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
Fall 2019

Agenda of Day 06

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

Announcement

- Rollcall does NOT have any impact on your score.
- I use it for recommendation, choosing graders, outstanding graduating senior awards, scholarships, job references, 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 your responsibility to check the Canvas at least once per day for any updates.

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)

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

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

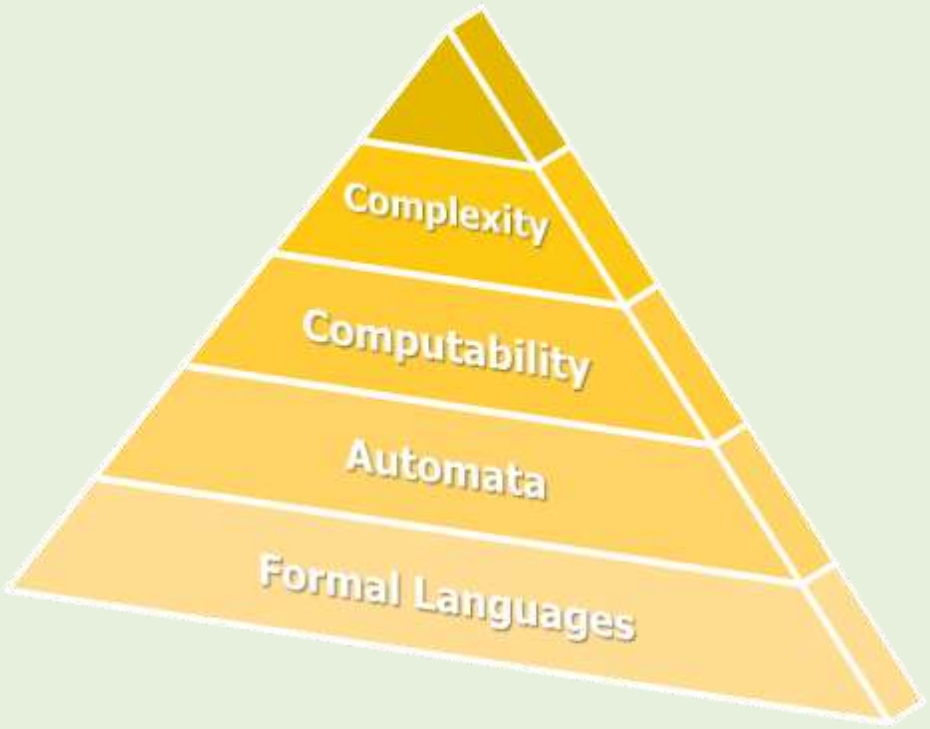
- Some surprising languages were introduced.
- Conclusion:
  All data in CS are strings.

Any question?

The Big Picture of the Course

- 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.
  - Of course, the meaning of "understanding" here is NOT the same meaning of understanding as in "AI" or "natural language processing (NLP)".

Automata

- These machines are called "automata" (plural of automaton).
  - This is a scientific (latin) name for computation machines.

Definition

- Automaton is a mathematical model of a computation 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 $\overline{L}$

- We design a machine M for a particular language L.
- When we input the strings of L, M understands (aka Accept) it.

- And when we input the strings of $\overline{L}$, M does NOT understand (aka Reject) it.

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 following template.

Template for Introducing a New Class of Automata

- To construct a new class of automata, we'll follow the following steps:

1. Why do we need this new class? (Justification)
2. Name of this new class
3. Building blocks of this new class
4. How do they work?
   - 4.1. What is the starting configuration?
   - 4.2. What would happen during a timeframe?
   - 4.3. When would the machine halt?
   - 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 this new class of machines?

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

- The general reason why we introduce this class is:

  "These machines are supposed to understand formal languages."

2. Name of this new class

- We name this class of automata as:

  Deterministic Finite Automata (DFA)

  (aka Finite State Machines)

Where is this name coming from?

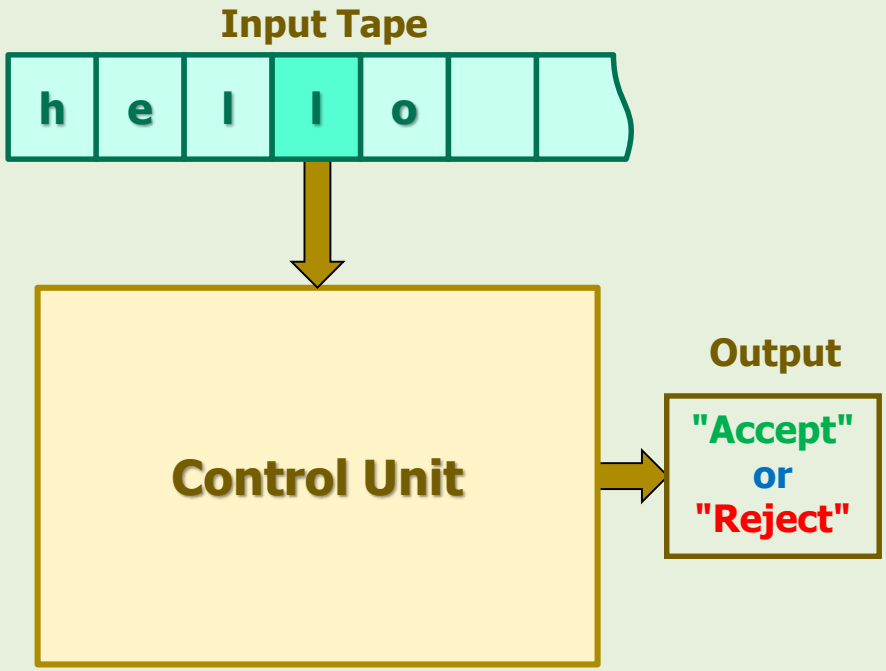
- We have already know that "automata" means machines.
- Also we know the meaning of "finite" but
  we don't know what part of the machine is finite?
- Also we don't know the meaning of "deterministic".

- Both of these will be explained later.

3. DFAs Building Blocks

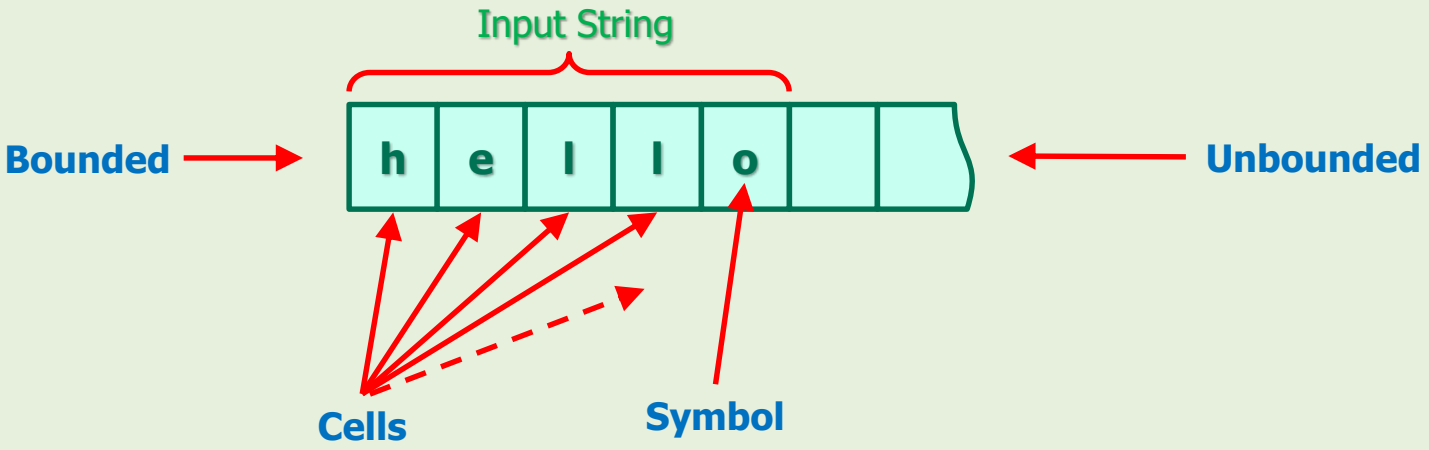
3. DFAs Building Blocks

- DFAs have 3 main blocks:
  1. Input Tape
  2. Control unit
  3. Output



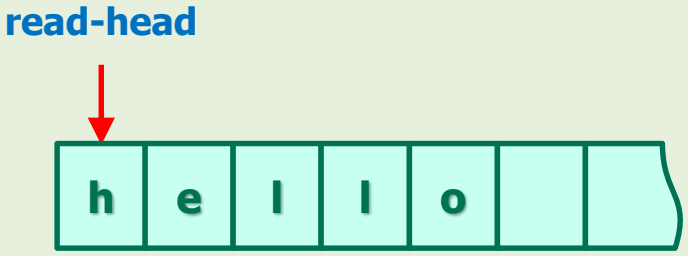
- Let's see each block in detail.

3.1. Input Tape: Structure



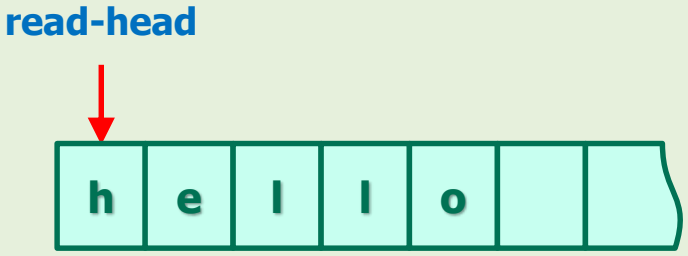
- The tape is bounded from the left side and ... unbounded from the right.
- It is divided into "cells".
- Each cell can hold one "symbol".
- The input data is a string written on the tape from the left-most cell.

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

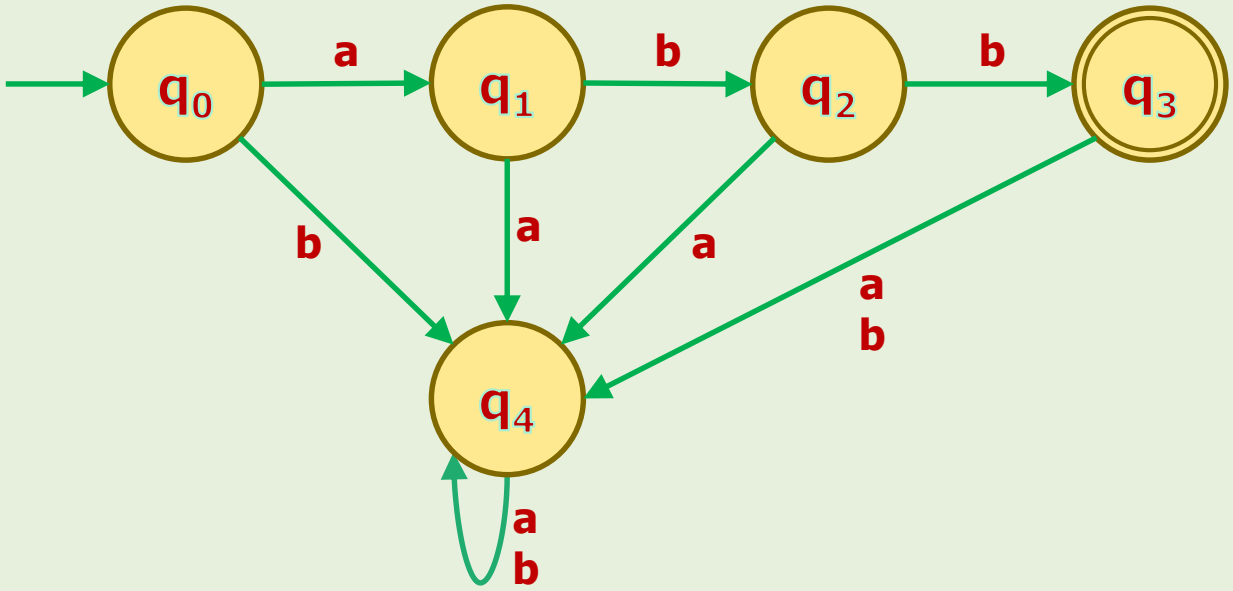


1. We use the words reading, and scanning interchangeably.
2. DFAs can "read" the input string but cannot change it. (Read-Only Tape)
3. The head only moves from left to right.
   - So, during the machine's operations, when the head moves one symbol to the right, there is no way to move it back. (Irreversible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

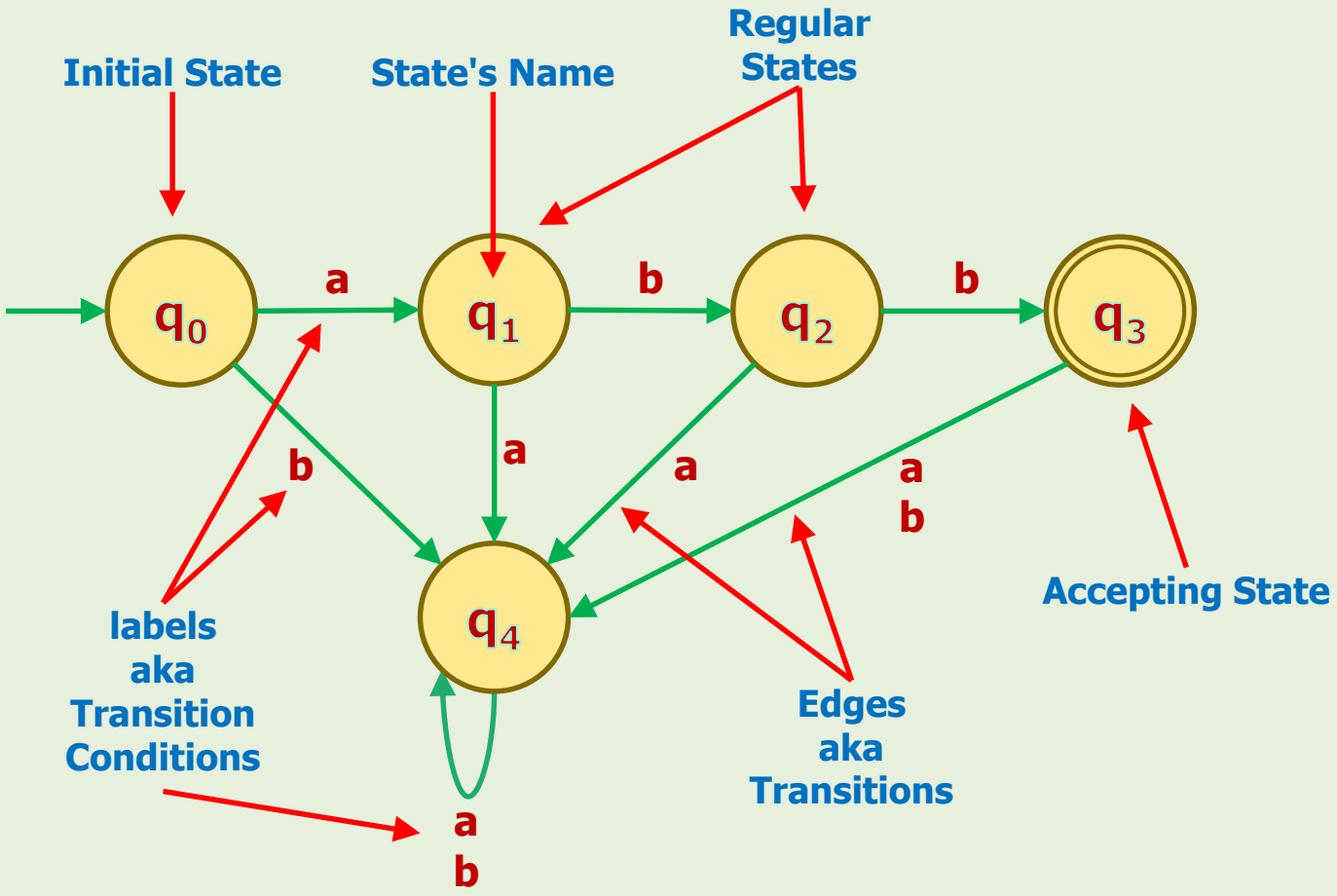
3.2. Control Unit: Structure

- The control unit is the brain (CPU) of every automaton like 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



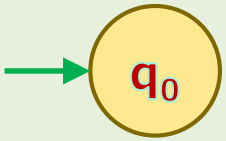
3.2. Control Unit: Transition Graph Ingredients



3.2. Control Unit: Transition Graph Ingredients

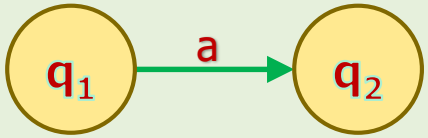
- As we mentioned before, vertices of the graphs are also known as "nodes" and "states".
- From now on, we call them "state".
  - The label q_1 is the name of the state.

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

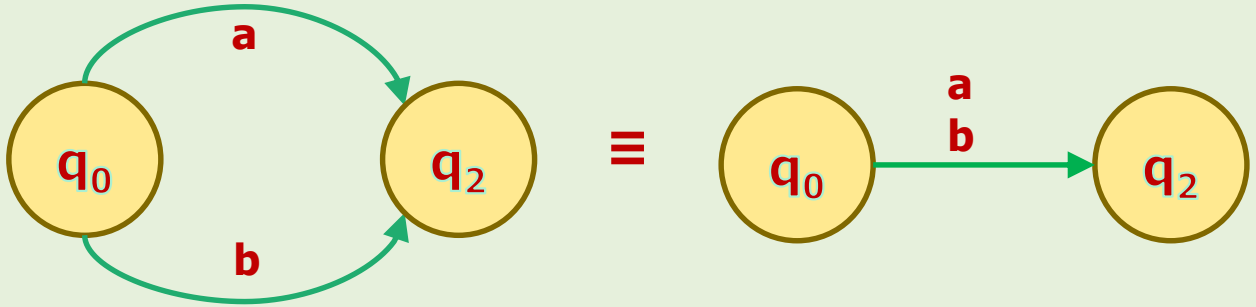


- An "accepting state" (aka "final state") is shown by double circle.

- Transition graph can have zero or more final states.

3.2. Control Unit: Notes

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



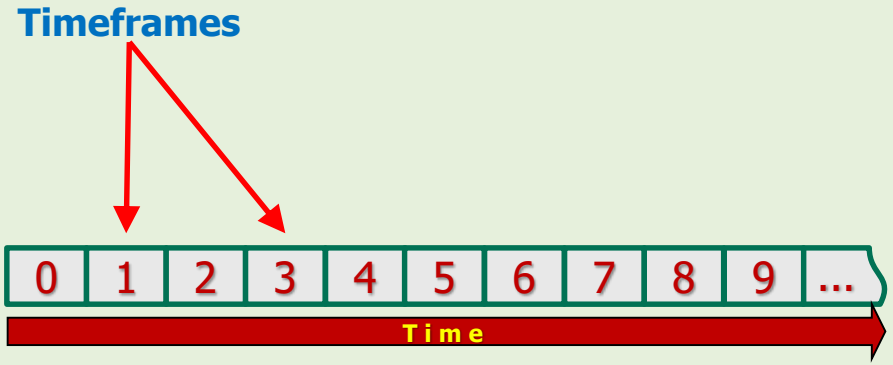
- Note that $\begin{matrix} a \\ b \end{matrix}$ means "a or b".
- In some books, you might see "a , b" that is confusing.
- Because comma usually means "AND".

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

3.2. Control Unit: Synchronization of Operations

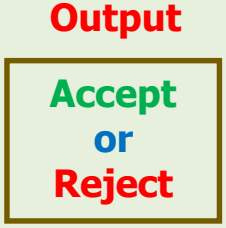
- For synchronization of operations,
  the control unit has a clock that produces discrete signals (ticks).
- We call each signal a "timeframe" (aka timestep).
  - The frequency of the clock does not matter in this course.

- In my lecture notes, I use the following figure
  to show a clock and its timeframes.



3.3. Output

- Output of DFAs have only two messages:
  - Accept (aka: understood, recognized)
  - Reject (aka: not understood, not recognized)



- 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".

4. How do DFAs Work?

4. How do DFAs Work?

- To understand how DFAs work, we should 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?

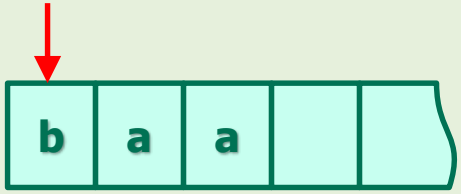
- We'll answer the first two questions right now and postpone the last two because we need some practices first.

4.1. DFAs Starting Configuration

Clock

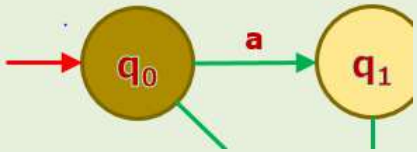
- The clock is set at timeframe 0.

Input Tape



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

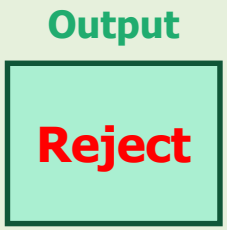
Control Unit



- The control unit is set to initial state.

Output

- The output shows "Reject".



What is Configuration?

Definition

- DFAs' configuration is the combination of the following data:

1. Input string + Position of the read-head
2. Current state of the transition graph
3. Timeframe number on the clock. (we'll remove this later.)

- In other words, configuration is a snapshot of the machine's status in one timeframe.
- If you have the configuration of a machine at timeframe n, you can continue its operation onward.

4.2. 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.

4.2. What Happens During a Timeframe

Transition Examples

Transition Examples

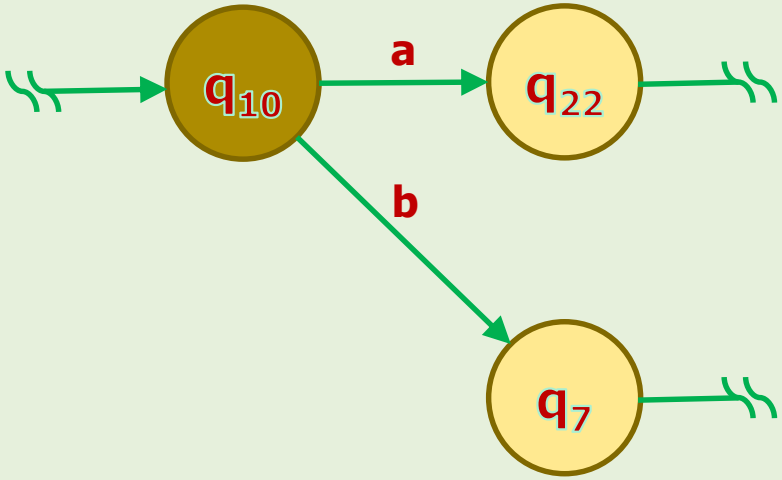
- The next examples will show:
  - a partial transition graph
  - an input tape
  - a clock

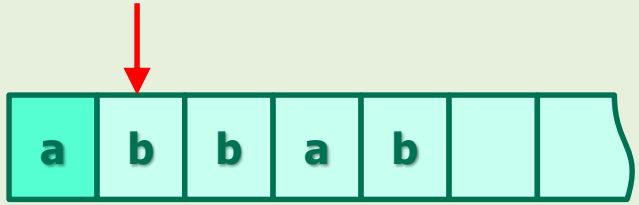
- We assume that the machine is in the middle of its operation at timeframe n.
- The question is: in what configuration would the machine be at timeframe n+1?

Transition Examples

Example 2

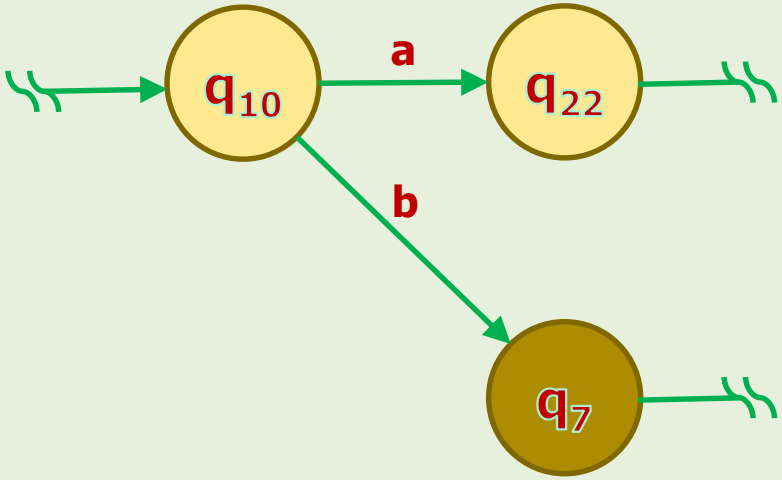
- What would be the DFA's configuration at the end of the next timeframe?

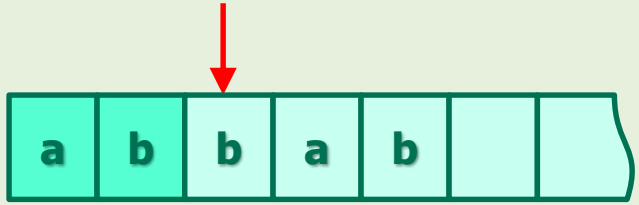




Transition Examples

Example 2 (cont'd)

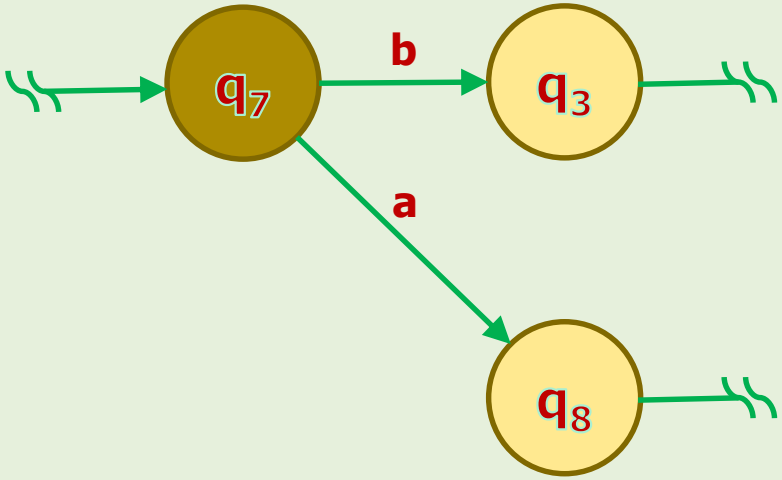


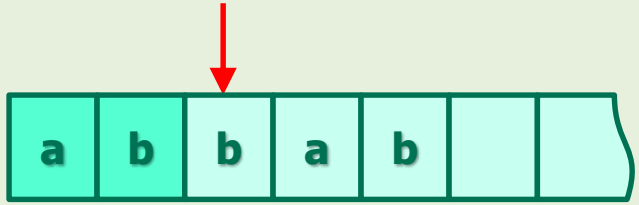


Transition Examples

Example 3

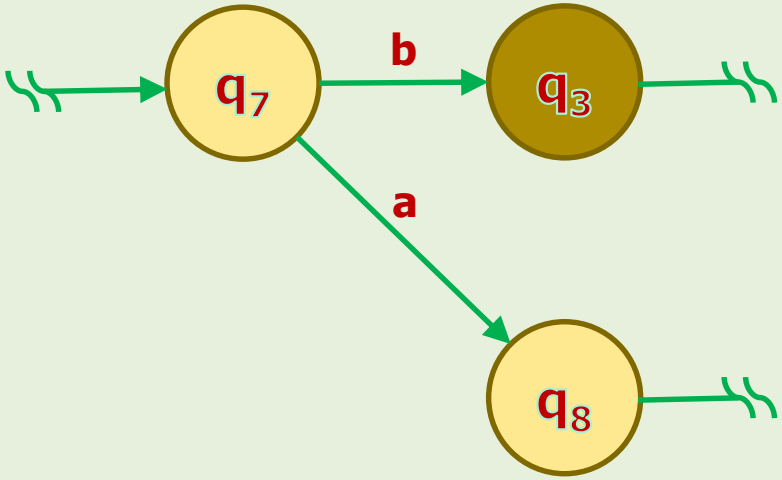
- What would be the DFA's configuration at the end of the next timeframe?

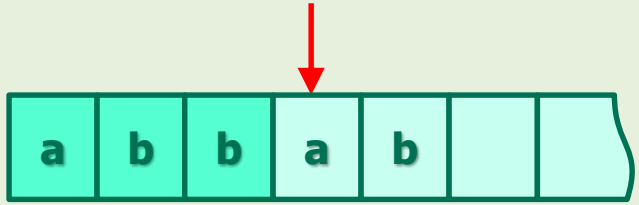




Transition Examples

Example 3 (cont'd)

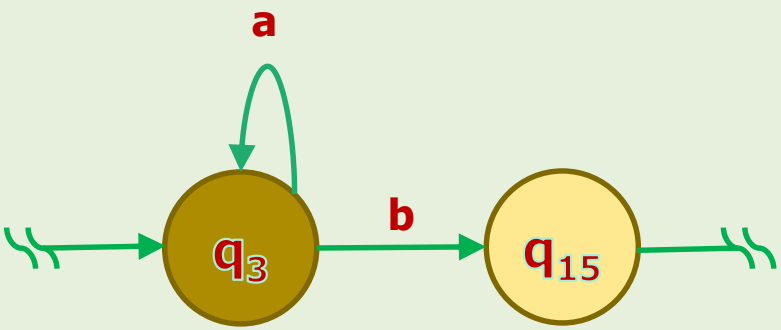


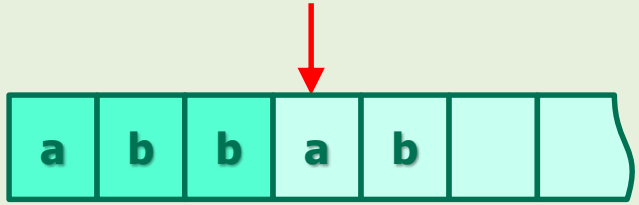


Transition Examples

Example 4

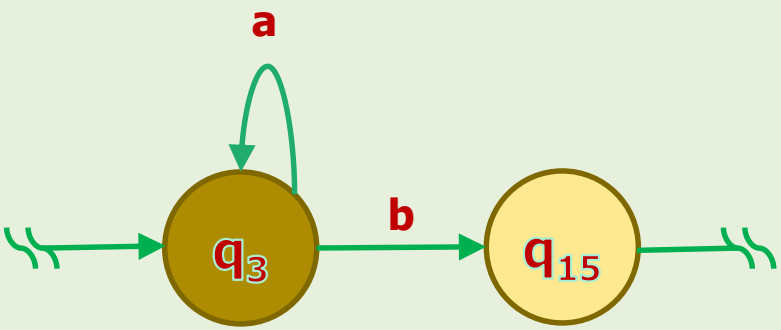
- What would be the DFA's configuration at the end of the next timeframe?

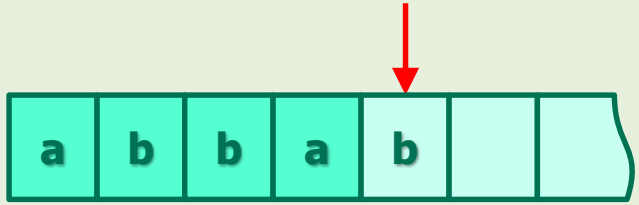




Transition Examples

Example 4 (cont'd)





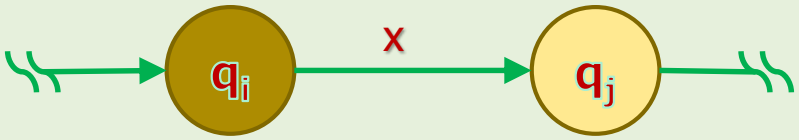
4.2. What Happens During a Timeframe

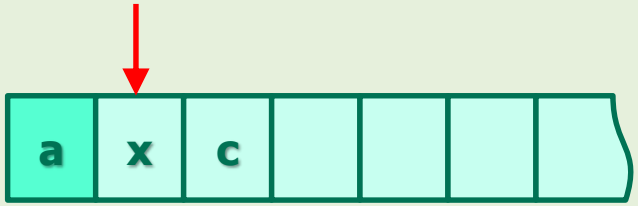
Summary of Transition

1. The symbol at which the read-head is pointing is read and is sent to the control unit.
2. The control unit makes its move based on the "logic of the transition". (See next slide!)
3. The control unit commands the tape to move the read-head one cell to the right.
   - Recall that:
     reading a symbol + moving the head = consuming a symbol

- Now let's see what the "logic of the transition" is?

DFAs' Logic of Transitions



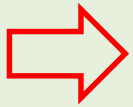


If (Condition)

in q_i

AND

the input symbol is 'x'



Then (Operation)

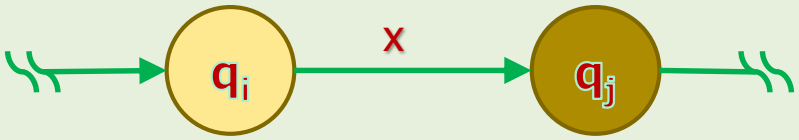
consume 'x'

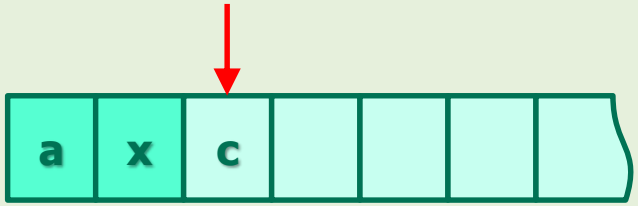
AND

transit to q_j

How does the machine look like after this transition?

DFAs' Logic of Transitions





After the previous transition ...

- You might ask: what if the input is not 'x'?
- Good question! We'll get back to this question later.

4. How do 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 timeframe.

5. DFAs in Action

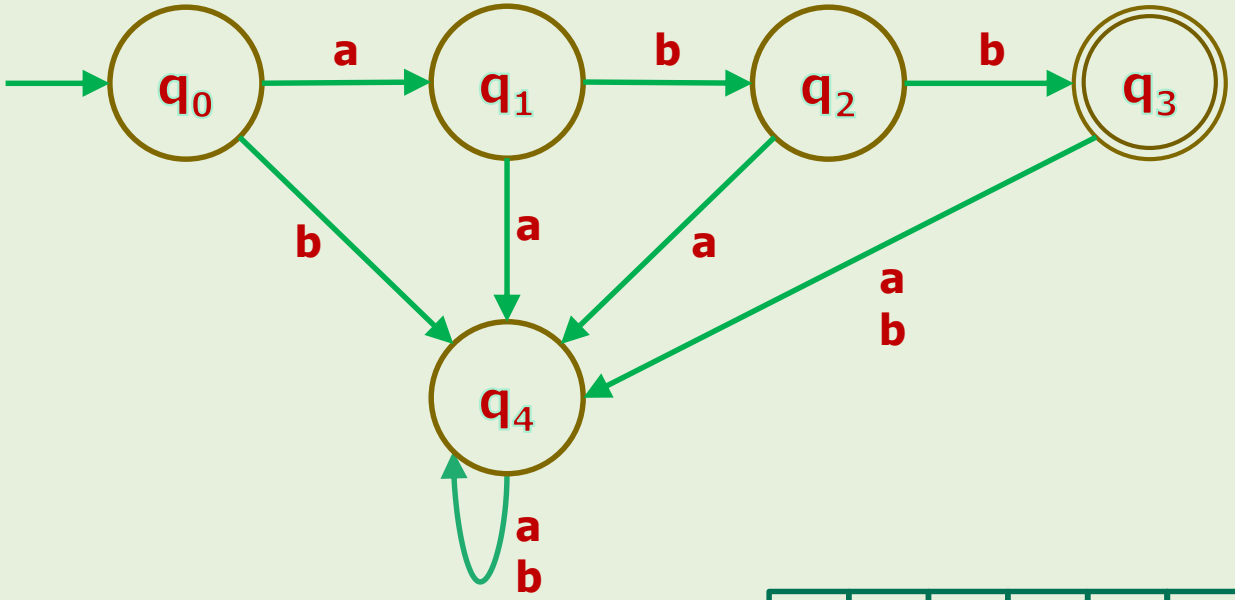
5. DFAs in Action

Example 5

The machine is off!

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

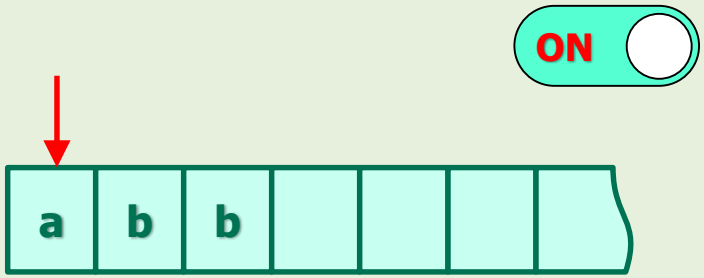
$w = abb$



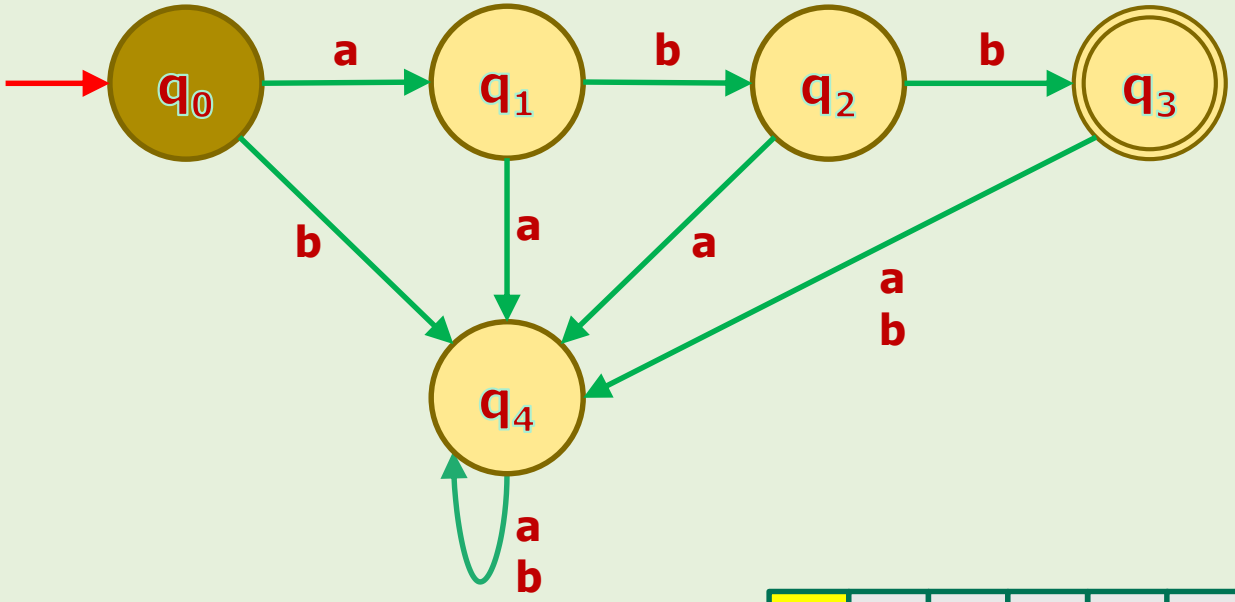
Output

5. DFAs in Action

Example 5



The machine is in starting configuration.

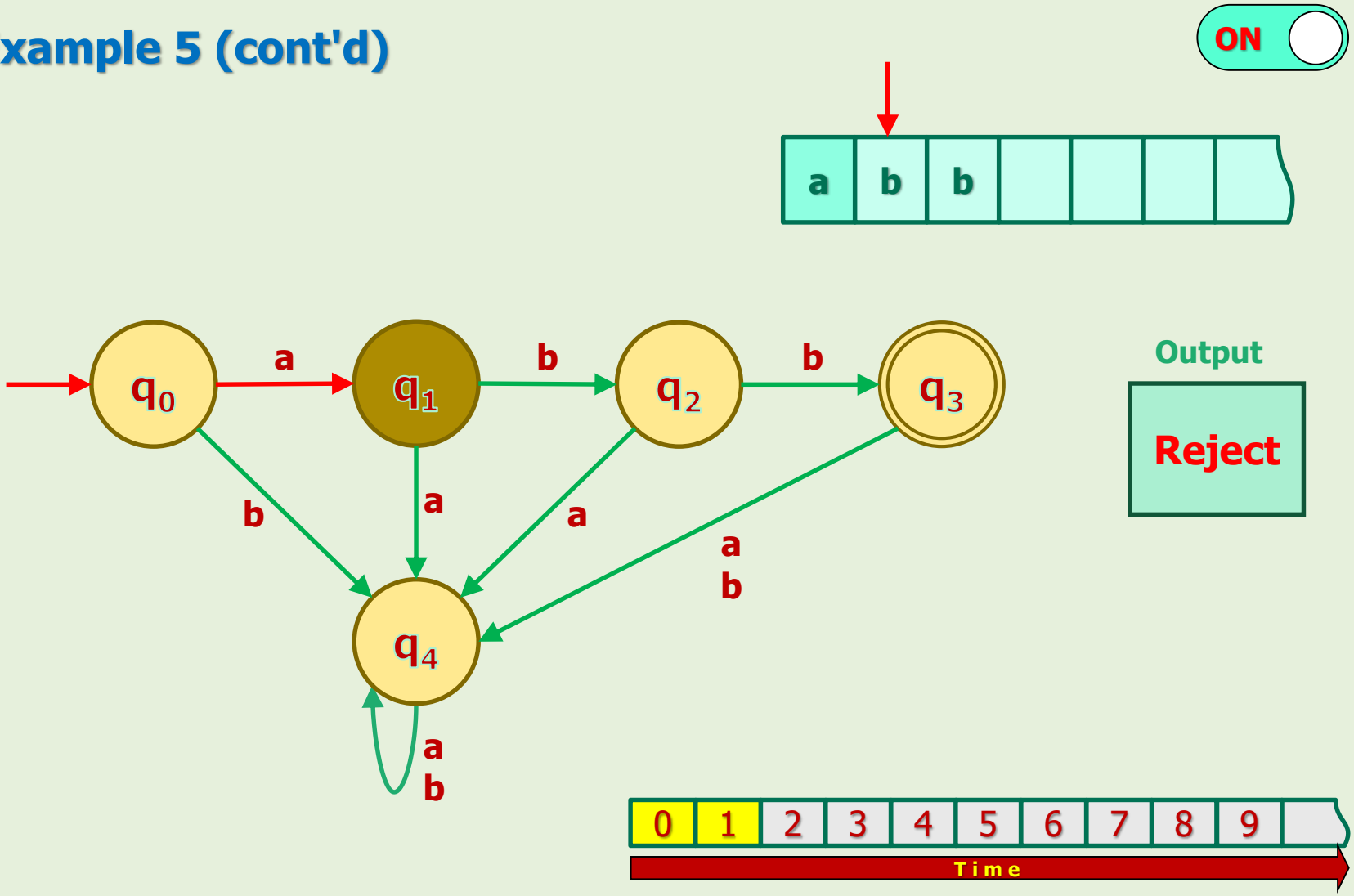


Output

Reject

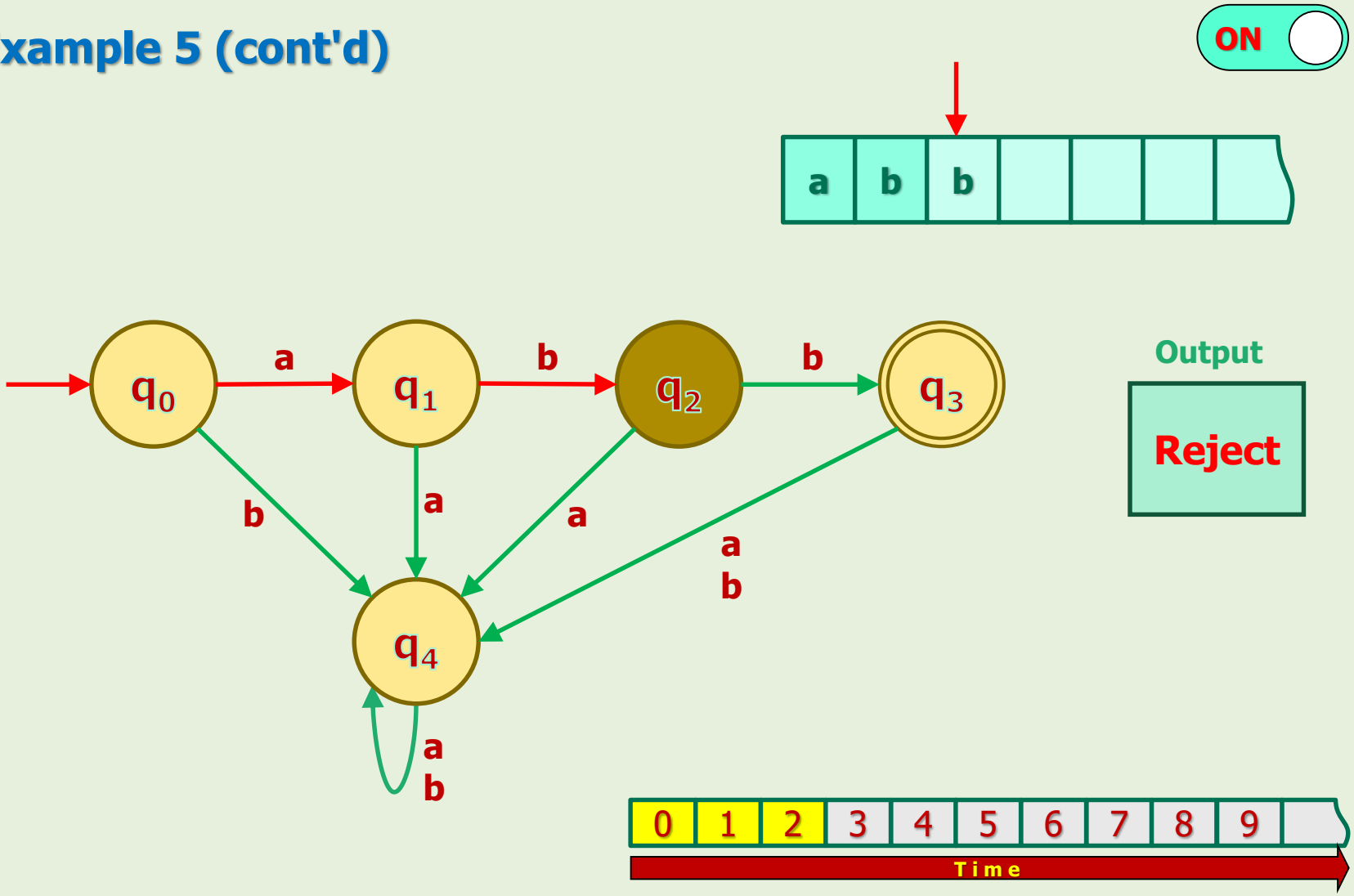
5. DFAs in Action

Example 5 (cont'd)



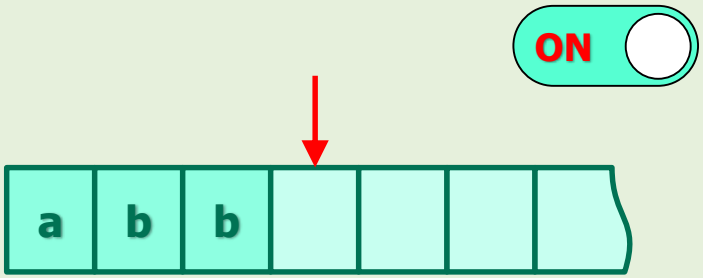
5. DFAs in Action

Example 5 (cont'd)

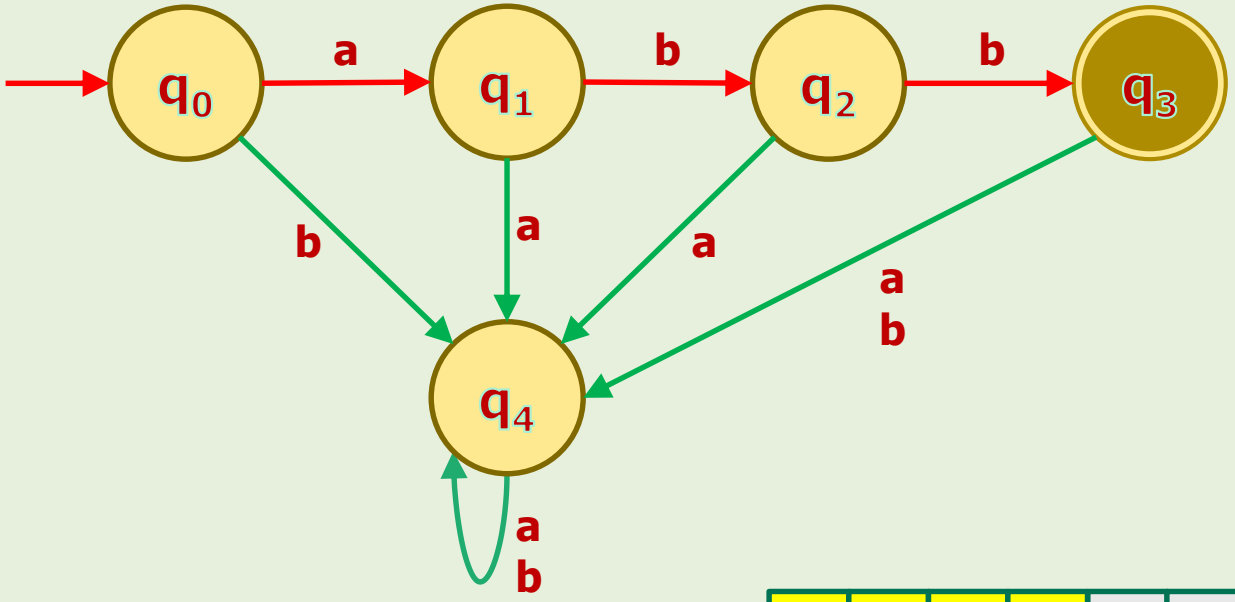


5. DFAs in Action

Example 5 (cont'd)



The machine understood "abb"!



Output

Accept

4.3. When DFAs Halt

- DFAs halt iff all input symbols are consumed.

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

Recap: Biconditional

p ↔ q ≡ if and only if

Logical Representation of Halting

DFAs halt. ≡ **h**

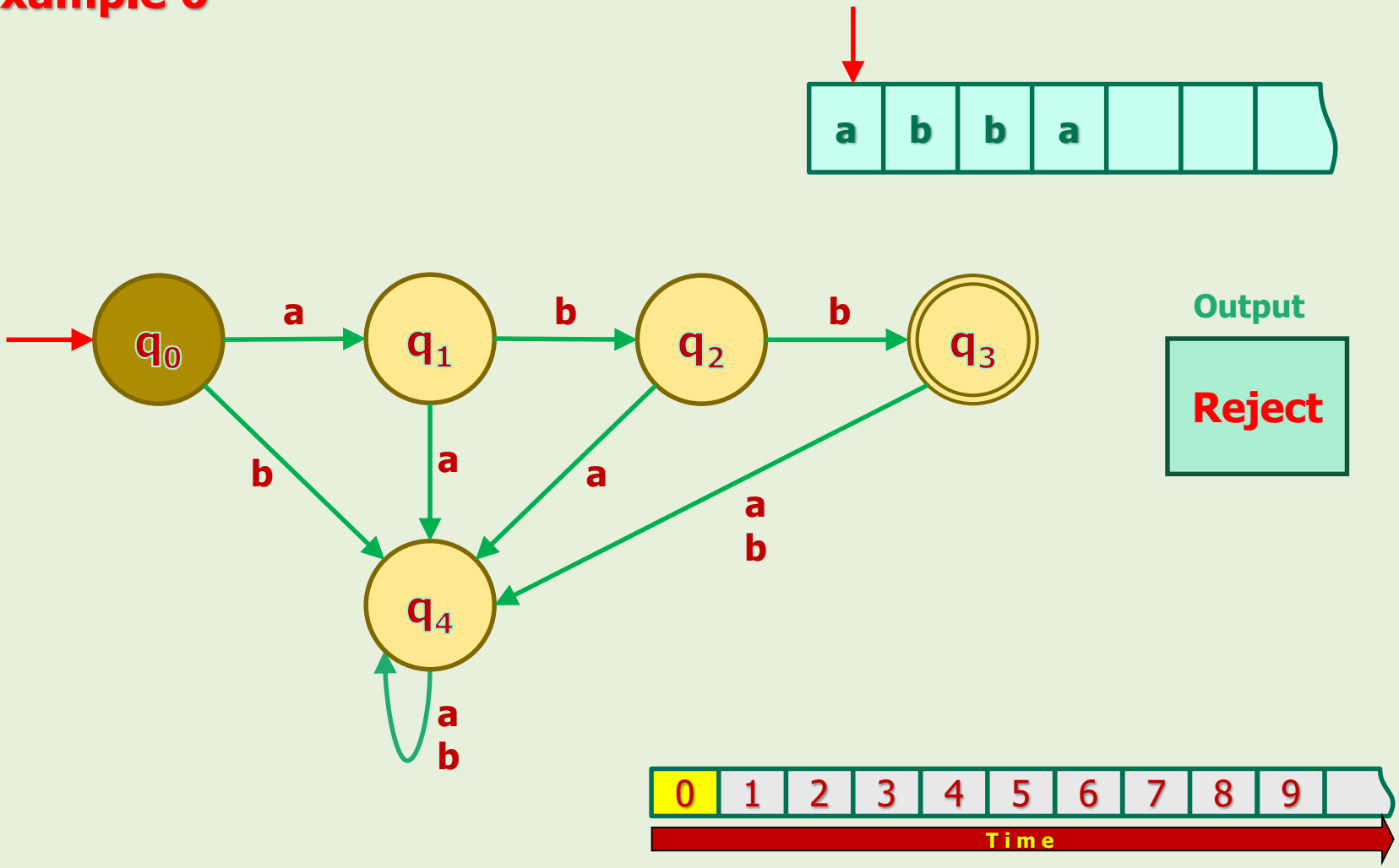
IFF

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

} **c ↔ h**

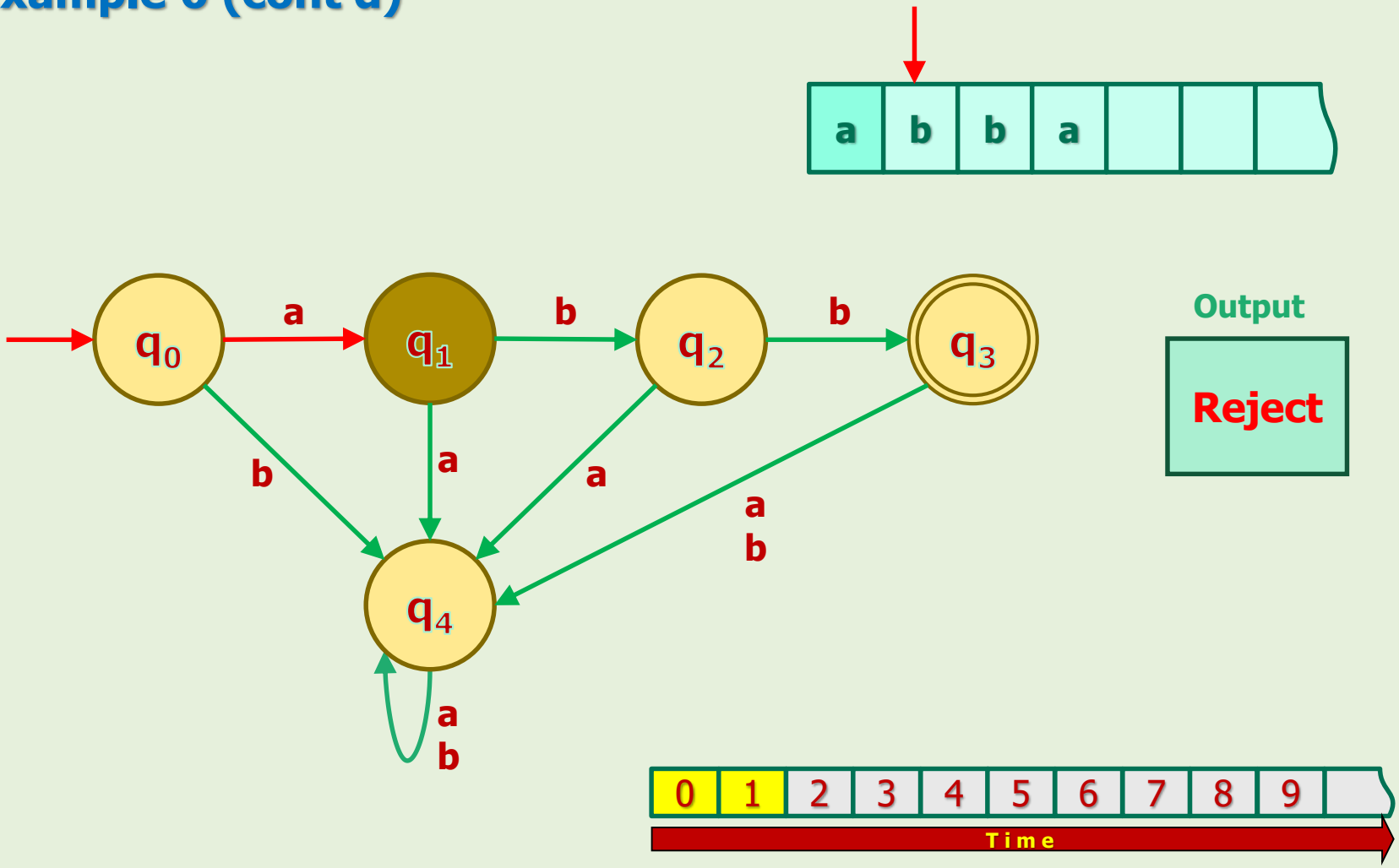
5. DFAs in Action

Example 6



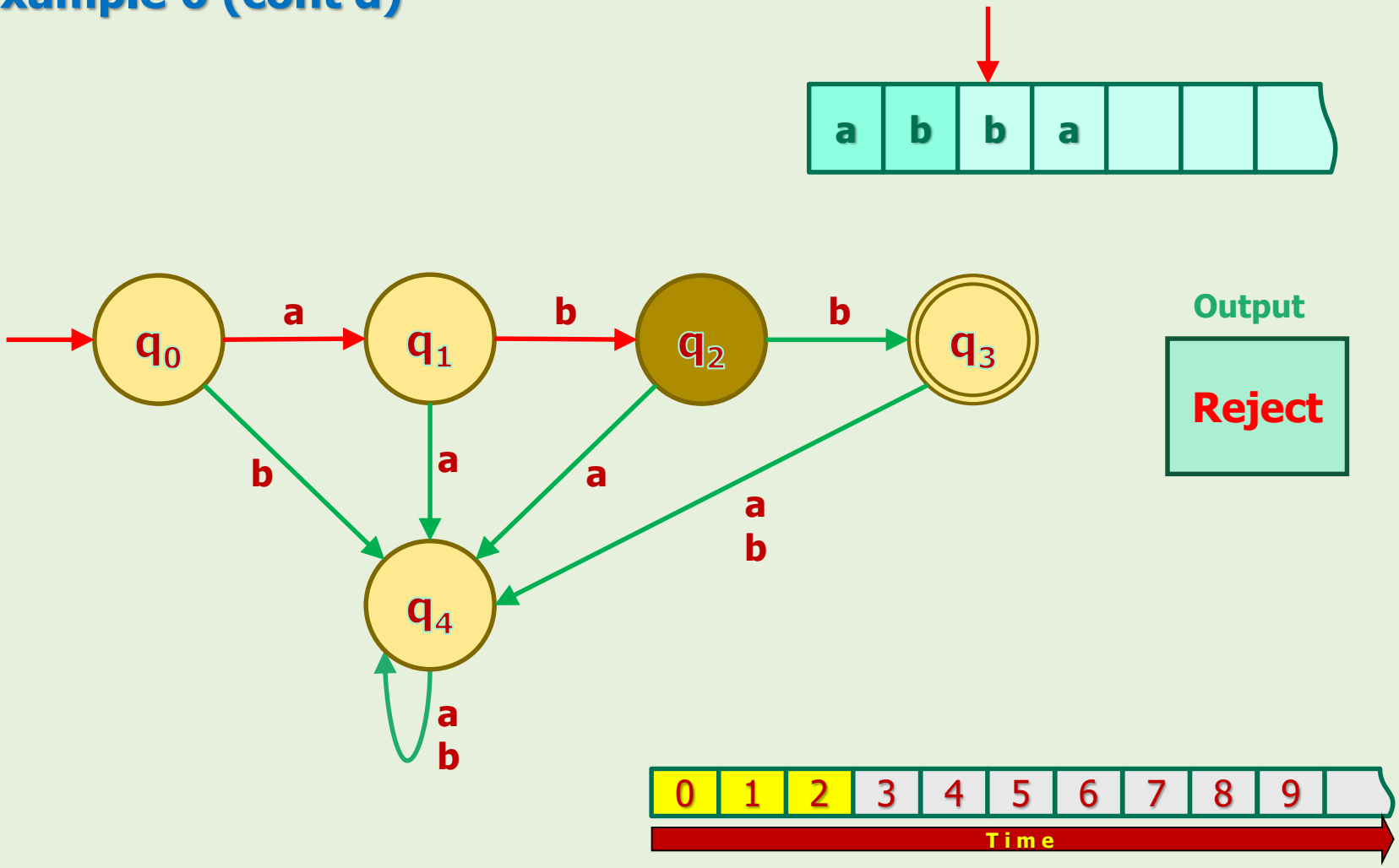
5. DFAs in Action

Example 6 (cont'd)



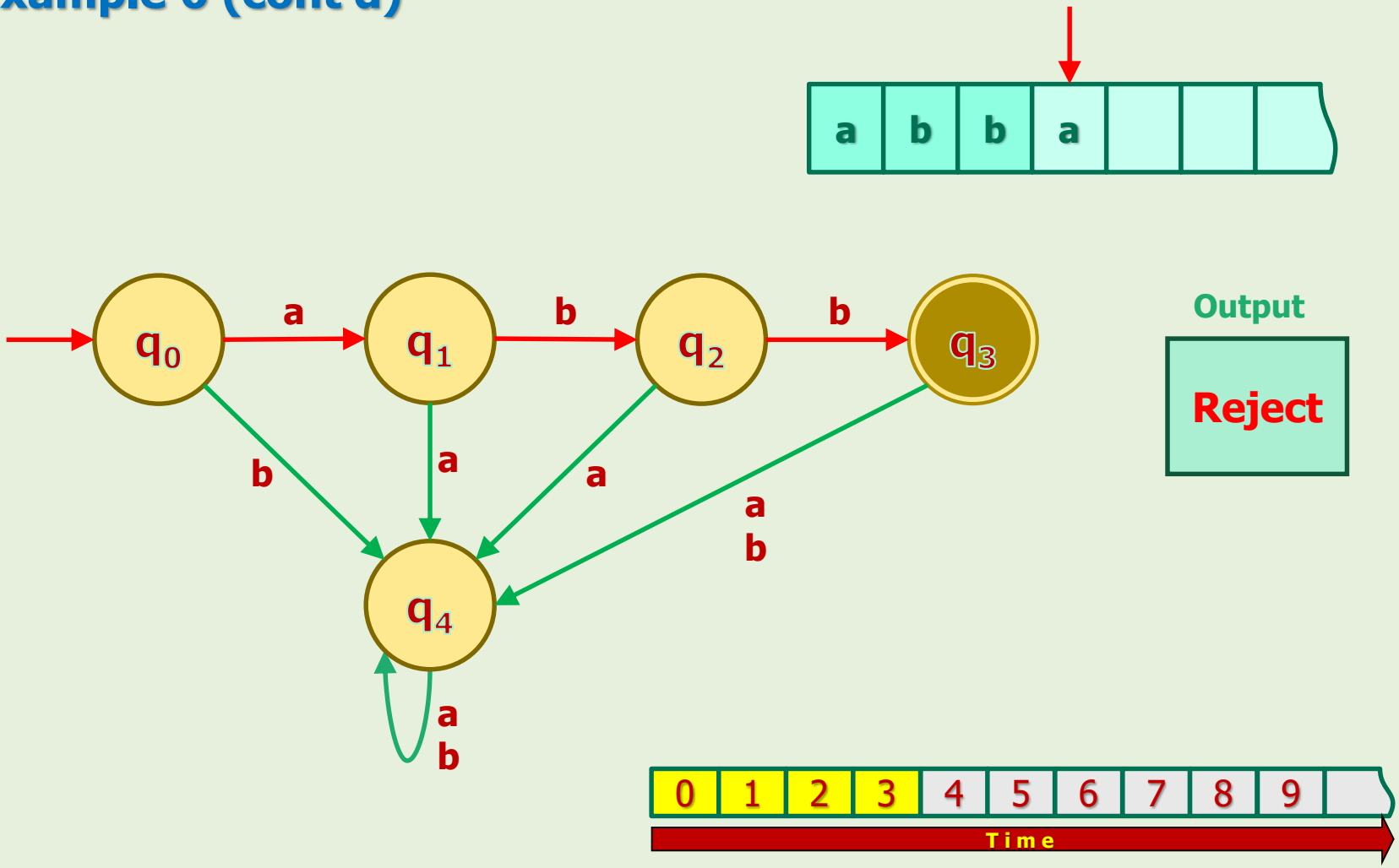
5. DFAs in Action

Example 6 (cont'd)



5. DFAs in Action

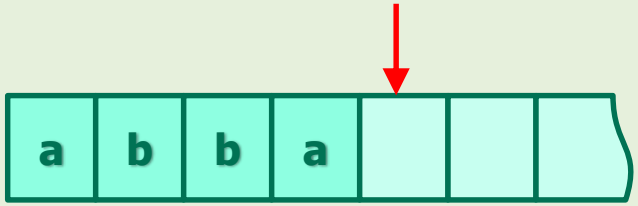
Example 6 (cont'd)

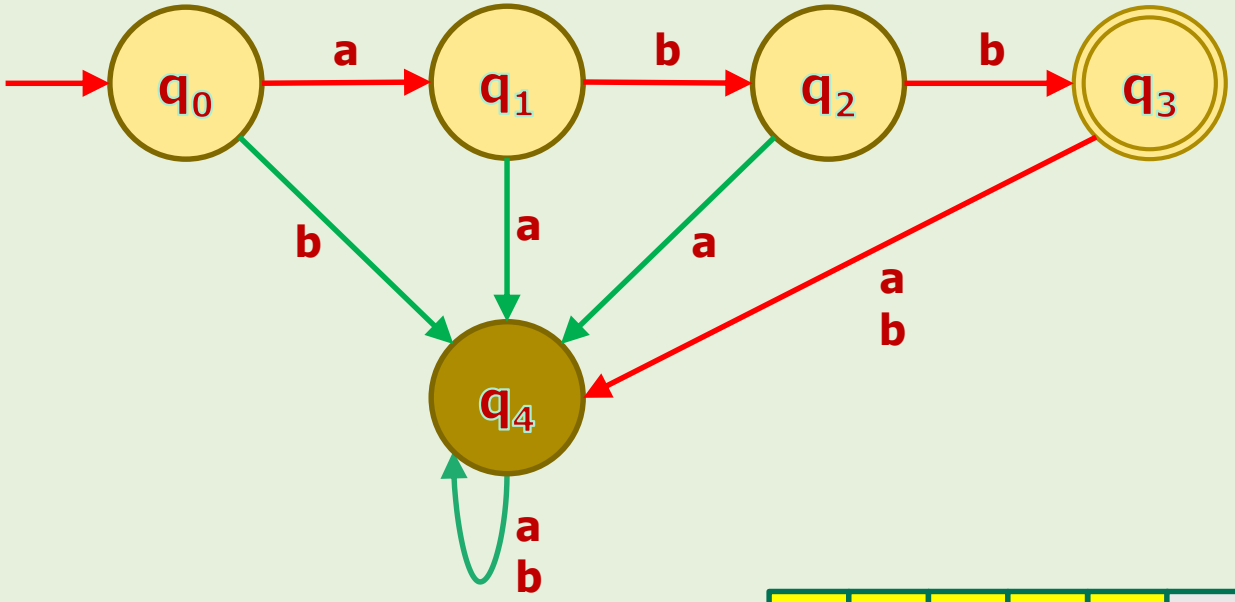


5. DFAs in Action

Example 6 (cont'd)

The machine did not understand "abba"!



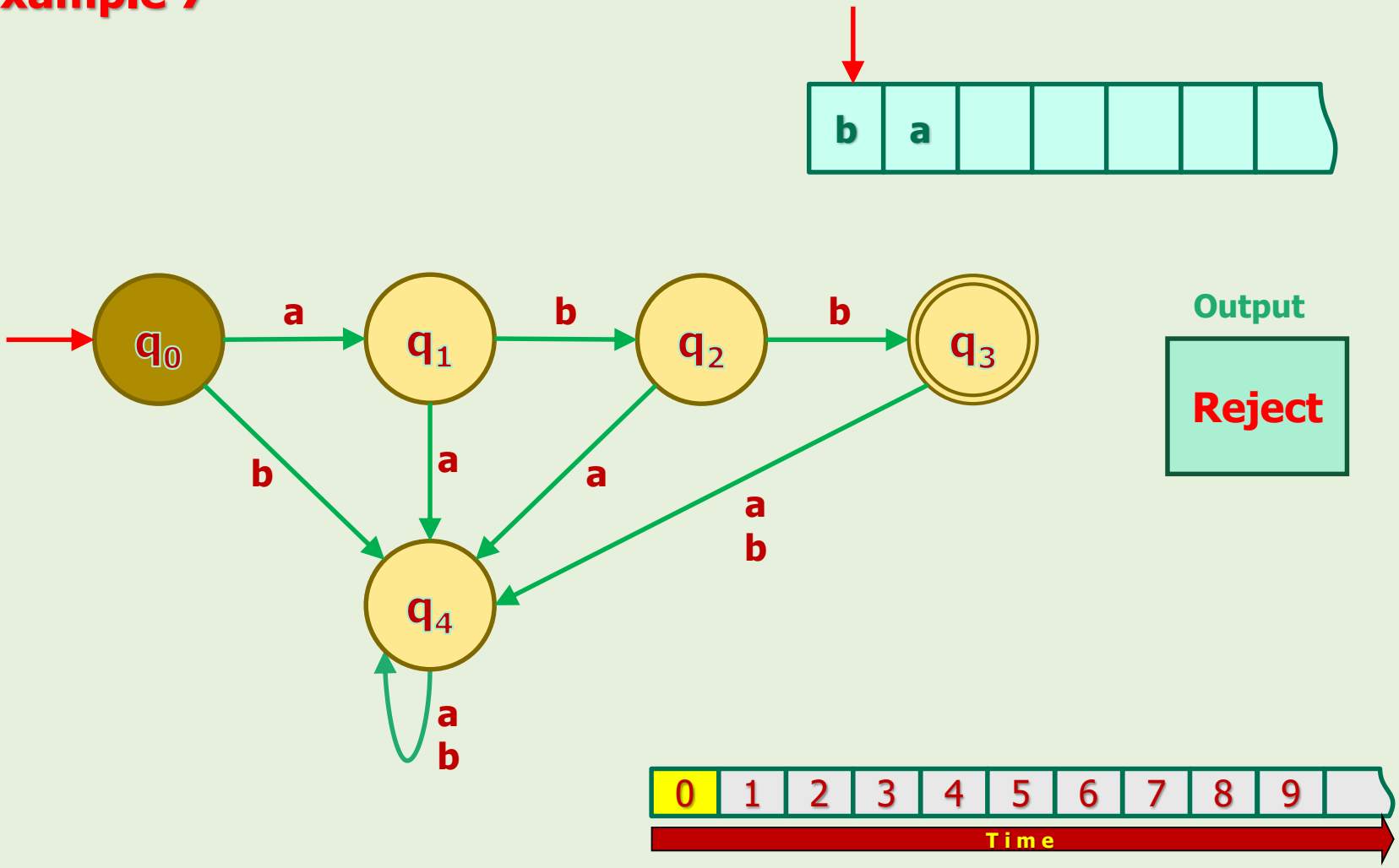


Output

Reject

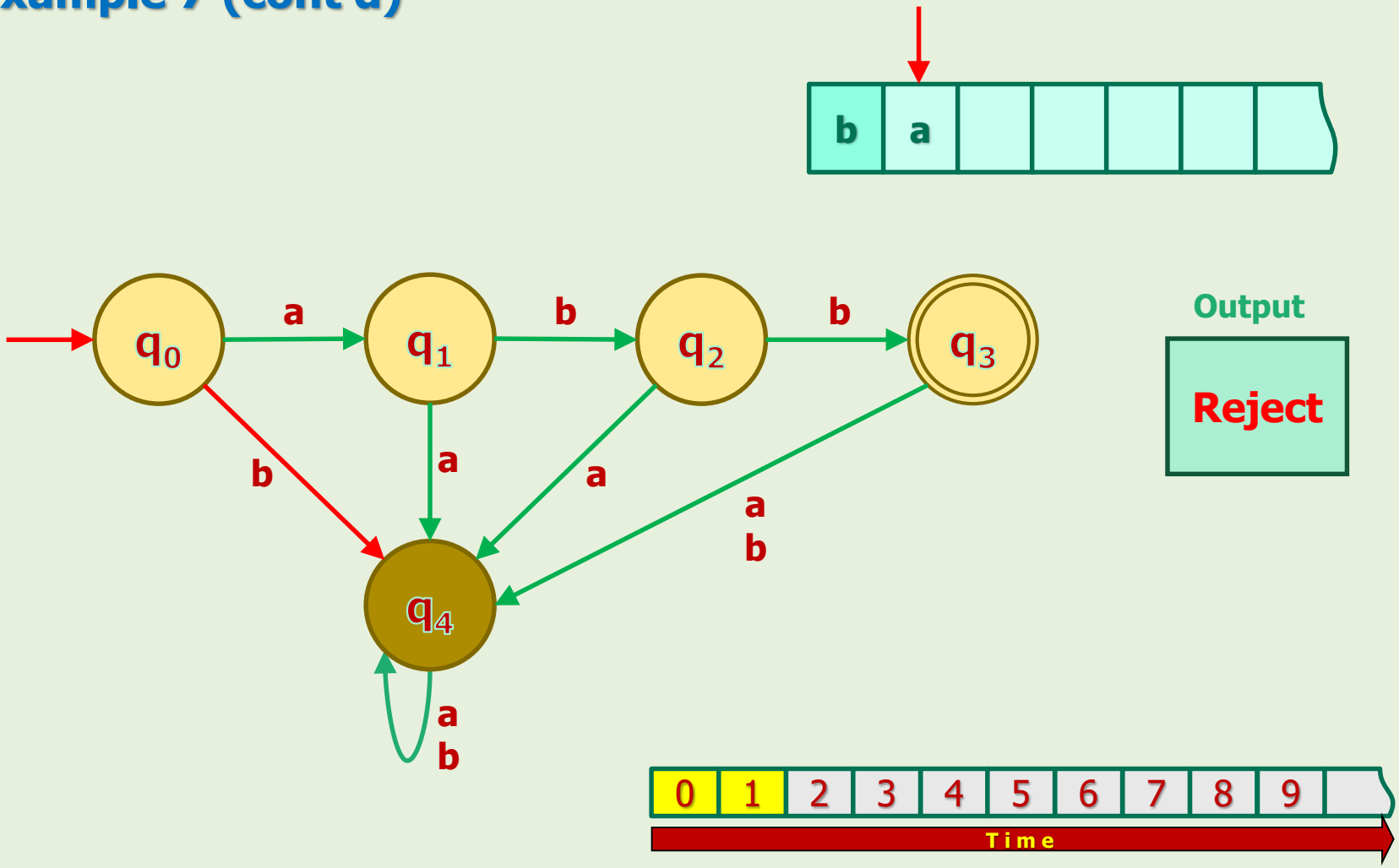
5. DFAs in Action

Example 7



5. DFAs in Action

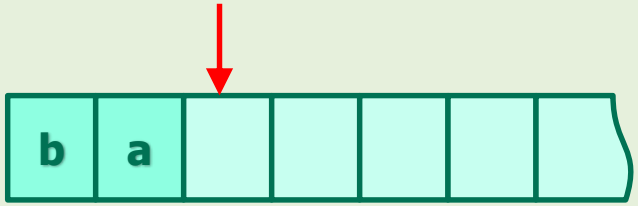
Example 7 (cont'd)

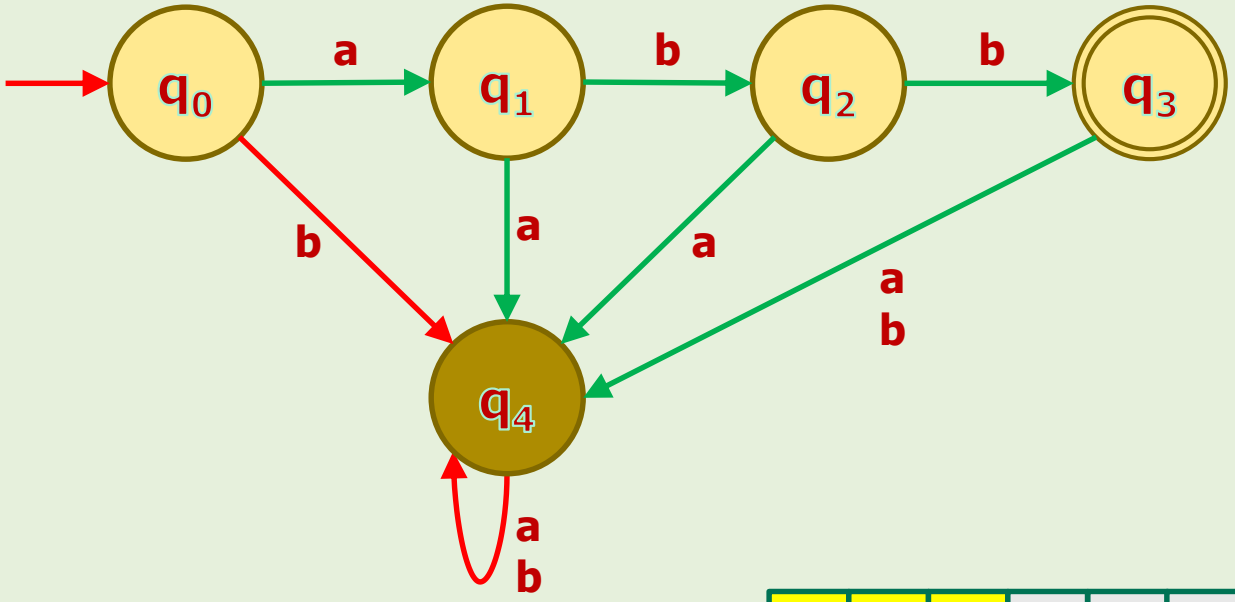


5. DFAs in Action

Example 7 (cont'd)

The machine did not understand "ba"!





Output

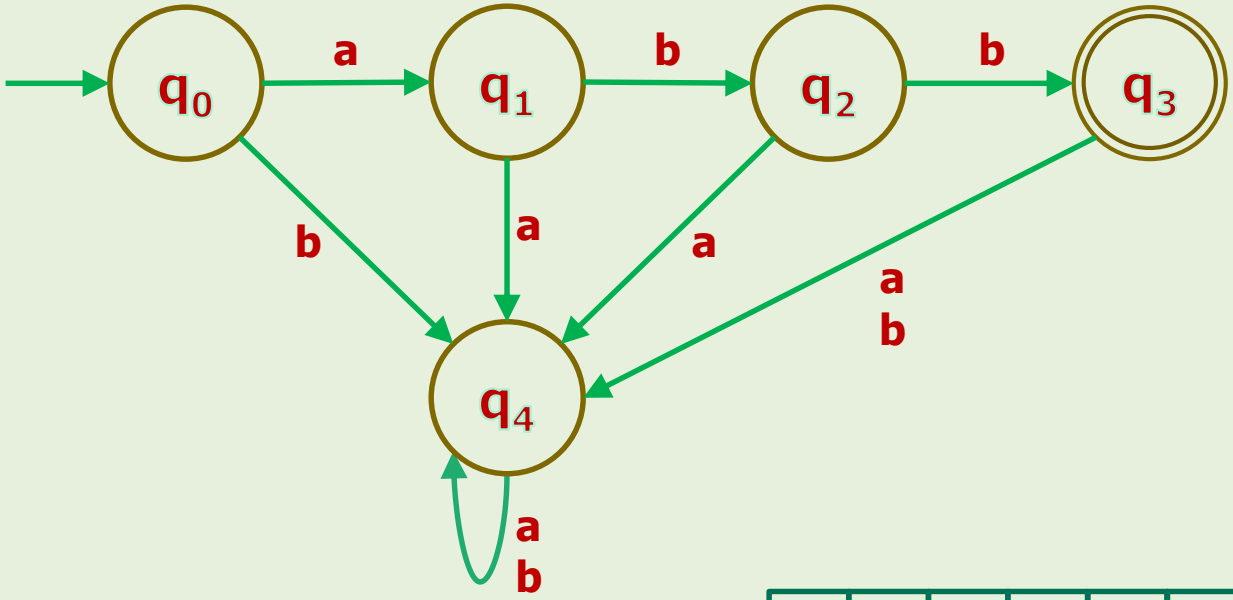
Reject

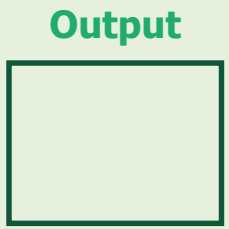
5. DFAs in Action

Example 8 **The machine is off!**

$w = \lambda$

- How'd it work?



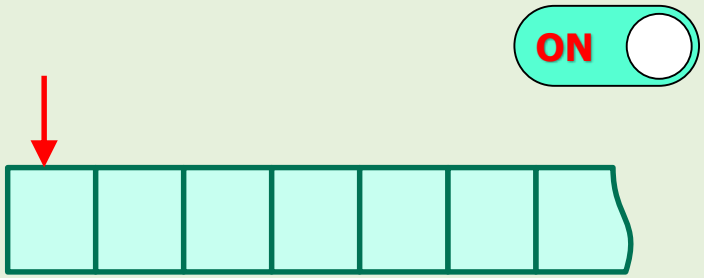


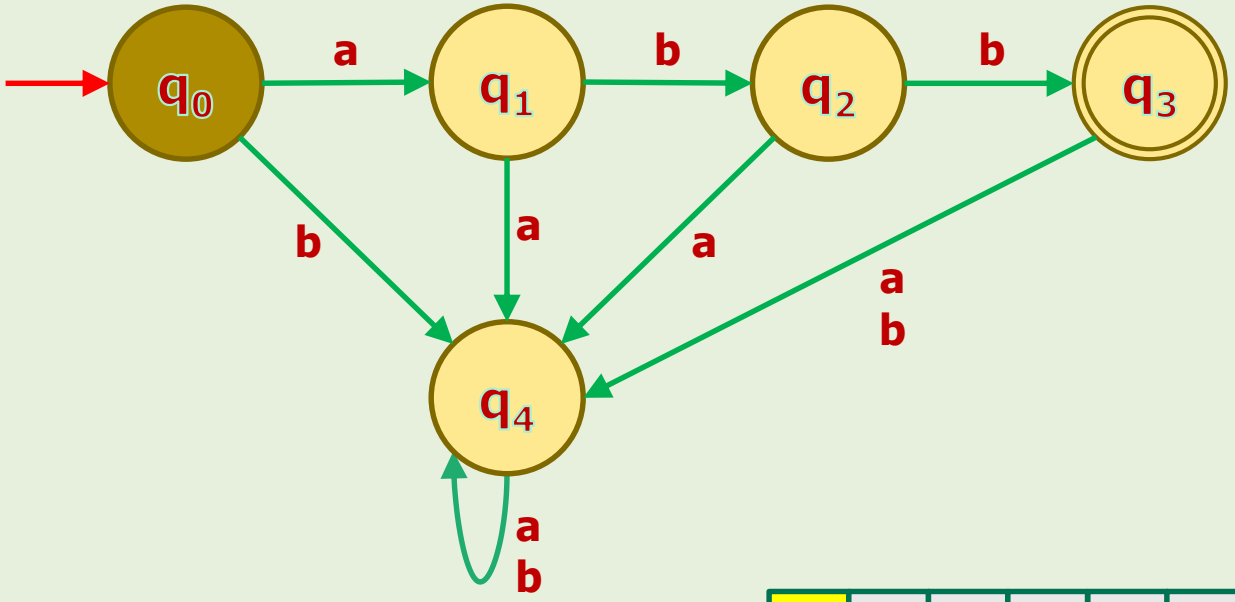
5. DFAs in Action

Example 8 (cont'd)

$w = \lambda$

The machine did not understand "λ"!





Output

Reject

4.4. How DFAs Accept/Reject Strings

Logical Representation of Accepting Strings

DFAs accept a string w. ≡ **a**

IFF

They halt. ≡ **h**

AND

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

AND

They are 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, for DFAs, we can simplify the accepting logic as: $(c \wedge f) \leftrightarrow a$

4.4. 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:

$$\mathbf{(c \wedge f) \leftrightarrow a}$$

$$\mathbf{\equiv \sim (c \wedge f) \leftrightarrow \sim a}$$

- Applying DeMorgan's rule on the left side:

$$\mathbf{\equiv (\sim c \vee \sim f) \leftrightarrow \sim a}$$

- What's the translation of this logical statement in plain English?

4.4. How DFAs Accept/Reject Strings

Logical Representation of Rejecting Strings

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

Translation

DFAs reject a string w. ≡ ~**a**

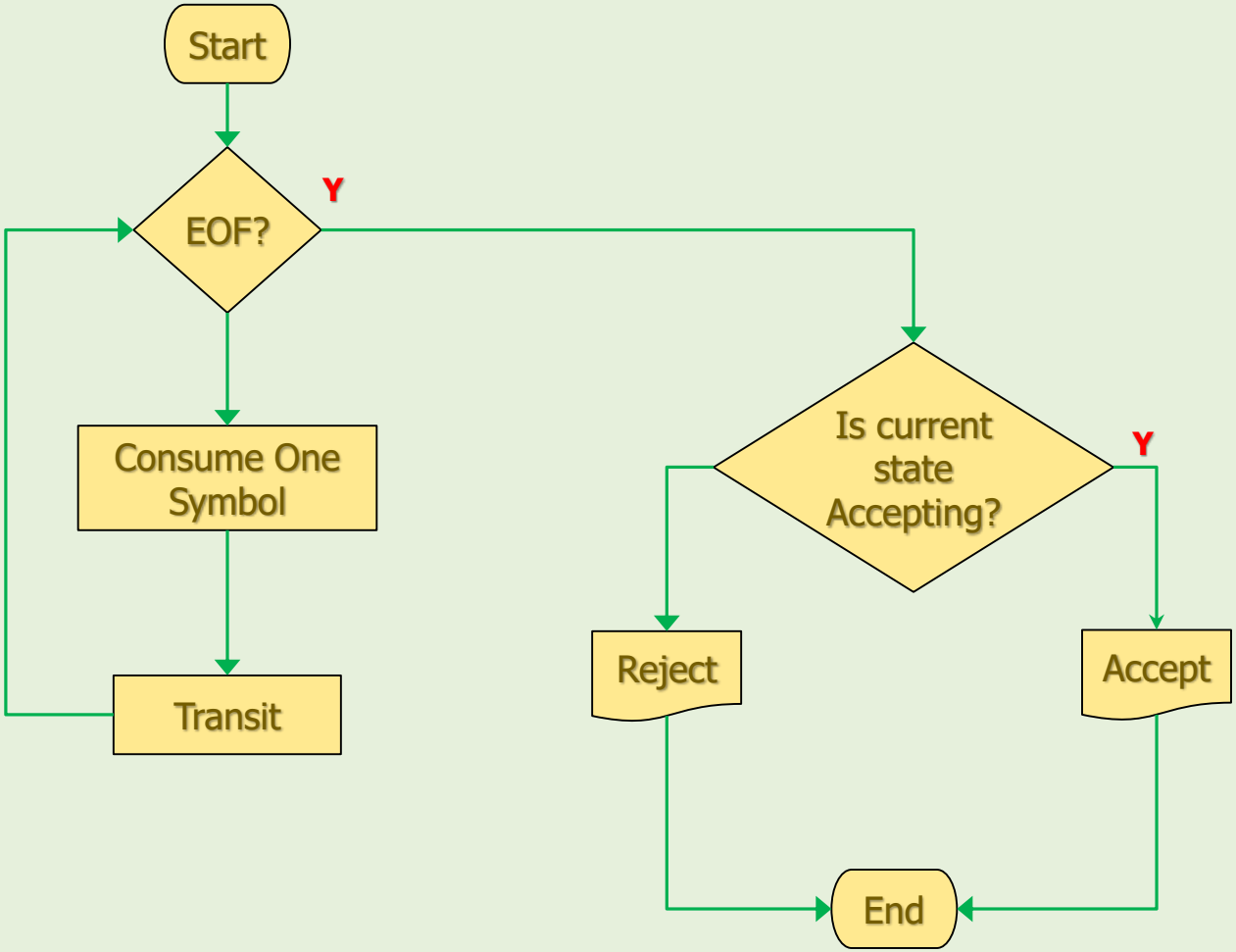
IFF

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

OR

They are NOT in an accepting (final) state. ≡ ~**f**

DFAs Operation Flowchart



DFAs Operation Pseudo Code

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