Deterministic Finite Automata
The machine **accepts** a string if the process ends in a double circle.
The alphabet of a finite automaton is the set where the symbols come from: \{0, 1\}

The language of a finite automaton is the set of strings that it accepts.
Notation

An alphabet $\Sigma$ is a finite set (e.g., $\Sigma = \{0,1\}$)

A string over $\Sigma$ is a finite-length sequence of elements of $\Sigma$

For a string $x$, $|x|$ is the length of $x$

The unique string of length 0 will be denoted by $\varepsilon$ and will be called the empty or null string

A language over $\Sigma$ is a set of strings over $\Sigma$
A finite automaton is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$

- $Q$ is the set of states
- $\Sigma$ is the alphabet
- $\delta : Q \times \Sigma \rightarrow Q$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states

$L(M) =$ the language of machine $M$

$= \text{set of all strings machine } M \text{ accepts}$
A language is regular if it is recognized (accepted) by a deterministic finite automaton

$L = \{ w \mid w \text{ contains 001}\}$ is regular

$L = \{ w \mid w \text{ has an even number of 1s}\}$ is regular
Union Theorem

Given two languages, $L_1$ and $L_2$, define the union of $L_1$ and $L_2$ as

$$L_1 \cup L_2 = \{ w \mid w \in L_1 \text{ or } w \in L_2 \}$$

Theorem: The union of two regular languages is also a regular language
Theorem: The union of two regular languages is also a regular language

Proof Sketch: Let

\[ M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1) \] be finite automaton for \( L_1 \)

and

\[ M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2) \] be finite automaton for \( L_2 \)

We want to construct a finite automaton

\[ M = (Q, \Sigma, \delta, q_0, F) \] that recognizes \( L = L_1 \cup L_2 \)
Theorem: The union of two regular languages is also a regular language
Theorem: The union of two regular languages is also a regular language

**Corollary:** Any finite language is regular
The Regular Operations

Union: $A \cup B = \{ w \mid w \in A \text{ or } w \in B \}$

Intersection: $A \cap B = \{ w \mid w \in A \text{ and } w \in B \}$

Reverse: $A^R = \{ w_1 \ldots w_k \mid w_k \ldots w_1 \in A \}$

Negation: $\neg A = \{ w \mid w \notin A \}$

Concatenation: $A \cdot B = \{ vw \mid v \in A \text{ and } w \in B \}$

Star: $A^* = \{ w_1 \ldots w_k \mid k \geq 0 \text{ and each } w_i \in A \}$
Regular Languages Are Closed Under The Regular Operations

We have seen part of the proof for Union. The proof for intersection is very similar. The proof for negation is easy.
The “Grep” Problem

**Input:** Text $T$ of length $t$, string $S$ of length $n$

**Problem:** Does string $S$ appear inside text $T$?

**Naïve method:**

\[ a_1, a_2, a_3, a_4, a_5, \ldots, a_t \]

**Cost:** Roughly $nt$ comparisons
Automata Solution

Build a machine M that accepts any string with S as a consecutive substring

Feed the text to M

Cost: \( t \) comparisons + time to build M

As luck would have it, the Knuth, Morris, Pratt algorithm builds M quickly
Real-life Uses of DFAs

Grep
Coke Machines
Thermostats (fridge)
Elevators
Train Track Switches
Lexical Analyzers for Parsers
Consider the language $L = \{ a^n b^n \mid n > 0 \}$
i.e., a bunch of $a$’s followed by an equal number of $b$’s
No finite automaton accepts this language
Can you prove this?
Pigeonhole principle:
Given $n$ boxes and $m > n$ objects, at least one box must contain more than one object.

Letterbox principle:
If the average number of letters per box is $x$, then some box will have at least $x$ letters (similarly, some box has at most $x$).
Theorem: \( L = \{a^n b^n \mid n > 0 \} \) is not regular

Proof (by contradiction):

Assume that \( L \) is regular

Then there exists a machine \( M \) with \( k \) states that accepts \( L \)

For each \( 0 \leq i \leq k \), let \( S_i \) be the state \( M \) is in after reading \( a^i \)

\( \exists i, j \leq k \) such that \( S_i = S_j \), but \( i \neq j \)

\( M \) will do the same thing on \( a^i b^i \) and \( a^j b^i \)

But a valid \( M \) must reject \( a^j b^i \) and accept \( a^i b^i \)
Deterministic Finite Automata

- Definition
- Testing if they accept a string
- Building automata

Regular Languages

- Definition
- Closed Under Union, Intersection, Negation
- Using Pigeonhole Principle to show language ain’t regular

Here’s What You Need to Know...