<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Deterministic Finite Automata (DFA)</td>
</tr>
<tr>
<td>• Regular Languages</td>
</tr>
<tr>
<td>• Regular Expressions</td>
</tr>
<tr>
<td>• Nondeterministic Finite Automata (NFA)</td>
</tr>
<tr>
<td>• Transformations</td>
</tr>
<tr>
<td>• Pumping Lemma</td>
</tr>
</tbody>
</table>
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.**

![Finite Automaton Diagram](image-url)
Multispeed fan

Problem

- Design the logic behind a multispeed fan.
Problem

- Design the logic behind a multispeed fan.

Solution

- Diagram.

- Finite Automaton.

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

- Design the logic behind automatic doors in Walmart.
Automatic doors

Solution

- **Diagram.**

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

- **Finite Automaton.**

```plaintext
neither  left, right

left, right

close  open

neither
```
• 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?

Definition

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

Some strings → Accept
Other strings → Reject
How does a DFA work?

Problem

- Does the DFA accept the string \( bbab \)?

```
<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>a, b</th>
</tr>
</thead>
<tbody>
<tr>
<td>q0</td>
<td>b→q1</td>
<td>a→q2</td>
<td></td>
</tr>
<tr>
<td>q1</td>
<td>b→q1</td>
<td>a→q2</td>
<td></td>
</tr>
<tr>
<td>q2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

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 $bbab$?

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 $aaba$?

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 \textit{aaba}?

Solution

The DFA rejects the string \textit{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?

- **bbab**
  - Start in state $q_0$.
  - Move to $q_1$ on $b$.
  - Move to $q_2$ on $a$.
  - Accept.

- **aaba**
  - Start in state $q_0$.
  - Move to $q_1$ on $a$.
  - Move to $q_2$ on $b$.
  - Reject.
How does a DFA work?

Problem

- What language does the DFA accept?

The DFA accepts the following strings:
- $b, ab, bb, aabbbb, ababababab, \ldots$
- $baa, abaa, ababaaaaaa, \ldots$

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

Problem

- What language does the DFA accept?

```
Problem
- What language does the DFA accept?

Examples
- The DFA accepts the following strings:
  - $b, ab, bb, aabbb, ababababab, \ldots$  ▶ 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) \( M \):

```
  q0
    \( a \)

start -----> q0
```
Construct DFA for $\Sigma = \{a\}$

**Problem**

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

**Problem**

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

**Solution**

- Language $L$: $\phi = \{}$
- Expression: $\phi$
- DFA $M$:

![DFA Diagram]

- Start state $q_0$
- Transition on $a$ from $q_0$ to itself

▷ Empty language
Problem

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

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

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

**Problem**

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

• Construct a DFA that accepts all strings from the language \( L = \{ \epsilon \} \)

Solution

• Language \( L = \{ \epsilon \} \)
• Expression: \( \epsilon \)
• DFA \( M \):

\[
\text{start} \rightarrow q_0 \xrightarrow{a} q_1
\]

\[
q_1 \xrightarrow{\epsilon} q_1
\]
Problem

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

```
Language L: {aaa}
Expression: aaa
DFA M:
```

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

DFA Diagram:
- Initial state: $q_0$
- Final state: $q_4$
- Transitions:
  - $q_0 \xrightarrow{a} q_1$
  - $q_1 \xrightarrow{a} q_2$
  - $q_2 \xrightarrow{a} q_3$
  - $q_3 \xrightarrow{a} q_4$
  - Self-loop on $q_4$
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 $M$:

```
start,q0 -- a -- q1

q0 -- a -- q1
```
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 $L = {\text{strings with odd size}}$</td>
</tr>
</tbody>
</table>
Problem

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

Solution

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

![DFA diagram]

- Start state: $q_0$
- Final state: $q_1$
- Transitions:
  - $a$: $q_0 \rightarrow q_0$
  - $a$: $q_0 \rightarrow q_1$
  - $a$: $q_1 \rightarrow q_1$
  - $a$: $q_1 \rightarrow q_0$
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 $L = {\text{strings of size divisible by 3}}$</td>
</tr>
</tbody>
</table>
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** $L$: $\{\epsilon, aaa, aaaaaa, aaaaaaaaaa, \ldots\}$
- **Expression**: $(aaa)^*$
- **DFA** $M$:

```
states = {q0, q1, q2}
start = q0
accept_states = {q2}
transitions = {
    (q0, a) => q1,
    (q1, a) => q2,
    (q2, a) => q0
}
```
Problem

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

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

Solution

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

```
\begin{align*}
\text{start} & \rightarrow q_0 \quad a \quad \rightarrow q_1 \quad a \quad \rightarrow q_2 \\
q_0 & \rightarrow q_1 \quad a \\
q_1 & \rightarrow q_2 \quad a
\end{align*}
```
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>
Problem

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

Solution

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

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

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

```
<table>
<thead>
<tr>
<th>Start</th>
<th>a</th>
<th>a</th>
<th>a</th>
<th>a</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>q0</td>
<td>q1</td>
<td>q2</td>
<td>q3</td>
<td>q4</td>
<td>q5</td>
</tr>
</tbody>
</table>
```

- Can you think of another approach?
Problem

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

Solution

• 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
<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Construct a DFA that accepts all strings from the language $L = {\text{strings with size } n \text{ where } n \mod 4 = 3}$</td>
</tr>
</tbody>
</table>
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, aaaaaaaaaaaa, \ldots\}
- Expression: $aaa(aa)^*$
- DFA $M$:

```
start   q0    a       q1    a       q2    a       q3    a       q4
       ↘     ↓       ↘     ↓       ↘     ↓       ↘     ↓       ↘
       a     a       a     a       a     a       a
       q4    q3    q2      q1
```

What about strings with size $n$ where $n \mod k = i$?
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 $L$: $\{aaa, aaaaaaaa, aaaaaaaaaaaa, \ldots\}$
- Expression: $aaa(aaaa)^*$
- DFA $M$:

```
start → q0 → a → q1 → a → q2 → a → q3 → a → q4
```
- What about strings with size $n$ where $n \mod k = i$?
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
- $n^2 - 5n + 6 = 0$
- $n \in [4, 37]$
- $n$ is a perfect cube
- $n$ is a prime number
- $n$ 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 $F \subseteq Q$. 


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 \( F \in Q \).
Acceptance and rejection of strings

**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.
What is a regular language?

**Definition**

- We say that a DFA $M$ **accepts** a language $L$ if $L = \{ w \mid M \text{ accepts } w \}$.
- A language is called a **regular language** if some DFA accepts it.
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 $L = {\text{strings with odd number of } b's}$</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 \( \Sigma = \{a, b\} \)

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

<table>
<thead>
<tr>
<th>Solution</th>
</tr>
</thead>
</table>
| • Language \( L \): \{strings with odd number of \( b\)'s\} 
• Expression: \( a^*b(a \mid ba^*b)^* \) or \( a^*ba^*(ba^*ba^*)^* \) 
• DFA \( M \): |
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 $Q = \{q_{even}, q_{odd}\}$
  
  Set of symbols is $\Sigma = \{a, b\}$
  
  Start state is $q_{even}$
  
  Set of accept states is $F = \{q_{even}\}$
  
  Transition function $\delta$ is:

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_{even}$</td>
<td>$q_{even}$</td>
<td>$q_{odd}$</td>
</tr>
<tr>
<td>$q_{odd}$</td>
<td>$q_{odd}$</td>
<td>$q_{even}$</td>
</tr>
</tbody>
</table>
Construct DFA for $\Sigma = \{a, b\}$

Problem

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

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

**Solution**

**States**

- \( q_b \): DFA is in this state if the last symbol read was \( b \), but the substring \( bab \) has not been read.
- \( q_{ba} \): DFA is in this state if the last two symbols read were \( ba \), but the substring \( bab \) has not been read.
- \( q_{bab} \): DFA is in this state if the substring \( bab \) has been read in the input string.
- \( q \): In all other cases, the DFA is in this state.
Problem
- Construct a DFA that accepts all strings from the language
  \( L = \{ \text{strings containing } bab \} \)

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

\[
\begin{array}{c}
\text{start} \\
\rightarrow q \\
\rightarrow b \\
\rightarrow a \\
\rightarrow qba \\
\rightarrow b \\
\rightarrow Qbab \\
\end{array}
\]

- Transition labels:
  - \( a \rightarrow q \rightarrow qba \rightarrow Qbab \)
  - \( b \rightarrow qb \rightarrow qba \rightarrow Qbab \)
  - \( a \rightarrow qba \rightarrow Qbab \)
  - \( b \rightarrow b \rightarrow Qbab \)

- \( q \) is the start state.
- \( Qbab \) is the accept state.
Construct DFA for $\Sigma = \{a, b\}$

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

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

<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.

- **Complement.** $\overline{L_1} = \{ x \mid x \in \Sigma^* \text{ and } x \notin L_1 \}$.
- **Union.** $L_1 \cup L_2 = \{ x \mid x \in L_1 \text{ or } x \in L_2 \}$.
- **Intersection.** $L_1 \cap L_2 = \{ x \mid x \in L_1 \text{ and } x \in L_2 \}$.
- **Concatenation.** $L_1 \cdot L_2 = \{ xy \mid x \in L_1 \text{ and } y \in L_2 \}$.
- **Star.** $L_1^* = \{ x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in L_1 \}$. 
Construct DFA for $L_1 \cup L_2$

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

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

```
\begin{tikzpicture}[node distance=2cm, thick, main/.style = {draw, circle}]
  \node[main, initial] (p0) {$p_0$};
  \node[main, right of=p0] (p1) {$p_1$};
  \node[main, below of=p1] (p2) {$p_2$};
  \node[main, left of=p0] (q0) {$q_0$};
  \node[main, right of=q0] (q1) {$q_1$};
  \node[main, right of=q1] (q2) {$q_2$};

  \draw[->, above] (p0) to node {$a$} (p1);
  \draw[->, below] (p1) to node {$a$} (p2);
  \draw[<-, right] (p0) to node {$a$} (p1);
  \draw[<-, above right] (p1) to node {$a$} (p2);

  \draw[->, above] (q0) to node {$a$} (q1);
  \draw[->, below] (q1) to node {$a$} (q2);
  \draw[<-, right] (q0) to node {$a$} (q1);
  \draw[<-, above right] (q1) to node {$a$} (q2);
\end{tikzpicture}
```
Solution (continued)

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

**Union**

- 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
  
  $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$  \(\triangleright\) Cartesian product
  
  $\Sigma = \Sigma_1 \cup \Sigma_2$
  
  $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)) \quad \forall (r_1, r_2) \in Q, a \in \Sigma$
  
  $q_0 = (q_1, q_2)$
  
  $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$
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 $L_2 = \{\text{strings with size multiples of 3}\}$

![Diagram of DFA](image-url)
Solution (continued)

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

![Diagram of DFA for $L_1 \cap L_2$](attachment:image.png)
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
  - \( Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\} \) ▶ Cartesian product
  - \( \Sigma = \Sigma_1 \cup \Sigma_2 \)
  - \( \delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)) \) \( \forall (r_1, r_2) \in Q, a \in \Sigma \)
  - \( q_0 = (q_1, q_2) \)
  - \( F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ and } r_2 \in F_2\} \)
Problems for practice

Assume $\Sigma = \{a, b\}$ unless otherwise mentioned.

Construct DFA’s for the following languages and generalize:

1. $L = \{w \mid |w| = 2\}$
2. $L = \{w \mid |w| \leq 2\}$
3. $L = \{w \mid |w| \geq 2\}$
4. $L = \{w \mid n_a(w) = 2\}$
5. $L = \{w \mid n_a(w) \leq 2\}$
6. $L = \{w \mid n_a(w) \geq 2\}$
7. $L = \{w \mid n_a(w) \mod 3 = 1\}$
8. $L = \{w \mid n_a(w) \mod 2 = 0 \text{ and } n_b(w) \mod 2 = 0\}$
9. $L = \{w \mid n_a(w) \mod 3 = 2 \text{ and } n_b(w) \mod 2 = 1\}$
10. $L = \{w \mid n_a(w) \mod 5 = 3, n_b(w) \mod 3 = 2, \text{ and } n_c(w) \mod 2 = 1\}$ for $\Sigma = \{a, b, c\}$
11. $L = \{w \mid n_a(w) \mod 3 \geq n_b(w) \mod 2\}$
Problems (continued)

- \( L = \{b \mid \text{binary number } b \mod 3 = 1\} \) for \( \Sigma = \{0, 1\} \)
- \( L = \{t \mid \text{ternary number } t \mod 4 = 3\} \) for \( \Sigma = \{0, 1, 2\} \)
- \( L = \{w \mid \text{w starts with } a\} \)
- \( L = \{w \mid \text{w contains } a\} \)
- \( L = \{w \mid \text{w ends with } a\} \)
- \( L = \{w \mid \text{w starts with } ab\} \)
- \( L = \{w \mid \text{w contains } ab\} \)
- \( L = \{w \mid \text{w ends with } ab\} \)
- \( L = \{w \mid \text{w starts with } a \text{ and ends with } b\} \)
- \( L = \{w \mid \text{w starts and ends with different symbols}\} \)
- \( L = \{w \mid \text{w starts and ends with the same symbol}\} \)
- \( L = \{w \mid \text{every } a \text{ in } w \text{ is followed by a } b\} \)
- \( L = \{w \mid \text{every } a \text{ in } w \text{ is never followed by a } b\} \)
Problems for practice

Problems (continued)

- \( L = \{ w \mid \text{every } a \text{ in } w \text{ is followed by } bb \} \)
- \( L = \{ w \mid \text{every } a \text{ in } w \text{ is never followed by } bb \} \)
- \( L = \{ w \mid w = a^m b^n \text{ for } m, n \geq 1 \} \)
- \( L = \{ w \mid w = a^m b^n \text{ for } m, n \geq 0 \} \)
- \( L = \{ w \mid w = a^m b^n c^\ell \text{ for } m, n, \ell \geq 1 \text{ for } \Sigma = \{a, b, c\} \}
- \( L = \{ w \mid w = a^m b^n c^\ell \text{ for } m, n, \ell \geq 0 \text{ for } \Sigma = \{a, b, c\} \}
- \( L = \{ w \mid \text{second symbol from left end of } w \text{ is } a \} \)
- \( L = \{ w \mid \text{second symbol from right end of } w \text{ is } a \} \)
- \( L = \{ w \mid w = a^3 b x a^3 \text{ such that } x \in \{a, b\}^* \} \)
Nondeterministic Finite Automata (NFA)
### Example NFA’s

#### Examples

![NFA Diagram](image)

<table>
<thead>
<tr>
<th>Difference</th>
<th>DFA</th>
<th>NFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple transitions</td>
<td>1 exiting arrow</td>
<td>≥ 0 exiting arrows</td>
</tr>
<tr>
<td>Epsilon transitions</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Missing transitions</td>
<td>No missing transitions</td>
<td>Missing transitions mean transitions to sink/reject state</td>
</tr>
<tr>
<td>Intuition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| **Nondeterministic computation = Parallel computation**  
(NFA searches all possible paths in a graph to the accept state) |
| • When NFA has multiple choices for the same input symbol,  
think of it as a process forking multiple processes for parallel  
computation. |
| • A string is accepted if any of the parallel processes accepts the  
string. |
| **Nondeterministic computation = Tree of possibilities**  
(NFA magically guesses a right path to the accept state) |
| • Root of the tree is the start of the computation. |
| • Every branching point is the decision-making point consisting  
of multiple choices. |
| • Machine accepts a string if any of the paths from the root of  
the tree to a leaf leads to an accept state. |
Why care for NFA’s?

Uses of NFA’s

- Constructing NFA’s is easier than directly constructing DFA’s for many problems.
  
Hence, construct NFA’s and then convert them to DFA’s.

- NFA’s are easier to understand than DFA’s.
Construct NFA for $\Sigma = \{a\}$

**Problem**
- Construct a NFA that accepts all strings from the language $L = \{\text{strings of size multiples of 2 or 3 or 5}\}$

**Solution**

What is the equivalent DFA for solving the problem?
Regular Expressions
### Example

- **Arithmetic expression.**
  \[(5 + 3) \times 4 = 32 = \text{Number}\]
- **Regular expression.**
  \[(a \cup b)a^* = \{a, b, aa, ba, aaaa, baaa, \ldots\} = \text{Regular language}\]

### Application

- **Regular expressions in Linux.**
  Used to find patterns in filenames, file content etc.
  Used in Linux tools such as awk, grep, and Perl
### Definition

- The following are **regular expressions**.
  \[ \epsilon, \phi, a \in \Sigma. \]
- If \( R_1 \) and \( R_2 \) are regular expressions, then the following are **regular expressions**.
  
  - (Union.) \( R_1 \cup R_2 \)
  
  - (Concatenation.) \( R_1 \circ R_2 \)
  
  - (Kleene star.) \( R_1^* \)
### Examples

<table>
<thead>
<tr>
<th>Regular language</th>
<th>Regular expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>{}</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td>{\epsilon}</td>
<td>(\epsilon)</td>
</tr>
<tr>
<td>{a}</td>
<td>(a)</td>
</tr>
<tr>
<td>{a, b}</td>
<td>(a \cup b)</td>
</tr>
<tr>
<td>{a} {b}</td>
<td>(ab)</td>
</tr>
<tr>
<td>{a} *</td>
<td>(a^*)</td>
</tr>
<tr>
<td>{a} * = {\epsilon, a, aa, aaa, \ldots}</td>
<td>((aab)^*(a \cup ab))</td>
</tr>
<tr>
<td>{aab} * {a, ab}</td>
<td>((aa \cup bb \cup (a \cup b)(aa)^<em>(ab \cup ba)))^</em></td>
</tr>
</tbody>
</table>

### Equality

- Two regular languages are equal if they describe the same regular language. E.g.:

\[
(a^*b^*)^* = (a \cup b)^*ab(a \cup b)^* \cup b^*a^* = (a \cup b)^* = \Sigma^*
\]
Examples

Let $\Sigma = a \cup b$, $R^+ = RR^*$, and $R^k = \underbrace{R \cdots R}_{k \text{ times}}$

- $L = \{w \mid |w| = 2\}$
  $R = \Sigma\Sigma$

- $L = \{w \mid |w| \leq 2\}$
  $R = \epsilon \cup \Sigma \cup \Sigma\Sigma$

- $L = \{w \mid |w| \geq 2\}$
  $R = \Sigma\Sigma\Sigma^*$

- $L = \{w \mid n_a(w) = 2\}$
  $R = b^*ab^*ab^*$

- $L = \{w \mid n_a(w) \leq 2\}$
  $R = \epsilon \cup b^*ab^* \cup b^*ab^*ab^*$