Automate,

Calculabilitate,

Complexitate

Properties of Regular Languages

For regular languages L_1 and L_2 we will prove that:

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: $L_1 *$

Reversal: L_1^R

Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

Are regular Languages

We say: Regular languages are **closed under**

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: $L_1 *$

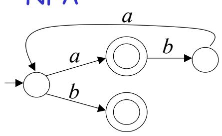
Reversal: L_1^R

Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

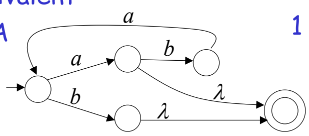
A useful transformation: use one accept state

NFA



2 accept states

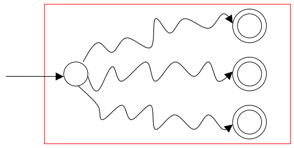
Equivalent NFA



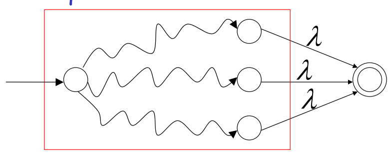
1 accept state

In General

NFA



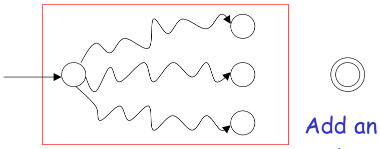
Equivalent NFA



Single accepting state

Extreme case

NFA without accepting state



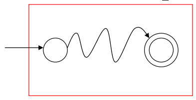
Add an accepting state without transitions

Take two languages

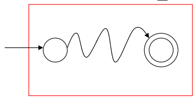
Regular language L_1 Regular language L_2

$L(M_1) = L_1$ $L(M_2) = L_2$

NFA M_1



NFA M_2



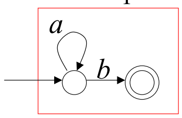
Single accepting state Single accepting state

Example

$n \geq 0$

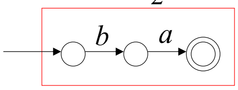
$L_1 = \{a^n b\}$

M_1



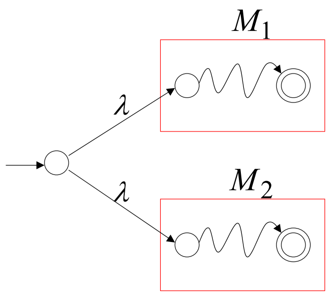
$L_2 = \{ba\}$

M_2



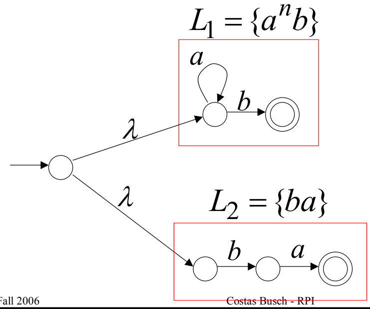
Union

NFA for $L_1 \cup L_2$



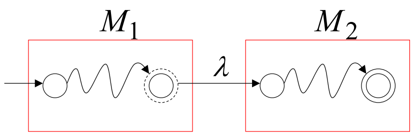
Example

NFA for $L_1 \cup L_2 = \{a^n b\} \cup \{ba\}$



Concatenation

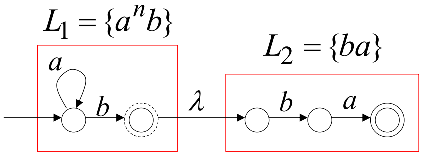
NFA for $L_1 L_2$



2

Example

NFA for $L_1 L_2 = \{a^n b\}\{ba\} = \{a^n bba\}$



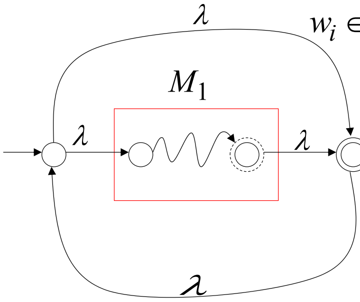
Star Operation

NFA for $L_1 *$

$w = w_1 w_2 \cdots w_k$

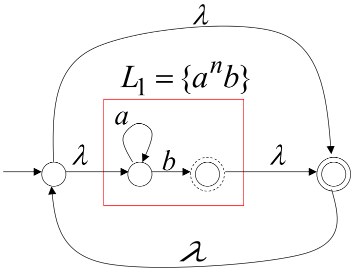
$w_i \in L_1$

$\lambda \in L_1 *$



Example

NFA for $L_1 * = \{a^n b\} *$



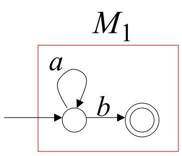
Reverse

NFA for L_1^R

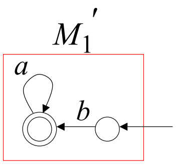
1. Reverse all transitions
2. Make initial state accepting state and vice versa

Example

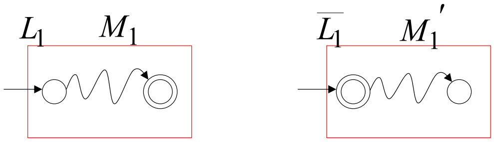
$L_1 = \{a^n b\}$



$L_1^R = \{ba^n\}$



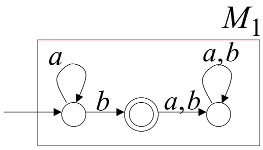
Complement



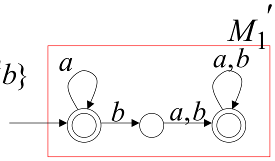
1. Take the DFA that accepts L_1
2. Make accepting states non-final, and vice-versa

Example

$L_1 = \{a^n b\}$



$\overline{L_1} = \{a,b\}^* - \{a^n b\}$



Intersection

L_1 regular
L_2 regular

We show $L_1 \cap L_2$ regular

DeMorgan's Law: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

L_1, L_2 regular

$\Rightarrow$ $\overline{L_1}$, $\overline{L_2}$ regular

$\Rightarrow$ $\overline{L_1} \cup \overline{L_2}$ regular

$\Rightarrow$ $\overline{\overline{L_1} \cup \overline{L_2}}$ regular

$\Rightarrow$ $L_1 \cap L_2$ regular

Example

$L_1 = \{a^n b\}$ regular
$L_2 = \{ab, ba\}$ regular

$\Rightarrow$ $L_1 \cap L_2 = \{ab\}$ regular

Another Proof for Intersection Closure

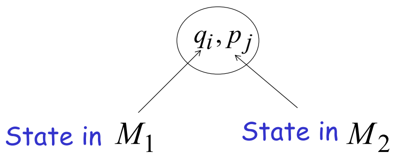
Machine M_1: DFA for L_1

Machine M_2: DFA for L_2

Construct a new DFA M that accepts $L_1 \cap L_2$

M simulates in parallel M_1 and M_2

States in M



DFA M_1

$q_1 \xrightarrow{a} q_2$

transition

DFA M_2

$p_1 \xrightarrow{a} p_2$

transition

DFA M

$q_1, p_1 \xrightarrow{a} q_2, p_2$

New transition

DFA M_1

$\rightarrow q_0$

initial state

DFA M_2

$\rightarrow p_0$

initial state

DFA M

$\rightarrow q_0, p_0$

New initial state

DFA M_1

q_i

accept state

DFA M_2

p_j p_k

accept states

DFA M

q_i, p_j q_i, p_k

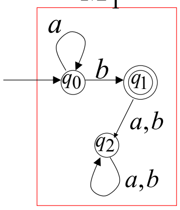
New accept states

Both constituents must be accepting states

Example:

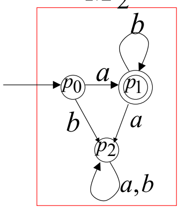
$L_1 = \{a^n b\}$ $n \geq 0$

M_1



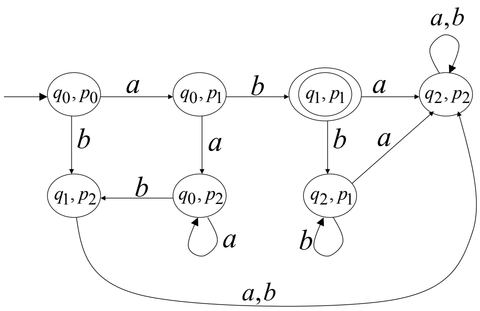
$L_2 = \{ab^m\}$ $m \geq 0$

M_2



Automaton for intersection

$$L = \{a^n b\} \cap \{ab^n\} = \{ab\}$$



M simulates in parallel M_1 and M_2

M accepts string w if and only if:

M_1 accepts string w

and M_2 accepts string w

$$L(M) = L(M_1) \cap L(M_2)$$

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: $\varnothing, \quad \lambda, \quad \alpha$

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

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

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

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

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

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

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

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

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 containing substring 00 }

Example

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

$L(r)$ = { 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 }

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

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

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

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

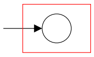
For any regular expression r
the language $L(r)$ is regular

Proof by induction on the size of r

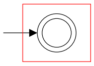
Induction Basis

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

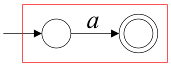
Corresponding NFAs



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



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



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

regular languages

49

Inductive Hypothesis

Suppose

that for regular expressions r_1 and r_2,

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

50

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

51

By definition of regular expressions:

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

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

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

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

52

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	$(L(r_1))*$

53

Therefore:

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

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

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

Are regular languages

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

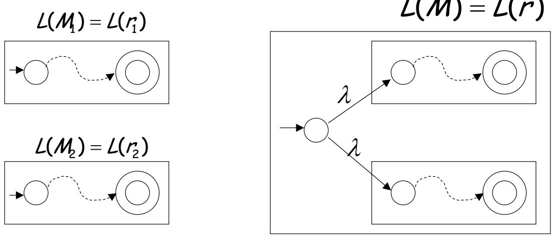
End of Proof-Part 1

54

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

Example: $r = r_1 + r_2$

$L(M) = L(r)$



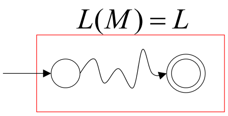
Proof - Part 2

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

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

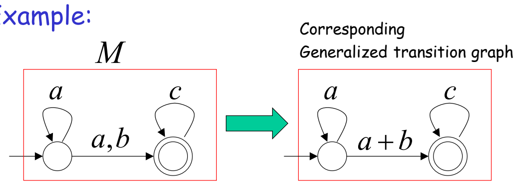
Since L is regular, there is a NFA M that accepts it



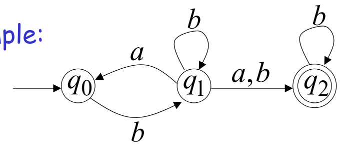
Take it with a single final state

From M construct the equivalent Generalized Transition Graph in which transition labels are regular expressions

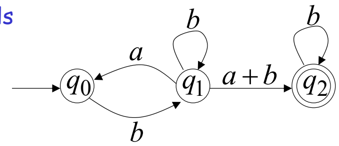
Example:



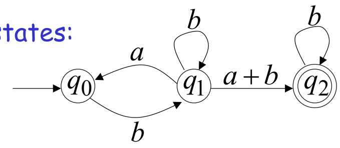
Another Example:



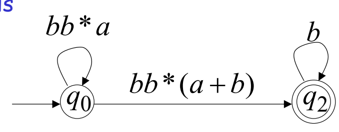
Transition labels are regular expressions



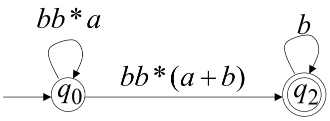
Reducing the states:



Transition labels are regular expressions



Resulting Regular Expression:

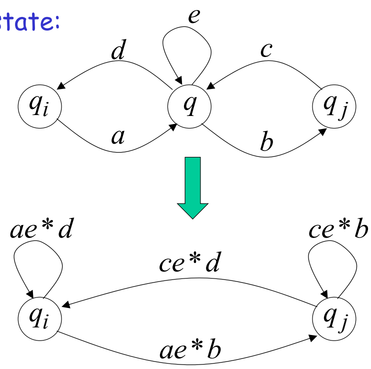


$$r = (bb^*a)^*bb^*(a+b)b^*$$

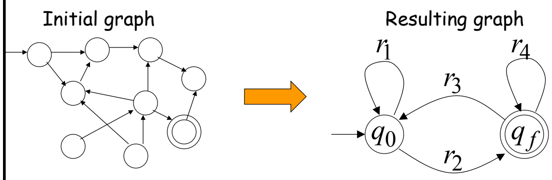
$$L(r) = L(M) = L$$

In General

Removing a state:



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



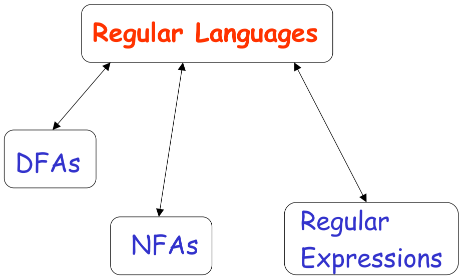
The resulting regular expression:

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

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

End of Proof-Part 2

Standard Representations of Regular Languages



When we say: We are given a Regular Language L

We mean: Language L is in a standard representation

(DFA, NFA, or Regular Expression)