San José State University
Department of Computer Science

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

Nondeterministic Finite Automata

(Part 2)

Lecture 10

Day 10/31

CS 154

Formal Languages and Computability

Spring 2019

Agenda of Day 10

- About Midterm 1
- Feedback and Solution of Quiz 2 and 3
- Summary of Lecture 09
- Lecture 10: Teaching ...
  - Nondeterministic Finite Automata (Part 2)

About Midterm 1

Reminder 2

- Midterm #1 (aka Quiz+)
  - Date: Thursday 02/28
  - Value: 10%
  - Topics: Everything covered from the beginning of the semester
  - Type: Closed y ∈ Material

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

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

Study Guide

- I've overviewed the type and number of questions via Canvas.

Solution and Feedback of Quiz 2 (Out of 30)

Section	Average	High Score	Low Score
01 (TR 3:00 PM)	26.1	30	12
02 (TR 4:30 PM)	26.57	30	13
03 (TR 6:00 PM)	27.18	30	21

Solution and Feedback of Quiz 3 (Out of 13)

Section	Average	High Score	Low Score
01 (TR 3:00 PM)	10.26	13	6
02 (TR 4:30 PM)	9.5	13	6
03 (TR 6:00 PM)	10.45	13	7

Summary of Lecture 09: We learned ...

DFAs

- Associated language to a DFA M is ...
  - ... the set of all strings that it accepts.
  - ... denoted by L(M).
- Two machines are equivalent iff ...
  - ... their associated languages are equal.
- Computation is ...
  - ... the sequence of configurations from when the machine starts until it halts.

- A machine is called deterministic iff ...
  - ... during any timeframe, there is NO MORE THAN ONE transition.

Any question?

Summary of Lecture 09: We learned ...

NFAs

- Two violations in DFAs were introduced...
- Violation #1
  During a timeframe, the machine has no (zero) transition.
  - The transition function is partial function.
- Violation #2
  During a timeframe, the machine has more than one transition.
  - The transition function is a multifunction.

Any question?

Let's Construct a New Class of Automata

Template for Introducing a New Class of Automata

Repeated

- To construct a new class of automata, we need to respond the following questions:

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

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

- 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 the New Class

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

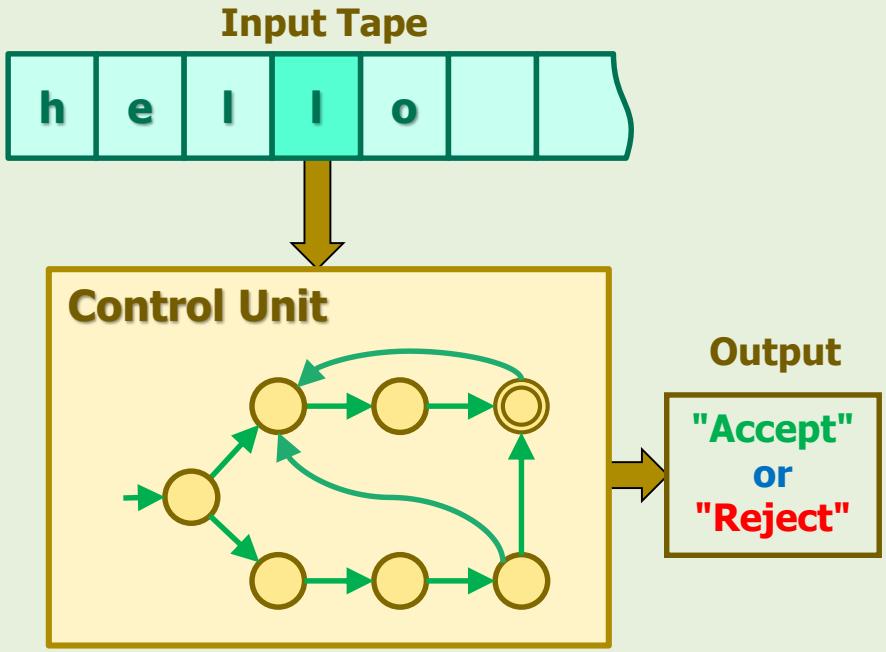
1. The second violation we mentioned earlier violates the definition of determinism.
   - In other words, during any timeframe, there might be more than one transitions.
   - So, this class should be "nondeterministic".
2. The number of states is still "finite".

- 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 NFAs Work

4. How NFAs Work

- To understand how NFAs work, 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?

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

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

4. How NFAs Work

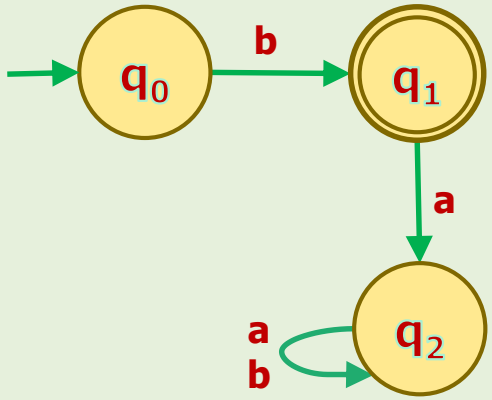
- DFAs' and NFAs' have the same building blocks.
- So, we expect their behavior be the same except ...
- **... for those two violations.**

- Therefore, we just need to know ...
  what NFAs would do when they encounter those two violations.
- And the rest would be exactly the same as 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

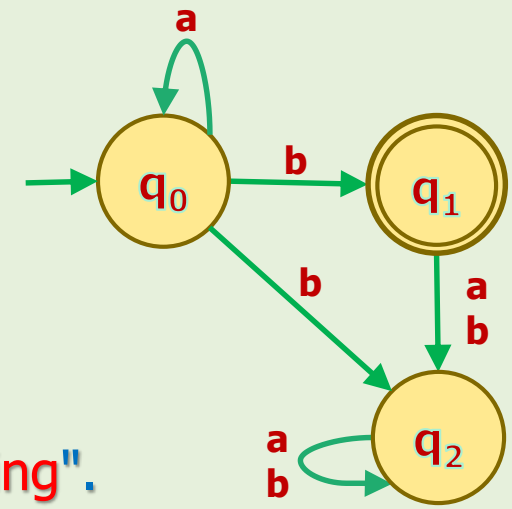
  They have zero transition. ≡ **z**

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

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 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

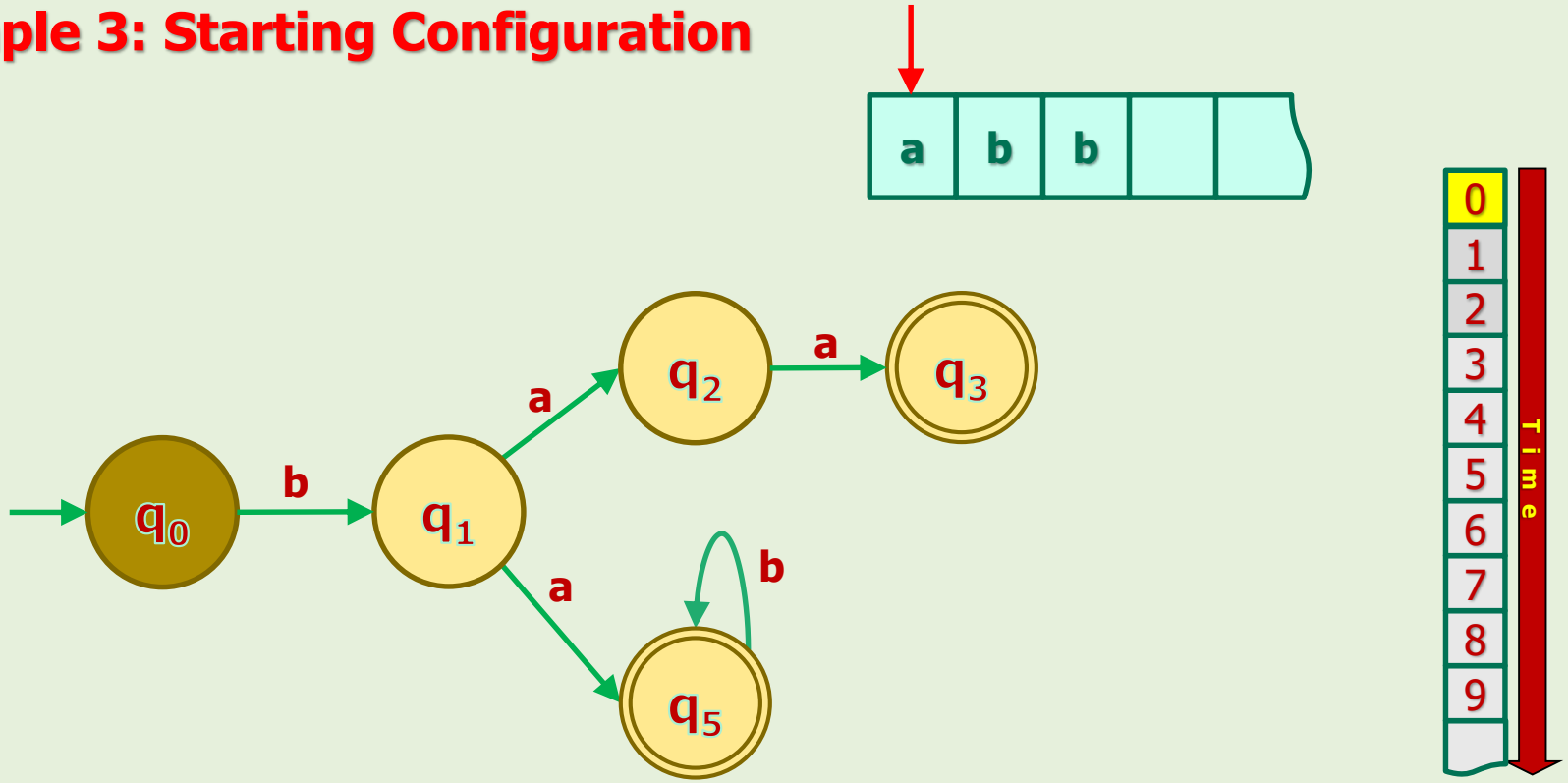
Review of NFAs' Input Tapes

- 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

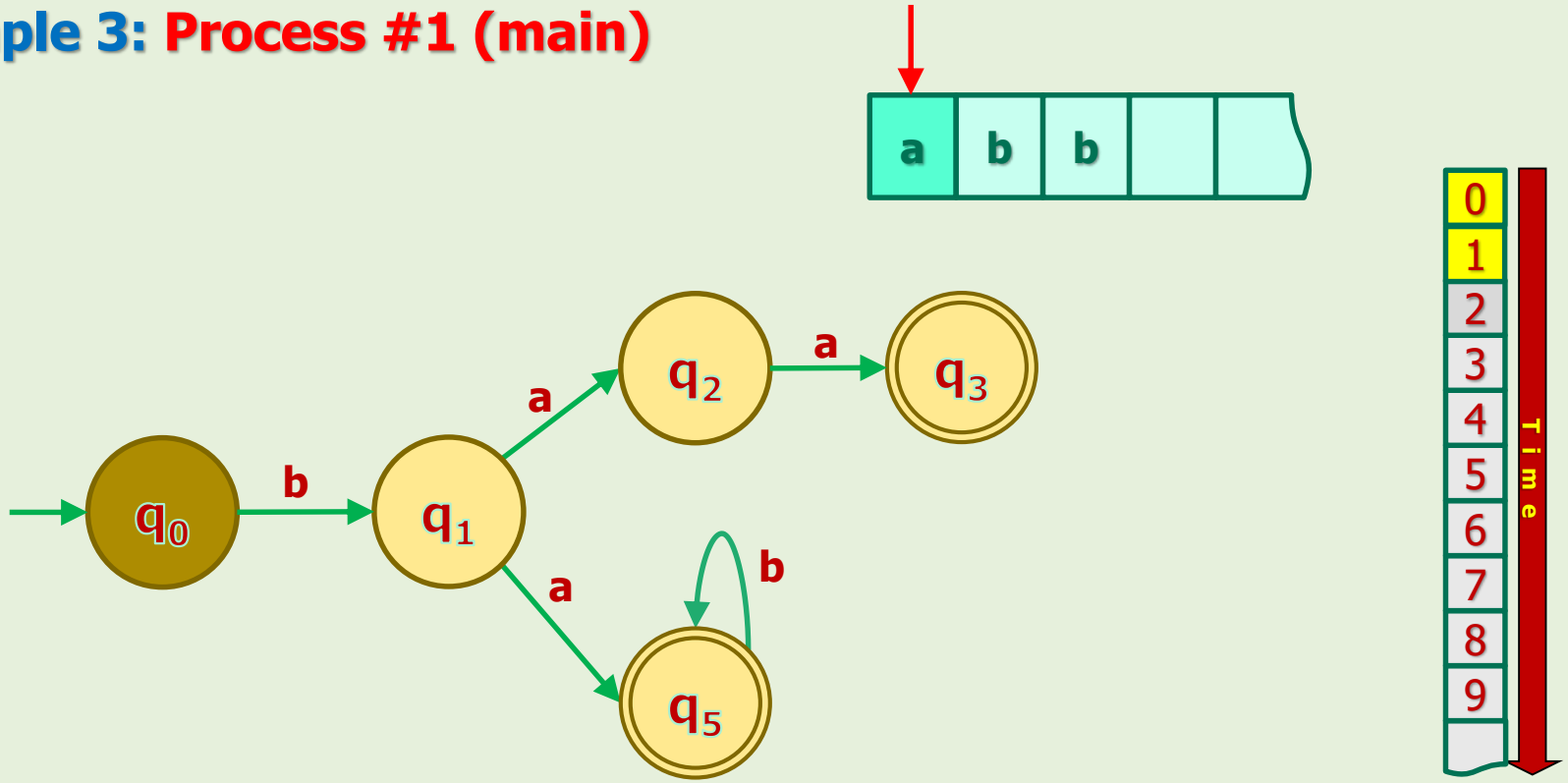
Example 3: Starting Configuration



- Process #1 (main) starts normally.

5. NFAs in Action

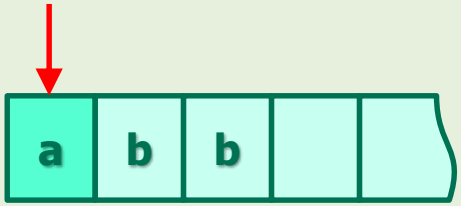
Example 3: Process #1 (main)



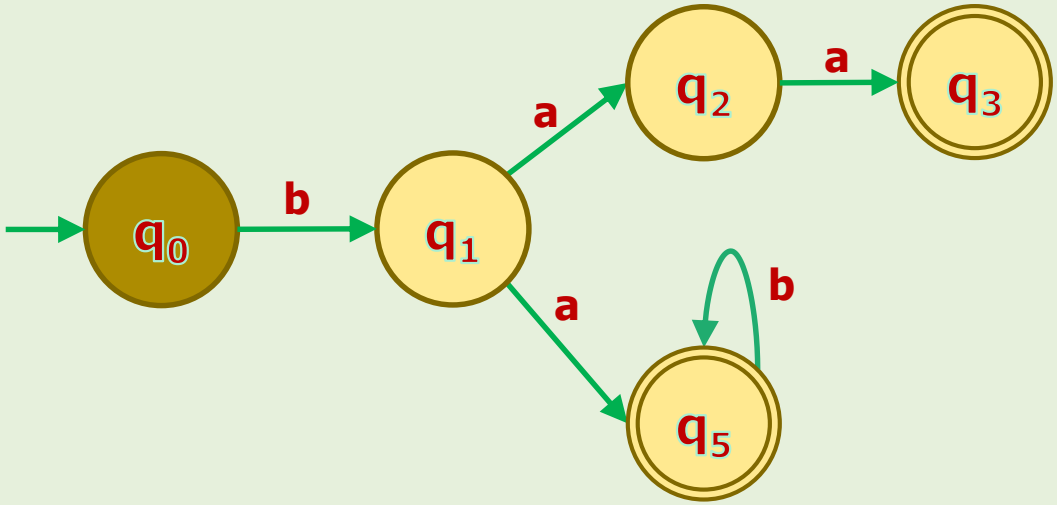
- Input tape reads 'a' and sends it to the control unit.
- The control unit makes a decision based on $\delta(q_0, a) = \{\}$

5. NFAs in Action

Example 3: Process #1 (main)



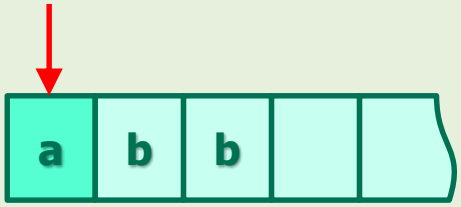
$\delta(q_0, a) = \{\ \}$



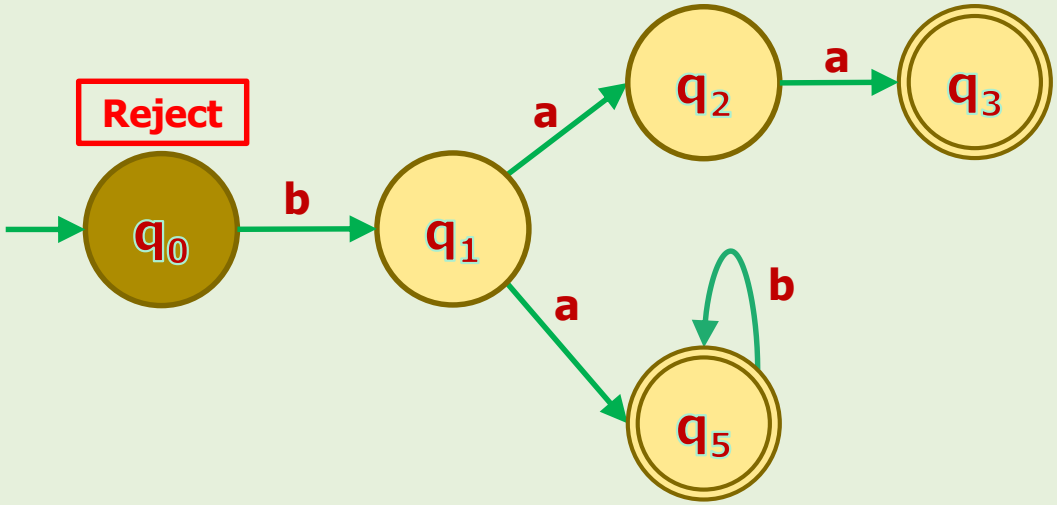
- The control unit cannot consume it because it has no choice for 'a'.
- The head DOES NOT move because control unit did not consume it.

5. NFAs in Action

Example 3: Process #1 (main)



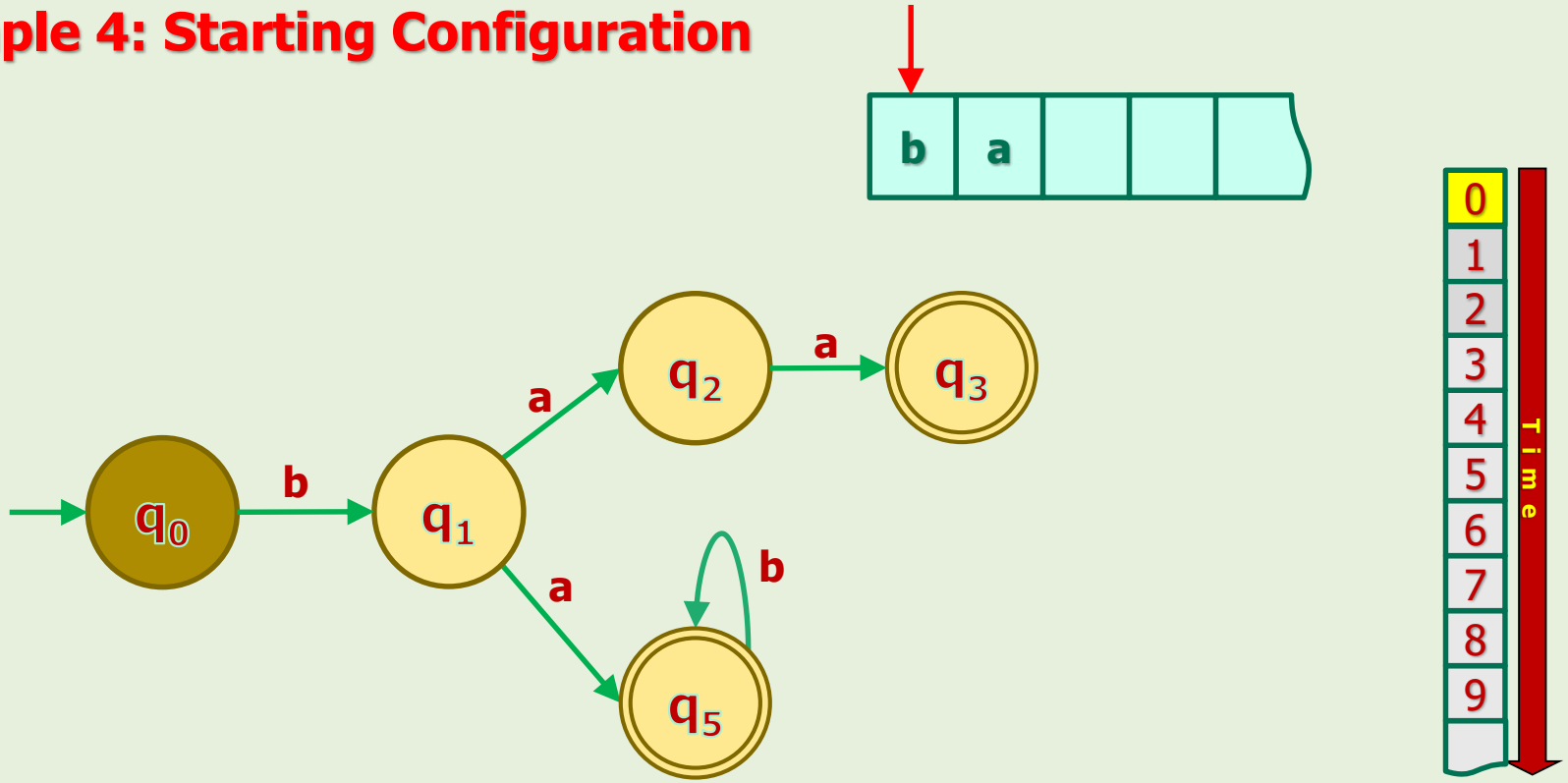
$\delta(q_0, a) = \{\ \}$



- 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

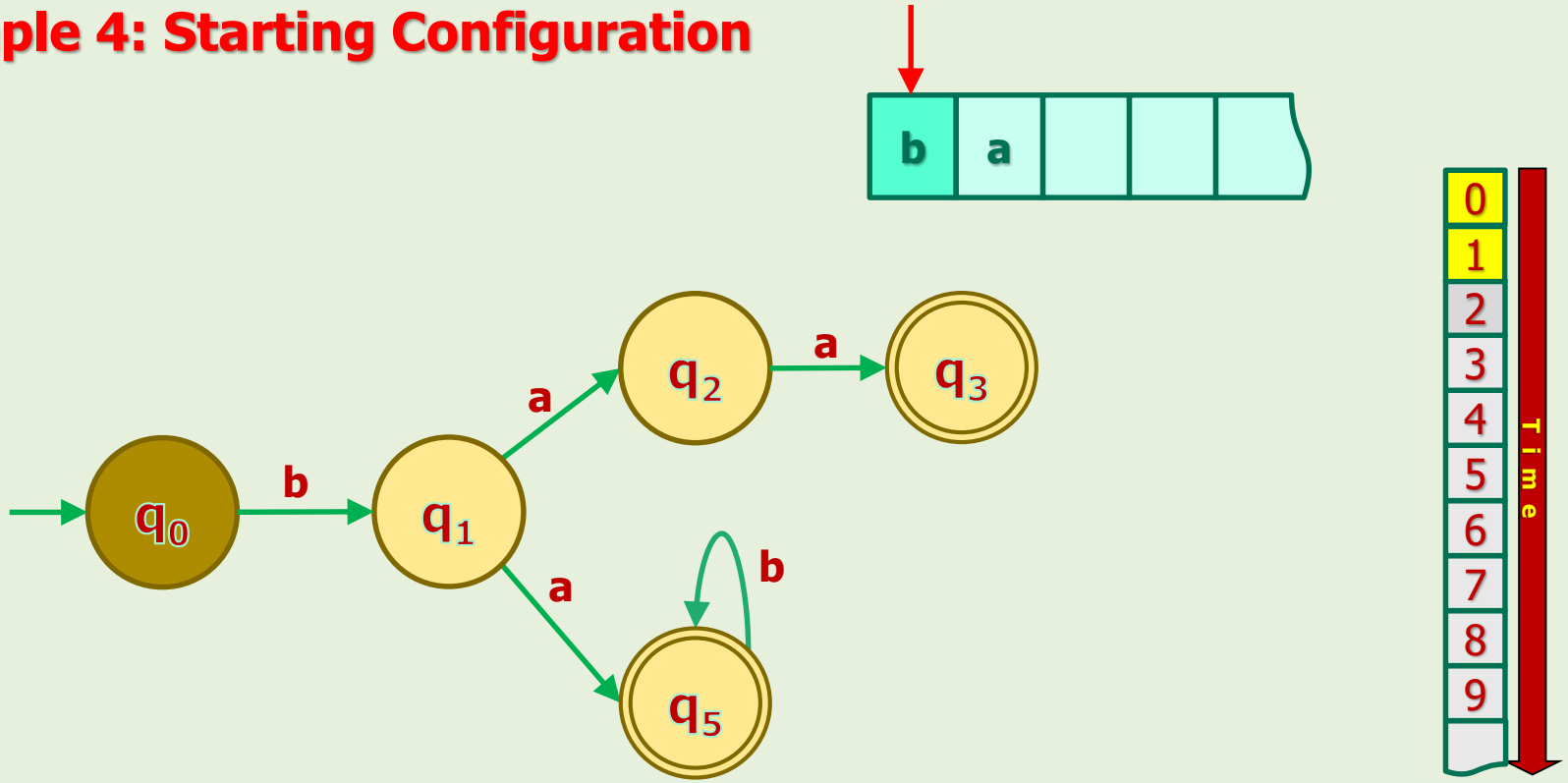
Example 4: Starting Configuration



- Process #1 (main) starts normally.

5. NFAs in Action

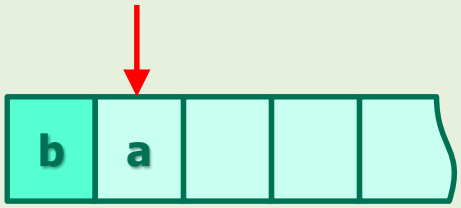
Example 4: Starting Configuration



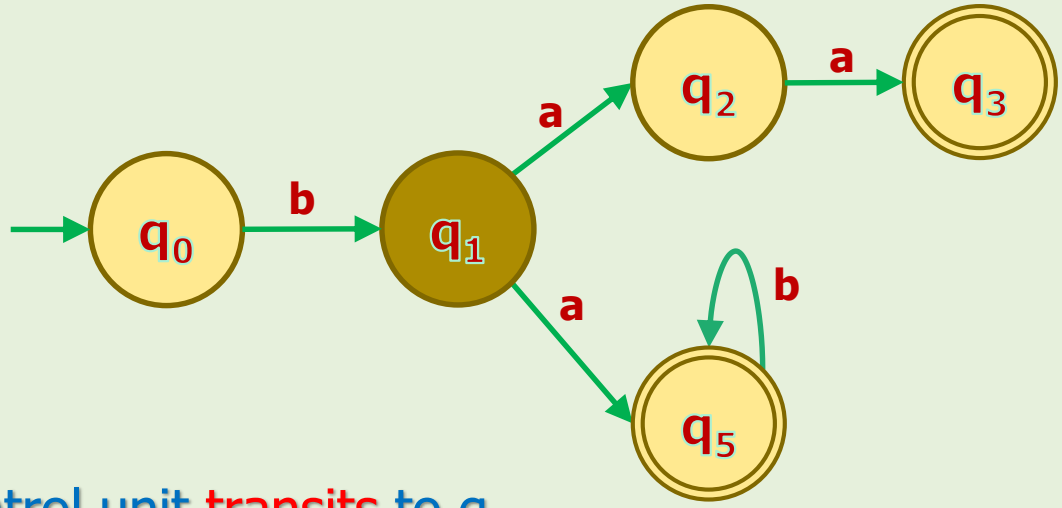
- Input tape reads 'b' and sends it to the control unit.
- The control unit makes a decision based on $\delta(q_0, b) = \{q_1\}$

5. NFAs in Action

Example 4: Process #1 (main)



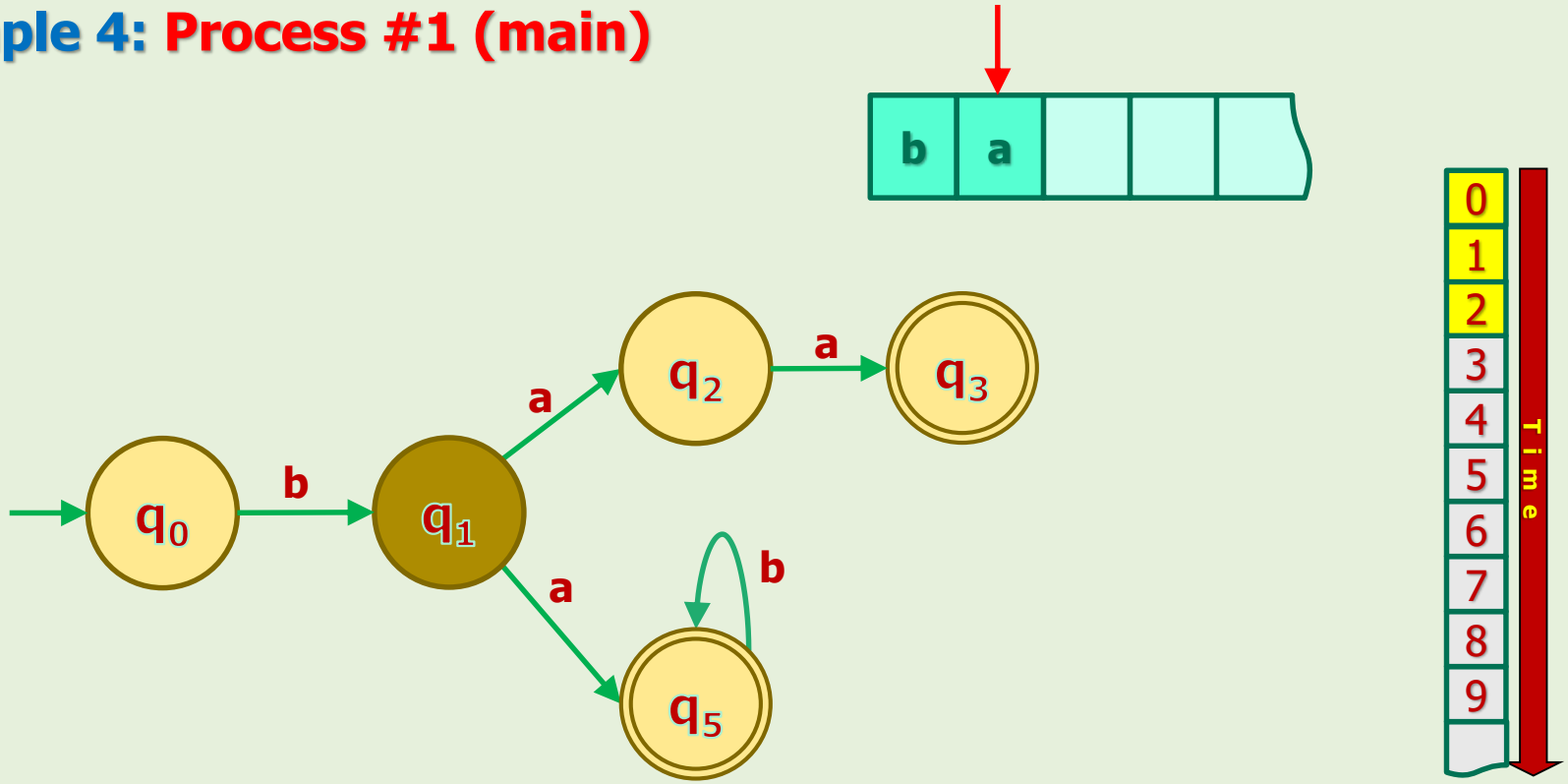
- $\delta(q_0, b) = \{q_1\}$



- 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

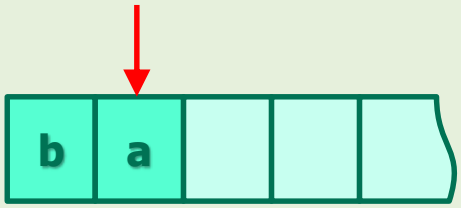
Example 4: Process #1 (main)



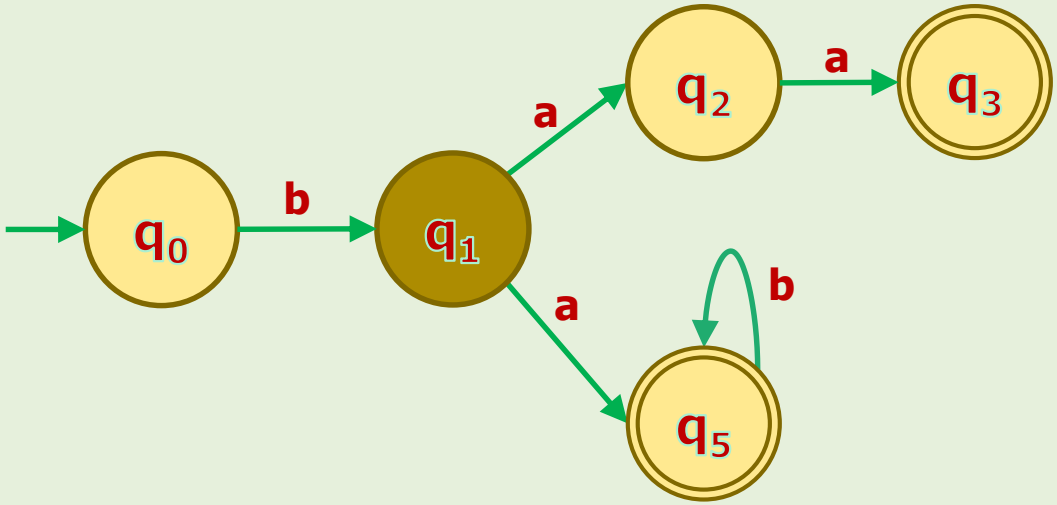
- Input tape reads 'a' and sends it to the control unit.
- The control unit makes a decision based on $\delta(q_1, a) = \{q_2, q_5\}$

5. NFAs in Action

Example 4: Process #1 (main)

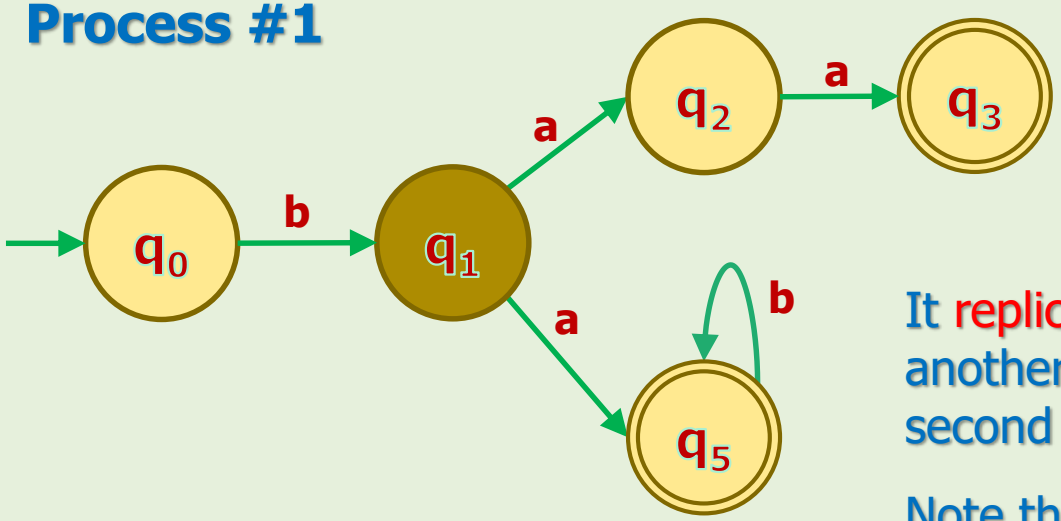


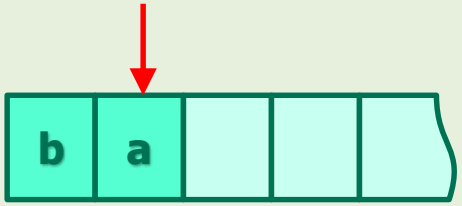
$\delta(q_1, a) = \{q_2, q_5\}$



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

Process #1

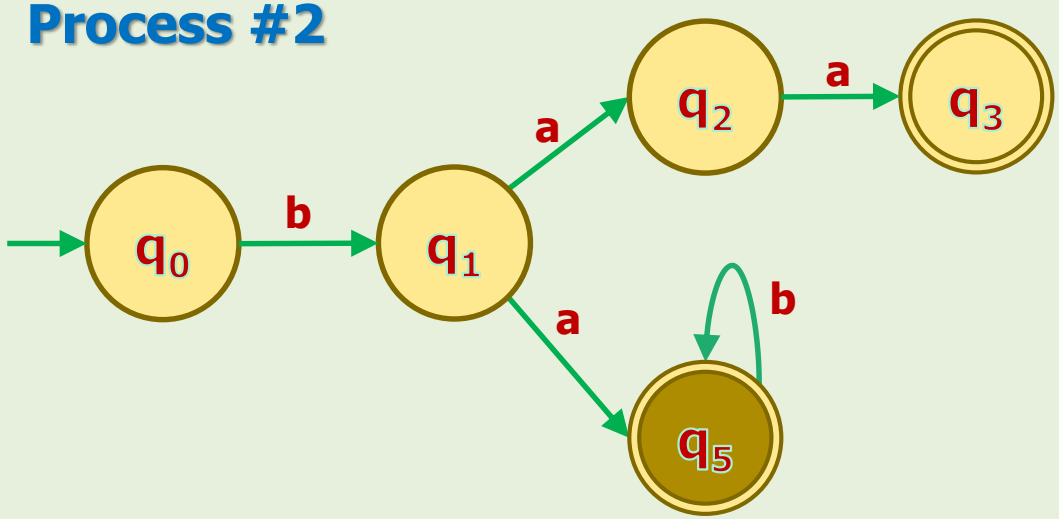


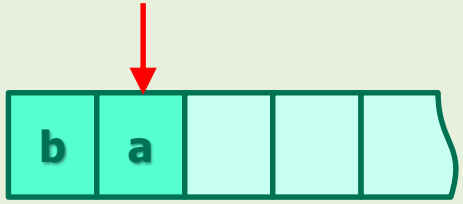


It replicates itself and another process will continue the second possibility.

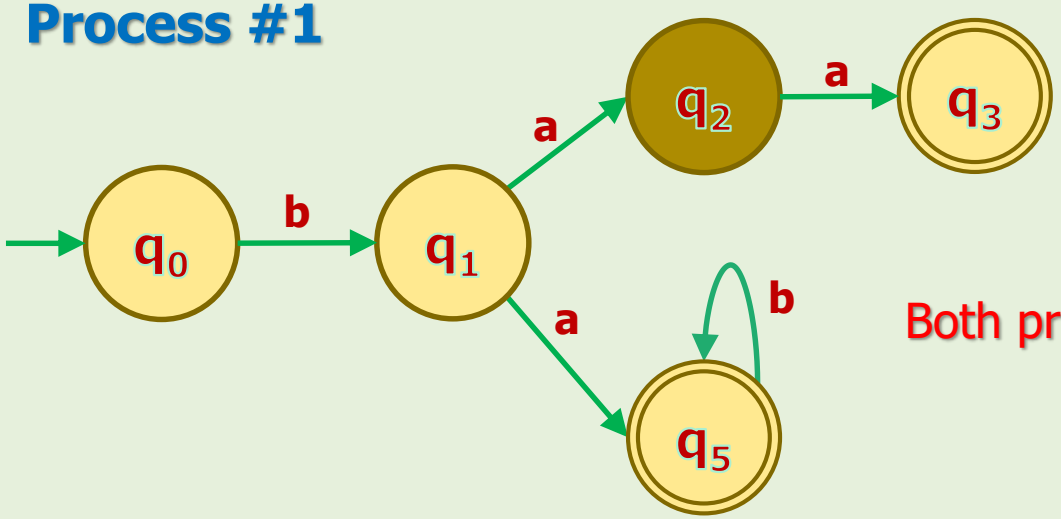
Note that 'a' is not consumed yet!

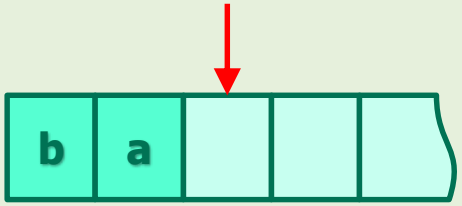
Process #2





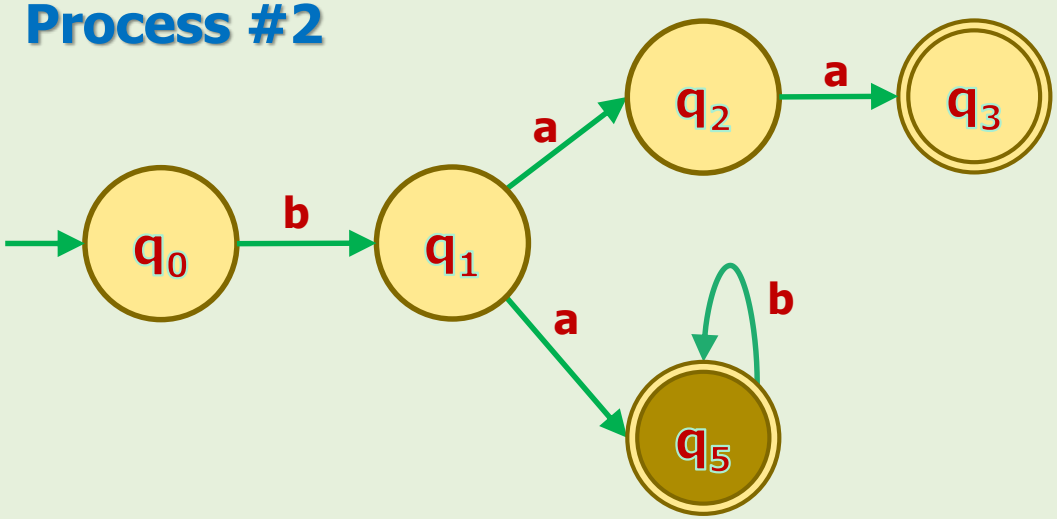
Process #1

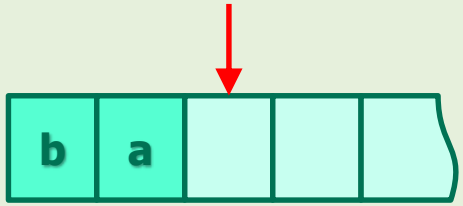




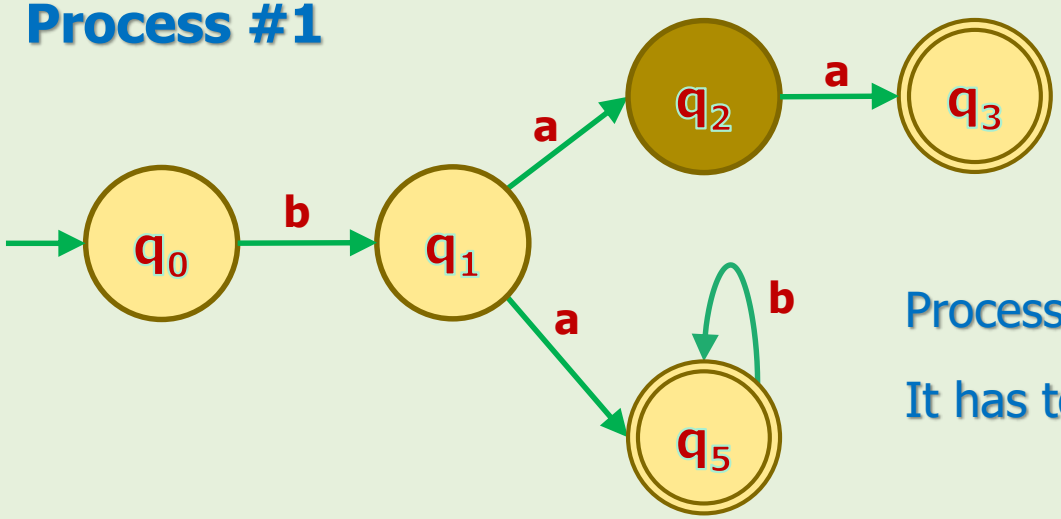
Both processes consume 'a'.

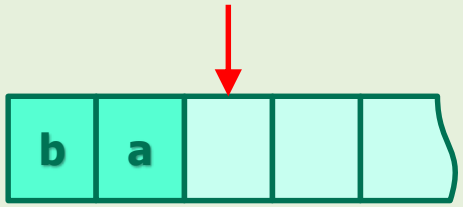
Process #2





Process #1

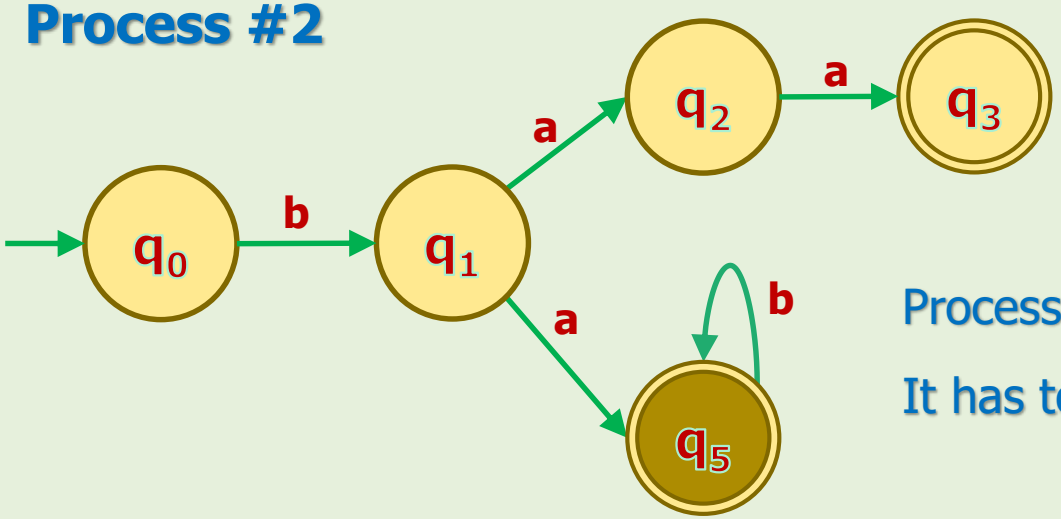


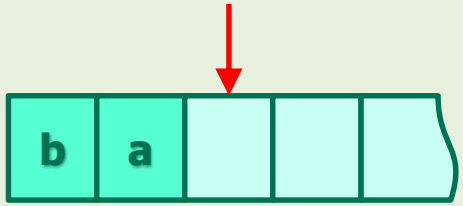


Process #1 is out of symbol.

It has to halt.

Process #2

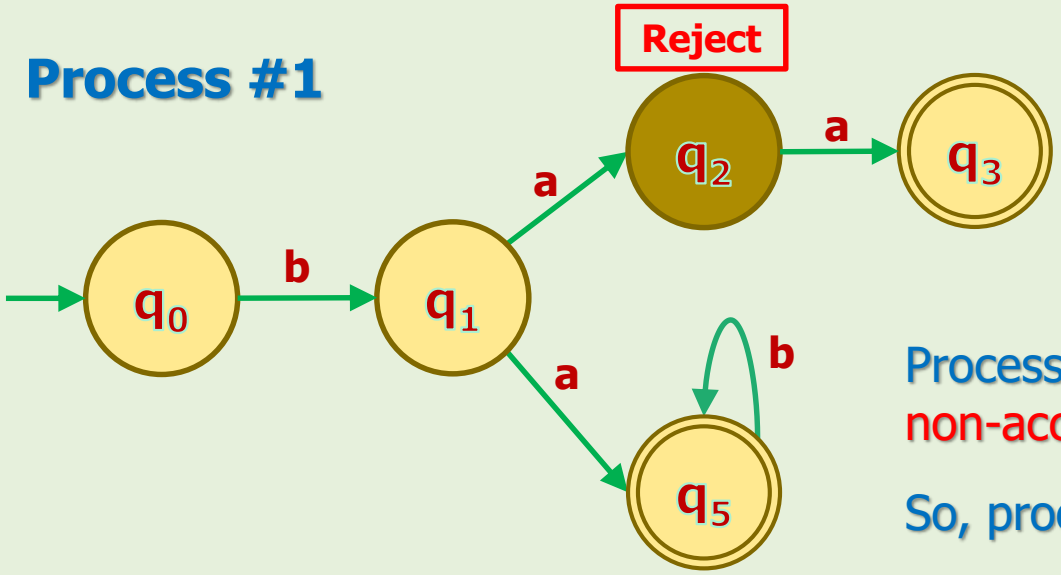


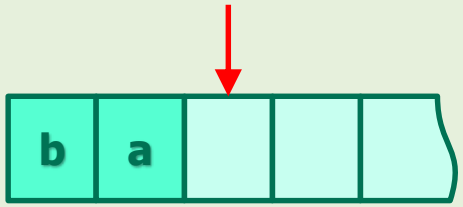


Process #2 is out of symbol.

It has to halt.

Process #1

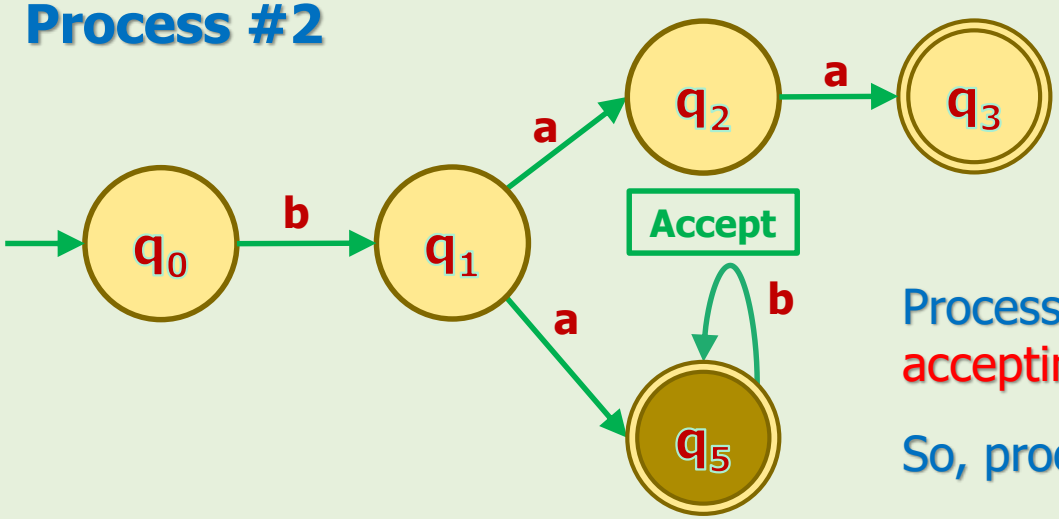


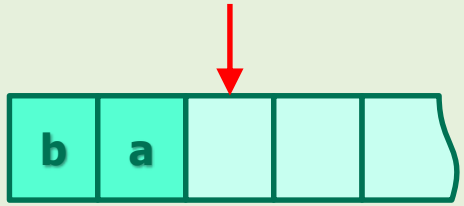


Process #1 halts in a non-accepting state.

So, process #1 rejects w.

Process #2



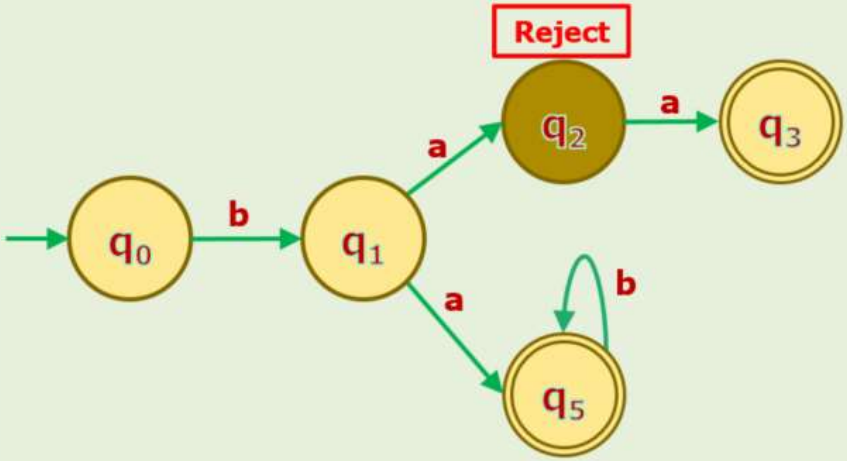


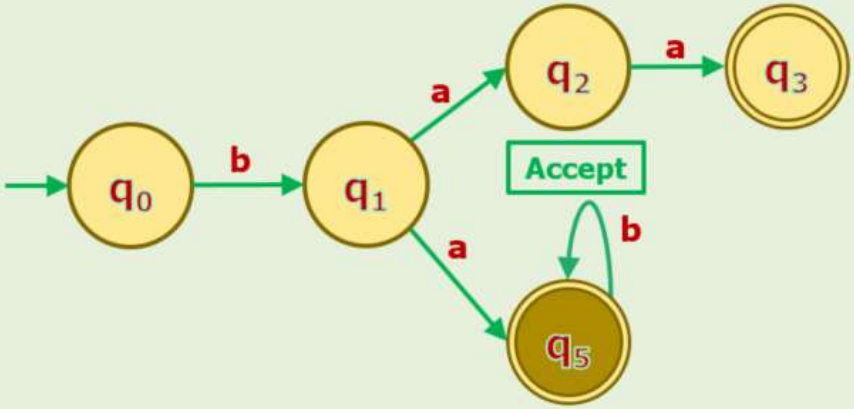
Process #2 halts in an accepting state.

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 the 3 conditions **(h ∧ c ∧ f)** are satisfied. (i.e.: **(h ∧ c ∧ f) ↔ a**)
  - Because **h** and **c** might have different values.

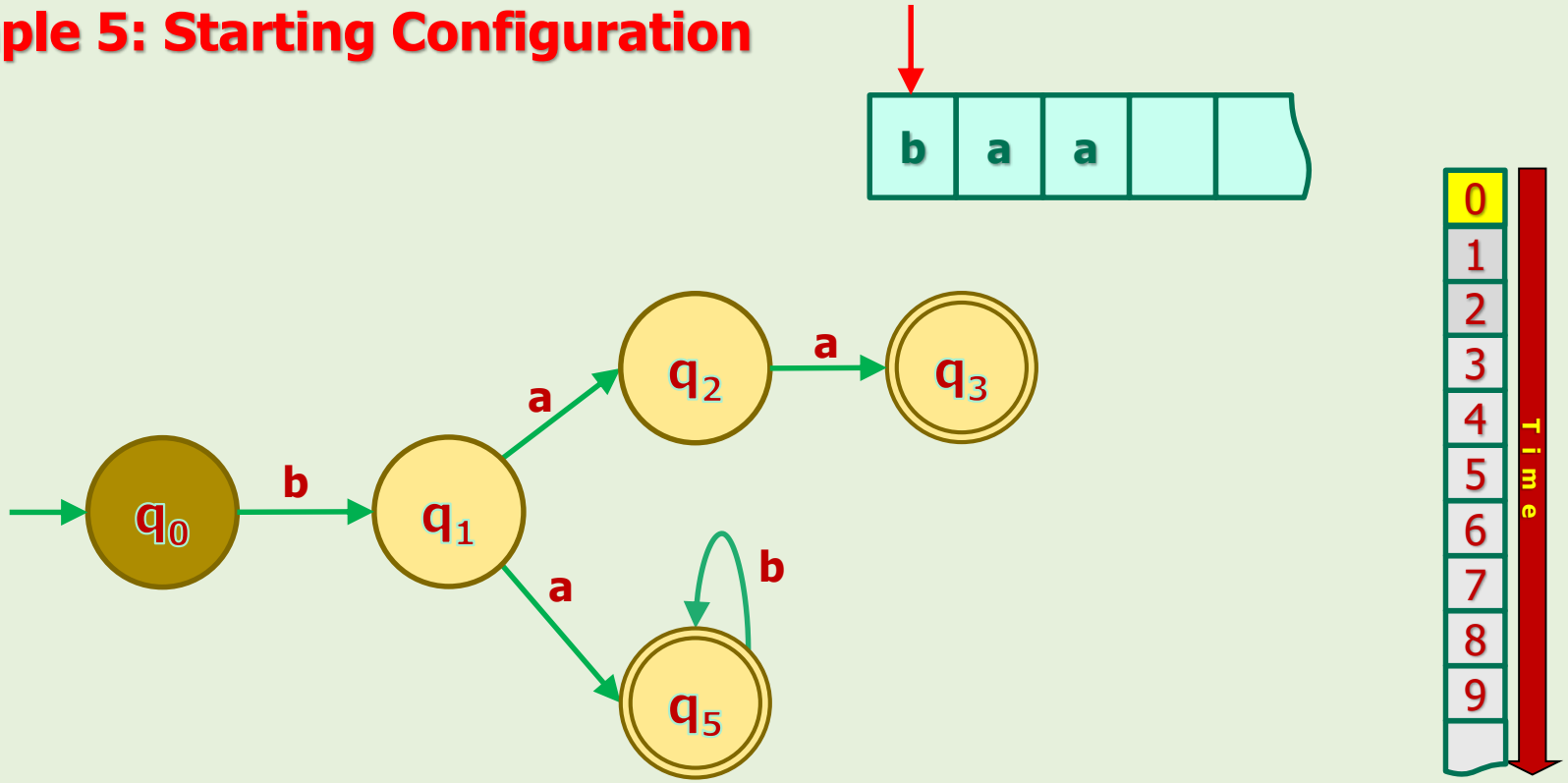
Rejecting Strings

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

- Let's take more examples.

5. NFAs in Action

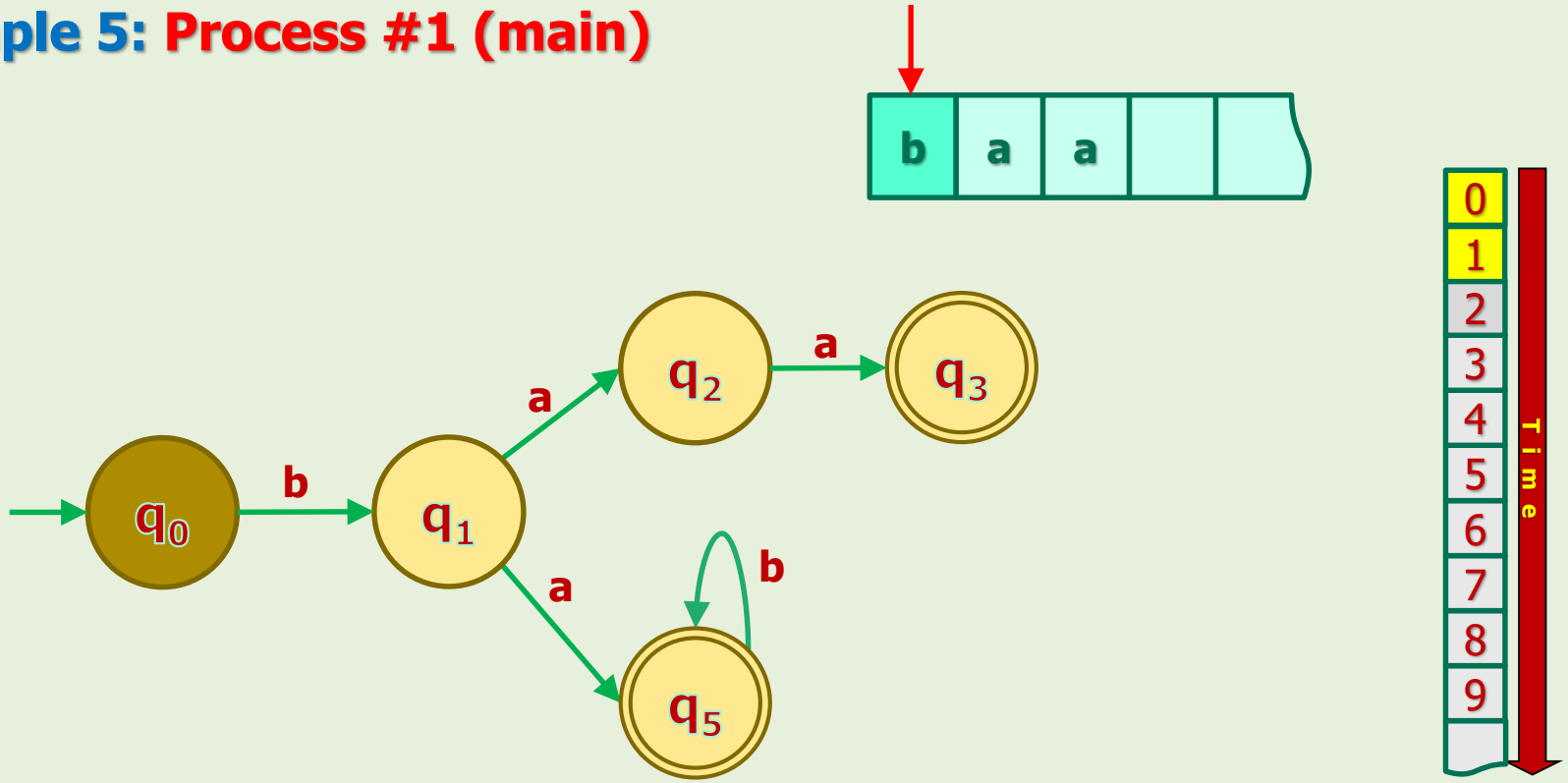
Example 5: Starting Configuration



- Process #1 (main) starts normally.

5. NFAs in Action

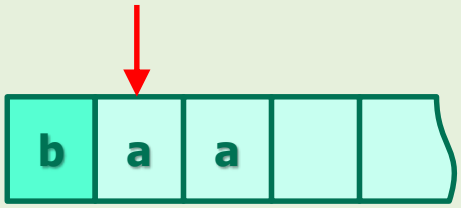
Example 5: Process #1 (main)



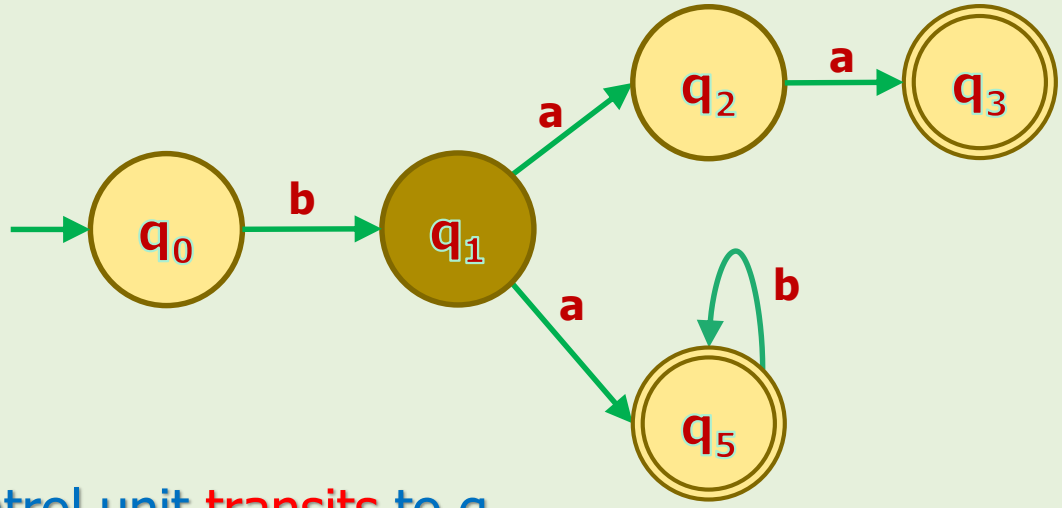
- Input tape reads 'b' and sends it to the control unit.
- The control unit makes a decision based on $\delta(q_0, b) = \{q_1\}$

5. NFAs in Action

Example 5: Process #1 (main)



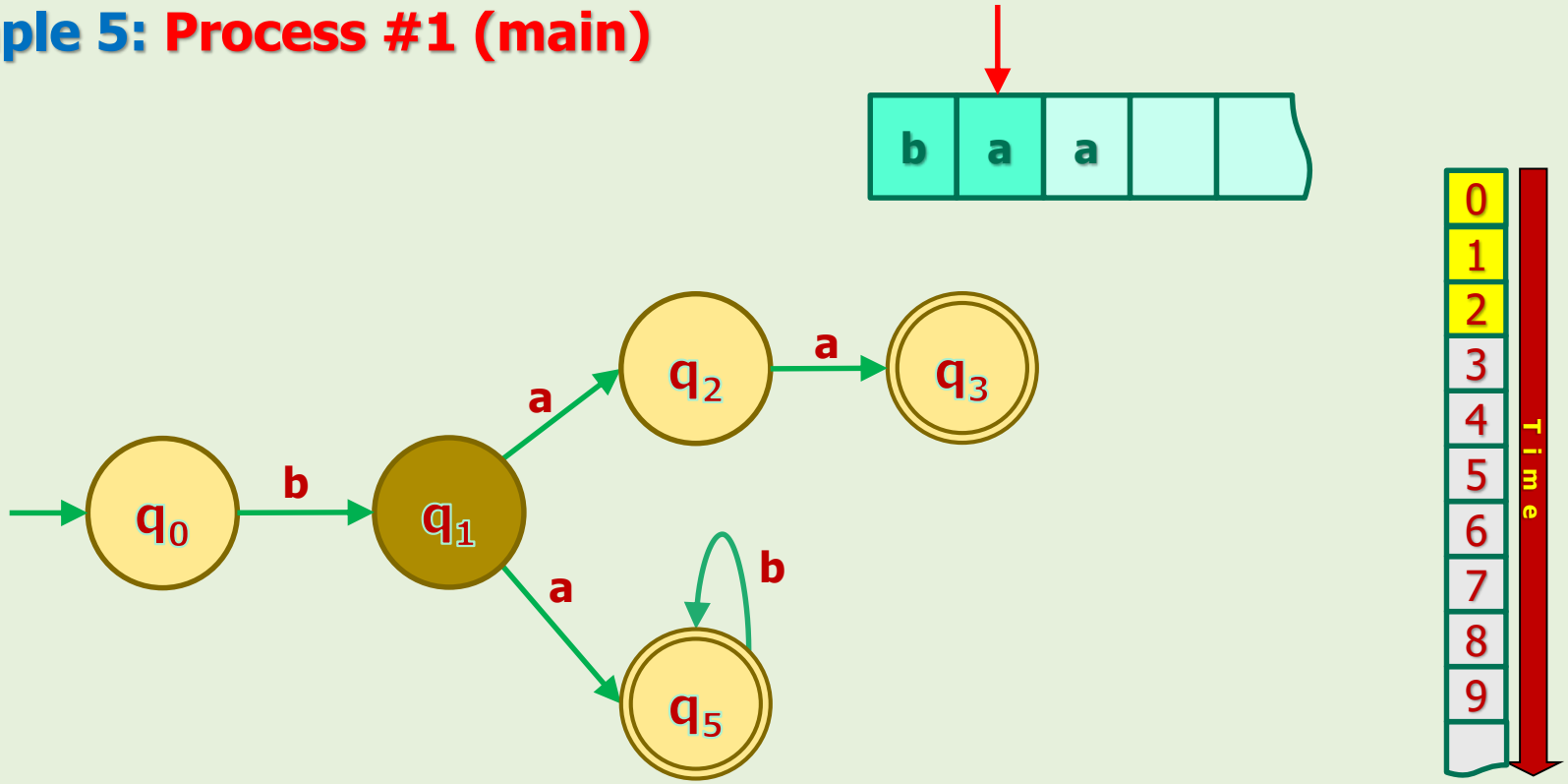
- $\delta(q_0, b) = \{q_1\}$



- 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

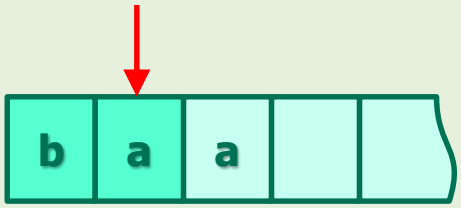
Example 5: Process #1 (main)



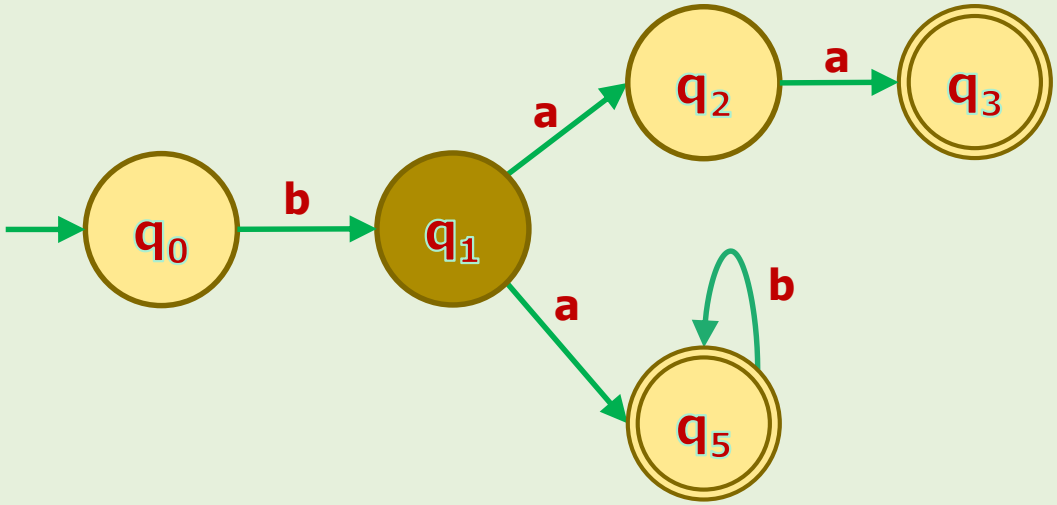
- Input tape reads 'a' and sends it to the control unit.
- The control unit makes a decision based on $\delta(q_1, a) = \{q_2, q_5\}$

5. NFAs in Action

Example 5: Process #1 (main)

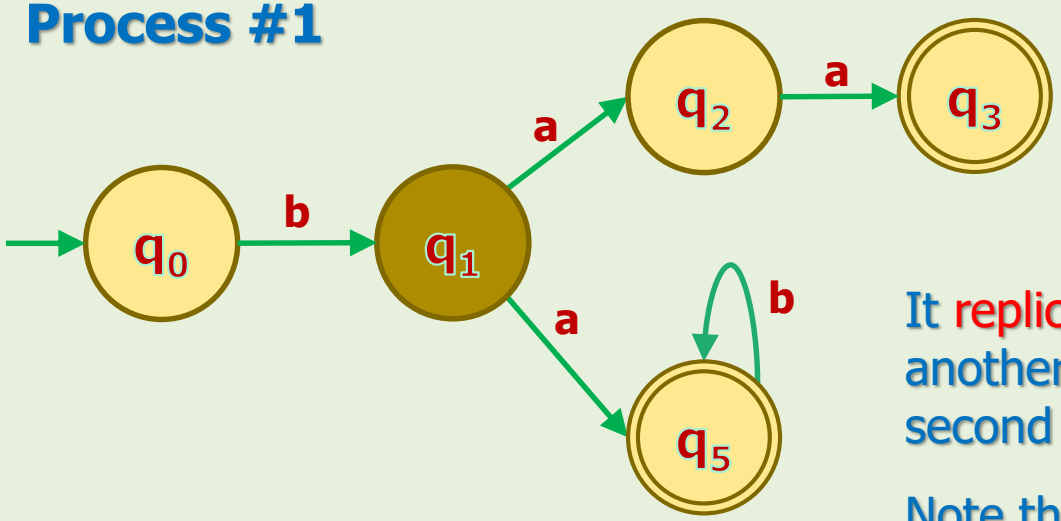


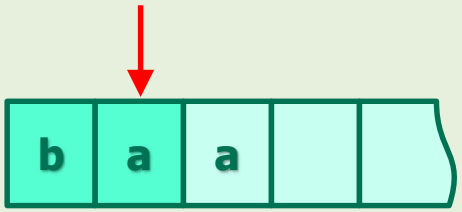
$\delta(q_1, a) = \{q_2, q_5\}$



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

Process #1

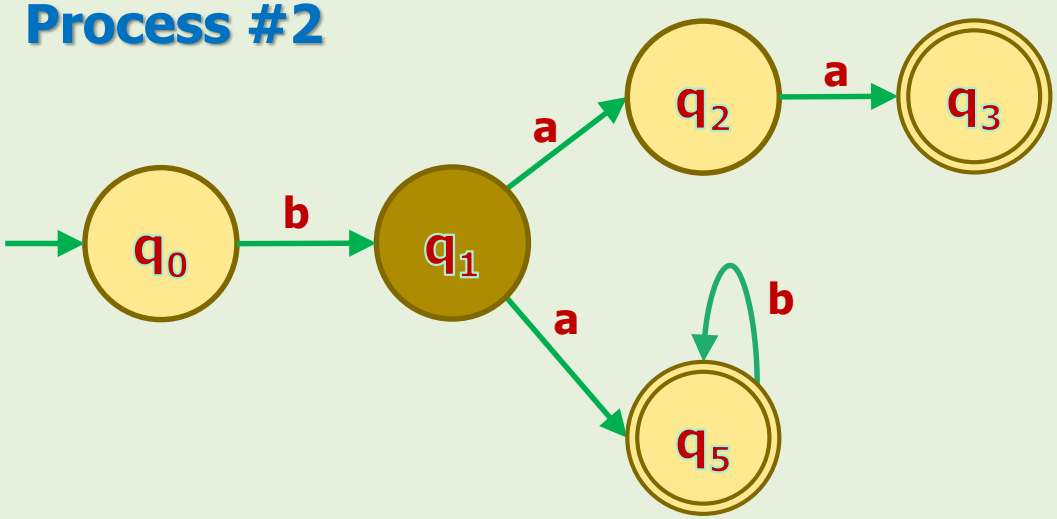


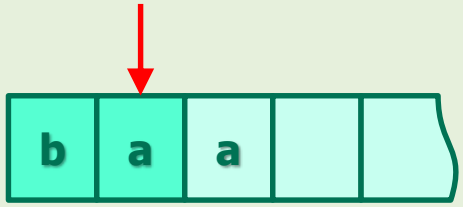


It replicates itself and another process will continue the second possibility.

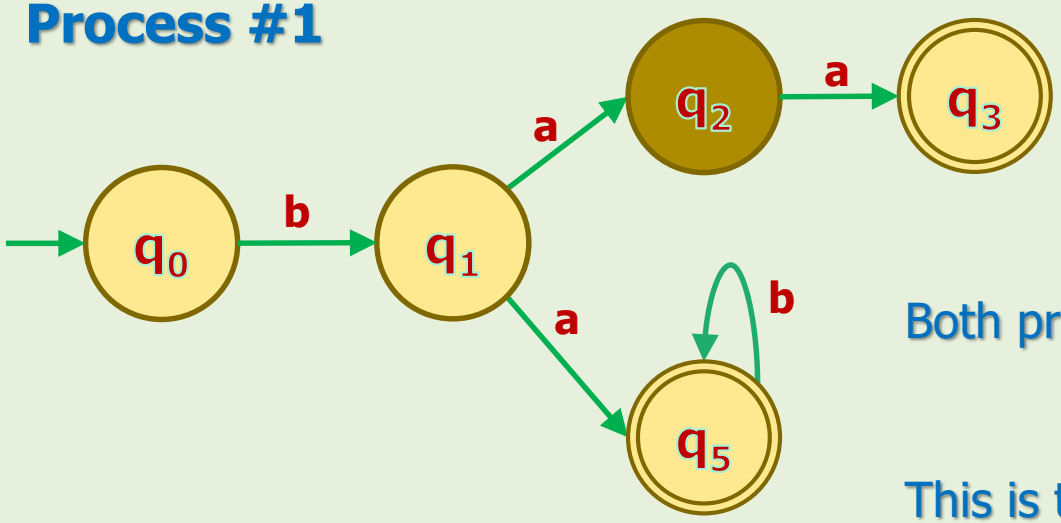
Note that 'a' is not consumed yet!

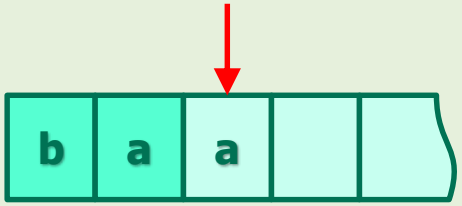
Process #2





Process #1

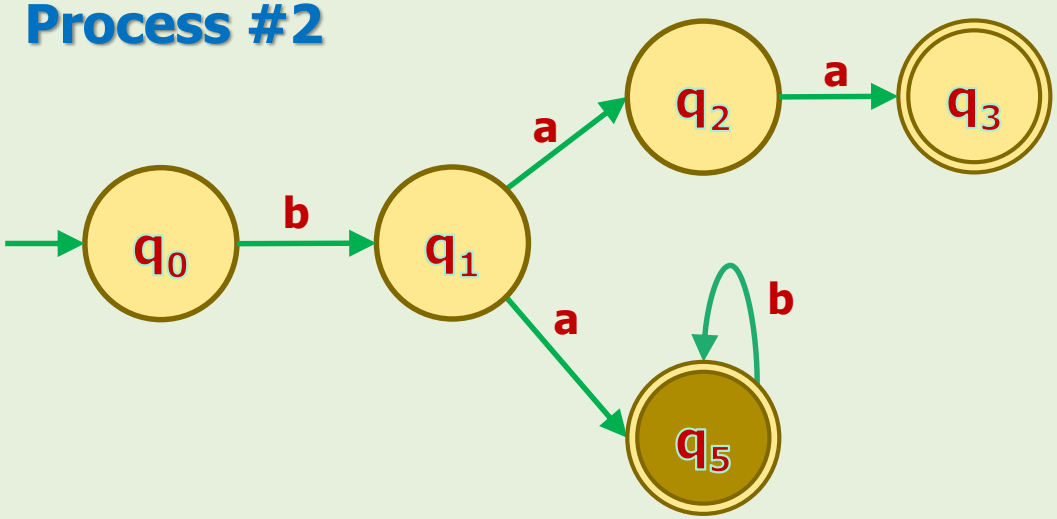


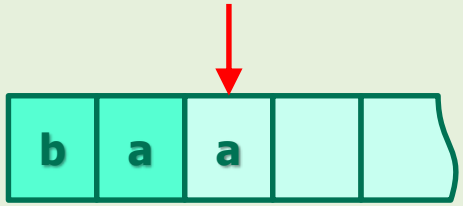


Both processes consume 'a'.

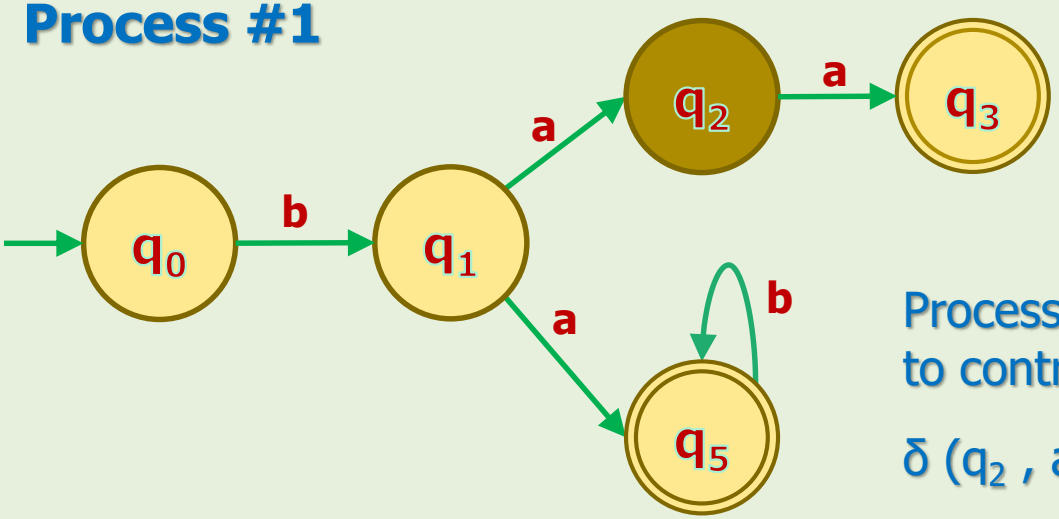
This is the end of timeframe 2.

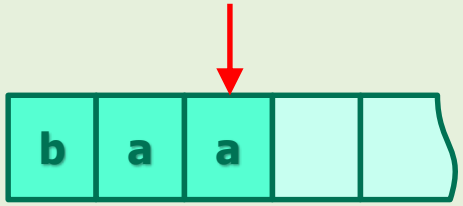
Process #2





Process #1

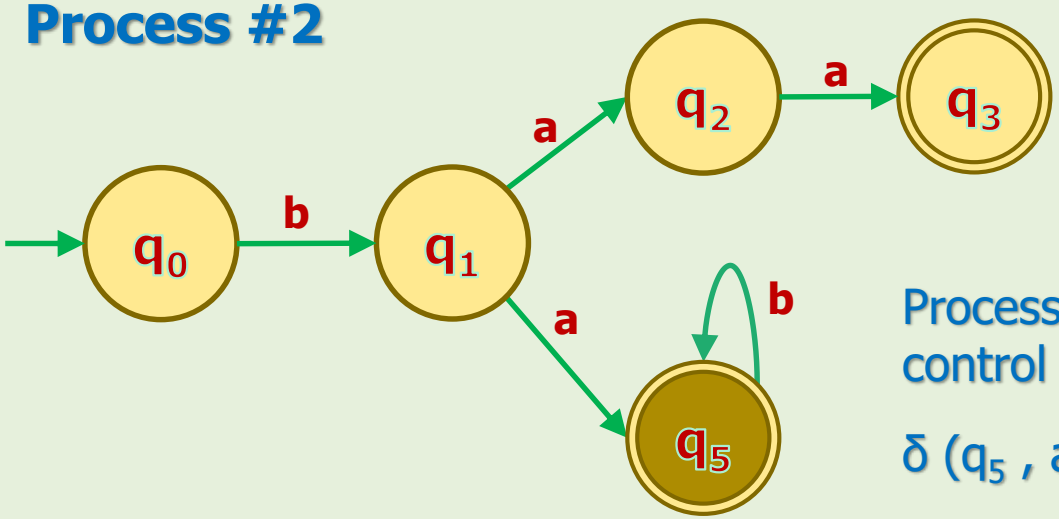


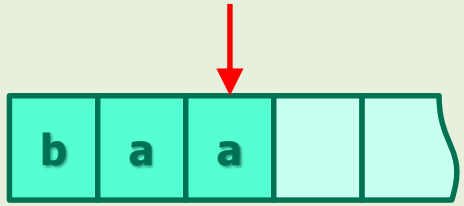


Process #1 reads 'a' and sends it to control unit.

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

Process #2

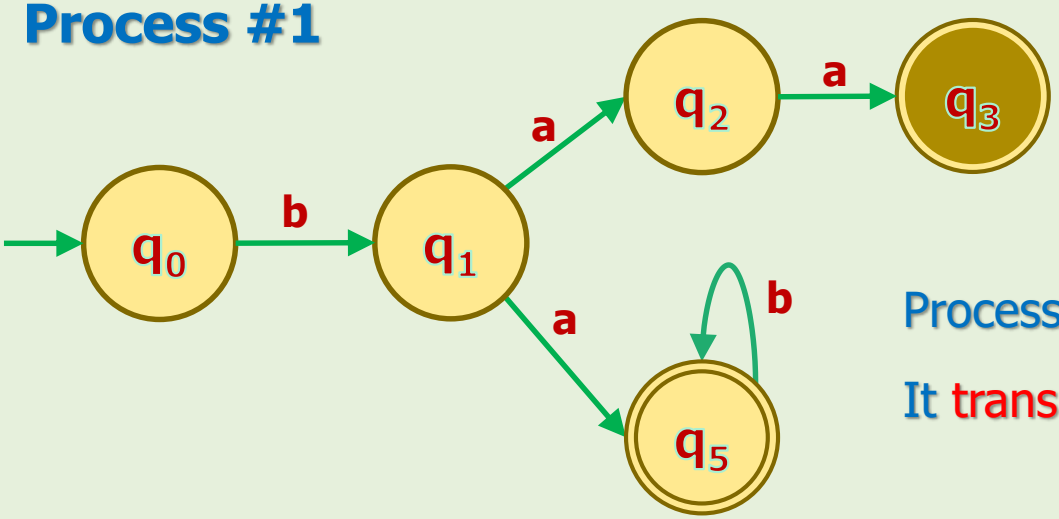


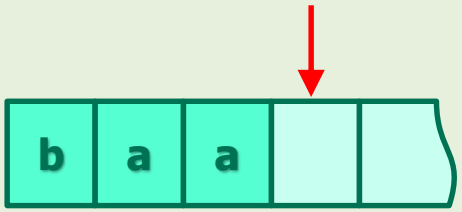


Process #2 reads 'a' and sends it to control unit.

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

Process #1

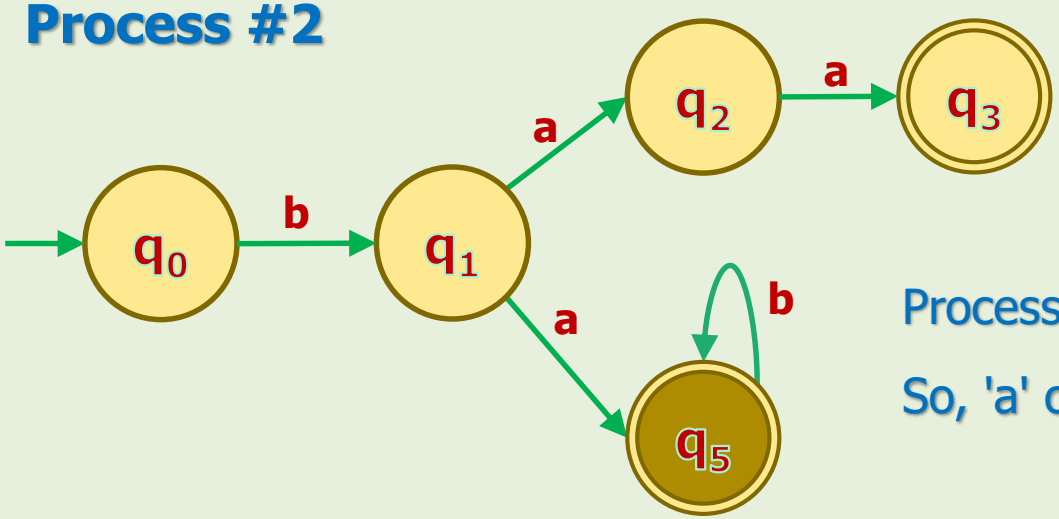


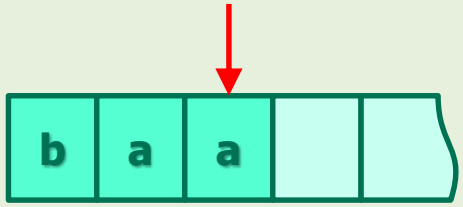


Process #1 consumes 'a'.

It transits to q_3.

Process #2

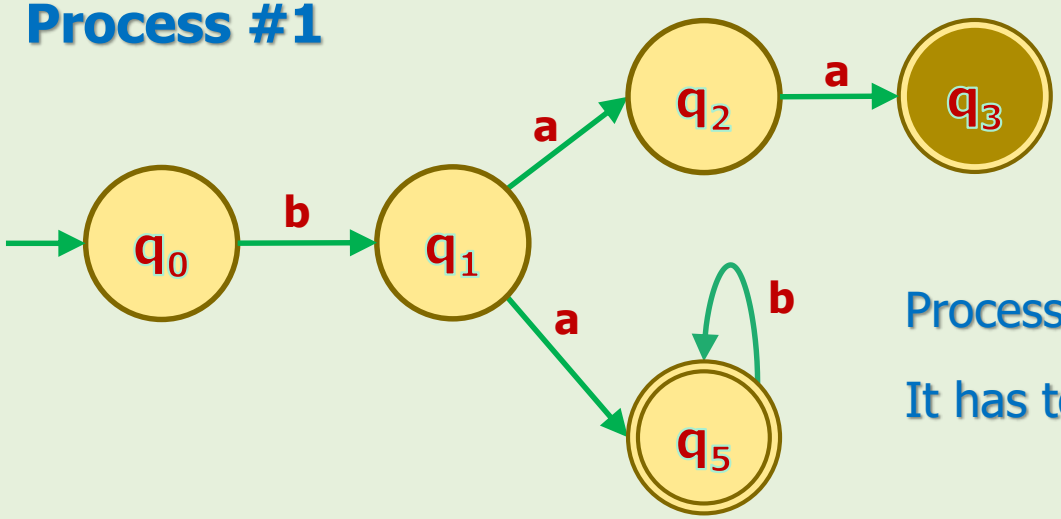


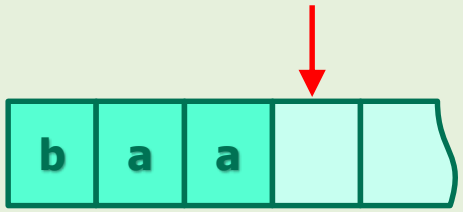


Process #2 has no choice for 'a'.

So, 'a' cannot be consumed.

Process #1

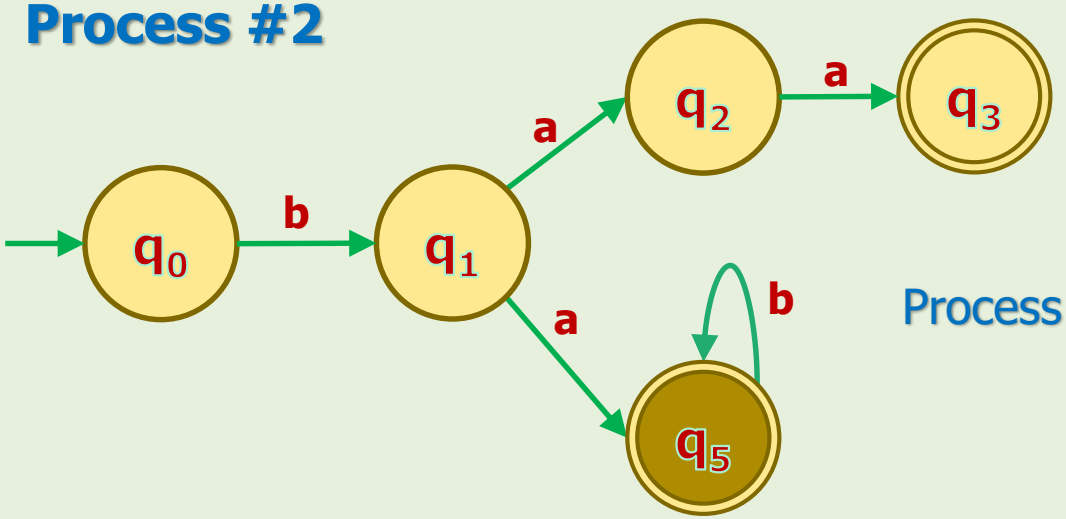


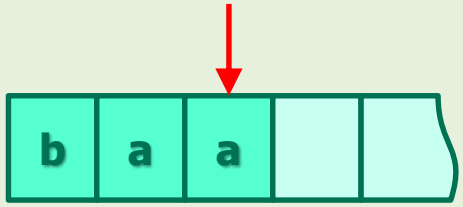


Process #1 is out of symbol.

It has to halt.

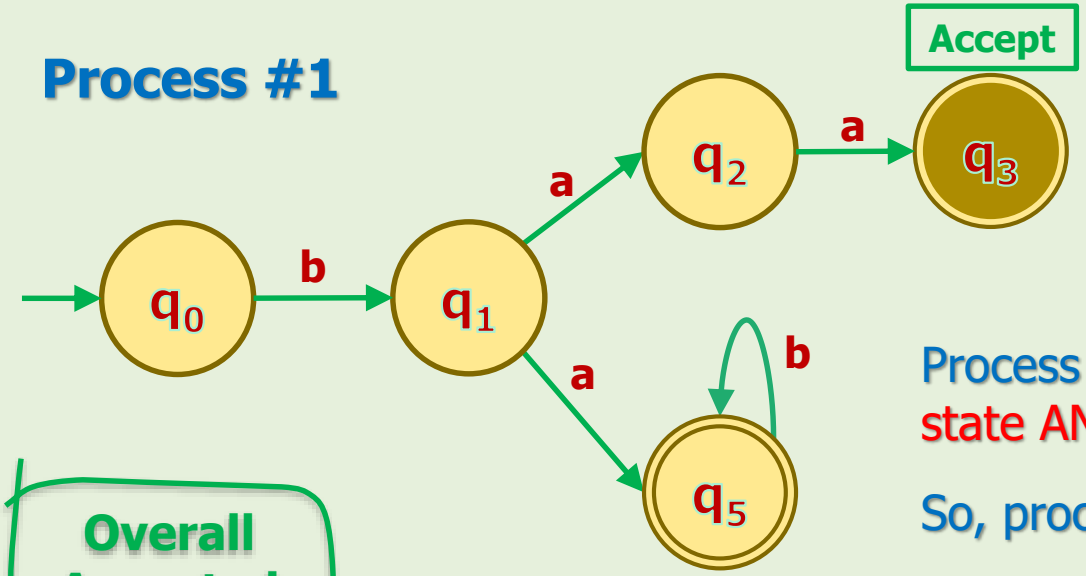
Process #2

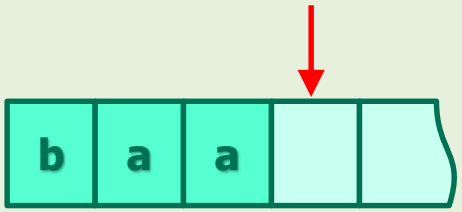




Process #2 hast to halt.

Process #1



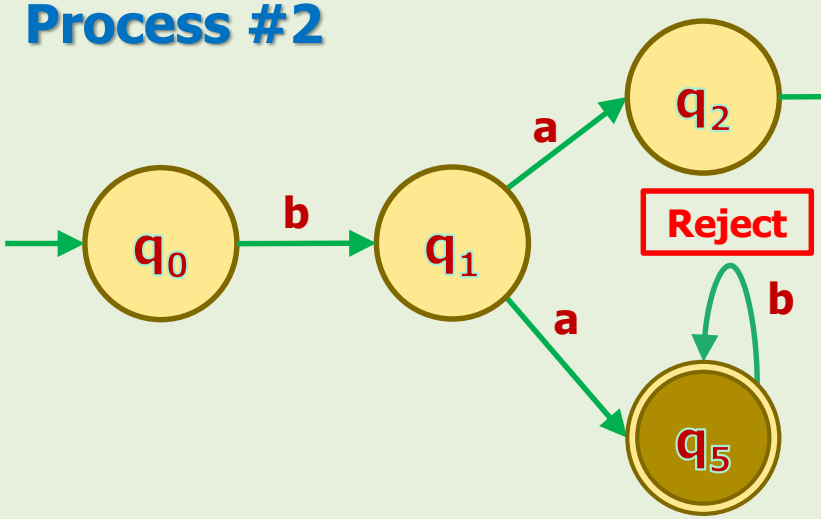


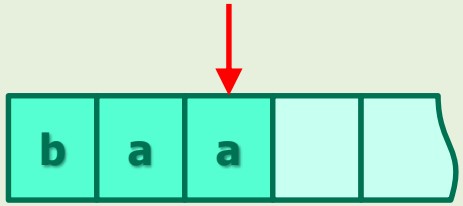
Process #1 halts in an accepting state AND all symbols are consumed.

So, process #1 accepts w.

Overall Accepted

Process #2



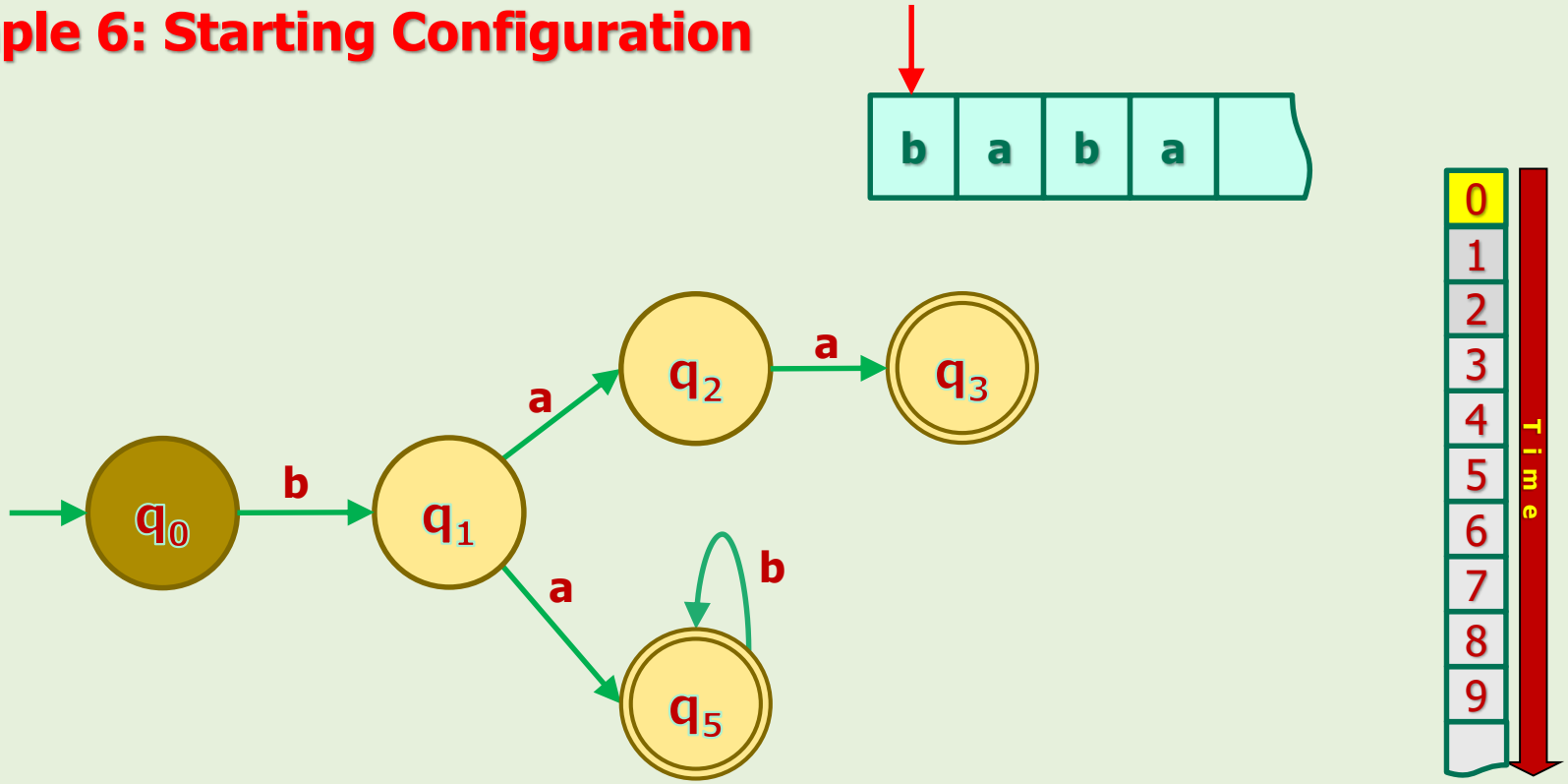


Process #2 halts in an accepting state BUT all symbols are not consumed.

So, process #2 rejects w.

5. NFAs in Action

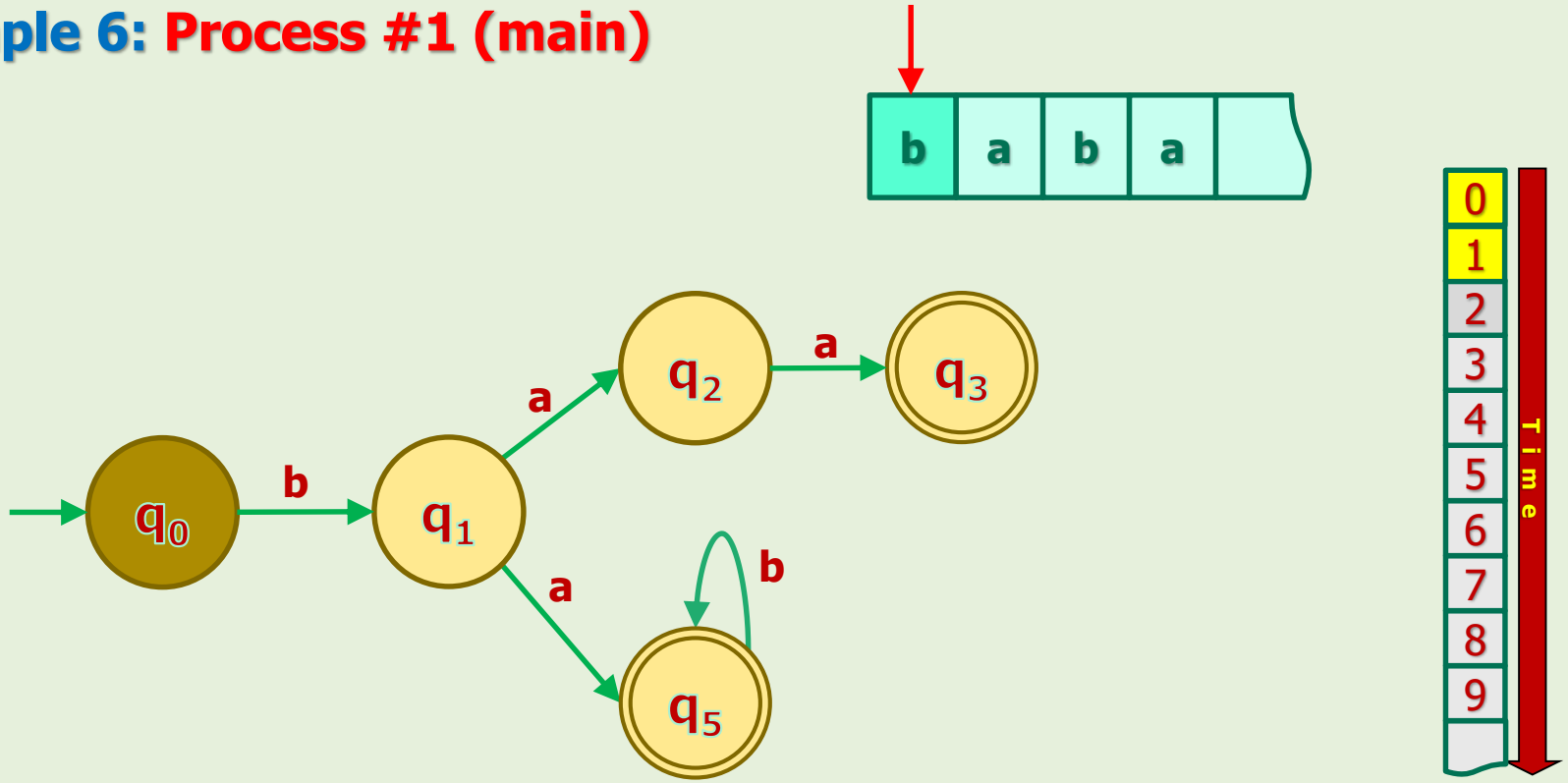
Example 6: Starting Configuration



- Process #1 (main) starts normally.

5. NFAs in Action

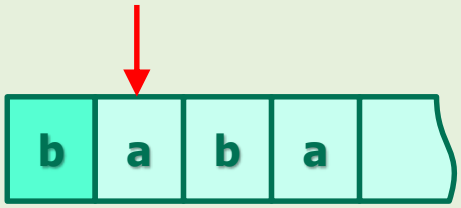
Example 6: Process #1 (main)



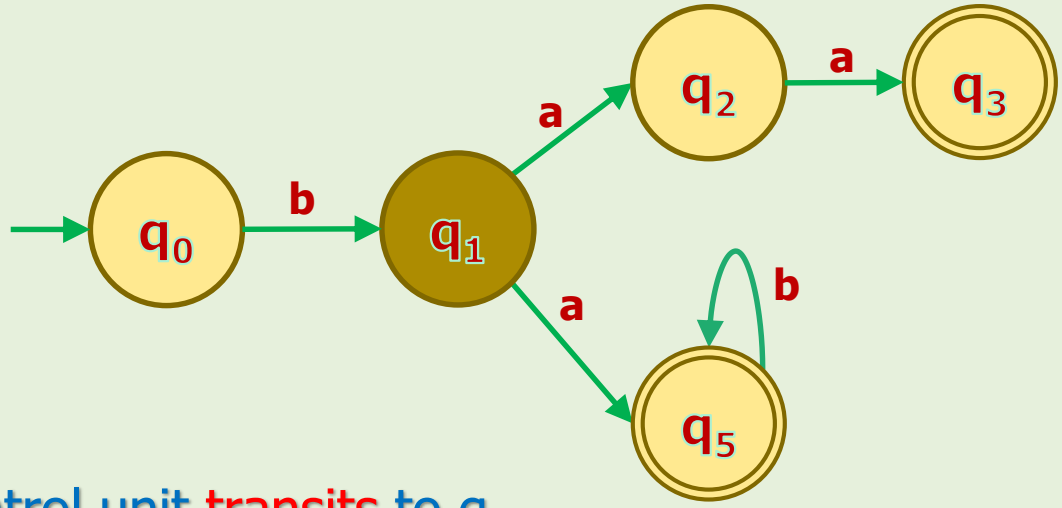
- Input tape reads 'b' and sends it to the control unit.
- The control unit makes a decision based on $\delta(q_0, b) = \{q_1\}$

5. NFAs in Action

Example 6: Process #1 (main)



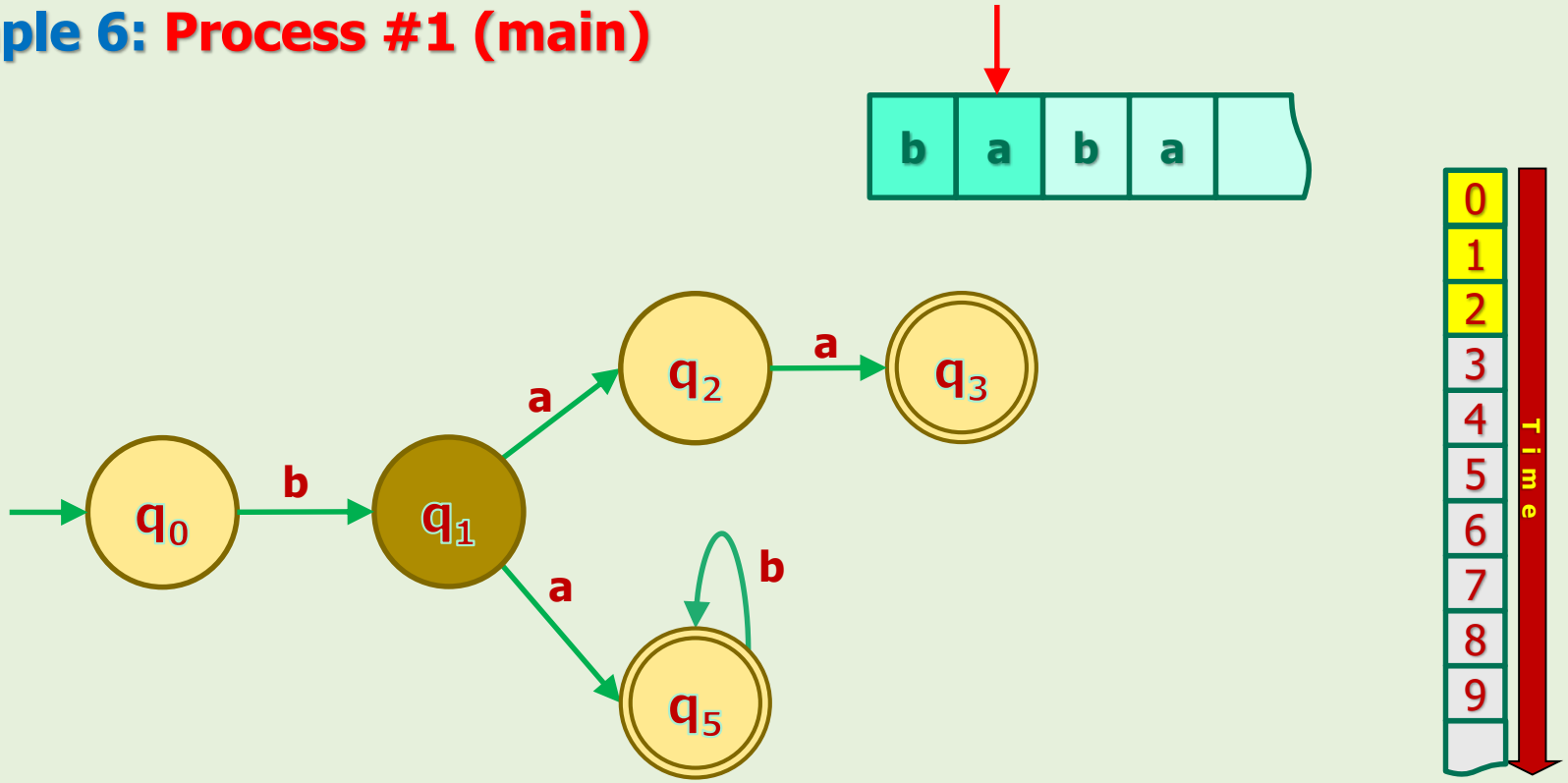
- $\delta(q_0, b) = \{q_1\}$



- 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

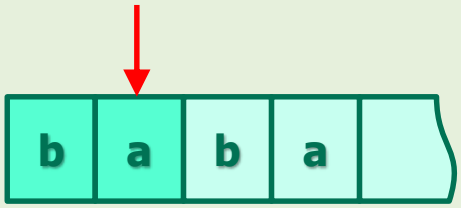
Example 6: Process #1 (main)



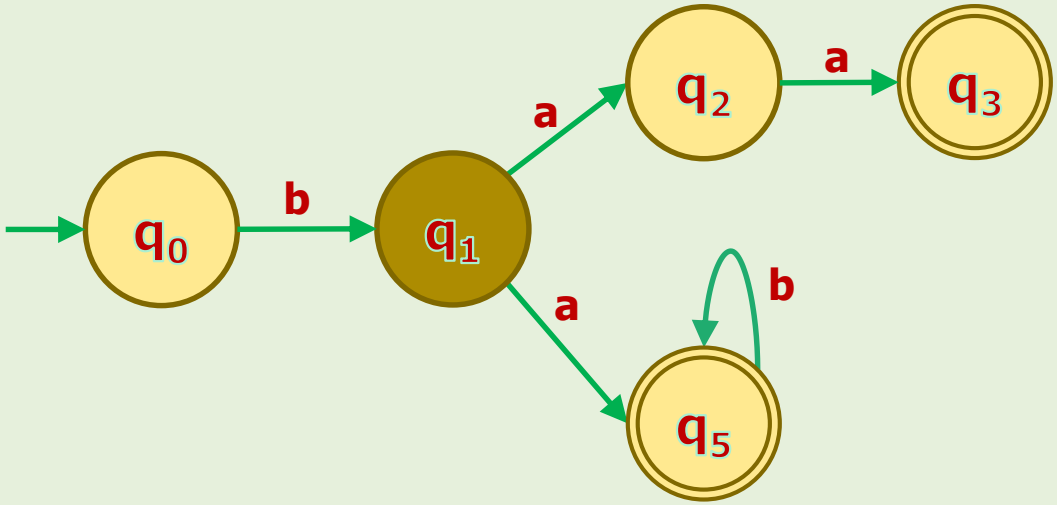
- Input tape reads 'a' and sends it to the control unit.
- The control unit makes a decision based on $\delta(q_1, a) = \{q_2, q_5\}$

5. NFAs in Action

Example 6: Process #1 (main)

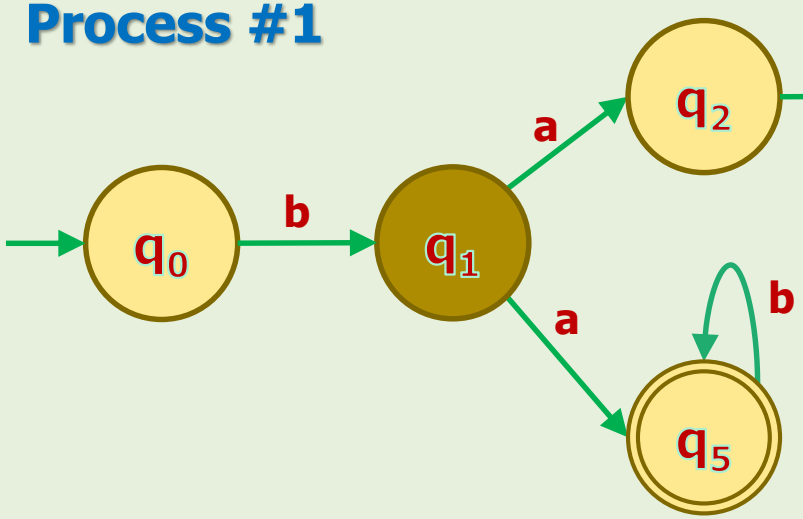


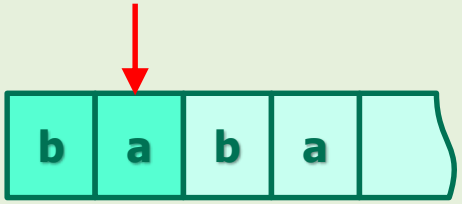
$\delta(q_1, a) = \{q_2, q_5\}$



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

Process #1

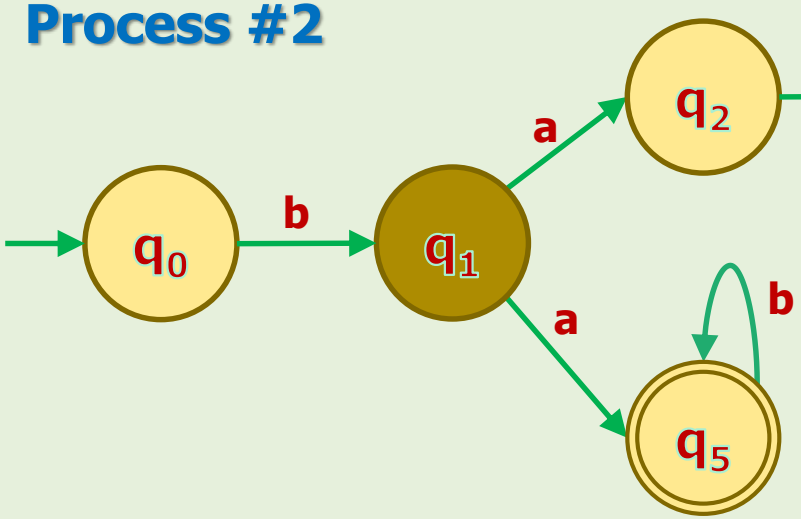


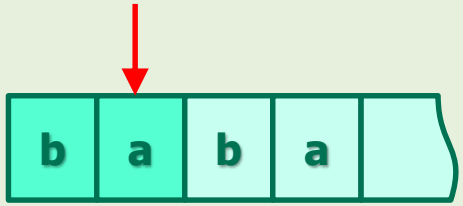


It replicates itself and another process will continue the second possibility.

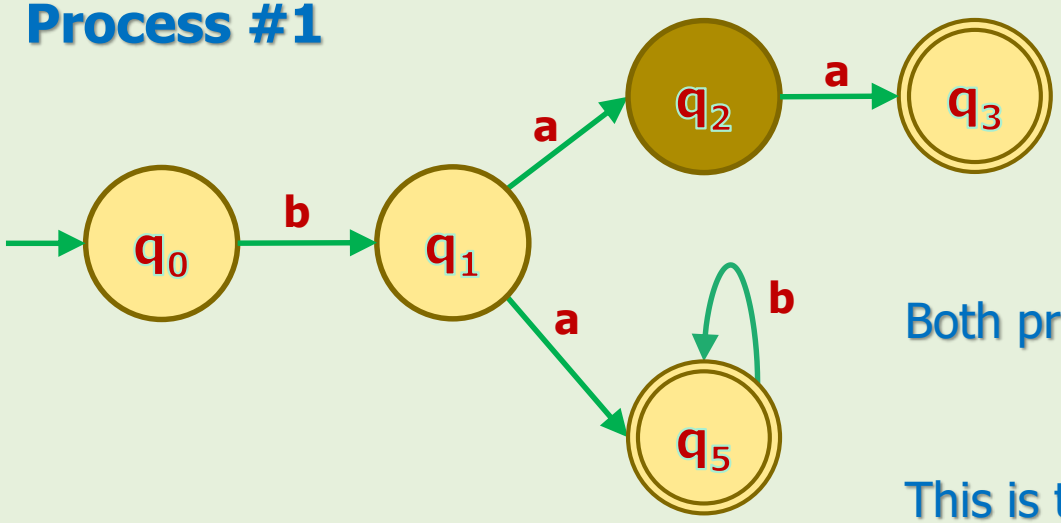
Note that 'a' is not consumed yet!

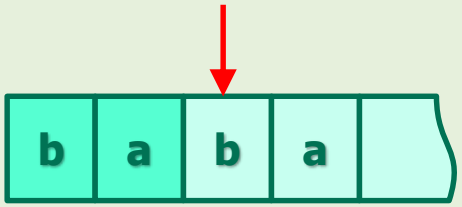
Process #2





Process #1

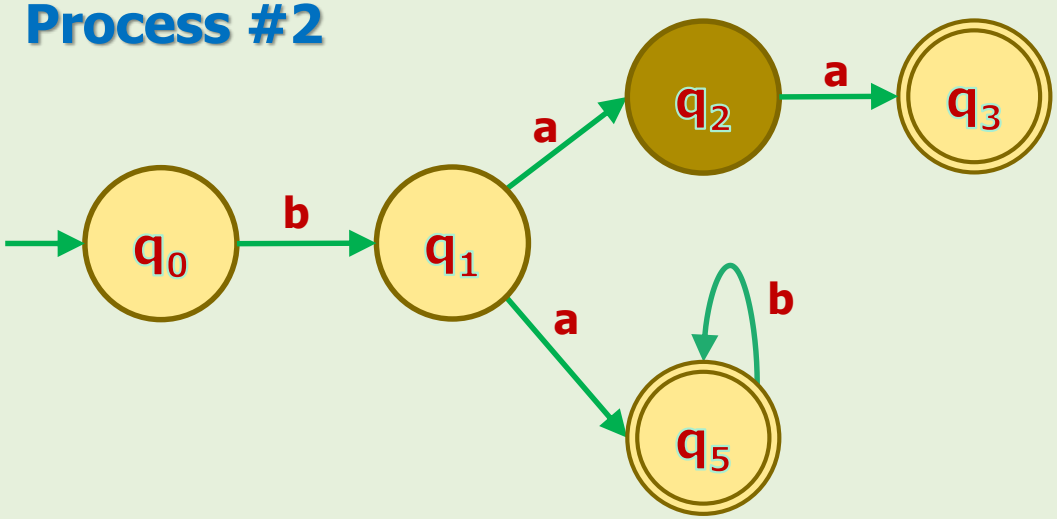


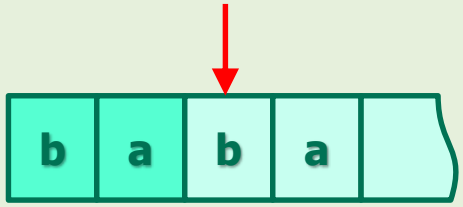


Both processes consume 'a'.

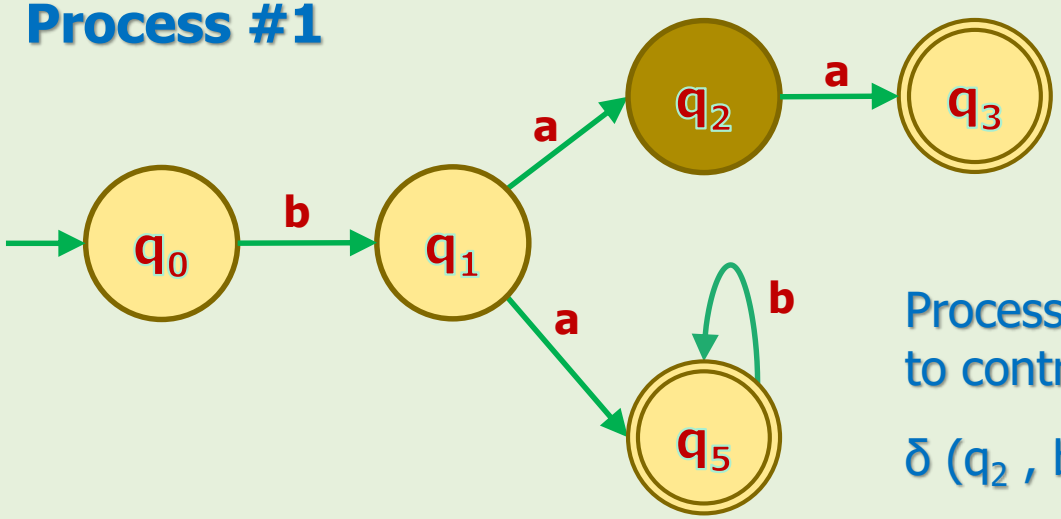
This is the end of timeframe 2.

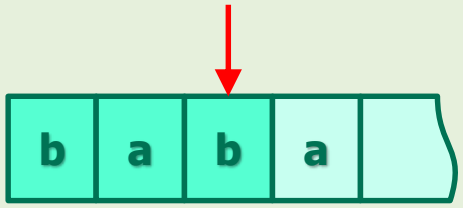
Process #2





Process #1

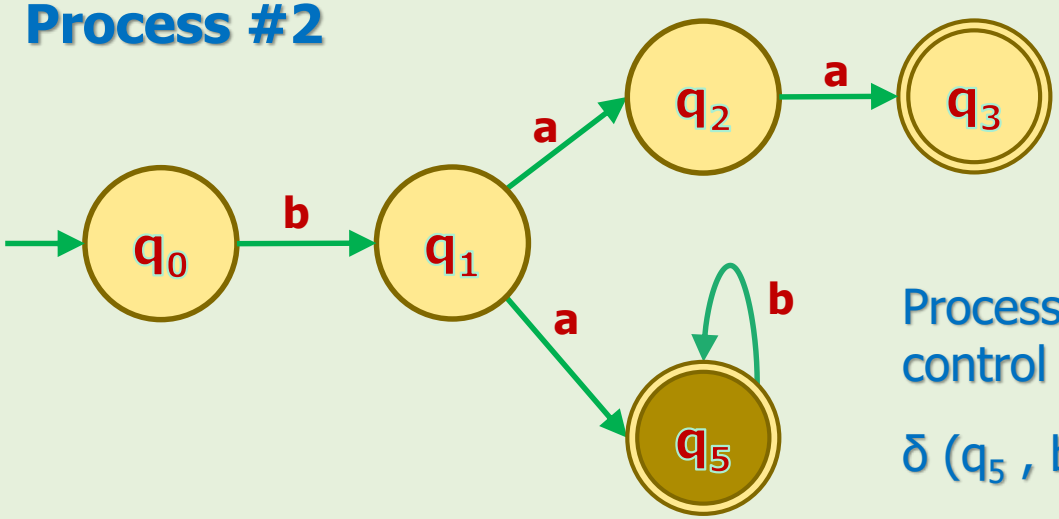


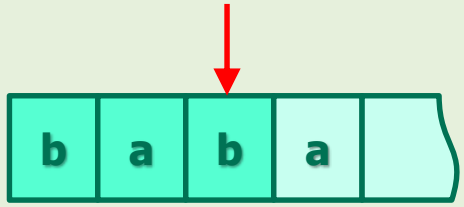


Process #1 reads 'b' and sends it to control unit.

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

Process #2

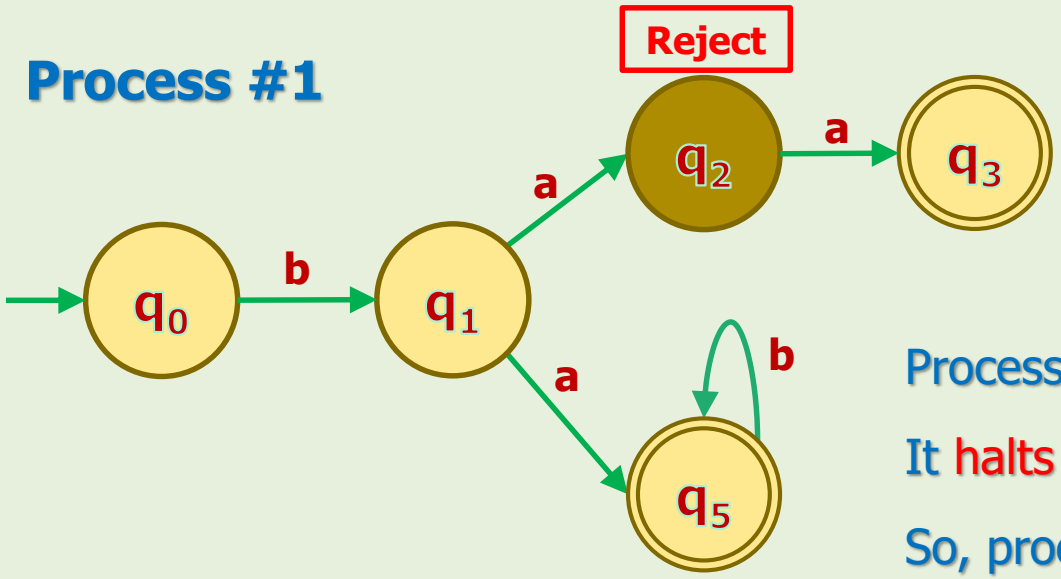


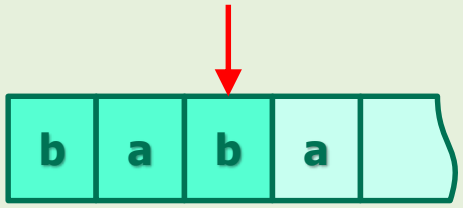


Process #2 reads 'b' and sends it to control unit.

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

Process #1



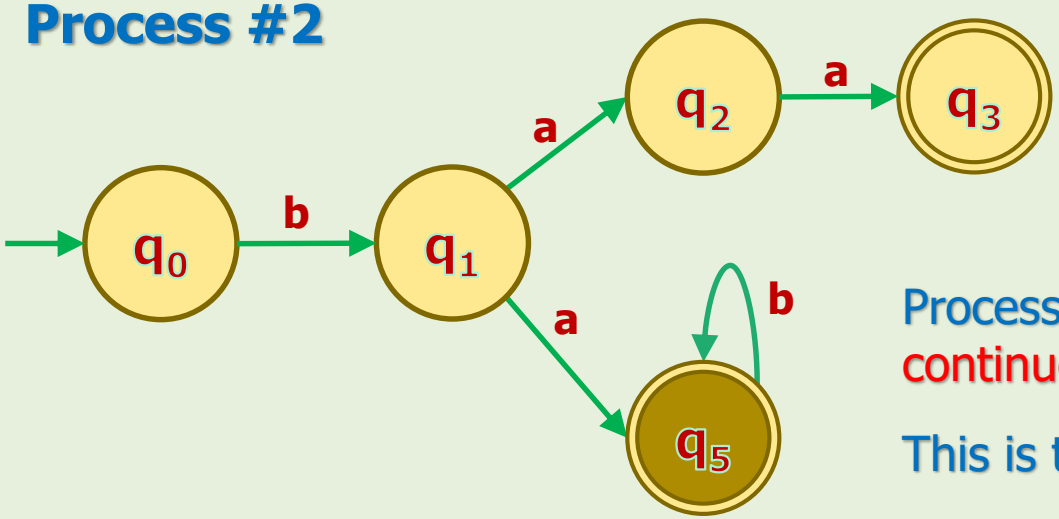


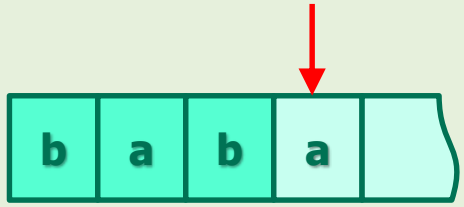
Process #1 has no choice for 'b'.

It halts in a non-accepting state.

So, process #1 rejects w.

Process #2

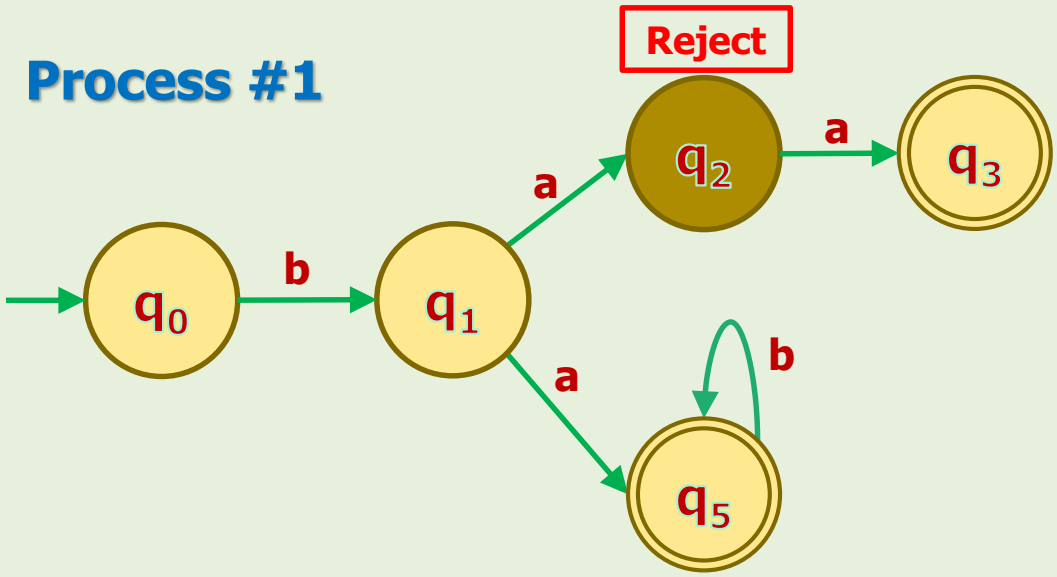


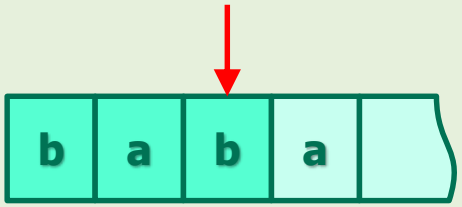


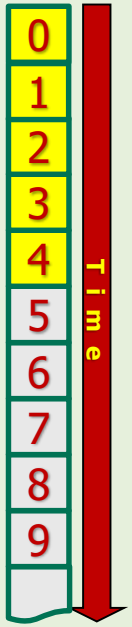
Process #2 consumed 'b' and continues.

This is the end of timeframe 3.

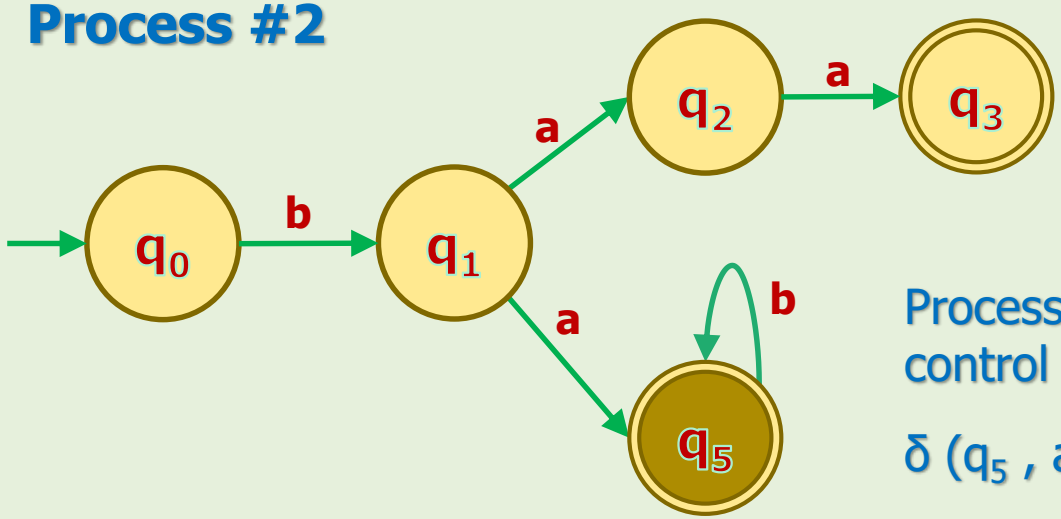
Process #1

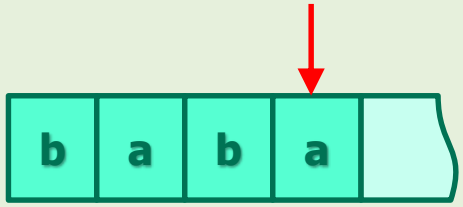






Process #2

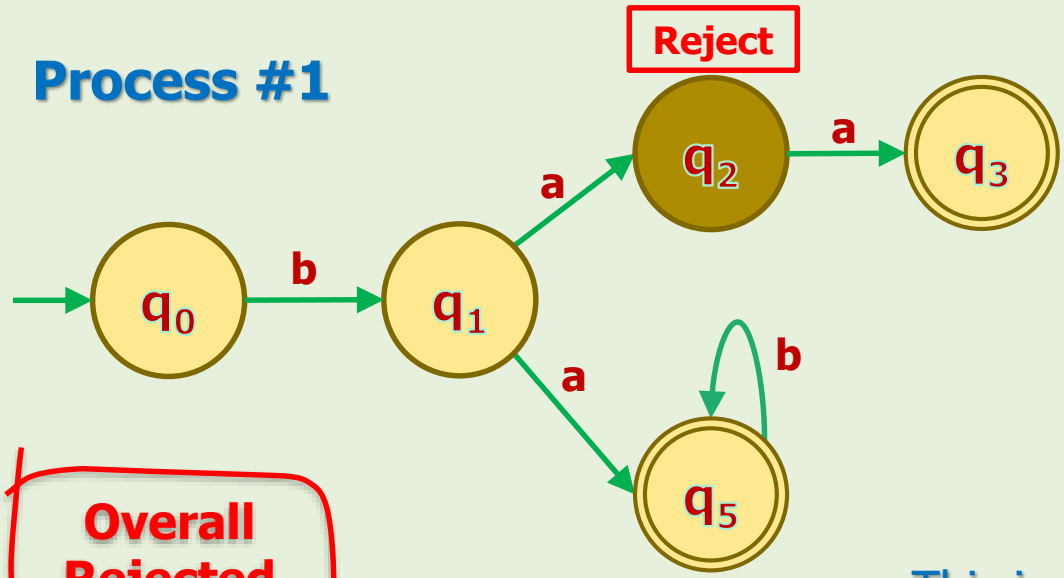


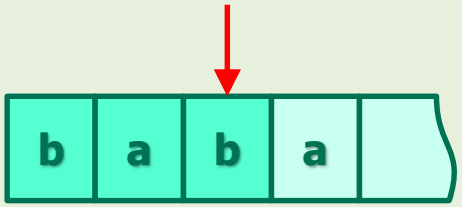


Process #2 reads 'a' and sends it to control unit.

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

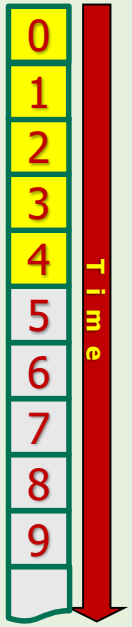
Process #1



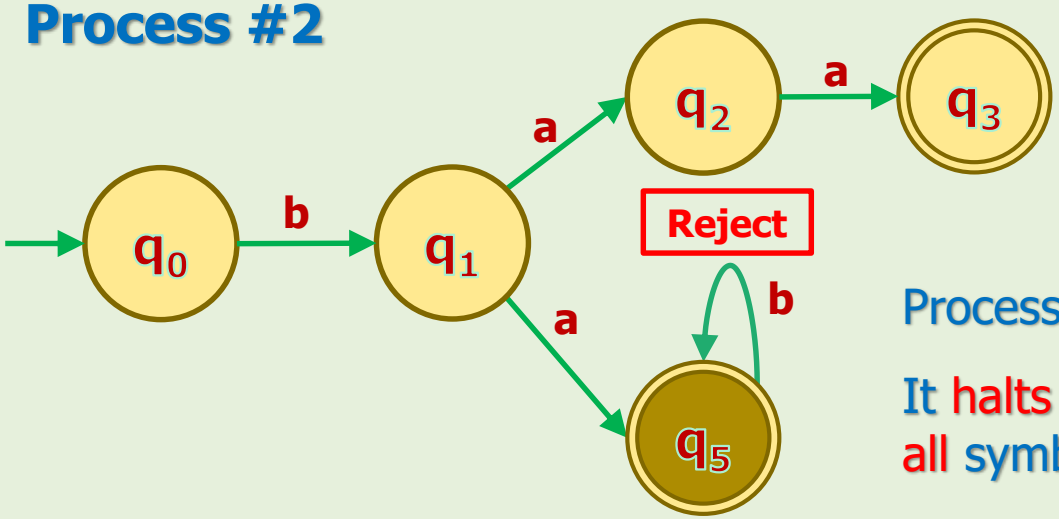


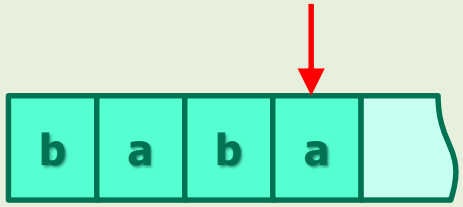
Overall Rejected

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.

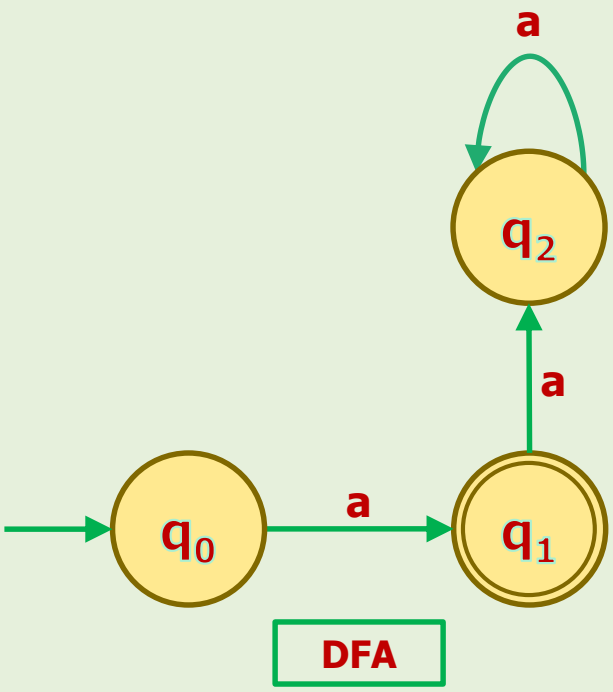
1. Why We Need a New Class

Revisited Question

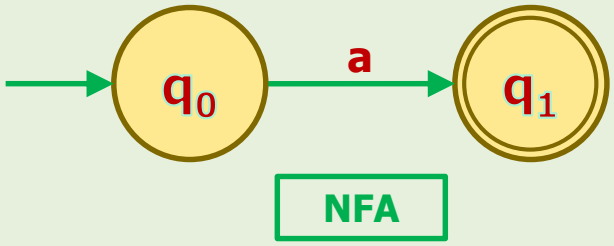
- NFAs are interesting because their transition graphs are simpler.

Example 7

Design a DFA and NFA for L = {a} over Σ = {a}



No trap needed!

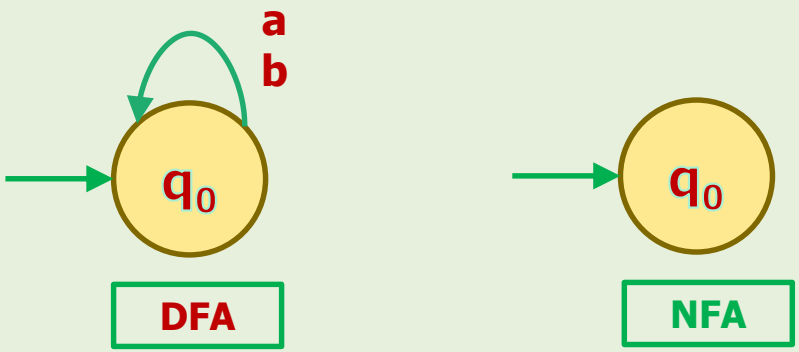


1. Why We Need a New Class

Revisited Question

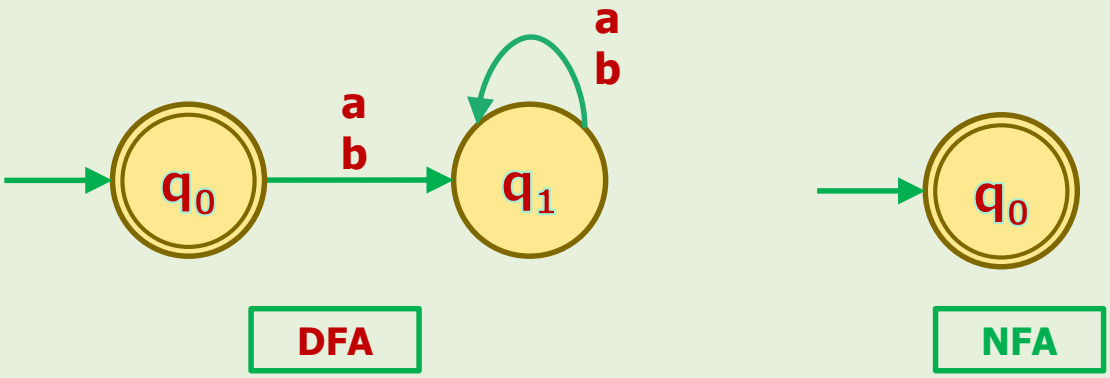
Example 8: Empty Language

L = { } over Σ = {a, b}



Example 9: Empty String Language

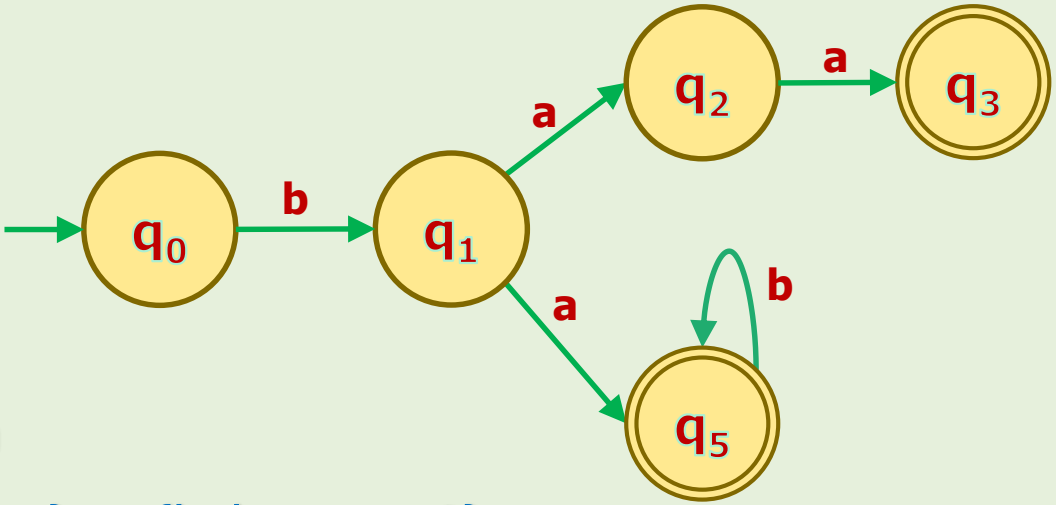
L = {λ} over Σ = {a, b}



Associated Language to NFAs

Example 10

- What is the associated language to the following automaton over Σ = {a , b}?



Solution

$L = \{baa\} \cup \{bab^n : n \geq 0\}$

- Design a DFA to accept L.

A Special Transition

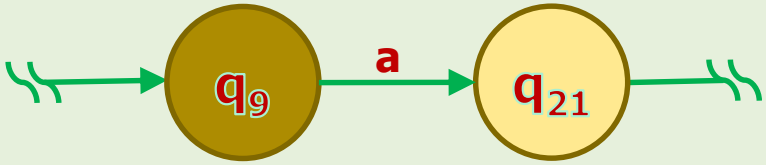
Introduction

- We are going to talk about a special kind of transition.
- Another possible transitions that are strictly prohibited in DFAs ...
- But allowed in NFAs.

Let's Shine our Knowledge

Question

- In the following transition, if the machine is in q_9, what is the "condition" for transition to q_{21}?



Answer

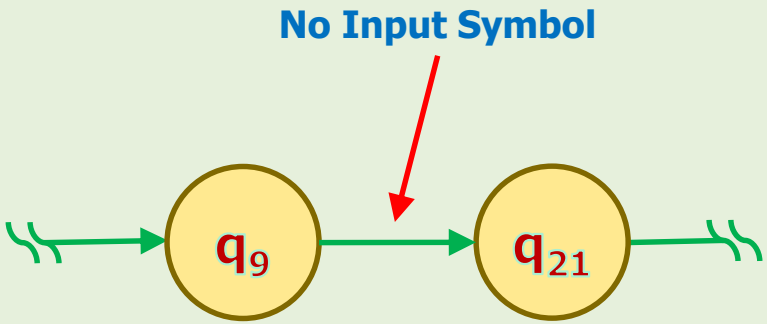
- If the machine is in q_9 AND the next input symbol is 'a', then the machine transits to q_{21}.

Conclusion

- The transition from q_9 to q_{21} is "conditional".

Let's Remove the Condition

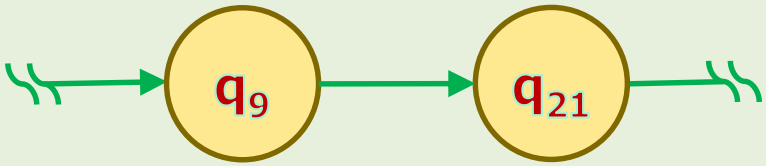
- What would happen if we remove the condition?



- Then we create a "short-circuit".
- A "short-circuit" is an edge with no input symbol.

What is the Meaning of Short-Circuit?

- If there is no symbol, then there is NO condition for transition!

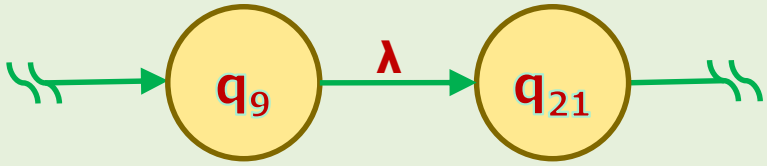


Consequently, the machine can transit unconditionally!

- In other words, if the machine is in q_9, it can unconditionally transit to q_{21}.

The Symbol of Short-Circuit

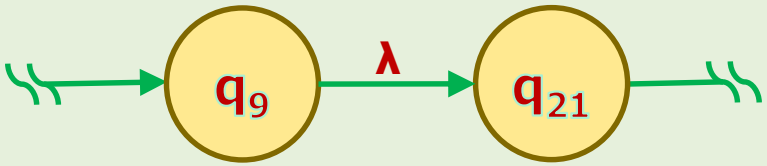
- The symbol "λ" was chosen to represent "short-circuit".



- Because of this symbol, this type of transitions are called "lambda transition" or "λ-transition".

- It has an important role in automata theory.

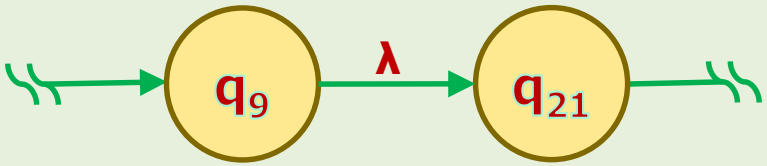
Meaning of λ From Different Angle



- We've already used λ to represent "empty string".
- As we said before, λ means "NO symbol".
  (Empty String = NO symbol, or zero symbol)
- A short-circuit has "no symbol" too.
- That's why the short-circuit is represented by λ.

- Be careful:
  - Using λ as "empty string" and the symbol of "short-circuit" can be confusing but you'll get used to it!
- But what is the meaning and consequence of this?

Input Tape and the Short-Circuit



- What would happen to the input tape?
- Does it need to read any symbol?

- No, it doesn't!

- In fact, the control unit does not need to wait to receive the input symbol for deciding where to transit.

λ-Transition Definition

Definition

- λ-transition in automata theory is a transition that the machine "may unconditionally transit".

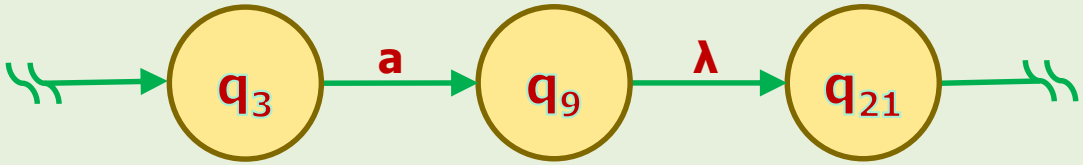
- This is a general definition for all types of automata.

- The concept of λ-transition changes our view about sub-rules of transition function.

- Let's take an example.

How To Represent the Sub-Rule

- What is the value of $\delta(q_3, a) = ?$



- Since the machine may transit unconditionally, it means that ...

  it may stay as well.

- So, when a machine encounters a λ-transition,

  it may stay or it may transit.

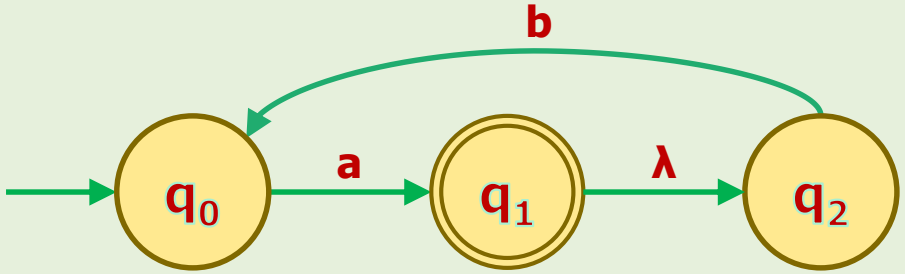
- Therefore, the sub-rule for this example is:

$$\delta(q_3, a) = \{q_9, q_{21}\}$$

Transition Function When λ-Transitions Present

Example 11

- Write the transition function δ of the following transition graph over Σ = {a, b} by using algebraic notation.



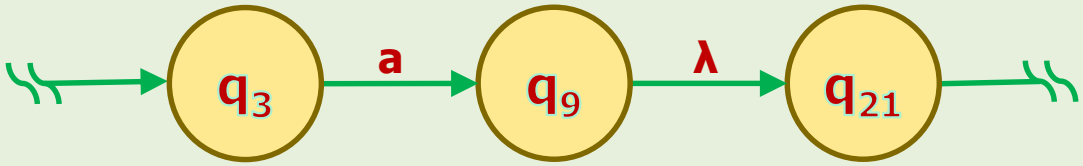
Solution

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

- We can eliminate the empty ranges.

How NFAs Behave If They Have Multiple Choices

- We learned that:



$$\delta(q_3, a) = \{q_9, q_{21}\}$$

- The NFA has multiple choices.

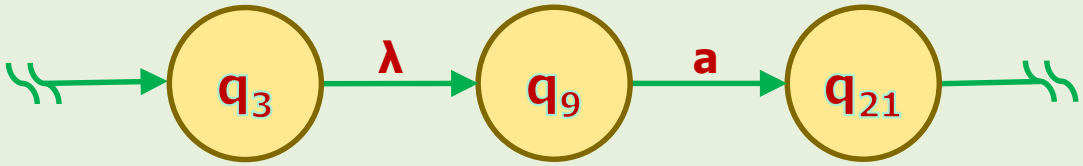
Stay in q_9, OR transit to q_{21}.

- How should it behave when it has multiple choices?
- It would check all possibilities by "parallel processing".

- The procedure of creating new processes is the same way that we learned before.

How NFAs Behave If They Have Multiple Choices

- Another situation:



$$\delta(q_3, a) = \{q_{21}\}$$

- Note that in this situation, the NFA has only one choice.
- Therefore, it does not need to initiate multiple processes.
- As a general rule:

NFAs initiate new processes when they encounter multiple choices.

- Now, let's see some practical examples!

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