The Basics of Computation

Shamini Biju, ZZ Zhang, and Sherri Wu

MIT Primes Circle

May 18, 2025

Overview

1. Introduction
2. Languages
3. Finite Automata
4. Deterministic Finite Automata
5. Nondeterministic Finite Automata
6. Different, But Equivalent: DFA vs NFA
7. Real World Applications

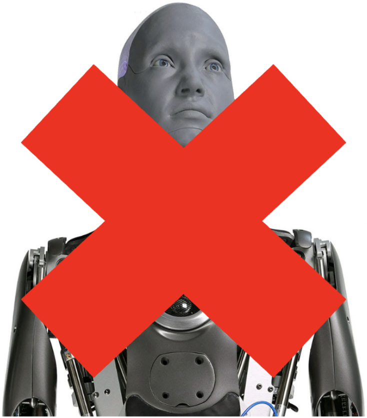
Overview

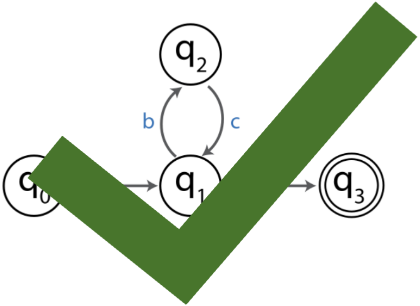
1. Introduction
2. Languages
3. Finite Automata
4. Deterministic Finite Automata
5. Nondeterministic Finite Automata
6. Different, But Equivalent: DFA vs NFA
7. Real World Applications

Introduction

- Computation theory uses different **machines** to explore what computers can and cannot do by recognizing different classes of languages

Machines





Introduction

Definition of a Machine

A machine is a mathematical model that processes inputs based on a set of rules to produce outputs. It follows specific instructions, step-by-step, to determine whether a given input is accepted or rejected.

- Computation theory uses different **machines** to explore what computers can and cannot do by recognizing different classes of languages
- Finite Automata are the simplest machines, recognizing basic patterns in strings

Introduction

Definition of a Machine

A machine is a mathematical model that processes inputs based on a set of rules to produce outputs. It follows specific instructions, step-by-step, to determine whether a given input is accepted or rejected.

- Computation theory uses different **machines** to explore what computers can and cannot do by recognizing different classes of languages
- Finite Automata are the simplest machines, recognizing basic patterns in strings
- More advanced machines like CFGs, PDAs, and Turing Machines handle complex structures and algorithms

Overview

1. Introduction
2. Languages
3. Finite Automata
4. Deterministic Finite Automata
5. Nondeterministic Finite Automata
6. Different, But Equivalent: DFA vs NFA
7. Real World Applications

What Is a Language?

What Is a Language?

Regular Language Example:
String over the alphabet (0,1) that end in 01

What types of strings might be in this language?

What Is a Language?

Definition of a Language

A language is a set of strings over a specified alphabet, where an alphabet is a nonempty finite set of symbols.

Regular Language Example:
String over the alphabet (0,1) that end in 01

What types of strings might be in this language?

What Is a Language?

Regular Language Example:
String over the alphabet (0,1) that end in 01

What types of strings might be in this language?

Definition of a Language

A language is a set of strings over a specified alphabet, where an alphabet is a nonempty finite set of symbols.

- Think of a language like a recipe book.

What Is a Language?

Regular Language Example:
String over the alphabet (0,1) that end in 01

What types of strings might be in this language?

Definition of a Language

A language is a set of strings over a specified alphabet, where an alphabet is a nonempty finite set of symbols.

- Think of a language like a recipe book.
  - Each recipe is a string of instructions made up of a specific set of ingredients (symbols).
  - A language defines which strings are "valid" inputs for a computational problem.
  - It only "accepts" recipes (strings) that follow the defined format.

Language Venn Diagram

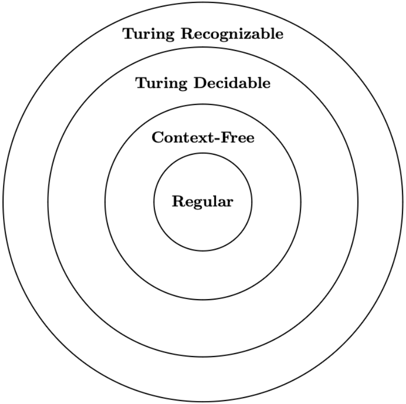


Figure: We will be talking about regular languages today.

Overview

1. Introduction
2. Languages
3. Finite Automata
4. Deterministic Finite Automata
5. Nondeterministic Finite Automata
6. Different, But Equivalent: DFA vs NFA
7. Real World Applications

Introduction to Finite Automata

Introduction to Finite Automata

- Recognizing a language: a machine takes an input and determines whether the string belongs to the language

Introduction to Finite Automata

- Recognizing a language: a machine takes an input and determines whether the string belongs to the language
- Finite Automata are the simplest model out of more advanced machines used to solve computability

Introduction to Finite Automata

- Recognizing a language: a machine takes an input and determines whether the string belongs to the language
- Finite Automata are the simplest model out of more advanced machines used to solve computability
- It doesn't have a memory, therefore, only having the ability to **recognize** the basic class of languages: regular languages

Introduction to Finite Automata

- Recognizing a language: a machine takes an input and determines whether the string belongs to the language
- Finite Automata are the simplest model out of more advanced machines used to solve computability
- It doesn't have a memory, therefore, only having the ability to **recognize** the basic class of languages: regular languages
- There are two types of finite automata: Deterministic and Nondeterministic

Overview

1. Introduction
2. Languages
3. Finite Automata
4. Deterministic Finite Automata
5. Nondeterministic Finite Automata
6. Different, But Equivalent: DFA vs NFA
7. Real World Applications

What is a DFA?

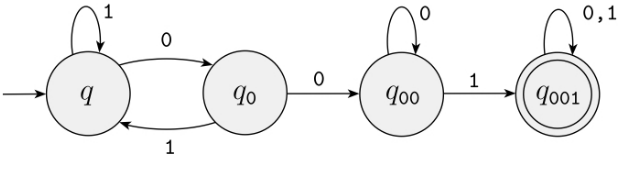


Figure: DFA Example

What is a DFA?

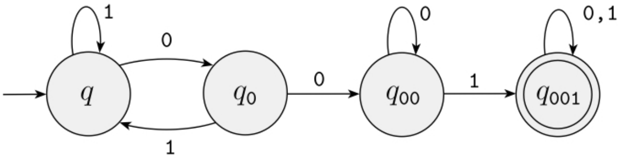


Figure: DFA Example

- States: Possible positions of the machine (e.g., q, q_0, q_{00}, q_{001}).
- Transitions: Arrows indicating state changes based on input (e.g., $q \rightarrow q_0$ on input 0).
- Start State: Initial state, marked by an incoming arrow (e.g., q).
- Accepting States: States where input is accepted, denoted by double circles (e.g., q_{001}).

DFA Definition

DFA Definition

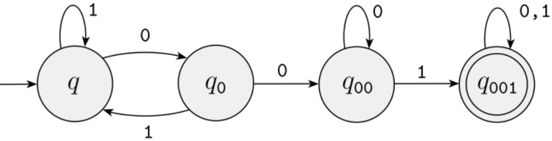


Figure: DFA Example

DFA Definition

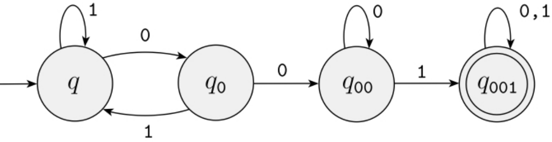


Figure: DFA Example

Definition of a DFA

A finite automata is a 5-tuple, which means it is defined by five components:

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

DFA Definition

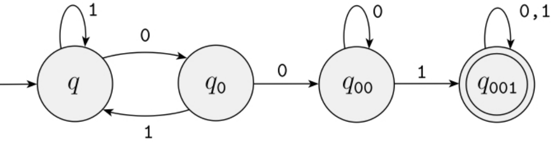


Figure: DFA Example

Definition of a DFA

A finite automata is a 5-tuple, which means it is defined by five components:

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

1. Q is a finite set of states,
2. Σ is the input alphabet (the symbols the DFA can read),
3. $\delta : Q \times \Sigma \to (Q)$ is the transition function, which tells the DFA how to move from one state to another,
4. $q \in Q$ is the start state, where the DFA begins, and
5. $F \subseteq Q$ is the set of accepting states.

Overview

1. Introduction
2. Languages
3. Finite Automata
4. Deterministic Finite Automata
5. Nondeterministic Finite Automata
6. Different, But Equivalent: DFA vs NFA
7. Real World Applications

What is a NFA?

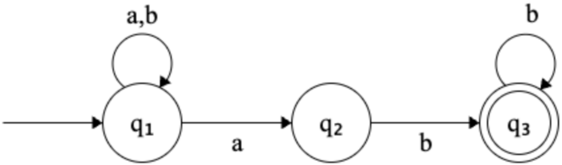


Figure: NFA Example

What is a NFA?

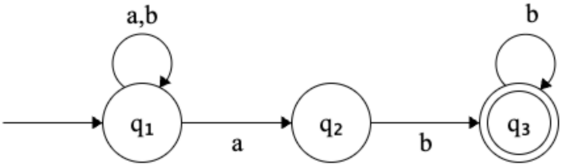


Figure: NFA Example

- States: Possible positions of the machine
- Transitions: Arrows indicating state changes based on input
- Start State: Initial state, marked by an incoming arrow
- Accepting States: States where input is accepted, denoted by double circles

NFA Definition

NFA Definition

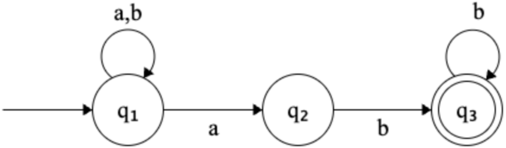


Figure: NFA Example

NFA Definition

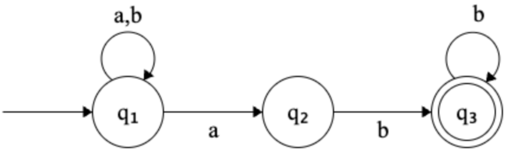


Figure: NFA Example

Definition of a NFA

The definition of a nondeterministic finite automata is a 5-tuple, which means it is defined by five components:

$$M = (Q, \Sigma, \delta, q_1, F)$$

NFA Definition

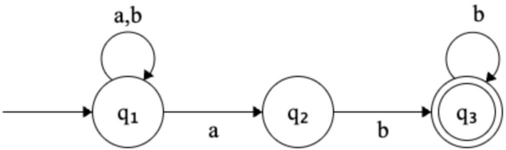


Figure: NFA Example

Definition of a NFA

The definition of a nondeterministic finite automata is a 5-tuple, which means it is defined by five components:

$$M = (Q, \Sigma, \delta, q_1, F)$$

1. Q is a finite set of states,
2. Σ is the input alphabet (the symbols the NFA can read),
3. $\delta : Q \times \Sigma\epsilon \rightarrow P(Q)$ is the transition function
4. $q_1 \in Q$ is the start state, where the NFA begins, and
5. $F \subseteq Q$ is the set of accepting states.

Overview

1. Introduction
2. Languages
3. Finite Automata
4. Deterministic Finite Automata
5. Nondeterministic Finite Automata
6. Different, But Equivalent: DFA vs NFA
7. Real World Applications

Different, But Equivalent: DFA vs NFA

Definition of Equivalence

Equivalence means that two different types of machines (like a DFA and an NFA) can recognize exactly the same set of strings or languages.

Different, But Equivalent: DFA vs NFA

Definition of Equivalence

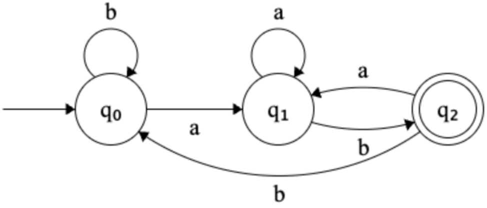
Equivalence means that two different types of machines (like a DFA and an NFA) can recognize exactly the same set of strings or languages.

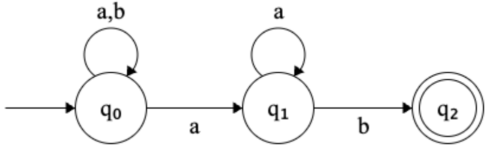
States: Both consist of states that process input symbols, with a start and one or more accepting states.

Pathways: DFA : Exactly one path per state-symbol pair (like a single-lane road).
NFA : Multiple or zero paths per state-symbol pair (like a choose-your-own-adventure book).

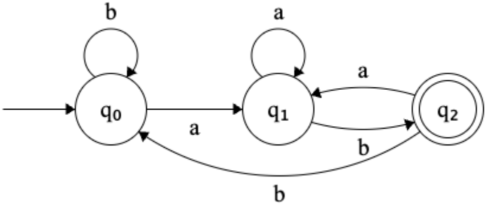
Acceptance: DFA: Accepts if a single path leads to an accepting state.
NFA: Accepts if *any* path leads to an accepting state.

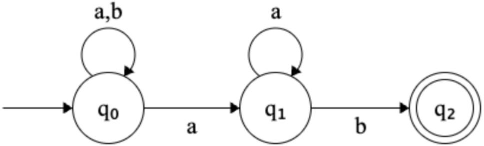
Example of an Equivalent DFA and NFA





Example of an Equivalent DFA and NFA





The set of all strings over the alphabet a,b that contain the substring "ab"

Overview

1. Introduction
2. Languages
3. Finite Automata
4. Deterministic Finite Automata
5. Nondeterministic Finite Automata
6. Different, But Equivalent: DFA vs NFA
7. Real World Applications

Real World Applications

Real World Applications

- All these may seem very abstract, but they're the fundamentals to understanding larger issues in computer science

Real World Applications

- All these may seem very abstract, but they're the fundamentals to understanding larger issues in computer science
- Finite automata and other machines: programming languages and parsing

Real World Applications

- All these may seem very abstract, but they're the fundamentals to understanding larger issues in computer science
- Finite automata and other machines: programming languages and parsing
- Finally, the theory of computation allows us to determine whether a problem is solvable or unsolvable

A Brief Introduction to TOC

This is Sherri's section on an introduction to TOC: Watch Video

References

Michael Sisper (2013)
Understanding the Theory of Computation 3rd ed.

Thank you to...

Katherine Taylor for guidance
Our wonderful research group
Everyone listening!