Finite Automata

CS 2800: Discrete Structures, Fall 2014

Sid Chaudhuri
A simplified model

String → Machine → String
A simplified model

String → Machine → String

Finite sequence of letters from an alphabet $\Sigma$, e.g. $\{0, 1\}$
A general-purpose computer

String → Turing Machine → String
A general-purpose computer

String → Turing Machine → String

we'll study these later
A general-purpose computer

Church-Turing Thesis: Any “effective/mechanical/real-world” calculation can be carried out on a Turing machine
Alan Turing, 1912 - 1954
A simple “computer”

String \[\rightarrow\] Deterministic Finite Automaton (DFA) \[\rightarrow\] Yes/No
An example

- Yes
- No

0 -> 1 -> 0
0 -> 1 -> 0
An example

(Binary input)
An example

Input: 01001
An example

Input: 01001
An example

Input: 01001
An example

Input: 01001
An example

Input: 01001
An example

Input: 01001
An example

Input: 01001
Input: 01001
Output: Yes!
An example

In general, on what binary strings does this DFA return Yes?
An example

Ans: All strings with an even number of 1's
Deterministic Finite Automaton

- A DFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$
Deterministic Finite Automaton

- A DFA is a 5-tuple \( M = (Q, \Sigma, \delta, q_0, F) \)
  - \( Q \) is a finite set of states
Deterministic Finite Automaton

- A DFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$
  - $Q$ is a finite set of **states**
  - $\Sigma$ is a finite input **alphabet** (e.g. $\{0, 1\}$)
Deterministic Finite Automaton

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

- $Q$ is a finite set of states
- $\Sigma$ is a finite input alphabet (e.g. $\{0, 1\}$)
- $\delta$ is a transition function $\delta : Q \times \Sigma \rightarrow Q$
Deterministic Finite Automaton

- A DFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$
  - $Q$ is a finite set of states
  - $\Sigma$ is a finite input alphabet (e.g. \{0, 1\})
  - $\delta$ is a transition function $\delta : Q \times \Sigma \to Q$
  - $q_0 \in Q$ is the start/initial state
Deterministic Finite Automaton

- A DFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$
  - $Q$ is a finite set of states
  - $\Sigma$ is a finite input alphabet (e.g. \{0, 1\})
  - $\delta$ is a transition function $\delta : Q \times \Sigma \rightarrow Q$
  - $q_0 \in Q$ is the start/initial state
  - $F \subseteq Q$ is the set of final/accepting states

![DFA Diagram](image)
What does this DFA accept?

$q_0$

(any symbol)
What does this DFA accept?

Answer: No strings
What does this DFA accept?

$q_0$

(any symbol)
What does this DFA accept?

Answer: All strings
What does this DFA accept?
What does this DFA accept?

Answer: Strings of length 1
What does this DFA accept?

Answer: Strings of length 1
What does this DFA accept?
What does this DFA accept?

Answer: Strings containing only 1's
What does this DFA accept?
What does this DFA accept?

Answer: Strings containing no two consecutive 1's.
Language

• Given alphabet $\Sigma$, a language $L$ is a set of strings over the alphabet, i.e. $L \subseteq \Sigma^*$
Language

• Given alphabet \( \Sigma \), a language \( L \) is a set of strings over the alphabet, i.e. \( L \subseteq \Sigma^* \)

• We say a language \( L \) is accepted/recognized by a DFA \( M \), if \( M \) accepts input string \( x \in \Sigma^* \) if and only if \( x \in L \)
Language

• Given alphabet $\Sigma$, a language $L$ is a set of strings over the alphabet, i.e. $L \subseteq \Sigma^*$

Set of all possible strings over $\Sigma$
Language

• Given alphabet $\Sigma$, a language $L$ is a set of strings over the alphabet, i.e. $L \subseteq \Sigma^*$.

• We say a language $L$ is accepted/recognized by a DFA $M$, if $M$ accepts input string $x \in \Sigma^*$ if and only if $x \in L$.
What language does this DFA accept?
What language does this DFA accept?

Answer: Only the string 1
What language does this DFA accept?

$q_0$ → $q_1$ (1) → $q_2$ (1) → $q_3$ (0, 1) → $q_3$ (0, 1)

$0$ → $q_1$ (1) → $q_2$ (1) → $q_3$ (0, 1) → $q_3$ (0, 1)

$0$ → $q_3$ (0, 1)
What language does this DFA accept?

Answer: Only the string 11
DFA's find it difficult to count

- A DFA that recognizes the language \( \{1^c\} \) (the single string of \( c \) 1's) must have at least \( c \) states
DFA's find it difficult to count

- A DFA that recognizes the language \( \{1^c\} \) (the single string of \( c \) 1's) must have at least \( c \) states
  - The parent alphabet is irrelevant (but must of course contain 1)

(Proof discussion to be completed next class)