Contents

- Deterministic Finite Automata (DFA)
- Regular Languages
- Non-Deterministic Finite Automata (NFA)
- Regular Expressions
- Transformations
- Pumping Lemma
Deterministic Finite Automata (DFA)
**Electric bulb**

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Design the logic behind an electric bulb.</td>
</tr>
</tbody>
</table>
Electric bulb

Problem
- Design the logic behind an electric bulb.

Solution
- **Diagram.**
- **Analysis.**
  
  States = \{nolight, light\}, Input = \{off, on\}
- **Finite Automaton.**
Problem

- Design the logic behind a multispeed fan.
Multispeed fan

Problem

- Design the logic behind a multispeed fan.

Solution

- Diagram.
- Finite Automaton.

- Analysis.
  States = \{0, 1, 2, 3\}
  Input = \{⟳, ⟲\}
Automatic doors

Problem

• Design the logic behind automatic doors in Walmart.
Solution

- **Diagram.**

- **Analysis.**
  States = \{close, open\}, Input = \{left, right, neither\}

- **Finite Automaton.**
Basic features of finite automata

- A finite automaton is a simple computer with extremely limited memory.
- A finite automaton has a finite set of states.
- Current state of a finite automaton changes when it reads an input symbol.
- A finite automaton acts as a language acceptor i.e., outputs “yes” or “no.”
Why should you care?

Deterministic Finite Automata (DFA) are everywhere.
- ATMs
- Ticket machines
- Vending machines
- Traffic signal systems
- Calculators
- Digital watches
- Automatic doors
- Elevators
- Washing machines
- Dishwashing machines
- Thermostats
- Train switches
- (CS) Compilers
- (CS) Search engines
- (CS) Regular expressions
Why should you care?

Probabilistic Finite Automata (PFA) are everywhere, too.

- Speech recognition
- Optical character recognition
- Thermodynamics
- Statistical mechanics
- Chemical reactions
- Information theory
- Queueing theory
- PageRank algorithm
- Statistics
- Reinforcement learning
- Price changes in finance
- Genetics
- Algorithmic music composition
- Bioinformatics
- Probabilistic forecasting
What is a decision problem?

A decision problem is a computational problem with a ‘yes’ or ‘no’ answer. A computer that solves a decision problem is a decider.

Input to a decider: A string $w$

Output of a decider: Accept ($w$ is in the language) or Reject ($w$ is not in the language)
What is a decision problem?

- Language $= \text{English language} = \{\text{milk, food, sleep,} \ldots\} \quad \Rightarrow \text{Accept}
- \text{Not in language} = \{\text{zffgb, cdcapqw,} \ldots\} \quad \Rightarrow \text{Reject}
What is a decision problem?

Some strings → Accept

Other strings → Reject
How does a DFA work?

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Does the DFA accept the string ( bbab )?</td>
</tr>
</tbody>
</table>

\[
\begin{array}{c}
\text{start} \\
q_0 \xrightarrow{a} q_1 \\
q_1 \xrightarrow{b} q_1 \\
q_1 \xrightarrow{a} q_2 \\
q_2 \xrightarrow{a, b} q_1 \\
\end{array}
\]
How does a DFA work?

**Problem**
- Does the DFA accept the string $bbab$?

![DFA Diagram](image)

**Solution**
The DFA accepts the string $bbab$. The computation is:
1. Start in state $q_0$
2. Read $b$, follow transition from $q_0$ to $q_1$.
3. Read $b$, follow transition from $q_1$ to $q_1$.
4. Read $a$, follow transition from $q_1$ to $q_2$.
5. Read $b$, follow transition from $q_2$ to $q_1$.
6. Accept because the DFA is in an accept state $q_1$ at the end of the input.
How does a DFA work?

**Problem**

- Does the DFA accept the string $aaba$?

![DFA Diagram]

**Solution**

The DFA rejects the string $aaba$. The computation is:

1. Start in state $q_0$
2. Read $a$, follow transition from $q_0$ to $q_0$.
3. Read $a$, follow transition from $q_0$ to $q_0$.
4. Read $b$, follow transition from $q_0$ to $q_1$.
5. Read $a$, follow transition from $q_1$ to $q_2$.
6. Reject because the DFA is in a reject state $q_2$ at the end of the input.
How does a DFA work?

Problem

• Does the DFA accept the string $aaba$?

Solution

The DFA rejects the string $aaba$. The computation is:
1. Start in state $q_0$
2. Read $a$, follow transition from $q_0$ to $q_0$.
3. Read $a$, follow transition from $q_0$ to $q_0$.
4. Read $b$, follow transition from $q_0$ to $q_1$.
5. Read $a$, follow transition from $q_1$ to $q_2$.
6. Reject because the DFA is in a reject state $q_2$ at the end of the input.
How does a DFA work?

**DFA Diagram:**
- **States:** $q_0$, $q_1$, $q_2$
- **Transitions:**
  - From $q_0$:
    - $a$ to $q_1$
    - $b$ to $q_1$
  - From $q_1$:
    - $a$ to $q_2$
    - $b$ to $q_2$
- **Accepting State:** $q_2$
- **Rejecting Input:** $aaba$
- **Accepting Input:** $bbab$
How does a DFA work?

Problem

- What language does the DFA accept?

The DFA accepts the following strings:
- b, ab, bb, aabbbb, ababababab, ...
- ba, ab, abaa, ababaaaaaa, ...

The DFA rejects the following strings:
- a, ba, babaaa, ...
How does a DFA work?

**Problem**

- What language does the DFA accept?

```
start → q0 → q1 → q2
```

- $a$ → $b$
- $b$ → $a$

<table>
<thead>
<tr>
<th>$q_0$</th>
<th>$q_1$</th>
<th>$q_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$b$</td>
<td>$a$, $b$</td>
</tr>
</tbody>
</table>

**Examples**

- The DFA accepts the following strings:
  - $b$, $ab$, $bb$, $aabbbb$, $ababababab$, $\ldots$  ▶ ends with $b$
  - $baa$, $abaa$, $ababaaaaaa$, $\ldots$ ▶ ends with $b$ followed by even $a$’s
- The DFA rejects the following strings:
  - $a$, $ba$, $babaaa$, $\ldots$
- What language does the DFA accept?
Construct DFA for $\Sigma = \{a\}$

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Construct a DFA that accepts all strings from the language $L = {\epsilon, a, aa, aaa, aaaa, \ldots}$</td>
</tr>
</tbody>
</table>
Construct DFA for $\Sigma = \{a\}$

### Problem
- Construct a DFA that accepts all strings from the language $L = \{\epsilon, a, aa, aaa, aaaa, \ldots\}$

### Solution
- **Language** $L$: $\Sigma^* = \{\epsilon, a, aa, aaa, aaaa, \ldots\}$
- **Expression**: $a^*$
- **Deterministic Finite Automaton (DFA) $M$**:

  ![DFA Diagram]

  \[ q_0 \]
  
  \[ \text{start} \rightarrow q_0 \]
Construct DFA for $\Sigma = \{a\}$

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Construct a DFA that accepts all strings from the language $L = {}$.</td>
</tr>
</tbody>
</table>
Problem

- Construct a DFA that accepts all strings from the language $L = \emptyset$

Solution

- Language $L$: $\phi = \emptyset$
- Expression: $\phi$
- DFA $M$:

```
start --- a ----> q₀
```

▷ Empty language
Construct DFA for $\Sigma = \{a\}$

**Problem**

- Construct a DFA that accepts all strings from the language $L = \{a, aa, aaa, aaaa, \ldots\}$
**Problem**

- Construct a DFA that accepts all strings from the language \( L = \{a, aa, aaa, aaaa, \ldots\} \)

**Solution**

- Language \( L: \Sigma^* - \{\epsilon\} = \{a, aa, aaa, aaaa, \ldots\} \)
- Expression: \( a^+ \)
- DFA \( M: \)

```
\begin{figure}
  \centering
  \begin{tikzpicture}
    \node[state, initial] (q0) at (0,0) {\(q_0\)};
    \node[state, accepting] (q1) at (1,0) {\(q_1\)};
    \draw (q0) edge[->] node {\(a\)} (q1);
    \draw (q1) edge[loop above] node {\(a\)} (q1);
  \end{tikzpicture}
\end{figure}
```
Construct DFA for $\Sigma = \{a\}$

**Problem**

- Construct a DFA that accepts all strings from the language $L = \{\epsilon\}$
<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Construct a DFA that accepts all strings from the language $L = {\epsilon}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Language $L$: ${\epsilon}$</td>
</tr>
<tr>
<td>• Expression: $\epsilon$</td>
</tr>
<tr>
<td>• DFA $M$:</td>
</tr>
</tbody>
</table>

```
start \rightarrow q_0 \xrightarrow{a} q_1
```

```python
class DFA:
    def __init__(self, start, final):
        self.start = start
        self.final = final
        self.transitions = {}

    def add_transition(self, state, symbol, next_state):
        if state not in self.transitions:
            self.transitions[state] = {}
        self.transitions[state][symbol] = next_state

def construct_dfa(Sigma):
    dfa = DFA(None, None)
    dfa.add_transition(None, 'a', 1)
    dfa.final = 1
    return dfa
```
Construct DFA for $\Sigma = \{a\}$

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Construct a DFA that accepts all strings from the language $L = {aaa}$</td>
</tr>
</tbody>
</table>
Construct DFA for $\Sigma = \{a\}$

**Problem**
- Construct a DFA that accepts all strings from the language $L = \{aaa\}$

**Solution**
- Language $L$: $\{aaa\}$
- Expression: $aaa$
- DFA $M$:

```
start  $\rightarrow$ q0  $\rightarrow$ q1  $\rightarrow$ q2  $\rightarrow$ q3  $\rightarrow$ q4
\hline
\hline
q0  $\xrightarrow{a}$ q1  $\xrightarrow{a}$ q2  $\xrightarrow{a}$ q3  $\xrightarrow{a}$ q4
```

$\xrightarrow{a}$ loops back to q4
Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings with even size}\}$
Construct DFA for $\Sigma = \{a\}$

**Problem**

• Construct a DFA that accepts all strings from the language $L = \{\text{strings with even size}\}$

**Solution**

• Language $L$: $\{\epsilon, aa, aaaa, aaaaaa, \ldots\}$
• Expression: $(aa)^*$
• DFA $M$:

```
q0  --a-->  q1
```

Start $q_0$
Construct DFA for $\Sigma = \{a\}$

**Problem**

- Construct a DFA that accepts all strings from the language $L = \{\text{strings with odd size}\}$
Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings with odd size}\}$

Solution

- Language $L$: $\{a, aaa, aaaaa, \ldots\}$
- Expression: $a(aa)^*$
- DFA $M$:
Construct DFA for $\Sigma = \{a\}$

**Problem**

- Construct a DFA that accepts all strings from the language $L = \{\text{strings of size divisible by 3}\}$
Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings of size divisible by } 3\}$

Solution

- Language $L$: $\{\epsilon, aaa, aaaaaaa, aaaaaaaaaa, \ldots\}$
- Expression: $(aaa)^*$
- DFA $M$:

$$
\text{start} \quad \xrightarrow{a} q_0 \quad \xrightarrow{a} q_1 \quad \xrightarrow{a} q_2
$$
Construct DFA for $\Sigma = \{a\}$

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Construct a DFA that accepts all strings from the language $L = {\text{strings of size not divisible by 3}}$</td>
</tr>
</tbody>
</table>
Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings of size not divisible by 3}\}$

Solution

- Language $L$: $\{a, aa, aaaa, aaaaaa, \ldots\}$
- Expression: $(a \mid aa)(aaa)^*$
- DFA $M$:
Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings of size divisible by 6}\}$
Construct DFA for \( \Sigma = \{a\} \)

**Problem**

- Construct a DFA that accepts all strings from the language \( L = \{\text{strings of size divisible by 6}\} \)

**Solution**

- Language \( L: \{\epsilon, aaaaaa, aaaaaaaaaaaa, \ldots\} \)
- Expression: \((aaaaaa)^*\)
- DFA \( M:\)

\[
\begin{array}{cccccc}
 q_0 & \xrightarrow{a} & q_1 & \xrightarrow{a} & q_2 & \xrightarrow{a} & q_3 & \xrightarrow{a} & q_4 & \xrightarrow{a} & q_5 \\
 \text{start} & & & & & & & & & & \\
\end{array}
\]

Can you think of another approach?
Construct DFA for \( \Sigma = \{a\} \)

### Problem

- Construct a DFA that accepts all strings from the language \( L = \{ \text{strings of size divisible by 6} \} \)

### Solution

- Language \( L: \{ \epsilon, aaaaaa, aaaaaaaa, \ldots \} \)
- Expression: \((aaaaaa)^*\)
- DFA \( M: \)

```

\[
\begin{align*}
\text{start} & \rightarrow q_0 & a & \rightarrow q_1 & a & \rightarrow q_2 & a & \rightarrow q_3 & a & \rightarrow q_4 & a & \rightarrow q_5 \\
& & & & & & & & & & & & a
\end{align*}
\]
```

- Can you think of another approach?
Construct DFA for $\Sigma = \{a\}$

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Construct a DFA that accepts all strings from the language $L = {\text{strings of size divisible by 6}}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solution</th>
</tr>
</thead>
</table>
| • Let $n = \text{string size}$  
• Observation  
  $n \mod 6 = 0 \iff n \mod 2 = 0 \text{ and } n \mod 3 = 0$  
• Idea  
  Build DFA $M_1$ for $n \mod 2 = 0$.  
  Build DFA $M_2$ for $n \mod 3 = 0$.  
  Run $M_1$ and $M_2$ in parallel.  
  Accept a string if both DFAs $M_1$ and $M_2$ accept the string.  
  Reject a string if at least one of the DFAs $M_1$ and $M_2$ reject the string.  
• It is possible to build complicated DFAs from simpler DFAs |
Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings with size } n \text{ where } n \mod 4 = 3\}$
Construct DFA for $\Sigma = \{a\}$

**Problem**
- Construct a DFA that accepts all strings from the language $L = \{\text{strings with size } n \text{ where } n \mod 4 = 3\}$

**Solution**
- Language $L$: $\{aaa, aaaaaaaa, aaaaaaaaaaaaa, \ldots\}$
- Expression: $aaa(aaaa)^*$
- DFA $M$:

```
start → q0 ← a → q1 → a → q2 → a → q3 ← a → q4
```

What about strings with size $n$ where $n \mod k = i$?
Problem

- Construct a DFA that accepts all strings from the language
  \( L = \{ \text{strings with size } n \text{ where } n \mod 4 = 3 \} \)

Solution

- Language \( L: \{aaa, aaaaaaaa, aaaaaaaaaaaaa, \ldots\} \)
- Expression: \( aaa(aaaa)^* \)
- DFA \( M: \)

```
start \rightarrow q_0 \rightarrow a \rightarrow q_1 \rightarrow a \rightarrow q_2 \rightarrow a \rightarrow q_3 \rightarrow a \rightarrow q_4
```

- What about strings with size \( n \) where \( n \mod k = i \)?
Construct DFA for $\Sigma = \{a\}$

More Problems

Construct a DFA that accepts all strings from the language $L = \{\text{strings with size } n\}$ such that
1. $n^2 - 5n + 6 = 0$
2. $n \in [4, 37]$
3. $n$ is a perfect cube
4. $n$ is a prime number
5. $n$ satisfies a mathematical function $f(n)$
The specification of DFA consists of:
- A (finite) alphabet
- A (finite) set of states
- Which state is the start state?
- Which states are the final states?
- What is the transition from each state, on each input character?
What is a deterministic finite automaton (DFA)?

- Deterministic = Events can be determined precisely
- Finite = Finite and small amount of space used
- Automaton = Computing machine

**Definition**

A deterministic finite automaton (DFA) $M$ is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where,

1. $Q$: A finite set (set of states).
2. $\Sigma$: A finite set (alphabet).
3. $\delta$: $Q \times \Sigma \rightarrow Q$ is the transition function.
4. $q_0$: The start state (belongs to $Q$).
5. $F$: The set of accepting/final states, where $F \in Q$. 


What is a deterministic finite automaton (DFA)?

- Deterministic = Events can be determined precisely
- Finite = Finite and small amount of space used
- Automaton = Computing machine

**Definition**

A **deterministic finite automaton (DFA)** $M$ is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where,

1. $Q$: A finite set (set of states). ▶ **Space** (computer memory)
2. $\Sigma$: A finite set (alphabet).
3. $\delta: Q \times \Sigma \rightarrow Q$ is the transition function. ▶ **Time** (computation)
4. $q_0$: The start state (belongs to $Q$).
5. $F$: The set of accepting/final states, where $F \subseteq Q$. 
We say that a DFA $M$ accepts a language $L$ if:

$$L = w | M \text{ accepts } w.$$  

A language is called a **regular language** if some DFA accepts it.
What is a regular language?

**Definition**

- A DFA **accepts** a string $w = w_1w_2\ldots w_k$ iff there exists a sequence of states $r_0, r_1, \ldots, r_k$ such that the current state starts from the start state and ends at a final state when all the symbols of $w$ have been read.
- A DFA **rejects** a string iff it does not accept it.
Construct DFA for $\Sigma = \{a, b\}$

**Problem**
- Construct a DFA that accepts all strings from the language $L = \{\text{strings with odd number of } b\text{'s}\}$
Construct DFA for $\Sigma = \{a, b\}$

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Construct a DFA that accepts all strings from the language $L = {\text{strings with odd number of } b\text{'s}}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>States</strong></td>
</tr>
<tr>
<td>• $q_{\text{odd}}$: DFA is in this state if it has read odd $b$'s.</td>
</tr>
<tr>
<td>• $q_{\text{even}}$: DFA is in this state if it has read even $b$'s.</td>
</tr>
</tbody>
</table>
Construct DFA for $\Sigma = \{a, b\}$

**Problem**
- Construct a DFA that accepts all strings from the language $L = \{\text{strings with odd number of } b\'}s\}$

**Solution**
- Language $L$: $\{\text{strings with odd number of } b\'}s\}$
- Expression: $a^*b(a \mid ba^*)^* \text{ or } a^*ba^*(ba^*ba^*)^*$
- DFA $M$:

![DFA Diagram]

- States: $q_{even}$ (even number of $b$'s), $q_{odd}$ (odd number of $b$'s)
- Start state: $q_{even}$
- Accepting states: $q_{odd}$
- Transitions:
  - $a$: $q_{even} \rightarrow q_{even}$
  - $b$: $q_{even} \rightarrow q_{odd}$, $q_{odd} \rightarrow q_{even}$
Construct DFA for $\Sigma = \{ a, b \}$

**Problem**
- Construct a DFA that accepts all strings from the language $L = \{ \text{strings with odd number of } b's \}$

**Solution (continued)**
- DFA $M$ is specified as:
  - Set of states is $Q = \{ q_{\text{even}}, q_{\text{odd}} \}$
  - Set of symbols is $\Sigma = \{ a, b \}$
  - Start state is $q_{\text{even}}$
  - Set of accept states is $F = \{ q_{\text{even}} \}$
  - Transition function $\delta$ is:

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_{\text{even}}$</td>
<td>$q_{\text{even}}$</td>
<td>$q_{\text{odd}}$</td>
</tr>
<tr>
<td>$q_{\text{odd}}$</td>
<td>$q_{\text{odd}}$</td>
<td>$q_{\text{even}}$</td>
</tr>
</tbody>
</table>
Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings containing } bab\}$
Construct DFA for $\Sigma = \{a, b\}$

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Construct a DFA that accepts all strings from the language $L = {\text{strings containing } bab}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>States</strong></td>
</tr>
<tr>
<td>• $q_b$: DFA is in this state if the last symbol read was $b$, but the substring $bab$ has not been read.</td>
</tr>
<tr>
<td>• $q_{ba}$: DFA is in this state if the last two symbols read were $ba$, but the substring $bab$ has not been read.</td>
</tr>
<tr>
<td>• $q_{bab}$: DFA is in this state if the substring $bab$ has been read in the input string.</td>
</tr>
<tr>
<td>• $q$: In all other cases, the DFA is in this state.</td>
</tr>
</tbody>
</table>
Construct DFA for $\Sigma = \{a, b\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings containing } bab\}$

Solution (continued)

- Language $L$: \{strings containing $bab$\}
- Expression: $(a^+ b^+ a^+ a)bab(a \mid b)^*$
- DFA $M$:

```
  q_0 \rightarrow a \rightarrow q_0 \rightarrow b \rightarrow q_a \rightarrow b \rightarrow q_{bab} \rightarrow \bullet
  q_0 \rightarrow a \rightarrow q_b \rightarrow a \rightarrow q_{ba} \rightarrow b \rightarrow q_{bab} \rightarrow \bullet
  q_0 \rightarrow a \rightarrow q_b \rightarrow a \rightarrow q_{ba} \rightarrow b \rightarrow q_{bab} \rightarrow \bullet
  q_0 \rightarrow a \rightarrow q_b \rightarrow a \rightarrow q_{ba} \rightarrow b \rightarrow q_{bab} \rightarrow \bullet
  q_0 \rightarrow a \rightarrow q_b \rightarrow a \rightarrow q_{ba} \rightarrow b \rightarrow q_{bab} \rightarrow \bullet
  q_0 \rightarrow a \rightarrow q_b \rightarrow a \rightarrow q_{ba} \rightarrow b \rightarrow q_{bab} \rightarrow \bullet
```

Construct DFA for $\Sigma = \{a, b\}$

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Construct a DFA that accepts all strings from the language $L = {\text{strings containing } bab}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solution (continued)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• DFA $M$ is specified as</td>
</tr>
<tr>
<td>Set of states is $Q = {q, q_b, q_{ba}, q_{bab}}$</td>
</tr>
<tr>
<td>Set of symbols is $\Sigma = {a, b}$</td>
</tr>
<tr>
<td>Start state is $q$</td>
</tr>
<tr>
<td>Set of accept states is $F = {q_{bab}}$</td>
</tr>
<tr>
<td>Transition function $\delta$ is:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q$</td>
<td>$q$</td>
<td>$q_b$</td>
</tr>
<tr>
<td>$q_b$</td>
<td>$q_{ba}$</td>
<td>$q_b$</td>
</tr>
<tr>
<td>$q_{ba}$</td>
<td>$q$</td>
<td>$q_{bab}$</td>
</tr>
<tr>
<td>$q_{bab}$</td>
<td>$q_{bab}$</td>
<td>$q_{bab}$</td>
</tr>
</tbody>
</table>
### Properties of regular languages

Let $L_1$ and $L_2$ be regular languages. Then, the following languages are regular.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complement</td>
<td>$\overline{L_1} = { x \mid x \in \Sigma^* \text{ and } x \notin L_1 }$</td>
</tr>
<tr>
<td>Union</td>
<td>$L_1 \cup L_2 = { x \mid x \in L_1 \text{ or } x \in L_2 }$</td>
</tr>
<tr>
<td>Intersection</td>
<td>$L_1 \cap L_2 = { x \mid x \in L_1 \text{ and } x \in L_2 }$</td>
</tr>
<tr>
<td>Concatenation</td>
<td>$L_1 \cdot L_2 = { xy \mid x \in L_1 \text{ and } y \in L_2 }$</td>
</tr>
<tr>
<td>Star</td>
<td>$L_1^* = { x_1x_2\ldots x_k \mid k \geq 0 \text{ and each } x_i \in L_1 }$</td>
</tr>
</tbody>
</table>
Construct DFA for $L_1 \cup L_2$

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Construct a DFA that accepts all strings from the language $L = {\text{strings with size multiples of 2 or 3}}$ where $\Sigma = {a}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solution</th>
</tr>
</thead>
</table>
| • Language $L_1 = \{\text{strings with size multiples of 2}\}$  
• Language $L_2 = \{\text{strings with size multiples of 3}\}$ |

![DFA Diagram]

DFA Diagram:  
- Start state $p_0$  
- Accept state $q_0$, $q_1$, $q_2$  
- Transitions: $a$ from $p_0$ to $p_1$, $q_0$ to $q_1$, $q_1$ to $q_2$
Construct DFA for $L_1 \cup L_2$

Solution (continued)

- Language $L_1 \cup L_2 = \{\text{strings with size multiples of 2 and 3}\}$

---

Start

Language $L_1 \cup L_2 = \{\text{strings with size multiples of 2 and 3}\}$
Construct DFA for \(L_1 \cup L_2\)

<table>
<thead>
<tr>
<th>Union</th>
</tr>
</thead>
</table>
| • Let \(M_1\) accept \(L_1\), where \(M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)\)  
  Let \(M_2\) accept \(L_2\), where \(M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)\)  
• Let \(M\) accept \(L_1 \cup L_2\), where \(M = (Q, \Sigma, \delta, q, F)\). Then  
  \(Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}\)  
  \(\Sigma = \Sigma_1 \cup \Sigma_2\)  
  \(\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))\)  
  \(\forall (r_1, r_2) \in Q, a \in \Sigma\)  
  \(q_0 = (q_1, q_2)\)  
  \(F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}\) |
Construct DFA for $L_1 \cap L_2$

**Problem**
- Construct a DFA that accepts all strings from the language $L = \{\text{strings with size multiples of 2 and 3}\}$ where $\Sigma = \{a\}$

**Solution**
- Language $L_1 = \{\text{strings with size multiples of 2}\}$
- Language $L_2 = \{\text{strings with size multiples of 3}\}$
Construct DFA for $L_1 \cap L_2$

Solution (continued)

• Language $L_1 \cap L_2 = \{\text{strings with size multiples of 2 and 3}\}$
Construct DFA for $L_1 \cap L_2$

Intersection

- Let $M_1$ accept $L_1$, where $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$
- Let $M_2$ accept $L_2$, where $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$
- Let $M$ accept $L_1 \cap L_2$, where $M = (Q, \Sigma, \delta, q, F)$. Then
  
  $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$  \quad \triangledown \text{ Cartesian product}

  $\Sigma = \Sigma_1 \cup \Sigma_2$

  $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)) \quad \forall (r_1, r_2) \in Q, a \in \Sigma$

  $q_0 = (q_1, q_2)$

  $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ and } r_2 \in F_2\}$