Topic 7
Finite-State Automata

TMA1201 Discrete Structures & Probability
Faculty of Computing & Informatics
Multimedia University

What you will learn in this lecture:

- Formal Language
- Finite-State Automata

What is a Formal Language?

- In English,
  - A word can be regarded as a string of letters, and
  - A sentence can be regarded as a string of words.

- Computer languages are similar to English in that
  - Certain strings of characters are legitimate words of the language, and
  - Certain strings of words can be put together according to certain rules to form syntactically correct programs.

- In computer science, it has proven useful to look at languages from a very abstract point of view as strings of certain fundamental units and allow any finite set of symbols to be used as an alphabet.

- A *formal language over an alphabet* is any set of strings of characters of the alphabet. It is specified by a well-defined set of rules of syntax.

Example 1

Let the alphabet $\Sigma = \{a, b\}$. (Note: We call a, b as symbols of the alphabet)

Find L_1, defined as the language consisting of all strings over Σ that begins with the character a and has length of at most three characters.

$$L_1 = \{a, aa, ab, aaa, aab, aba, abb\}$$

Find L_2, defined as the language consisting of all strings over Σ that ends with the character b and has length of exactly three characters.

$$L_2 = \{aab, abb, bab, bbb\}$$

Example 1

Let the alphabet $\Sigma = \{a, b\}$. (Note: We call a, b as symbols of the alphabet)

Find L_1, defined as the language consisting of all strings over Σ that begins with the character a and has length of at most three characters.

$$L_1 = \{a, aa, ab, aaa, aab, aba, abb\}$$

Find L_2, defined as the language consisting of all strings over Σ that ends with the character b and has length of exactly three characters.

$$L_2 = \{aab, abb, bab, bbb\}$$

What is a Finite-State Automaton?

- A ***finite-state automaton*** is an idealized machine that embodies the essential idea of a sequential circuit, where the output depends not only on the input, but also on the ***state*** of the system when the input is received.

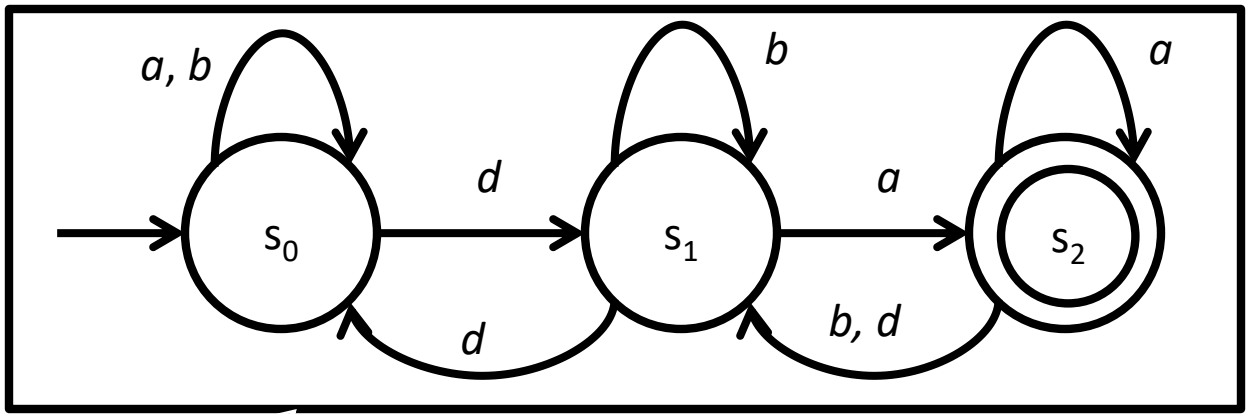
- Each piece of input to a finite-state automaton leads to a change in the state of the automaton, which in turn affects how subsequent input is processed.

- An example is the act of dialing a telephone number. Dialing 1–300 puts the telephone circuit in a state of readiness to receive the final seven digits of a toll-free call, whereas dialing 013 leads to a state of expectation for the other seven digits of a mobile call.

Definition of a Finite-State Automaton

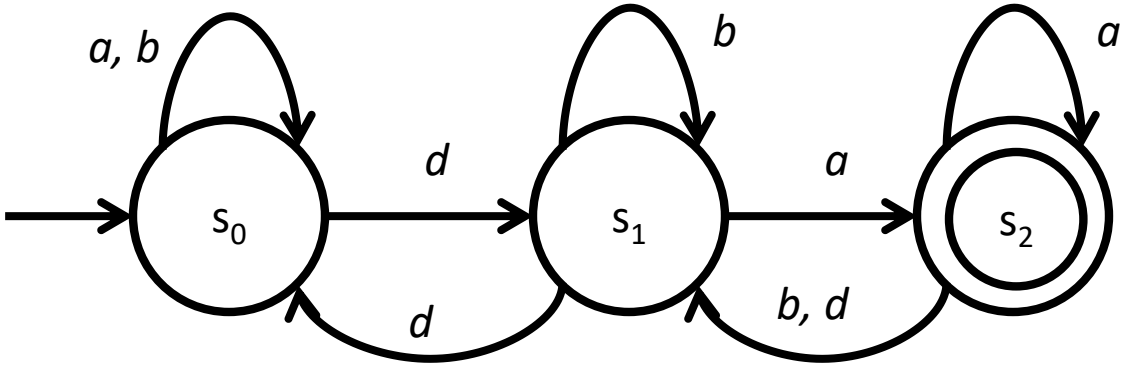
A *finite-state automaton* consists of five objects:

- A finite set of *states*, $S = \{s_0, s_1, \ldots, s_n\}$;
- An initial state, normally denoted by s_0;
- A set of accepting states, $A \subseteq S$.
- A finite set of *input symbols*, I;
- A *next-state function,* $f: S \times I \rightarrow S$ that assigns a next state to every pair of state and input.



This is known as a state-transition diagram.

Example 2



The finite-state automaton above is defined as M = {S, s_0, A, I, F}, where

S = {s_0, s_1, s_2}
A = {s_2 }
I = {*a, b, d*}
F = {$f_x \mid x \in \mathrm{I}$}

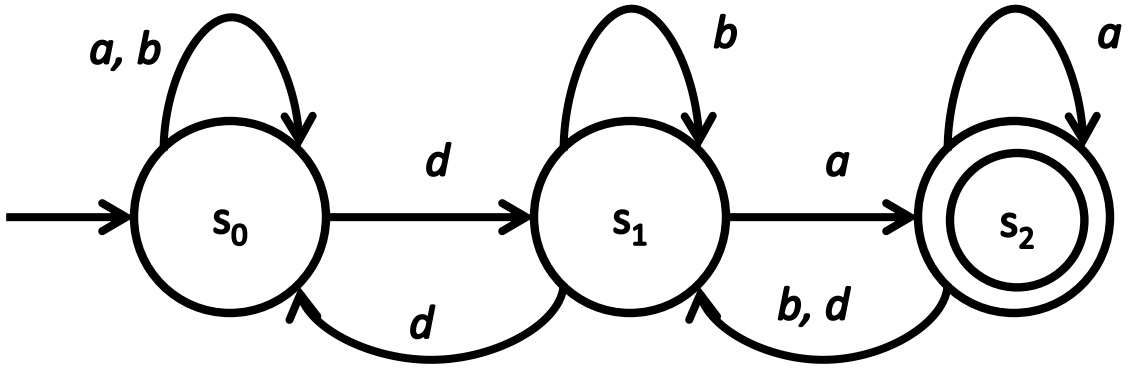
- The initial state is indicated by an incoming arrow;
- The accepting state is marked by a double circle.

Next-State Table

State	Input		
	a	*b*	*d*
s_0	s_0	s_0	s_1
s_1	s_2	s_1	s_0
s_2	s_2	s_1	s_1

A next-state table shows the values of the next-state function *f* for all possible states *s* and input symbols *i*.

Next states

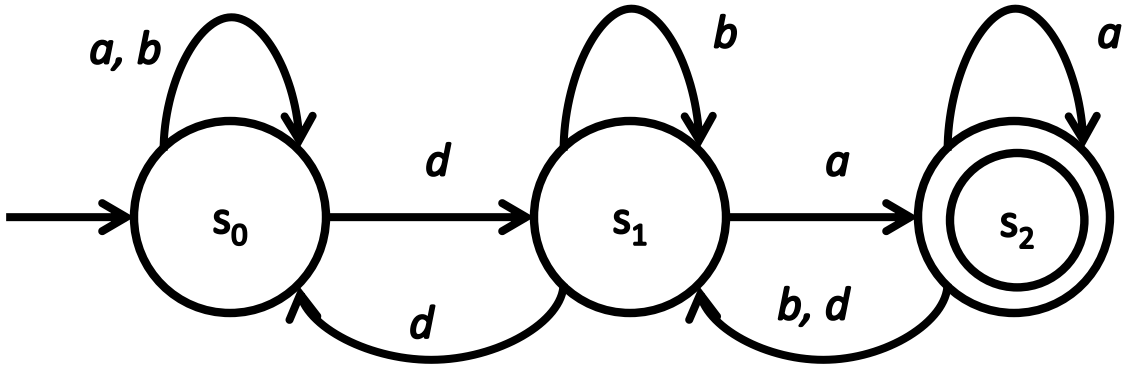


Example 3

State	Input		
	a	*b*	*d*
s_0	s_0	s_0	s_1
s_1	s_2	s_1	s_0
s_2	s_2	s_1	s_1

Which state the automaton will be in when:

1) **It receives input *a* at initial time?**
2) **It receives input *b* at s_1?**
3) **It receives input *add* at s_2?**



Formal Language and Finite-State Automata

- A compiler for a computer language analyzes the stream of characters in a program by:
  1. Firstly, recognizing individual word and sentence units.
  2. Secondly, analyzing the syntax, or grammar, of the sentences.
  3. Finally, translating the sentences into machine code.

- Regular languages, which are defined by regular expressions, are used extensively for matching patterns within text and for lexical analysis in computer language compilers.

- Regular expressions of a formal language make it possible to replace a long, complicated set of if-then-else statements with code that is easy both to produce and to understand.

- According to Kleene's Theorem, **the set of languages defined by regular expressions is identical to the set of languages accepted by finite-state automata.**

Defining Language by a Regular Expression

Regular expression is one of the most useful ways to define a language. The following describes ways in which language can be formed by regular expression:

Let Σ be an alphabet, with *a* and *b* as symbols/strings for regular expressions over Σ, the following are also regular expressions over Σ:

Regular Expressions	**Meaning**
(*ab*)	Concatenation of *a* and *b*
(*a* | *b*)	Either one of *a* or *b*
(*a**)	Concatenation of *a* with itself any finite number (including zero) of times

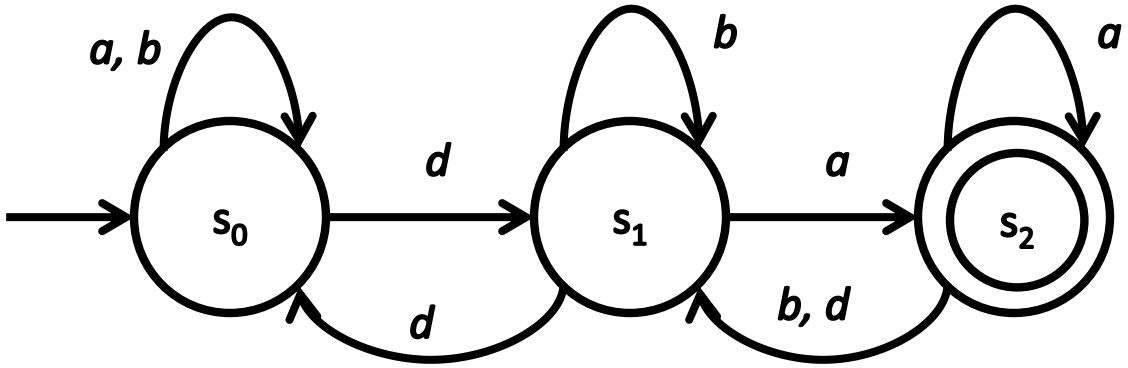
What is a Language Accepted by an Automaton?

- Suppose a string of input symbols is fed into a finite-state automaton in sequence. At the end of the process, after each successive input symbol has changed the state of the automaton, the automaton ends up in a certain state, which may be either an accepting state or a non-accepting state.

- Those strings that send the automaton to an accepting state are said to be *accepted* by the automaton.

- The language accepted by the automaton M, denoted as $L(M)$, is the set of all strings that are accepted by M.

Example 4



Remember that an accepting state is indicated by the double circle.

- This finite-state automaton accepts the strings of inputs that stops at s_2.

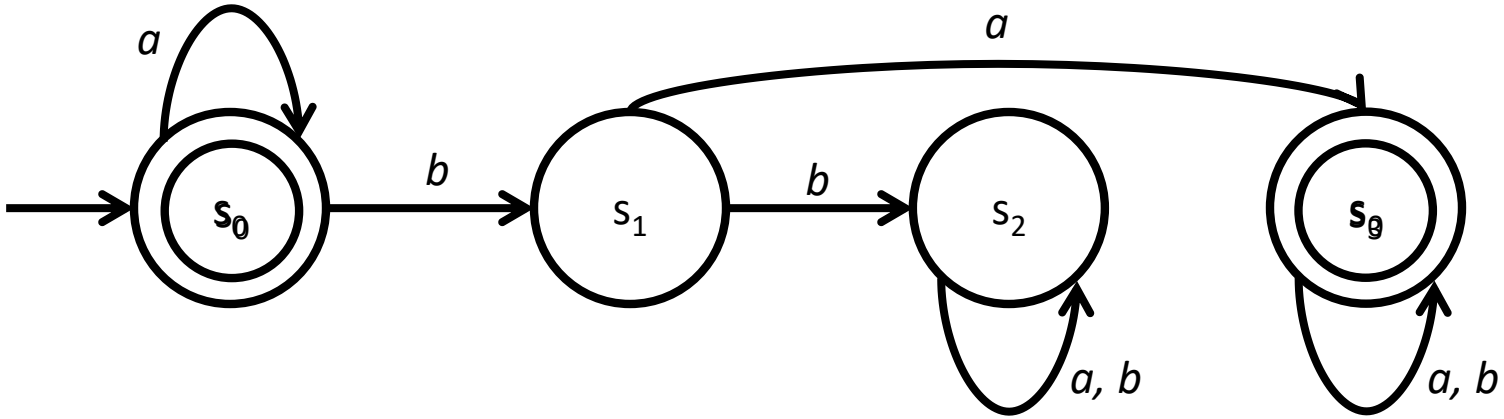
- The automaton accepts the following inputs:

da	*ada*	*bda*	*aaaaaaabbbbbdabbbbbbaaaaa*
dba	*adba*	*bdba*	*bbbbbbbaaaaaaaadaaaaaaaaa*
abda	*bada*	*daaaaaaaaaaaaaaaaaaaaaaaaaaaaa*	

- The automaton does not accept the following inputs:

abbba	*adbb*	*bbabd*	*aababddbab*
abdda	*dbab*	*bdbdd*	*bdbdaaadb*
abdddbaab		*badbaabbabbadb*	

Example 5



Given the finite-state automaton *M* as shown above,

i. Give three strings that are accepted by *M*.
ii. Give three strings that are not accepted by *M*.
iii. What is the language accepted by the *M*?

Solution:

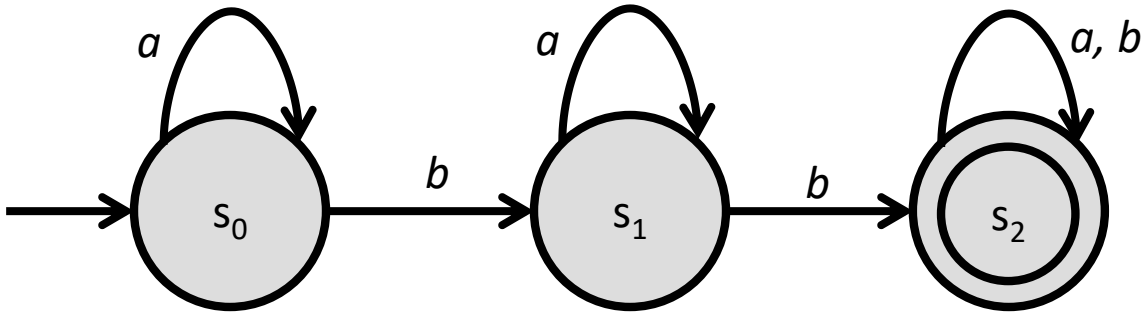
i. *a*, *aaba*, *abab*

ii. *abb*, *bbbb*, *bbaab*

iii. $L(M) = \{a^k, a^n bax$ where $k \in Z^+$, $n \in N$, and x is any string of *a*'s and *b*'s$\}$.

$= \{a^* \mid a^* ba(a|b)^*\}$

Example 6



Given the finite-state automaton *M* as shown above,

i. Give three strings that are accepted by *M*.
ii. Give three strings that are not accepted by *M*.
iii. What is the language accepted by the *M*?

Solution:

i. *bb*, *abab*, *babaab*

ii. *aa*, *abaa*, *aaabaa*

iii. $L(M)$ = The set of strings that contain at least two *b*'s.

$= \{a^n b a^n b x$ where $n \in \mathrm{N}$ and *x* is any string of *a*'s and *b*'s$\}$.

$= \{a^* b a^* b (a|b)^*\}$

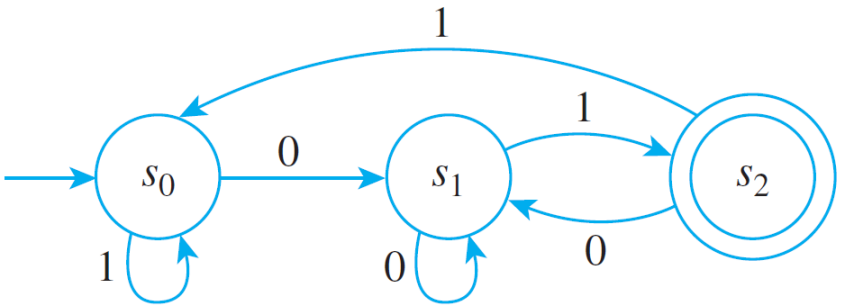
Summary

You have learned the following concepts related to *finite-state automata*:

- The meaning of finite-state automata
- The link between finite-state automata and formal language
- Recognizing the language accepted by a finite-state automaton

Self Test 1

Given the following finite-state automaton, M:



a) Find its
   i. States.
   ii. Input symbols.
   iii. Initial state.
   iv. Accepting state(s).

b) Construct the next-state table for M.

c) To what states does M go if the symbols of the following strings are input to M in sequence, starting from the initial state?
   (i) 01 (ii) 0011 (iii) 0101100 (iv) 10101

d) Which of the strings in part (c) send M to an accepting state?

e) What is the language accepted by M?