Formal Languages
Regular Expressions

Regular Expressions

Regular expressions
describe regular languages

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

describes the language

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

Recursive Definition

Primitive regular expressions: $\emptyset, \quad \lambda, \quad \alpha$

Given regular expressions r_1 and r_2

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

Examples

A regular expression: $$(a+b\cdot c)^*\cdot(c+\emptyset)$$

Not a regular expression: $$(a+b+)$$

Languages of Regular Expressions

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

Example

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

Definition

For primitive regular expressions:

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

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

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

Definition (continued)

For regular expressions r_1 and r_2

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

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

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

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

Definitions?

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

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

$$\left(L(r_1)\right)^*$$

L(r)?

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

Example

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

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

L(r)?

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

Example

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

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

L(r)?

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

Example

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

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

L(r)?

$$(a+b\cdot c)^{*}\cdot(c+\varnothing)$$

Regular expression?

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

Example

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

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

Regular expression?

$L(r)$ = { all strings without 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 }

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

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

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

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

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

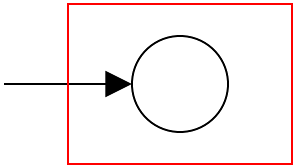
For any regular expression r
the language $L(r)$ is regular

Proof by induction on the size of r

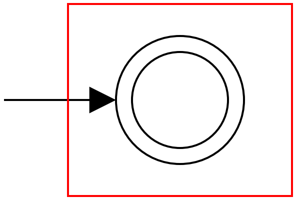
Induction Basis

Primitive Regular Expressions: $\emptyset$, λ, α

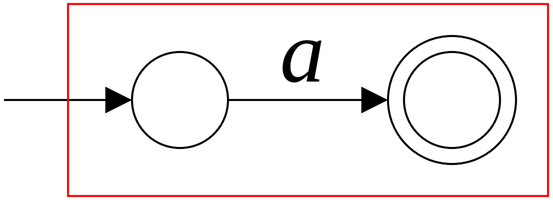
NFAs



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



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



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

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:

$$L(r_1 + r_2)$$

$$L(r_1 \cdot r_2)$$

$$L(r_1 *)$$

$$L((r_1))$$

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:

$$\left.\begin{aligned}
&L(r_1 + r_2) = L(r_1) \cup L(r_2) \\[2em]
&L(r_1 \cdot r_2) = L(r_1)\, L(r_2) \\[2em]
&L(r_1 *) = (L(r_1))*
\end{aligned}\right\}\ \text{Are regular languages}$$

And trivially:

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

Proof - Part 2

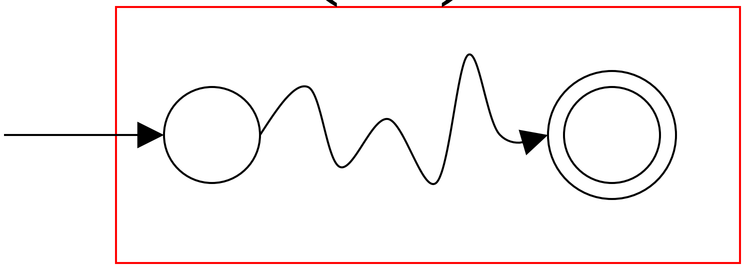
$$\left\{ \begin{array}{l} \text{Languages} \\ \text{Generated by} \\ \text{Regular Expressions} \end{array} \right\} \supseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

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

$$L(M) = L$$



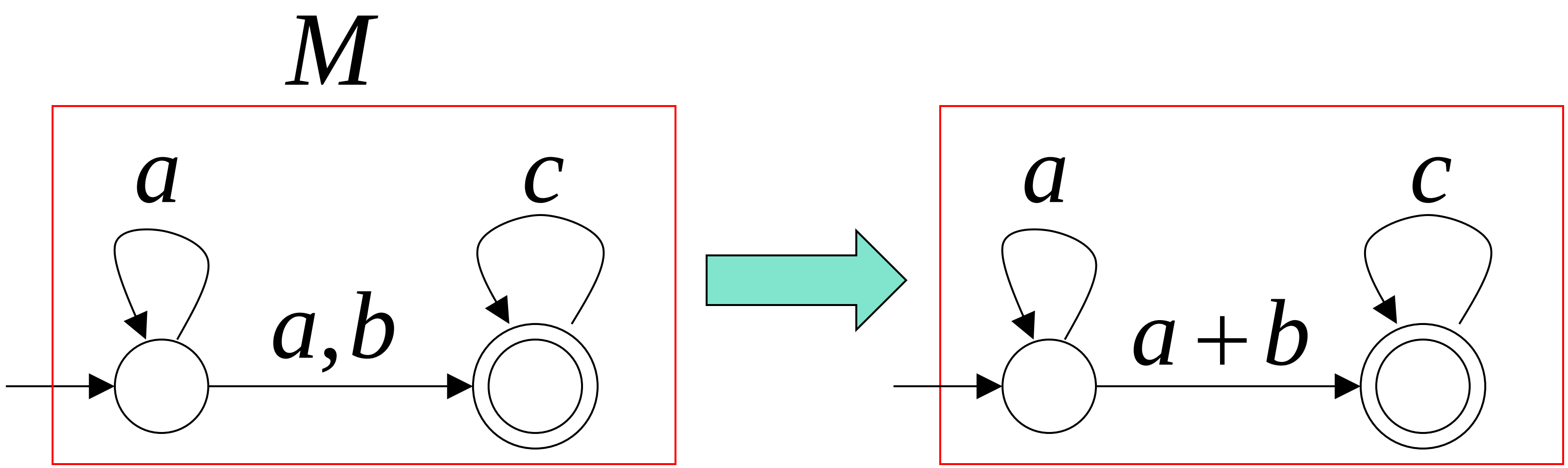
Single final state

From M construct the equivalent

Generalized Transition Graph

in which transition labels are regular expressions

Example:



Procedure nfa-rex

1. Start with an nfa with states q0,q1, ...qn and a single final state , distinct from its initial state

Procedure nfa-rex

1. Start with an nfa with states q0,q1, ...qn and a single final state , distinct from its initial state
2. Convert the nfa into a complete generalized transition graph.

   Let r_{ij} stand for the label of the edge from qi to qj

Procedure nfa-rex

1. .
2.

3. If the generalized transition graph(GTG) has only 2 states with qi as initial and qj as final, as its associated regular expression is

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

4. If GTG has 3 states with the initial state qi and final state qj and the third state qk, introduce new edge labelled

$r_{pq} + r_{pk} r_{kk}^{*} r_{kq}$ for p =i, j and q=i, j. When this is done the remove the vertex q_k and its associated edges.

5 . If GTG has 4 or more edges , pick a state q_k to be removed. Apply rule 4 for all pairs of states (q_i ,q_j). $i \neq k$, $j \neq k$. At each step apply the simplifying rules

$r+\Phi = r$

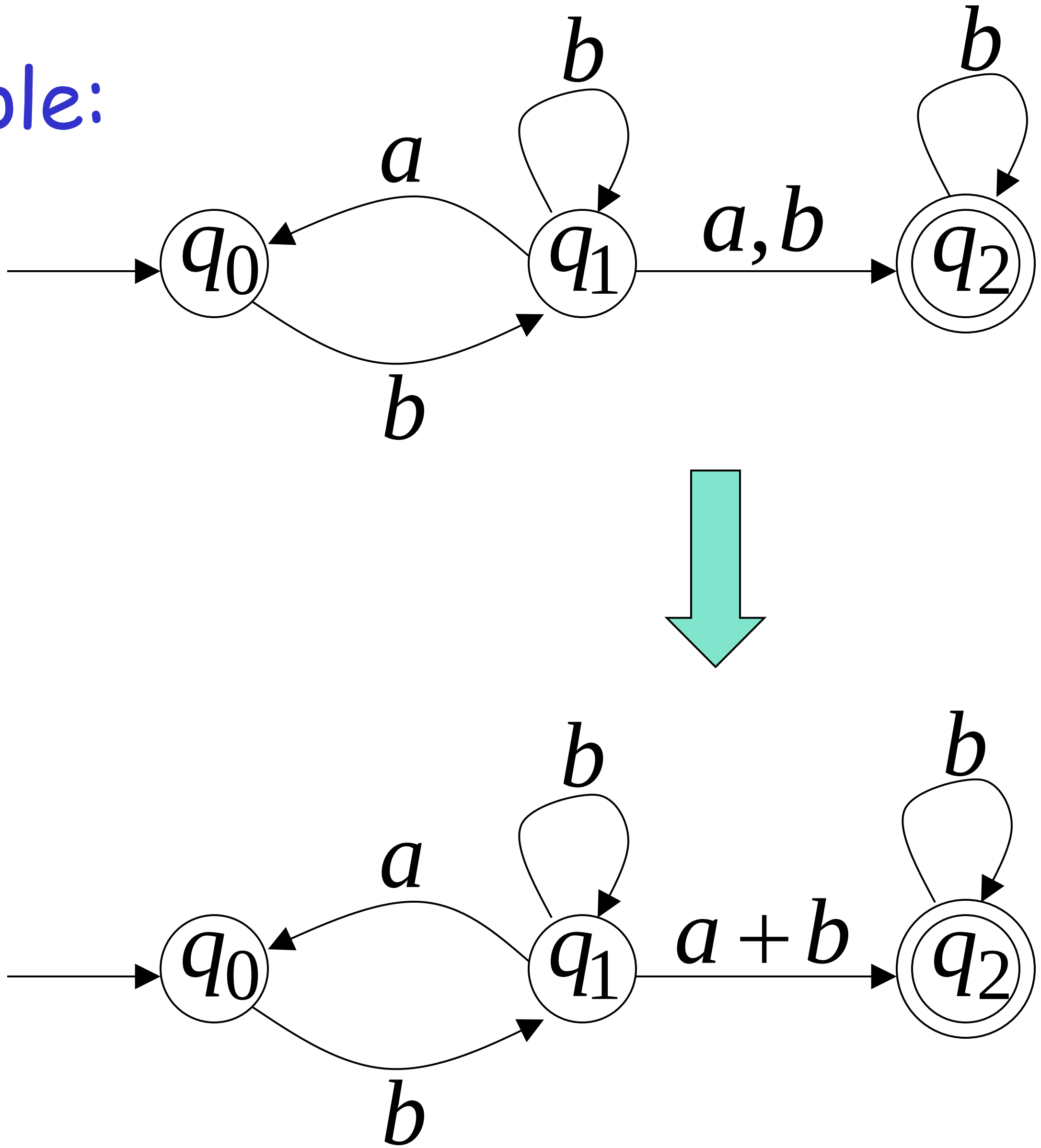
$r \Phi = \Phi$

$\Phi^* = \Lambda$

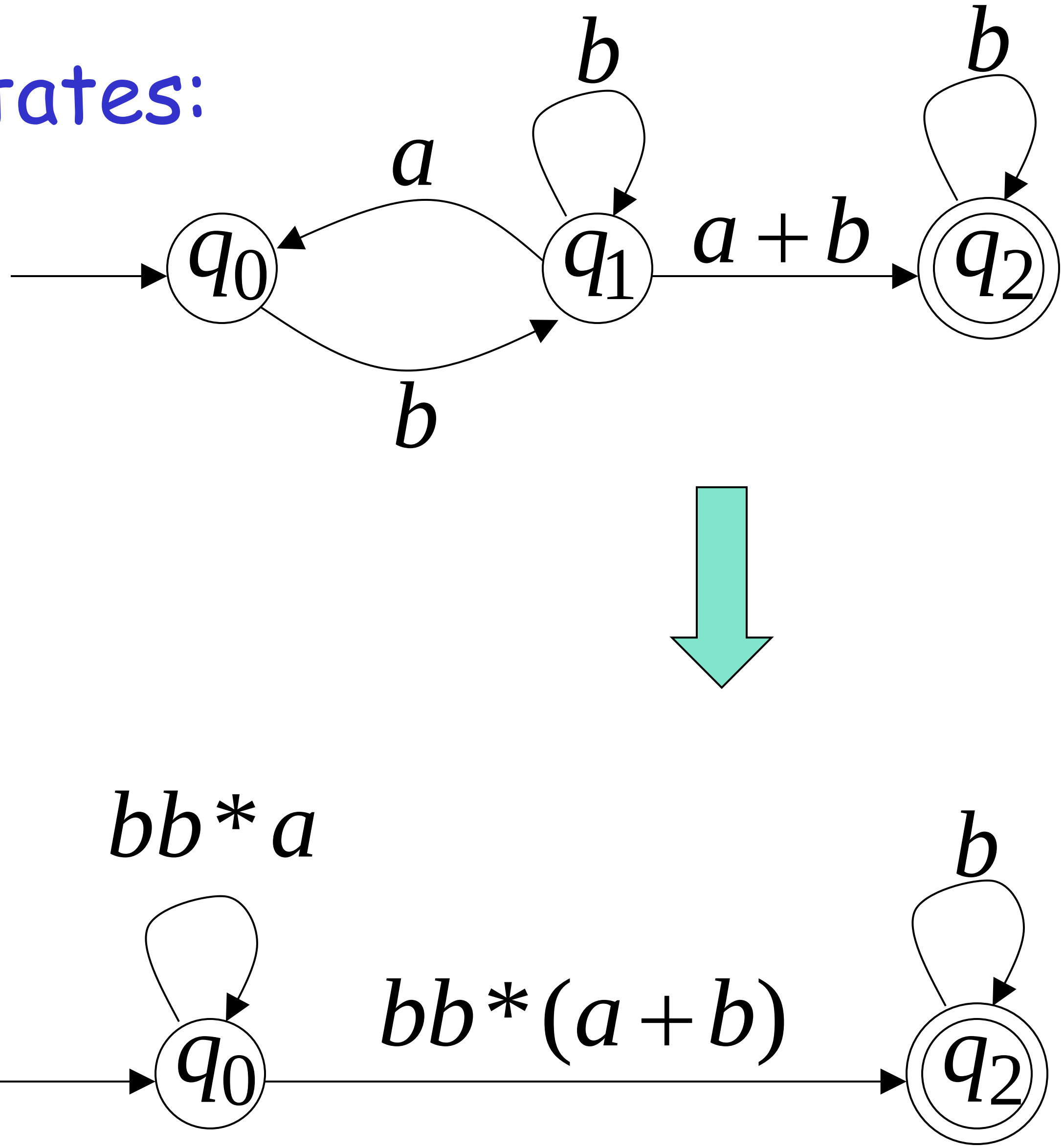
Wherever possible. When this is done , remove q_k

6. Repeat step 3 to 5 until the correct regular expression is obtained.

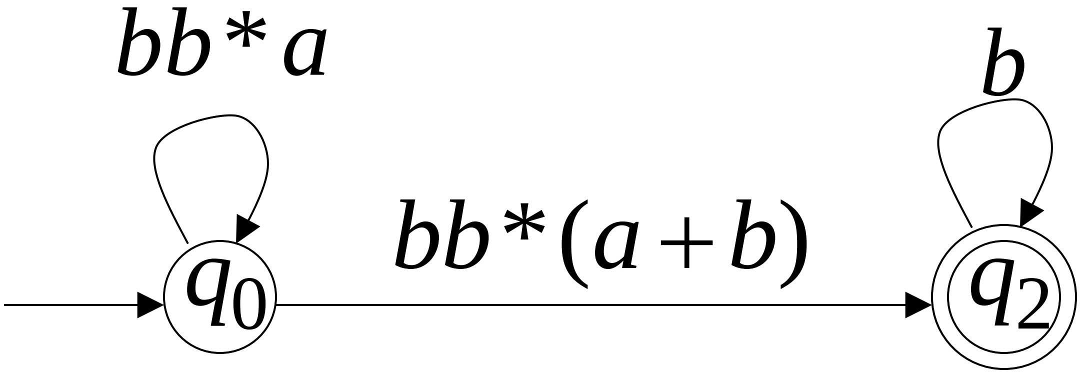
Another Example:



Reducing the states:



Resulting Regular Expression:

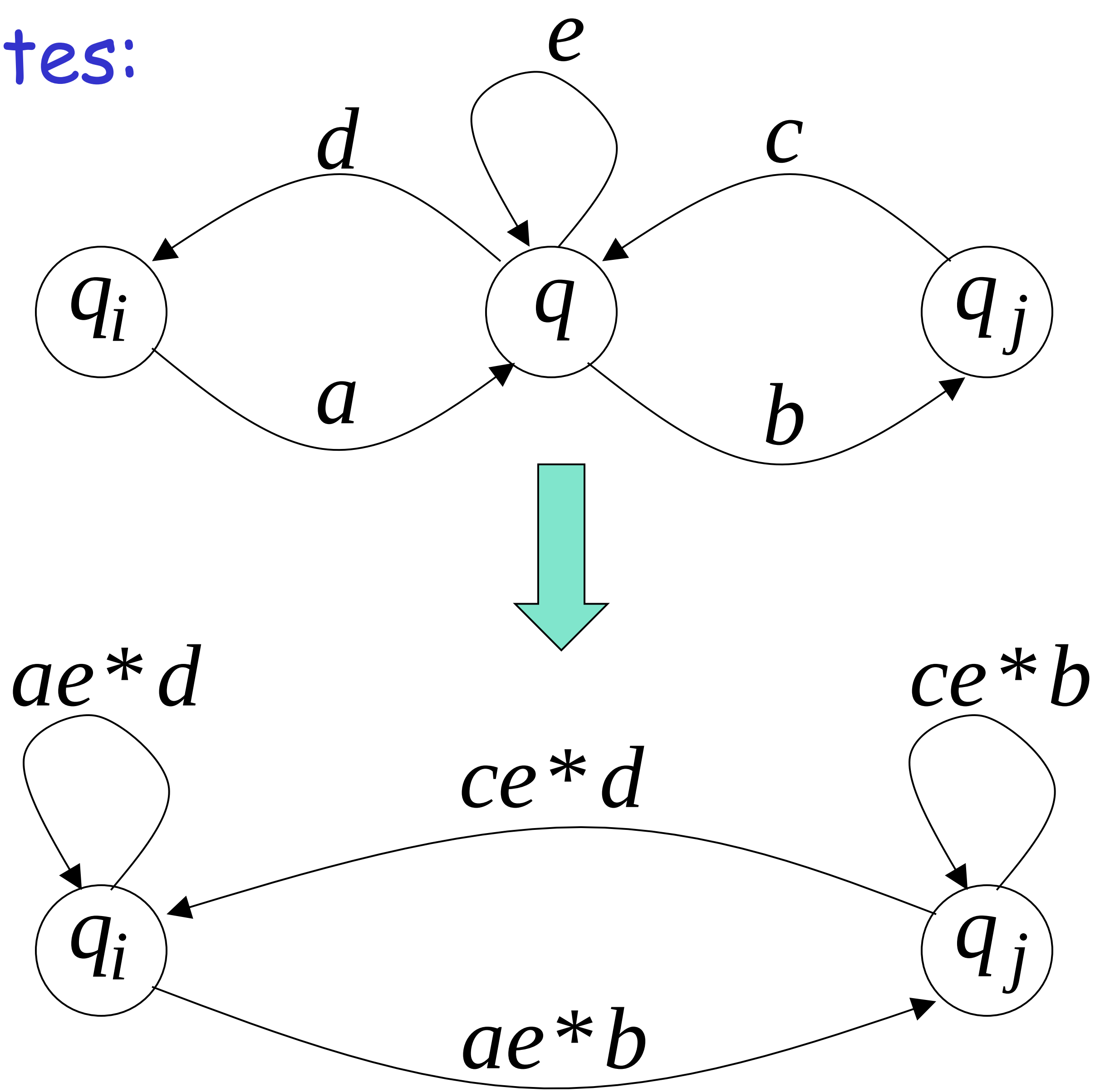


$$r = (bb^*a)^*bb^*(a+b)b^*$$

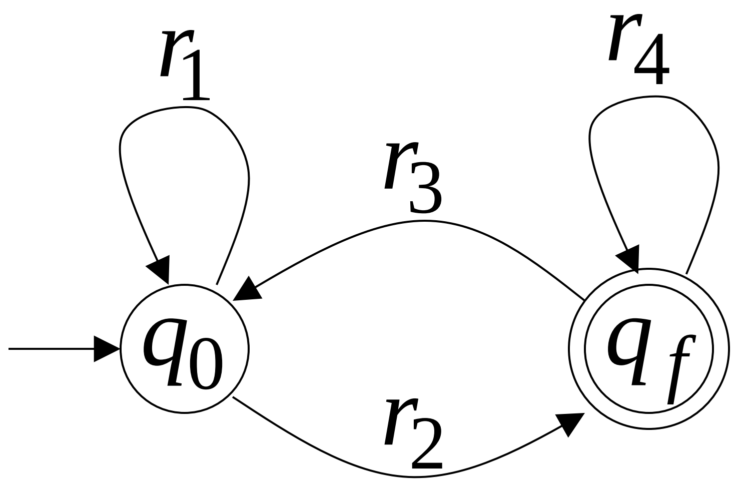
$$L(r) = L(M) = L$$

In General

Removing states:



The final transition graph:

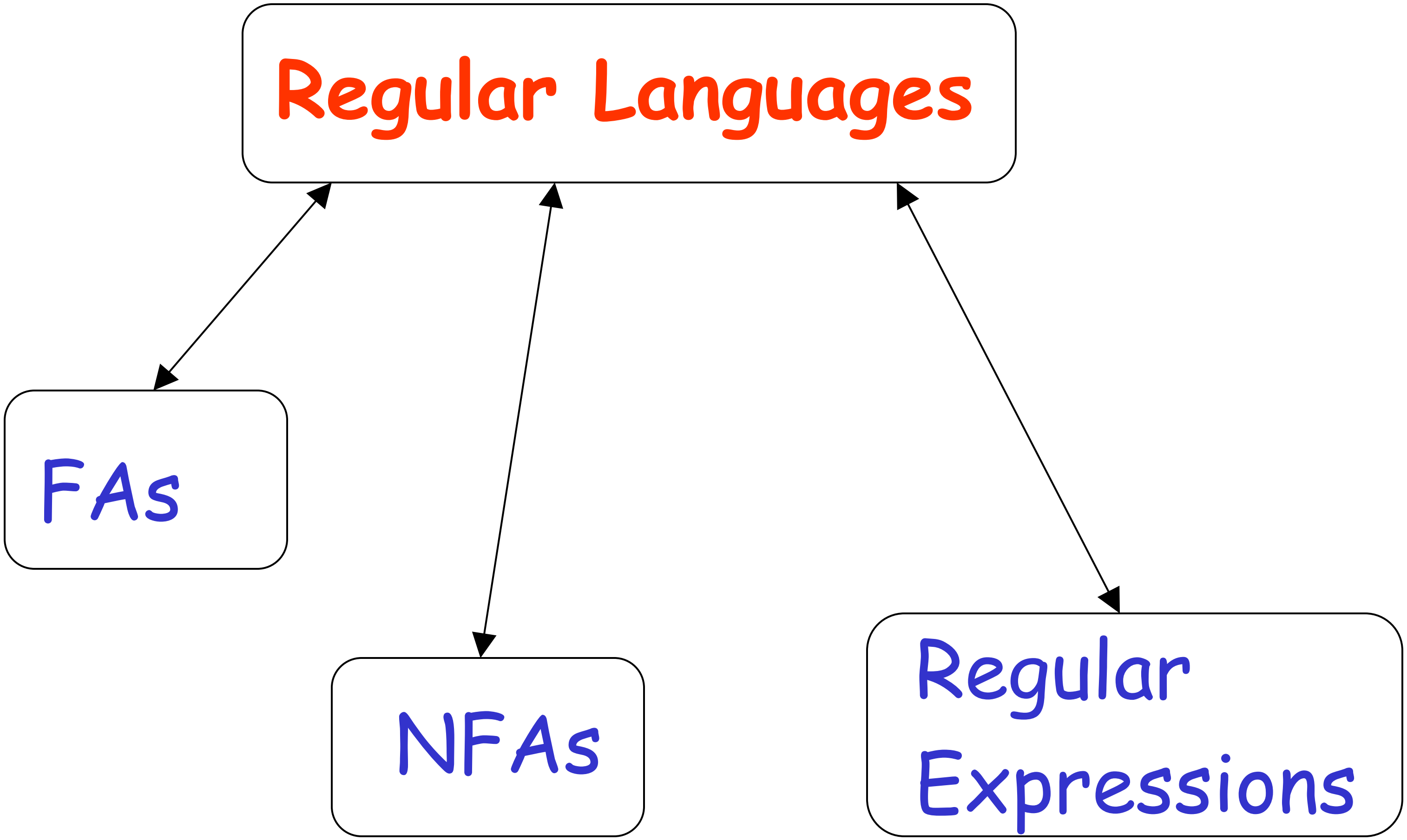


The resulting regular expression:

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

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

Standard Representations of Regular Languages



When we say: We are given a Regular Language L

We mean: Language L is in a standard representation

Elementary Questions

about

Regular Languages

Membership Question

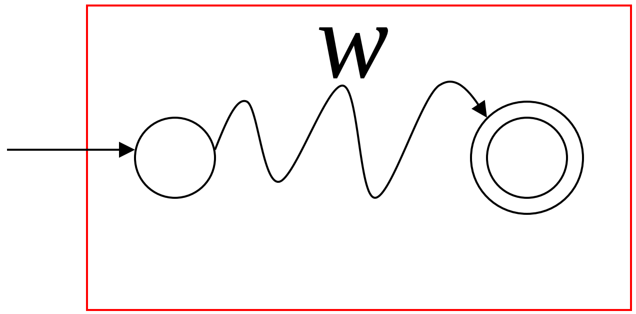
Question: Given regular language L
and string w
how can we check if $w \in L$?

Membership Question

Question: Given regular language L and string w how can we check if $w \in L$?

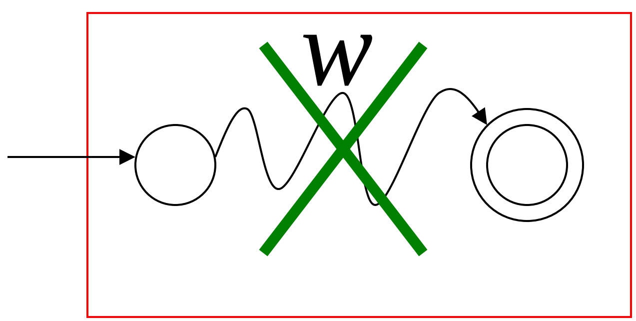
Answer: Take the DFA that accepts L and check if w is accepted

DFA



$w \in L$

DFA



$w \notin L$

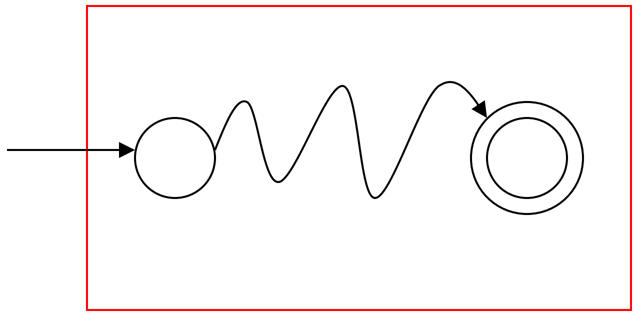
Question: Given regular language L how can we check if L is empty: $(L = \emptyset)$?

Question: Given regular language L how can we check if L is empty: $(L = \emptyset)$?

Answer: Take the DFA that accepts L

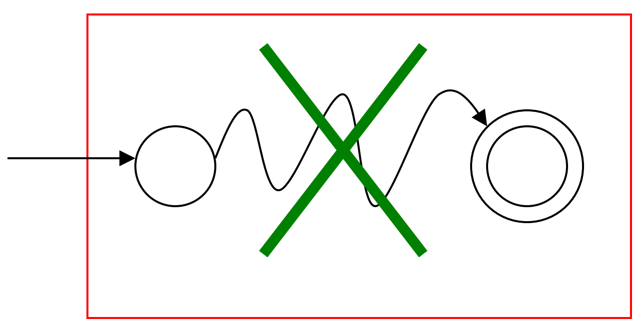
Check if there is any path from the initial state to a final state

DFA



$$L \neq \varnothing$$

DFA



$$L = \varnothing$$

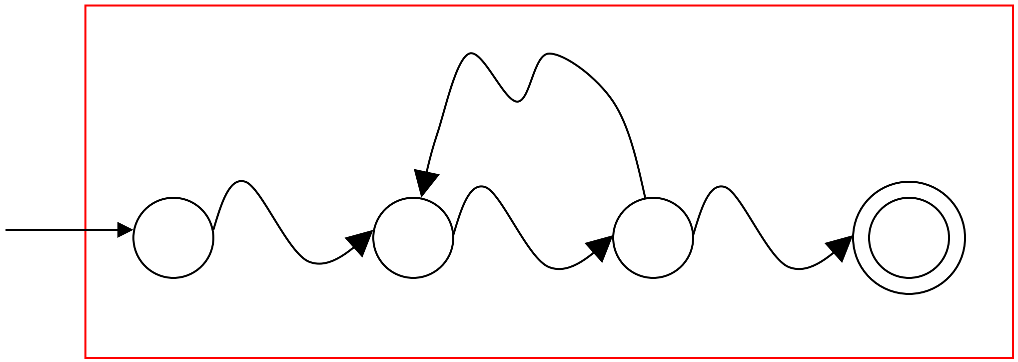
Question: Given regular language L
how can we check
if L is finite?

Question: Given regular language L
how can we check
if L is finite?

Answer: Take the DFA that accepts L

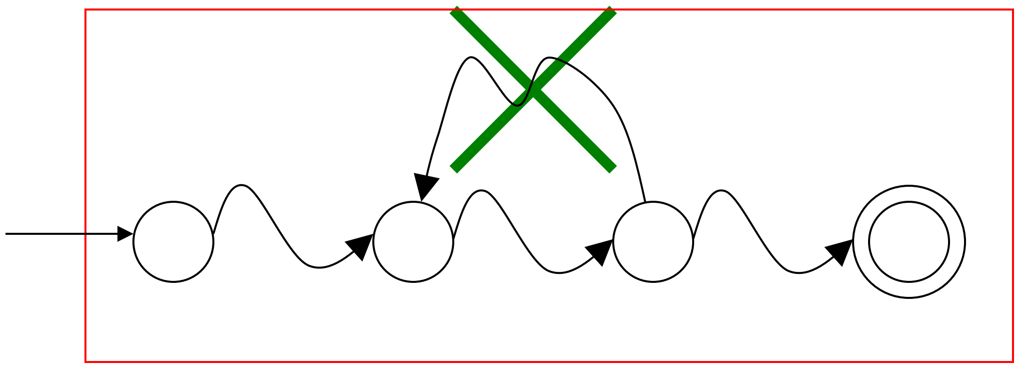
Check if there is a walk with cycle
from the initial state to a final state

DFA



L is infinite

DFA



L is finite

Question: Given regular languages L_1 and L_2 how can we check if $L_1 = L_2$?

Question: Given regular languages L_1 and L_2 how can we check if $L_1 = L_2$?

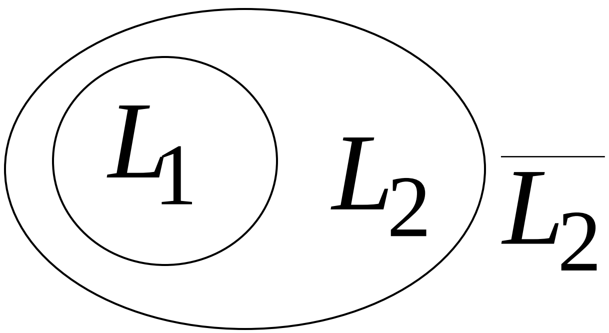
Answer: Find if $(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \emptyset$

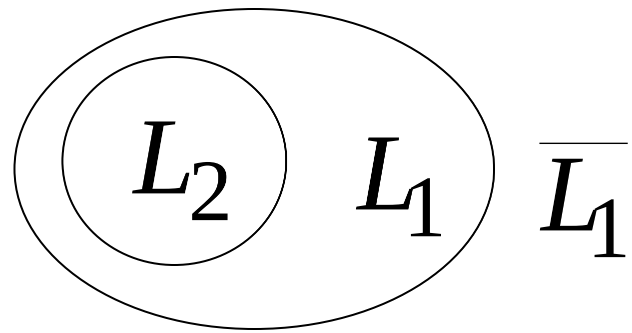
$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \emptyset$$

$$L_1 \cap \overline{L_2} = \emptyset$$

and

$$\overline{L_1} \cap L_2 = \emptyset$$





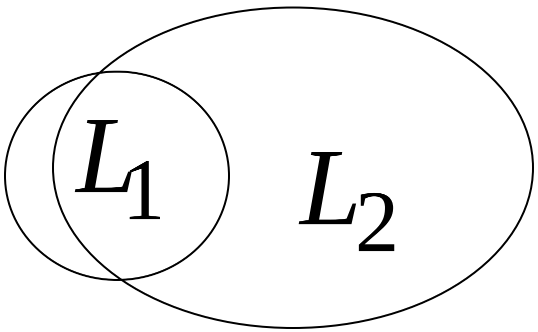
$$L_1 \subseteq L_2$$

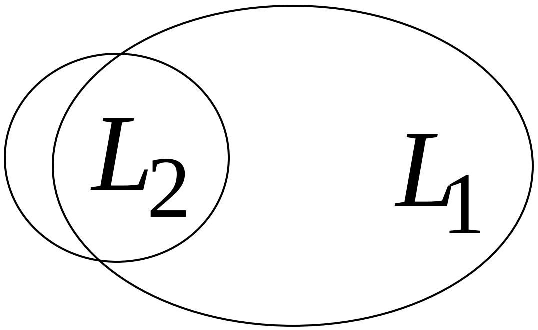
$$L_2 \subseteq L_1$$

$$L_1 = L_2$$

$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) \neq \emptyset$$

$$L_1 \cap \overline{L_2} \neq \emptyset \quad \text{or} \quad \overline{L_1} \cap L_2 \neq \emptyset$$





$$L_1 \not\subset L_2 \qquad L_2 \not\subset L_1$$

$$L_1 \neq L_2$$