Single Final State for NFAs and DFAs
Observation

Any Finite Automaton (NFA or DFA) can be converted to an equivalent NFA with a single final state
Example

NFA

Equivalent NFA
In General

NFA

Equivalent NFA

Single final state
Extreme Case

NFA without final state

Add a final state
Without transitions
Some Properties of Regular Languages
Properties

For regular languages $L_1$ and $L_2$ we will prove that:

Union: $L_1 \cup L_2$

Concatenation: $L_1L_2$

Star: $L_1^*$

Are regular Languages
We Say:
Regular languages are **closed under**

**Union:** \( L_1 \cup L_2 \)

**Concatenation:** \( L_1L_2 \)

**Star:** \( L_1^* \)
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
Example

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

$M_1$

$L_2 = \{ba\}$

$M_2$
NFA for $L_1 \cup L_2$
Example

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

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

$L_2 = \{b a\}$
Concatenation

NFA for $L_1L_2$
Example

NFA for \( L_1L_2 = \{a^n b\}\ \{ba\} = \{a^n bba\} \)

\[ L_1 = \{a^n b\} \]

\[ L_2 = \{ba\} \]

15
Star Operation

NFA for $L_1^*$
Example

NFA for \( L_1^* = \{a^n b\}^* \)
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: $\emptyset$, $\lambda$, $\alpha$

Given regular expressions $r_1$ and $r_2$

\[
\begin{align*}
\{ & r_1 + r_2 \\ & r_1 \cdot r_2 \\ & r_1^* \\ & (r_1) \}
\end{align*}
\]

Are regular expressions
Examples

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

Not a regular expression: \((a + b +)\)
Languages of Regular Expressions

\[ L(r) : \text{ language of regular expression } r \]

Example

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

For primitive regular expressions:

\[ L(\emptyset) = \emptyset \]

\[ L(\lambda) = \{ \lambda \} \]

\[ L(a) = \{ a \} \]
Definition (continued)

For regular expressions $r_1$ and $r_2$

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

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

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

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

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

$L((a + b) \cdot a^*) = L((a + b)) \, L(a^*)$

$= L(a + b) \, L(a^*)$

$= (L(a) \cup L(b)) \, (L(a))^*$

$= ([a] \cup [b]) \, ([a])^*$

$= \{a, b\} \, \{\lambda, a, aa, aaaa, \ldots\}$

$= \{a, aa, aaaa, \ldots, b, ba, baa, \ldots\}$
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 \( r = (0+1)^*00(0+1)^* \)

\[ L(r) = \{ \text{all strings with at least two consecutive } 0 \} \]
Example

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

\[ L(r) = \{ \text{all strings without two consecutive 0} \} \]
Equivalent Regular Expressions

Definition:

Regular expressions $r_1$ and $r_2$ are equivalent if $L(r_1) = L(r_2)$
Example

\[ L = \{ \text{all strings with at least two consecutive 0} \} \]

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

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

\[ L(r_1) = L(r_2) = L \quad \rightarrow \quad r_1 \text{ and } r_2 \text{ are equivalent regular expr.} \]
Regular Expressions
and
Regular Languages
Theorem

\[ \{ \text{Languages} \} = \{ \text{Regular Languages} \} \]

\{ Generated by \}
\{ Regular Expressions \}
Theorem - Part 1

\[
\left\{ \text{Languages} \right\} \subseteq \left\{ \text{Regular Languages} \right\}
\]

1. For any regular expression \( r \), the language \( L(r) \) is regular
Theorem - Part 2

\[
\begin{align*}
\text{Languages} & \quad \supseteq \quad \text{Regular Languages} \\
\text{Generated by} & \quad \text{Regular Expressions}
\end{align*}
\]

2. For any regular language \( L \) there is a regular expression \( r \) with \( L(r) = L \)
Proof - Part 1

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

Proof by induction on the size of $r$
Induction Basis

Primitive Regular Expressions: $\emptyset$, $\lambda$, $\alpha$

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:

\[ L(r_1 + r_2) = L(r_1) \cup L(r_2) \]

\[ L(r_1 \cdot r_2) = L(r_1) \cdot L(r_2) \]

\[ L(r_1^*) = (L(r_1))^* \]

\[ L((r_1)) = L(r_1) \]
By inductive hypothesis we know:

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

We also know:

Regular languages are closed under

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

concatenation $\quad L(r_1) L(r_2)$

star $\quad (L(r_1))^*$
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
And trivially:

\[ L((r_1)) \] is a regular language
**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. Transition labels are regular expressions.

Example:
Another Example:
Reducing the states:

$q_0 \xrightarrow{b} q_1 \xrightarrow{a+b} q_2$

$q_0 \xrightarrow{bb*} q_2$
Resulting Regular Expression:

\[ r = (bb * a) * bb * (a + b) b * \]

\[ L(r) = L(M) = L \]
In General

Removing states:

\[ q_i \xrightarrow{a} q \xrightarrow{e} q_j \]

\[ q_i \xleftarrow{ae^*b} \]

\[ q_i \xrightarrow{ae^*d} q_j \xleftarrow{ce^*b} q_j \]
Obtaining the final regular expression:

\[
r = r_1 \ast r_2 (r_4 + r_3 r_1 \ast r_2)^* \]

\[
L(r) = L(M) = L
\]