Single Final State for NFAs

class 4

1

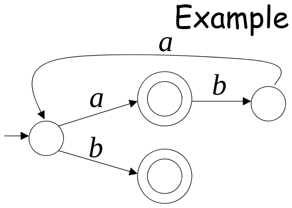
Any NFA can be converted

to an equivalent NFA

with a single final state

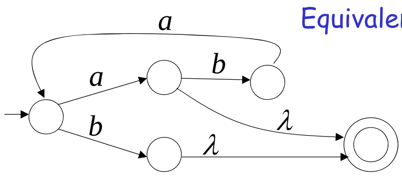
2

Example



NFA

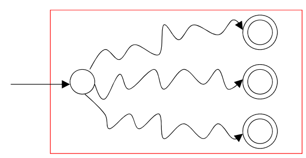
Equivalent NFA



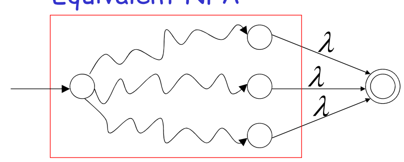
3

In General

NFA



Equivalent NFA

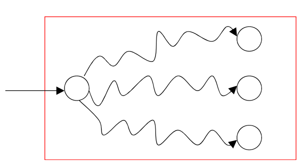


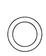
Single
final state

4

Extreme Case

NFA without final state





Add a final state
Without transitions

5

Properties of Regular Languages

6

For regular languages L_1 and L_2
we will prove that:

Union:	$L_1 \cup L_2$	Are regular Languages
Concatenation:	$L_1 L_2$	
Star:	$L_1 *$	
Reversal:	$L_1{}^R$	
Complement:	$\overline{L_1}$	
Intersection:	$L_1 \cap L_2$	

7

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

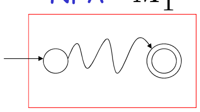
Union:	$L_1 \cup L_2$
Concatenation:	$L_1 L_2$
Star:	$L_1 *$
Reversal:	$L_1{}^R$
Complement:	$\overline{L_1}$
Intersection:	$L_1 \cap L_2$

8

Regular language L_1

$L(M_1) = L_1$

NFA M_1

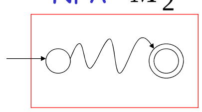


Single final state

Regular language L_2

$L(M_2) = L_2$

NFA M_2



Single final state

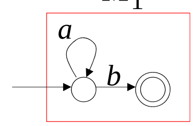
9

Example

M_1

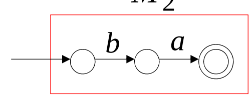
$n \geq 0$

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



M_2

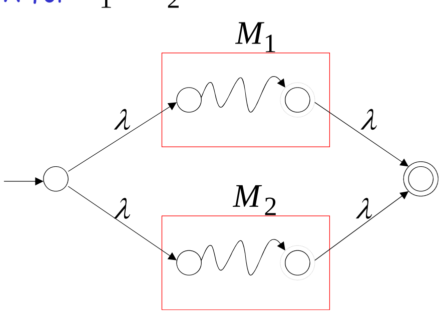
$L_2 = \{ba\}$



10

Union

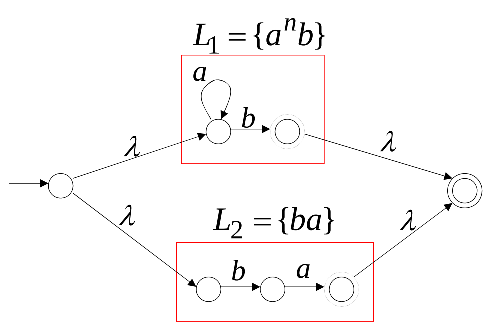
NFA for $L_1 \cup L_2$



11

Example

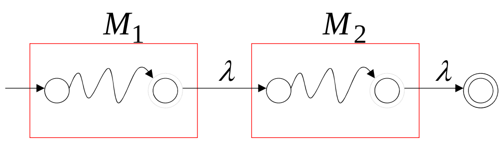
NFA for $L_1 \cup L_2 = \{a^n b\} \cup \{ba\}$



12

Concatenation

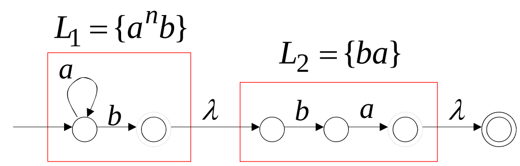
NFA for L_1L_2



13

Example

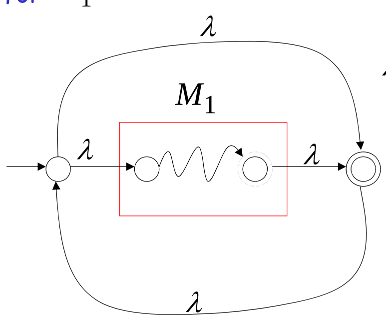
NFA for $L_1L_2 = \{a^nb\}\{ba\} = \{a^nbba\}$



14

Star Operation

NFA for L_1*



$\lambda \in L_1*$

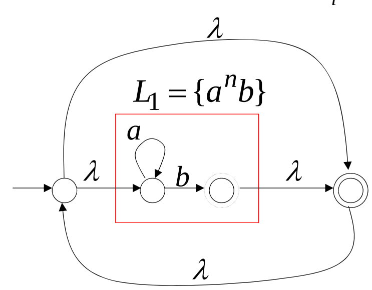
15

Example

NFA for $L_1* = \{a^nb\}*$

$w = w_1w_2\cdots w_k$

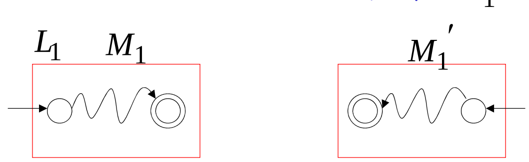
$w_i \in L_1$



16

Reverse

NFA for L_1^R

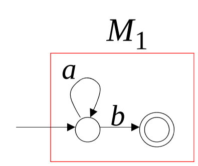


1. Reverse all transitions
2. Make initial state final state and vice versa

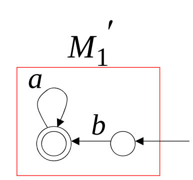
17

Example

$L_1 = \{a^nb\}$

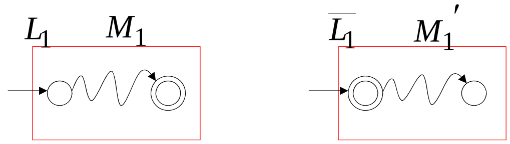


$L_1^R = \{ba^n\}$



18

Complement



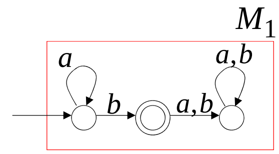
1. Take the **DFA** that accepts L_1

2. Make final states non-final, and vice-versa

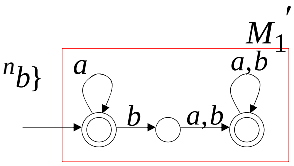
19

Example

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



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



20

Intersection

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

21

Example

$$\left.\begin{array}{l} L_1 = \{a^n b\} \text{ regular} \\ L_2 = \{ab, ba\} \text{ regular} \end{array}\right\} \Rightarrow L_1 \cap L_2 = \{ab\}$$

regular

22

Regular Expressions

23

Regular Expressions

Regular expressions
describe regular languages

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

describes the language

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

24

Recursive Definition

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

Given regular expressions r_1 and r_2

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

25

Examples

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

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

26

Languages of Regular Expressions

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

Example

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

27

Definition

For primitive regular expressions:

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

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

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

28

Definition (continued)

For regular expressions r_1 and r_2

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

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

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

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

29

Example

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

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

30

Example

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

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

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 }

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

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

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

Regular Expressions and Regular Languages

37

Theorem

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

38

Theorem - Part 1

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

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

39

Theorem - Part 2

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

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

40

Proof - Part 1

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

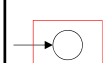
Proof by induction on the size of r

41

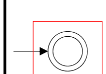
Induction Basis

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

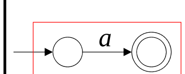
NFAs



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



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



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

regular languages

42

Inductive Hypothesis

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

43

Inductive Step

We will prove:

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

44

By definition of regular expressions:

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

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

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

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

45

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	$(L(r_1))*$

46

Therefore:

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

47

And trivially:

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

48

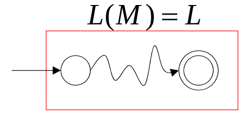
Proof - Part 2

2. 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

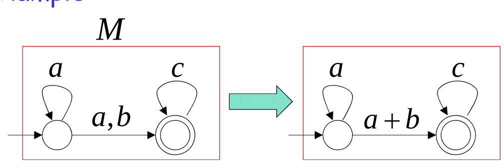
$L(M) = L$



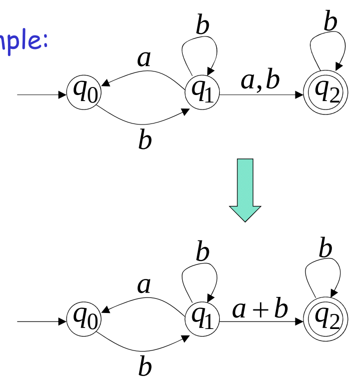
Single final state

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

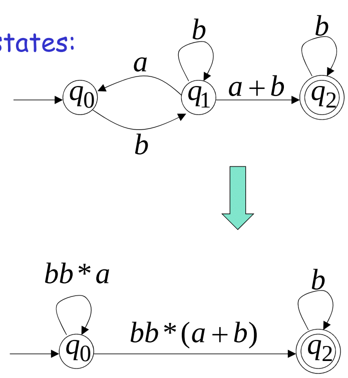
Example:



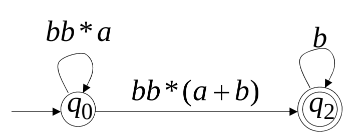
Another Example:



Reducing the states:



Resulting Regular Expression:

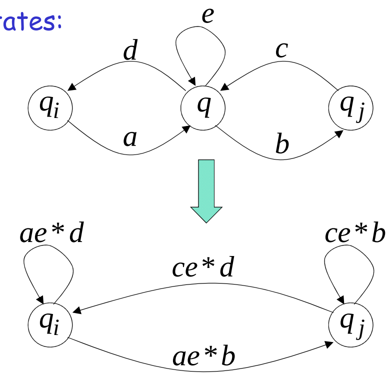


$$r = (bb^*a)^*bb^*(a+b)b^*$$

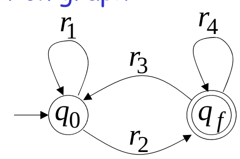
$$L(r) = L(M) = L$$

In General

Removing states:



The final transition graph:



The resulting regular expression:

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

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