San José State University
Department of Computer Science

Ahmad Yazdankhah

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

Deterministic Finite Automata

(Part 1)

Lecture 06

Day 06/31

CS 154

Formal Languages and Computability

Spring 2019

Agenda of Day 06

- Announcement
- Solution and Feedback of Quiz 1
- Summary of Lecture 05
- Lecture 06: Teaching ...
  - Deterministic Finite Automata (Part 1)

Announcement

- Please check your name spelling in the rollcall form and correct it if it is misspelled.

- Rollcall does NOT have any impact on your score.
- I use it for recommendation, choosing graders, outstanding graduating senior awards, scholarships, and so on.

- Canvas is supposed to send you a notification for assignments, announcements, etc..
- Sometimes, for some unknown reasons, it doesn't!
- So, that's why it is your responsibility to check Canvas at least once per day.

Solution and Feedback of Quiz 1 (Out of 20)

Section	Average	High Score	Low Score
01 (TR 3:00 PM)	13.73	19.5	5
02 (TR 4:30 PM)	13.05	18	6.5
03 (TR 6:00 PM)	14.11	19	6.5

Summary of Lecture 05: We learned ...

Operations on Languages

- Regular set operations
  union, intersection, minus

- Complement of L

  $\overline{L} = U - L = \Sigma^* - L$

- Reverse of L

  $L^R = \{w : w^R \in L\}$

- Concatenation of L_1 and L_2

  $L_1L_2 = \{xy : x \in L_1, y \in L_2\}$

  $\phi L = L \phi = \phi$

  $\{\lambda\} L = L \{\lambda\} = L$

- Exponential Operator

  $L^n = L L \ldots L$ (n times concatenation)

  $L L^n = L^n L = L^{n+1}$

  $L^0 = \{\lambda\}$

- Some surprising languages were introduced.

- Conclusion:
  All data in CS are strings.

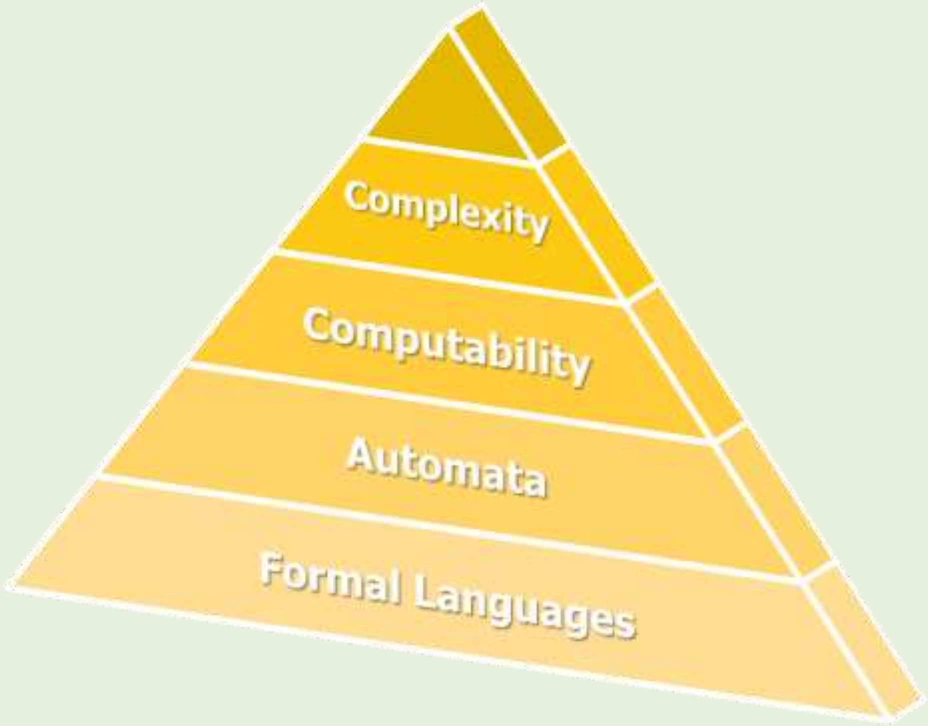
Any question?

The Big Picture of the Course

Recap

- We finished the "first round" of Formal Languages!
  - We'll get back to it several times later.

- Let's start "Automata"!



Automata

Objective of Automata Lectures

- In the previous lectures, we defined formal languages.
  - That are the mathematical model of all type of languages.

- From this lecture onward, we start constructing machines.

- These machines are supposed to "understand" formal languages.

Automata

- These machines are called "automata" (plural of automaton).
  - This is a scientific name for computation machines.
- In fact, they are mathematical models of computing devices.

Definition

- Automaton is a mathematical model of a computing device.

- In this course, we'll define several "classes of automata".
- Each class has different "power".
  - The definition of "power" will come later.
- In fact, we'll witness the evolution of languages and automata.

Automata: How They Operate on Strings of L and $\overline{L}$

- When we feed them the strings of language L, they understand (aka Accept).

- But when we feed them the strings of $\overline{L}$, they do not understand (aka Reject).

Automata

- We start with the simplest class of automata called:

  Deterministic Finite Automata (DFA)

- To introduce a new class of machines, we'll use the next slide template.

Template for Introducing a New Class of Automata

- To construct a new class of automata, we need to deal with the following items:

1. Why do we need a new class of machines? (Justification)
2. Name of the new class
3. Building blocks of the new class
4. How they work
   - 4.1. What is the starting configuration?
   - 4.2. What would happen during a timeframe?
   - 4.3. When would the machines halts?
   - 4.4. How would a string be Accepted/Rejected?
5. The automata in action
6. Formal definition
7. Their power: this class versus previous class
8. What would be the next possible class?

Deterministic Finite Automata

1. Why do we need a new class of machines?

- This is the first class and really we don't need to justify it!

- In fact, the main goal of introducing this class of machines was mentioned in the objective of these lectutres:

  "These machines are supposed to understand formal languages."

2. Name of the new class

- We name this class of automata as:

Deterministic Finite Automata (DFA)

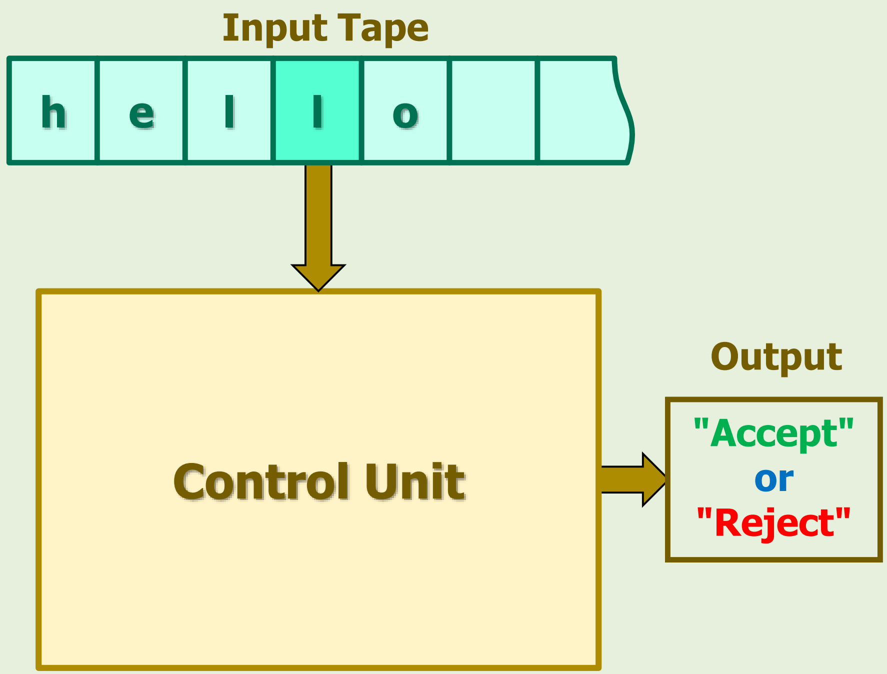
Where is this name coming from?

- We have already know that "automata" means machines.
- Also we know the meaning of "finite" but
  we don't know what part of the machine is finite?
- Also we don't know the meaning of "deterministic".
- Both of these will be explained later.

3. DFAs Building Blocks

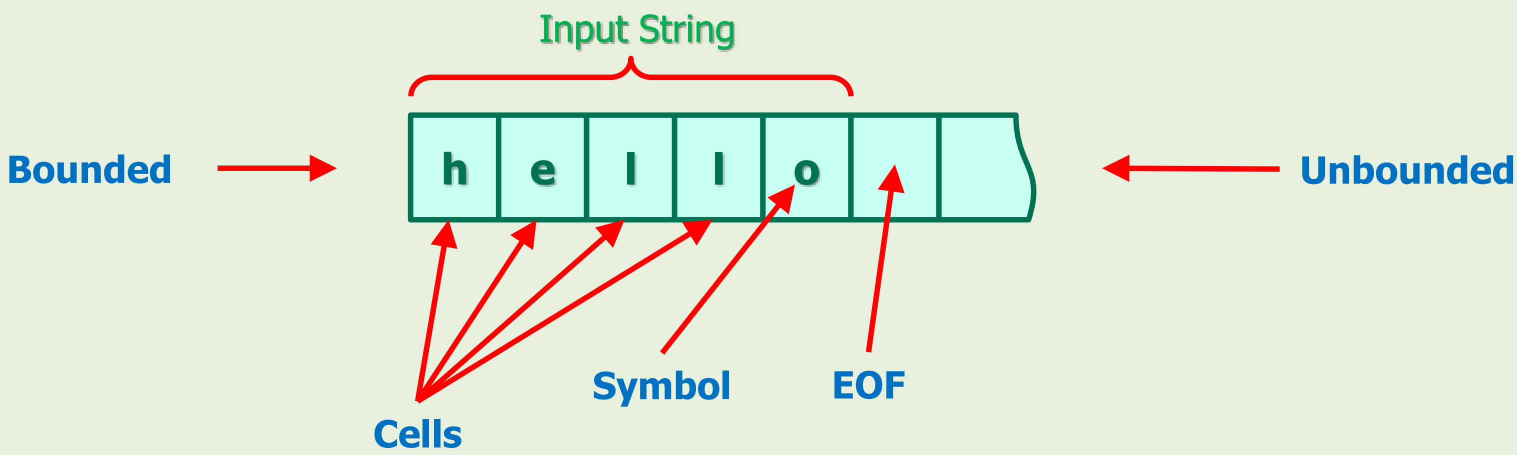
3. DFAs Building Blocks

- A DFA has 3 main blocks:
  1. Input Tape
  2. Control unit
  3. Output



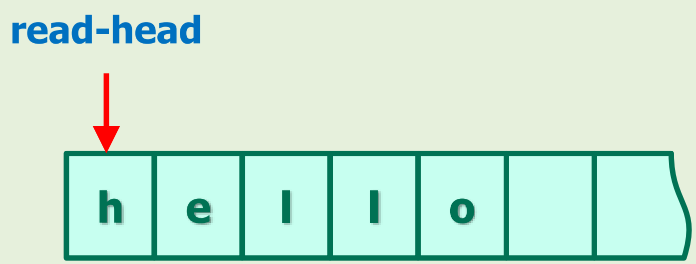
- Let's see each block in detail.

3.1. Input Tape: Structure



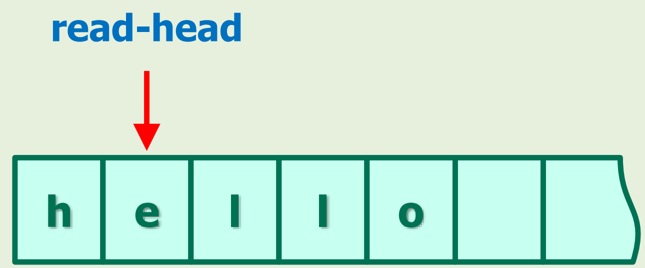
- The tape is "unbounded" from the right side and bounded from the left.
- It is divided into "cells".
- Each cell can hold one "symbol".
- The input data is a string written on the tape from the left-most cell.
- We show the end of a string as a blank cell and we call it EOF (stands for "End Of File").

3.1. Input Tape: How It Works



- The tape has a "read-head".
- It can read one symbol at a time.
- The read-head reads the symbol at which it is pointing and sends it to the control unit.
- Then, the control unit commands the head to move one cell to the right.
- We call these two operations as "consuming of the symbol".
- Consuming = reading + moving the read-head to the right

3.1. Input Tape: Notes

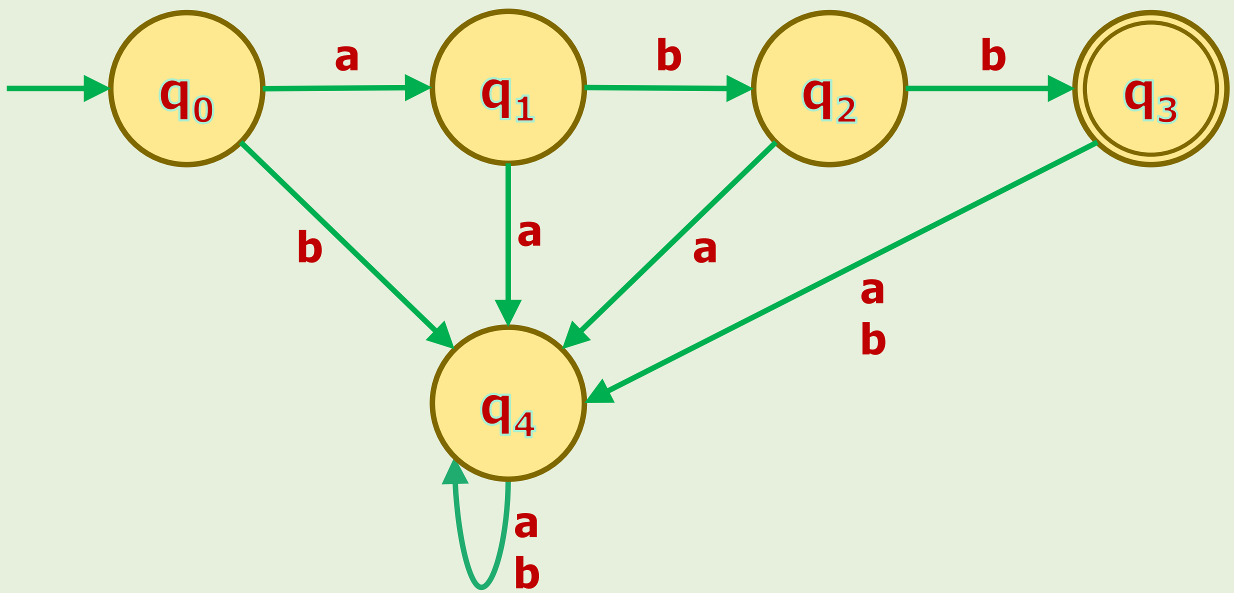


1. We use the words reading, and scanning interchangeably.
2. DFAs can "read" the input string but cannot change it. (Read-Only Tape)
3. The head only moves from left to right.
   - So, during the machine's operations, when the head moves one symbol to the right, there is no way to move it back. (Irreversible Operation)
4. The input mechanism can detect the end of the input string (EOF).
   - For example, in the above illustration, when the machine consumes 'o', it knows that it is the last symbol of the string!

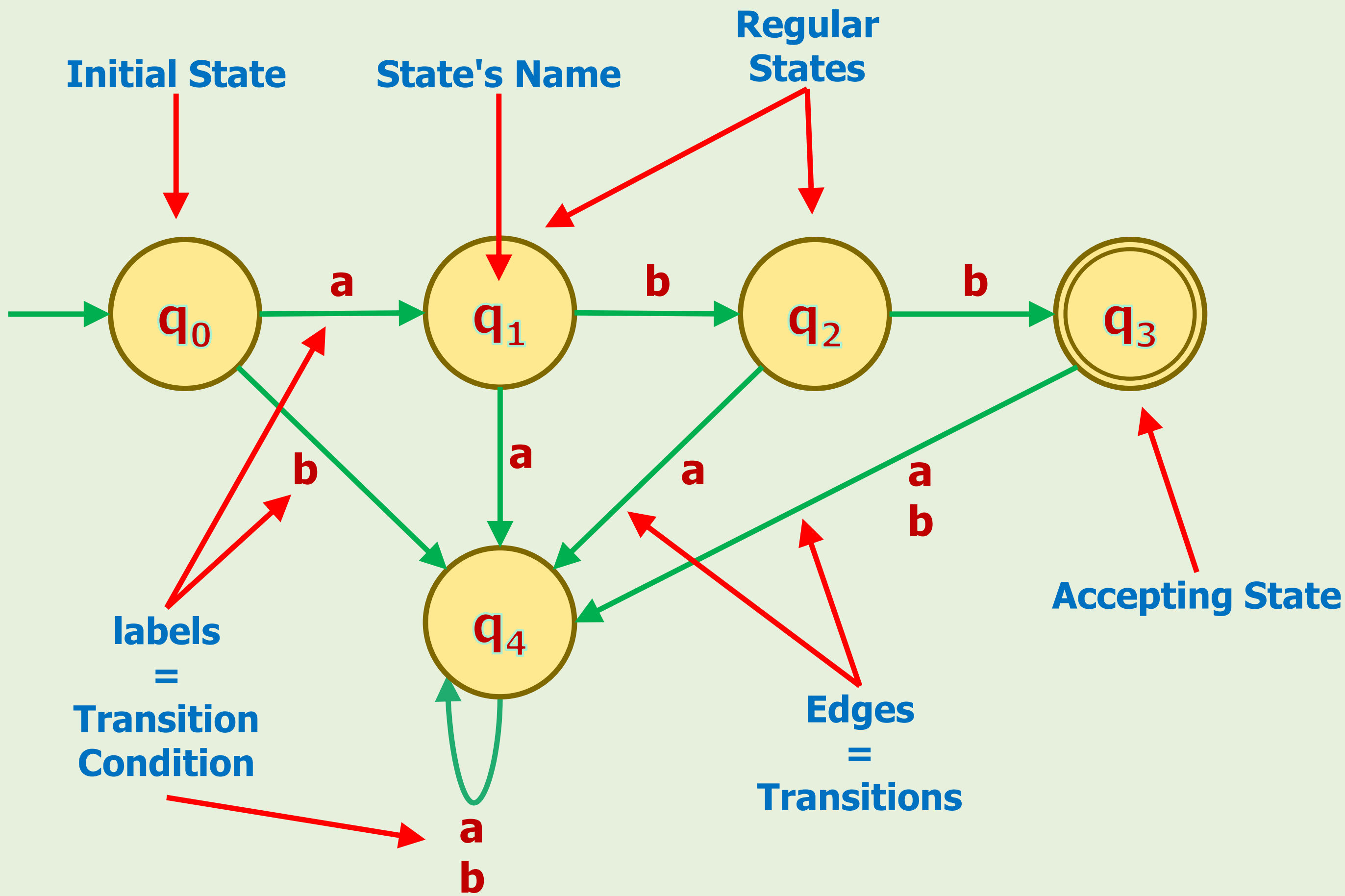
3.2. Control Unit: Structure

- The control unit is the brain (CPU) of DFAs.
- We represent its decision making part by a graph called "transition graph".
- The following example shows an instance of a transition graph.

Example 1

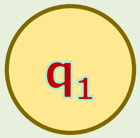


3.2. Control Unit: Transition Graph Ingredients

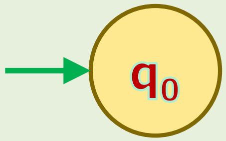


3.2. Control Unit: Transition Graph Ingredients

- A vertex of the graph represents a "state of the DFA".
  - The label q_1 is the name of the state.

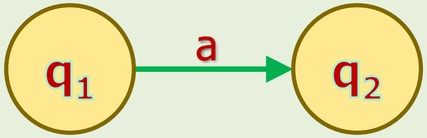


- The "initial state" is identified by an incoming unlabeled arrow, not originated from any vertex.
  - We usually name it q_0 but it can be named anything else.
  - The machine is restricted to have one and only one initial state.



3.2. Control Unit: Transition Graph Ingredients

- An edge between two vertices represents a "transition".
- The label on an edge is the "transition condition". (will be covered later.)

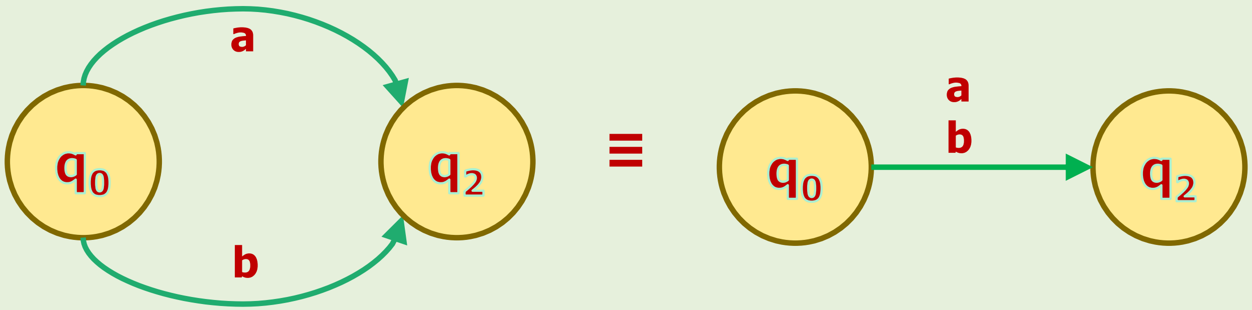


- An "accepting state" (aka "final state") is shown by double circle.

  – Transition graph can have zero or more final states.

3.2. Control Unit: Notes

1. We use the following shorthand to simplify the graph.

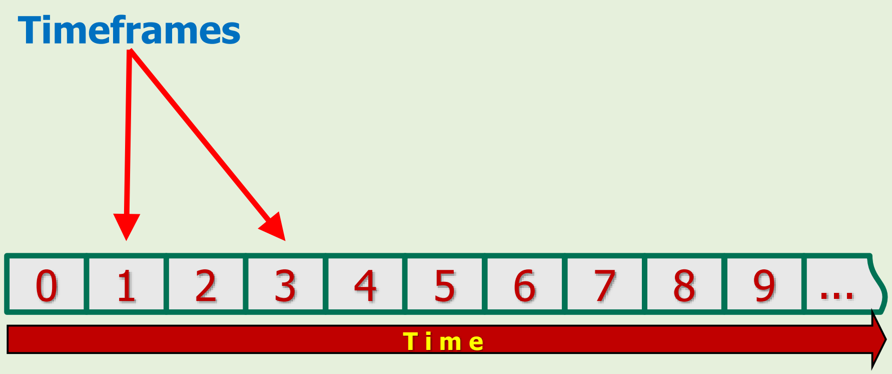


- Note that a b means "a or b".
- In some books, you might see "a , b" that is confusing.
- Because comma usually means "AND".

2. DFAs have finite number of states.
   - That's where the "finite" in "deterministic finite Automata" comes from.

3.2. Control Unit: Synchronization of Operations

- For synchronization of operations,
  the control unit has a clock that produces discrete signals (ticks).
- We call each signal a "timeframe" (aka timestep).
  - The frequency of the clock does not matter in this course.
- In my lecture notes, I use the following figure
  to show a clock and its timeframes.



3.3. Output

- Output of DFAs have two messages:
  - Accept (aka: understood, recognized, Yes)
  - Reject (aka: not understood, not recognized, No)

Output

Accept **or** **Reject**

- If the machine understands the input string, the output will be "Accept".

- If the machine does not understand the input string, the output will be "Reject".

References

1. Linz, Peter, "An Introduction to Formal Languages and Automata, 5th ed.," Jones & Bartlett Learning, LLC, Canada, 2012
2. Kenneth H. Rosen, "Discrete Mathematics and Its Applications, 7th ed.," McGraw Hill, New York, United States, 2012
3. Michael Sipser, "Introduction to the Theory of Computation, 3rd ed.," CENGAGE Learning, United States, 2013
   ISBN-13: 978-1133187790