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 2018

Agenda of Day 06

- Rollcall form and your list-number
- Solution and Feedback of Quiz 1
- Summary of Lecture 05
- Lecture 06: Teaching ...
  - Deterministic Finite Automata (Part 1)

Solution and Feedback of Quiz 1

Metrics	Section 1	Section 2	Section 3
Average	19	18	17
High Score	23	22	22
Low Score	11	10	12

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 Operation

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

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

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

  $\{\lambda\}^n = \{\lambda\}$
- Star-closure

  $L^* = L^0 \cup L^1 \cup L^2 \cup \ldots$
- Positive-closure

  $L^+ = L^1 \cup L^2 \cup \ldots$

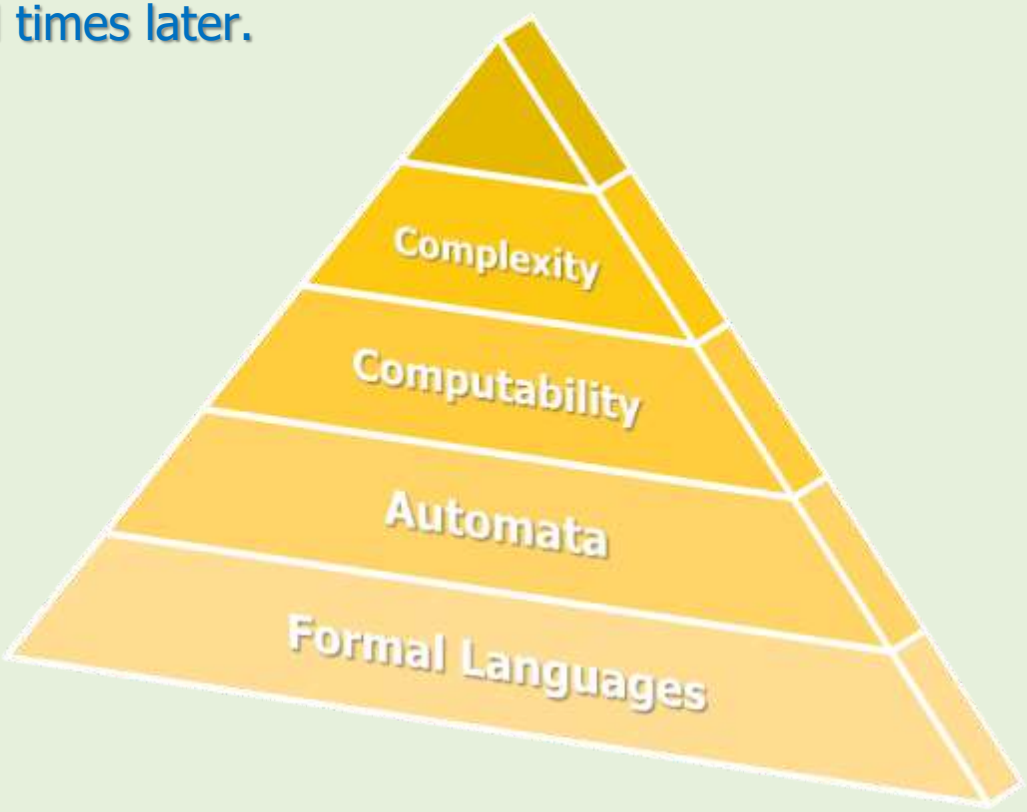
  $L^* = L^0 \cup L^+$
- We introduced some surprising languages.
- In computer science, all data are strings.

Any question?

The Big Picture of the Course

Computer Science Foundation

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



Deterministic Finite Automata

Objective of This Lecture

- In the previous lectures, we defined mathematical languages.
  - We called them "formal languages".
- In this lecture, we start constructing some "mathematical machine".
- These machines "understand" formal languages.
- We start with the simplest class of machines.

Introduction

- Those machines are called "automata" (plural of automaton).
- They are mathematical models of a computing devices.
- 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.

Introduction

- We start with the simplest class of automata called:

  Deterministic Finite Automata (DFA)

Where is this name coming from?

- So far, we have learned that "automata" means machines.
- Also we know the meaning of "finite" but what part of the machine is finite.
- Also we don't know the meaning of "deterministic".

- The reason of using "finite" and the meaning of "deterministic" will be explained later.

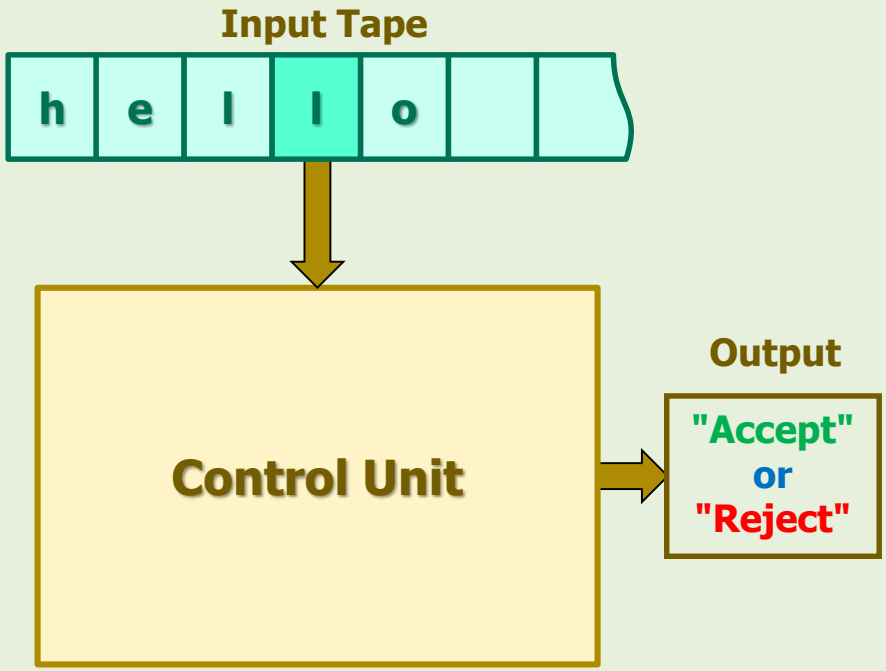
Roadmap

- Here is our roadmap to introduce DFAs:
  1. We'll describe DFAs' building blocks!
  2. We'll describe how DFAs work.
  3. We'll see DFAs in action.
  4. We'll analyze some DFAs and find out what languages they understand.
  5. We'll design some DFAs to understand the given languages.
  6. We'll have formal (mathematical) definition of DFAs

DFAs Building Blocks

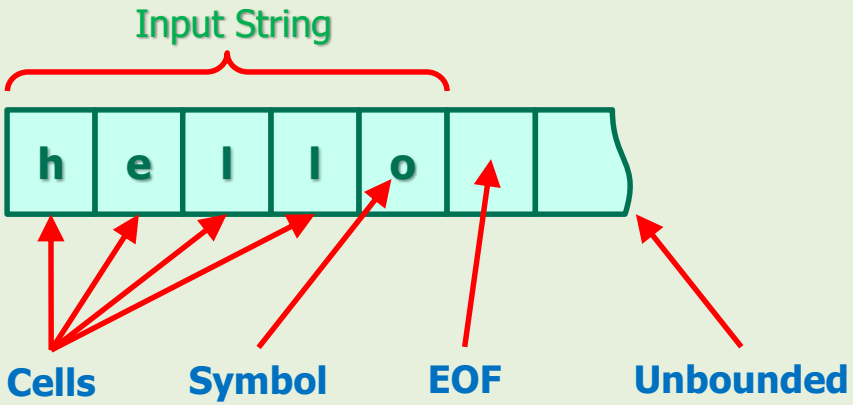
DFAs Building Blocks

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



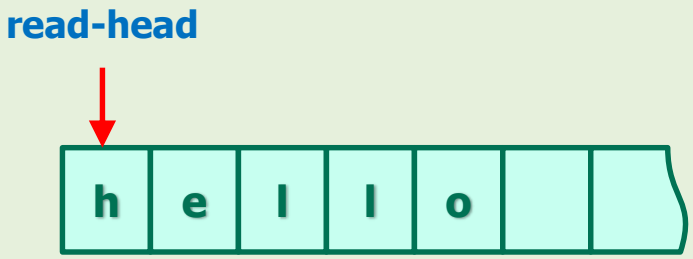
- Let's see each block in detail.

1. Input Tape: Structure



- The "input tape" is divided into "cells".
- Each cell can hold one "symbol".
- The tape is "unbounded" from right side.
- 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").

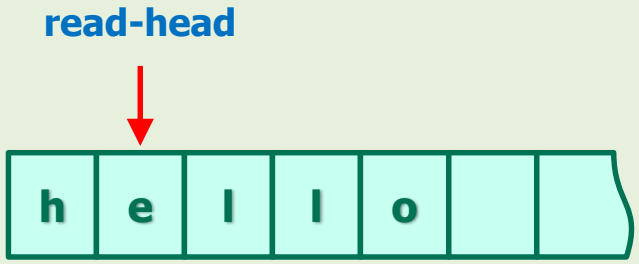
1. Input Tape: How It Works



- 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 head moves one cell to the right.

1. Input Tape: Notes



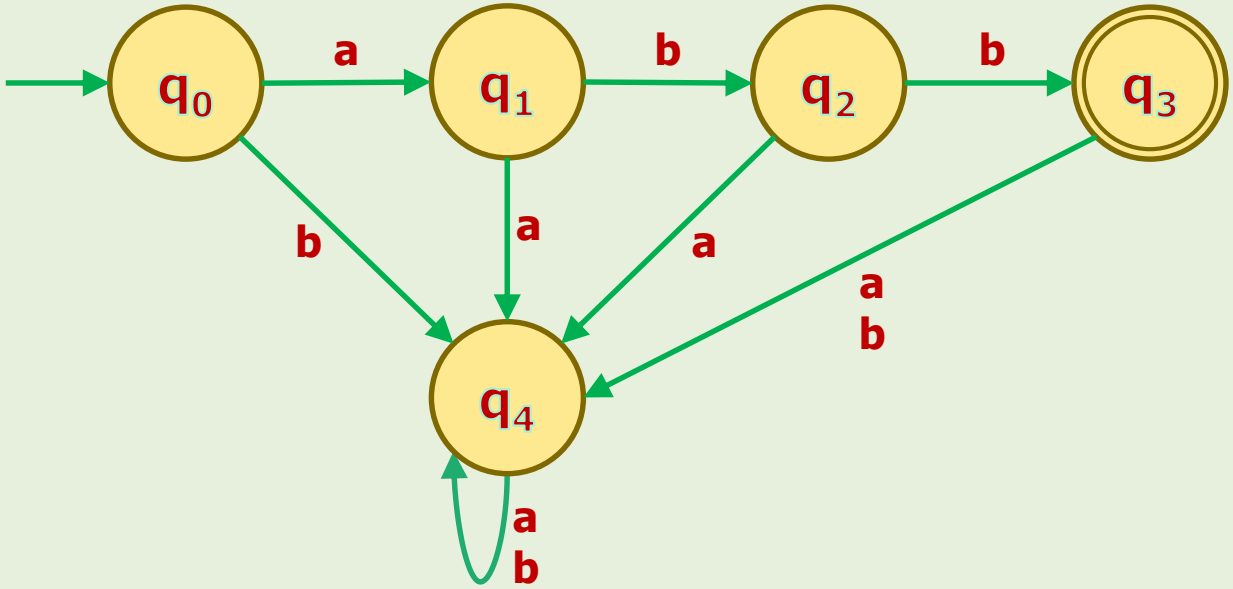
1. We use the words reading, consuming, and scanning interchangeably.
2. DFAs can "read" the input string but cannot change it. (Read-Only Tape)
3. The head only moves from left to right.
   - So, during the machine's operation, when the head moves one symbol to the right, there is no way to move it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

2. Control Unit: Structure

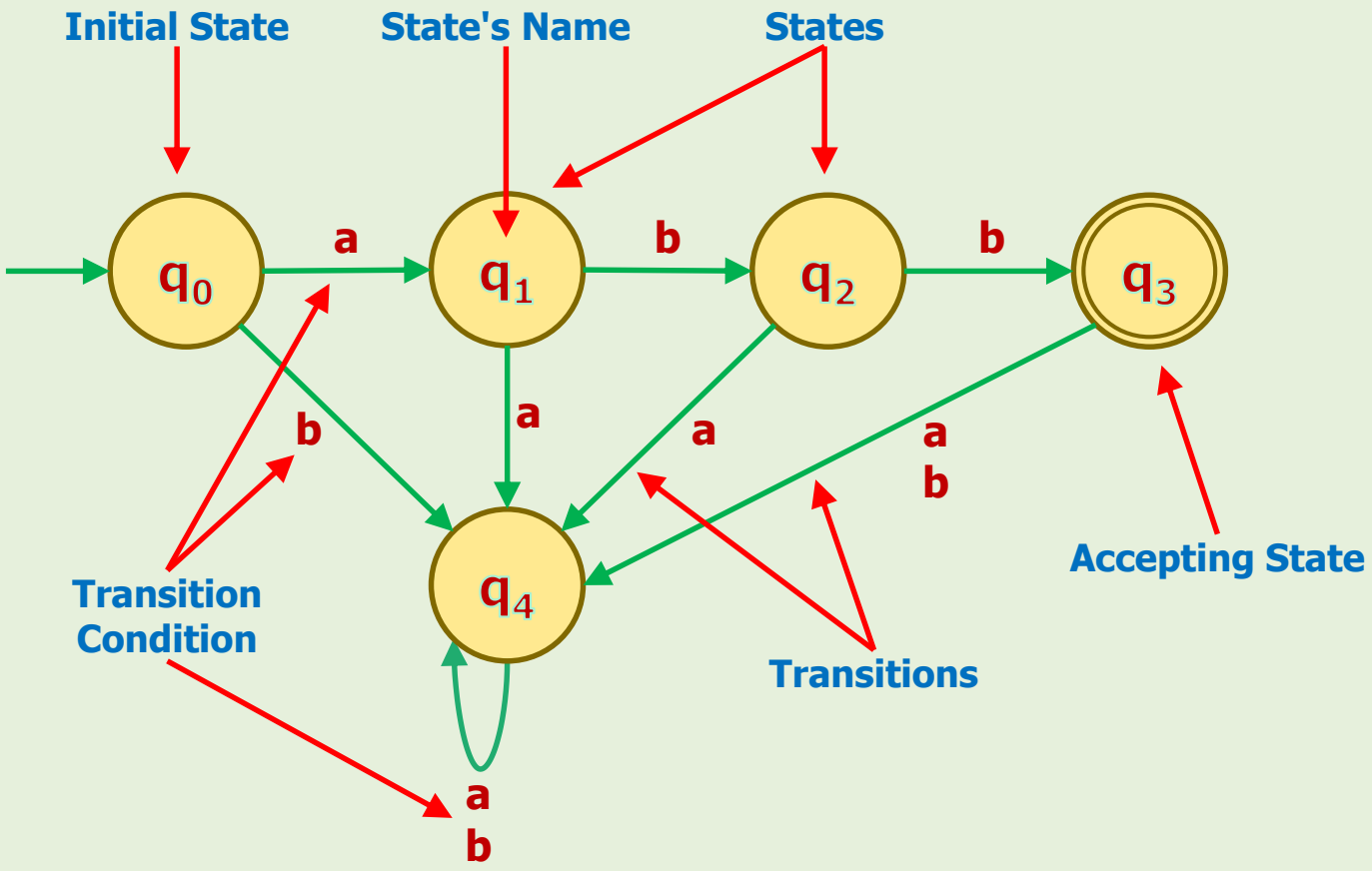
- The control unit is the brain (CPU) of DFAs.
- We represent part of it by a graph called "transition graph".

- The following example shows an instance of a transition graph.

Example 1



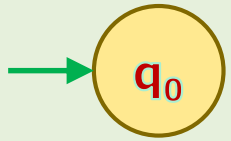
2. Control Unit: Transition Graph



2. Control Unit: Transition Graph Ingredients

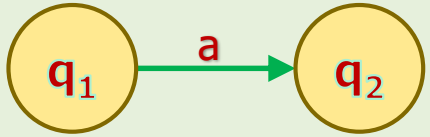
- A vertex of the graph represents a "state of the DFA".
  - The label q_1 is the name of the state.

- 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.



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.)

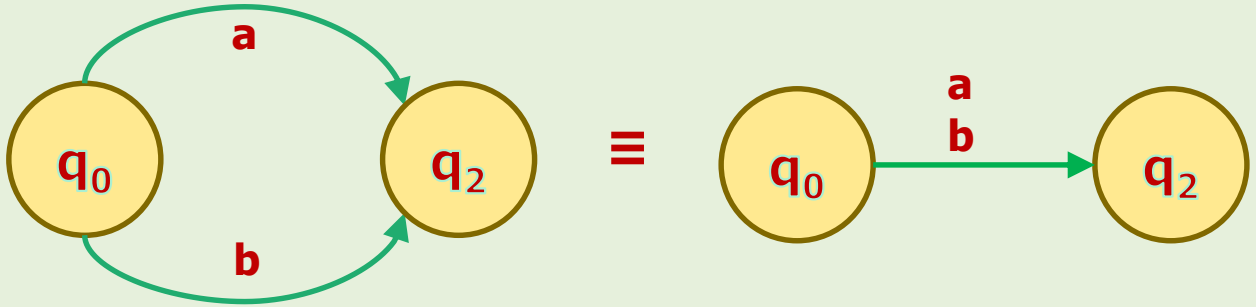


- An "accepting state" (aka "final state") is shown by double circle.

  - Transition graph can have zero or more final states.

2. Control Unit: Notes

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



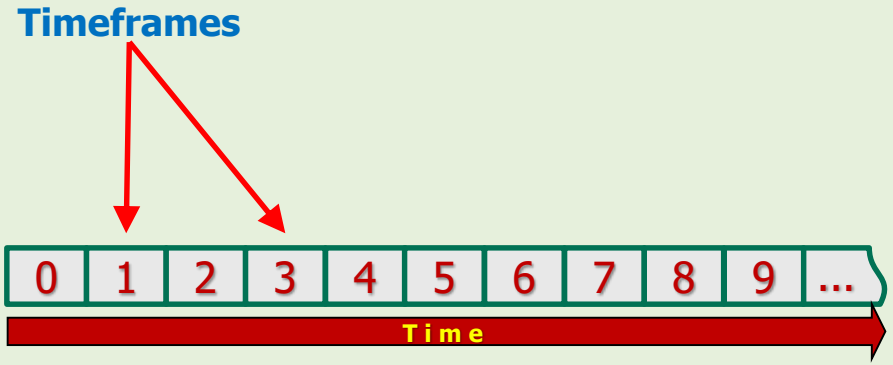
- Note that $\begin{matrix} a \\ b \end{matrix}$ means "a or b".
- In some books, you might see "a , b" that is confusing.
- Because comma usually means "AND".

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

2. Control Unit: Synchronization of Operations

- For synchronization of operations, the control unit has a clock that produces signals.
- We call each signal a "timeframe".
  - 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. Output

- Output of DFAs have two distinct values:
  - 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".

How DFAs Work

How DFAs Work

- To understand the operations of DFAs (and any other class of machines), we should clearly respond to the following questions:

1. What is the "starting configuration"?
2. What would happen during a timeframe?
3. When would the machine halt (stop)?
4. How would a string be Accepted/Rejected?

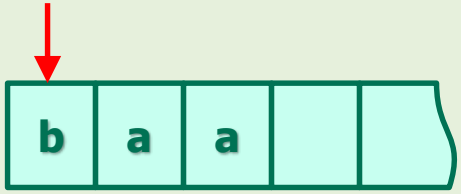
- We'll answer the first three questions right now and postpone the last one because we need some practices first.

1. DFAs Starting Configuration

Clock

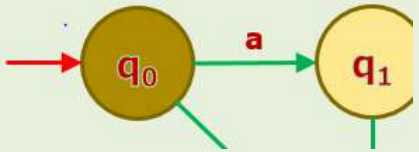
- The clock is set at timeframe 0.

Input Tape



- The input string has already been written on the tape.
- The read-head is pointing to the left-most symbol.

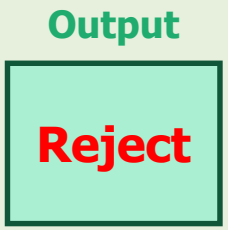
Control Unit



- The control unit is set to initial state.

Output

- The output shows "Reject".



What is Configuration?

Definition

- A DFA's configuration (aka "snapshot") is the combination of the following data:
  - Timeframe of the clock
  - Input string
  - Position of the read-head
  - Current state of the transition graph

- In fact, configuration is a snapshot of the machine's status.

2. What Happens During a Timeframe

- During a timeframe,
  the machine "transits" (aka "moves") from one configuration to another.
  - Several tasks happen during a timeframe.
  - The combination of these tasks is called a "transition".

- Let's first visualize these tasks through some examples.
- Then, we'll summarize them in one slide.

Transition Examples

Transition Examples

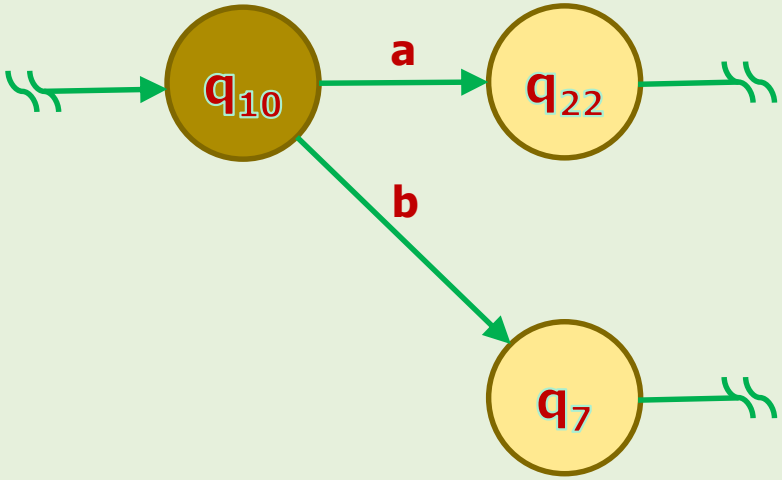
- The examples will show:
  - a partial transition graph
  - an input tape
  - a clock

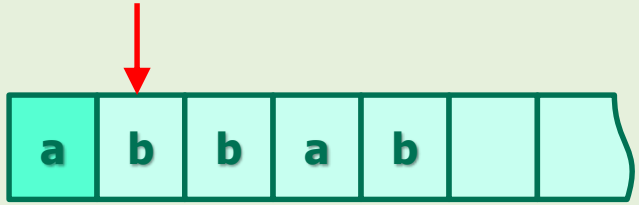
- We assume that the machine is in the middle of its operation at timeframe n.
- The question is in what configuration would the machine be at timeframe n+1?

Transition Examples

Example 2: Timeframe 1

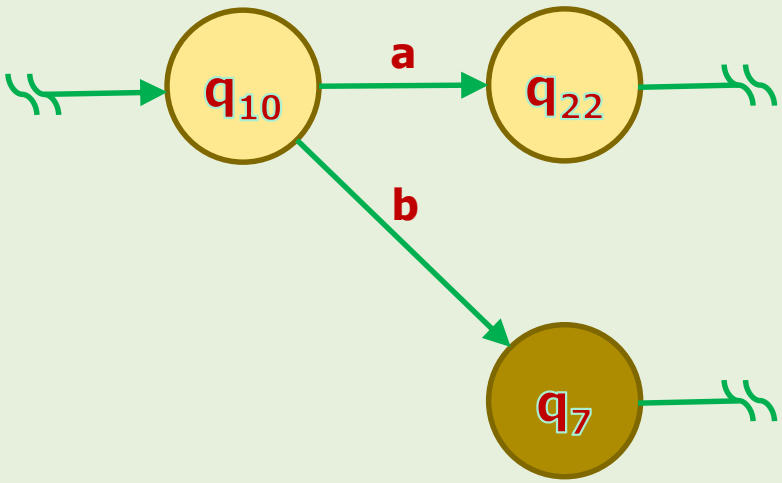
- What would be the DFA's configuration after the next timeframe?

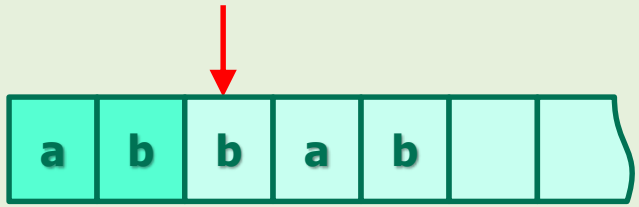




Transition Examples

Example 2: Timeframe 2

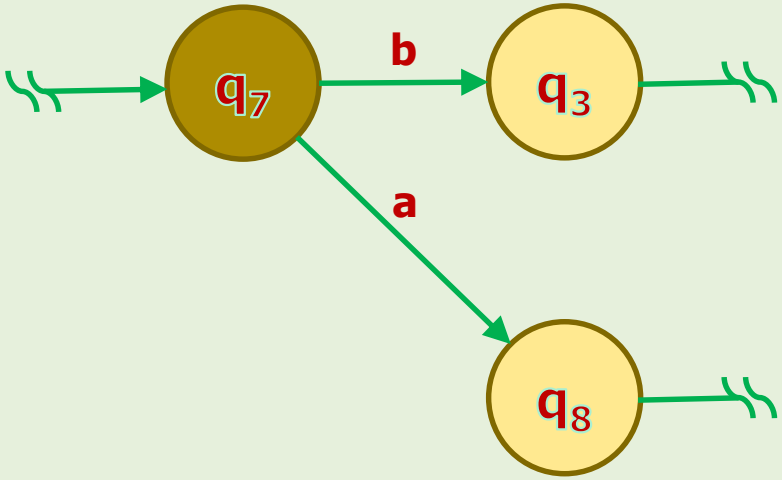


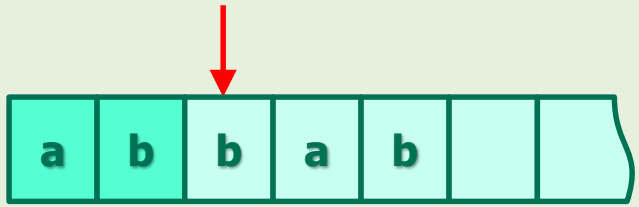


Transition Examples

Example 3: Timeframe 2

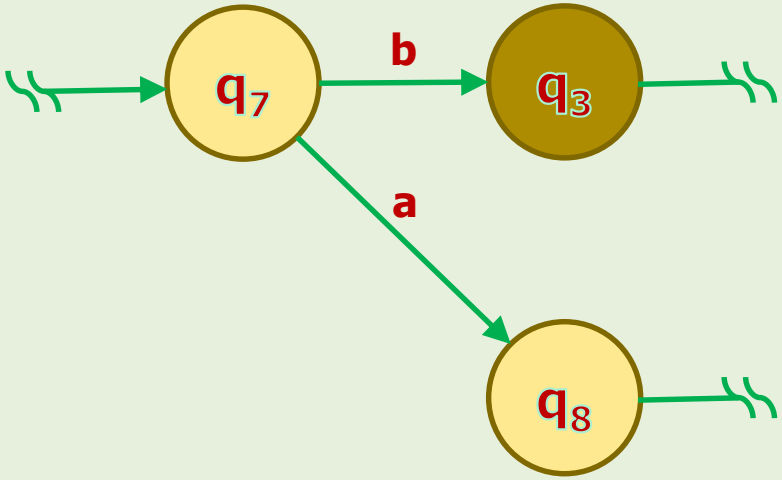
- What would be the DFA's configuration after the next timeframe?

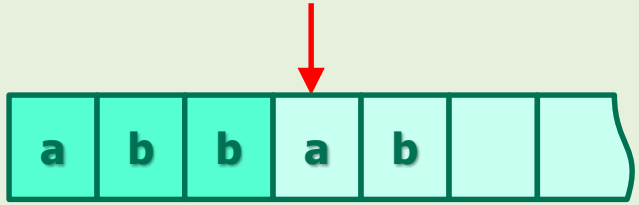




Transition Examples

Example 3: Timeframe 3

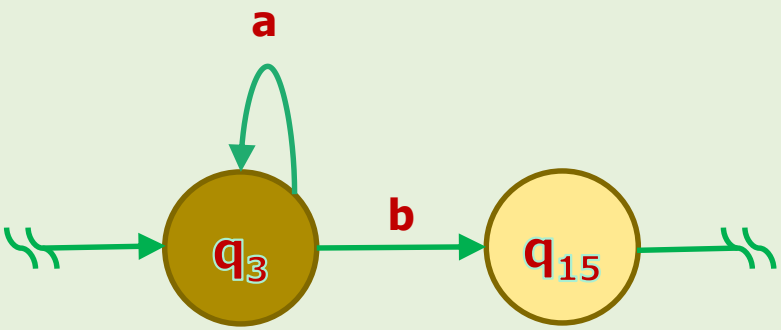


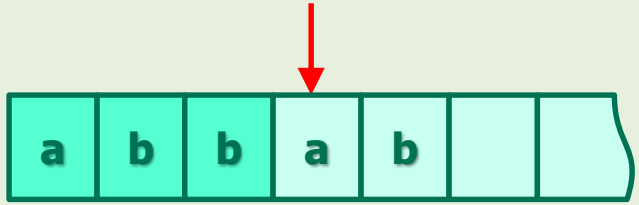


Transition Examples

Example 4: Timeframe 3

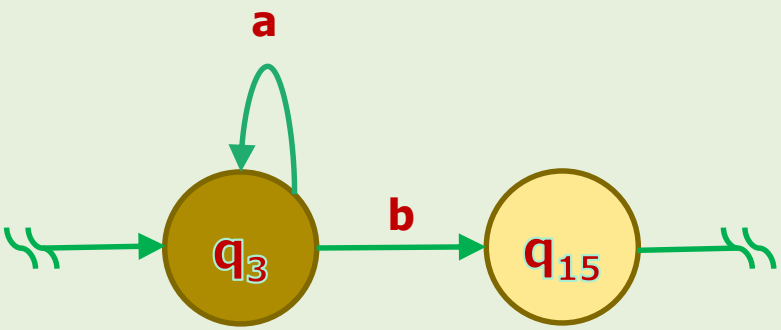
- What would be the DFA's configuration after the next timeframe?

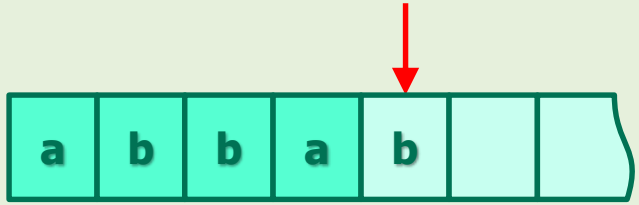




Transition Examples

Example 4: Timeframe 4





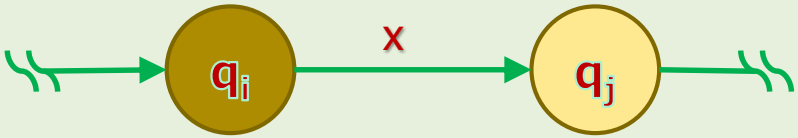
2. What Happens During a Timeframe

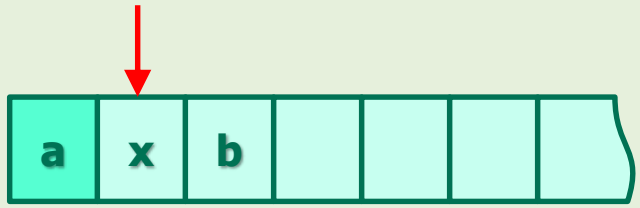
Summary of Transition

1. The symbol at which the read-head is pointing is consumed and is sent to the control unit.
2. The read-head moves one cell to the right.
3. The control unit makes its move based on the "logic of the transition".

- What is the "logic of the transition"?

The Logic of Transitions





If (Condition)

- the input symbol is 'x'

Then (Operation)

- consume 'x' †

AND

- transit to q_j

Definition

- The logic of the transition is the "decision" that the control unit makes during every timeframe.

† Consume 'x' = "read 'x', send it to control unit, and move one cell to the right."

3. When DFAs Halt

- A DFA halts iff all input symbols are consumed.

- In other words, for DFAs, the following logical statement is true:
- (All input symbols are consumed.) ↔ (machine halts.)

Recap: Biconditional

p ↔ q ≡ if and only if

Logical Representation of Halting

DFAs halt. ≡ **h**

IFF

All input symbols are consumed. ≡ **c**

} **c ↔ h**

How DFAs Work

- At this moment, our knowledge is enough to work with DFAs.
- Now let's see DFAs in action!
- We'll analyze the behavior of some DFAs.
- We'll put some strings on the tape and will follow the machine's behavior after each tic of the clock.

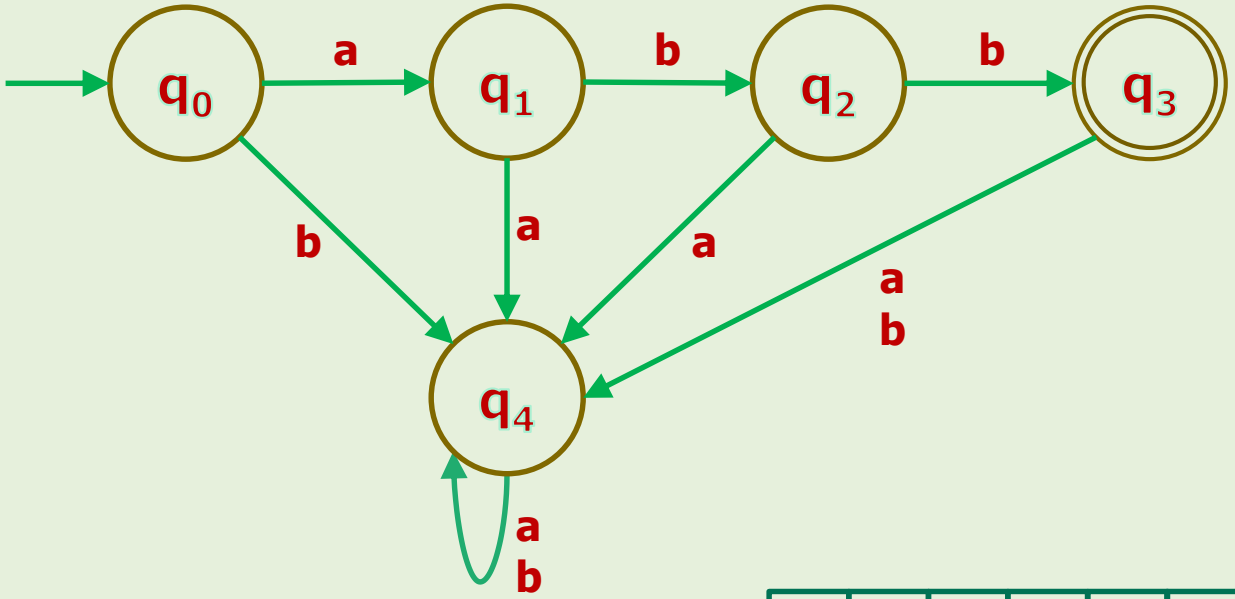
DFAs in Action

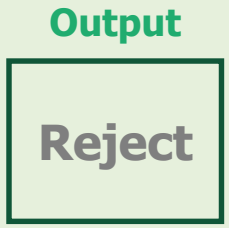
DFAs in Action

Example 5

The machine is off!

- $\Sigma = \{a , b\}$
- $w = abb$



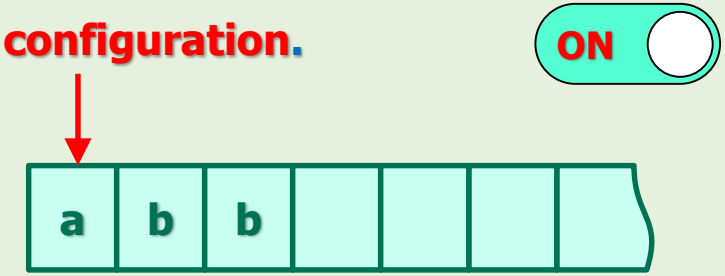


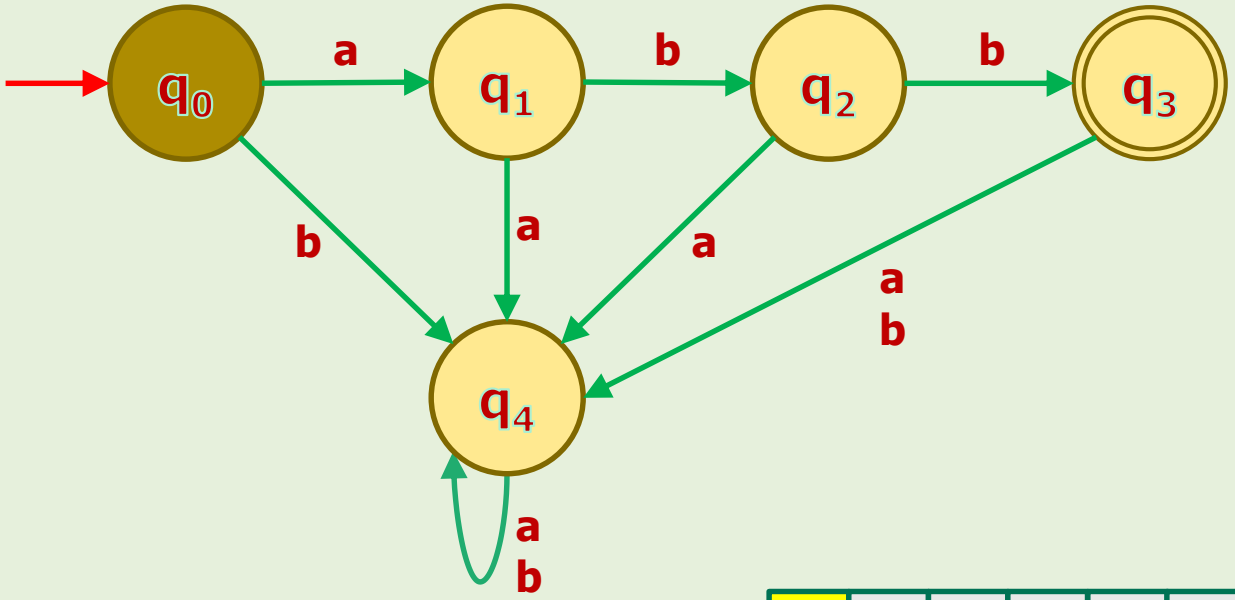
DFAs in Action

Example 5

- $\Sigma = \{a , b\}$
- $w = abb$

The machine is in starting configuration.





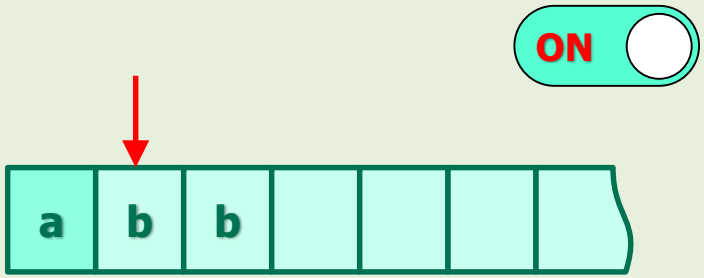
Output

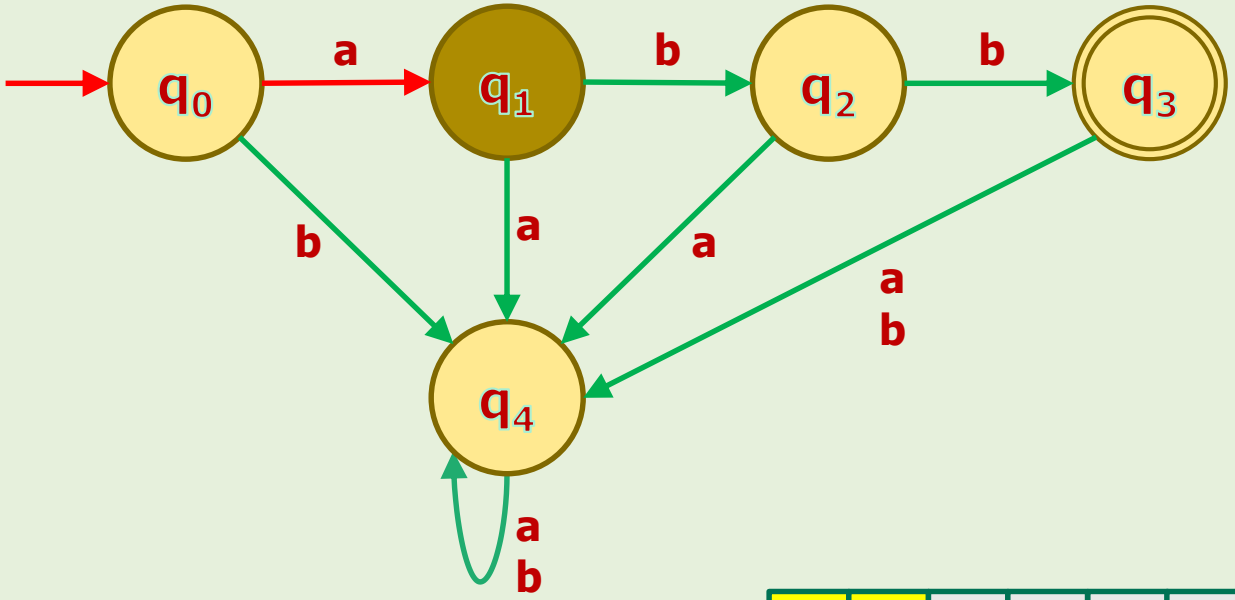
Reject

DFAs in Action

Example 5

- $\Sigma = \{a , b\}$
- $w = abb$





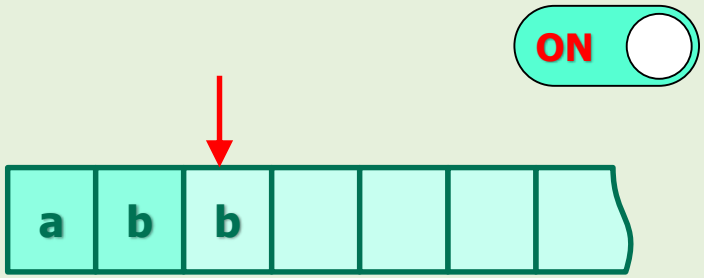
Output

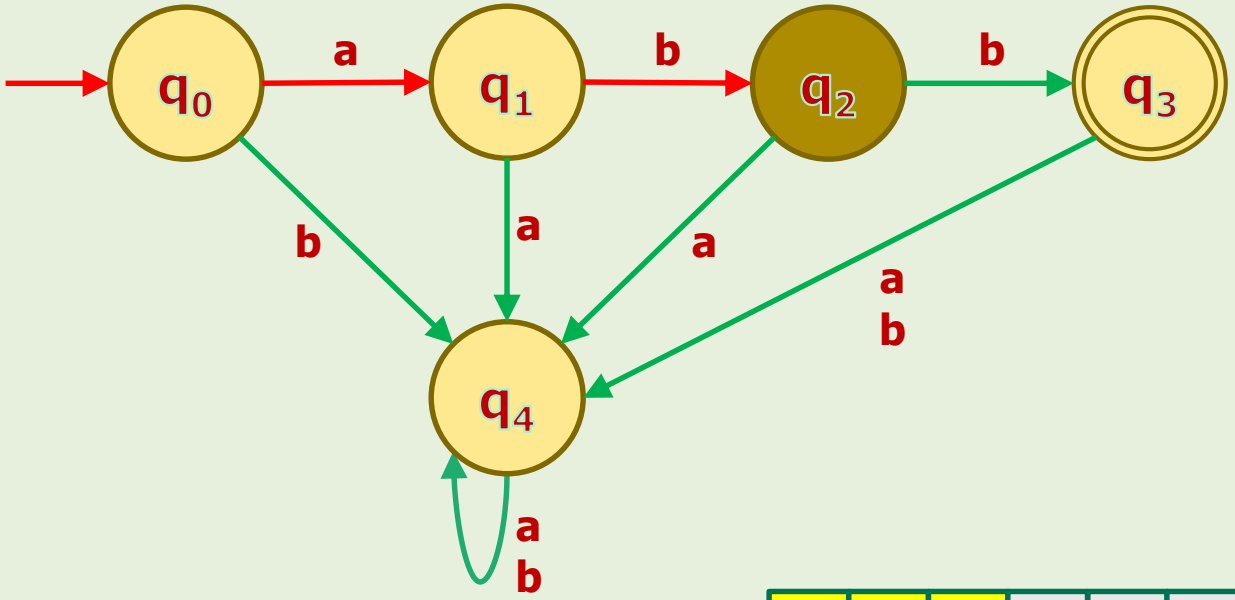
Reject

DFAs in Action

Example 5

- $\Sigma = \{a , b\}$
- $w = abb$





Output

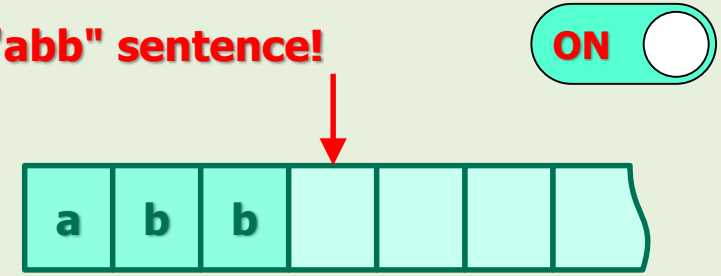
Reject

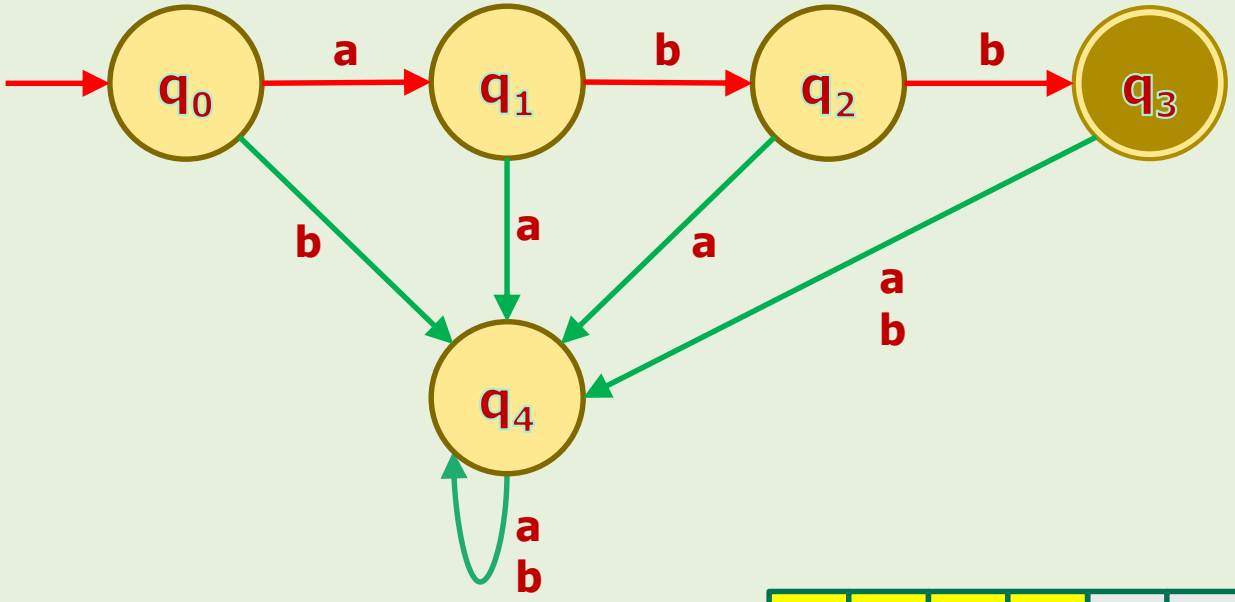
DFAs in Action

Example 5

- $\Sigma = \{a , b\}$
- $w = abb$

The machine understood "abb" sentence!





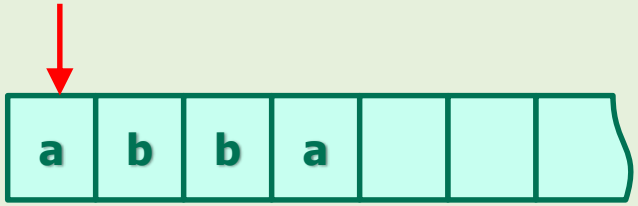
Output

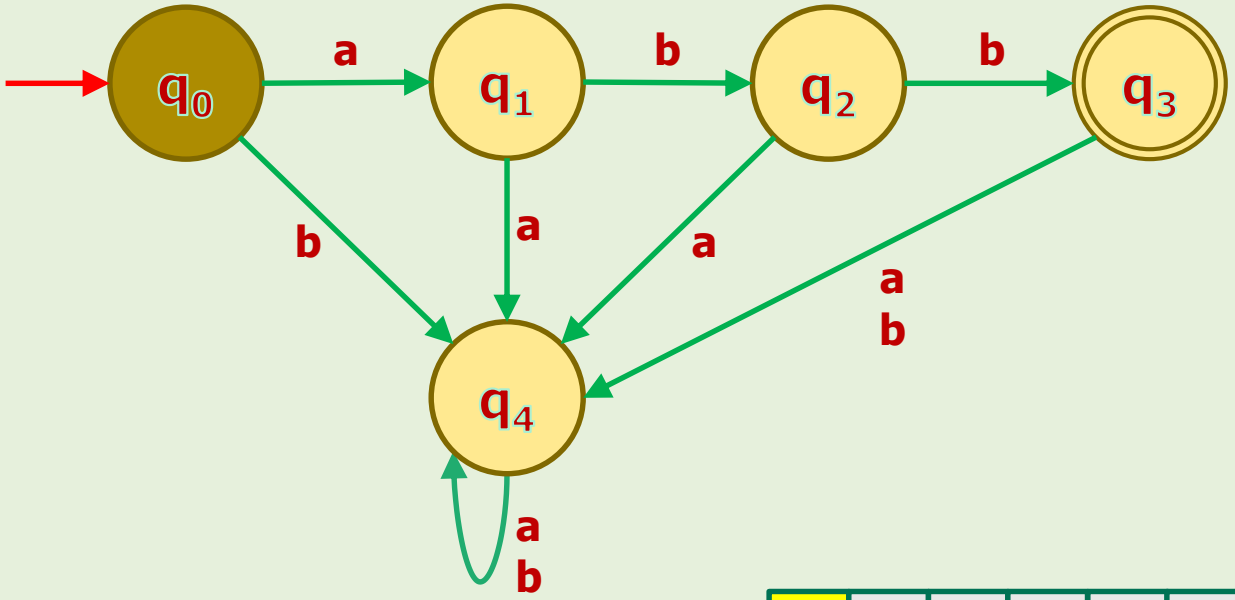
Accept

DFAs in Action

Example 6

- $\Sigma = \{a , b\}$
- w = abba





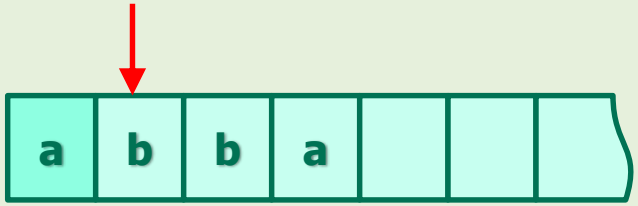
Output

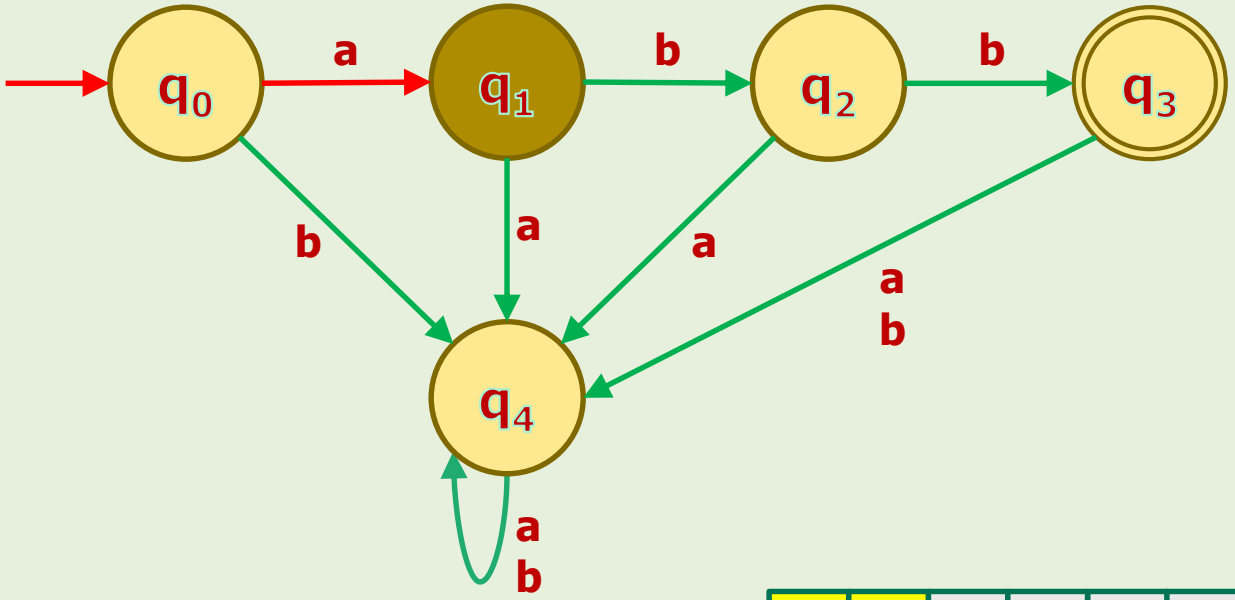
Reject

DFAs in Action

Example 6

- $\Sigma = \{a , b\}$
- w = abba





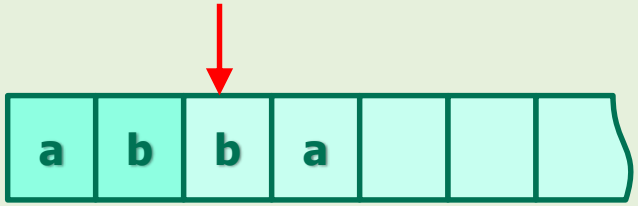
Output

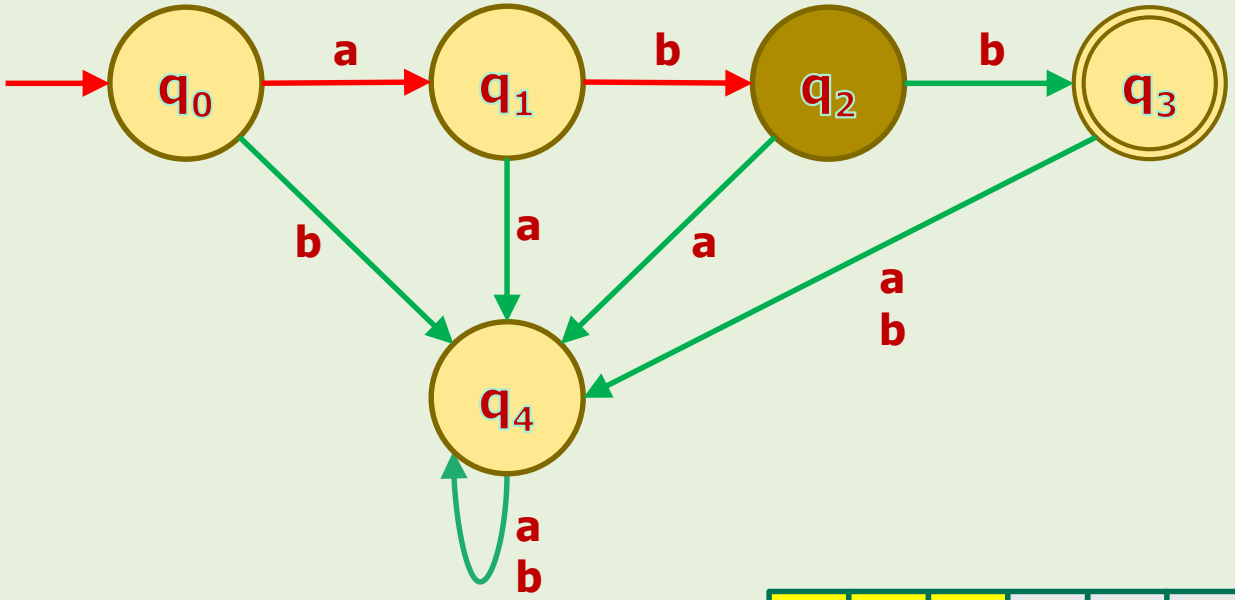
Reject

DFAs in Action

Example 6

- $\Sigma = \{a , b\}$
- w = abba





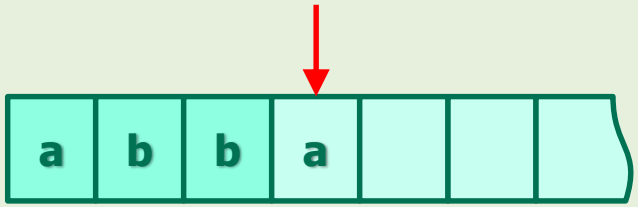
Output

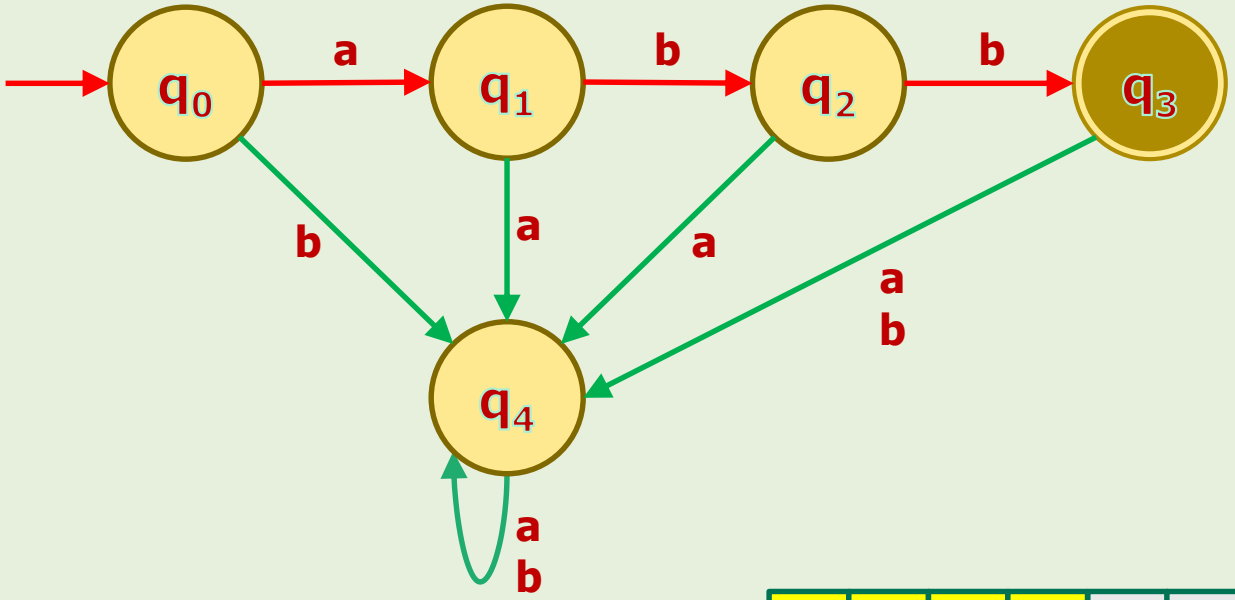
Reject

DFAs in Action

Example 6

- $\Sigma = \{a , b\}$
- w = abba





Output

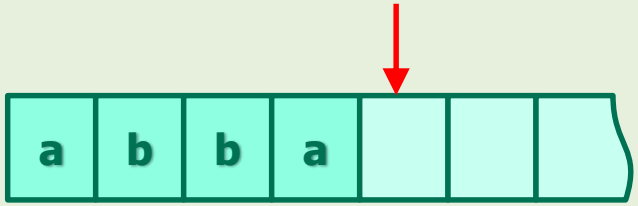
Reject

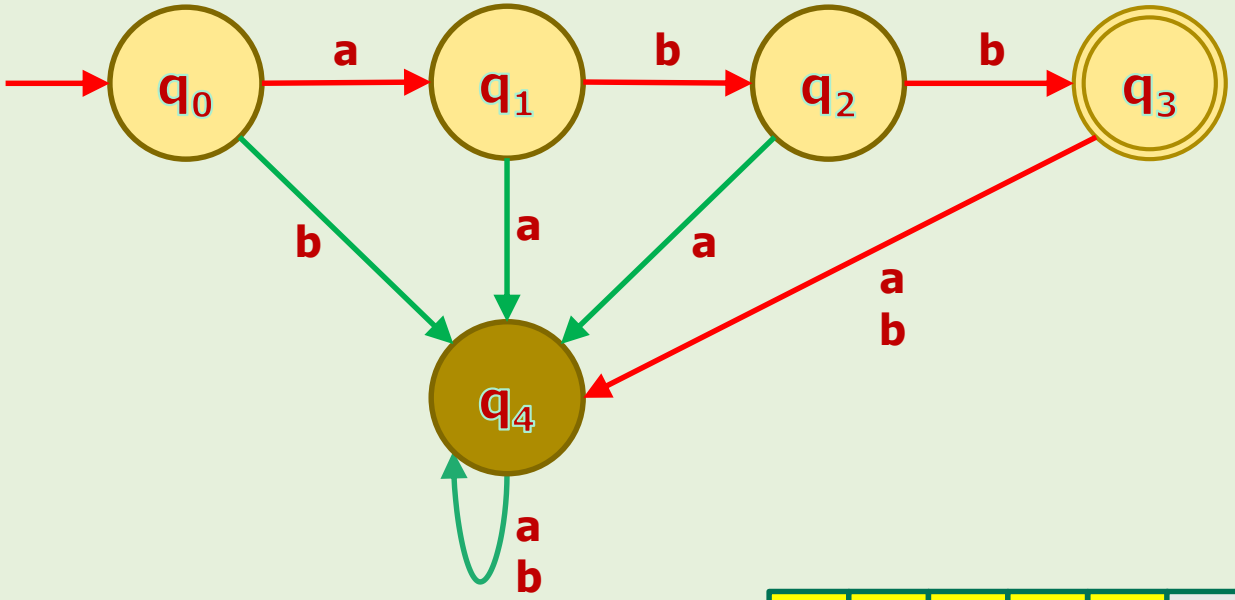
DFAs in Action

Example 6

- $\Sigma = \{a , b\}$
- w = abba

The machine did not understand "abba" sentence!





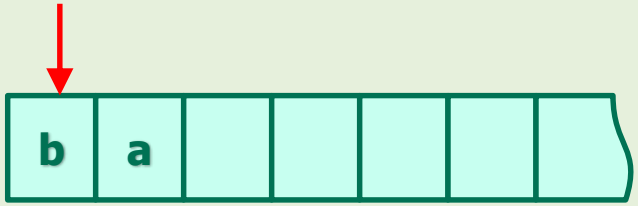
Output

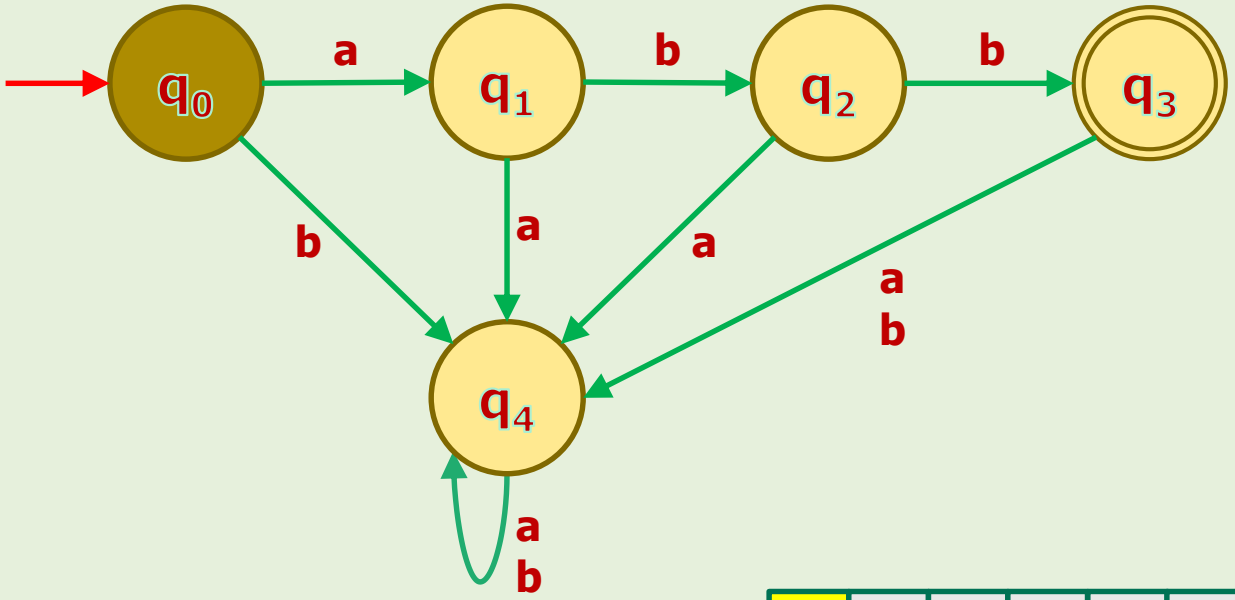
Reject

DFAs in Action

Example 7

- $\Sigma = \{a , b\}$
- $w = ba$





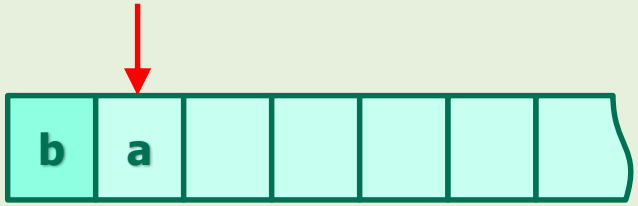
Output

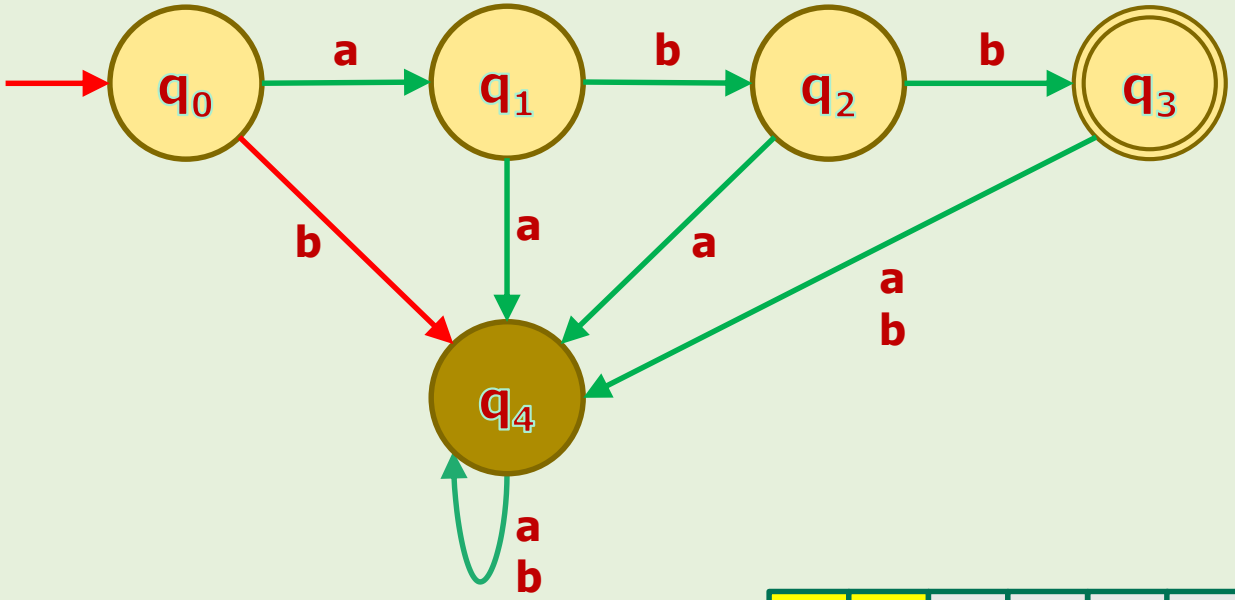
Reject

DFAs in Action

Example 7

- $\Sigma = \{a , b\}$
- $w = ba$





Output

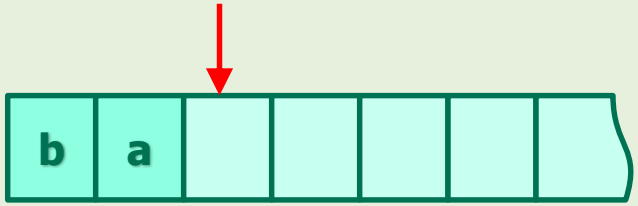
Reject

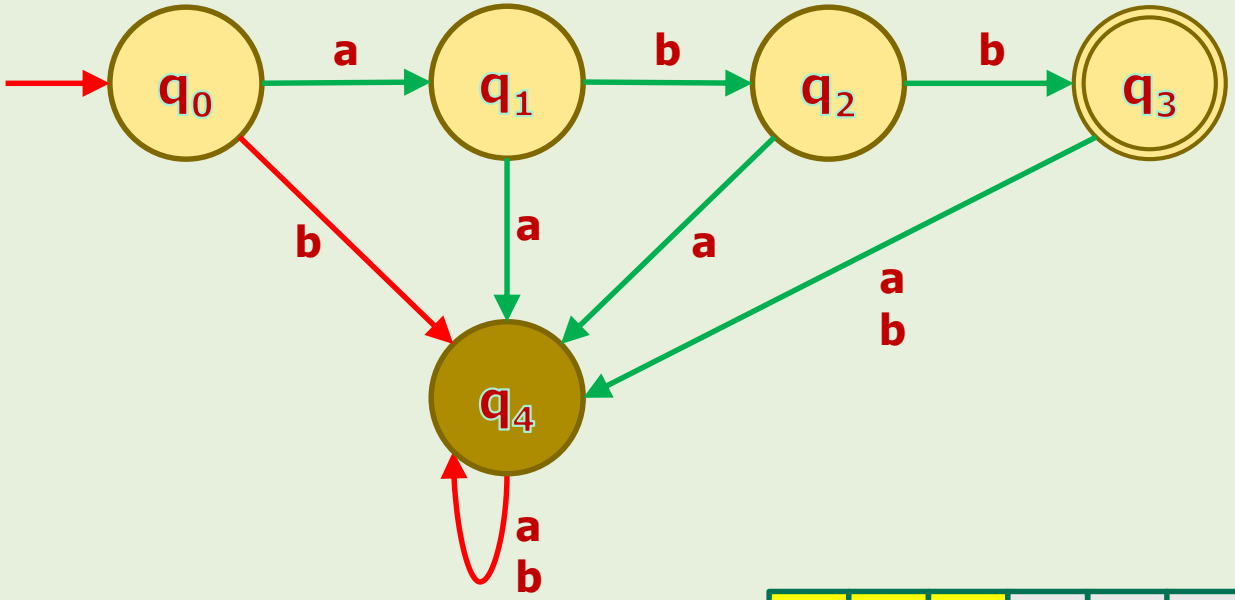
DFAs in Action

Example 7

- $\Sigma = \{a , b\}$
- $w = ba$

The machine did not understand "ba" sentence!





Output

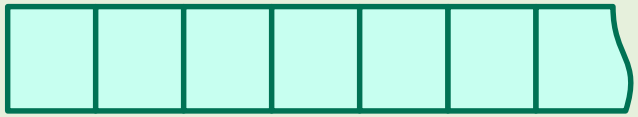
Reject

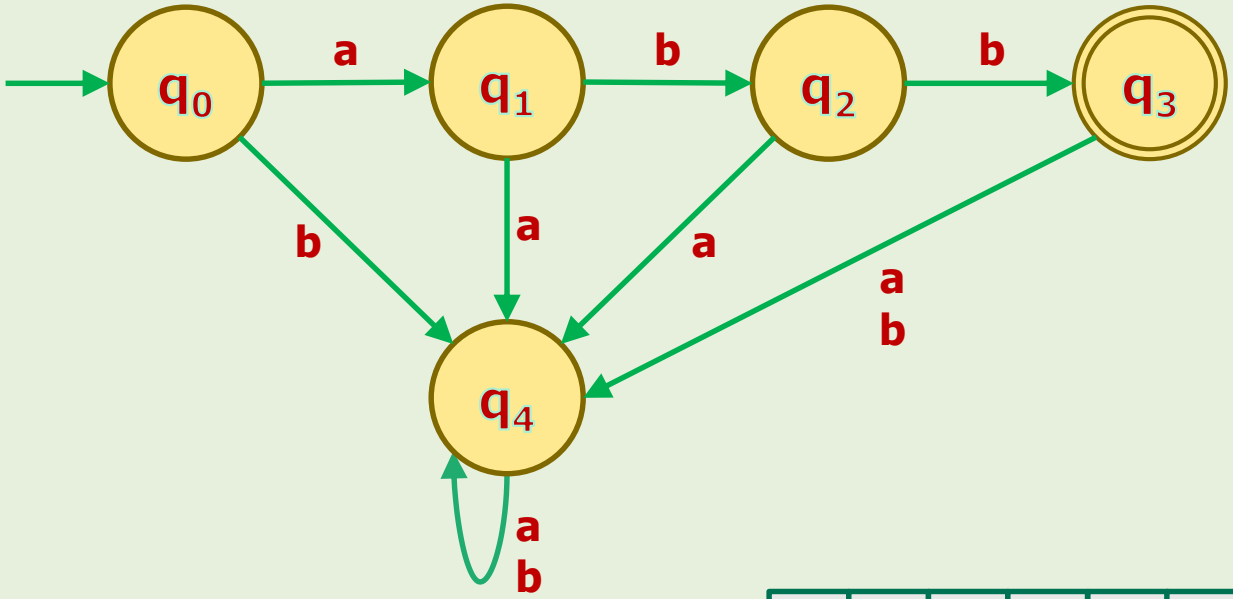
DFAs in Action

Example 8

The machine is off!

- $\Sigma = \{a , b\}$
- $w = \lambda$





Output

Reject

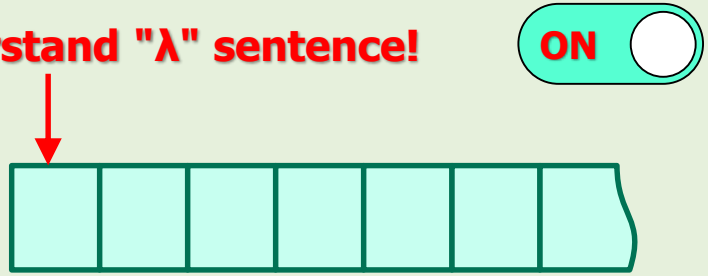
- How'd it work?

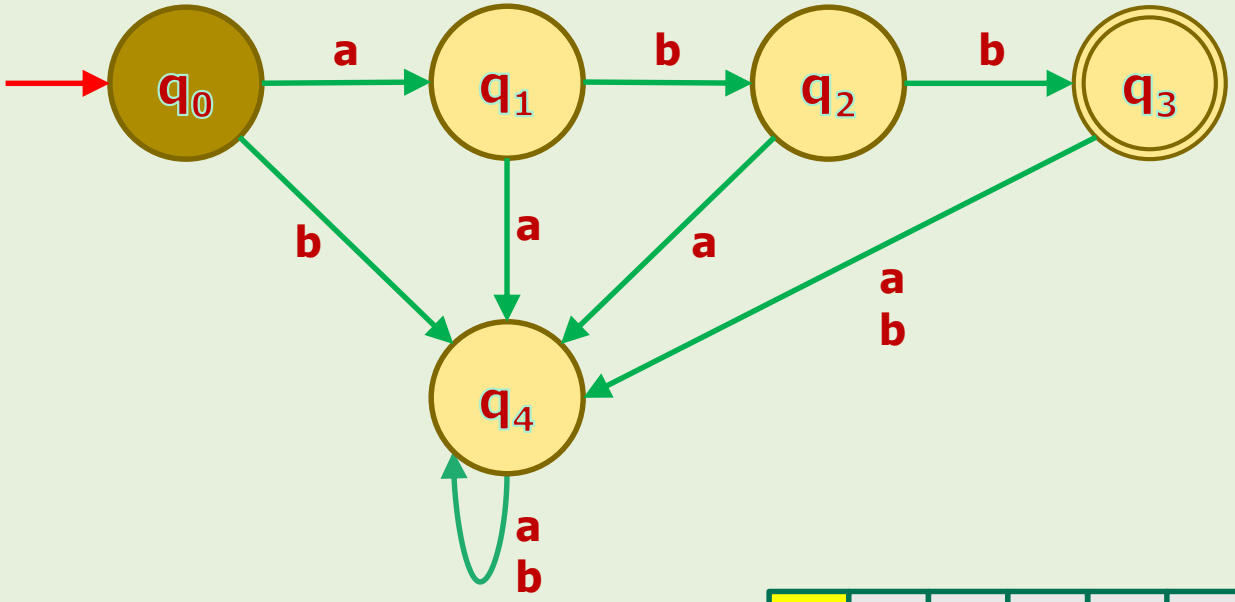
DFAs in Action

Example 8

- $\Sigma = \{a , b\}$
- $w = \lambda$

The machine did not understand "λ" sentence!





Output

Reject

4. How DFAs Accept/Reject Strings

Logical Representation of Accepting Strings

The string w is accepted (understood). ≡ **a**

IFF

The DFA halts. ≡ **h**

AND

All symbols of w are consumed. ≡ **c**

AND

The DFA is in an accepting (final) state. ≡ **f**

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

- Note that, for DFAs, h and c have the same value. (both T or both F)
  - But it might NOT be true for other classes of automata.

- So, we can simplify it for DFAs as: $(c \wedge f) \leftrightarrow a$

4. How DFAs Accept/Reject Strings

Logical Representation of Rejecting Strings

Recap: Math 42

$$p \leftrightarrow q \equiv \sim p \leftrightarrow \sim q$$

$$\sim (p \wedge q) \equiv (\sim p \vee \sim q)$$

- We know the following equivalencies:
- Let's apply these rules to the logical representation of accepting strings:

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

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

- Applying DeMorgan's rule on the left side:

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

- Now let's translate this new logical statement in English.

4. How DFAs Accept/Reject Strings

Logical Representation of Rejecting Strings

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

Translation

The string w is rejected (not understood). ≡ ~a

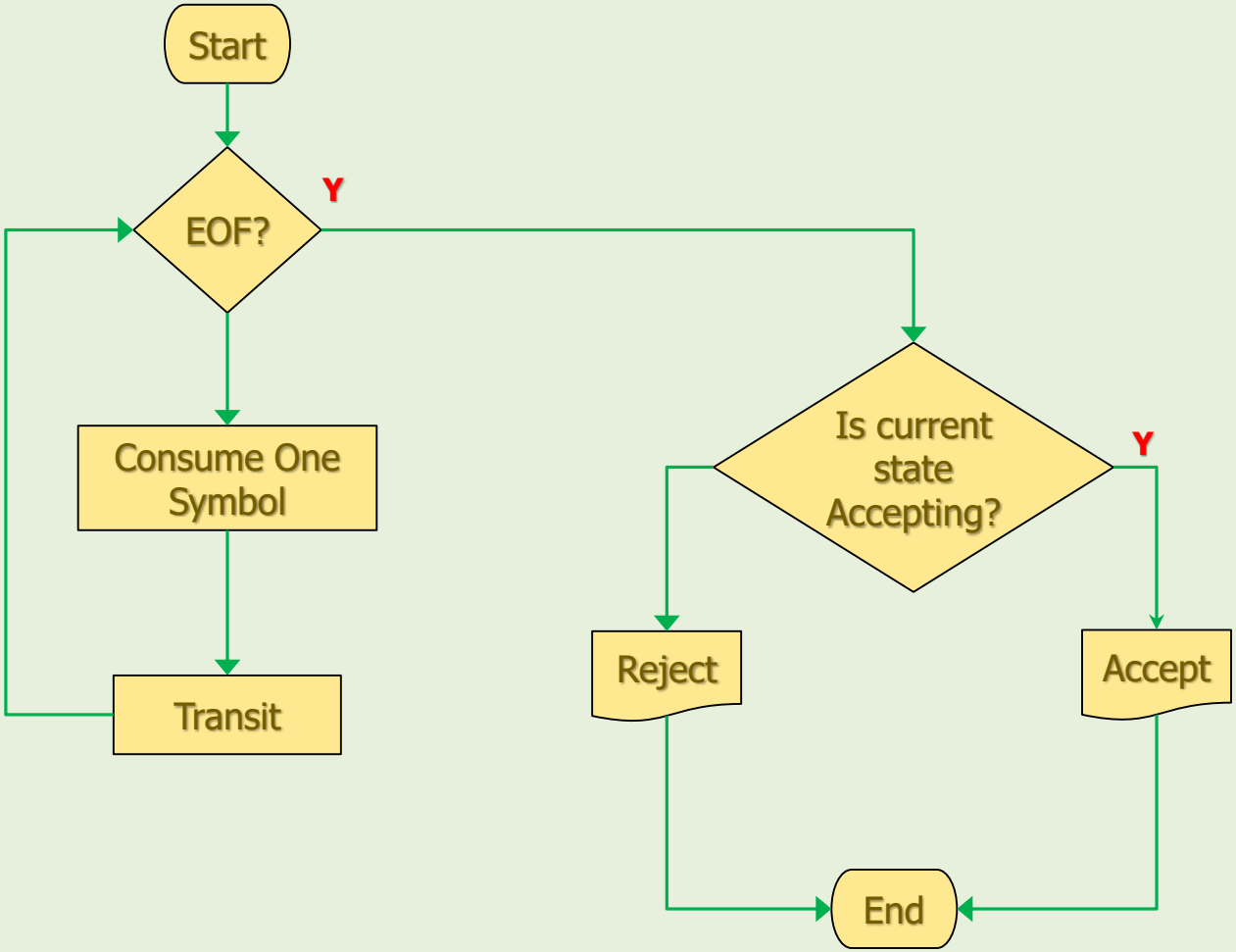
IFF

At least one symbol of w is NOT consumed. ≡ ~c

OR

The DFA is NOT in an accepting (final) state. ≡ ~f

DFAs Operation Flowchart



DFAs Operation Flowchart

1. Go to step 5 if EOF.
2. Consume a symbol.
3. Transit based on the logic of the current state.
4. Go to step 1
5. If the current state is "accepting state", change the output to "Accept".

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