Regular Expressions

Regular Expressions

recognizes → tanımlar mı
↓
accept?

Regular expressions
describe regular languages

aabc bcaa
abc bca

a bcbcbc

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

a bc bc a a bc bc

describes the language

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

Recursive Definition

Primitive regular expressions: $\overset{\text{empty set}}{\emptyset}, \quad \overset{\text{empty string}}{\varepsilon}, \quad \alpha$

Given regular expressions r_1 and r_2

Operator precedence

$*$ → higest

$\cdot$

$+$

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

Examples

A regular expression: $(a+b\cdot c)^* \cdot (c+\underset{d}{\overset{\varepsilon}{\emptyset}})$

Not a regular expression: $\cancel{(a+b+)}$

Languages of Regular Expressions

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

Example

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

Definition

For primitive regular expressions:

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

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

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

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

ε

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

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

Example

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

abababaabb aba a bb

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

Example

Regular expression $$r = (\xcancel{aa})^* (bb)^* b$$

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

Example

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

$L(r)$ = { all strings containing substring 00 }

Example

Regular expression $r = (1+01)^*(0+\cancel{\varepsilon})$

[illegible]

$L(r)$ = { all strings without substring 00 }

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 substring 00 }

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

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

$L(r_1) = L(r_2) = L$ ⇒ r_1 and r_2 are equivalent regular expressions

Regular Expressions and Regular Languages

Theorem

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

Proof:

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

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

Proof - Part 1

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

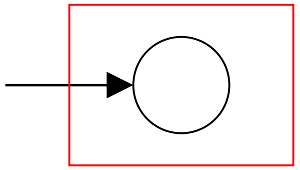
For any regular expression r
the language $L(r)$ is regular

Proof by induction on the size of r

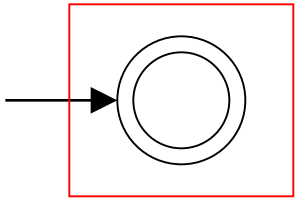
Induction Basis

Primitive Regular Expressions: $\emptyset, \quad \varepsilon, \quad \alpha$

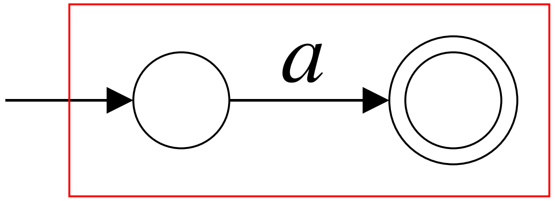
Corresponding NFAs



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



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



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

regular languages

Inductive Hypothesis

Suppose

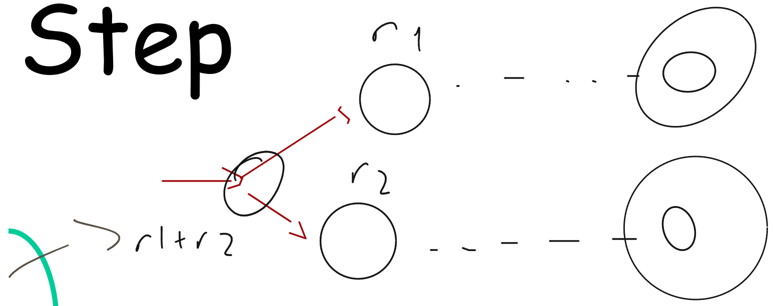
that for regular expressions r_1 and r_2,

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

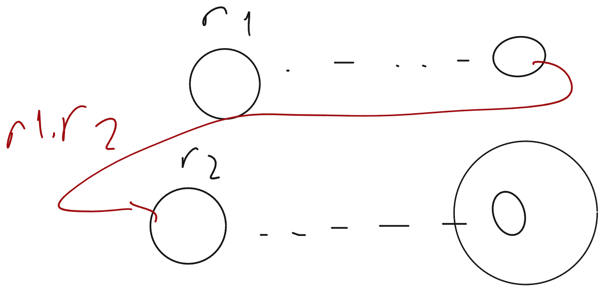
Inductive Step

We will prove:

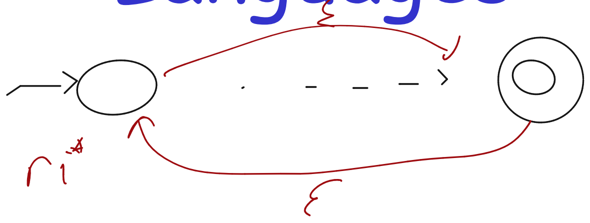
$$L(r_1 + r_2)$$



$$L(r_1 \cdot r_2)$$



$$L(r_1^*)$$



$$L((r_1)) \; = L(r_1)$$

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:

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

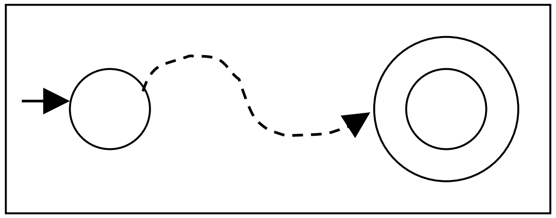
$L((r_1)) = L(r_1)$ is trivially a regular language (by induction hypothesis)

End of Proof-Part 1

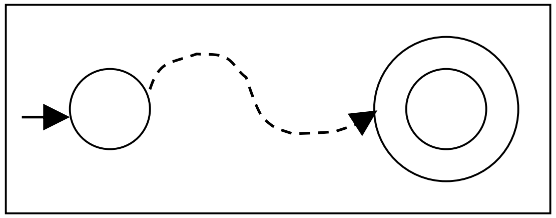
Using the regular closure of operations, we can construct recursively the NFA M that accepts $L(M) = L(r)$

Example: $r = r_1 + r_2$

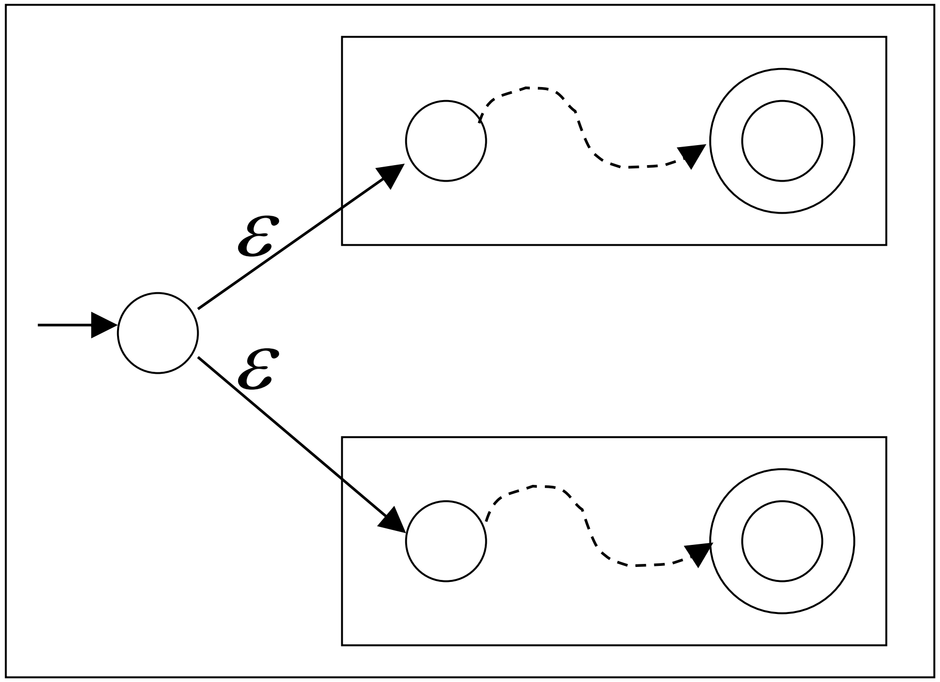
$L(M_1) = L(r_1)$



$L(M_2) = L(r_2)$



$L(M) = L(r)$



Proof - Part 2

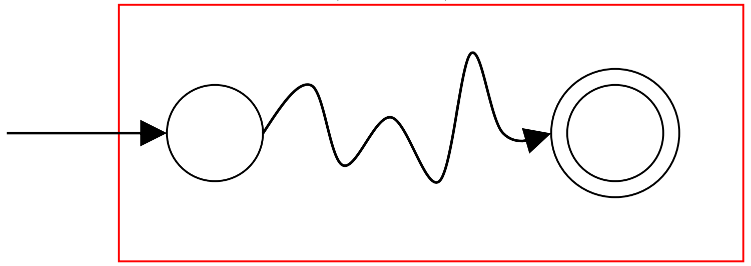
$$\left\{\begin{array}{l}\text{Languages}\\ \text{Generated by}\\ \text{Regular Expressions}\end{array}\right\} \supseteq \left\{\begin{array}{l}\text{Regular}\\ \text{Languages}\end{array}\right\}$$

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

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

Since L is regular, there is a NFA M that accepts it

$$L(M) = L$$



Take it with a single accept state

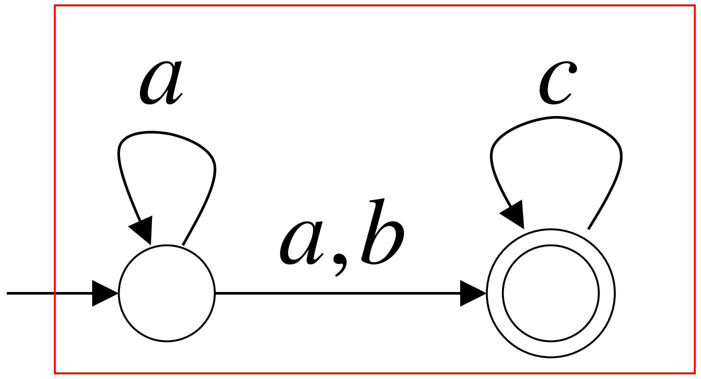
From M construct the equivalent

Generalized Transition Graph

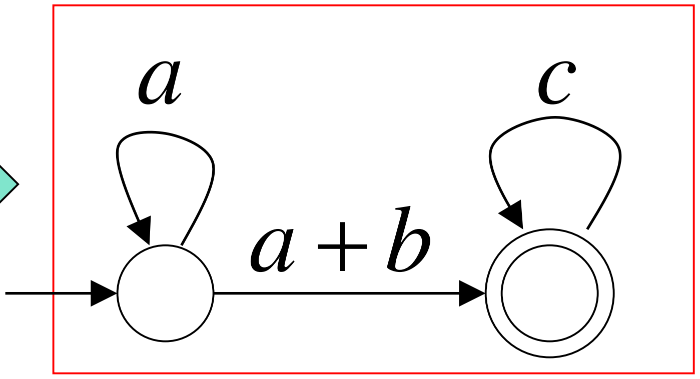
in which transition labels are regular expressions

Example:

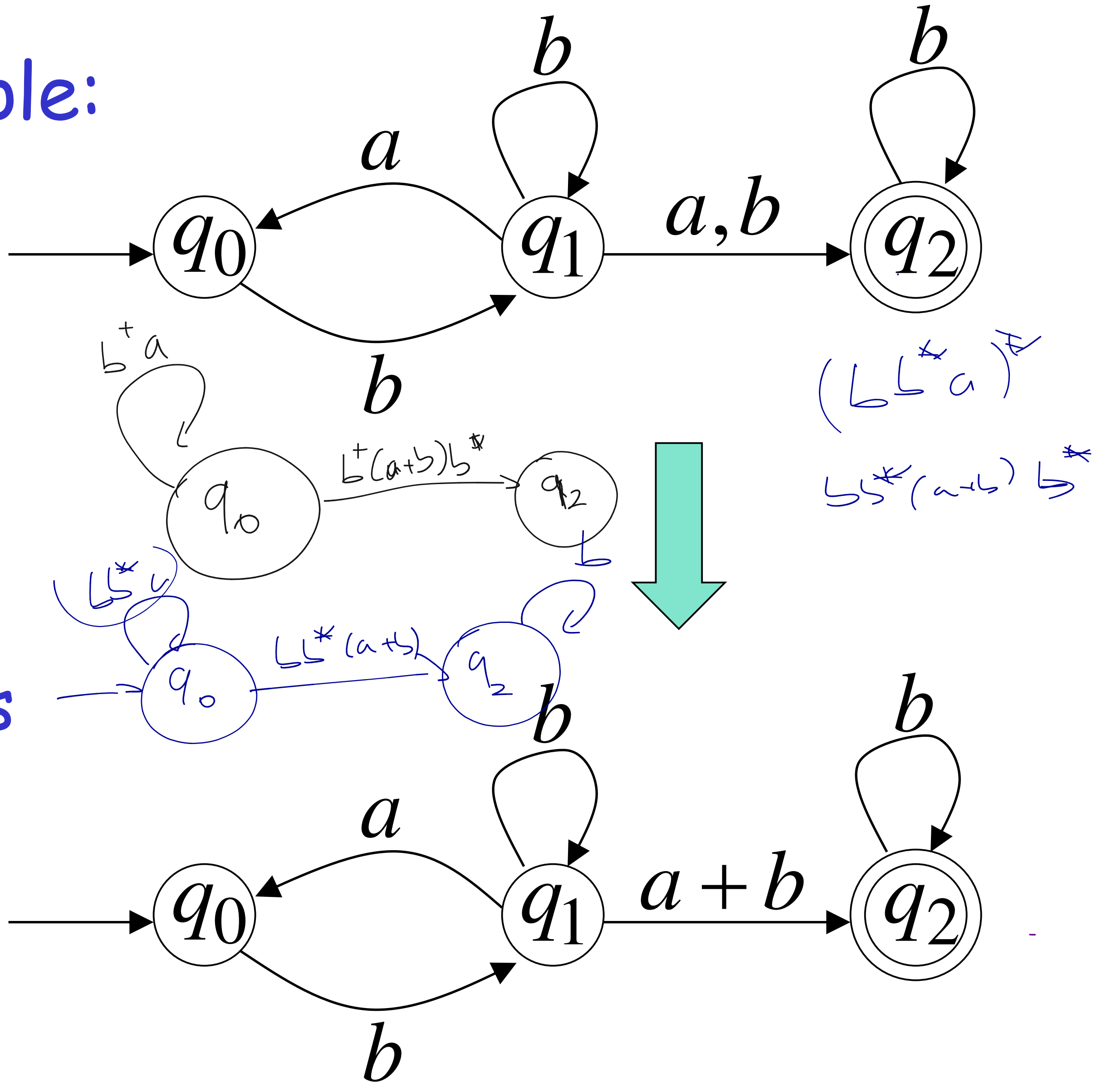
M



Corresponding
Generalized transition graph

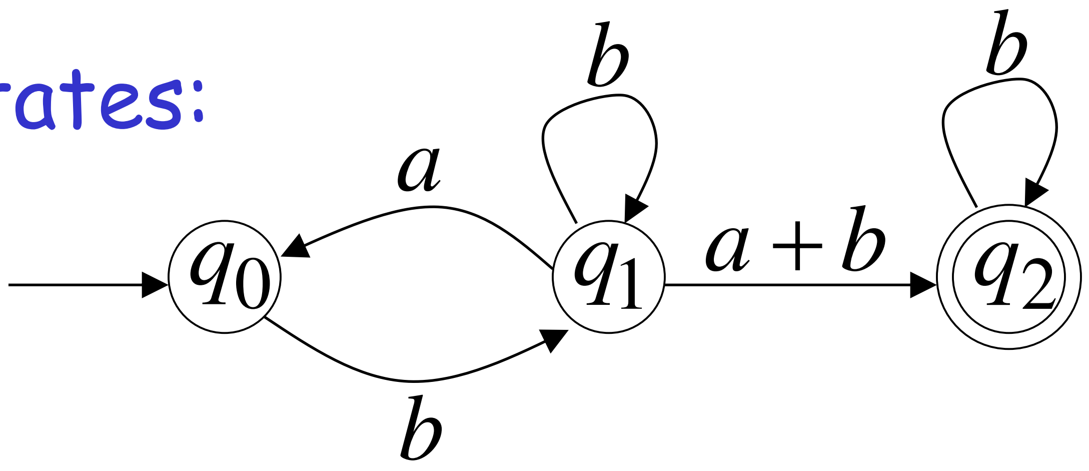


Another Example:



Transition labels are regular expressions

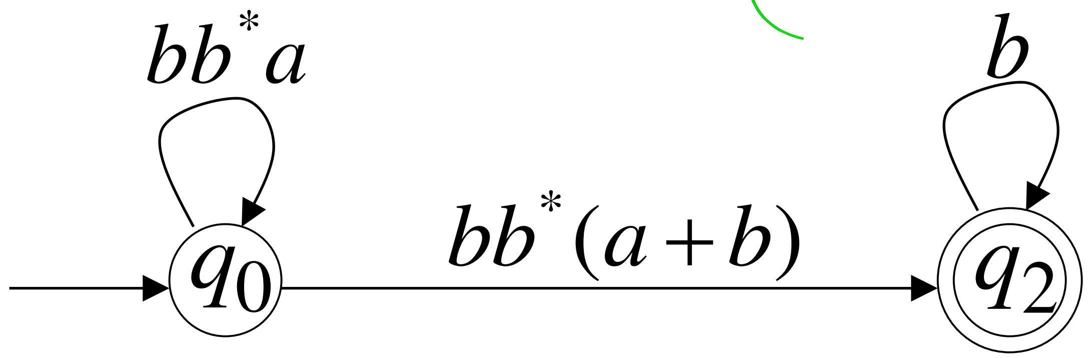
Reducing the states:



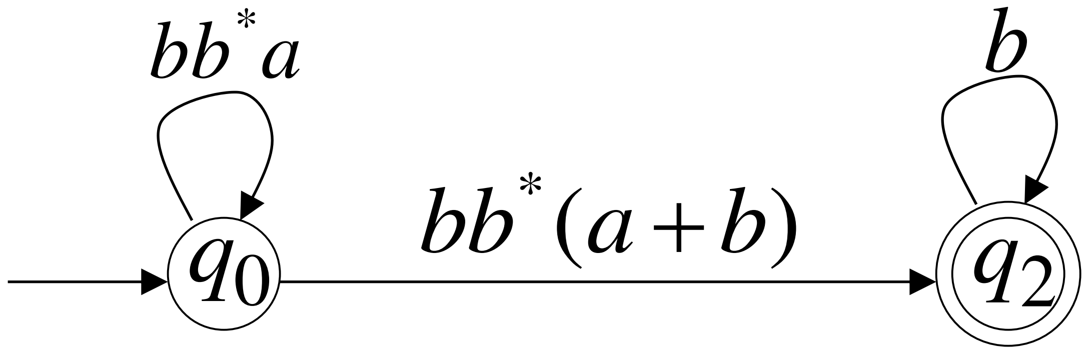
$(bb^*a)^* bb^*(a+b) b^*$

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

Transition labels are regular expressions



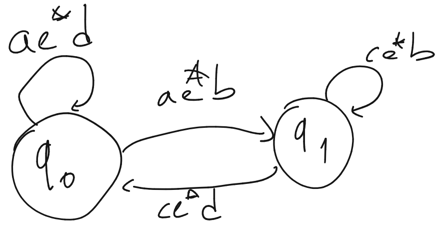
Resulting Regular Expression:


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

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

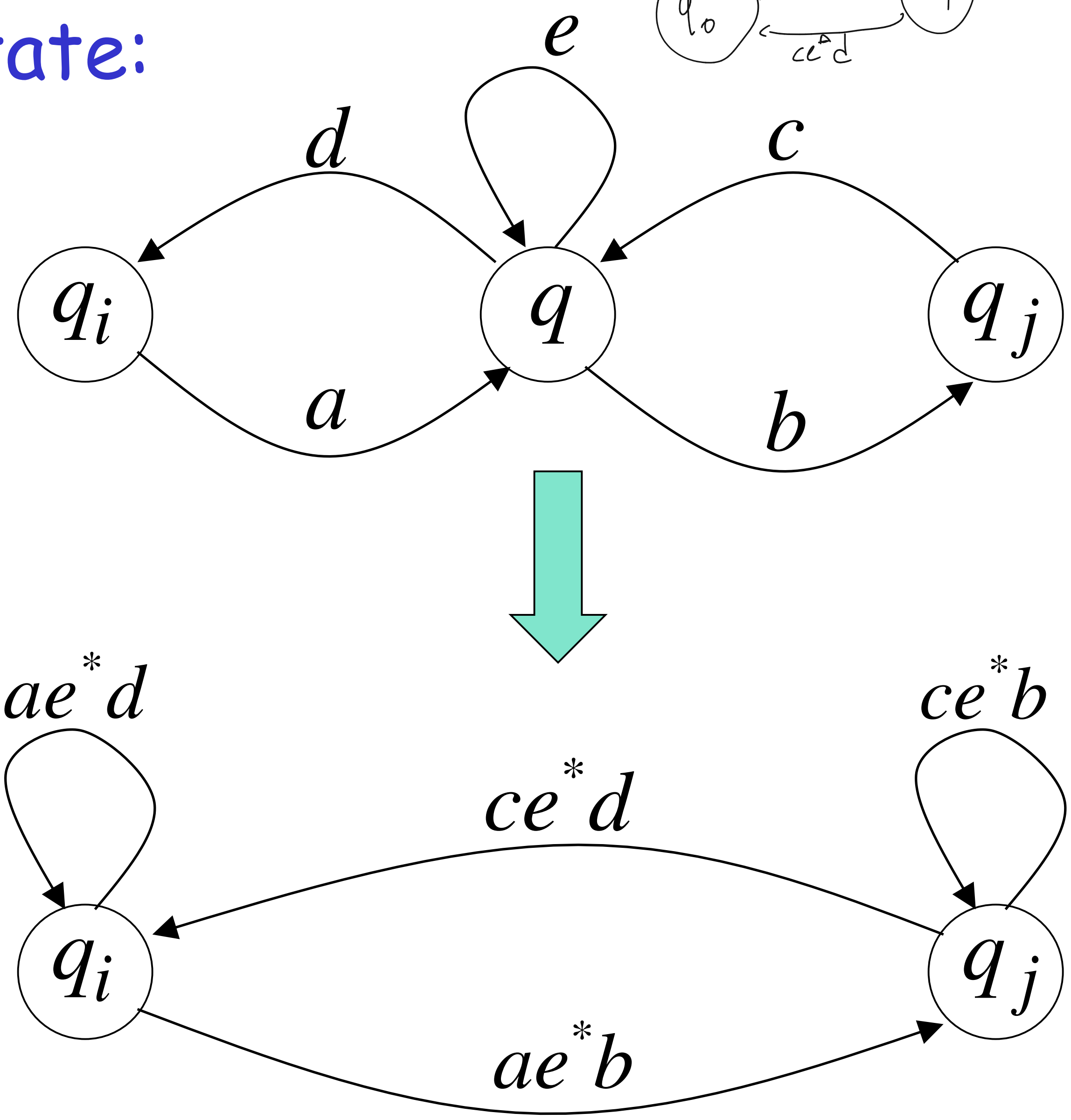
$$L(r) = L(M) = L$$

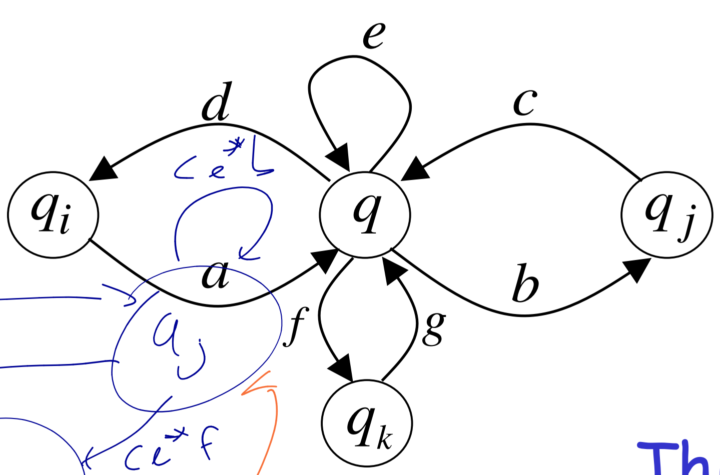
In General

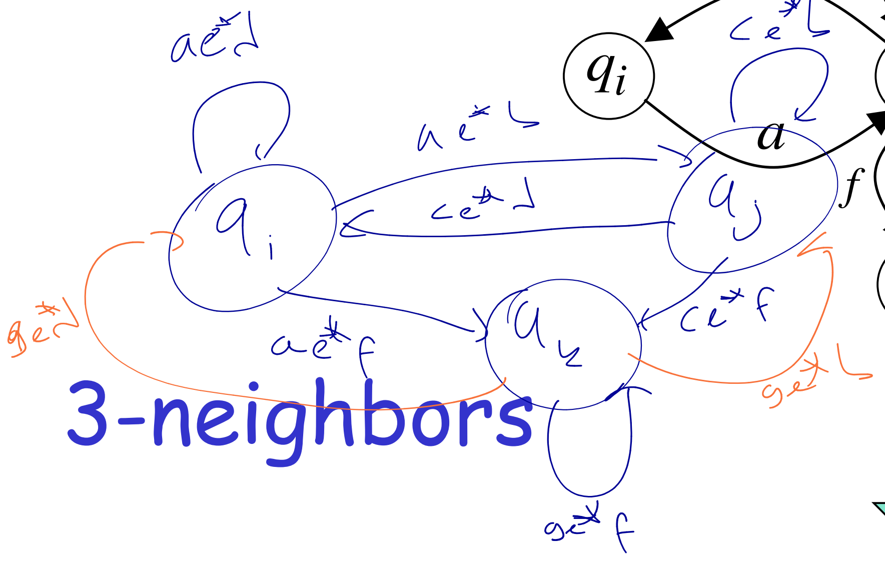


Removing a state:

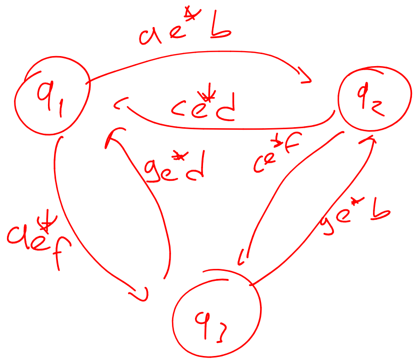
2-neighbors



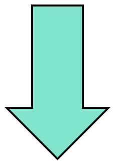


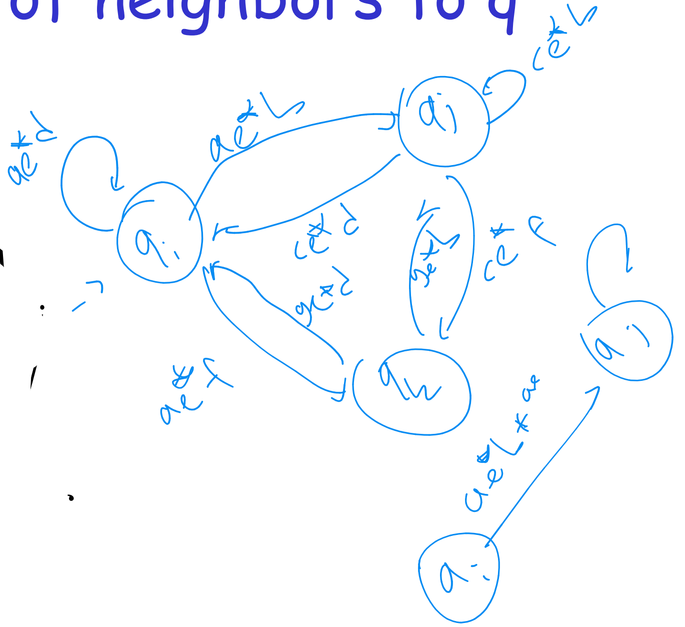


3-neighbors



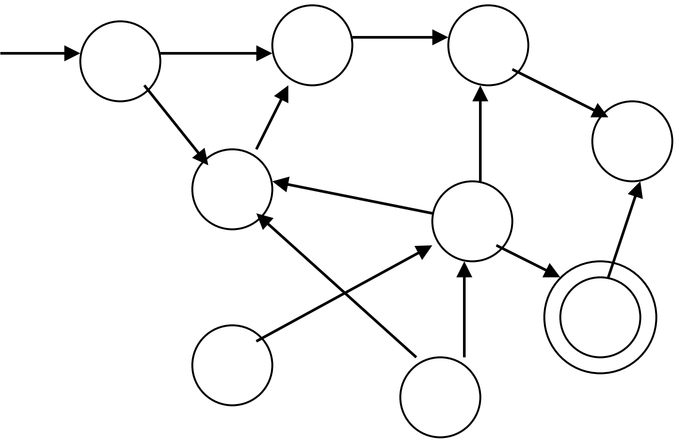
This can be generalized to arbitrary number of neighbors to q

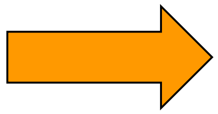




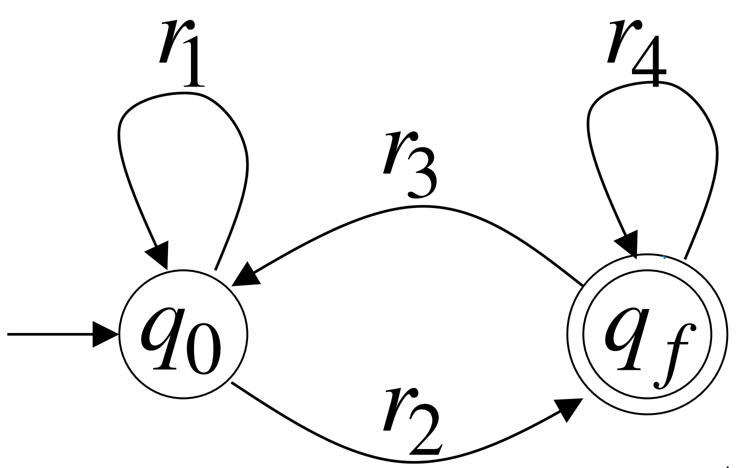
By repeating the process until two states are left, the resulting graph is

Initial graph





Resulting graph



The resulting regular expression:

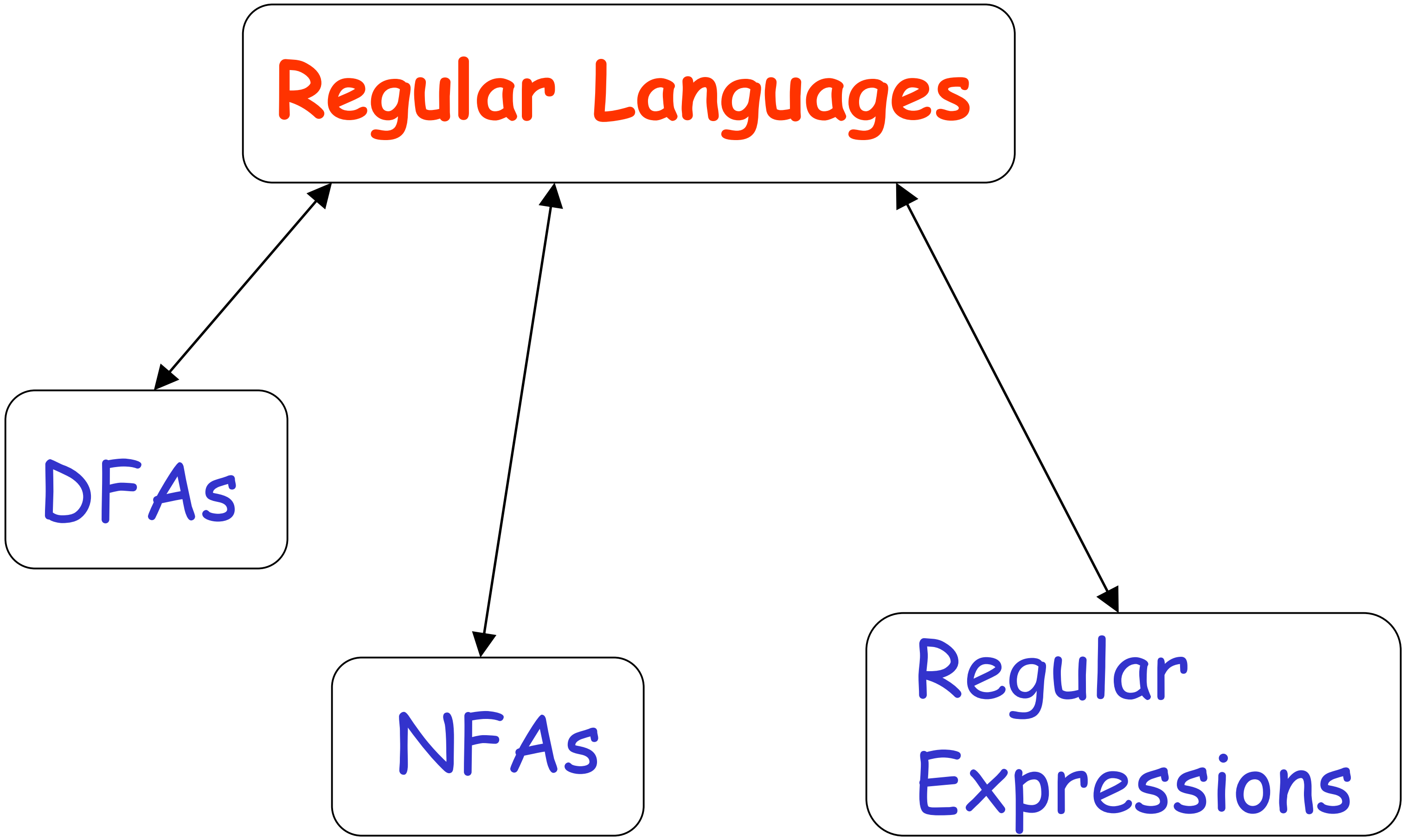
$r_1^* r_2 . (r_4^* + r_3 r_1^* r_2)^*$

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

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

End of Proof-Part 2

Standard Representations of Regular Languages



When we say: We are given a Regular Language L

[illegible]

1

0 0 0
0 1 0
1 0 0

We mean: Language L is in a standard representation

(DFA, NFA, or Regular Expression)