San José State University
Department of Computer Science

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

Nondeterministic Finite Automata

(Part 1)

Lecture 09

Day 09/31

CS 154

Formal Languages and Computability

Fall 2019

Agenda of Day 09

- About Midterm 1
- Summary of Lecture 08
- Quiz 3
- Lecture 09: Teaching ...
  - Nondeterministic Finite Automata (Part 1)

About Midterm 1

Reminder 1

- Midterm #1 (aka Quiz+)
  - Date: Thursday 09/26
  - Value: 10%
  - Topics: Everything covered from the beginning of the semester
  - Type: Closed $y \in$ Material

    Material = {Book, Notes, Electronic Devices, Chat, ... }

- The cutoff for this midterm is the end of lecture 09.

Study Guide

- I'll overview the type and number of questions via Canvas.

Summary of Lecture 08: We learned ...

DFAs

- A DFA M is defined by a quintuple:

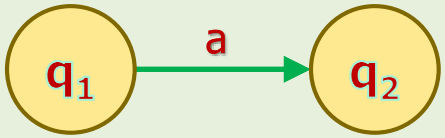
  $$M = (Q, \Sigma, \delta, q_0, F)$$

- 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 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.
- Every sub-rule like $\delta(q_1, a) = q_2$ represents a transition in transition graph.



Any question?

Summary of Lecture 08: We learned ...

DFAs

- Why is δ total function?
  - ... because if not, in some situations, the DFA does not know where to go!
- DFAs constraint: ...
  - ... every state must have an outgoing transition for every symbols of Σ.
- In other words:
  - At any timeframe, there is one and only one transition for every symbols of Σ.
- Associated language to a DFA M is ...
  - ... the set of all strings that it accepts.
  - ... denoted by L(M).

- Two machines are equivalent if ...
  - ... their associated languages are equal.
- Computation is ...
  - ... the sequence of configurations from when the machine starts until it halts.
- A machine is called deterministic if ...
  - ... at any timeframe, there is NO MORE THAN ONE possible transition.

Any question?

NAME	Alan M. Turing		
SUBJECT	CS 154	TEST NO.	3
DATE	09/19/2019	PERIOD	1 / 2 / 3

TEST RECORD	
PART 1	123
PART 2	
TOTAL	

Your list # goes here!

Quiz 3

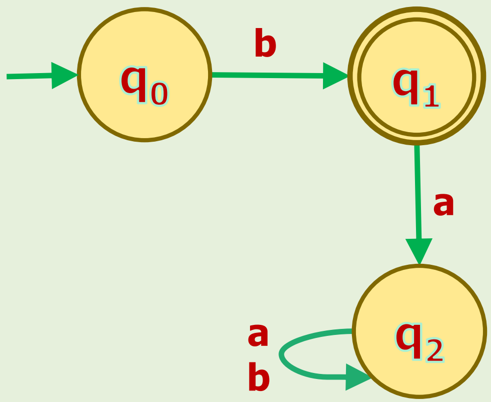
Use Scantron

Nondeterministic Finite Automata (NFA)

DFAs Constraint Violations

DFAs Constraint Violation #1

- What is the problem of the following DFA over $\Sigma = \{a, b\}$?



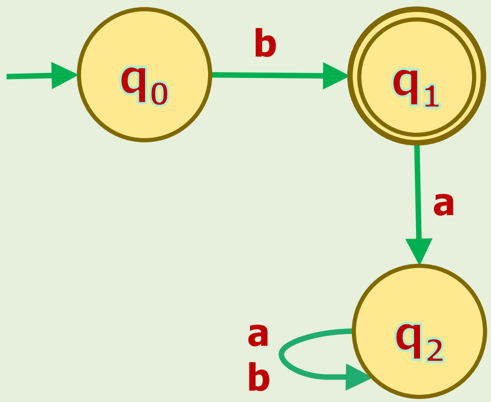
Violation

- The machine has no (zero) transition when it is in state q_0 and the input is a!
  - There is more like this in this graph, what are they?

- In other words, there are some timeframes that the machine does NOT KNOW WHERE TO GO?
  - Because there is no choice!

DFAs Constraint Violation #1

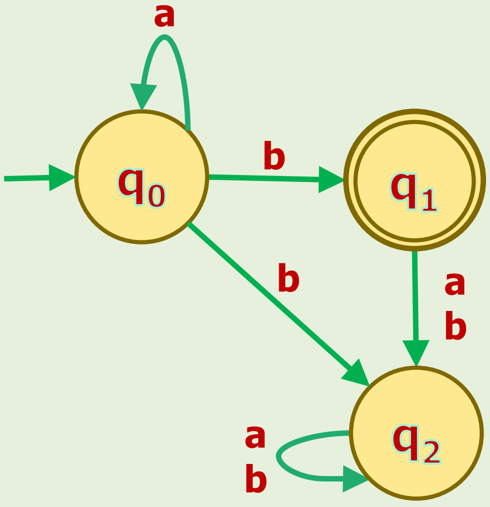
- What is the value of $\delta(q_0, a)$?
  - $\delta(q_0, a)$ = "Undefined"



- This violation makes δ a partial function.
- Also, it violates the DFAs' constraint ...
  - At any timeframe, there is one and only one transition for every symbols of Σ.
- So, the machine is NOT a DFA because it violates DFAs constraint!

DFAs Constraint Violation #2

- What is the problem of the following DFA over $\Sigma = \{a, b\}$?



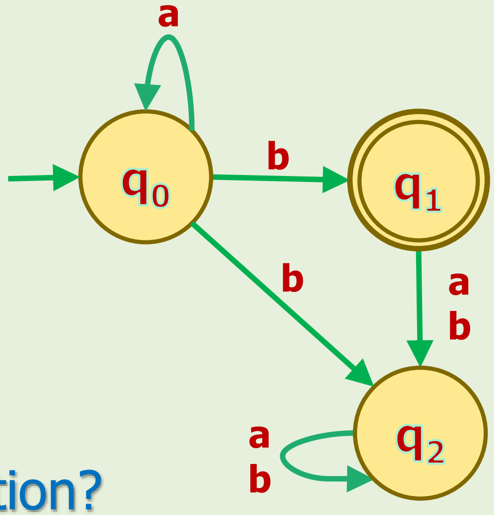
Violation

- The machine has more than one transition when it is in state q_0 and the input is b!

- In other words, there are some timeframes that the machine does NOT KNOW WHERE TO GO?
  - Because there are more than one choice!

DFAs Constraint Violation #2

- What is the value of $\delta(q_0, b)$?
  - $\delta(q_0, b) = \{q_1, q_2\}$
  - The range has more than one value!
  - So, we have to put them in a set.



- What type of function is the transition function?
  - It is NOT a regular function because it violates the definition of function.
  - It is called "multifunction" (aka multivalued function) in math.

- So, the machine is NOT a DFA because it violates DFAs constraint!

DFAs Constraint Violations Summary

- **Violation #1**: There are some timeframes that the machine has no (zero) transition.
  - The transition function is NOT total function.

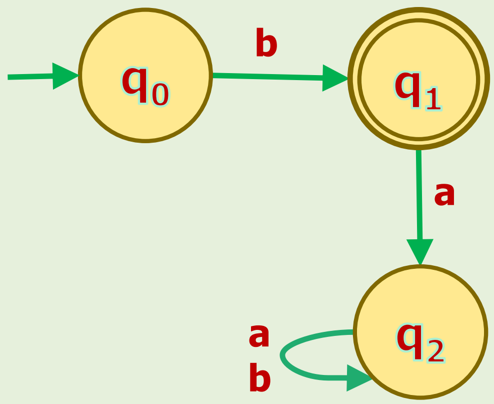
- **Violation #2**: There are some timeframes that the machine has more than one transition.
  - The transition function is multifunction (aka multivalued function).

- Now, let's relax the DFAs constraint and define a new class of machines!
- We are going to define a new class that these violations become legal!

Relaxed Transition Function Examples

Example 1

- Write the rule of the following transition graph over $\Sigma = \{a, b\}$ by using set in the range.



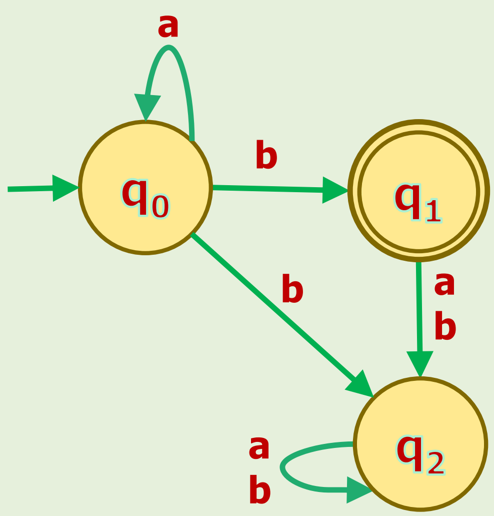
Solution

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

Relaxed Transition Function Examples

Example 2

- Write the rule of the following transition graph over $\Sigma = \{a, b\}$ by using set in the range.



Solution

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

Let's Construct a New Class of Automata

Repeated

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?

1. Why do we need a new class?

- The goal of introducing a new class is always having "more powerful" machines.
  - To understand the meaning of "power", we need more knowledge about formal languages that will be provided later.

- For now, let's claim that ...
- ... we relaxed the DFAs constraint to have simpler transition graph.

- We'll get back to this topic shortly.

2. Name of this New Class

- To figure out what to call this new class, let's review its characteristics.

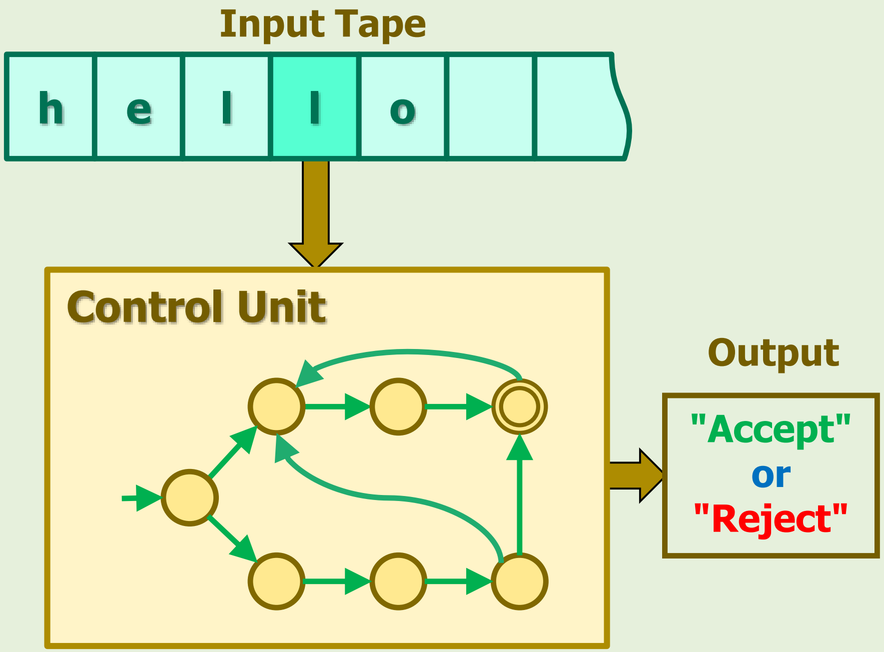
1. The second violation we mentioned earlier violates the definition of determinism.
   - In other words, during some timeframes, there might be more than one transitions.
   - So, this new class should be "nondeterministic".
2. The number of states is still "finite".

- Therefore, this new class is called:

"Nondeterministic Finite Automata (NFA)"

3. NFAs Building Blocks

- NFAs have the same building blocks as DFAs:
  1. Input Tape
  2. Control unit
  3. Output



- As usual, we don't need to show the output.

4. How do NFAs Work?

4. How do NFAs Work?

- To understand how NFAs 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?

- The starting configuration of NFAs is the same as DFAs'.
  – So, the first question is done.

- But we need to respond to the other three questions.

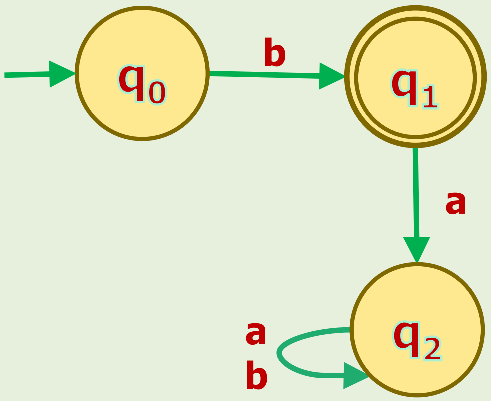
4. How do NFAs Work?

- DFAs' and NFAs' have the same building blocks.
- So, we expect their behavior be the same except ...
- … **for those two violations**.
- Therefore, we just need to define ...
  how NFAs behave when they encounter those two violations.
- And for the rest, it would behave exactly like DFAs.

NFAs' Behavior For the Violation #1

Violation #1

- There are some timeframes that NFAs have no (zero) transition.
  - e.g.: $\delta(q_0, a) = \{\ \}$



NFAs' Behavior

- NFAs halt. ≡ **h**

- So, NFAs halt iff:

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

  OR

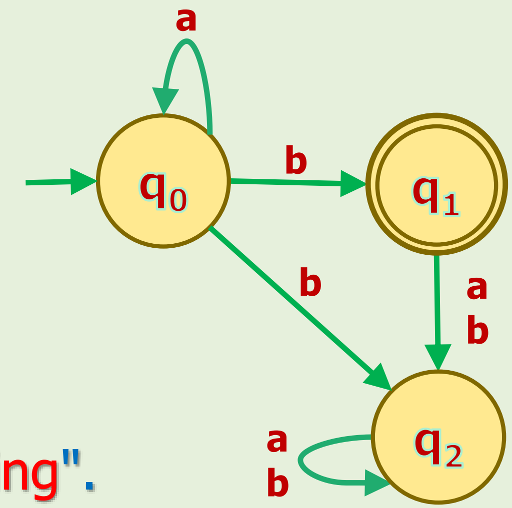
  They have zero transition. ≡ **z**

$$(c \lor z) \leftrightarrow h$$

NFAs' Behavior For the Violation #2

Violation #2

- There are some timeframes that NFAs have more than one transition.
  - e.g.: $\delta(q_0, b) = \{q_1, q_2\}$



NFAs' Behavior

- They check all possibilities by "parallel processing".
  1. They initiate another process.
  2. They replicate its entire structure.
  3. They initialize the new process with the current configuration.
  4. The new process independently continues processing the rest of the input string.

Summary of "4. How do NFAs Work?"

- So far, we've responded three out of four questions:

#	Question	Answer
1	What is the "starting configuration"?	Same as DFAs
2	What would happen during a timeframe?	Halting if Violation #1 Parallel processing if Violation #2 Same as DFAs for the rest
3	When would the machine halt?	$(c \vee z) \leftrightarrow h$
4	How would a string be Accepted/Rejected?	???

- Before responding to the last question,
  let's take some examples and see NFAs in Action!

5. NFAs in Action

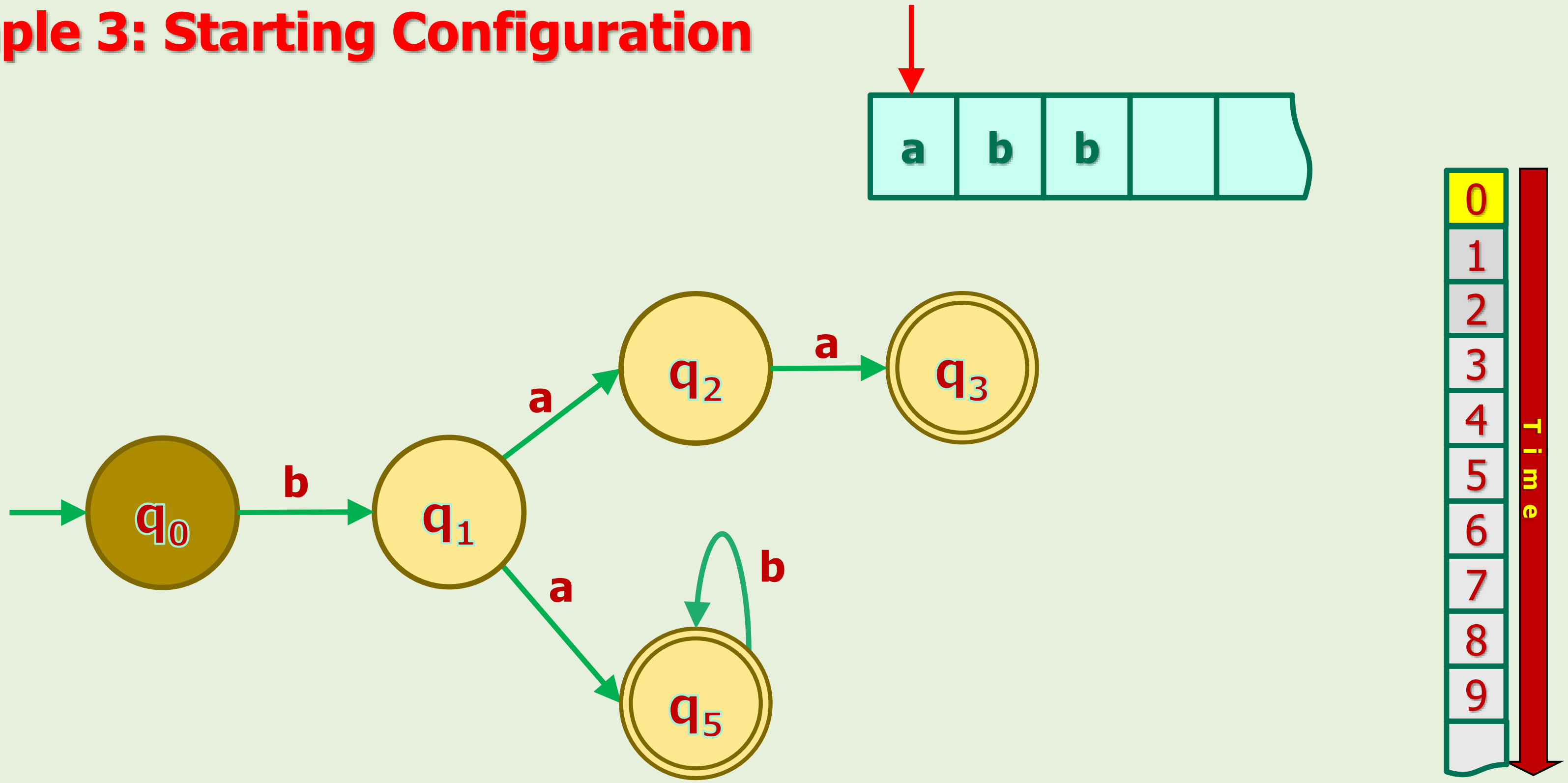
Review of NFAs' Input Tapes

- An NFA's input tape follows the following steps to consume a symbol:

1. It reads the symbol at which it is pointing and sends it to the control unit.
2. If the control unit can make a transition, then the read-head moves one cell to the right. Otherwise, the read-head stays put.

5. NFAs in Action

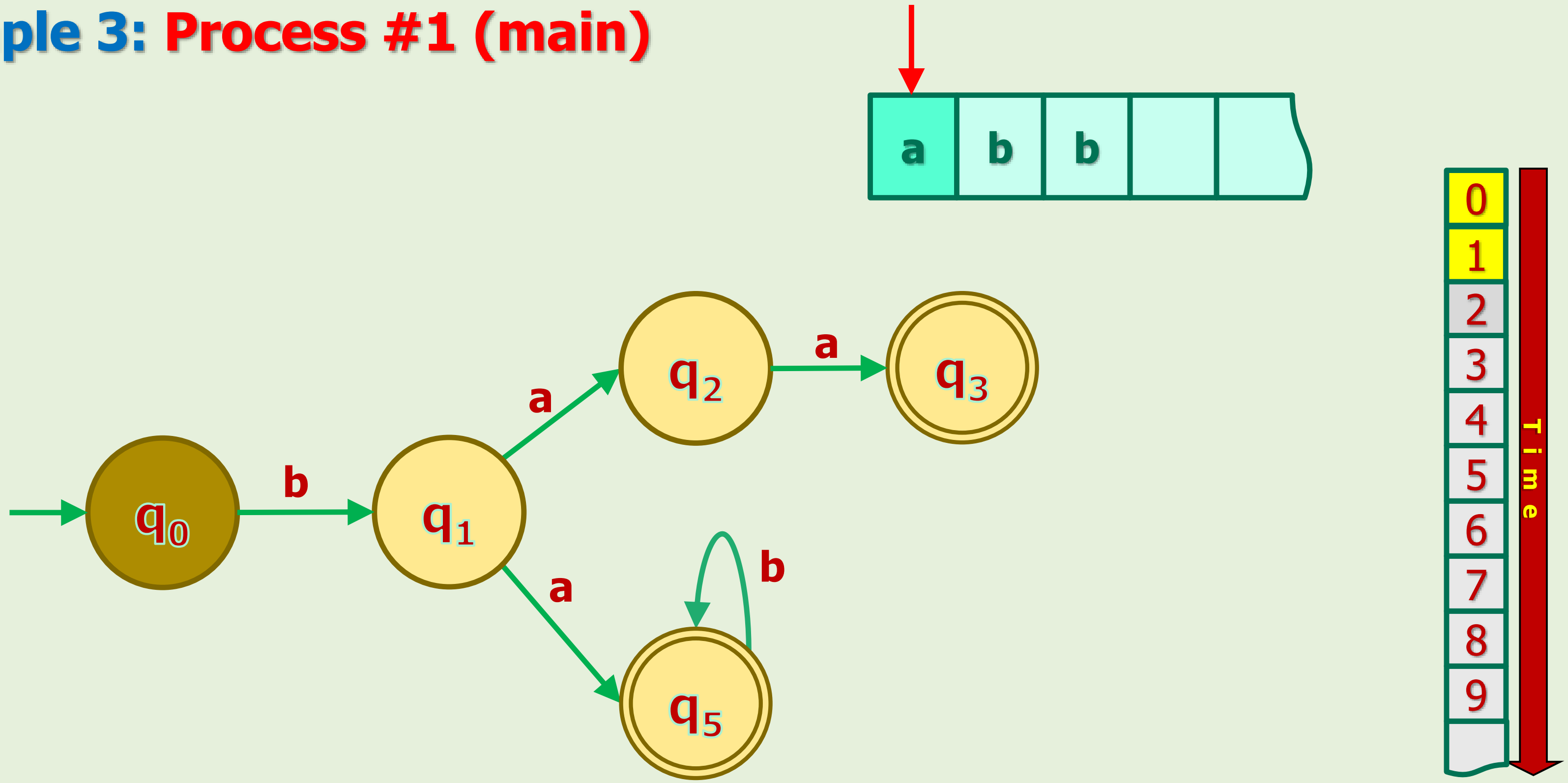
Example 3: Starting Configuration



- Process #1 (main) starts normally.

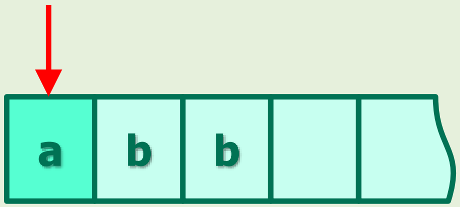
5. NFAs in Action

Example 3: Process #1 (main)

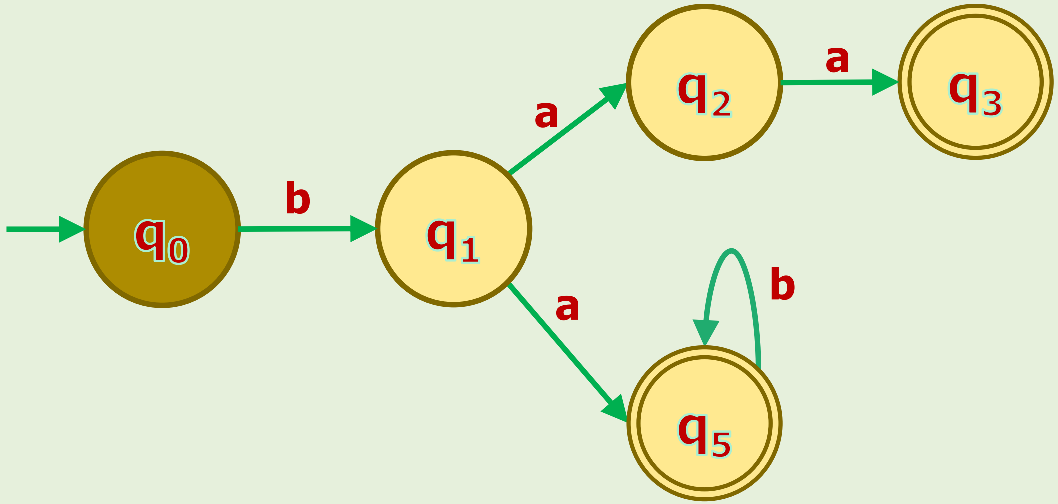


- Input tape reads 'a' and sends it to the control unit.
- The control unit makes a decision based on $\delta(q_0, a) = \{\ \}$

Example 3: Process #1 (main)



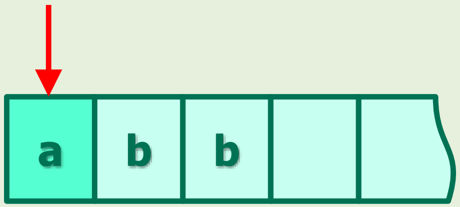
$\delta(q_0, a) = \{\ \}$



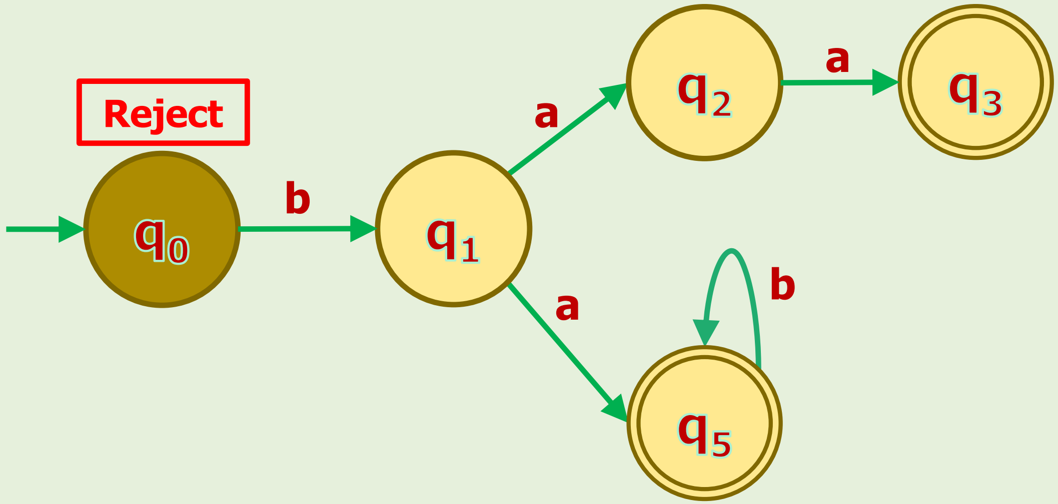
- The control unit cannot consume it because it has no choice for 'a'.
- The head DOES NOT move because control unit did not consume it.

5. NFAs in Action

Example 3: Process #1 (main)



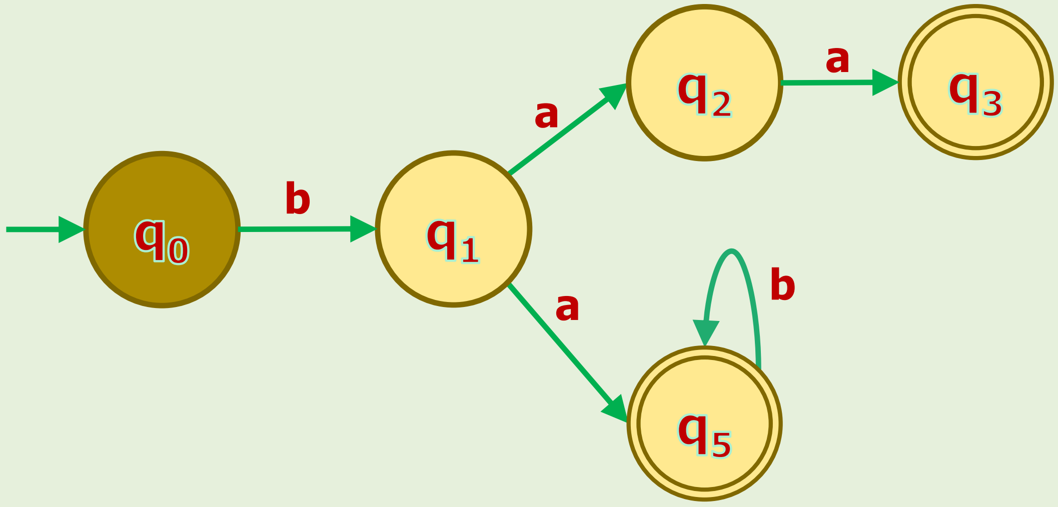
$\delta\,(q_0\,, a) = \{\ \}$



- It halts in the non-accepting state q_0. So, the string w is rejected.
- Also, note that all symbols are not consumed but one reason is enough for rejection!

5. NFAs in Action

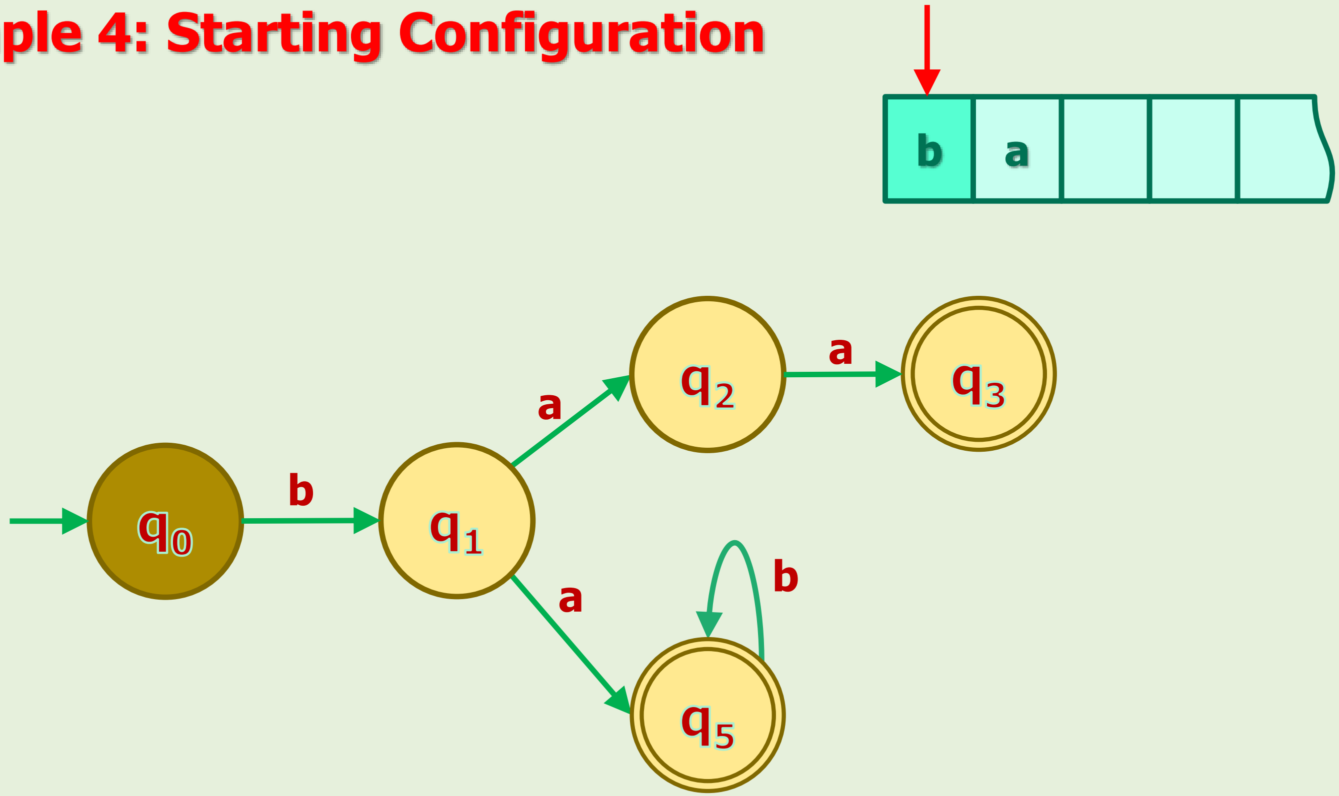
Example 4: Starting Configuration



- Process #1 (main) starts normally.

5. NFAs in Action

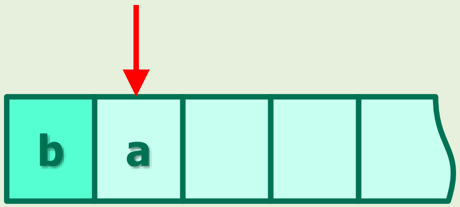
Example 4: Starting Configuration



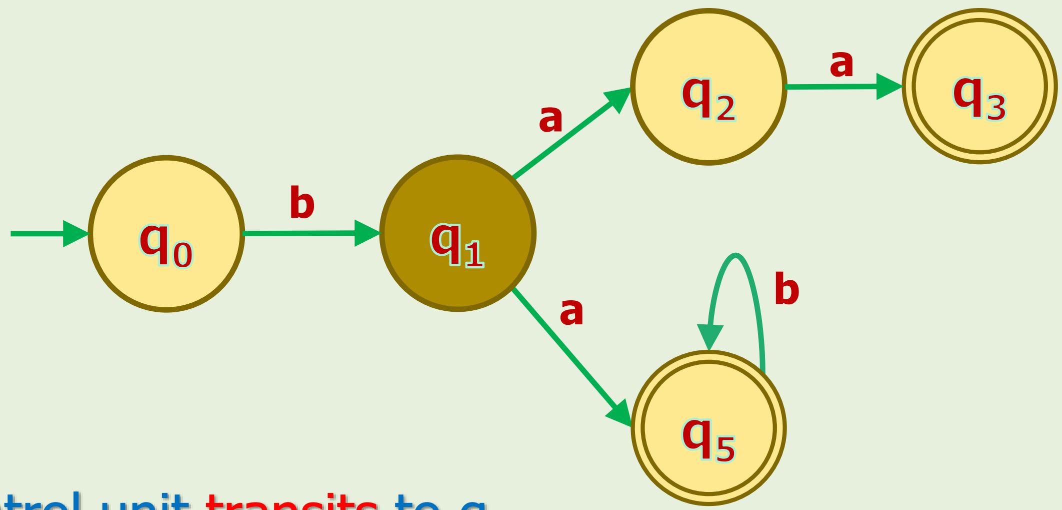
- Input tape reads 'b' and sends it to the control unit.
- The control unit makes a decision based on $\delta\ (q_0\ ,\ b) = \{q_1\}$

5. NFAs in Action

Example 4: Process #1 (main)



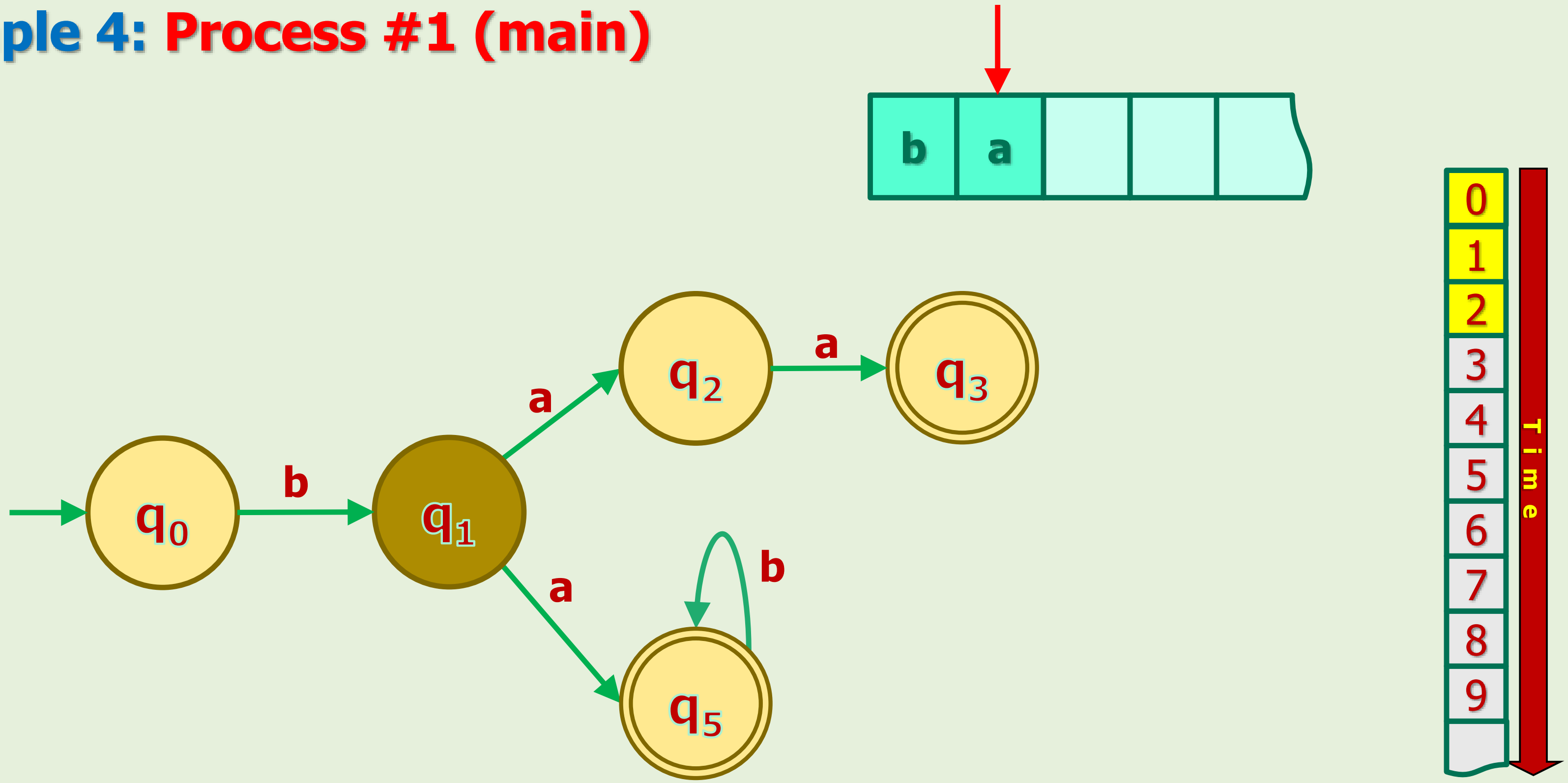
- $\delta(q_0, b) = \{q_1\}$



- Control unit transits to q_1.
- This is the end of timeframe 1.
- Up to this point, everything looks like DFAs'.
- What'd happen in the timeframe #2?

5. NFAs in Action

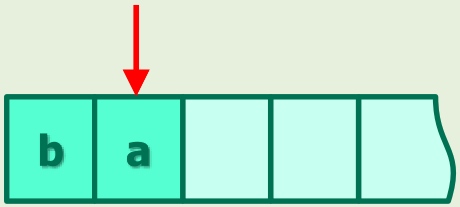
Example 4: Process #1 (main)



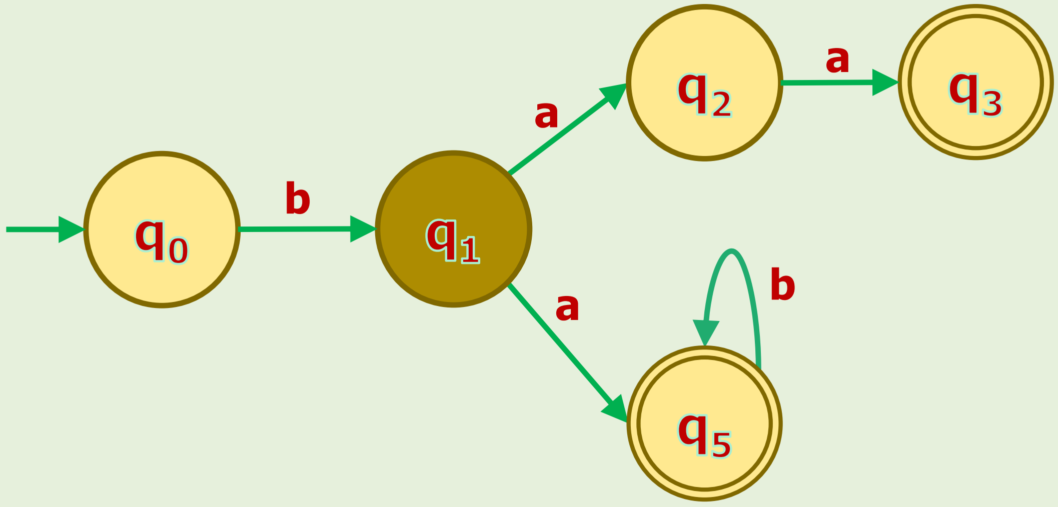
- Input tape reads 'a' and sends it to the control unit.
- The control unit makes a decision based on $\delta(q_1, a) = \{q_2, q_5\}$

5. NFAs in Action

Example 4: Process #1 (main)

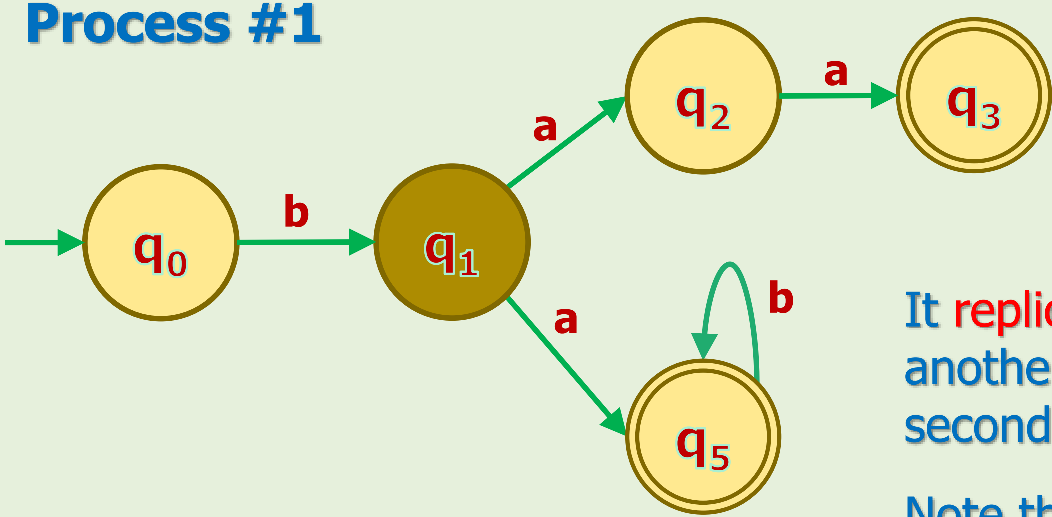


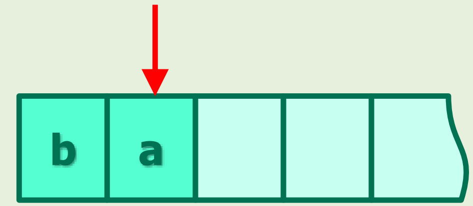
$\delta\ (q_1\ ,\ a) = \{q_2\ ,\ q_5\}$



- It encounters two possibilities: transition to q_2 or q_5.
- So, parallel processing starts!

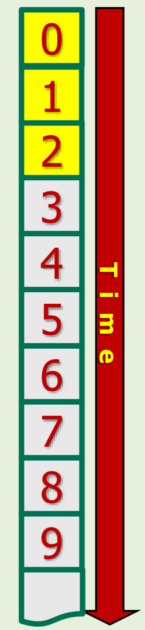
Process #1



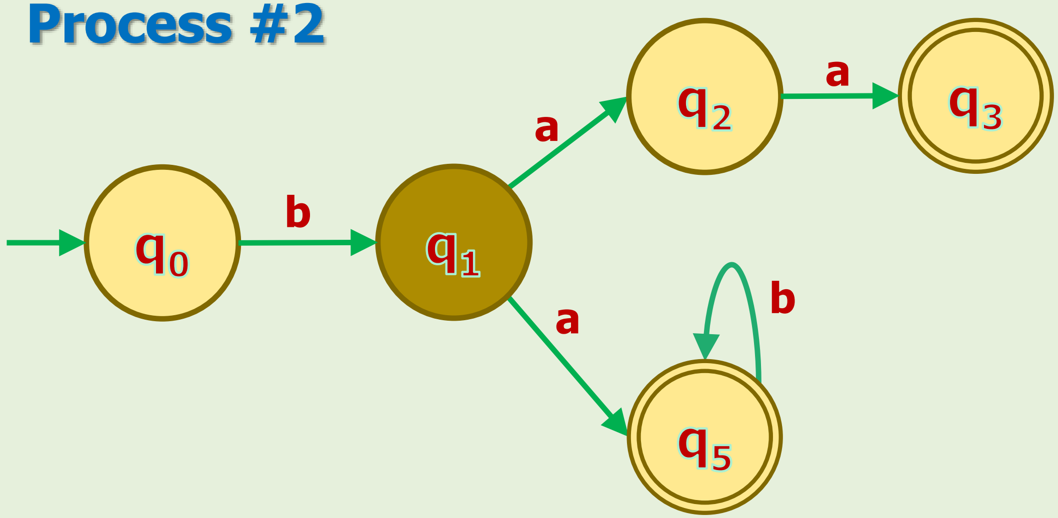


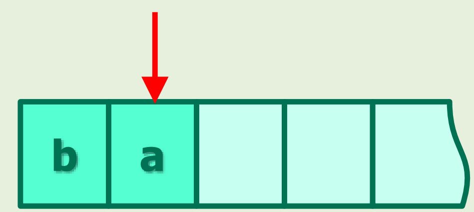
It replicates itself and another process will continue the second possibility.

Note that 'a' is not consumed yet!

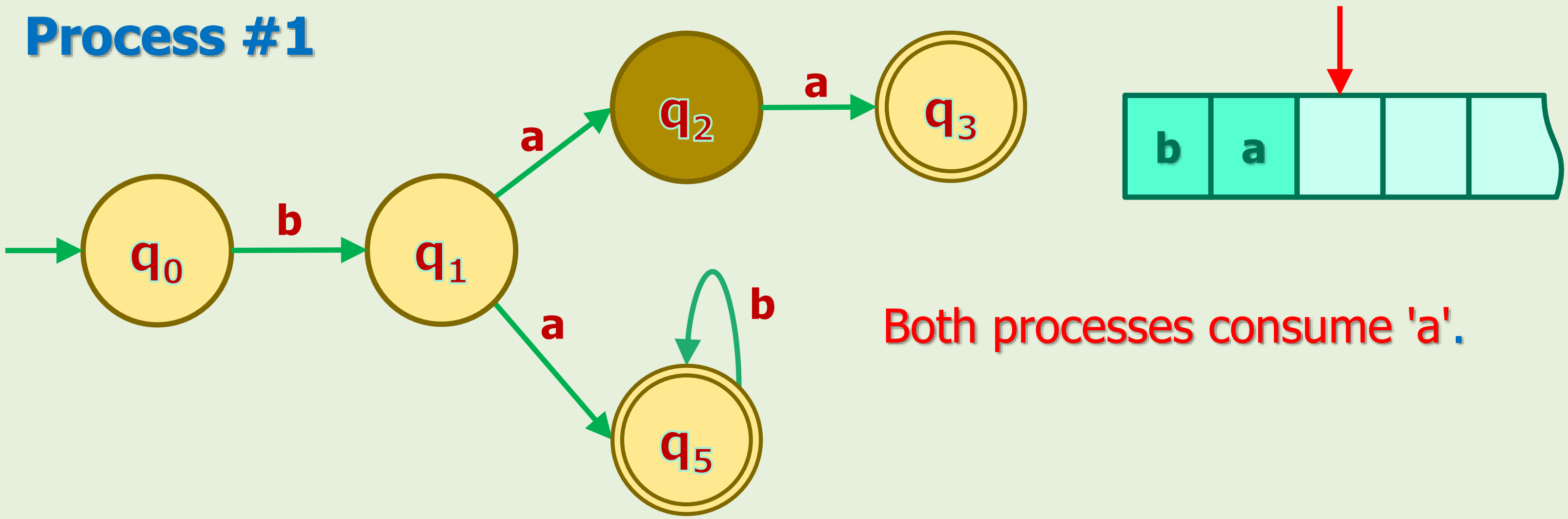


Process #2



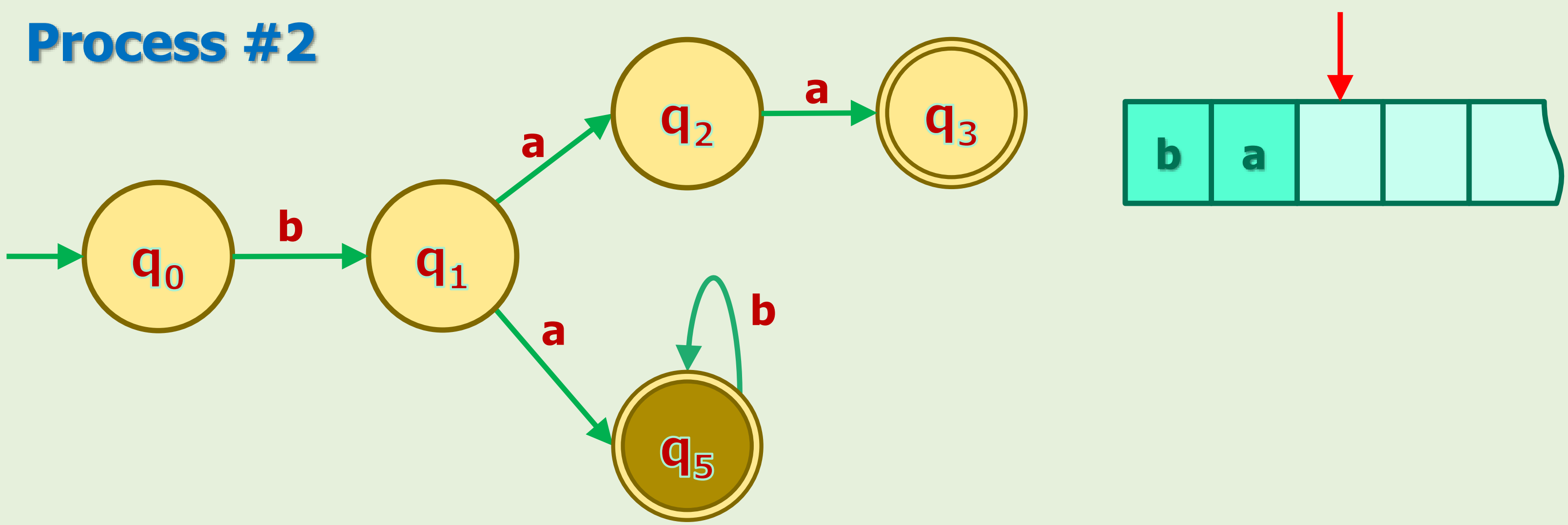


Process #1

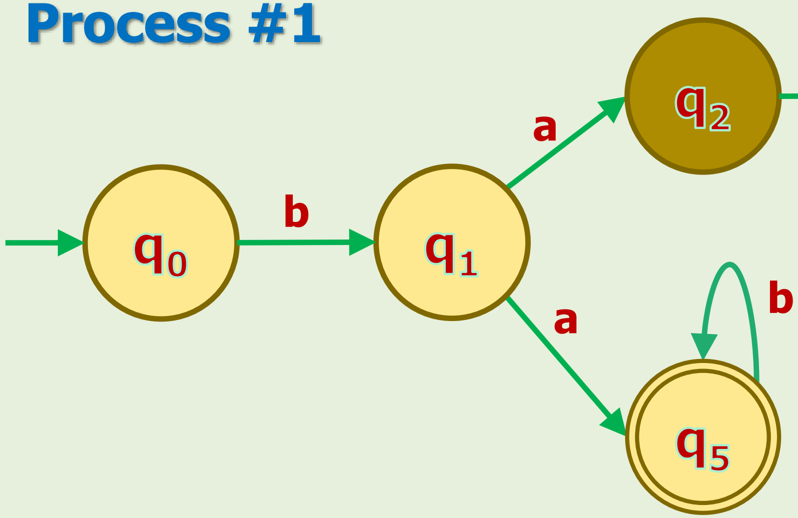


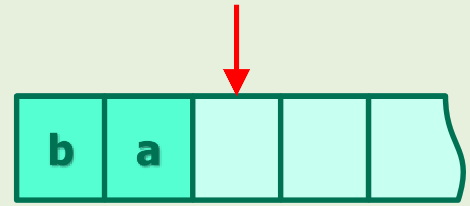
Both processes consume 'a'.

Process #2



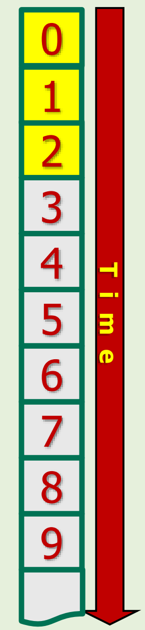
Process #1



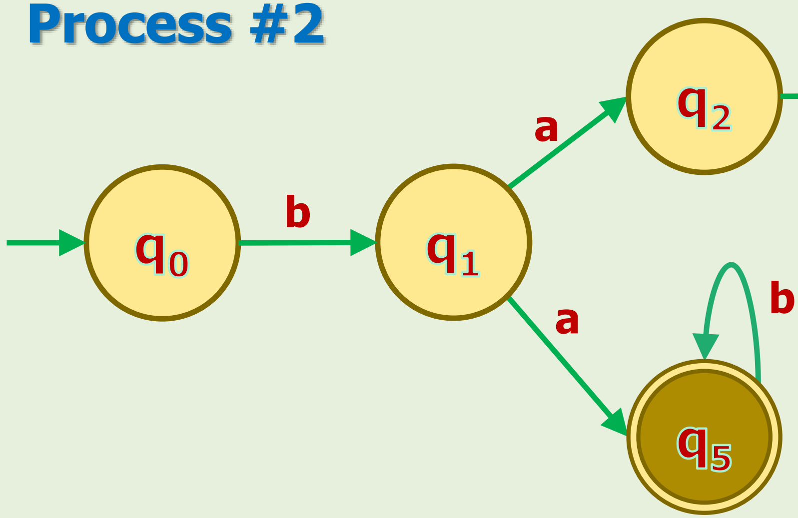


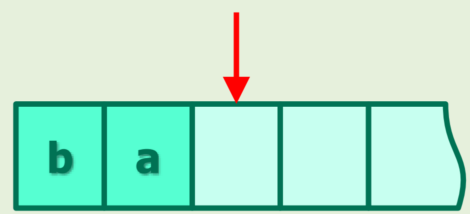
Process #1 is out of symbol.

It has to halt.



Process #2

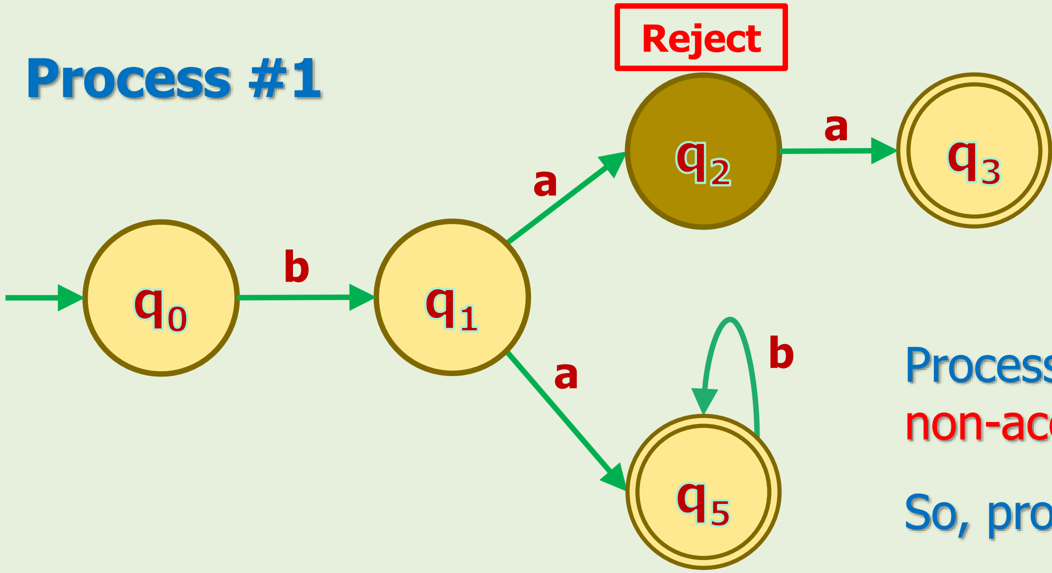


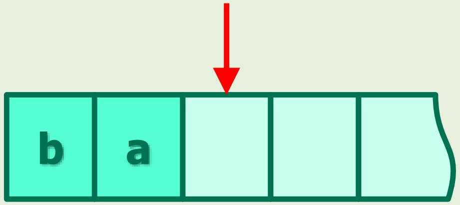


Process #2 is out of symbol.

It has to halt.

Process #1

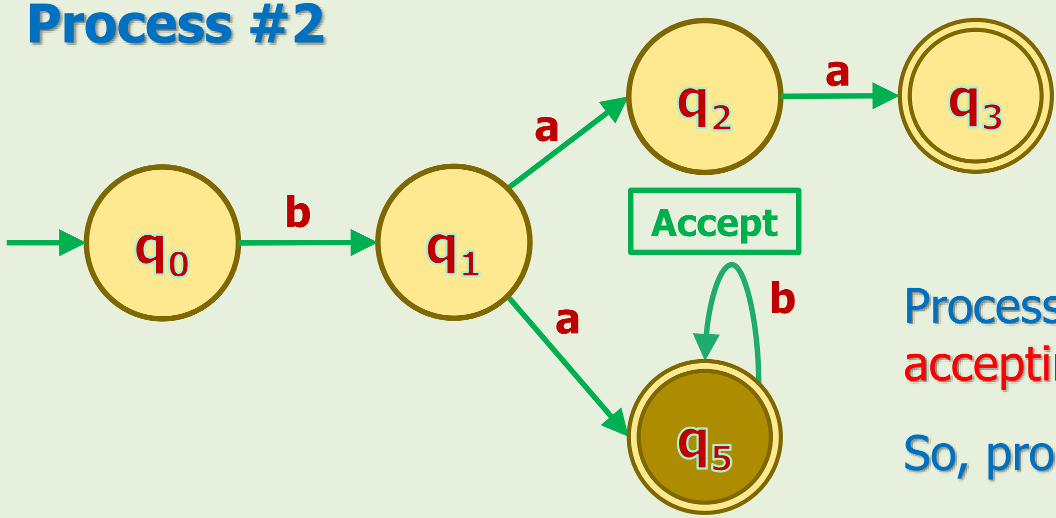


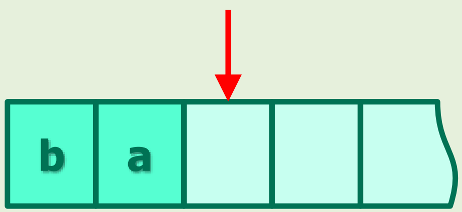


Process #1 halts in a non-accepting state.

So, process #1 rejects w.

Process #2



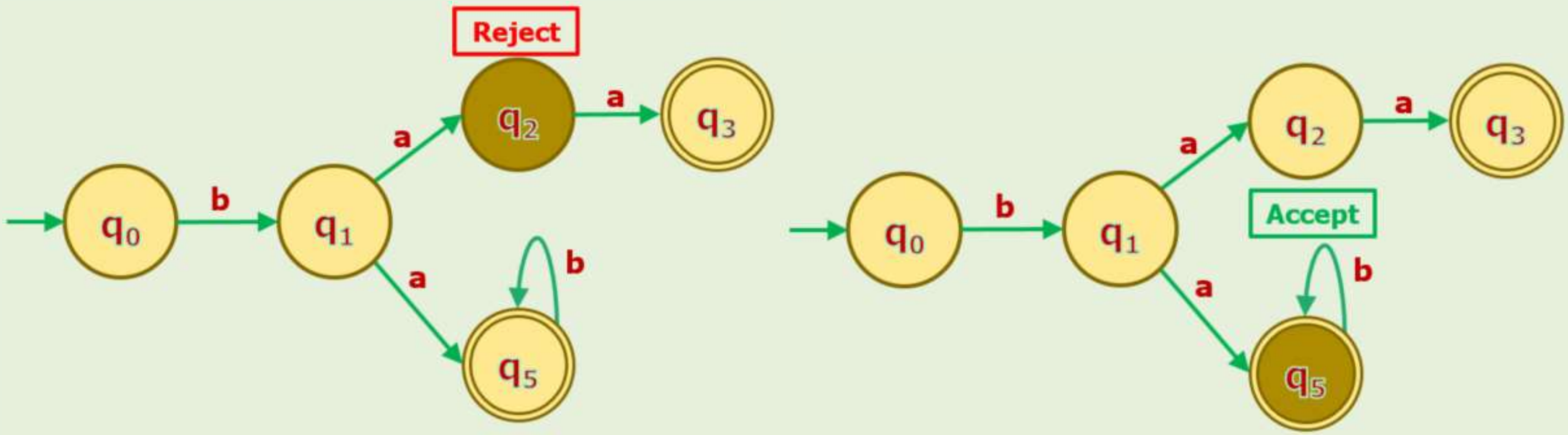


Process #2 halts in an accepting state.

So, process #2 accepts w.

5. NFAs in Action

Example 4: Overall Result



Process #1 REJECTED w = ba

Process #2 ACCEPTED w = ba

- Overall, the string was ACCEPTED because at least one process (#2) accepted it.

4.4 How NFAs Accept/Reject Strings

Accepting Strings

- NFAs accept a string iff at least one process accepts it.
- Note that a process accepts a string if all 3 conditions **($h \wedge c \wedge f$)** are satisfied. (i.e.: **($h \wedge c \wedge f$)** $\leftrightarrow$ **a**)
  - Because **h** and **c** might have different values.

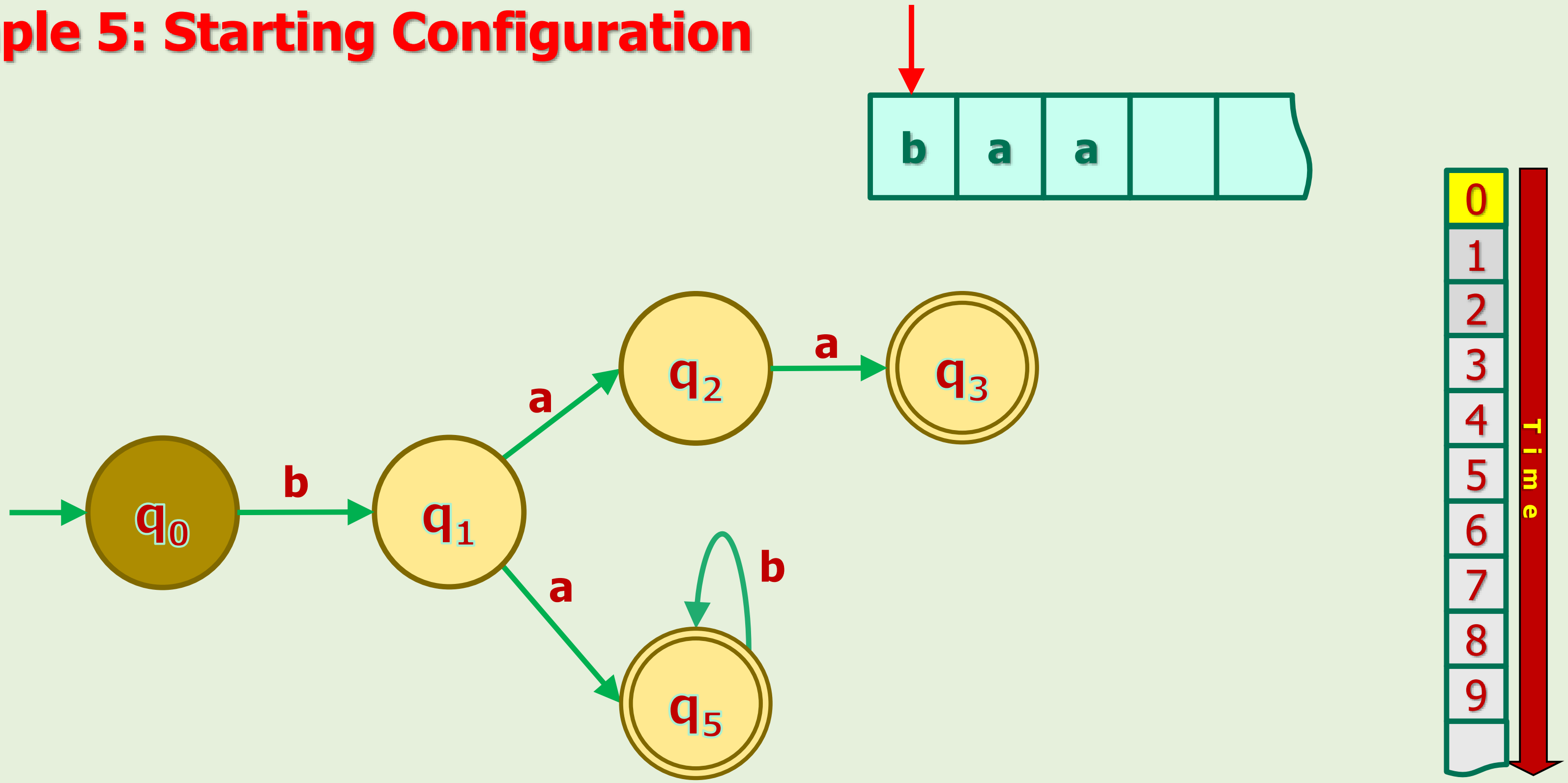
Rejecting Strings

- NFAs reject a string iff all processes reject it.
- Note that a process rejects a string if at least one of the 3 conditions **($\sim h \vee \sim c \vee \sim f$)** are satisfied. (i.e.: **($\sim h \vee \sim c \vee \sim f$)** $\leftrightarrow$ **~a**)

- Let's take more examples.

5. NFAs in Action

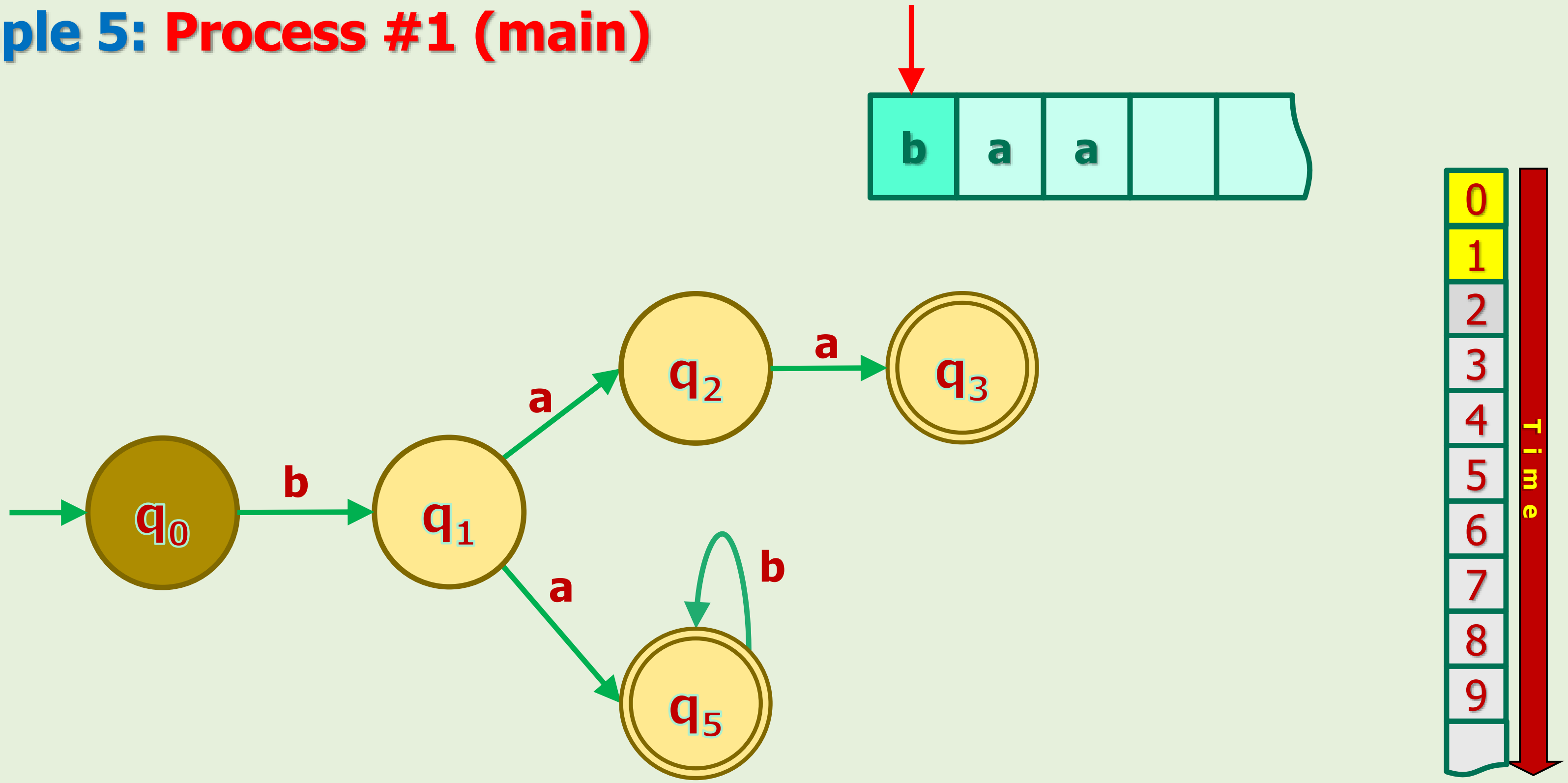
Example 5: Starting Configuration



- Process #1 (main) starts normally.

5. NFAs in Action

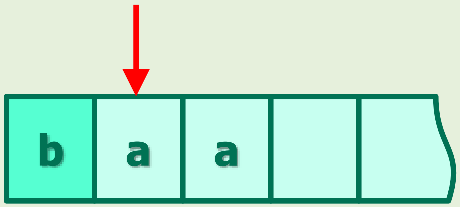
Example 5: Process #1 (main)



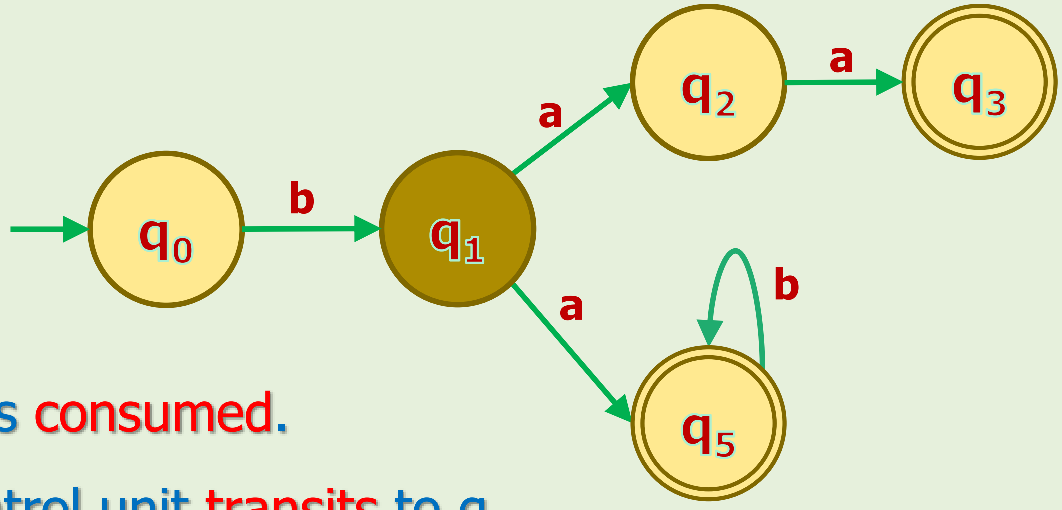
- Input tape reads 'b' and sends it to the control unit.
- The control unit makes a decision based on $\delta\ (q_0\ ,\ b) = \{q_1\}$

5. NFAs in Action

Example 5: Process #1 (main)



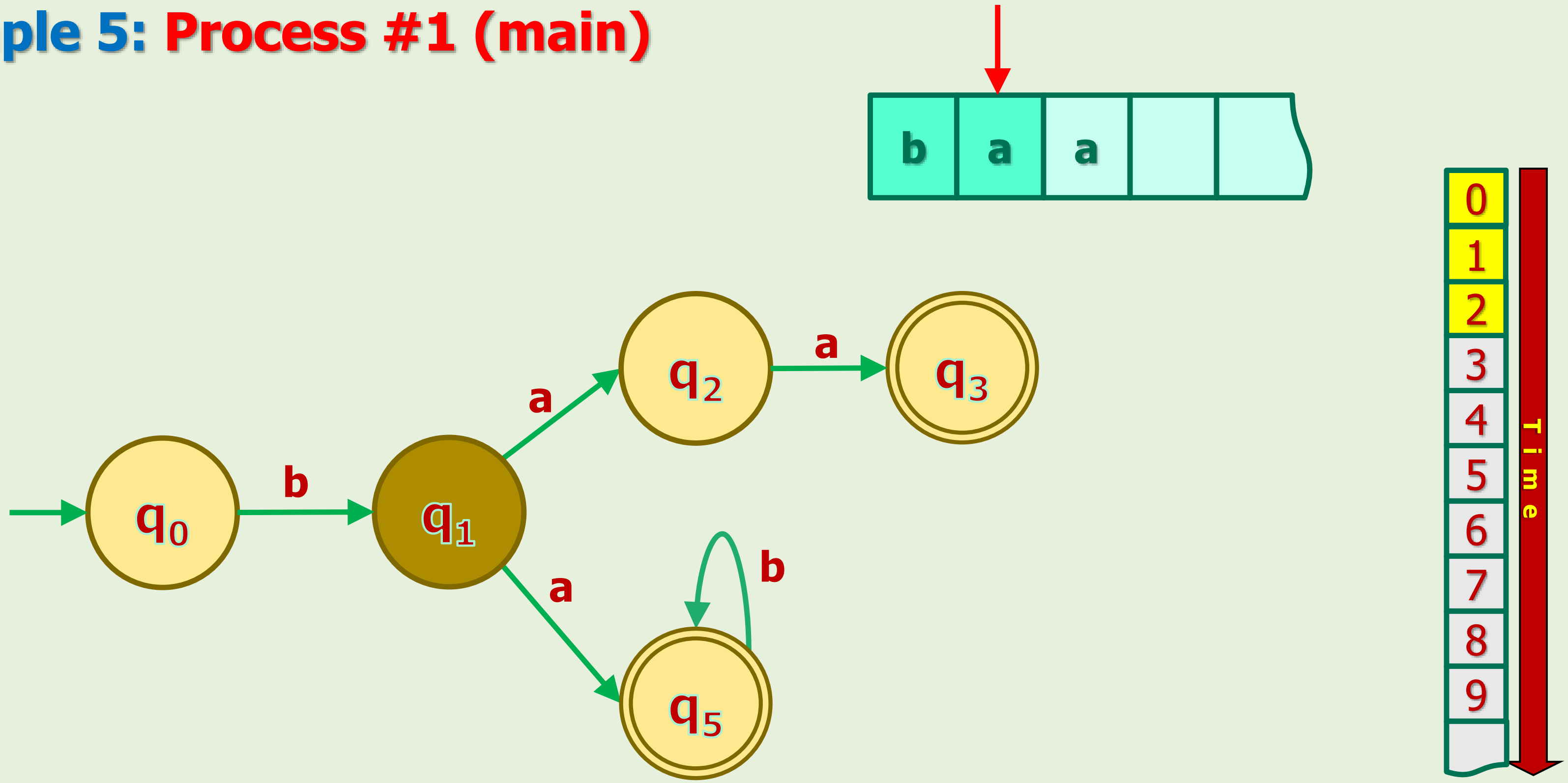
- $\delta(q_0, b) = \{q_1\}$



- 'b' is consumed.
- Control unit transits to q_1.
- This is the end of timeframe 1.
- Up to this point, everything looks like DFAs'.
- What'd happen in the timeframe #2?

5. NFAs in Action

Example 5: Process #1 (main)

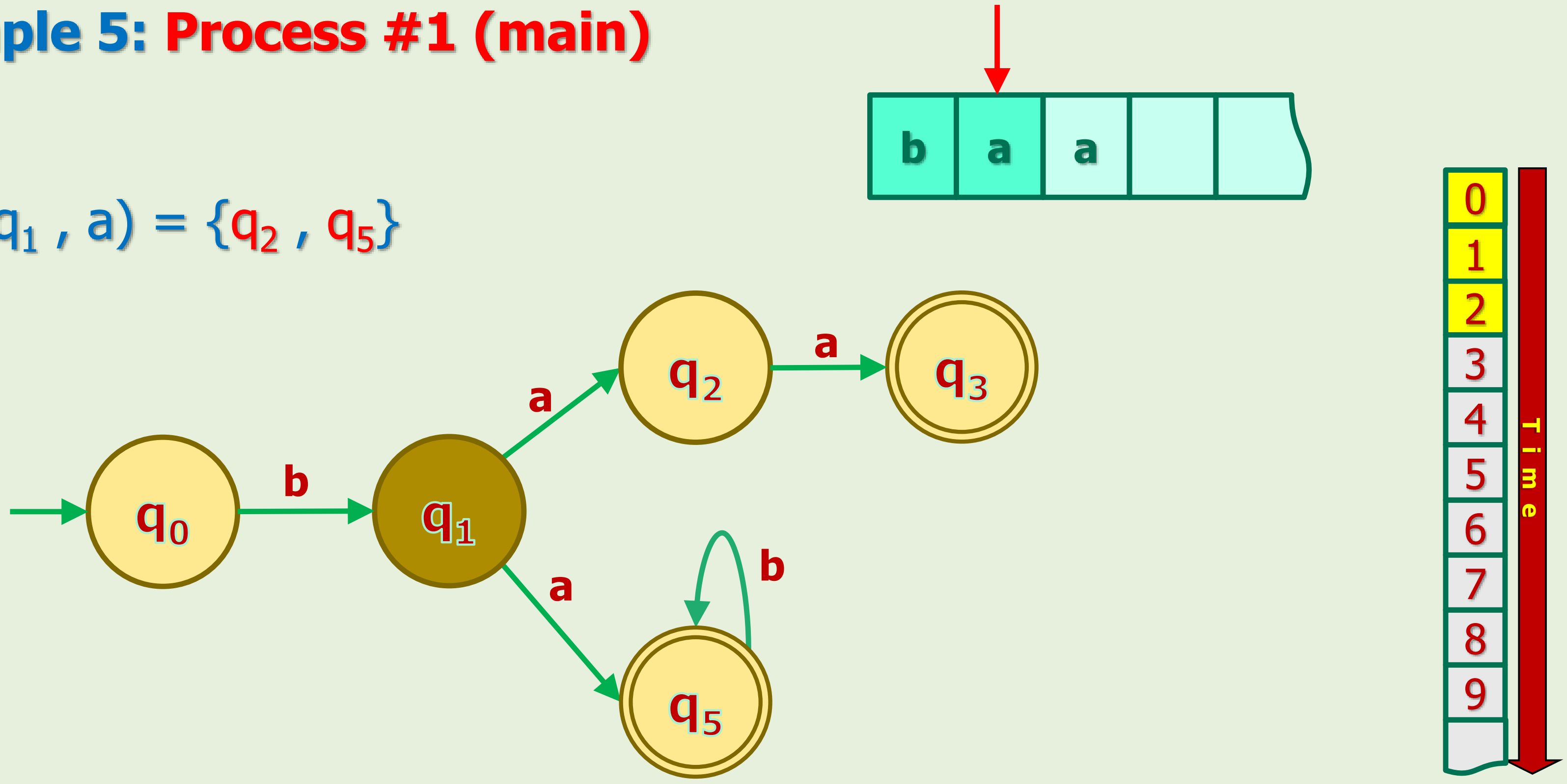


- Input tape reads 'a' and sends it to the control unit.
- The control unit makes a decision based on $\delta(q_1, a) = \{q_2, q_5\}$

5. NFAs in Action

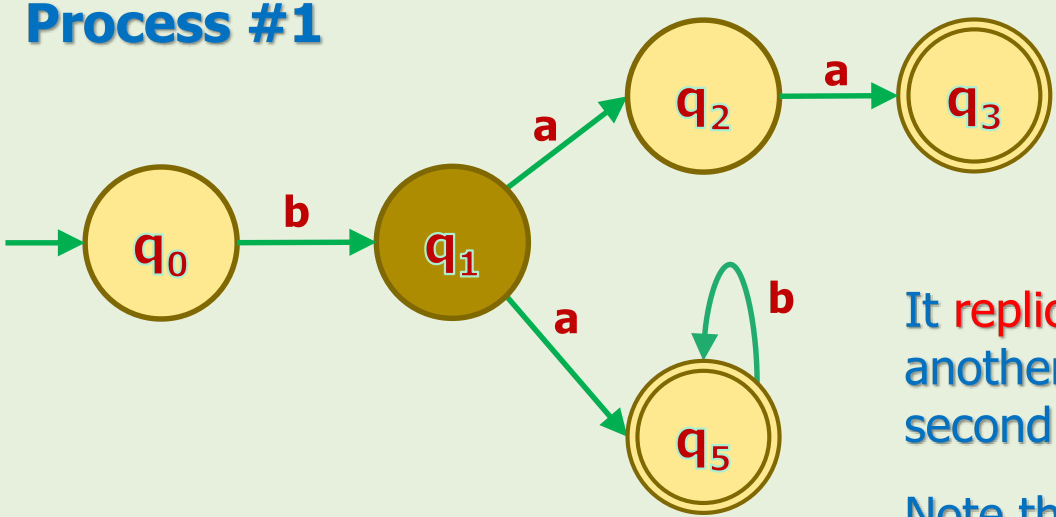
Example 5: Process #1 (main)

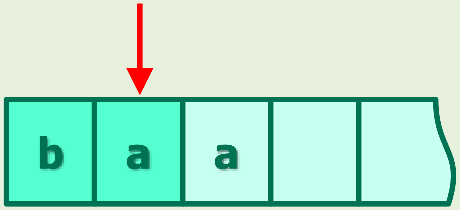
$\delta(q_1, a) = \{q_2, q_5\}$



- It encounters two possibilities: transition to q_2 or q_5.
- So, parallel processing starts!

Process #1

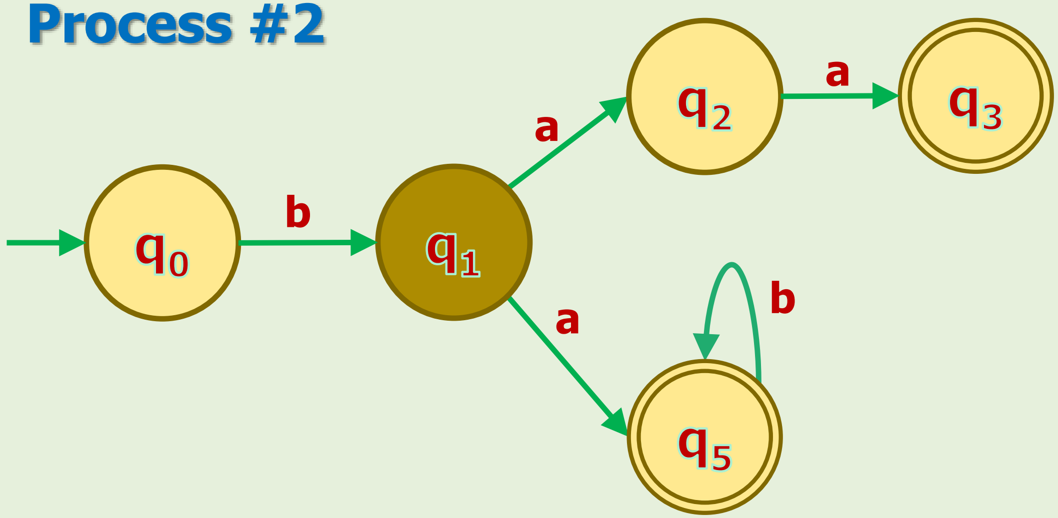


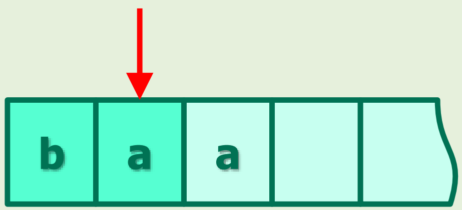


It replicates itself and another process will continue the second possibility.

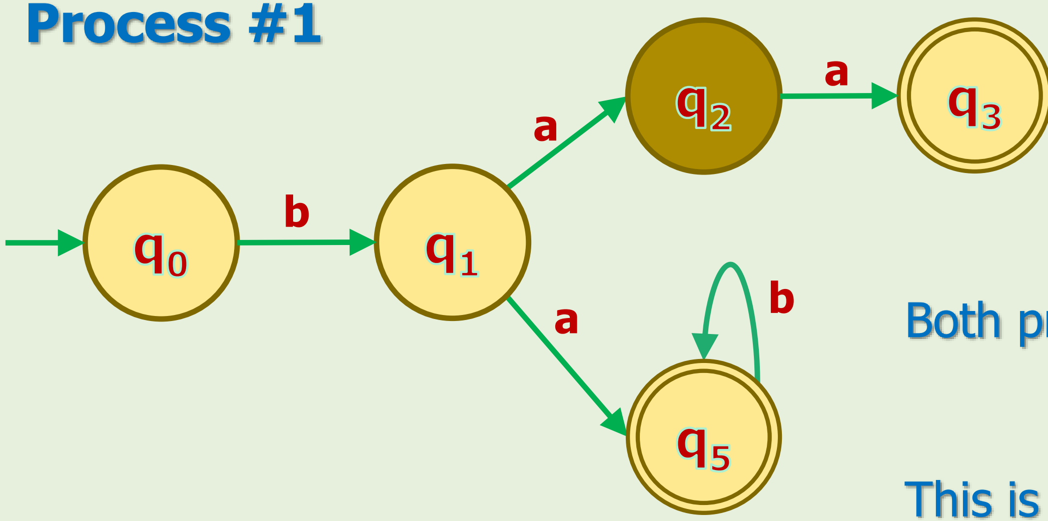
Note that 'a' is not consumed yet!

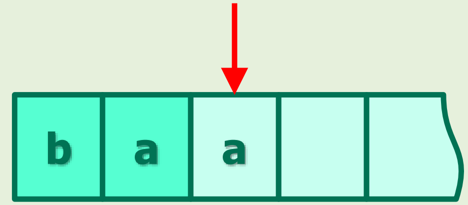
Process #2





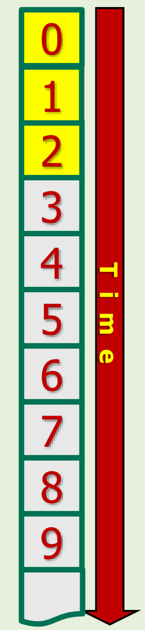
Process #1



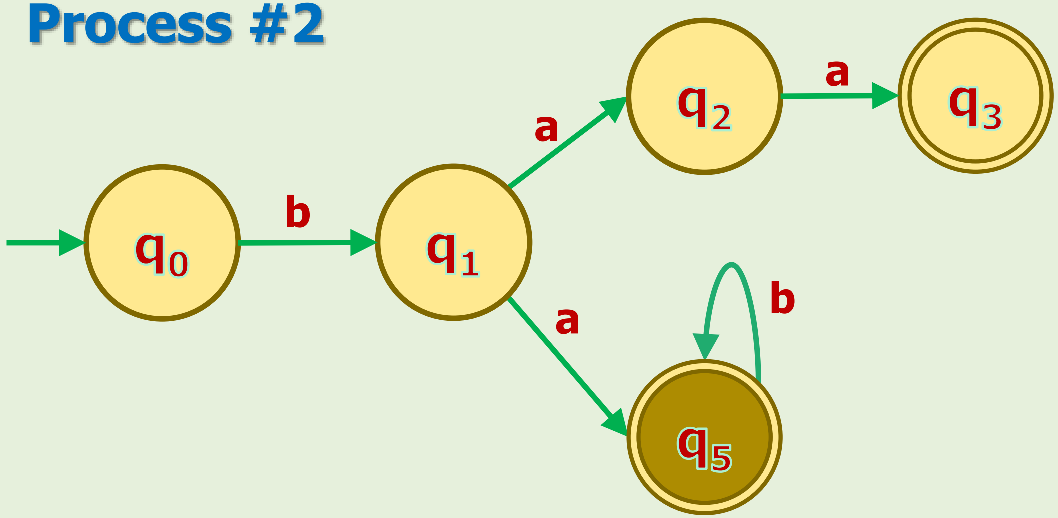


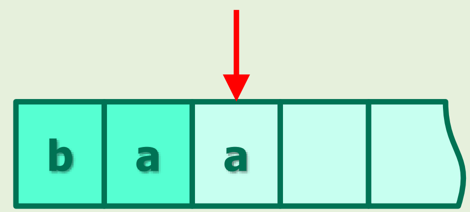
Both processes consume 'a'.

This is the end of timeframe 2.

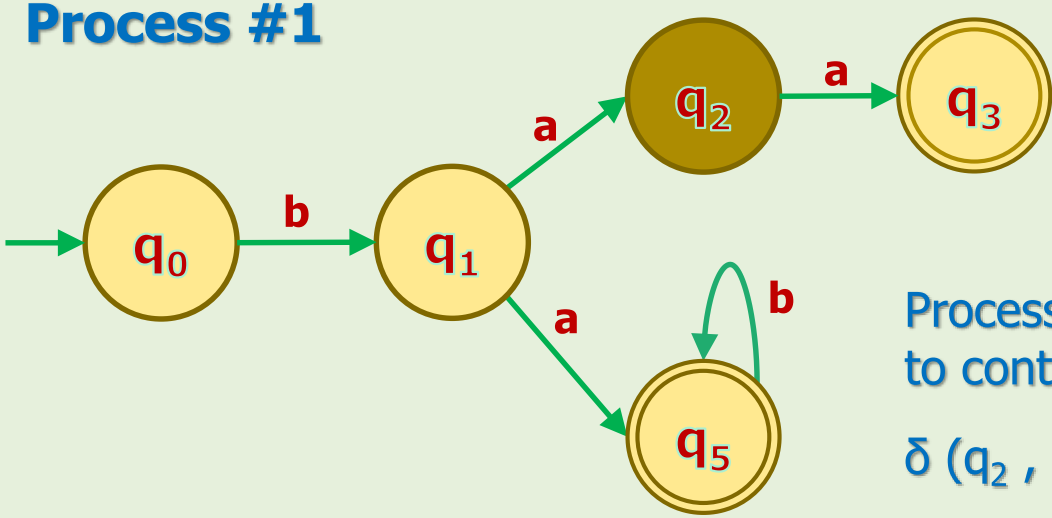


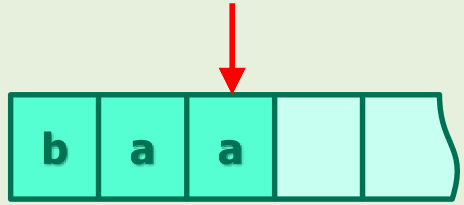
Process #2





Process #1

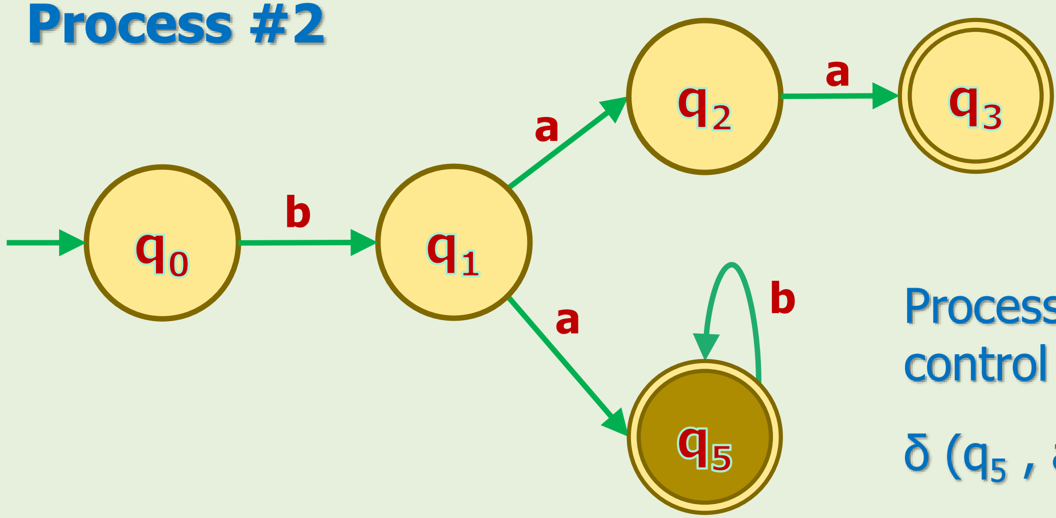


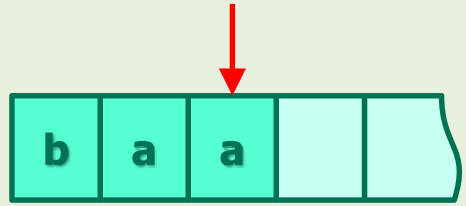


Process #1 reads 'a' and sends it to control unit.

$\delta(q_2, a) = \{q_3\}$

Process #2

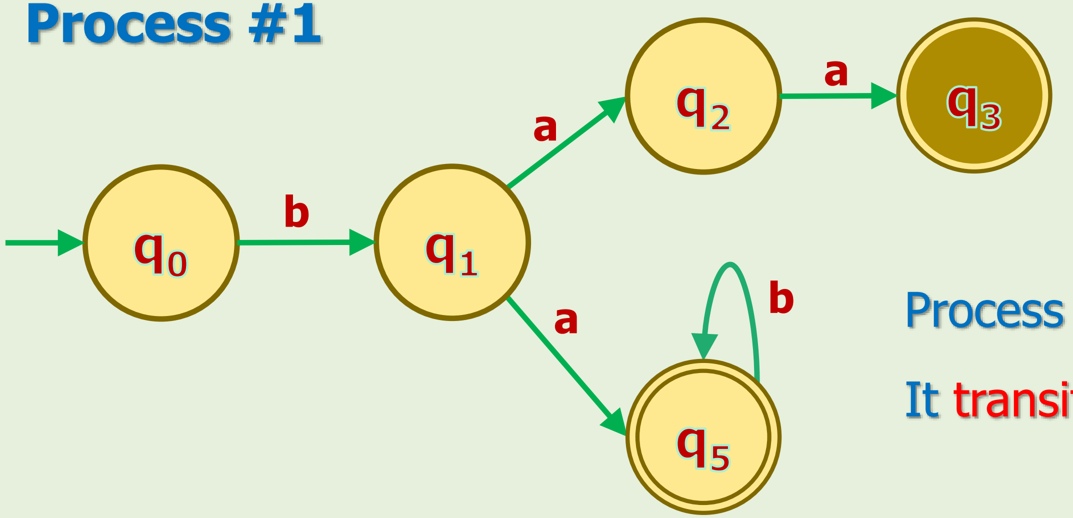


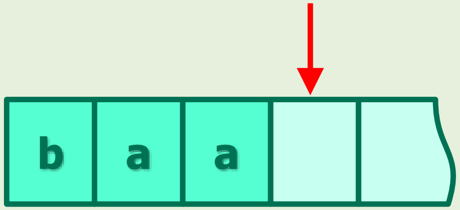


Process #2 reads 'a' and sends it to control unit.

$\delta(q_5, a) = \{\ \}$

Process #1

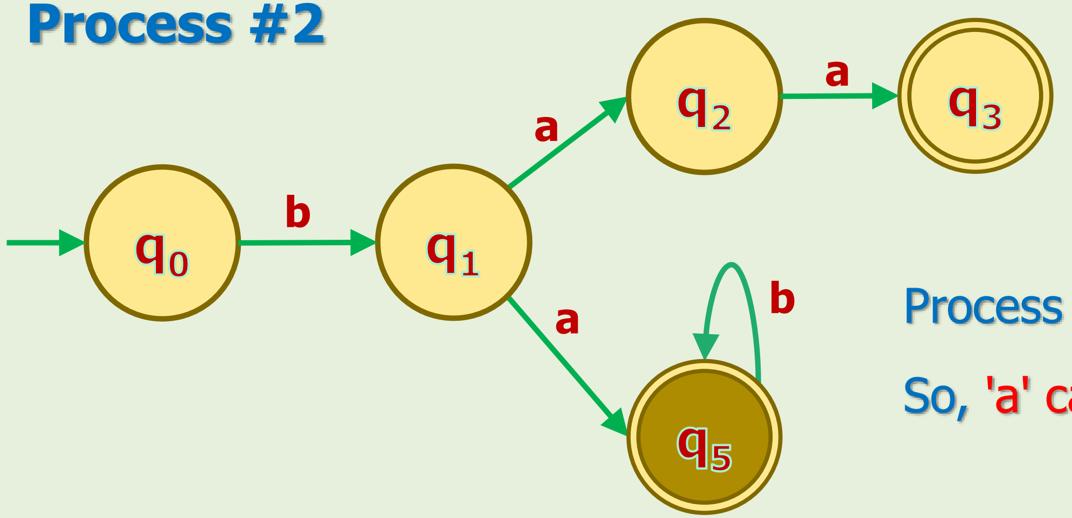


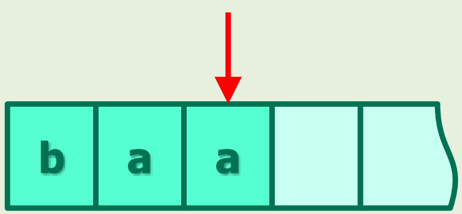


Process #1 consumes 'a'.

It transits to q_3.

Process #2

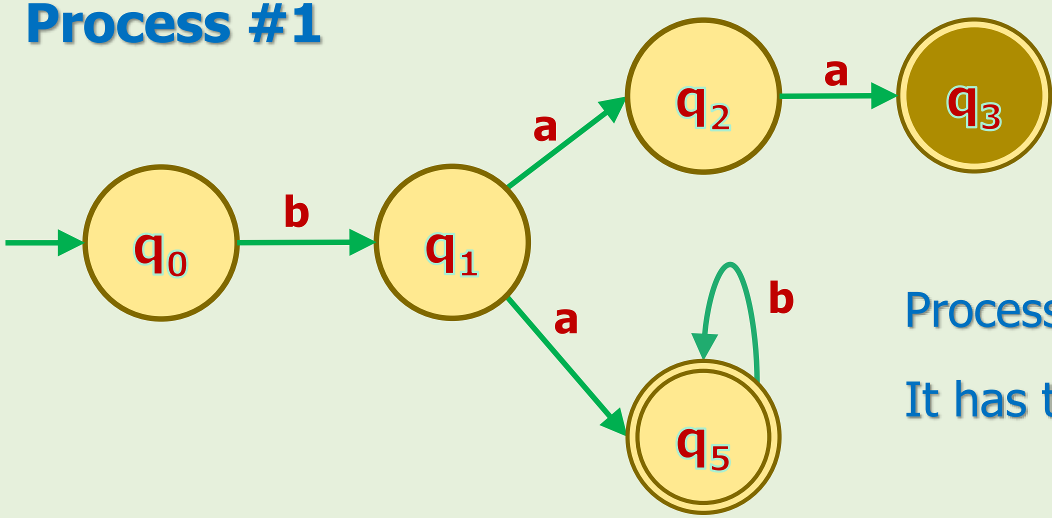


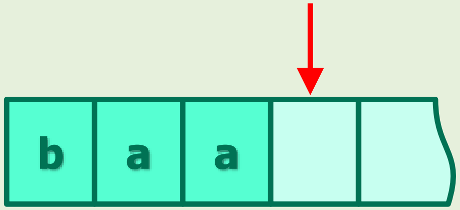


Process #2 has no choice for 'a'.

So, 'a' cannot be consumed.

Process #1

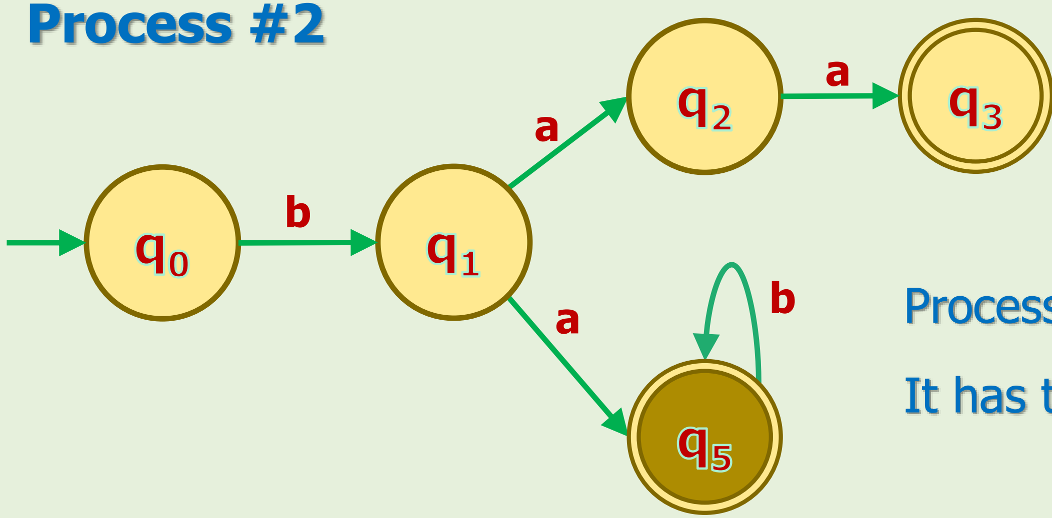


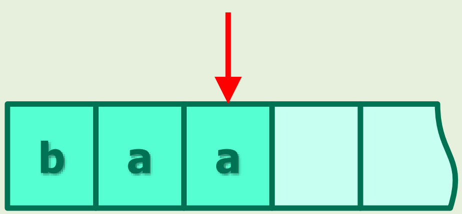


Process #1 is out of symbol.

It has to halt.

Process #2

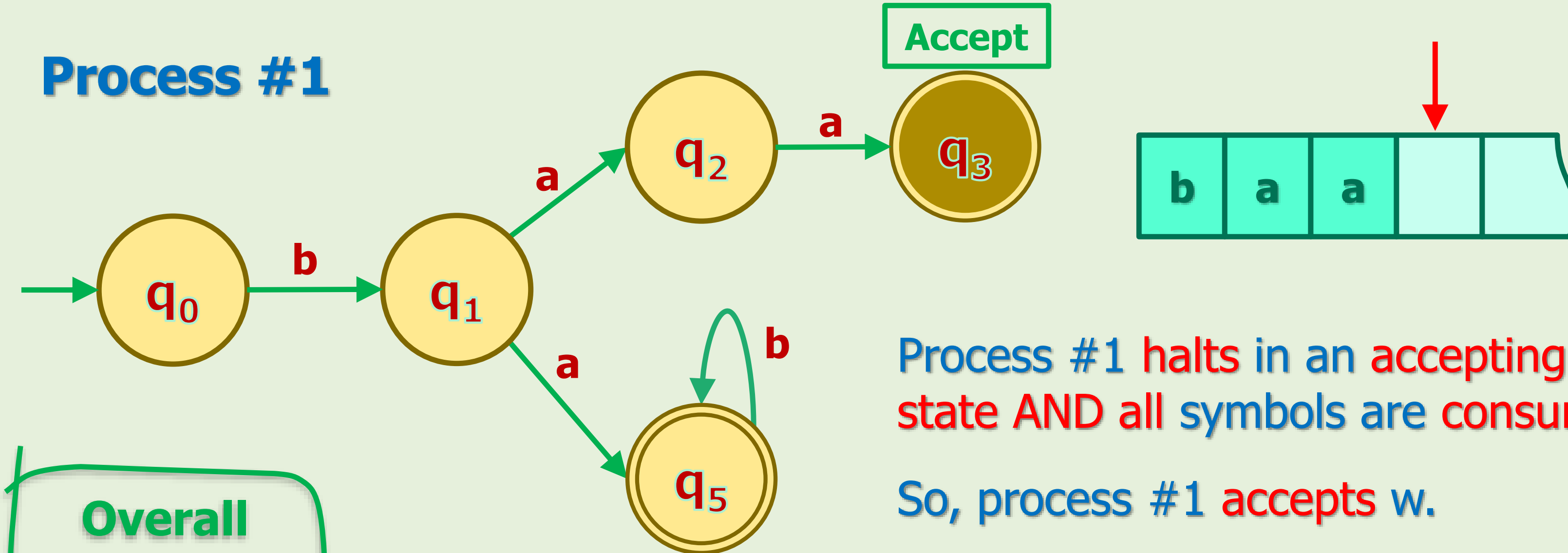




Process #2 has no choice for 'a'.

It has to halt.

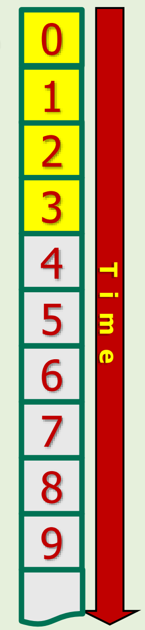
Process #1



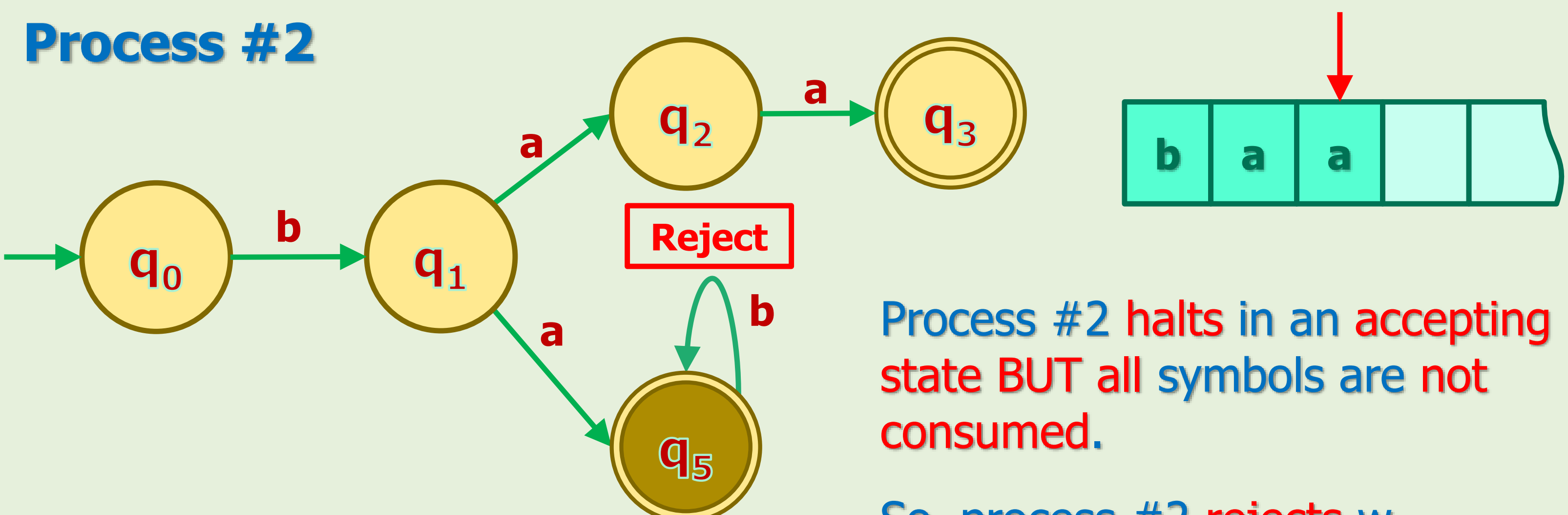
Process #1 halts in an accepting state AND all symbols are consumed.

So, process #1 accepts w.

Overall Accepted



Process #2

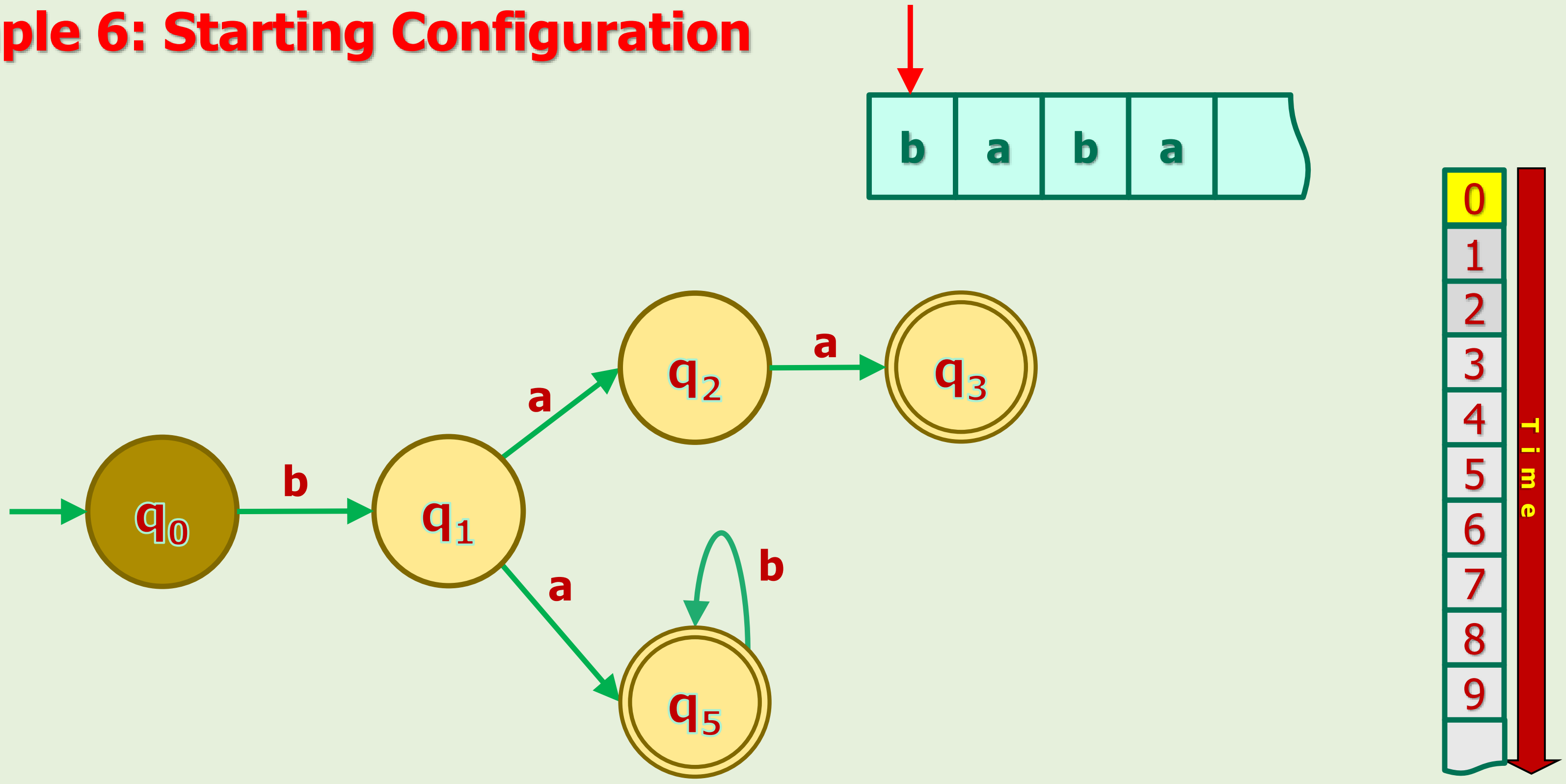


Process #2 halts in an accepting state BUT all symbols are not consumed.

So, process #2 rejects w.

5. NFAs in Action

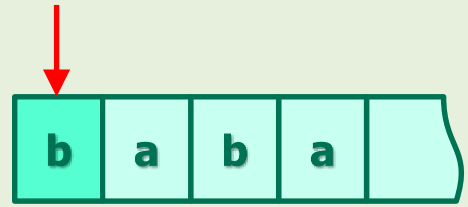
Example 6: Starting Configuration

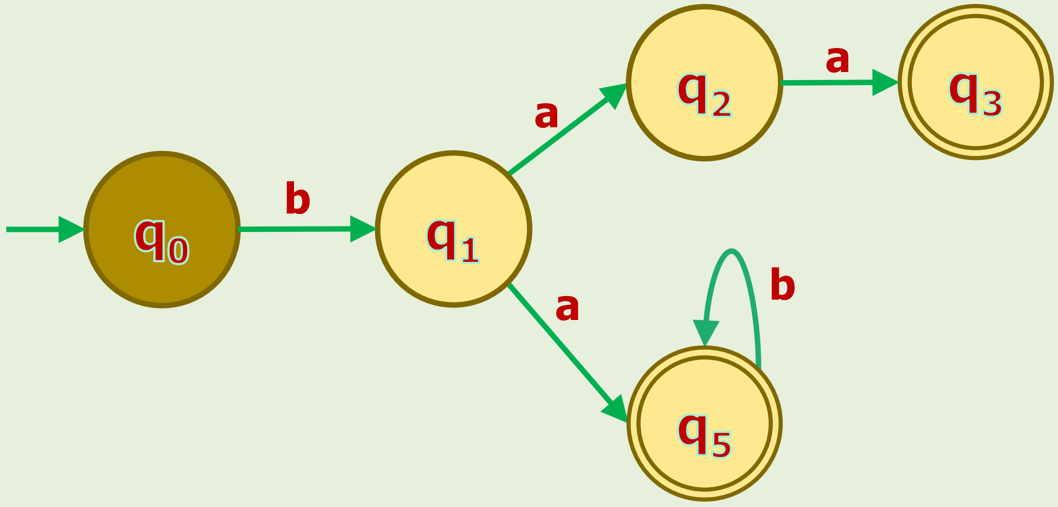


- Process #1 (main) starts normally.

5. NFAs in Action

Example 6: Process #1 (main)

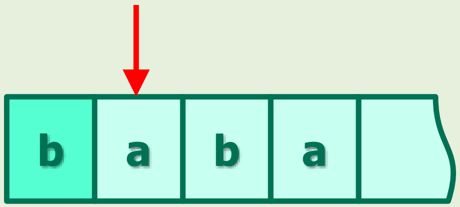




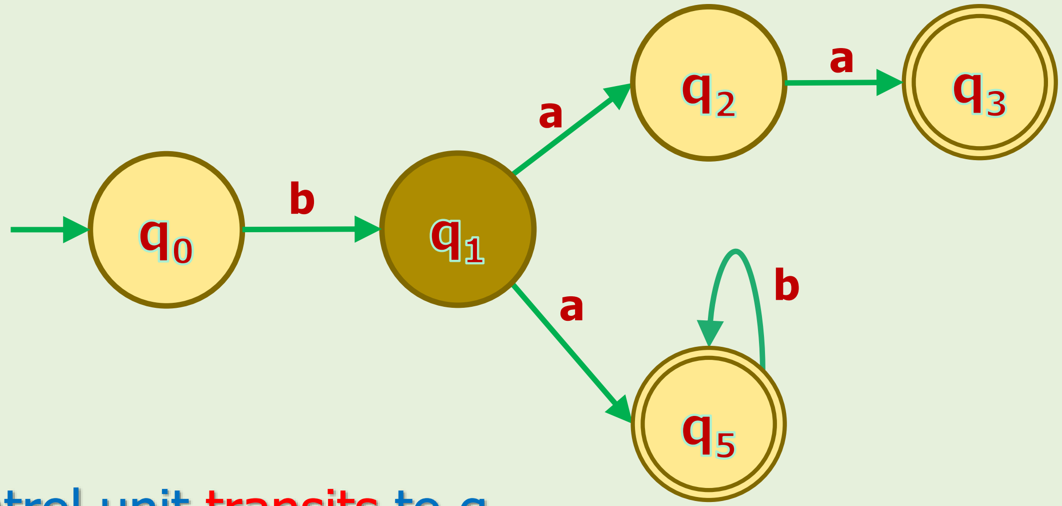
- Input tape reads 'b' and sends it to the control unit.
- The control unit makes a decision based on $\delta(q_0, b) = \{q_1\}$

5. NFAs in Action

Example 6: Process #1 (main)



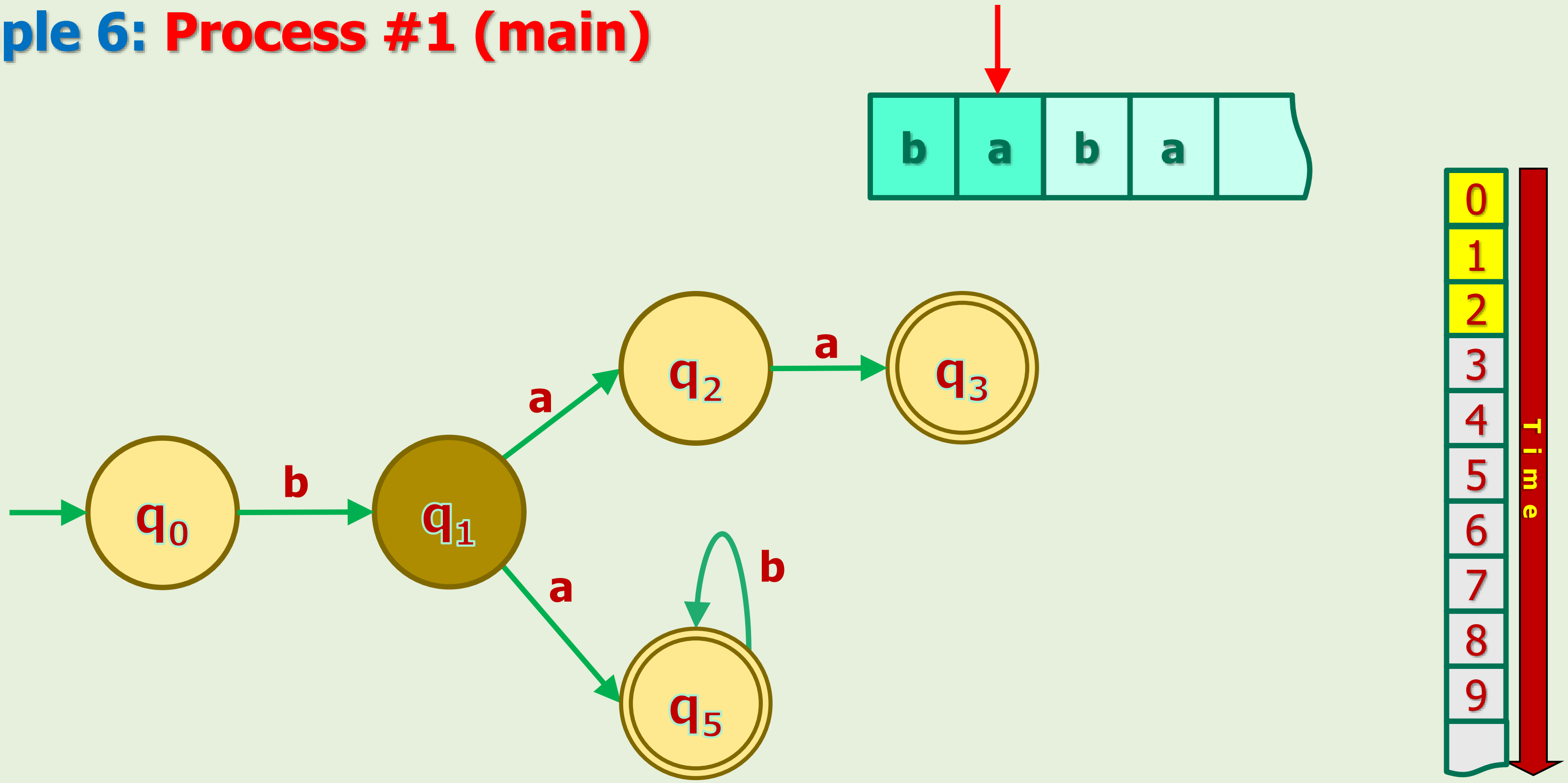
- $\delta(q_0, b) = \{q_1\}$



- Control unit transits to q_1.
- This is the end of timeframe 1.
- Up to this point, everything looks like DFAs'.
- What'd happen in the timeframe #2?

5. NFAs in Action

Example 6: Process #1 (main)

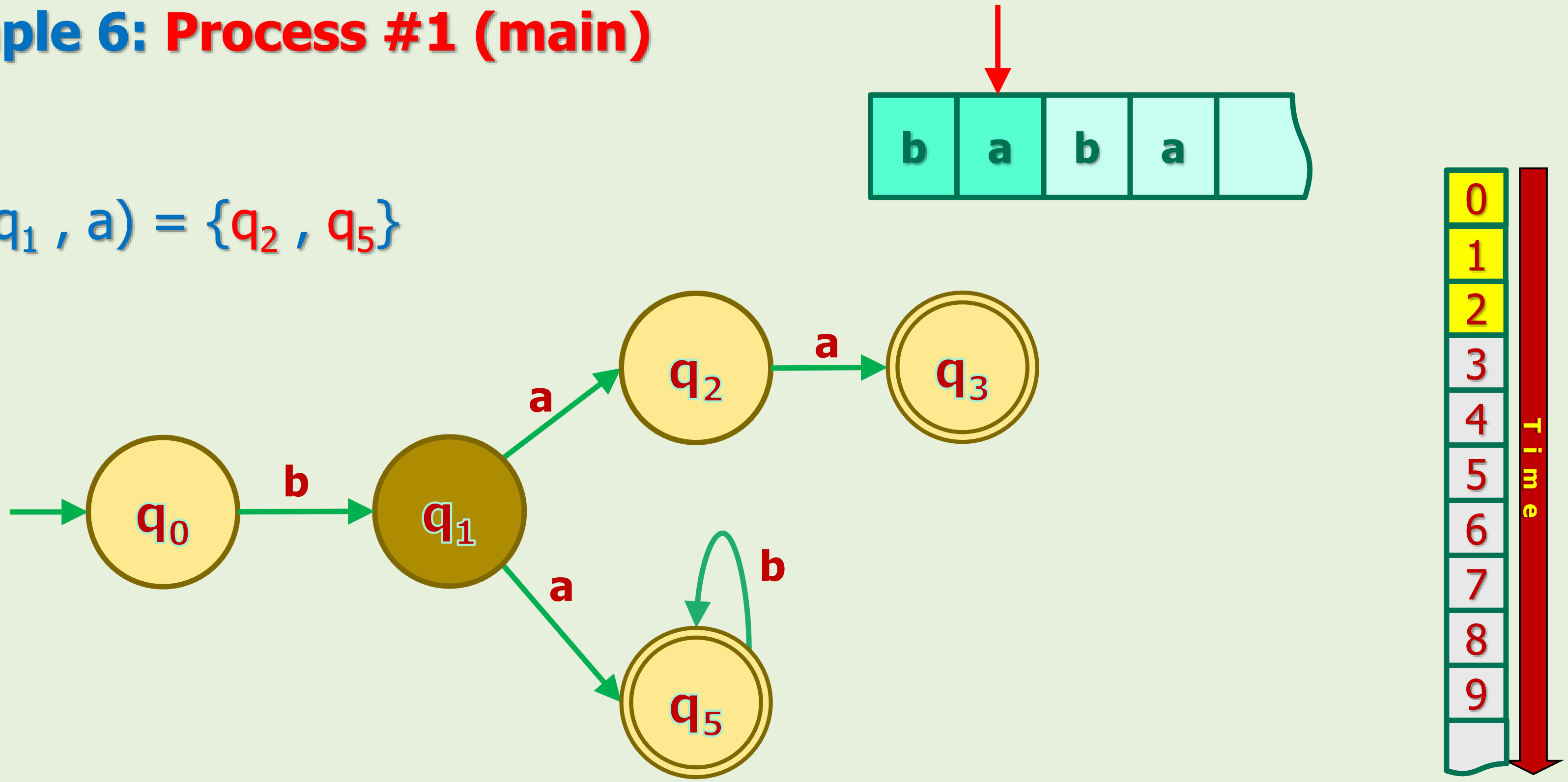


- Input tape reads 'a' and sends it to the control unit.
- The control unit makes a decision based on $\delta\ (q_1\ ,\ a) = \{q_2\ ,\ q_5\}$

5. NFAs in Action

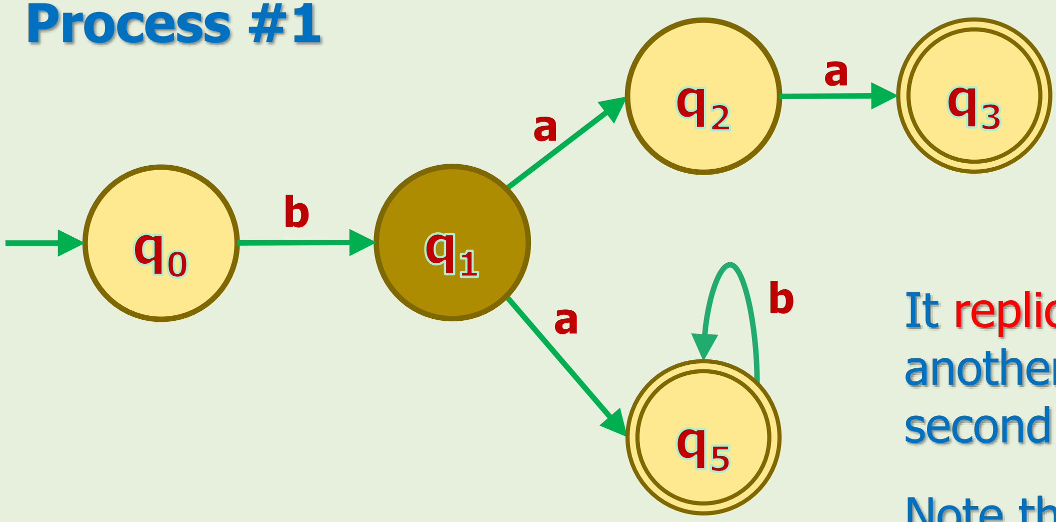
Example 6: Process #1 (main)

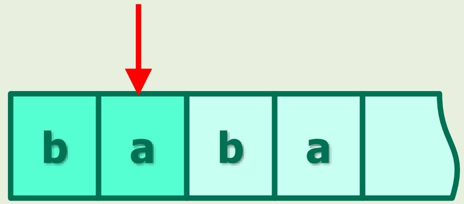
$$\delta(q_1, a) = \{q_2, q_5\}$$



- It encounters two possibilities: transition to q_2 or q_5.
- So, parallel processing starts!

Process #1

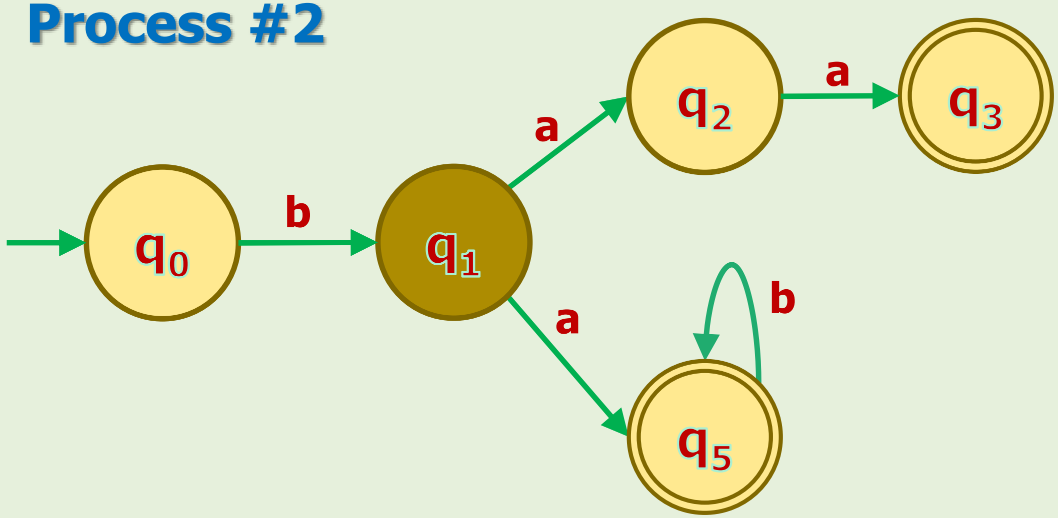


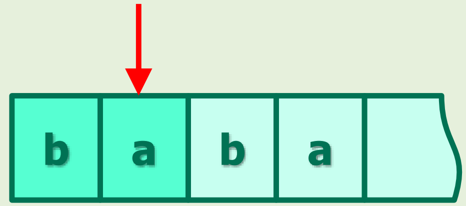


It replicates itself and another process will continue the second possibility.

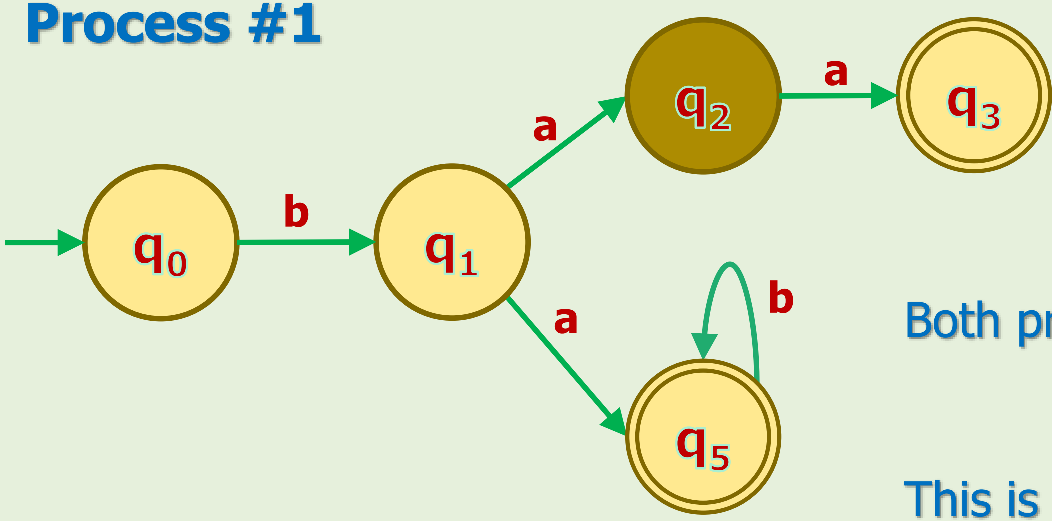
Note that 'a' is not consumed yet!

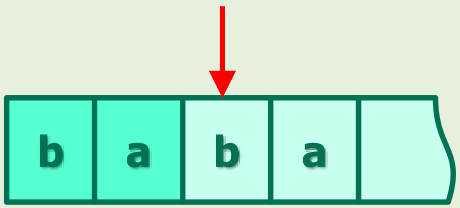
Process #2





Process #1

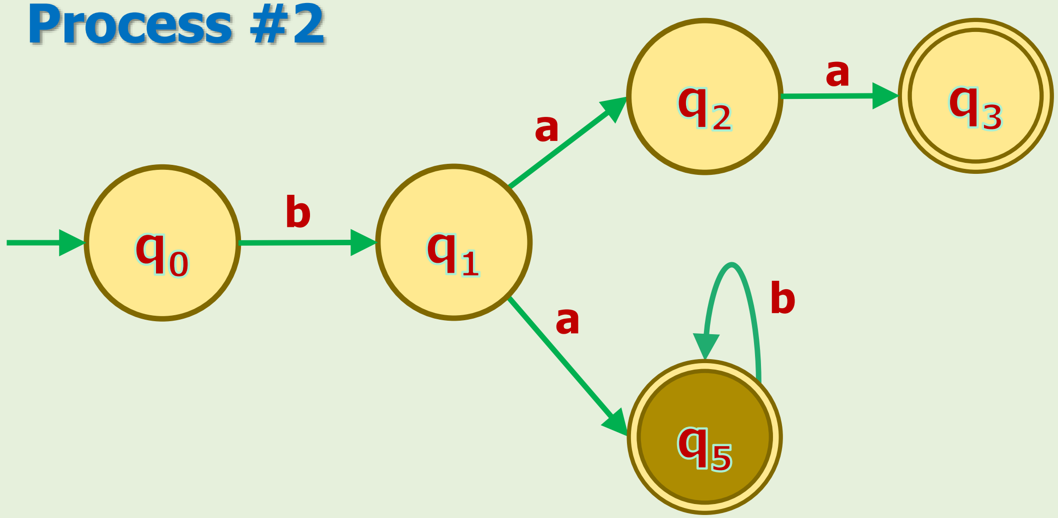


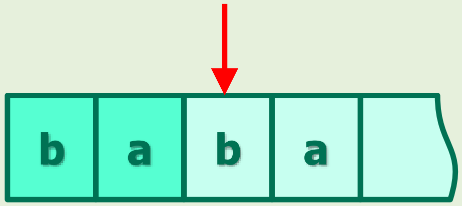


Both processes consume 'a'.

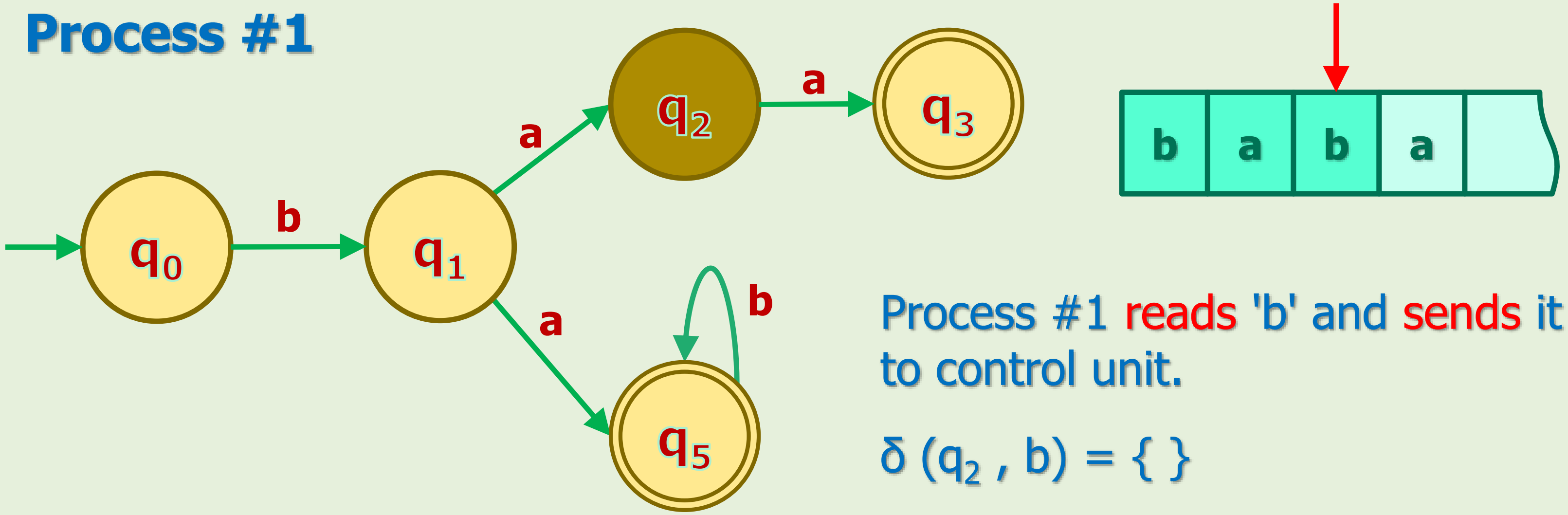
This is the end of timeframe 2.

Process #2





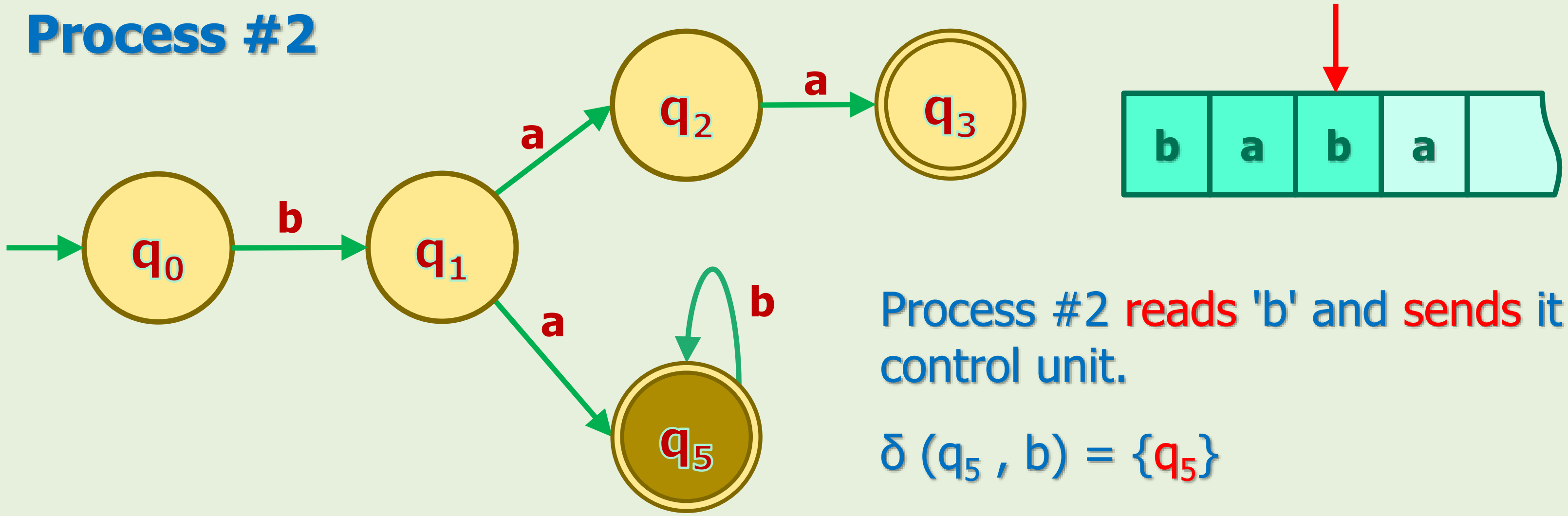
Process #1



Process #1 reads 'b' and sends it to control unit.

$\delta\,(q_2\,,\,b) = \{\ \}$

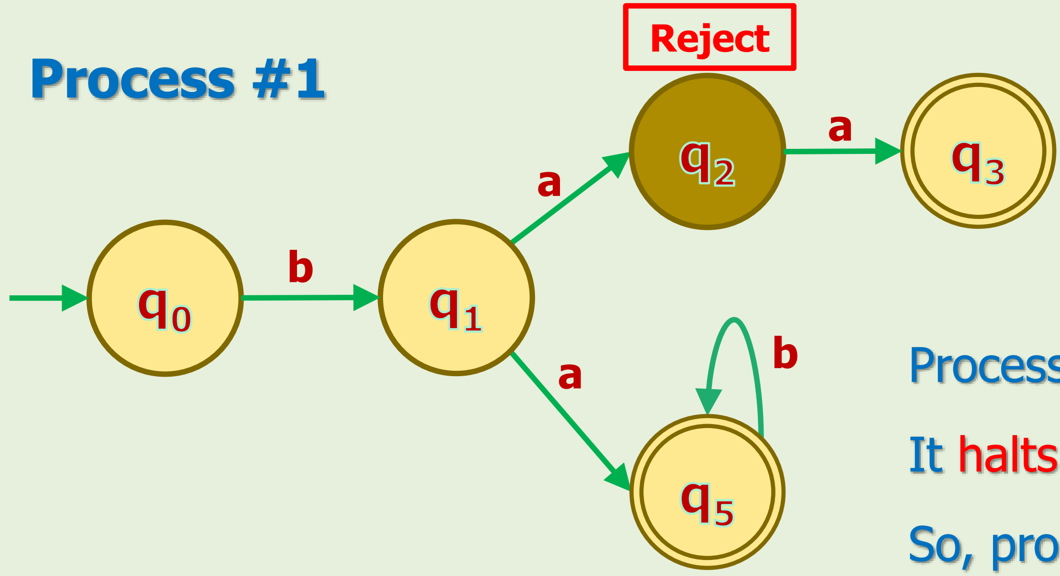
Process #2

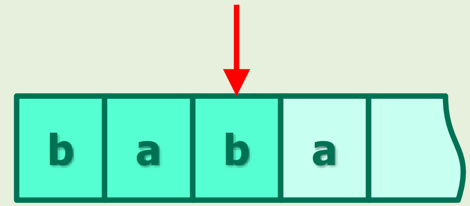


Process #2 reads 'b' and sends it to control unit.

$\delta\,(q_5\,,\,b) = \{q_5\}$

Process #1

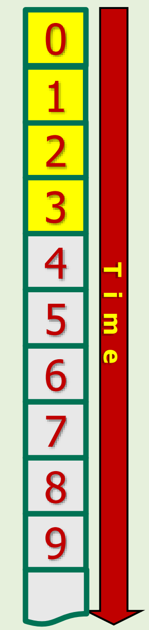




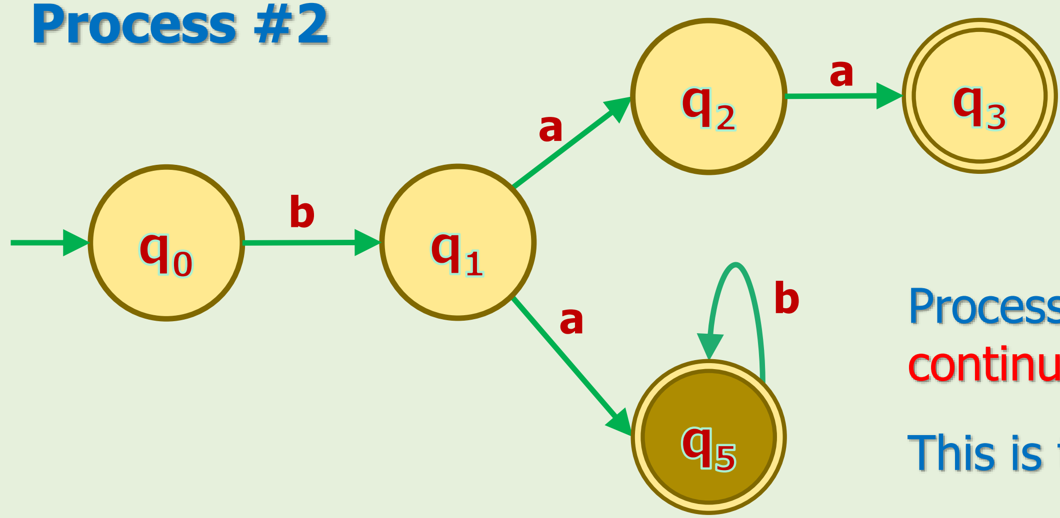
Process #1 has no choice for 'b'.

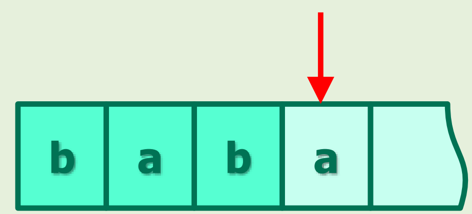
It halts in a non-accepting state.

So, process #1 rejects w.



Process #2

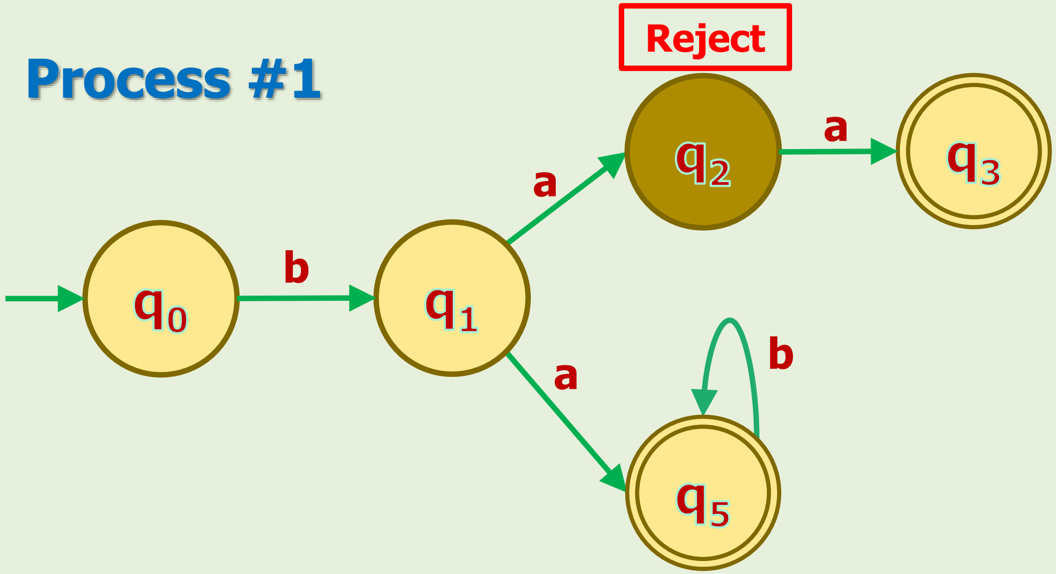


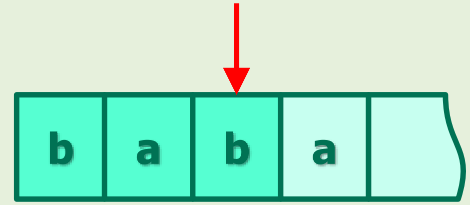


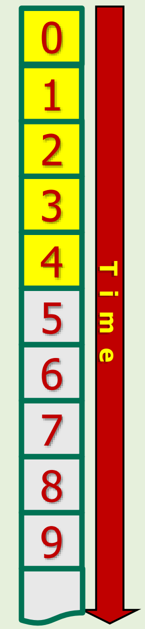
Process #2 consumed 'b' and continues.

This is the end of timeframe 3.

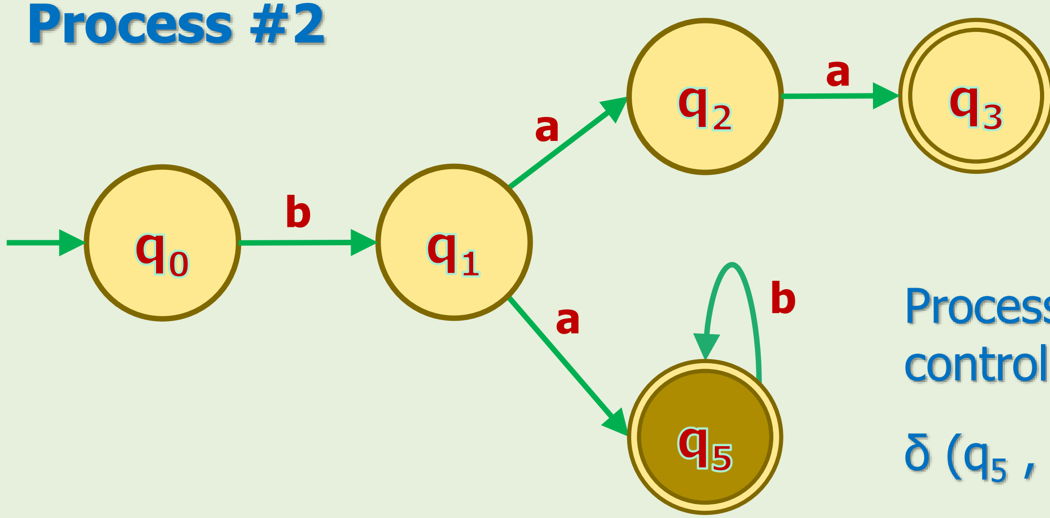
Process #1

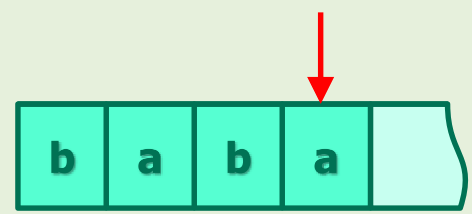






Process #2

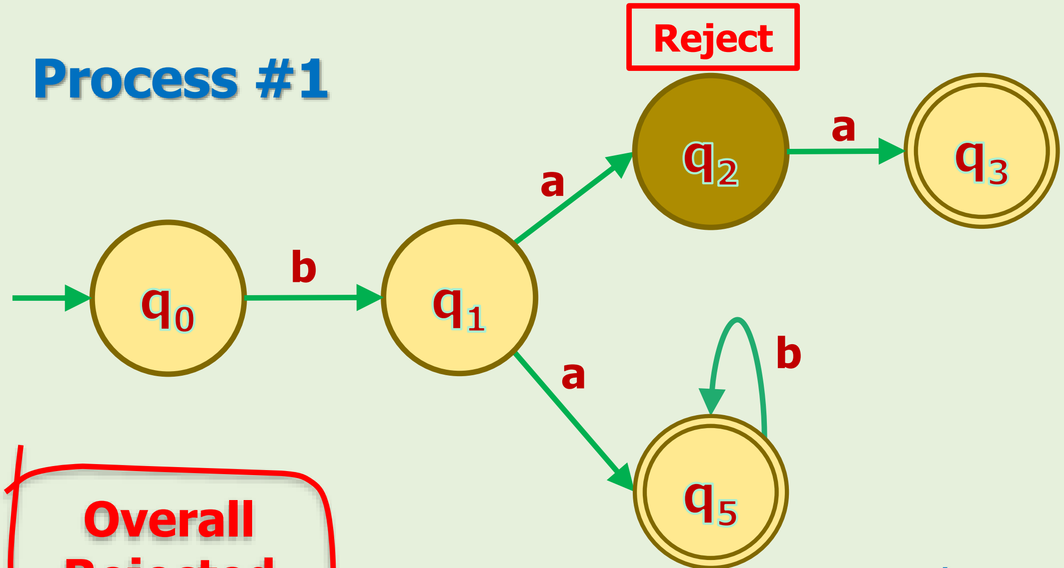


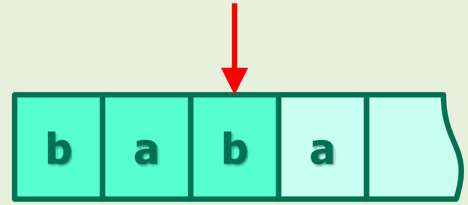


Process #2 reads 'a' and sends it to control unit.

$\delta\ (q_5\ ,\ a) = \{\ \}$

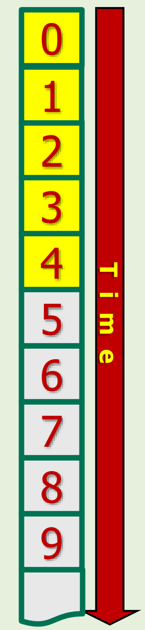
Process #1



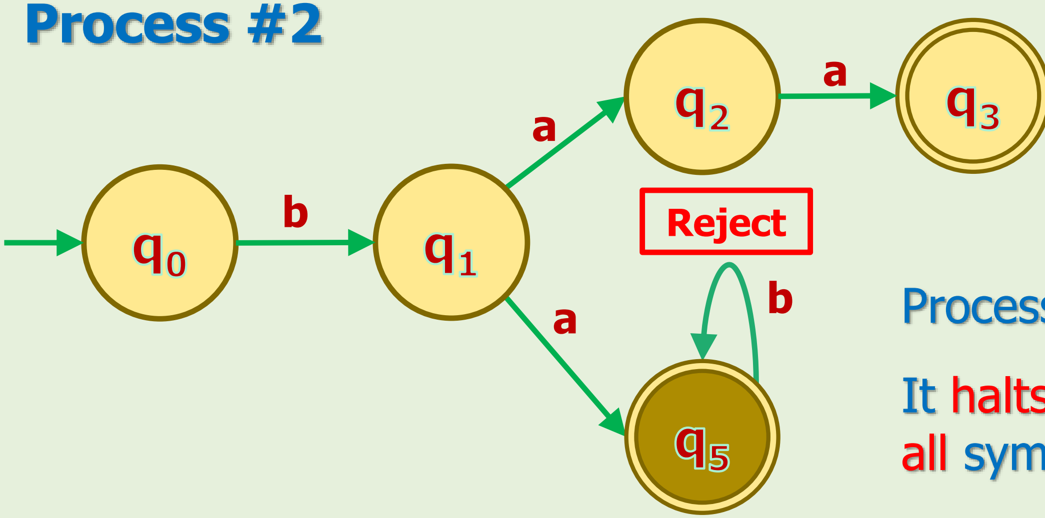


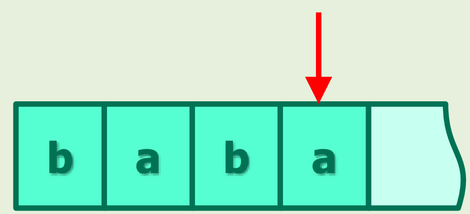
Overall Rejected

This is the end of timeframe 4.



Process #2





Process #2 has no choice for 'a'.

It halts in an accepting state BUT all symbols are not consumed.

So, process #2 rejects w.

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