Lecture 3 – Deterministic Finite Automata (DFA)

COSE215: Theory of Computation

Jihyeok Park

PLRG

2023 Spring

Recall

1. Mathematical Preliminaries
   - Mathematical Notations
   - Inductive Proofs
   - Notations in Languages
2. Basic Introduction of Scala
   - Basic Features
   - Object-Oriented Programming (OOP)
   - Functional Programming (FP)
   - Immutable Collections (Data Structures)

Contents

Definition of DFA

Definition (Deterministic Finite Automata (DFA))

A **deterministic finite automaton** (DFA) is a 5-tuple:

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

- Q is a finite set of **states**
- Σ is a finite set of **symbols**
- $\delta : Q \times \Sigma \to Q$ is the **transition function**
- $q_0 \in Q$ is the **initial state**
- $F \subseteq Q$ is the set of **final states**

$$D = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$$

$$\delta(q_0, 0) = q_1 \qquad \delta(q_1, 0) = q_2 \qquad \delta(q_2, 0) = q_2$$

$$\delta(q_0, 1) = q_0 \qquad \delta(q_1, 1) = q_0 \qquad \delta(q_2, 1) = q_0$$

Definition of DFA

```scala
// The type definitions of states and symbols
type State = Int
type Symbol = Char
// The definition of DFA
case class DFA(
  states: Set[State],
  symbols: Set[Symbol],
  trans: Map[(State, Symbol), State],
  initState: State,
  finalStates: Set[State],
)
// An example of DFA
val dfa: DFA = DFA(
  states = Set(0, 1, 2),
  symbols = Set('0', '1'),
  trans = Map(
    (0, '0') -> 1, (1, '0') -> 2, (2, '0') -> 2,
    (0, '1') -> 0, (1, '1') -> 0, (2, '1') -> 0,
  ),
  initState  = 0,
  finalStates = Set(2),
)
```

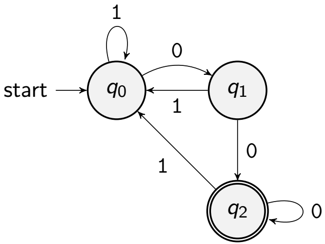
Transition Diagram and Transition Table

$$D = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$$

$$\delta(q_0, 0) = q_1 \qquad \delta(q_1, 0) = q_2 \qquad \delta(q_2, 0) = q_2$$

$$\delta(q_0, 1) = q_0 \qquad \delta(q_1, 1) = q_0 \qquad \delta(q_2, 1) = q_0$$

Transition Diagram



Transition Table

q	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
$*q_2$	q_2	q_0

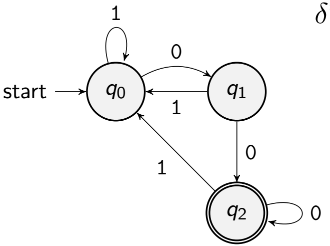
Extended Transition Function

Definition (Extended Transition Function)

For a given DFA $D = (Q, \Sigma, \delta, q_0, F)$, the **extended transition function** $\delta^* : Q \times \Sigma^* \to Q$ is defined as follows:

- **(Basis Case)** $\delta^*(q, \epsilon) = q$
- **(Induction Case)** $\delta^*(q, aw) = \delta^*(\delta(q, a), w)$

$D =$



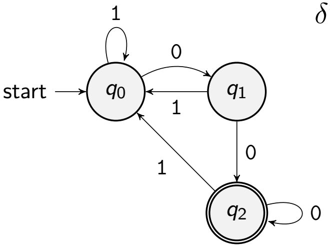
$\delta^*(q_0, 100)$

Extended Transition Function

Definition (Extended Transition Function)

For a given DFA $D = (Q, \Sigma, \delta, q_0, F)$, the **extended transition function** $\delta^* : Q \times \Sigma^* \to Q$ is defined as follows:

- **(Basis Case)** $\delta^*(q, \epsilon) = q$
- **(Induction Case)** $\delta^*(q, aw) = \delta^*(\delta(q, a), w)$

$D =$ 

$$
\begin{aligned}
\delta^*(q_0, 100) &= \delta^*(\delta(q_0, 1), 00) = \delta^*(q_0, 00) \\
&= \delta^*(\delta(q_0, 0), 0) = \delta^*(q_1, 0) \\
&= \delta^*(\delta(q_1, 0), \epsilon) = \delta^*(q_2, \epsilon) \\
&= q_2
\end{aligned}
$$

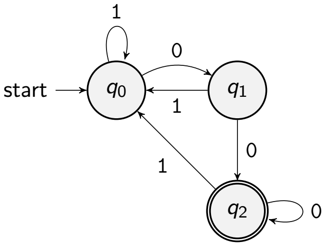
Extended Transition Function

```scala
// The type definition of words
type Word = String

// A helper function to extract first symbol and rest of word
object `<|` { def unapply(w: Word) = w.headOption.map((_, w.drop(1))) }

// The extended transition function of DFA
def extTrans(dfa: DFA)(q: State, w: Word): State = w match
  case "" => q
  case a <| x => extTrans(dfa)(dfa.trans(q, a), x)

// An example transition for a word "100"
extTrans(dfa)(0, "100") // 2
```

Definition (Acceptance of a Word)

For a given DFA $D = (Q, \Sigma, \delta, q_0, F)$, we say that D **accepts** a word $w \in \Sigma^*$ if and only if $\delta^*(q_0, w) \in F$

$D =$



$$\delta^*(q_0, 100) = q_2 \in F$$

It means that D accepts 100.

Acceptance of a Word

```
// The acceptance of a word by DFA
def accept(dfa: DFA)(w: Word): Boolean =
  val curSt: State = extTrans(dfa)(dfa.initState, w)
  dfa.finalStates.contains(curSt)

// An example acceptance of a word "100"
accept(dfa)("100") // true
```

Language of DFA (Regular Language)

Definition (Language of DFA)

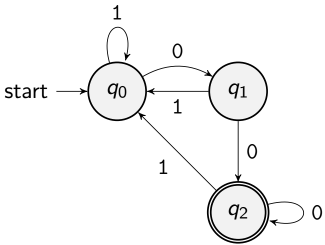
For a given DFA $D = (Q, \Sigma, \delta, q_0, F)$, the **language** of D is defined as follows:

$$L(D) = \{w \in \Sigma^* \mid D \text{ accepts } w\}$$

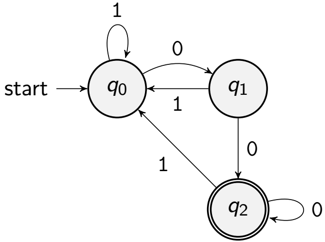
Definition (Regular Language)

A language L is **regular** if and only if there exists a DFA D such that $L(D) = L$

Example 1

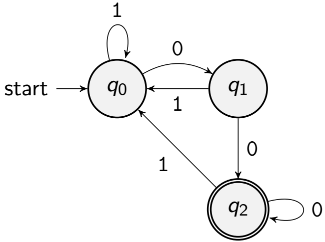
$D =$ 

Example 1

$D =$ 

$$\delta^*(q_0, 100) = q_2 \in F \quad \Rightarrow \quad D \text{ accepts } 100 \quad \Rightarrow \quad 100 \in L(D)$$

Example 1

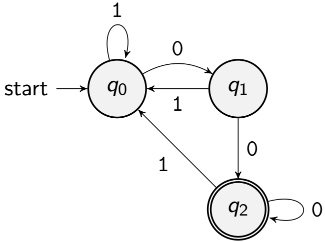
$D =$ 

$$\delta^*(q_0, 100) = q_2 \in F \quad \Rightarrow \quad D \text{ accepts } 100 \quad \Rightarrow \quad 100 \in L(D)$$

$$\epsilon, 0, 1, 01, 10, 11, 001, 010, 011, 101, \cdots \notin L(D)$$

$$00, 000, 100, 0000, 0100, 1000, 1100, \cdots \in L(D)$$

Example 1

$D =$ 

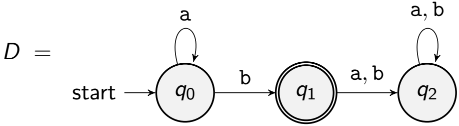
$$\delta^*(q_0, 100) = q_2 \in F \quad \Rightarrow \quad D \text{ accepts } 100 \quad \Rightarrow \quad 100 \in L(D)$$

$$\epsilon, 0, 1, 01, 10, 11, 001, 010, 011, 101, \cdots \notin L(D)$$

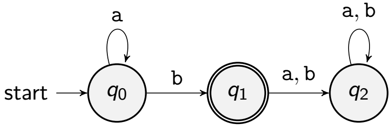
$$00, 000, 100, 0000, 0100, 1000, 1100, \cdots \in L(D)$$

$$L(D) = \{w00 \mid w \in \{0,1\}^*\}$$

Example 2

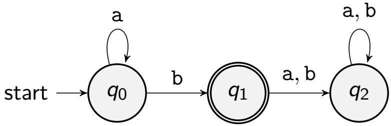


Example 2

$D =$ 

$$\delta^*(q_0, \texttt{aab}) = q_1 \in F \quad \Rightarrow \quad D \text{ accepts } \texttt{aab} \quad \Rightarrow \quad \texttt{aab} \in L(D)$$

Example 2

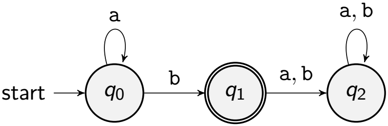
$D =$ 

$$\delta^*(q_0, \text{aab}) = q_1 \in F \quad \Rightarrow \quad D \text{ accepts aab} \quad \Rightarrow \quad \text{aab} \in L(D)$$

$$\epsilon, \text{a}, \text{aa}, \text{ba}, \text{bb}, \text{aaa}, \text{aba}, \text{abb}, \text{baa}, \text{bab}, \text{bba}, \cdots \notin L(D)$$

$$\text{b}, \text{ab}, \text{aab}, \text{aaab}, \text{aaaab}, \text{aaaaab}, \text{aaaaaab}, \cdots \in L(D)$$

Example 2

$D =$ 

$$\delta^*(q_0, \text{aab}) = q_1 \in F \quad \Rightarrow \quad D \text{ accepts aab} \quad \Rightarrow \quad \text{aab} \in L(D)$$

$$\epsilon, \text{a}, \text{aa}, \text{ba}, \text{bb}, \text{aaa}, \text{aba}, \text{abb}, \text{baa}, \text{bab}, \text{bba}, \cdots \notin L(D)$$

$$\text{b}, \text{ab}, \text{aab}, \text{aaab}, \text{aaaab}, \text{aaaaab}, \text{aaaaaab}, \cdots \in L(D)$$

$$L(D) = \{\text{a}^n\text{b} \mid n \geq 0\}$$

Example 3

Theorem

The language $L = \{w \in \{0, 1\}^* \mid w$ *is a binary format (allowing leading zeros) of natural numbers divisible by* $3\}$ *is regular.*

Proof)

Example 3

Theorem

The language $L = \{w \in \{0, 1\}^* \mid w$ *is a binary format (allowing leading zeros) of natural numbers divisible by* $3\}$ *is regular.*

Proof) You need to construct a DFA D such that $L(D) = L$.

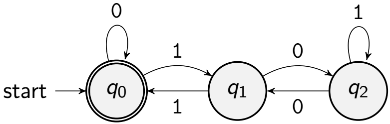
Example 3

Theorem

The language $L = \{w \in \{0, 1\}^* \mid w$ *is a binary format (allowing leading zeros) of natural numbers divisible by* $3\}$ *is regular.*

Proof) You need to construct a DFA D such that $L(D) = L$. Consider the following DFA D:

$D =$



□

Example 4

Theorem

The language $L = \{a^n b^n \mid n \geq 0\}$ is regular.

You need to construct a DFA D such that $L(D) = L$.

Example 4

Theorem

The language $L = \{a^n b^n \mid n \geq 0\}$ is regular.

You need to construct a DFA D such that $L(D) = L$. However, it is impossible because L is actually not regular.

Example 4

Theorem

The language $L = \{a^n b^n \mid n \geq 0\}$ is regular.

You need to construct a DFA D such that $L(D) = L$. However, it is impossible because L is actually not regular.

Then, is it possible to prove that L is not regular?

Example 4

Theorem

The language $L = \{a^n b^n \mid n \geq 0\}$ *is regular.*

You need to construct a DFA D such that $L(D) = L$. However, it is impossible because L is actually not regular.

Then, is it possible to prove that L is not regular?

Yes, it is possible BUT you will learn how to prove it (using Pumping Lemma) later in this course.

Summary

1. Deterministic Finite Automata (DFA)
   - Definition
   - Transition Diagram and Transition Table
   - Extended Transition Function
   - Acceptance of a Word
   - Language of DFA (Regular Language)
   - Examples

Next Lecture

PLRG

- Nondeterministic Finite Automata (NFA)

Jihyeok Park
jihyeok_park@korea.ac.kr
https://plrg.korea.ac.kr