


01204211 Discrete Mathematics

Lecture 9a: Finite automata¹

Jittat Fakcharoenphol

August 26, 2025

¹Based on lecture notes of *Models of Computation* course by Jeff Erickson. 

Example: syntax highlighting

HTML tokenizer

Game programming

State-transition graphs

More examples over $\Sigma = \{0, 1\}$

All strings, except 010.

Strings containing the subsequence 010.

Formal definitions

A **finite-state machine** or a **deterministic finite-state automaton** (DFA) has five components:

Formal definitions

A **finite-state machine** or a **deterministic finite-state automaton** (DFA) has five components:

- ▶ the input alphabet Σ ,

Formal definitions

A **finite-state machine** or a **deterministic finite-state automaton** (DFA) has five components:

- ▶ the input alphabet Σ ,
- ▶ a finite set of states Q ,

Formal definitions

A **finite-state machine** or a **deterministic finite-state automaton** (DFA) has five components:

- ▶ the input alphabet Σ ,
- ▶ a finite set of states Q ,
- ▶ a transition function δ

Formal definitions

A **finite-state machine** or a **deterministic finite-state automaton** (DFA) has five components:

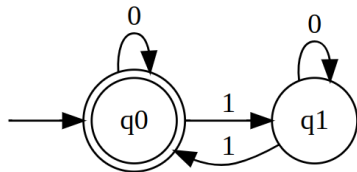
- ▶ the input alphabet Σ ,
- ▶ a finite set of states Q ,
- ▶ a transition function $\delta : Q \times \Sigma \longrightarrow Q$

Formal definitions

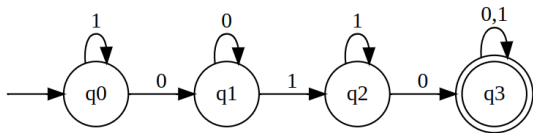
A **finite-state machine** or a **deterministic finite-state automaton** (DFA) has five components:

- ▶ the input alphabet Σ ,
- ▶ a finite set of states Q ,
- ▶ a transition function $\delta : Q \times \Sigma \longrightarrow Q$
- ▶ a start state $s \in Q$, and
- ▶ a subset $A \subseteq Q$ of accepting states.

Example 1



Example 2



Moves

One step move: from state q with input symbol a , the machine changes its state to

Moves

One step move: from state q with input symbol a , the machine changes its state to $\delta(q, a)$.

Extension: from state q with input string w , the machine changes its state to $\delta^*(q, w)$ defined as

Moves

One step move: from state q with input symbol a , the machine changes its state to $\delta(q, a)$.

Extension: from state q with input string w , the machine changes its state to $\delta^*(q, w)$ defined as

$$\delta^*(q, w) = \begin{cases} q & \text{if } w = \varepsilon, \\ \delta^*(\delta(q, a), x) & \text{if } w = ax. \end{cases}$$

The signature of δ^* is $Q \times \Sigma^* \longrightarrow Q$.

Acceptance

For a finite-state machine with starting state s and accepting states A , it accepts string w iff

Acceptance

For a finite-state machine with starting state s and accepting states A , it accepts string w iff

$$\delta^*(s, w) \in A.$$

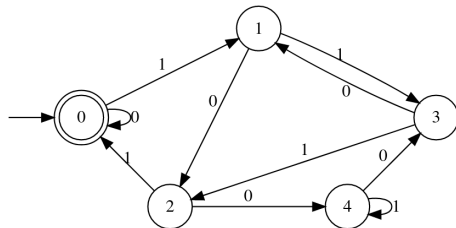
Multiple of 5

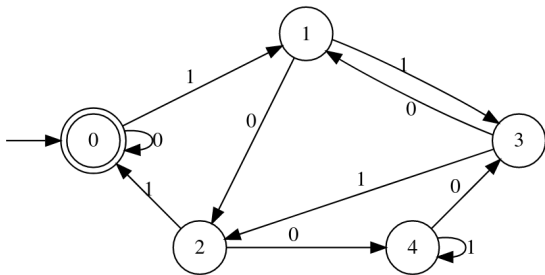
Multiple of 5

```
def multiple_of_5(w):  
    r = 0  
    for i in w:  
        r = (2*r + w) % 5  
    return r == 0
```

Multiple of 5

```
def multiple_of_5(w):  
    r = 0  
    for i in w:  
        r = (2*r + w) % 5  
    return r == 0
```





Digital design: Implementation

Digital design: Moore and Mealy machines

In the digital design class, you will encounter finite-state machines as well. The version we consider in this class is referred to as a **Moore machine**.

In practice, there is another variant of FSM called **Mealy machines**, whose outputs depend on input symbols as well.

Digital design: Moore and Mealy machines

In the digital design class, you will encounter finite-state machines as well. The version we consider in this class is referred to as a **Moore machine**.

In practice, there is another variant of FSM called **Mealy machines**, whose outputs depend on input symbols as well.

Formally, they differ in output function.

- ▶ Moore machine: $G : Q \longrightarrow [0, 1]$
- ▶ Mealy machine: $G : Q \times \Sigma \longrightarrow [0, 1]$

Example: even number of 1's

Example: strings containing 00 as a substring

Combining DFAs

What if we want to build a DFA that accepts strings with an even number of 1's and containing 00 as a substring?

Product construction

Product construction (formally)

Given a DFA $M_1 = (\Sigma, Q_1, \delta_1, s_1, A_1)$ that accepts strings from language L_1 and $M_2 = (\Sigma, Q_2, \delta_2, s_2, A_2)$ that accepts strings from language L_2 , we can construct a DFA $M = (\Sigma, Q, \delta, s, A)$ that accepts strings from $L_1 \cap L_2$ as follows:

- ▶ Let $Q = Q_1 \times Q_2$,

Product construction (formally)

Given a DFA $M_1 = (\Sigma, Q_1, \delta_1, s_1, A_1)$ that accepts strings from language L_1 and $M_2 = (\Sigma, Q_2, \delta_2, s_2, A_2)$ that accepts strings from language L_2 , we can construct a DFA $M = (\Sigma, Q, \delta, s, A)$ that accepts strings from $L_1 \cap L_2$ as follows:

- ▶ Let $Q = Q_1 \times Q_2$,
- ▶ Let $\delta : (Q_1 \times Q_2) \times \Sigma \longrightarrow (Q_1 \times Q_2)$ be such that

Product construction (formally)

Given a DFA $M_1 = (\Sigma, Q_1, \delta_1, s_1, A_1)$ that accepts strings from language L_1 and $M_2 = (\Sigma, Q_2, \delta_2, s_2, A_2)$ that accepts strings from language L_2 , we can construct a DFA $M = (\Sigma, Q, \delta, s, A)$ that accepts strings from $L_1 \cap L_2$ as follows:

- ▶ Let $Q = Q_1 \times Q_2$,
- ▶ Let $\delta : (Q_1 \times Q_2) \times \Sigma \longrightarrow (Q_1 \times Q_2)$ be such that

$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)),$$

Product construction (formally)

Given a DFA $M_1 = (\Sigma, Q_1, \delta_1, s_1, A_1)$ that accepts strings from language L_1 and $M_2 = (\Sigma, Q_2, \delta_2, s_2, A_2)$ that accepts strings from language L_2 , we can construct a DFA $M = (\Sigma, Q, \delta, s, A)$ that accepts strings from $L_1 \cap L_2$ as follows:

- ▶ Let $Q = Q_1 \times Q_2$,
- ▶ Let $\delta : (Q_1 \times Q_2) \times \Sigma \longrightarrow (Q_1 \times Q_2)$ be such that

$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)),$$

- ▶ Let $s = (s_1, s_2)$, and

Product construction (formally)

Given a DFA $M_1 = (\Sigma, Q_1, \delta_1, s_1, A_1)$ that accepts strings from language L_1 and $M_2 = (\Sigma, Q_2, \delta_2, s_2, A_2)$ that accepts strings from language L_2 , we can construct a DFA $M = (\Sigma, Q, \delta, s, A)$ that accepts strings from $L_1 \cap L_2$ as follows:

- ▶ Let $Q = Q_1 \times Q_2$,
- ▶ Let $\delta : (Q_1 \times Q_2) \times \Sigma \longrightarrow (Q_1 \times Q_2)$ be such that

$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)),$$

- ▶ Let $s = (s_1, s_2)$, and
- ▶ Let $A = A_1 \times A_2$.

Recall the definition of $\delta^*(q, w)$, i.e.,

Recall the definition of $\delta^*(q, w)$, i.e.,

$$\delta^*(q, w) = \begin{cases} q & \text{if } w = \varepsilon, \\ \delta^*(\delta(q, a), x) & \text{if } w = ax \text{ where } a \in \Sigma \end{cases}$$

Recall the definition of $\delta^*(q, w)$, i.e.,

$$\delta^*(q, w) = \begin{cases} q & \text{if } w = \varepsilon, \\ \delta^*(\delta(q, a), x) & \text{if } w = ax \text{ where } a \in \Sigma \end{cases}$$

Lemma 1

$\delta^*((q_1, q_2), w) = (\delta_1^*(q_1, w), \delta_2^*(q_2, w))$ for any string w .

Proof.

We prove by induction. I.H.: Assume that $\delta^*((q_1, q_2), x) = (\delta_1^*(q_1, x), \delta_2^*(q_2, x))$, for any string x such that $|x| < |w|$.



Correctness

From the previous lemma, we have that

$$\delta^*(s, w) = \delta^*((s_1, s_2), w)$$

Correctness

From the previous lemma, we have that

$$\begin{aligned}\delta^*(s, w) &= \delta^*((s_1, s_2), w) \\ &= (\delta_1^*(s_1, w), \delta_2^*(s_2, w))\end{aligned}$$

Correctness

From the previous lemma, we have that

$$\begin{aligned}\delta^*(s, w) &= \delta^*((s_1, s_2), w) \\ &= (\delta_1^*(s_1, w), \delta_2^*(s_2, w))\end{aligned}$$

Thus, for an input w , M would reach the state $(\delta_1^*(s_1, w), \delta_2^*(s_2, w))$; it accepts w when

$$(\delta_1^*(s_1, w), \delta_2^*(s_2, w)) \in A_1 \times A_2.$$

This implies that M accepts w when $\delta_1^*(s_1, w) \in A_1$ and $\delta_2^*(s_2, w) \in A_2$, i.e., M accepts w iff M_1 and M_2 accept w .

Finally, we conclude that M accepts strings from language $L_1 \cap L_2$.

Language of a DFA

$L(M)$

For a DFA M , let $L(M)$ be the set of all strings that M accepts. More formally, for $M = (\Sigma, Q, \delta, s, A)$,

$$L(M) = \{w \in \Sigma^* \mid \delta^*(s, w) \in A\}.$$

We refer to $L(M)$ as the language of M .

Closure

Lemma 2

If L_1 and L_2 are languages of some DFAs M_1 and M_2 , we have that

- ▶ *there is a DFA M that accepts $L_1 \cap L_2$,*

Lemma 2

If L_1 and L_2 are languages of some DFAs M_1 and M_2 , we have that

- ▶ *there is a DFA M that accepts $L_1 \cap L_2$,*
- ▶ *there is a DFA M that accepts $L_1 \cup L_2$,*

Closure

Lemma 2

If L_1 and L_2 are languages of some DFAs M_1 and M_2 , we have that

- ▶ *there is a DFA M that accepts $L_1 \cap L_2$,*
- ▶ *there is a DFA M that accepts $L_1 \cup L_2$,*
- ▶ *there is a DFA M that accepts $L_1 \setminus L_2$,*

Closure

Lemma 2

If L_1 and L_2 are languages of some DFAs M_1 and M_2 , we have that

- ▶ *there is a DFA M that accepts $L_1 \cap L_2$,*
- ▶ *there is a DFA M that accepts $L_1 \cup L_2$,*
- ▶ *there is a DFA M that accepts $L_1 \setminus L_2$,*
- ▶ *there is a DFA M that accepts $\Sigma^* \setminus L_1$,*

Automatic languages²

Definition (for now)

A language L is “**automatic**” if there is a DFA M such that $L(M) = L$.

²Taken directly from Erickson's lecture notes

Automatic languages²

Definition (for now)

A language L is “**automatic**” if there is a DFA M such that $L(M) = L$.

Lemma 3

If L_1 and L_2 are automatic languages over alphabet Σ , then

► $L_1 \cap L_2$,

²Taken directly from Erickson's lecture notes

Automatic languages²

Definition (for now)

A language L is “**automatic**” if there is a DFA M such that $L(M) = L$.

Lemma 3

If L_1 and L_2 are automatic languages over alphabet Σ , then

- ▶ $L_1 \cap L_2$,
- ▶ $L_1 \cup L_2$,

²Taken directly from Erickson's lecture notes

Automatic languages²

Definition (for now)

A language L is “**automatic**” if there is a DFA M such that $L(M) = L$.

Lemma 3

If L_1 and L_2 are automatic languages over alphabet Σ , then

- ▶ $L_1 \cap L_2$,
- ▶ $L_1 \cup L_2$,
- ▶ $L_1 \setminus L_2$, *and*

²Taken directly from Erickson's lecture notes

Automatic languages²

Definition (for now)

A language L is “**automatic**” if there is a DFA M such that $L(M) = L$.

Lemma 3

If L_1 and L_2 are automatic languages over alphabet Σ , then

- ▶ $L_1 \cap L_2$,
- ▶ $L_1 \cup L_2$,
- ▶ $L_1 \setminus L_2$, and
- ▶ $\Sigma^* \setminus L_1$,

are also automatic.

²Taken directly from Erickson's lecture notes

Automatic languages²

Definition (for now)

A language L is “**automatic**” if there is a DFA M such that $L(M) = L$.

Lemma 3

If L_1 and L_2 are automatic languages over alphabet Σ , then

- ▶ $L_1 \cap L_2$,
- ▶ $L_1 \cup L_2$,
- ▶ $L_1 \setminus L_2$, *and*
- ▶ $\Sigma^* \setminus L_1$,

are also automatic.

The set of automatic languages is closed under these boolean operations.

²Taken directly from Erickson's lecture notes