San José State University
Department of Computer Science

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

Deterministic Finite Automata

(Part 3)

Lecture 08

Day 08/31

CS 154

Formal Languages and Computability

Fall 2019

Agenda of Day 08

- Solution and Feedback of Quiz 2
- Summary of Lecture 07
- Lecture 08: Teaching ...
  - Deterministic Finite Automata (Part 3)
- JFLAP Demo

Solution and Feedback of Q2 (Out of 25)

Section	Average	High Score	Low Score
01 (TR 3:00 PM)	19.37	24	10
02 (TR 4:30 PM)	18.5	23	14
03 (TR 6:00 PM)	20.1	24	14.5

Summary of Lecture 07: We learned ...

DFAs

- The role of trap in DFAs ...

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

Any question?

Definitions

Formal Definition of DFAs

- Here is the formal (mathematical) definition of DFAs:
- A DFA M is defined by a quintuple (5-tuple):

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

- Where:
  - Q is a finite and nonempty set of states of the transition graph.
  - Σ is a finite and nonempty set of symbols called input alphabet.
  - δ is called transition function (aka delta function) and is defined as:

$$\delta: Q \times \Sigma \rightarrow Q$$

δ is total function.

  - $q_0 \in Q$ is the initial state of the transition graph.
  - $F \subseteq Q$ is the set of accepting states of the transition graph.

Formal Definition of DFAs

Set of States

Transition Function

Set of Final States

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

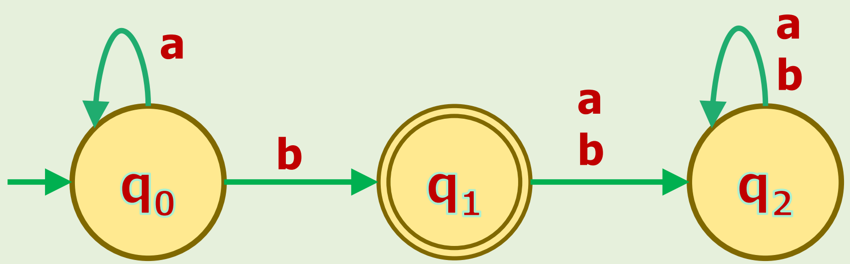
Input Alphabet

Initial State

Formal Definition of DFAs: Example

Example 22

- Consider the following DFA:

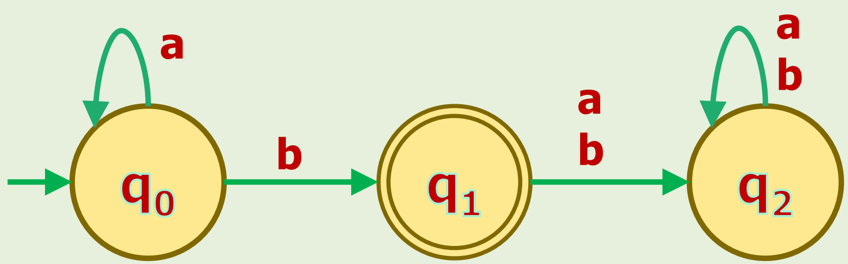


- Find all elements of $M = (Q, \Sigma, \delta, q_0, F)$.

Solution

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- For δ, look at the next slide.
- $q_0 = q_0$
- $F = \{q_1\}$

Formal Definition of DFAs: Example



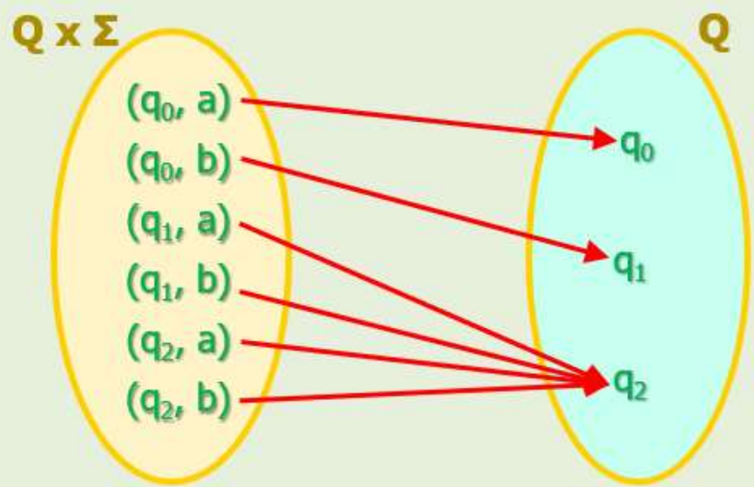
Example 22 (cont'd)

- $M = (Q, \Sigma, \delta, q_0, F)$
- $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$
- $\delta : Q \times \Sigma \rightarrow Q$
- Domain: $Q \times \Sigma = \{(q_0, a), (q_0, b), (q_1, a), (q_1, b), (q_2, a), (q_2, b)\}$
- Range: $Q = \{q_0, q_1, q_2\}$
- Function Rule:

$$\begin{cases} \delta(q_0, a) = q_0 \\ \delta(q_0, b) = q_1 \\ \delta(q_1, a) = q_2 \\ \delta(q_1, b) = q_2 \\ \delta(q_2, a) = q_2 \\ \delta(q_2, b) = q_2 \end{cases}$$

Algebraic Notation

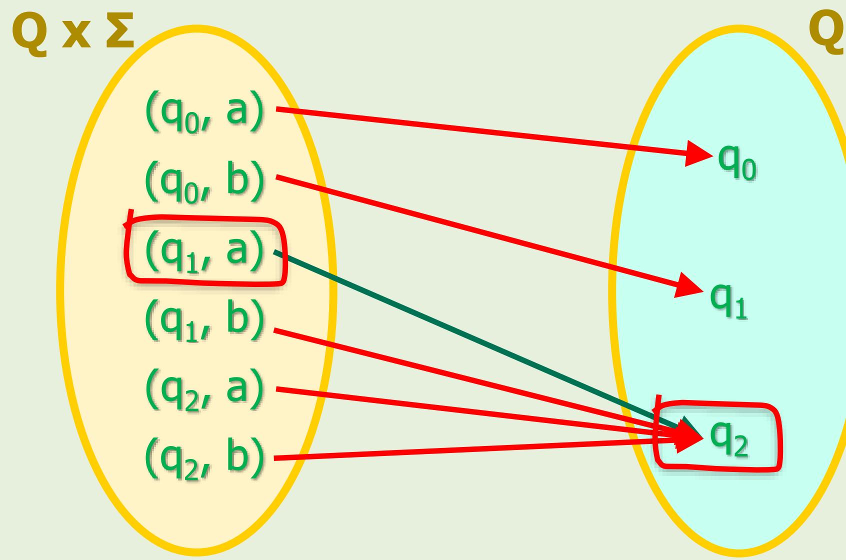
Venn Diagram

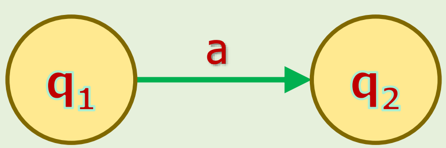


Formal Definition of DFAs: Notes

- As the previous example showed, δ is a "total function".
- Every sub-rule of δ is a transition in transition graph.
- For example:

$$\begin{cases} \delta(q_0, a) = q_0 \\ \delta(q_0, b) = q_1 \\ \boxed{\delta(q_1, a) = q_2} \\ \delta(q_1, b) = q_2 \\ \delta(q_2, a) = q_2 \\ \delta(q_2, b) = q_2 \end{cases}$$





Homework

- Draw a transition graph for the DFA M defined as:

  $Q = \{q_0, q_1, q_2, q_3\}$

  $\Sigma = \{a , b\}$

$$\begin{cases} \delta(q_0, a) = q_1 \\ \delta(q_0, b) = q_3 \\ \delta(q_1, a) = q_3 \\ \delta(q_1, b) = q_2 \\ \delta(q_2, a) = q_2 \\ \delta(q_2, b) = q_2 \\ \delta(q_3, a) = q_3 \\ \delta(q_3, b) = q_3 \end{cases}$$

  Initial state = q_0

  $F = \{q_2\}$

Homework

- Draw a transition graph for

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

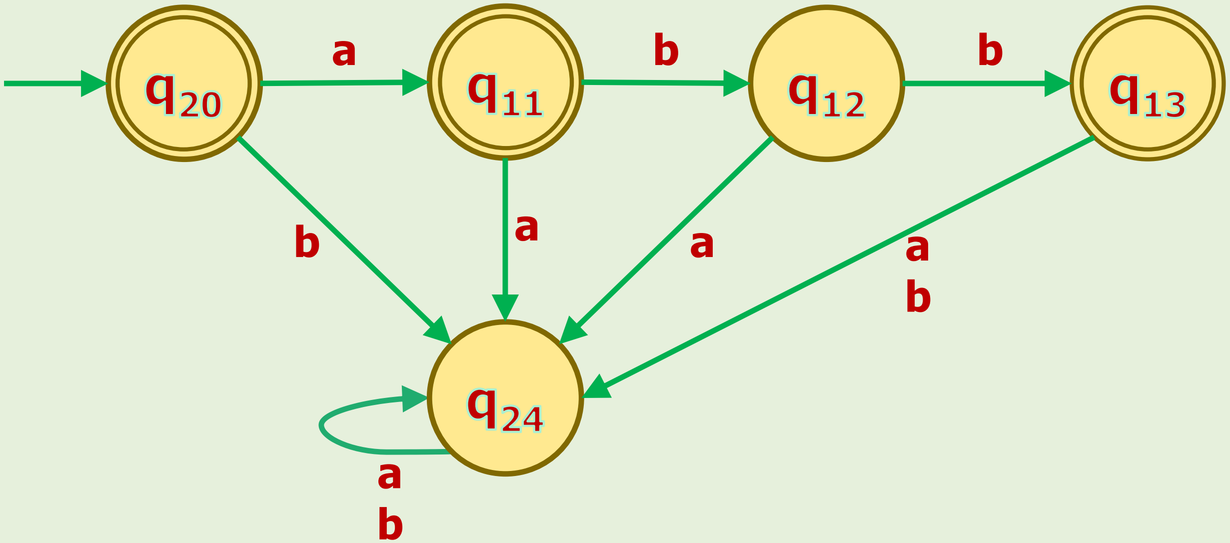
$$\begin{cases} \delta(q_0, 0) = q_0 \\ \delta(q_1, 0) = q_0 \\ \delta(q_2, 0) = q_2 \\ \delta(q_0, 1) = q_1 \\ \delta(q_1, 1) = q_2 \\ \delta(q_2, 1) = q_1 \end{cases}$$

- Which strings from the following set are accepted?

{01, 00, 101, 0111, 11001, 100, 1100}

Homework

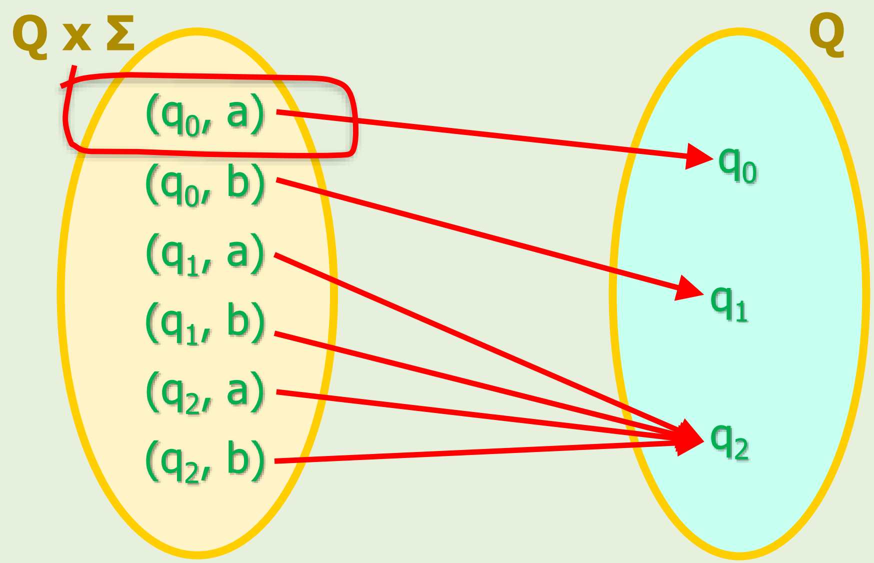
- Write all elements of the following transition graph.
- $Q = ?$
- $\Sigma = ?$
- $\delta = ?$
- $q_0 = ?$
- $F = ?$



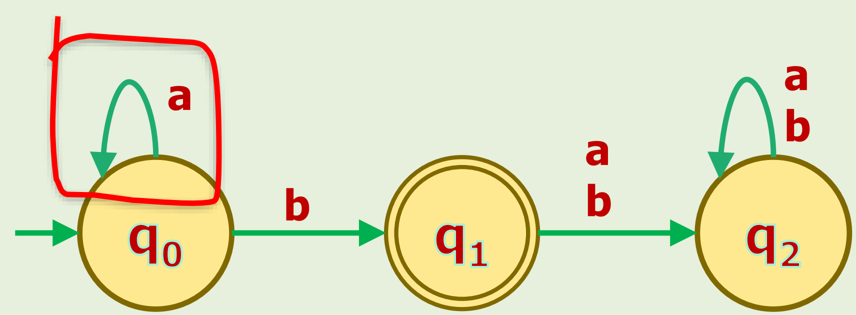
Why δ is Total Function

Example 22 (cont'd)

- What would happen if δ is NOT total function?
- Then at least one member of domain is undefined!



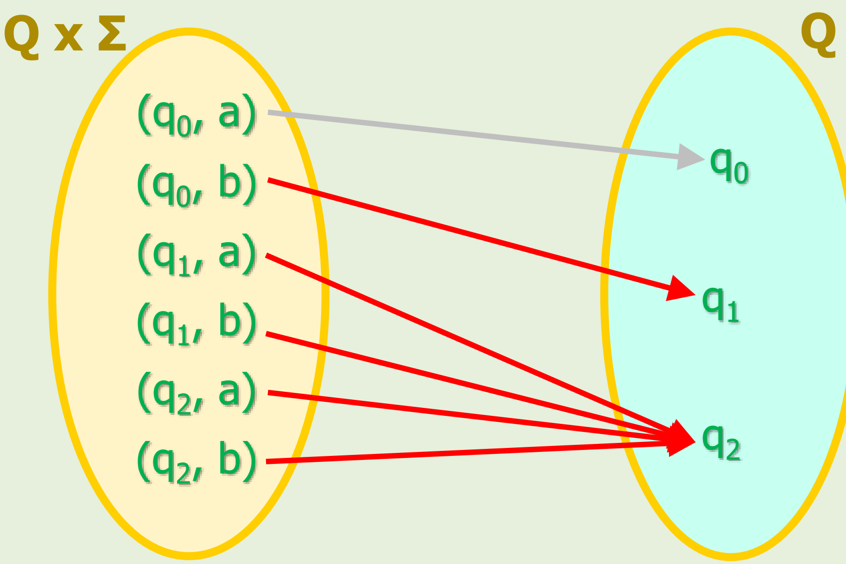
- To see the effect, let's modify δ by making (q_0, a) undefined.



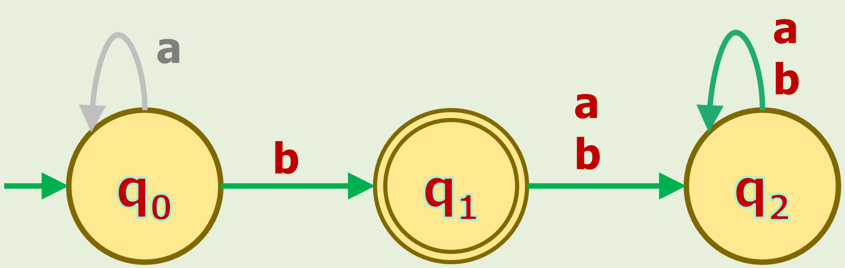
Why δ is Total Function

Example 22 (cont'd)

- What is the effect of δ being partial function?
- If the DFA is in q_0 and the input is a, it does not know what to do?
- To resolve this issue, we have two options:
  1. Define a new behavior
  2. Prevent it being partial

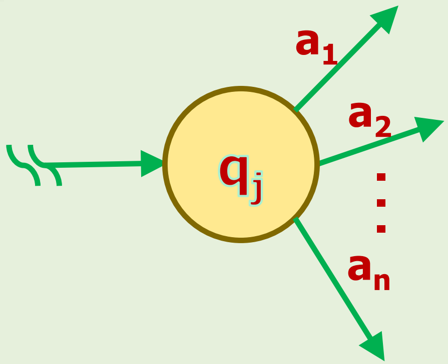


- The second option has been chosen for DFAs!



DFAs' δ from Different Angle: $\delta : Q \times \Sigma \rightarrow Q$

- DFAs must know where to go at any timeframe.
- This means, in general:
  - If $\Sigma = \{a_1, a_2, \ldots, a_n\}$ is the alphabet of a DFA, then ...
  - ... every state $q_j \in Q$ must have one and only one outgoing transition for all a_k where $k = 1, 2, \ldots, n$.



- This is "DFAs' constraint".

  In other words:
  At any timeframe, there is one and only one transition for every symbols of Σ.

Transition Table

- To represent a transition function,
  we can also use a table called "transition table".

Example 23

- Represent the following transition function by a transition table.

$$\delta: \begin{cases} \delta(q_0, a) = q_0 \\ \delta(q_1, a) = q_0 \\ \delta(q_2, a) = q_2 \\ \delta(q_0, b) = q_1 \\ \delta(q_1, b) = q_2 \\ \delta(q_2, b) = q_1 \end{cases}$$

Alphabet

δ	a	b
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_2	q_1

Current States

Range: Next State

Associated Language to DFAs

- Every DFA is designed to accept a set of strings.
  - Of course the set can be empty!
- A set of string is called language.
- Therefore, every DFA accepts a language.

Definition

- The associated language to a DFA M is
  the set of ALL STRINGS that M accepts.
- The associated language L to the machine M is denoted by L(M).

- Note that this definition can be extended to all types of automata.

Equivalency of Machines

- When are two machines M_1 and M_2 equivalent?

Definition

- Machine M_1 is equivalent to machine M_2 if $L(M_1) = L(M_2)$ over Σ.
  - M_1 and M_2 are equivalent if their associated languages are equal.

- This is also a general definition for all types of automata.

- What is wrong with the following definition?

  Two machines are equivalent if both accept the same language.

What is Computation?

- A machine during its operation transits (aka moves) from one configuration to another.
- Ultimately, when the machine halts, it accepts or rejects a string.
- This sequence of configurations is called "computation".

Definition

- "Computation" is the sequence of configurations from when the machine starts until it halts.

What is Determinism?

Etymology

- Merriam-Webster dictionary defines "determinism" as:
  - "the belief that all events are caused by things that happened before them and that people have no real ability to make choices or control what happens"

- This is a philosophical definition.
- If something is deterministic, then it will happen with 100% certainty and there won't be any other choices.

- Now let's see what does it mean in computer science world.

What is Determinism?

Is DFAs' behavior predictable?

- You, as AN OBERSERVER, are given a known DFA's configuration at timeframe n.
- Can you predict its configuration at timeframe n+1?
- Yes, we, as AN OBERSERVER, can predict its behavior with 100% certainty.
- Because at any timeframe, there is one and only one transition for every symbols of Σ.
- In other words, there is no randomness in DFAs behavior!

What is Determinism?

Definition

- A machine is called deterministic if at any timeframe, there is NO MORE THAN ONE possible transition.
  - Therefore, this definition is satisfied if ... the number of possible transitions at any timeframe is zero or one.
- Note that for DFAs, it is one and only one transition but for other deterministic machines, it could be zero or one. (Will be covered later.)

Prepare Your Development Environment

JFLAP (Java Formal Language and Automata Package)

- From now one, we'll be using this tool to develop and test our automata.

- **Download it from Canvas**: Files/Misc/JFLAP7.1.jar
- For uniformity purpose, please use the copy that I provided!
- Tutorial: http://www.jflap.org/tutorial/

JFLAP Demo

- **Official website**: http://www.jflap.org/
- The stable version 7.1 (Jul 27, 2018)

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