San José State University
Department of Computer Science

Ahmad Yazdankhah
ahmad.yazdankhah@sjsu.edu
www.cs.sjsu.edu/~yazdankhah

Deterministic Finite Automata

(Part 2)

Lecture 07

Day 07/31

CS 154

Formal Languages and Computability

Fall 2019

Agenda of Day 07

- Summary of Lecture 06
- Quiz 2
- Lecture 07: Teaching ...
  - Deterministic Finite Automata (Part 2)

Summary of Lecture 07: We learned ...

Automata

- Formal languages are mathematical model of all languages.
- We are going to construct some machines to understand these languages.
- We call these machines automata.
- Automaton is ...
- ... a mathematical model of a computing device.
- We'll construct several classes of machines in this course.

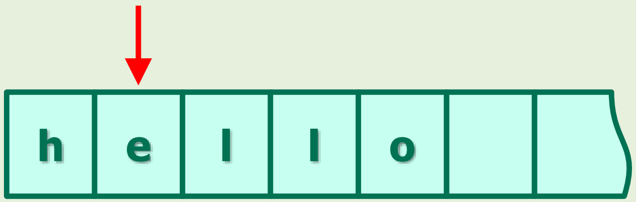
DFAs

- DFAs are the simplest ones.
- DFA stands for ...
- ... Deterministic Finite Automata
- Its building blocks contains ...
- ... Input tape, Control unit, Output

Any question?

Summary of Lecture 07: We learned ...

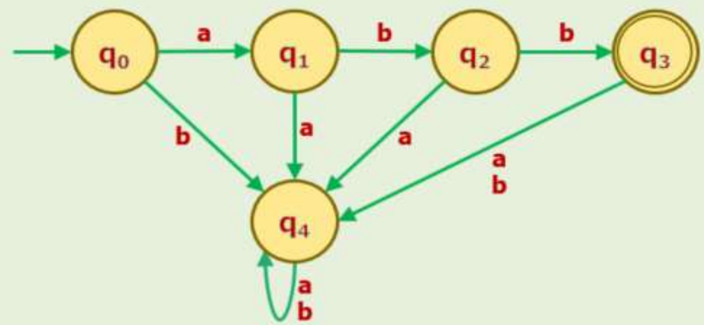
Input Tape



- The input tape is read-only.
- The read-head moves from left-to-right.
  - We cannot move the head back.
- Consuming a symbol = reading the symbol + moving the read-head to the right

Control Unit

- Its decision making part is represented by a transition graph.



- There is only one initial state.
- There can be zero or more accepting state (aka final state).
- The number of states is finite.
- That's why we call this class Deterministic Finite Automata.

Any question?

Summary of Lecture 07: We learned ...

Output

Output

Accept **or** **Reject**

- The output has two messages:
  - Accept (aka: understood, recognized)
  - Reject (aka: not understood, not recognized)

Any question?

Summary of Lecture 07: We learned ...

When do DFAs halt?

- When all input symbols are consumed.

$$h \leftrightarrow c$$

How DFAs accept a string w

- Three conditions should be satisfied:
  - The DFA halts. $\equiv$ **h**
  - All symbols of w are consumed. $\equiv$ **c**
  - The DFA is in an accepting state. $\equiv$ **f**

$$(h \wedge c \wedge f) \leftrightarrow a$$

- For DFAs, h and c are equivalent.
- So, the logical statement of accepting a string is:

$$(c \wedge f) \leftrightarrow a$$

How DFAs reject a string w

- We negate the previous statement:

$$\sim (c \wedge f) \leftrightarrow \sim a$$

$$\equiv (\sim c \vee \sim f) \leftrightarrow \sim a$$

- Translation:
  - At least one symbol is NOT consumed.

  OR

  - The DFA is NOT in an accepting state.

Any question?

NAME	Alan M. Turing		
SUBJECT	CS 154	TEST NO.	2
DATE	09/12/2019	PERIOD	1 / 2 / 3

TEST RECORD	
PART 1	123
PART 2	
TOTAL	

Your list # goes here!

Quiz 2

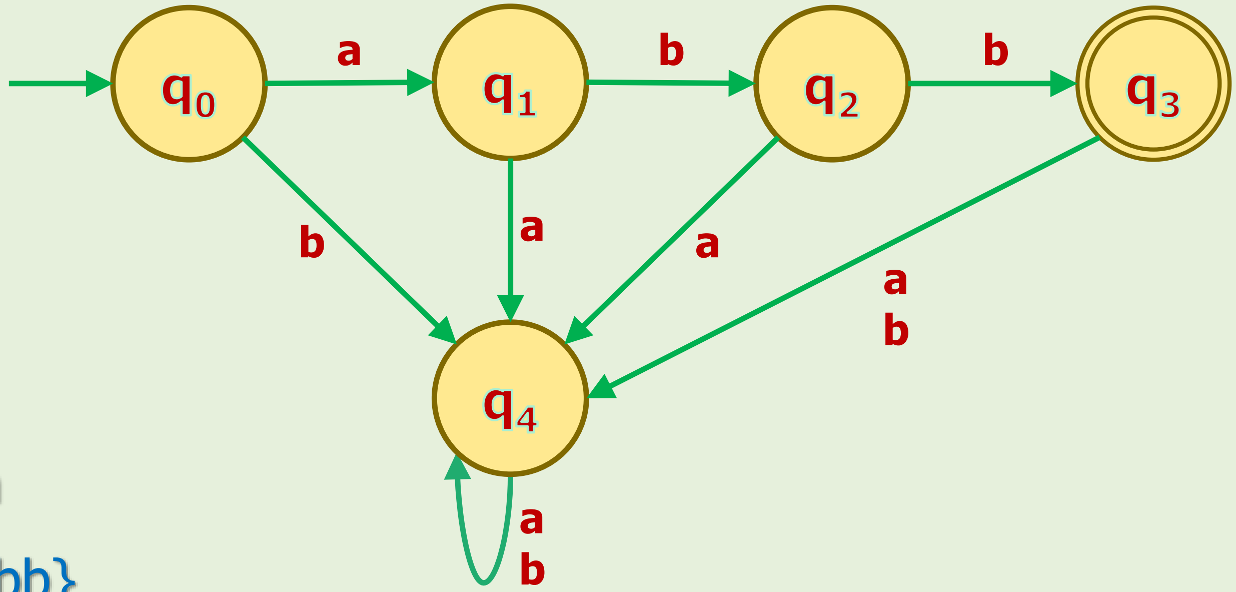
Use Scantron

Analysis Examples

Analysis Examples

Example 9

- What language does the following DFA accept?



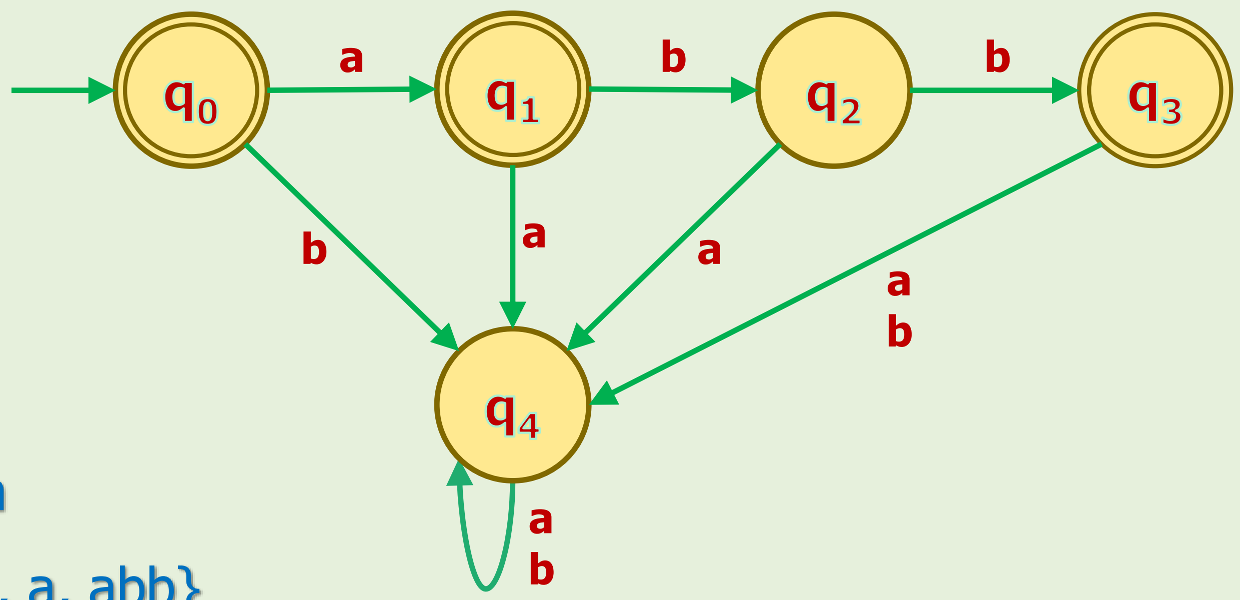
Solution

- L = {abb}

Analysis Examples

Example 10

- What language does the following DFA accept?

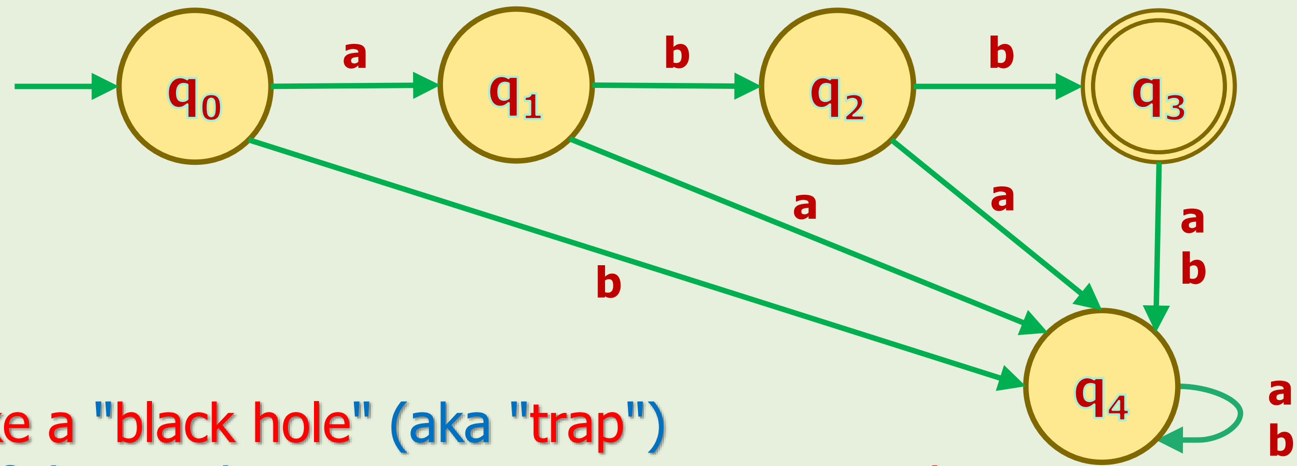


Solution

- $L = \{\lambda, a, abb\}$

Analysis Examples: Notes

1. We don't need to show the "output" and the "clock" any longer!
2. The role of q_4 in the previous examples:

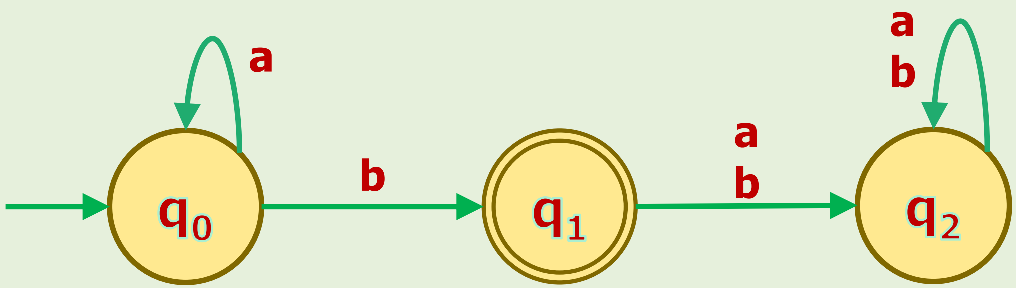


- q_4 acts like a "black hole" (aka "trap")
  because if the machine transits to q_4, it gets stuck
  in it and would not be able to transit to any other states.
- We use it to reject majority strings of $\overline{L}$.
- Sometimes I call it "hell"!
- Now we understand the CS equivalent of "Go to hell!"!

Analysis Examples

Example 11

- What language does the following DFA accept?
  - Represent the language by a set builder.



Solution

Design Examples

Design Examples

Example 12

- Let $L = \{a^n : n \geq 0\}$ over $\Sigma = \{a , b\}$.

  a. Design a DFA to accept L.

  b. What would be the design if $n \geq 1$?

Solution

Design: Note

- To test your machine, all accepted strings and rejected strings should be picked from Σ^*.
  - We are not allowed to input strings from outside of Σ^*.

A million dollar question!

- Can you ever claim that your design (code) is bug-free?
- Never, because Σ^* is infinite
  and you cannot test your code with infinite test cases.
- So, theoretically every design (code) has potential bugs!

This is me!

Design Examples

Example 13

- Let L be the set of strings starting with prefix ab over $\Sigma = \{a , b\}$.

  a. Write a set-builder for L.

  b. Design a DFA to accept L.

Solution

Design Examples

Example 14

- Let L be the set of strings that contains even number of 1's over $\Sigma = \{1\}$.

   a. Write a set-builder for L.

   b. Design a DFA to accept L.

Solution

Design Examples

Example 15

- Let L be the set of strings that contains even unary numbers over $\Sigma = \{1\}$.

  a. Write a set-builder for L.

  b. Design a DFA to accept L.

Solution

Design Examples: DFA over $\Sigma = \{a , b\}$

Example 16: Empty Language

- $L = \{ \}$

Example 17: All Strings

- $L = \Sigma^*$

Example 18

- $L = \{\lambda\}$

Homework

- $L = \Sigma^+$

Homework: DFA Design

- For each of the following languages over $\Sigma = \{a , b\}$:
  a. Write a set-builder to represent the language.
  b. Design a DFA to accept the language.

1. The set of strings containing exactly one 'a'
2. The set of strings containing at least one 'a'
3. The set of strings ending with suffix ab

Homework: DFA Design

- For each of the following languages over $\Sigma = \{a , b\}$:
  a. Write a set-builder to represent the language.
  b. Design a DFA to accept the language.

1. All strings with no more than three 'a's
2. All strings with at least three 'a's

Homework: DFA Design

- Design a DFA for each of the following languages over $\Sigma = \{a , b\}$:

1. All strings that every odd positions is 'a'.
   (Indexing of symbols is 1-based.)
2. All strings that no two consecutive a's occur.
3. All strings that does not end with ab.
4. All strings in which every a is followed by bb.

Homework: DFA Design

- Design a DFA over $\Sigma = \{a, b\}$ for each of the following languages:

1. All strings with exactly one 'a' and exactly two 'b's
2. All strings with at least one 'a' and exactly two 'b's
3. All strings with exactly two 'a's and more than two 'b's

References

1. Linz, Peter, "An Introduction to Formal Languages and Automata, 5th ed.," Jones & Bartlett Learning, LLC, Canada, 2012
2. Michael Sipser, "Introduction to the Theory of Computation, 3rd ed.," CENGAGE Learning, United States, 2013
   ISBN-13: 978-1133187790