Theory of Computation
(Finite Automata)

Pramod Ganapathi
Department of Computer Science
State University of New York at Stony Brook

January 24, 2021
<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>• Non-Regular Languages</td>
</tr>
</tbody>
</table>
Deterministic Finite Automata (DFA)
## Electric bulb

### Problem

- Design the logic behind an electric bulb.

### Solution

- **Diagram.**
- **Analysis.**
- **States**
  
  - `nolight`
  - `light`

- **Input**
  
  - `off`
  - `on`
## Electric bulb

### Problem
- Design the logic behind an electric bulb.

### Solution
- **Diagram.**

- **Analysis.**
  
  States = \{nolight, light\}, Input = \{off, on\}

- **Finite Automaton.**

![Finite Automaton Diagram]

- Diagram of the finite automaton for an electric bulb showing states and transitions.
Problem

- Design the logic behind a multispeed fan.
## Multispeed fan

### Problem
- Design the logic behind a multispeed fan.

### Solution
- **Diagram.**

  ![Diagram](image)

- **Finite Automaton.**

  ![Finite Automaton](image)

- **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.

```
neither       left, right

left, right

close         open

neither
```
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 = English language = \{milk, food, sleep, \ldots\} ▶ Accept
- Not in language = \{zffgb, cdcapqw, \ldots\} ▶ Reject
What is a decision problem?

Some strings → Accept

Other strings → Reject
How does a DFA work?

Problem

- Does the DFA accept the string $bbab$?

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 $aaba$?

![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?

The DFA diagram shows states labeled $q_0$, $q_1$, and $q_2$. The input strings are $bbab$ and $aaba$. The DFA accepts $bbab$ and rejects $aaba$. The transitions are labeled with $a$, $b$, and $a,b$, indicating the valid inputs for each state transition.
**How does a DFA work?**

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>● What language does the DFA accept?</td>
</tr>
</tbody>
</table>

### DFA Diagram

- **Start State:** $q_0$
- States: $q_0$, $q_1$, $q_2$
- Transitions:
  - $a$: $q_0 \rightarrow q_1$
  - $b$: $q_0 \rightarrow q_2$
  - $a$: $q_1 \rightarrow q_2$
  - $b$: $q_2 \rightarrow q_1$
- **Accepting States:** $q_2$

### Examples

- The DFA accepts the following strings:
  - $b$, $ab$, $bb$, $aabb$, $ababababab$, ...
  - Ends with $b$

- The DFA accepts the following strings:
  - $baa$, $abaa$, $ababaaaaaa$, ...
  - Ends with $b$ followed by even $a$'s

- The DFA rejects the following strings:
  - $a$, $ba$, $babaaa$, ...

# How does a DFA work?

## Problem

- What language does the DFA accept?

![DFA Diagram]

- The DFA accepts the following strings:
  - `b, ab, bb, aabbbb, ababababab, ...`  
  - `baa, abaa, ababaaaaaa, ...`  
  - Ends with `b` and `b` followed by even `a`'s

- The DFA rejects the following strings:
  - `a, ba, babaaa, ...`

- What language does the DFA accept?
Construct DFA for $\Sigma = \{a\}$

Problem

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

$$\begin{align*}
&\quad a \\
\text{start} &\rightarrow q_0
\end{align*}$$
Construct DFA for $\Sigma = \{a\}$

Problem

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

<table>
<thead>
<tr>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Language $L$: $\phi = {}$ ➤ Empty language</td>
</tr>
<tr>
<td>• Expression: $\phi$</td>
</tr>
<tr>
<td>• DFA $M$:</td>
</tr>
</tbody>
</table>

![DFA Diagram](image-url)
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 $L = \{a, aa, aaa, aaaa, \ldots\}$

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

```
  start → q0 ← q1
       ↓    ↓    ↓
        a  a  a
```
Construct DFA for $\Sigma = \{a\}$

**Problem**

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

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

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

![DFA Diagram](image)
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>
Problem

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

Solution

- Language \( L: \{aaa\} \)
- Expression: \( aaa \)
- DFA \( M: \)

```
start ----> q0 ----> q1 ----> q2 ----> q3 ----> q4
```

\( a \) transitions are shown on the diagram.
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 $L = {\text{strings with even size}}$</td>
</tr>
</tbody>
</table>
Problem

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

Solution

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

```plaintext
start -----> q0 -----> q1
         \^ a \\
         \ | \\
         \ v a
```

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>
Construct DFA for $\Sigma = \{a\}$

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 $M$:

```
start ----> q0 ----> q1
```

- Transition:
  - $q_0 \xrightarrow{a} q_1$
  - $q_1 \xrightarrow{a} q_0$

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 $M$:

  ![DFA Diagram]

- DFA Diagram:
  - $q_0$: Start state
  - $q_1$: Second state
  - $q_2$: Third state
  - Transitions:
    - $a$: $q_0 \rightarrow q_1, q_1 \rightarrow q_2$
    - $a$: $q_2 \rightarrow q_0$

Problem

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

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: $(a \cup aa)(aaa)^*$
- DFA $M$:

```
q_0 \rightarrow a \rightarrow q_1 \rightarrow a \rightarrow q_2
```

DFA Diagram:
Construct DFA for $\Sigma = \{a\}$

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

```
q0 -> q1 -> q2 -> q3 -> q4 -> q5
```

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

  ![DFA Diagram]

- 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 |
Construct DFA for $\Sigma = \{a\}$

### Problem
- Construct a DFA that accepts all strings from the language $L = \{\text{strings with size } n \text{ where } n \mod 4 = 2\}$
### 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 $L = {\text{strings with size } n \text{ where } n \mod 4 = 2}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Language $L$: ${aaa, aaaaaaaa, aaaaaaaaaaaaa, \ldots}$</td>
</tr>
<tr>
<td>• Expression: $aa(aaaa)^*$</td>
</tr>
<tr>
<td>• DFA $M$:</td>
</tr>
</tbody>
</table>

![DFA Diagram](attachment:image.png)

• 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
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 or recognizes it.

<table>
<thead>
<tr>
<th>Definition</th>
</tr>
</thead>
</table>
| • 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 or recognizes 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 $L = {\text{strings with odd number of } b'\text{s}}$</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 $L = \{\text{strings with odd number of } b\text{'s}\}$

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

![DFA Diagram]

- Start state: $q_{even}$
- Accept state: $q_{odd}$
- Transitions:
  - $a \rightarrow q_{even}$
  - $b \rightarrow q_{odd}$
  - $b \rightarrow q_{even}$
  - $a \rightarrow q_{odd}$
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 $F = \{q_{\text{odd}}\}$
  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>
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>
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: $(a*b^+aa)^*bab(a \cup b)^*$
- DFA $M$:

![DFA Diagram]

Start state $q$, states $q_b$, $q_{ba}$, and $q_{bab}$. Transitions:
- $q$ to $q_b$ on $b$ and $a$ to $q_{ba}$ on $b$.
- $q_b$ to $q_{ba}$ on $a$.
- $q_{ba}$ to $q_{bab}$ on $a$ and $b$.
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 $Q = {q, qb, qba, qbab}$</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 $F = {qbab}$</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>$qb$</td>
</tr>
<tr>
<td>$qb$</td>
<td>$qba$</td>
<td>$qb$</td>
</tr>
<tr>
<td>$qba$</td>
<td>$q$</td>
<td>$qbab$</td>
</tr>
<tr>
<td>$qbab$</td>
<td>$qbab$</td>
<td>$qbab$</td>
</tr>
</tbody>
</table>
Closure properties of regular languages

Properties

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.** $L_1^* = \{ x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in L_1 \}$. 
# Closure properties for languages

<table>
<thead>
<tr>
<th>Language</th>
<th>$L_1 \cup L_2$</th>
<th>$L_1 \cap L_2$</th>
<th>$L'$</th>
<th>$L_1 L_2$</th>
<th>$L^*$</th>
<th>$L^R$</th>
<th>$L^T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DCFL</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>CFL</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Recursive</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>R.E.</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- $L_1 \cup L_2$ = Union of $L_1$ and $L_2$
- $L_1 \cap L_2$ = Intersection of $L_1$ and $L_2$
- $L'$ = Complement of $L$
- $L_1 L_2$ = Concatenation of $L_1$ and $L_2$
- $L^*$ = Powers of $L$
- $L^R$ = Reverse of $L$
- $L^T$ = Finite transduction of $L$ (may include: intersection/shuffle/perfect-shuffle/quotient with arbitrary regular languages)
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 $L_2 = \{\text{strings with size multiples of 3}\}$

![DFA Diagram]

- DFA for $L_1$:
  - Start state: $p_0$
  - Final state: $p_1$
  - Transition on $a$: $p_0 \rightarrow p_0$, $p_0 \rightarrow p_1$, $p_1 \rightarrow p_1$

- DFA for $L_2$:
  - Start state: $q_0$
  - Final state: $q_2$
  - Transition on $a$: $q_0 \rightarrow q_1$, $q_1 \rightarrow q_2$, $q_2 \rightarrow q_2$
Solution (continued)

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

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

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

- From $p_0$, on input $a$, go to $p_1$.
- From $p_1$, on input $a$, go back to $p_0$.

- From $q_0$, on input $a$, go to $q_1$.
- From $q_1$, on input $a$, go to $q_2$.
- From $q_2$, on input $a$, go back to $q_0$. 

Notes:
- The DFA transitions are based on the input $a$.
- The start state is $p_0$.
- The DFA accepts strings that are multiples of both 2 and 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 $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
- Let $M_2$ accept $L_2$, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$
- Let $M$ accept $L_1 \cap L_2$, where $M = (Q, \Sigma, \delta, q, F)$. Then
  - $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$  
    $\triangleright$ Cartesian product
  - $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)) \quad \forall (r_1, r_2) \in Q, a \in \Sigma$
  - $q_0 = (q_1, q_2)$
  - $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ and } r_2 \in F_2\}$
Assume $\Sigma = \{a, b\}$ unless otherwise mentioned.

Construct DFA’s for the following languages and generalize:

- $L = \{w \mid |w| = 2\}$
- $L = \{w \mid |w| \leq 2\}$
- $L = \{w \mid |w| \geq 2\}$
- $L = \{w \mid n_a(w) = 2\}$
- $L = \{w \mid n_a(w) \leq 2\}$
- $L = \{w \mid n_a(w) \geq 2\}$
- $L = \{w \mid n_a(w) \mod 3 = 1\}$
- $L = \{w \mid n_a(w) \mod 2 = 0 \text{ and } n_b(w) \mod 2 = 0\}$
- $L = \{w \mid n_a(w) \mod 3 = 2 \text{ and } n_b(w) \mod 2 = 1\}$
- $L = \{w \mid n_a(w) \mod 5 = 3, n_b(w) \mod 3 = 2, \text{ and } n_c(w) \mod 2 = 1\}$ for $\Sigma = \{a, b, c\}$
- $L = \{w \mid n_a(w) \mod 3 \geq n_b(w) \mod 2\}$
• $L = \{b \mid \text{binary number } b \mod 3 = 1\}$ for $\Sigma = \{0, 1\}$
• $L = \{t \mid \text{ternary number } t \mod 4 = 3\}$ for $\Sigma = \{0, 1, 2\}$
• $L = \{w \mid w \text{ starts with } a\}$
• $L = \{w \mid w \text{ contains } a\}$
• $L = \{w \mid w \text{ ends with } a\}$
• $L = \{w \mid w \text{ starts with } ab\}$
• $L = \{w \mid w \text{ contains } ab\}$
• $L = \{w \mid w \text{ ends with } ab\}$
• $L = \{w \mid w \text{ starts with } a \text{ and ends with } b\}$
• $L = \{w \mid w \text{ starts and ends with different symbols}\}$
• $L = \{w \mid w \text{ starts and ends with the same symbol}\}$
• $L = \{w \mid \text{every } a \text{ in } w \text{ is followed by a } b\}$
• $L = \{w \mid \text{every } a \text{ in } w \text{ is never followed by a } b\}$
Problems for practice

Problems (continued)

- $L = \{w \mid$ every $a$ in $w$ is followed by $bb\}$
- $L = \{w \mid$ every $a$ in $w$ is never followed by $bb\}$
- $L = \{w \mid w = a^m b^n$ for $m, n \geq 1\}$
- $L = \{w \mid w = a^m b^n$ for $m, n \geq 0\}$
- $L = \{w \mid w = a^m b^n c^\ell$ for $m, n, \ell \geq 1\}$ for $\Sigma = \{a, b, c\}$
- $L = \{w \mid w = a^m b^n c^\ell$ for $m, n, \ell \geq 0\}$ for $\Sigma = \{a, b, c\}$
- $L = \{w \mid$ second symbol from left end of $w$ is $a\}$
- $L = \{w \mid$ second symbol from right end of $w$ is $a\}$
- $L = \{w \mid w = a^3 b x a^3$ such that $x \in \{a, b\}^*\}$
Two machines or computational models are **computationally equivalent** if they accept/recognize the same language.

The following models are computationally equivalent: DFA, regular expressions, NFA, and regular grammars.
Closure properties for languages

<table>
<thead>
<tr>
<th>Language</th>
<th>$L_1 \cup L_2$</th>
<th>$L_1 \cap L_2$</th>
<th>$\overline{L}$</th>
<th>$L_1 \circ L_2$</th>
<th>$L^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFA</td>
<td>Easy</td>
<td>Easy</td>
<td>Easy</td>
<td>Hard</td>
<td>Hard</td>
</tr>
<tr>
<td>Regex</td>
<td>Easy</td>
<td>Hard</td>
<td>Hard</td>
<td>Easy</td>
<td>Easy</td>
</tr>
<tr>
<td>NFA</td>
<td>Easy</td>
<td>Hard</td>
<td>Hard</td>
<td>Easy</td>
<td>Easy</td>
</tr>
</tbody>
</table>

- $L_1 \cup L_2 =$ Union of $L_1$ and $L_2$
- $L_1 \cap L_2 =$ Intersection of $L_1$ and $L_2$
- $\overline{L} =$ Complement of $L$
- $L_1 \circ L_2 =$ Concatenation of $L_1$ and $L_2$
- $L^* =$ Powers of $L$
Regular Expressions
Example

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

Application

- **Regular expressions in Linux.**
  Used to find patterns in filenames, file content etc.
  Used in Linux tools such as awk, grep, and Perl.
What is a regular expression?

**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 ∪ b</td>
</tr>
<tr>
<td>{a}{b}</td>
<td>ab</td>
</tr>
<tr>
<td>{a}^*</td>
<td>a^*</td>
</tr>
<tr>
<td>{aab}^*{a, ab}</td>
<td>(aab)^*(a ∪ ab)</td>
</tr>
<tr>
<td>({aa, bb} ∪ {a, b}{aa}^<em>{ab, ba})^</em></td>
<td>(aa ∪ bb ∪ (a ∪ b)(aa)^<em>(ab ∪ ba))^</em></td>
</tr>
</tbody>
</table>

Equality

- Two regular expressions are equal if they describe the same regular language. E.g.:
  \((a^*b^*)^* = (a ∪ b)^*ab(a ∪ b)^* ∪ b^*a^* = (a ∪ b)^* = \Sigma^*\)
Examples

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Beware of $\phi$ and $\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suppose $R$ is a regular expression.</td>
</tr>
<tr>
<td>- $R \cup \phi = R$</td>
</tr>
<tr>
<td>- $R \circ \epsilon = R$</td>
</tr>
<tr>
<td>- $R \cup \epsilon$ may not equal $R$ (e.g.: $R = 0$, $L(R) = {0}$, $L(R \cup \epsilon) = {0, \epsilon}$)</td>
</tr>
<tr>
<td>- $R \circ \phi$ may not equal $R$ (e.g.: $R = 0$, $L(R) = {0}$, $L(R \circ \phi) = \phi$)</td>
</tr>
</tbody>
</table>
Suppose $R_1, R_2, R_3$ are regular expressions. Then

- $R_1 \phi = \phi R_1 = \phi$
- $R_1 \epsilon = \epsilon R_1 = R_1 \cup \phi = \phi \cup R_1 = R_1$
- $R_1 \cup R_1 = R_1$
- $R_1 \cup R_2 = R_2 \cup R_1$
- $R_1 (R_2 \cup R_3) = R_1 R_2 \cup R_1 R_3$
- $(R_1 \cup R_2) R_3 = R_1 R_3 \cup R_2 R_3$
- $R_1 (R_2 R_3) = (R_1 R_2) R_3$
- $\phi^* = \epsilon$
- $(\epsilon \cup R_1)^* = (\epsilon \cup R_1)^+ = R_1^*$
- $R_1^* (\epsilon \cup R_1) = (\epsilon \cup R_1) R_1^* = R_1^*$
- $R_1^* R_2 \cup R_2 = R_1^* R_2$
- $R_1 (R_2 R_1)^* = (R_1 R_2)^* R_1$
- $(R_1 \cup R_2)^* = (R_1 \ast R_2)^* R_1^* = (R_2 R_1)^* R_2^*$
Construct a regex for $\Sigma = \{a, b\}$

Problem

- Construct a regular expression to describe the language $L = \{w \mid n_a(w) \text{ is odd}\}$
Construct a regular expression to describe the language

\[ L = \{ w \mid n_a(w) \text{ is odd} \} \]

Solution

- **Incorrect expressions.**
  - \( b^*ab^*(ab^*a)^*b^* \)  
  - \( b^*a(b^*ab^*ab^*)^* \)

- **Correct expressions.**
  - \( b^*ab^*(b^*ab^*ab^*)^* \)
  - \( b^*ab^*(ab^*ab^*)^* \)
  - \( b^*a(b^*ab^*a)^*b^* \)
  - \( b^*a(b \cup ab^*a)^* \)
  - \( (b \cup ab^*a)^*ab^* \)
Construct a regex for $\Sigma = \{a, b\}$

**Problem**

- Construct a regular expression to describe the language $L = \{w \mid w$ ends with $b$ and does not contain $aa\}$
Construct a regex for $\Sigma = \{a, b\}$

**Problem**
- Construct a regular expression to describe the language $L = \{w \mid w$ ends with $b$ and does not contain $aa\}$

**Solution**
- A string not containing $aa$ means that every $a$ in the string:
  - is immediately followed by $b$, or
  - is the last symbol of the string
- Each string in the language has to end with $b$.
- Hence, every $a$ in each string of the language is immediately followed by $b$
- Regular expression is: $(b \cup ab)^+$
Problem

- **Identifiers** are the names you supply for variables, types, functions, and labels.
- Construct a regular expression to recognize the **identifiers** in the C programming language i.e., $L = \{\text{identifiers in C}\}$
Problem
- **Identifiers** are the names you supply for variables, types, functions, and labels.
- Construct a regular expression to recognize the identifiers in the C programming language i.e., \( L = \{ \text{identifiers in C} \} \)

Solution
- C identifier = FirstLetter OtherLetters
  FirstLetter = English letter or underscore
  OtherLetters = Alphanumeric letters or underscore
- Let \( L = \{ a, \ldots, z, A, \ldots, Z \} \) and \( D = \{ 0, 1, \ldots, 9 \} \)
- Regular expression is:
  \( R = \text{FirstLetter} \circ \text{OtherLetters} \)
  FirstLetter = \( (L \cup _) \)
  OtherLetters = \( (L \cup D \cup _) \)
Construct a regex to recognize decimals in C

Problem

- Construct a regular expression to recognize the **decimal numbers** in the C programming language i.e.,
  \( L = \{ \text{decimal numbers in C} \} \)
- Examples: 14, +1, −12, 14.3, −.99, 16., 3E14, −1.00E2, 4.1E−1, and .3E + 2
Construct a regex to recognize decimals in C

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
</table>
| • Construct a regular expression to recognize the **decimal numbers** in the C programming language i.e., $L = \{\text{decimal numbers in C}\}$  
  Examples: 14, +1, −12, 14.3, −.99, 16., 3E14, −1.00E2, 4.1E−1, and .3E + 2 |

<table>
<thead>
<tr>
<th>Solution</th>
</tr>
</thead>
</table>
| • C decimal number = Sign Decimals Exponent  
  Let $D = \{0, 1, \ldots, 9\}$  
  Regular expression is: $R = \text{Sign} \circ \text{Decimals} \circ \text{Exponent}$  
  Sign = $(+ \cup − \cup \epsilon)$  
  Decimals = $(D^+ \cup D^+.D^* \cup D^*.D^+)$  
  Exponent = $(\epsilon \cup E \text{ Sign } D^+)$ |
Nondeterministic Finite Automata (NFA)
### Example NFA's

#### Examples

![NFA Diagrams](image)

#### Differences

<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>(\geq 0) exiting arrows</td>
</tr>
<tr>
<td>Epsilon transitions</td>
<td>(\times)</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>Missing transitions</td>
<td>No missing transitions</td>
<td>Missing transitions mean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>transitions to sink/reject</td>
</tr>
<tr>
<td></td>
<td></td>
<td>state</td>
</tr>
</tbody>
</table>
**Intuition**

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 $\Sigma = \{0, 1\}$

**Problem**
- Construct a NFA that accepts all strings from the language $L = \{\text{strings containing 11 or 101}\}$

**Solution**

- How does the machine work for the input 010110?
- What is the equivalent DFA for solving the problem?
Construct NFA for $\Sigma = \{0, 1\}$

Solution (continued)

Source: Anil Maheshwari and Michiel Smid’s Theory of Computation
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**

```
\begin{tikzpicture}
  \node[state, initial] (1) {};
  \node[state, below of=1, yshift=-1cm] (2) {2};
  \node[state, below of=2, yshift=-1cm] (3) {3};
  \node[state, below of=3, yshift=-1cm] (4) {5};
  \node[state, right of=4, xshift=2cm] (5) {};

  \draw[->] (1) edge [loop above] node {$\epsilon$} (1);
  \draw[->] (1) edge node {$\epsilon$} (2);
  \draw[->] (2) edge [loop above] node {$a$} (2);
  \draw[->] (2) edge node {$a$} (3);
  \draw[->] (3) edge [loop above] node {$a$} (3);
  \draw[->] (3) edge node {$a$} (4);
  \draw[->] (4) edge [loop above] node {$a$} (4);
  \draw[->] (4) edge node {$a$} (5);
  \draw[->] (5) edge [loop above] node {$a$} (5);

\end{tikzpicture}
```

- What is the equivalent DFA for solving the problem?
What is a nondeterministic finite automaton (NFA)?

- Nondeterministic = Event paths cannot be determined precisely
- Finite = Finite and small amount of space used
- Automaton = Computing machine

**Definition**

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

1. $Q$: A finite set (set of states). $\triangleright$ **Space** (computer memory)
2. $\Sigma$: A finite set (alphabet).
3. $\delta : Q \times (\Sigma \cup \epsilon) \rightarrow P(Q)$ is the transition function, where $P(Q)$ is the power set of $Q$. $\triangleright$ **Time** (computation)
4. $q_0$: The start state (belongs to $Q$).
5. $F$: The set of **accepting/final states**, where $F \in Q$. 

Problem

- Convert the NFA to a DFA.

Source: Anil Maheshwari and Michiel Smid’s Theory of Computation
Construct DFA for the given NFA

Solution

- NFA $M$ is specified as
  - Set of states is $Q = \{1, 2, 3\}$
  - Set of symbols is $\Sigma = \{a, b\}$
  - Start state is 1
  - Set of accept states is $F = \{1\}$
  - Transition function $\delta$ is:

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>$a$</th>
<th>$b$</th>
<th>$\epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>${3}$</td>
<td>$\phi$</td>
<td>${2}$</td>
</tr>
<tr>
<td>2</td>
<td>${1}$</td>
<td>$\phi$</td>
<td>$\phi$</td>
</tr>
<tr>
<td>3</td>
<td>${2}$</td>
<td>${2, 3}$</td>
<td>$\phi$</td>
</tr>
</tbody>
</table>

- How do you convert this NFA to DFA?
Construct DFA for the given NFA

Solution

• NFA $M$ is specified as
  
  Set of states is $Q = \{1, 2, 3\}$
  
  Set of symbols is $\Sigma = \{a, b\}$
  
  Start state is $1$
  
  Set of accept states is $F = \{1\}$
  
  Transition function $\delta$ is:

  \[
  \begin{array}{c|ccc}
  \delta & a & b & \epsilon \\
  \hline
  1 & \{3\} & \phi & \{2\} \\
  2 & \{1\} & \phi & \phi \\
  3 & \{2\} & \{2, 3\} & \phi \\
  \end{array}
  \]

• How do you convert this NFA to DFA?
  
  If NFA has states $Q$, then construct a DFA with states $P(Q)$. 
Construct DFA for the given NFA

### Solution (continued)

<table>
<thead>
<tr>
<th>State</th>
<th>Transition</th>
<th>New State</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>$a$</td>
<td>$\phi$</td>
</tr>
<tr>
<td>$\phi$</td>
<td>$b$</td>
<td>$\phi$</td>
</tr>
<tr>
<td>${1}$</td>
<td>$a$</td>
<td>${3}$</td>
</tr>
<tr>
<td>${1}$</td>
<td>$b$</td>
<td>$\phi$</td>
</tr>
<tr>
<td>${2}$</td>
<td>$a$</td>
<td>${1, 2}$</td>
</tr>
<tr>
<td>${2}$</td>
<td>$b$</td>
<td>$\phi$</td>
</tr>
<tr>
<td>${3}$</td>
<td>$a$</td>
<td>${2}$</td>
</tr>
<tr>
<td>${3}$</td>
<td>$b$</td>
<td>${2, 3}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>Transition</th>
<th>New State</th>
</tr>
</thead>
<tbody>
<tr>
<td>${1, 2}$</td>
<td>$a$</td>
<td>${1, 2}$</td>
</tr>
<tr>
<td>${1, 2}$</td>
<td>$b$</td>
<td>${1, 2}$</td>
</tr>
<tr>
<td>${1, 3}$</td>
<td>$a$</td>
<td>${1, 3}$</td>
</tr>
<tr>
<td>${1, 3}$</td>
<td>$b$</td>
<td>${1, 3}$</td>
</tr>
<tr>
<td>${2, 3}$</td>
<td>$a$</td>
<td>${2, 3}$</td>
</tr>
<tr>
<td>${2, 3}$</td>
<td>$b$</td>
<td>${2, 3}$</td>
</tr>
<tr>
<td>${1, 2, 3}$</td>
<td>$a$</td>
<td>${1, 2, 3}$</td>
</tr>
<tr>
<td>${1, 2, 3}$</td>
<td>$b$</td>
<td>${1, 2, 3}$</td>
</tr>
</tbody>
</table>
Construct DFA for the given NFA

Solution (continued)

Source: Anil Maheshwari and Michiel Smid’s Theory of Computation
Construct DFA for the given NFA

Solution (continued)

Source: Anil Maheshwari and Michiel Smid’s Theory of Computation
Construct DFA for the given NFA

Convert NFA to DFA

- Let $N = (Q, \Sigma, \delta, q, F)$ be the NFA.
  - Let $M = (Q', \Sigma, \delta', q', F')$ be the DFA. Then
- $Q' = P(Q)$  \hspace{1cm} \triangleright \text{Power set of } Q$
- $q' = C_\epsilon(\{q\})$  \hspace{1cm} \triangleright \text{ } \epsilon\text{-closure of the start state}
- $F' = \{S \in Q' \mid S \cap F \neq \phi\}$  \hspace{1cm} \triangleright S \cap F \neq \phi \text{ means that } S$ contains at least one accept state of $N$
- $\delta' : Q' \times \Sigma \rightarrow Q'$ is defined as follows:
  - For all state $S \in Q'$ and for all letter $a \in \Sigma$,
    $$\delta'(S, a) = \bigcup_{s \in S} C_\epsilon(\delta(s, a))$$
Union of NFA

Source: Margaret Fleck and Sariel Har-Peled’s Notes on Theory of Computation
Concatenation of NFA

Source: Margaret Fleck and Sariel Har-Peled's Notes on Theory of Computation
Star of NFA

Source: Margaret Fleck and Sariel Har-Peled’s Notes on Theory of Computation
Problem

- Construct a NFA for the regular expression \((aa \cup aab)^*b\).
Problem

- Construct a NFA for the regular expression \((aa \cup aab)^*b\).

Solution

Source: John Martin's Introduction to Languages and the Theory of Computation.
Problem

- Construct a NFA for the regular expression \((aab)^*(a \cup aba)^*\).
Problem

Construct a NFA for the regular expression \((aab)^*(a \cup aba)^*\).

Solution

Source: John Martin's Introduction to Languages and the Theory of Computation.
Non-Regular Languages
<table>
<thead>
<tr>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let $\Sigma = {a, b}$ unless mentioned otherwise. Check if the languages are regular or non-regular ($X$):</td>
</tr>
<tr>
<td>- $L = {w \mid w = a^n \text{ and } n \leq 10^{100}}$</td>
</tr>
<tr>
<td>- $L = {w \mid w = a^n \text{ and } n \geq 1}$</td>
</tr>
<tr>
<td>- $L = {w \mid w = a^m b^n \text{ and } m, n \geq 1}$</td>
</tr>
<tr>
<td>- $L = {w \mid w = a^* b^*}$</td>
</tr>
<tr>
<td>- $L = {w \mid w = a^n b^n \text{ and } n \geq 1}$</td>
</tr>
<tr>
<td>- $L = {ww^R \mid</td>
</tr>
<tr>
<td>- $L = {ww^R \mid</td>
</tr>
<tr>
<td>- $L = {w \mid w = w^R \text{ and }</td>
</tr>
<tr>
<td>- $L = {w \mid w = a^{2i+1} b^{3j+2} \text{ and } i, j \geq 1}$</td>
</tr>
<tr>
<td>- $L = {w \mid w = a^n \text{ and } n \text{ is a square}}$</td>
</tr>
<tr>
<td>- $L = {w \mid w = a^n \text{ and } n \text{ is a prime}}$</td>
</tr>
<tr>
<td>- $L = {w \mid w = a^i b^{j^2} \text{ and } i, j \geq 1}$</td>
</tr>
</tbody>
</table>
Regular or non-regular languages

Problems

Let $\Sigma = \{a, b\}$ unless mentioned otherwise. Check if the languages are regular or non-regular ($\times$):

- $L = \{w \mid w = a^n \text{ and } n \leq 10^{100}\}$
- $L = \{w \mid w = a^n \text{ and } n \geq 1\}$
- $L = \{w \mid w = a^m b^n \text{ and } m, n \geq 1\}$
- $L = \{w \mid w = a^* b^*\}$
- $L = \{w \mid w = a^n b^n \text{ and } n \geq 1\}$ ........................................... $\times$
- $L = \{ww^R \mid |w| = 3\}$
- $L = \{ww^R \mid |w| \geq 1\}$ .......................................................... $\times$
- $L = \{w \mid w = w^R \text{ and } |w| \geq 1\}$ .......................................... $\times$
- $L = \{w \mid w = a^{2i+1} b^{3j+2} \text{ and } i, j \geq 1\}$
- $L = \{w \mid w = a^n \text{ and } n \text{ is a square}\}$ ........................................... $\times$
- $L = \{w \mid w = a^n \text{ and } n \text{ is a prime}\}$ ........................................... $\times$
- $L = \{w \mid w = a^i b^{j^2} \text{ and } i, j \geq 1\}$ .......................................... $\times$
### Problems (continued)

- \( L = \{ w \mid n_a(w) = n_b(w) \} \)
- \( L = \{ w \mid n_a(w) \text{ mod } 3 \geq n_b(w) \text{ mod } 5 \} \)
- \( L = \{ w \mid w = a^i b^j \text{ and } j > i \geq 1 \} \)
- \( L = \{ w x w^R \mid x \in \Sigma^* \text{, } |w|, |x| \geq 1 \text{, and } |x| \leq 5 \} \)
- \( L = \{ w x w^R \mid x \in \Sigma^* \text{ and } |w|, |x| \geq 1 \} \)
- \( L = \{ x w w^R y \mid x, y \in \Sigma^* \text{ and } |w|, |x|, |y| \geq 1 \} \)
- \( L = \{ x w w^R \mid x \in \Sigma^* \text{ and } |w|, |x| \geq 1 \} \)
- \( L = \{ w w^R y \mid y \in \Sigma^* \text{ and } |w|, |y| \geq 1 \} \)
Problems (continued)

- \( L = \{ w \mid n_a(w) = n_b(w) \} \) .................................................. \( \times \)
- \( L = \{ w \mid n_a(w) \text{ mod } 3 \geq n_b(w) \text{ mod } 5 \} \)
- \( L = \{ w \mid w = a^i b^j \text{ and } j > i \geq 1 \} \) ........................................ \( \times \)
- \( L = \{ wxw^R \mid x \in \Sigma^*, |w|, |x| \geq 1, \text{ and } |x| \leq 5 \} \) ................. \( \times \)
- \( L = \{ wxw^R \mid x \in \Sigma^* \text{ and } |w|, |x| \geq 1 \} \)
- \( L = \{ xww^Ry \mid x, y \in \Sigma^* \text{ and } |w|, |x|, |y| \geq 1 \} \)
- \( L = \{ xww^R \mid x \in \Sigma^* \text{ and } |w|, |x| \geq 1 \} \) ........................................ \( \times \)
- \( L = \{ ww^Ry \mid y \in \Sigma^* \text{ and } |w|, |y| \geq 1 \} \) ........................................ \( \times \)
How to prove that certain languages are not regular?

### Pumping lemma
- Many languages are not regular.
- **Pumping lemma** is a method to prove that certain languages are not regular.

### Pumping property
- If a language is regular, then it must have the **pumping property**.
- If a language does not have the pumping property, then the language is not regular. ▶ Proof by contraposition

### How to prove languages non-regular using pumping lemma?
- **Proof by contradiction.**
  Assume that the language is regular.
  Show that the language does not have the pumping property.
  Contradiction! Hence, the language has to be non-regular.
Pumping property of regular languages

- Suppose a DFA $M$ with $s$ number of states accepts a very long string $w$ such that $|w| \geq s$ from a language $L$.
- From pigeonhole principle, at least one state is visited twice.
- This implies that the string went through a loop.
Pumping property of regular languages

Observations

- Suppose string $w$ has more characters than the number of states in the DFA, i.e., $|w| \geq s$
- String $w$ can be split into three parts i.e., $w = xyz$ where
  - $x$: string before the first loop
  - $y$: string of the first loop
  - $z$: string after the first loop (might contain loops)
- Loop must appear i.e., $|y| \geq 1$
  ($x$ and $z$ can be empty)
- Loop must appear in the first $s$ characters of $w$ i.e., $|xy| \leq s$
Idea

- An infinite number of strings can be pumped with loop length and they must also be in the language.
- Formally, for all \( i \geq 0 \), \( xy^iz \) must be in the language.
- \( xz, xyz, xyyz, xyyyz \), etc must also belong to the language.
**Theorem**

Suppose $L$ is a regular language over alphabet $\Sigma$. Suppose $L$ is accepted by a finite automaton $M$ having $s$ states. Then, every long string $w \in L$ satisfying $|w| \geq s$ can be split into three strings $w = xyz$ such that the following three conditions are true.

- $|xy| \leq s$.
- $|y| \geq 1$.
- For every $i \geq 0$, the string $xy^iz$ also belongs to $L$. 
Problem

Prove that \( L = \{a^n b^n \mid n \geq 0\} \) is not a regular language.
**Problem**

- Prove that $L = \{a^n b^n \mid n \geq 0\}$ is not a regular language.

**Solution**

- Suppose $L$ is regular. Then it must satisfy pumping property.
- Suppose $w = a^s b^s$.
- Let $w = xyz = a^p a^q a^r b^s$
- where $|xy| \leq s$, $|y| \geq 1$, and $p + q + r = s$.
- Also, $x y^i z$ must belong to $L$ for all $i \geq 0$.
- But, $x y y z$ is not in $L$.
  - Reason: $x y y z = a^p a^q a^q a^r b^s = a^{s+q} b^s \not\in L$.
  - $x y y z$ has more $a$'s than $b$'s.
- Contradiction! Hence, $L$ is not regular.
Problem

Prove that \( L = \{ w \mid n_a(w) = n_b(w) \} \) is not a regular language.

Mistakes

Incorrect solution! Why?

1. If we cannot find a contradiction, that does not prove anything.
2. We must try for all possible values of \( x, y \) such that \( |xy| \leq s \).
3. The chosen string \( (ab)^s \) is a bad string to work on.
Problem

- Prove that \( L = \{ w \mid n_a(w) = n_b(w) \} \) is not a regular language.

Solution

- Suppose \( L \) is regular. Then it must satisfy pumping property.
- Suppose \( w = (ab)^s \).
- Let \( w = xyz = \varepsilon (ab)^1 (ab)^{s-1} \).
- We have \( |xy| \leq s \) and \( |y| \geq 1 \).
- Also, \( xy^iz \) must belong to \( L \) for all \( i \geq 0 \).
- \( xy^iz \) belongs to \( L \) for all \( i \geq 0 \).
- No contradiction! Hence, \( L \) is regular.
**L = \{w \mid n_a(w) = n_b(w)\} is non-regular**

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Prove that (L = {w \mid n_a(w) = n_b(w)}) is not a regular language.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Suppose (L) is regular. Then it must satisfy pumping property.</td>
</tr>
<tr>
<td>• Suppose (w = (ab)^s).</td>
</tr>
<tr>
<td>• Let (w = xyz = \epsilon(ab)^1(ab)^{s-1})</td>
</tr>
<tr>
<td>• We have (</td>
</tr>
<tr>
<td>• Also, (xy^iz) must belong to (L) for all (i \geq 0).</td>
</tr>
<tr>
<td>• (xy^iz) belongs to (L) for all (i \geq 0).</td>
</tr>
<tr>
<td>• No contradiction! Hence, (L) is regular.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mistakes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect solution! Why? Multiple reasons:</td>
</tr>
<tr>
<td>1. If we cannot find a contradiction, that does not prove anything.</td>
</tr>
<tr>
<td>2. We must try for all possible values of (x, y) such that (</td>
</tr>
<tr>
<td>3. The chosen string ((ab)^s) is a bad string to work on.</td>
</tr>
</tbody>
</table>
Problem

- Prove that \( L = \{ w \mid n_a(w) = n_b(w) \} \) is not a regular language.

Solution

- Suppose \( L \) is regular. Then it must satisfy pumping property.
- Suppose \( w = a^s b^s \).
- Let \( w = xyz = \boxed{a^p a^q a^r b^s} \) where \( |xy| \leq s \), \( |y| \geq 1 \), and \( p + q + r = s \).
- Also, \( xy^i z \) must belong to \( L \) for all \( i \geq 0 \).
- But, \( xyyz \) is not in \( L \).
  - Reason: \( xyyz = a^p a^q a^q a^r b^s = a^{s+q} b^s \notin L \).
  - \( xyyz \) has more \( a \)'s than \( b \)'s.
- Contradiction! Hence, \( L \) is not regular.
Problem

• Prove that \( L = \{w \mid n_a(w) = n_b(w)\} \) is not a regular language.

Solution

• Suppose \( L \) is regular. Then it must satisfy pumping property.
• Suppose \( w = a^s b^s \).
• Let \( w = xyz = a^p a^q a^r b^s \)
  where \( |xy| \leq s, |y| \geq 1, \) and \( p + q + r = s \).
• Also, \( xy^i z \) must belong to \( L \) for all \( i \geq 0 \).
• But, \( xyyz \) is not in \( L \).
  Reason: \( xyyz = a^p a^q a^q a^r b^s = a^{s+q} b^s \not\in L \).
  \( xyyz \) has more \( a \)'s than \( b \)'s.
• Contradiction! Hence, \( L \) is not regular.

Takeaway

1. Reduction! Reduce a problem to another. Reuse its solution.
Superset of a non-regular language

Problem

\( \{a^n b^n\} \) is a subset of \( \{w \mid n_a(w) = n_b(w)\} \).

We used the fact that \( \{a^n b^n\} \) is non-regular to prove that \( \{w \mid n_a(w) = n_b(w)\} \) is non-regular.

Is a superset of a non-regular language non-regular?

Solution

\( \Sigma^* \) is a superset of every non-regular language. But, it is regular.
Superset of a non-regular language

Problem

- $\{a^n b^n\}$ is a subset of $\{w \mid n_a(w) = n_b(w)\}$.

We used the fact that $\{a^n b^n\}$ is non-regular to prove that $\{w \mid n_a(w) = n_b(w)\}$ is non-regular.

Is a superset of a non-regular language non-regular?

Solution

- No!
  - $\Sigma^*$ is a superset of every non-regular language.
  - But, it is regular.
\[ L = \{ w \mid n_a(w) = n_b(w) \} \] is non-regular

**Problem**

- Prove that \( L = \{ w \mid n_a(w) = n_b(w) \} \) is not a regular language.

**Solution (without using pumping lemma)**

- Suppose \( L \) is regular.
- We know that \( L' = \{ w \mid w = a^i b^j, i, j \geq 0 \} \) is regular.
- As regular languages are closed under intersection, \( L \cap L' \) must also be regular.
- We see that \( L \cap L' = \{ w \mid w = a^n b^n \text{ and } n \geq 0 \} \).
- But, this language was earlier proved to be non-regular.
- Contradiction! Hence, \( L \) is not regular.
Problem

• Prove that $L = \{ww\}$ is not a regular language.

$L = \{ww\}$ is non-regular
Problem

• Prove that $L = \{ww\}$ is not a regular language.

Solution

• Suppose $L$ is regular. Then it must satisfy pumping property.
• Suppose $ww = a^s a^s$.
• Let $ww = xyz = a^p a^1 a^{s-p-1} a^s$.
• We have $|xy| \leq s$ and $|y| \geq 1$.
• Also, $xy^i z$ must belong to $L$ for all $i \geq 0$.
• But, $xyyz$ is not in $L$.
  Reason: $xyyz = a^p a^1 a^1 a^{s-p-1} a^p = a^{s+1} a^s \notin L$.
  $xyyz$ has odd number of $a$’s.
• Contradiction! Hence, $L$ is not regular.
**Problem**

- Prove that \( L = \{ww\} \) is not a regular language.

**Solution**

- Suppose \( L \) is regular. Then it must satisfy pumping property.
- Suppose \( ww = a^s a^s \).
- Let \( ww = xyz = a^p a^1 a^{s-p-1} a^s \).
- We have \(|xy| \leq s \) and \(|y| \geq 1\).
- Also, \( xy^i z \) must belong to \( L \) for all \( i \geq 0 \).
- But, \( xyyz \) is not in \( L \).
  
  **Reason:** \( xyyz = a^p a^1 a^1 a^{s-p-1} a^p = a^{s+1} a^s \notin L \).
  
  \( xyyz \) has odd number of \( a \)'s.
- Contradiction! Hence, \( L \) is not regular.

**Mistakes**

**Incorrect solution! Why?**

1. We must try all possible values of \( x, y \) such that \(|xy| \leq s\).
2. Try pumping with \( y \in \{a^2, a^4, \ldots\} \) such that \(|y| \leq s\).
Problem

• Prove that $L = \{ww\}$ is not a regular language.
Problem

• Prove that \( L = \{ww\} \) is not a regular language.

Solution

• Suppose \( L \) is regular. Then it must satisfy pumping property.
• Suppose \( ww = a^s b^s a^s b^s \).
• Let \( ww = xyz = \color{blue}a^p \color{red}a^q \color{green}a^r b^s a^s b^s \)
  where \( |xy| \leq s \), \( |y| \geq 1 \), and \( p + q + r = s \).
• Also, \( xy^i z \) must belong to \( L \) for all \( i \geq 0 \).
• But, \( xyyz \) is not in \( L \).
  Reason: \( xyyz = a^p a^q a^q a^r b^s a^s b^s = a^{s+q} b^s a^s b^s \notin L \).
• Contradiction! Hence, \( L \) is not regular.
Problem

• Prove that \( L = \{ w \mid w = a^{n^2} \} \) is not a regular language.
**Problem**

- Prove that \( L = \{ w \mid w = a^{n^2} \} \) is not a regular language.

**Solution**

- Suppose \( L \) is regular. Then it must satisfy pumping property.
- Suppose \( w = a^{s^2} \).
- Let \( w = xyz = a^p a^q a^r a^{s^2-s} \), where \( |xy| \leq s \), \( |y| \geq 1 \), and \( p + q + r = s \).
- Also, \( xy^iz \) must belong to \( L \) for all \( i \geq 0 \).
- But, \( xyyz \) is not in \( L \).
  
  **Reason:** \( xyyz = a^p a^q a^q a^r a^{s^2-s} = a^{s^2+q} \notin L \).
  
  Because, \( s^2 < s^2 + q < (s + 1)^2 \).

- Contradiction! Hence, \( L \) is not regular.
Problem

- Prove that $L = \{w \mid w = a^n, n \text{ is prime}\}$ is not regular.
Problem

- Prove that \( L = \{w \mid w = a^n, n \text{ is prime}\} \) is not regular.

Solution

- Suppose \( L \) is regular. Then it must satisfy pumping property.
- Suppose \( w = a^m \), where \( m \) is prime and \( m \geq s \).
- Let \( w = xyz = \underbrace{a^p}_{\text{blue}} \underbrace{a^q}_{\text{red}} \underbrace{a^r}_{\text{green}} \) where \( |xy| \leq s, |y| \geq 1 \), and \( p + q + r = m \).
- Also, \( xy^i z \) must belong to \( L \) for all \( i \geq 0 \).
- But, \( xy^{m+1} z \) is not in \( L \).
  - Reason: \( xy^{m+1} z = a^p a^q(m+1) a^r = a^{m(q+1)} \notin L \).
- Contradiction! Hence, \( L \) is not regular.
Prove that \( L = \{w \mid w = a^m b^n, m > n\} \) is non-regular.

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Prove that ( L = {w \mid w = a^m b^n, m &gt; n} ) is not regular.</td>
</tr>
</tbody>
</table>
Problem
• Prove that \( L = \{w \mid w = a^m b^n, m > n\} \) is not regular.

Solution
• Suppose \( L \) is regular. Then it must satisfy pumping property.
• Suppose \( w = a^{s+1} b^s \).
• Let \( w = xyz = a^p a^q a^r b^s \)
  where \(|xy| \leq s\), \(|y| \geq 1\), and \(p + q + r = s + 1\).
• Also, \( xy^i z \) must belong to \( L \) for all \( i \geq 0 \).
• But, \( xz \) is not in \( L \). ▶ Pumping down
  Reason: \( xz = a^p a^r b^s = a^{p+r} b^s \notin L \).
  Because, \( p + r \leq s \) i.e., \#a’s is not greater than \#b’s.
• Contradiction! Hence, \( L \) is not regular.
Problem

• Prove that \( L = \{w \mid w = a^m b^n, m \neq n\} \) is not regular.
**Problem**

- Prove that $L = \{w \mid w = a^m b^n, m \neq n\}$ is not regular.

**Solution**

- Suppose $L$ is regular. Then it must satisfy pumping property.
- Suppose $w = a^s b^{s+s!}$.
- Let $w = xyz = a^p a^q a^r b^{s+s!}$ where $|xy| \leq s$, $|y| \geq 1$, and $p + q + r = s$.
- Also, $xy^i z$ must belong to $L$ for all $i \geq 0$.
- But, $xy^i z$ is not in $L$ for some $i$.
  - We pump $a^q$ to get $a^{s+s!} b^{s+s!}$.
  - Reason: $xy^i z = a^p a^{qi} a^r b^{s+s!} = a^{s+(i-1)q} b^{s+s!} \not\in L$.
  - This means $(i-1)q = s! \implies i = s!/q + 1$.
- Contradiction! Hence, $L$ is not regular.
Problem

- Prove that $L = \{w \mid w = a^mb^n, m \neq n\}$ is not regular.

Solution (without using pumping lemma)

- Suppose $L$ is regular.
- We know that $L' = \{w \mid w = a^ib^j, i, j \geq 0\}$ is regular.
- Let $L'' = \{w \mid w = a^nb^n, n \geq 0\}$.
- As regular languages are closed under intersection and complementation, $L = L' - L'' = L' \cap \bar{L''}$ is regular.
  This implies that $L''$ is regular.
- But, the language $L''$ was earlier proved to be non-regular.
- Contradiction! Hence, $L$ is not regular.