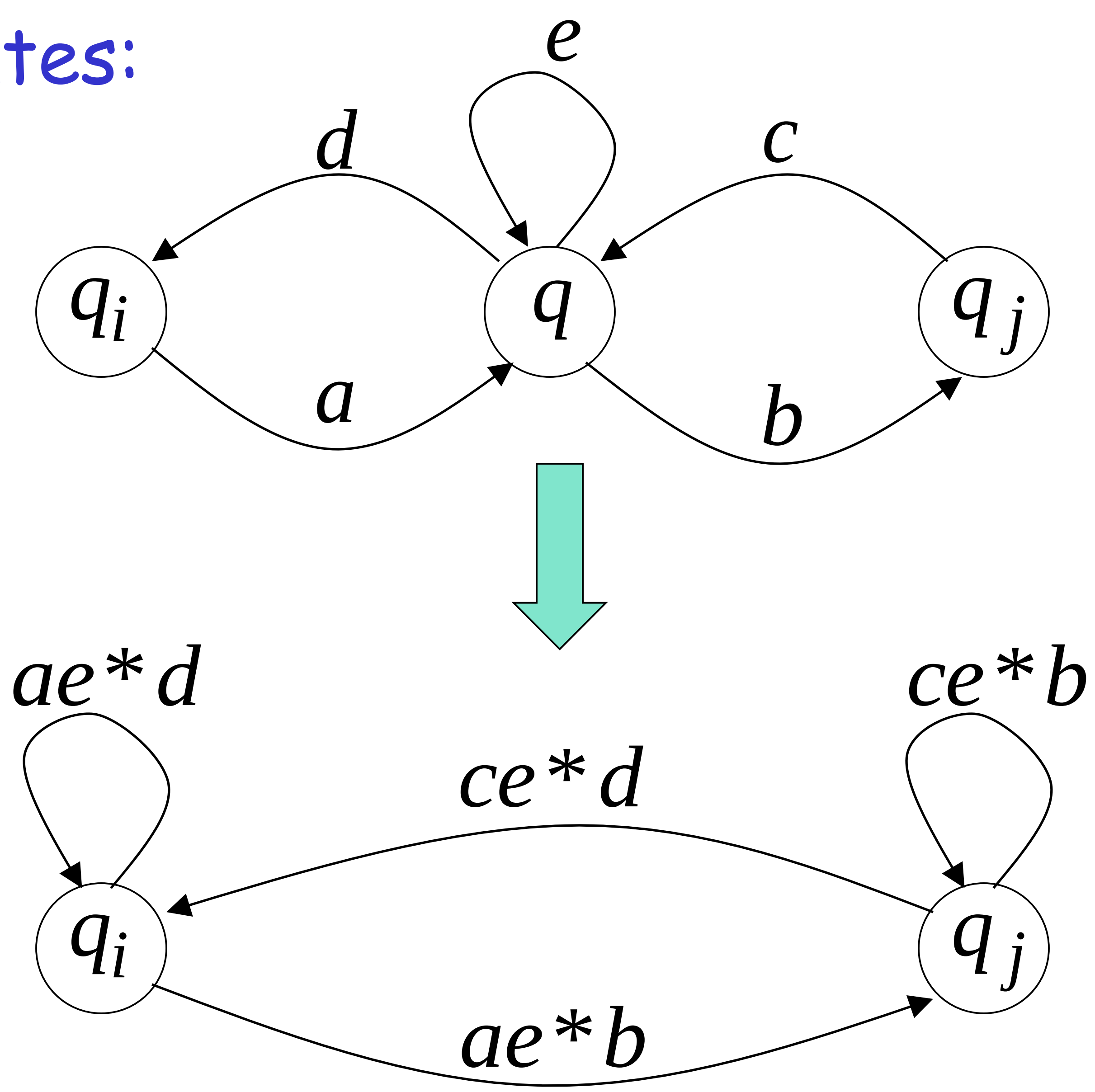
$$\begin{tikzcd}
\arrow[r] & q_0 \arrow[loop above, "bb^*a"] \arrow[rr, "bb^*(a+b)"] & & q_2 \arrow[loop above, "b"]
\end{tikzcd}$$

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

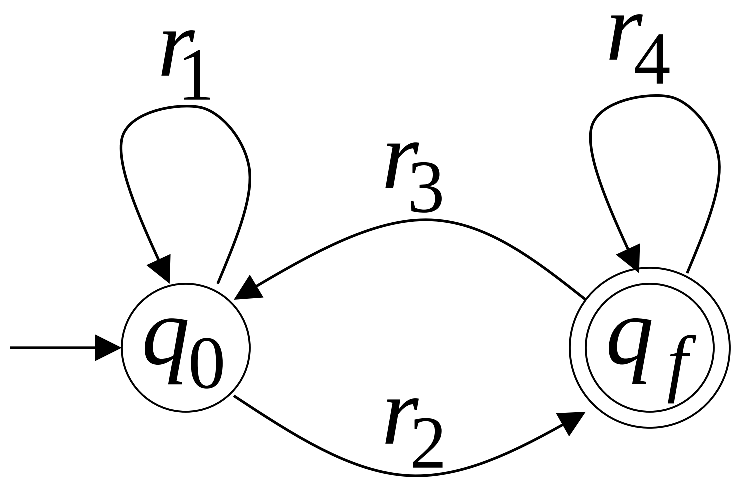
$$L(r) = L(M) = L$$

In General

Removing states:



The final transition graph:



The resulting regular expression:

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

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

When we say: We are given a Regular Language L

We mean: Language L is in a standard representation