Deterministic Finite Automata

finite-automata

Sara Sorahi

Introduction to the Theory of Computation

What Is a Deterministic Finite Automaton?

A **deterministic finite automaton (DFA)** is the simplest formal model of computation we study.

A DFA:

- reads the input from left to right,
- has a finite number of internal states,
- has no external memory,
- makes exactly one transition for each input symbol.

A DFA can only remember *which state it is currently in*.

Formal Definition of a DFA

A DFA is a 5-tuple:

$$M = (Q, \Sigma, \delta, q_0, F)$$

- Q: finite set of states
- Σ: finite input alphabet
- $\delta : Q \times \Sigma \to Q$: transition function
- q_0: start state
- $F \subseteq Q$: accepting states

Determinism means: for every state and symbol, there is exactly one next state.

How a DFA Processes an Input

Given an input string:

1. Start in the start state q_0
2. Read the input symbol by symbol
3. Follow the transition function
4. Accept if the final state is in F

A DFA always halts after reading the entire input.

Example 1: Even Number of 1s

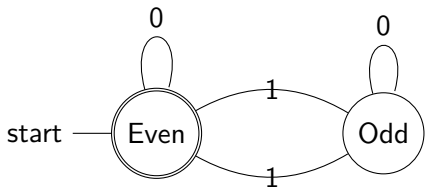
We want a DFA that accepts exactly those strings over $\{0, 1\}$ that contain an **even number of 1s**.

Key observation:

- Only two situations matter: *even* or *odd*
- The exact number does not matter

Graph Explanation: Even Number of 1s

- State **Even**: an even number of 1s has been seen
- State **Odd**: an odd number of 1s has been seen
- Reading 0 does not change parity
- Reading 1 switches parity



Example 2: Strings Ending in 01

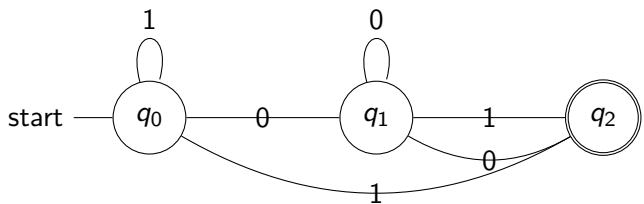
We want a DFA that accepts exactly those strings whose **last two symbols are** 01.

Key idea:

- The DFA only needs to remember a short suffix
- The entire string is irrelevant

Graph Explanation: Ending in 01

- q_0: no relevant suffix
- q_1: last symbol was 0
- q_2: last two symbols were 01



Example 3: Contains Substring ab

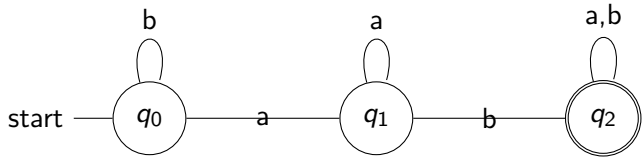
We want a DFA that accepts strings over $\{a, b\}$ that **contain the substring** ab.

Key insight:

- Once ab is seen, acceptance is permanent

Graph Explanation: Contains ab

- q_0: ab has not started
- q_1: a has been seen
- q_2: ab has been seen



Example 4: Binary Numbers Divisible by 3

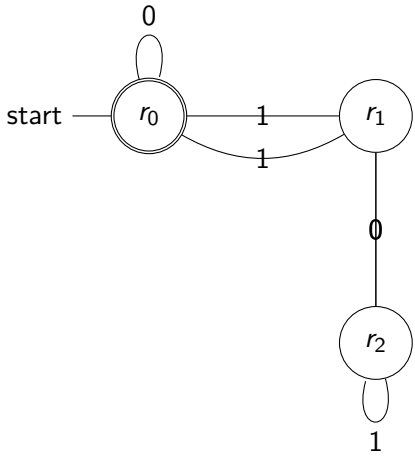
We want a DFA that accepts binary numbers whose numeric value is divisible by 3.

Key idea:

- States represent remainders modulo 3

Graph Explanation: Divisible by 3

- r_0: remainder 0 (accepting)
- r_1: remainder 1
- r_2: remainder 2



Example 5: No Two Consecutive 1s

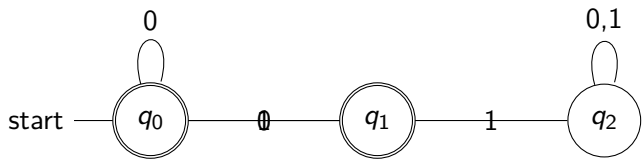
We want a DFA that accepts strings that **do not contain the substring** 11.

Key idea:

- The DFA remembers whether the previous symbol was 1

Graph Explanation: No 11

- q_0: previous symbol was not 1
- q_1: previous symbol was 1
- q_2: forbidden pattern seen (dead state)



Conceptual Takeaway

A DFA can only remember information that fits into a fixed, finite number of states.

This is both its strength and its limitation.